JON BONSO, NESTOR MAGYAMA JR., &
NIKEE PEARL TOMAS

AWS CERTIFIED SECURITY SPECIALTY EXAM

SCS-C03

Tutorials Dojo Study Guide

TABLE OF CONTENTS

INTRODUCTION

In the fast-paced IT industry today, there will always be a growing demand for certified IT Professionals that can design highly secure AWS cloud architectures. Companies are spending millions of dollars to optimize the performance of their applications and scale their infrastructure globally to serve customers around the world. They need a reliable and skillful IT staff to build highly available, fault-tolerant, and secure applications that are safe from common web exploits or even large-scale distributed denial-of-service (DDoS) attacks. Most companies have a dedicated IT Security team to improve the infrastructure security of their cloud environment, establish real-time security monitoring and encrypt their data both in transit and at rest.

This Study Guide and Cheat Sheets eBook for AWS Certified Security Specialty aims to equip you with the necessary knowledge and practical skill sets needed to pass the latest version of the AWS Certified Security Specialty exam. This eBook contains the essential concepts, exam domains, exam tips, sample questions, cheat sheets, and other relevant information about the AWS Certified Security Specialty exam. This study guide begins with the presentation of the exam structure, giving you an insight into the question types, exam domains, scoring scheme, and the list of benefits you'll receive once you pass the exam.

We used the official AWS exam guide to structure the contents of this guide, where each section discusses a particular exam domain. Various AWS concepts, related AWS services, and technical implementations are covered to give you an idea of what to expect on the actual exam.

Security Specialty Exam Notes:

Don't forget to read the boxed "**exam tips**" (like this one) scattered throughout the eBook, as these are the key concepts that you will likely encounter on your test. After covering the five domains, we have added a bonus section containing a curated list of AWS Cheat Sheets to fast-track your review. The last part of this guide includes a collection of articles that compares two or more similar AWS services to supplement your knowledge.

The AWS Certified Security Specialty certification exam is a difficult test to pass; therefore, anyone who wants to take it must allocate ample time for review. The exam registration cost is not cheap, which is why we spent considerable time and effort to ensure that this study guide provides you with the essential and relevant knowledge to increase your chances of passing the Security Specialty exam.

******Note:*** *This eBook is meant to be just a supplementary resource when preparing for the exam. We highly recommend working on hands-on sessions, and practice exams to further expand your knowledge and improve your test-taking skills.*

AWS CERTIFIED SECURITY SPECIALTY EXAM OVERVIEW

The AWS Certified Security Specialty exam is intended for IT Professionals who perform a security role in their respective organizations. This exam checks your ability to effectively demonstrate knowledge and your ability to secure your resources in AWS.

Exam Details

The AWS Certified Security Specialty certification is intended for individuals who perform a security role in their organization. This exam doesn't have any prerequisites, but it is recommended that you have at least three years of hands-on experience in implementing security controls to various AWS workloads. This validates your ability to demonstrate your understanding of specialized data classifications, data encryption methods, secure Internet protocols, security operations, and risk mitigation in AWS. It also checks your working knowledge of various AWS security services and features that you can use to implement a secure production environment.

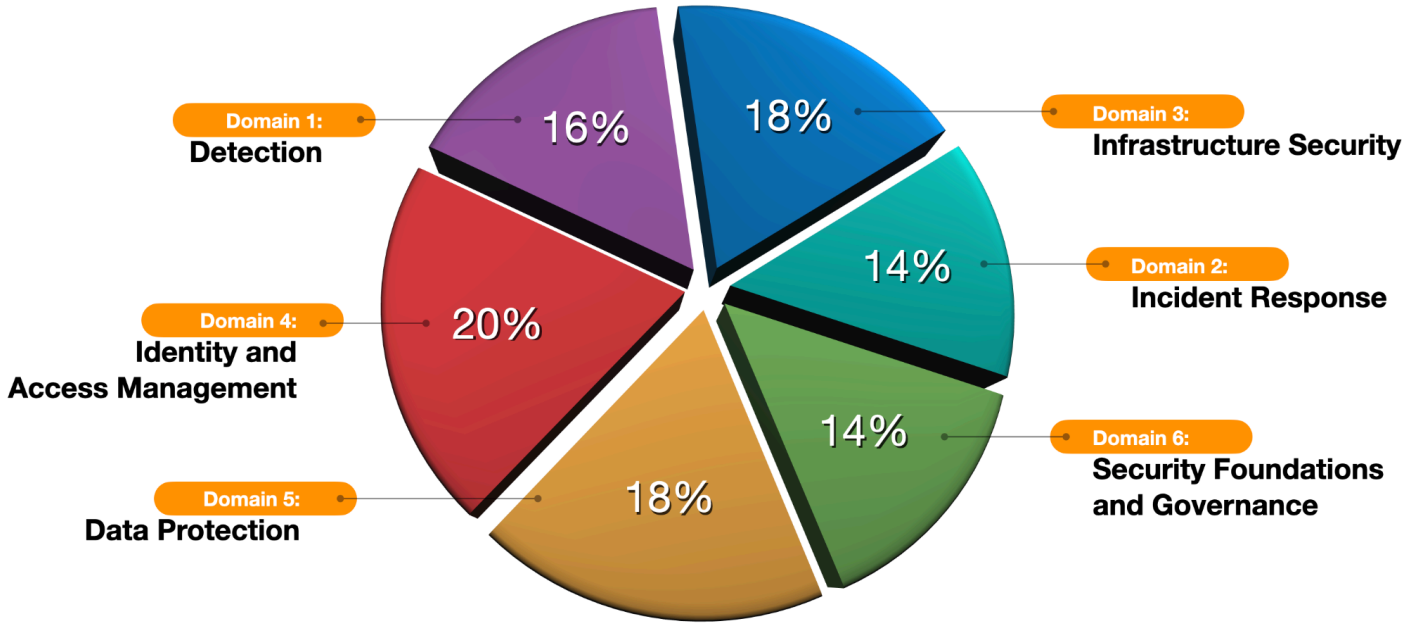
It is composed of scenario-based questions in various formats, including multiple-choice questions, where you select one correct response out of four options, multiple-response questions, where you select two or more correct responses out of five or more options, ordering questions, where you arrange 3–5 responses in the correct order to complete a task, and matching questions, where you match 3–7 prompts with the correct responses. The exam can be taken at a local testing center or online from the comfort of your home.

Exam Code:	SCS-C03
Release Date:	December 2025
Prerequisites:	None
No. of Questions:	65
Score Range:	100 - 1000
Cost:	300 USD
Passing Score:	750/1000
Time Limit:	170 minutes
Delivery Method:	Testing center or online proctored exam.

Don't be confused if you see in your Pearson Vue booking that the duration is 180 minutes since they included an additional 10 minutes for reading the Non-Disclosure Agreement (NDA) at the start of the exam and the survey at the end of it.

Exam Domains

The AWS Certified Security Specialty (SCS-C03) exam has 6 different domains, each with corresponding weight and topic coverage. The exam domains are as follows: Detection (16%), Incident Response (14%), Infrastructure Security (18%), Identity and Access Management (20%), Data Protection (18%) and Security Foundations and Governance (14%):



These are the six exam domains that you should prepare for if you are planning to take the AWS Certified Security Specialty (SCS-C03) test. The list of exam domains can be found in the official SCS-C03 Exam Guide.

Each exam domain has a corresponding weighting, so some sections can have more or fewer questions than others. One exam domain is comprised of several task statements. A task statement is a sub-category of the exam domain that contains the required cloud concepts, knowledge, and skills for you to accomplish a particular task or activity in AWS. Let's look at each of these domains one by one.

Domain 1: Detection

1.1. Design and implement monitoring and alerting solutions for an AWS account or organization.
1.2. Design and implement logging solutions.
1.3. Troubleshoot security monitoring, logging, and alerting solutions.

Domain 2: Incident Response

2.1. Design and test an incident response plan.
2.2. Respond to security events.

Domain 3: Infrastructure Security

3.1. Design, implement, and troubleshoot security controls for network edge services.
3.2. Design, implement, and troubleshoot security controls for compute workloads.
3.3. Design and troubleshoot network security controls.

Domain 4: Identity and Access Management

4.1. Design, implement, and troubleshoot authentication strategies.
4.2. Design, implement, and troubleshoot authorization strategies.

Domain 5: Data Protection

5.1. Design and implement controls for data in transit.
5.2. Design and implement controls for data at rest.
5.3. Design and implement controls to protect confidential data, credentials, secrets, and cryptographic key materials.
5.4. Design and implement controls to protect credentials, secrets, and cryptographic key materials.

Domain 6: Security Foundations and Governance

6.1. Develop a strategy to centrally deploy and manage AWS accounts.
6.2. Implement a secure and consistent deployment strategy for cloud resources.
6.3. Evaluate the compliance of AWS resources.

Exam Scoring System

You can get a score from 100 to 1,000 with a minimum passing score of **750** when you take the Security Specialty exam. AWS uses a scaled scoring model to equate scores across multiple exam types that may have different difficulty levels. The complete score report will be sent to you by email after a few days.

Individuals who unfortunately do not pass the AWS exam must wait 14 days before they are allowed to retake the exam. Fortunately, there is no hard limit on exam attempts until you pass the exam. Take note that on each attempt, the full registration price of the AWS exam must be paid.

Within 5 business days of completing your exam, your AWS Certification Account will have a record of your complete exam results. The score report contains a table of your performance at each section/domain, which indicates whether you met the competency level required for these domains or not. AWS uses a compensatory scoring model, which means that you do not necessarily need to pass each and every individual section, only the overall examination. Each section has a specific score weighting that translates to the number of questions; hence, some sections have more questions than others. The Score Performance table highlights your strengths and weaknesses that you need to improve on.

Exam Benefits

If you successfully pass any AWS exam, you will be eligible for the following benefits:

- **Exam Discount** - You'll get a 50% discount voucher that you can apply for your recertification or any other exam you plan to pursue. To access your discount voucher code, go to the "Benefits" section of your AWS Certification Account, and apply the voucher when you register for your next exam.

- **Certification Digital Badges** - You can showcase your achievements to your colleagues and employers with digital badges on your email signatures, Linkedin profile, or on your social media accounts. You can also show your Digital Badge to gain exclusive access to Certification Lounges at AWS re:Invent, regional Appreciation Receptions, and select AWS Summit events. To view your badges, simply go to the "Digital Badges" section of your AWS Certification Account.

You can visit the official AWS Certification FAQ page to view the frequently asked questions about getting AWS Certified and other information about the AWS Certification: https://aws.amazon.com/certification/faqs/.

AWS CERTIFIED SECURITY SPECIALTY EXAM - STUDY GUIDE AND TIPS

The AWS Specialty certification exams are intended for people who handle more specific responsibilities in AWS Cloud. Since these responsibilities demand a more advanced skill set with prior experience from a person, these AWS specialty exams are built so that they can reinforce and validate a person's eligibility for that role. There are no associate and professional levels in a specialty learning path, so the exams serve as the whole package already. And since they are made that way, expect no less from the specialty certification exams, as they will be as tough as the professional exams.

The name of the certificate immediately points out what to focus on AWS Security. Although we mentioned earlier that specialty exams tackle more specific roles, security in AWS is quite broad and extensive. There are a lot of topics involved when we speak about AWS security, whether it be native AWS services or other third-party tools. If you need comprehensive review material for learning these topics, then this study guide is for you.

The AWS Certified Security – Specialty (SCS-C03) exam is meant for IT professionals who perform a security role such as, but not limited to, Security Engineers, DevSecOps Engineers, Solutions Architects, Cloud Security Analysts, and others. This Security-focused exam validates a person's ability to effectively demonstrate knowledge and skills about securing various types of AWS workloads.

The SCS-C03 version of the exam introduces updated coverage areas to reflect today's security landscape. In addition to traditional AWS security topics, this version also incorporates modern security concerns such as generative AI, machine learning system security, enhanced detection and incident response, and updated best practices that reflect new AWS service capabilities.

The SCS-C03 exam also validates whether a candidate has the following:

- An understanding of specialized data classifications and AWS data protection mechanisms
- An understanding of data-encryption methods and AWS mechanisms to implement them
- An understanding of secure internet protocols and AWS mechanisms to implement them
- A working knowledge of AWS security services and features to provide a secure production environment.
- Competency from 2 or more years of production deployment experience in using AWS security services and features
- The ability to make tradeoff decisions with regard to cost, security, and deployment complexity to meet a set of application requirements
- An understanding of security operations and risks.
- An understanding of emerging threats and modern architectures including generative AI, ML-based applications, and enhanced detection and response workflows.

Study Materials

Having prior knowledge and experience in handling (cloud) security will allow you to understand the concepts and strategies that appear in AWS reference materials. You will also find it easier to comprehend scenario-type questions in your exam. To know more about the AWS Security specialty exam, check out the official AWS Exam Blueprint here.

AWS documentation and whitepapers would be helpful for you to build your security knowledge in the AWS Cloud. Aside from our SCS-C03 AWS Certified Security Specialty Practice Exams, these literary pieces are your primary source of information. We recommend reading the following papers:

1. Introduction to AWS Security
2. AWS: Overview of Security Processes
3. AWS Well-Architected Framework
4. Security Pillar – AWS Well-Architected Framework
5. AWS Security Best Practices
6. AWS Key Management Service Best Practices
7. AWS Key Management Service Cryptographic Details
8. Encrypting File Data with Amazon Elastic File System
9. Secure Content Delivery with Amazon CloudFront
10. Use AWS WAF to Mitigate OWASP's Top 10 Web Application Vulnerabilities
11. Data Residency
12. AWS Best Practices for DDoS Resiliency
13. Application Logging in AWS
14. AWS Security Incident Response Guide
15. Best Practices for Security, Identity, & Compliance

Compliance whitepapers:

1. AWS Security by Design
2. AWS Risk & Compliance
3. Architecting for HIPAA Security and Compliance on AWS
4. Navigating GDPR Compliance on AWS
5. Architecting for PCI DSS Scoping and Segmentation on AWS
6. FedRAMP

AWS Services to Focus On

When we talk about security as a discipline, especially in the context of the cloud, we are tackling it as a combination of different domains. AWS enumerates its catalog of services and features under different domains based on their purposes. In this section, we will try to do the same and group AWS services according to their domains.

Identity and Access Control AWS IAM Identity Center

- AWS Identity and Access Management - You must learn every detail of AWS IAM since this is AWS' primary user management and access control service. Practice writing your own IAM policies.
- Resource-Based Policies - Although resource-based policies fall under AWS IAM, they tend to be ignored compared to user-based policies. Take note of which services support this type of policy and how they are different from user-based policies.
- S3 Presigned URLs - Know what is the purpose of S3 presigned URLs and how they differ from CloudFront signed URLs.
- CloudFront Signed URLs - Know what is the purpose of CloudFront signed URLs and how they differ from S3 presigned URLs or CloudFront signed cookies.
- Amazon Cognito - Read through the benefits of AWS Cognito and how to integrate it with web and mobile applications. Differentiate user pools from identity pools.
- AWS IAM Identity Center - Learn how you can use AWS IAM Identity Center together with other authentication protocols to securely authenticate users in your environment. AWS IAM Identity Center is commonly integrated with LDAP.
- AWS Security Token Service - Know the purpose and use cases of Amazon STS. Try building a program that utilizes temporary tokens as credentials.
- AWS Directory Service - Know the different options you have for AWS Directory Service. Each option solves a different requirement, and it is up to you to figure out how you can get your directory to gain access to your users and other information.
- AWS Organizations - AWS Organizations is a very helpful service when dealing with large-scale enterprises with multiple AWS accounts. Know the benefits of using this service (like consolidated billing feature) and how to build an organization hierarchy with Organization Units and Service Control Policies.
- AWS Resource Access Manager - AWS RAM allows you to securely share resources with other AWS accounts. Experiment with this service to know how to share your resources and what restrictions are involved.

Application and Infrastructure Security

- EC2 key pairs - This goes without saying, but EC2 key pairs play a very important role in protecting your EC2 instances.
- AWS Systems Manager - AWS SSM secures your applications through services like Patch Baselines, Run Command, Session Manager, and more. By utilizing automation and code, you run less risk in human error and unwanted/untracked changes to your application.
- AWS WAF - AWS WAF is essential in protecting your applications from common exploits like SQL injection or XSS attacks. Differentiate WAF from Shield and Firewall Manager.
- AWS Shield - AWS Shield complements AWS WAF since this service offers DDoS protection. Read what features are different between Shield Basic and Shield Advanced.
- AWS Firewall Manager - This service simplifies administration overhead when setting up AWS WAF, AWS Shield, and VPC security groups. Best to do hands-on service.

Data Security

- AWS KMS - Study the different types of KMS keys available and how you should manage them. Determine which AWS services support using AWS KMS for encryption.
- Amazon CloudHSM - Know when to use AWS KMS vs CloudHSM for your encryption needs.
- AWS SSM Parameter Store - It is important to know how AWS SSM Parameter Store can protect your referenceable information through *SecureString*.
- Amazon Secrets Manager - Secrets Manager is similar to Parameter Store, wherein you can store and retrieve sensitive strings in AWS securely.
- SSE-S3 Encryption - Read when it is better to use SSE-S3 keys or KMS keys for server-side encryption. Also, read how your encrypted buckets and objects are handled during operations such as replication, deletion, etc.
- S3 Glacier Vault Lock - Know the purpose of a Glacier Vault Lock and try implementing a policy yourself.
- Amazon Macie - Read how Macie automatically classifies and protects your data. This is one of those services that you will just understand better if you try it out.
- AWS Certificate Manager - Know which services integrate with your certificates stored in Certificate Manager. Try creating your own private CA and issue some custom certificates.

Network Security

- Amazon VPC - Know everything on VPCs since they are basic building blocks for a protected AWS environment. Differentiate security groups vs network ACLs. Study VPC endpoints, too.
- Amazon CloudFront - Study how CloudFront protects your endpoints from being publicly accessible. Read on setting up Origin Access Identity with S3 buckets. Know which services integrate with CloudFront, such as API Gateway and WAF. CloudFront has a feature that allows content access to only selected locations.
- AWS ELB - Study how ELB protects your web traffic and endpoints from malicious attacks. Understand how SSL certificates are being handled by ELB.

- Amazon API Gateway - Similar to ELB, API Gateway also protects your endpoints from being exposed to the public internet. Commonly used in serverless applications, study how APIs can secure Lambda functions. Also, know what services it integrates with, such as WAF.
- AWS VPN - Although AWS VPN is fairly new, you should have an overview of what this service is and how to set it up in your AWS environment.
- AWS Direct Connect - Read how a dedicated line from your network to AWS can protect your inbound and outbound traffic. A common way to secure your traffic in Direct Connect is by using an AWS Site-to-Site VPN.

Logging and Monitoring

- Amazon CloudWatch - Know everything about Cloudwatch (Logs, Alarms, Events, Metrics)
- Amazon CloudTrail - Know everything about CloudTrail, like how to store and encrypt your log files, how to monitor different regions and capture different types of data.
- Service Logs (VPC, ELB, API Gateway, S3, CloudFront) - Multiple AWS services support logging, which they forward to an S3 bucket. It would be good to have an idea of which services support logging. Logs are crucial when conducting incident response and analysis.
- Amazon Route 53 - Study how Route 53 can quickly handle network issues by performing DNS and endpoint health checks. Route 53 also helps in making your environment more resilient by performing automatic failovers.
-

Threat Detection, Prevention, Response and Remediation

- Amazon GuardDuty - Have an understanding of the use cases of Amazon GuardDuty.
- Amazon Inspector - Have an understanding of the use cases of Amazon Inspector.
- Amazon Detective - Know which services integrate with Amazon Detective. Also, have an understanding of the use cases of Amazon Detective.
- AWS Security Hub - Have an understanding of the use cases of AWS Security Hub.

Risk and Compliance Management

- AWS Artifact - Know the purpose of AWS Artifact and what kinds of reports it provides for you.
- AWS Config - AWS Config is an important compliance monitoring tool that you should learn about. Study the concepts and how they work. Practice writing a Config rule of your own to have a better understanding of the service.

Lastly, as we have repeatedly talked about, specialty exams are intended for experienced individuals. Therefore, you should go try out the services above in your own AWS account. Also, do not limit yourself to the Management Console. Some implementations can only be done via AWS CLI, AWS CDK or AWS SDK. Be comfortable with them all.

Common Exam Scenarios

Scenario	Solution
AWS Config	
A company requires a solution that will automatically detect and enable disabled VPC Flow Logs.	Create an AWS Config rule that will detect disabled VPC Flow Logs. Create an Amazon EventBridge rule based on that Config Rule to trigger a Lambda Function for enabling VPC Flow Logs.
Verify if EC2 instances are using approved AMI. Create a notification if non-compliant instances are detected.	Utilize the approved-amis-by-id managed rule in AWS Config to check if running instances are using an approved AMI. Use CloudWatch Alarms for notifications.
A Security Analyst needs to remediate the risks of having security groups that allow inbound traffic for the 0.0.0.0/0 CIDR range (Anywhere). The security group must only allow inbound traffic for the company's firewall IP address.	Create an AWS Config rule that will automatically detect security groups that allow inbound traffic from the 0.0.0.0/0 CIDR range. Associate a Lambda function in the Config rule to update the security group's inbound rule with the company's firewall IP address.
You need to build a solution that will allow the Security team to review the IAM policy assigned to an IAM user before and after a security incident has occurred.	Use AWS Config
Automatically detect and remediate an incident where API logging is disabled.	Create an AWS Config rule to detect disabled CloudTrail settings. Configure the rule to use an AWS Systems Manager Automation document to automatically re-enable CloudTrail logs.
Detect if someone is using the AWS account's root access to create new API keys without proper approval.	Set up an AWS Config rule to track the usage of the create-api-key command by the root IAM user.
AWS KMS	
A company requires a KMS key that automatically rotates every year.	Create a KMS key with AWS-generated key material.

A company needs to rotate a KMS key with imported key material.	Create a new KMS key with the new imported key material and point the existing alias to the new KMS key.
A company has to manage the access control for hundreds of KMS keys without having to edit key policies.	Use grants in AWS KMS.
A Security Specialist must use additional authenticated data (AAD) to prevent tampering against the ciphertext.	Add the kms:EncryptionContext condition when defining the key policy for the KMS key.
A company needs to migrate AWS resources encrypted with KMS into another region.	Use a new KMS key in the target region.
AWS WAF, AWS Shield	
An application hosted on an EC2 instance needs protection from common web exploits. Also, the outgoing traffic from the instance should be restricted only to trusted URLs.	Use AWS WAF for common web exploits protection and use a third-party solution to whitelist URLs for outbound traffic.
A Security Specialist needs to block high-volume requests from specific user-agent HTTP headers.	Use AWS WAF rate-based rule to limit the number of requests.
Which AWS Services has direct integration with AWS WAF?	Amazon CloudFront & Application Load Balancer
A company is serving static content using Amazon CloudFront, Amazon S3, and Amazon Route53. They must respond to DDoS attacks at L7, L4, and L3.	Use AWS Shield Advanced
AWS CloudTrail	
Protect CloudTrail Logs from tampering and unauthorized access	Enable the CloudTrail log file validation

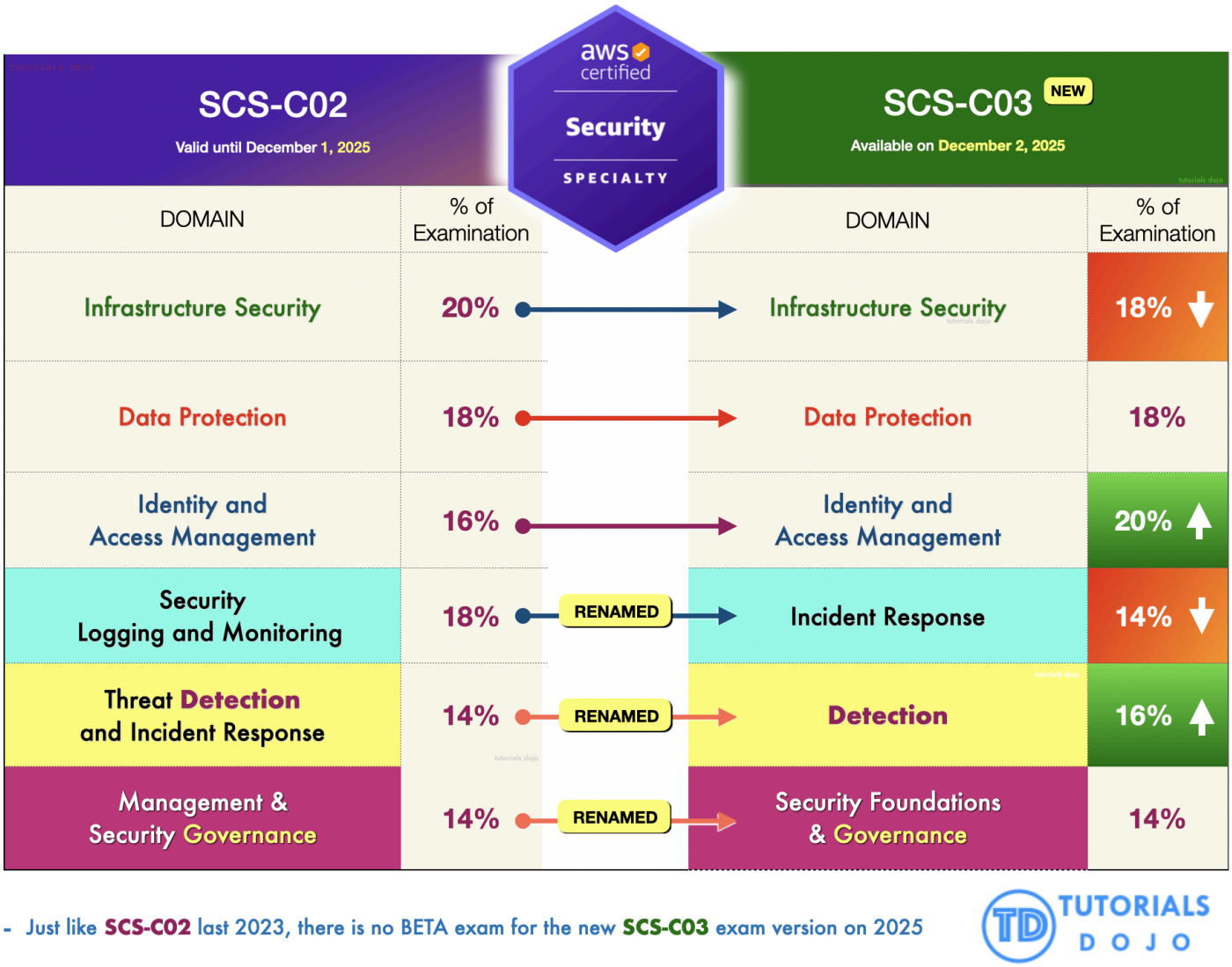
Some AWS accounts can't send CloudTrail logs in a centralized logging account. What are the steps to troubleshoot the issue?	1. Check if the AWS Account IDs are included within the Central account's S3 bucket policy. 2. Check if the AWS Accounts are using the correct S3 bucket name for centralized logging. 3. Check if all trails are active
A Security Specialist has updated the log file prefix for a trail but encountered a "There is a problem with the bucket policy." error.	First, update the new log file prefix in the S3 bucket policy, then specify the updated log file prefix in the CloudTrail Console.
A Security Engineer needs to review user activities from a specific access key within the past 3 months.	Review the user activities through the CloudTrail Console.
Amazon CloudWatch	
Some EC2 instances stop sending CloudWatch logs after a security incident. What are the steps to troubleshoot this issue?	1. Check if CloudWatch Logs agent is active and running in the EC2 instances. 2. Check if the EC2 instances have Internet access. 3. Check the validity of the OS Log rotation rules.
After an update to IAM policy, an application stops sending custom metrics to AWS CloudWatch.	Add the cloudwatch:putMetricData permission in the IAM policy.
A Security Engineer must build a near-real-time logging solution to collect logs from different AWS Accounts.	Use the Amazon CloudWatch cross-account log data sharing with subscriptions. Use Amazon Amazon Data Firehose to deliver the logs.
A company has set up a notification system using CloudWatch and CloudTrail that will alert a	Make sure that the value of the consecutive periods' alarm threshold is equal to or greater than 1.

Security Team when new access keys are created. The team is not receiving notifications.	
Amazon GuardDuty	
A company needs a threat detection system for monitoring malicious activities in an AWS Account.	Use Amazon GuardDuty
A company is using an Active Directory server to resolve DNS for EC2 instances in a VPC. A security engineer noticed that one of the instances is being used for command-and-control (C2C) operations, but GuardDuty has failed to recognize it.	GuardDuty does not recognize DNS requests coming from third-party DNS servers.
A company wants to perform a network port scan against EC2 instances in VPC but does not want to get alerts for specific instances.	Add the EIP of the specific instances to the trusted IP lists in Amazon GuardDuty.
Infrastructure Security	
A company has complex connectivity rules for Amazon EC2 instances. How should they manage these connection rules with no additional cost?	Implement the rules using the built-in host-based firewall such as iptables
A Security Engineer needs to inspect packet data.	1. Use a proxy software hosted on an EC2 instance. 2. Use a host-based agent on an EC2 instance. *Note that you can only perform packet data analysis with third-party solutions.*
A Security Engineer has a virtual security appliance. The Engineer is using a security group and NACL to comply with security requirements. How can he allow traffic through the virtual security appliance?	Disable the Source/Destination check of the Elastic Network Interface (ENI) associated with the virtual security appliance.

A Security Engineer needs to remediate the risk of users exploiting the instance metadata service to access AWS resources in other accounts.	Restrict the access to the instance metadata service using iptables.
The DevSecOps team needs help to securely create or connect the company's workforce identities and manage their access centrally across AWS accounts and applications. What is the is the recommended approach to use for workforce authentication and authorization on AWS?	Use IAM Identity Center
The AWS Security Hub service consumes, aggregates, organizes, and prioritizes findings from AWS security services and from the third-party product integrations. What is the standard findings format used by the Security Hub which eliminates the need for time-consuming data conversion efforts?	AWS Security Finding Format (ASFF)
A company has several AWS accounts that are connected using AWS Organizations. There's a requirement to centrally manage the security services and aggregating findings from all the accounts. What should the Security Engineer do?	Use delegated administration via AWS Config aggregators and AWS Security Hub.
A multinational corporation needs to continuously audit their AWS usage to simplify how they assess risk and compliance with regulations and industry standards. What service should be used to automate the process of evidence collection to make it easier to assess if the corporation's policies, procedures, and activities, are operating effectively?	AWS Audit Manager

The Old SCS-C02 vs the New SCS-C03 Exam Version

The latest AWS Certified Security Specialty exam SCS-C03 has undergone a few changes. These changes primarily involve domain renaming and adjustments to domain weightings. One notable change is the renaming of three domains to better reflect their focus: "Threat Detection and Incident Response" is now called "Detection"; "Security Logging and Monitoring" is now called "Incident Response"; and "Management and Security Governance" is now called "Security Foundations and Governance."



- Just like **SCS-C02** last 2023, there is no BETA exam for the new **SCS-C03** exam version on 2025

Some existing exam domains have had changes in their weightings. Some weights have decreased, others have increased, while two remain unchanged, as shown below:

- "Detection" domain increased from 14% to 16%
- "Incident Response " domain went down from 18% to 14%
- "Infrastructure Security" domain was slightly cut down from 20% to only 18%
- "Identity and Access Management" domain increased from 16% to 20%
- "Data Protection" remained 18%
- “Security Foundations and Governance” remained 14%

Validate Your Knowledge

We recommend that you try the Exam Readiness: AWS Certified Security - Specialty Course that you can get for free the AWS Skill Builder page. They provide sample questions that you can follow along and answer.

AWS also provides a sample exam on the AWS Certified Security Specialty page, which you can find here. Although this sample exam is not on the same level of difficulty one might expect on the real exam, it is still a helpful resource for your reviews.

Lastly, **Tutorials Dojo** also has a set of high-quality **SCS-C03 practice exams**, and this study guide eBook for the AWS Security Specialty certification. The practice exams and study guide eBook will help boost your preparedness for the real exam, and it will also help you determine which areas you are weak in, so you can focus your efforts on studying those areas.

Sample Practice Test Questions:

Question 1

An organization is implementing a security policy in which their cloud-based users must be contained in a separate authentication domain and prevented from accessing on-premises systems. Their IT Operations team is launching and maintaining a number of Amazon RDS for SQL Server databases and EC2 instances. The organization also has an on-premises Active Directory service that contains the administrator accounts that must have access to the databases and EC2 instances.

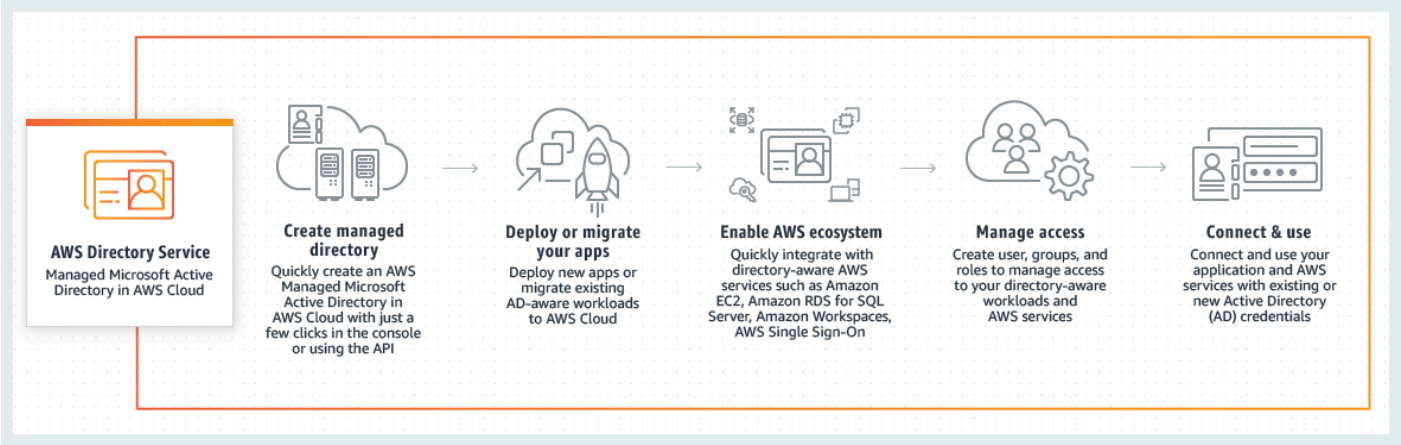
How would the Security Engineer manage the AWS resources of the organization in the MOST secure manner? (Select TWO.)

1. Using AWS Directory Service, set up an AWS Managed Microsoft AD to manage the RDS databases and EC2 instances.
2. Set up and configure AWS Service Catalog to manage the RDS databases and EC2 instances.
3. Set up a one-way incoming trust relationship from the existing Active Directory in the on-premises data center to the new Active Directory service in AWS.
4. Set up a one-way incoming trust relationship from the new Active Directory in AWS to the existing Active Directory service in the on-premises data center.
5. Set up a two-way trust relationship between the new Active Directory in AWS and the existing Active Directory service in the on-premises data center.

Correct Answer: 1,3

In **Active Directory**, trust relationships enable access to various resources that can be either one-way or two-way. A one-way trust is a unidirectional authentication path created between two domains. In a one-way trust between Domain A and Domain B, users in Domain A can access resources in Domain B. However, users in Domain B can't access resources in Domain A. Some one-way trusts can be either non-transitive or transitive, depending on the type of trust being created.

You can configure one and two-way external and forest trust relationships between your AWS Directory Service for Microsoft Active Directory and on-premises directories, as well as between multiple AWS Managed Microsoft AD directories in the AWS cloud. AWS Managed Microsoft AD supports all three trust relationship directions: Incoming, Outgoing, and Two-way (Bi-directional). When setting up trust relationships, you must ensure that your on-premises directory is and remains compatible with AWS Directory Services.



If you already have an AD infrastructure and want to use it when migrating AD-aware workloads to the AWS Cloud, AWS Managed Microsoft AD can help. You can use AD trusts to connect AWS Managed Microsoft AD to your existing AD. This means your users can access AD-aware and AWS applications with their on-premises AD credentials without needing you to synchronize users, groups, or passwords.

For example, your users can sign in to the AWS Management Console and Amazon WorkSpaces by using their existing AD user names and passwords. Also, when you use AD-aware applications such as SharePoint with AWS Managed Microsoft AD, your logged-in Windows users can access these applications without needing to enter credentials again.

There are three trust relationship directions:

1. **One-way:incoming** - Users in the specified realm will not be able to access any resources in this domain.
2. **One-way:outgoing** - Users in this domain will not be able to access any resources in the specified realm.
3. **Two-way (Bi-directional)** - Users in this domain and users in the specified realm will be able to access resources in ***either*** domain or realm.

Hence, the correct answers are:

- **Using AWS Directory Service, set up an AWS Managed Microsoft AD to manage the RDS databases and EC2 instances.**
- **Set up a one-way incoming trust relationship from the existing Active Directory in the on-premises data center to the new Active Directory service in AWS.**

The option that says: **Set up and configure AWS Service Catalog to manage the RDS databases and EC2 instances** is incorrect because AWS Service Catalog simply allows organizations to create and manage catalogs of IT services that are approved for use on AWS. You have to use AWS Directory Service instead.

The option that says: **Set up a one-way incoming trust relationship from the new Active Directory in AWS to the existing Active Directory service in the on-premises data center** is incorrect because this should be the other way around. Instead, you have to set up a one-way trust relationship from the existing Active Directory in the on-premises data center to the new Active Directory service in AWS.

The option that says: **Set up a two-way trust relationship between the new Active Directory in AWS and the existing Active Directory service in the on-premises data center** is incorrect because the scenario explicitly mentioned that the cloud-based users must be prevented from accessing on-premises systems. Hence, you have to use a one-way trust relationship only.

References:
https://docs.aws.amazon.com/directoryservice/latest/admin-guide/ms_ad_setup_trust.html
https://docs.aws.amazon.com/directoryservice/latest/admin-guide/usecase5.html
https://docs.microsoft.com/en-us/azure/active-directory-domain-services/concepts-forest-trust

Check out this AWS Directory Service Cheat Sheet:
https://tutorialsdojo.com/aws-directory-service/

Question 2

A company is planning to migrate its on-premises application to AWS. The application will be hosted in Elastic Beanstalk, which uses an external RDS database and an S3 bucket configured to use Server-Side Encryption with Customer-Provided Encryption Keys (SSE-C). In this configuration, Amazon S3 does not store the encryption key you provide but instead stores a randomly salted hash-based message authentication code (HMAC) value of the encryption key in order to validate future requests. The Security Engineer was assigned to implement the required security measures for the application.

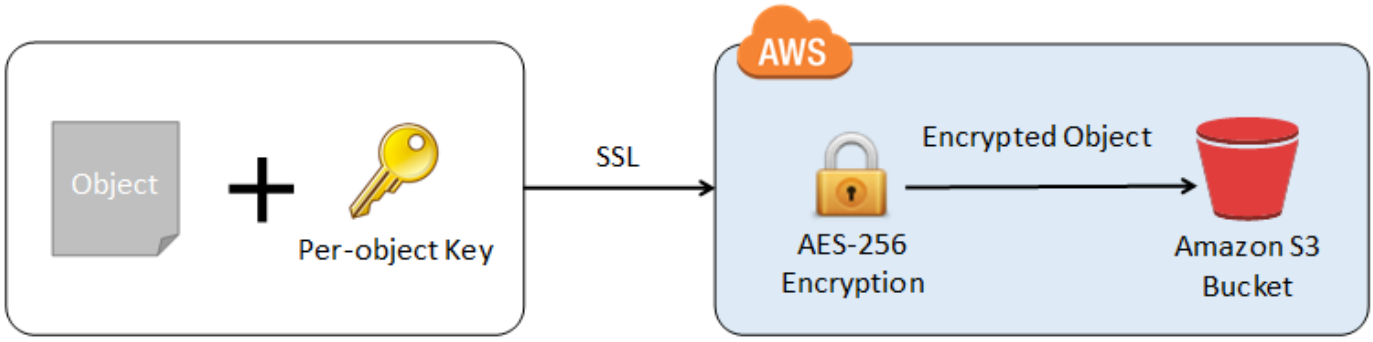
Which of the following is a valid consideration that the Engineer should keep in mind when implementing this architecture?

1. The salted HMAC value can be used to derive the value of the encryption key.
2. You will lose access to the S3 object if you lose the encryption key.
3. The salted HMAC value can be used to decrypt the contents of the encrypted object.
4. The salted HMAC value can be used to decrypt the S3 object in the event that you lose the encryption key.

Correct Answer: 2

Server-side encryption is about protecting data at rest. Using server-side encryption with customer-provided encryption keys (SSE-C) allows you to set your own encryption keys. With the encryption key you provide as part of your request, Amazon S3 manages both the encryption, as it writes to disks, and decryption, when you

access your objects. Therefore, you don't need to maintain any code to perform data encryption and decryption. The only thing you do is manage the encryption keys you provide.



When you upload an object, Amazon S3 uses the encryption key you provide to apply AES-256 encryption to your data and removes the encryption key from memory. It is important to note that Amazon S3 does not store the encryption key you provide. Instead, it is stored in a randomly salted HMAC value of the encryption key in order to validate future requests. The salted HMAC value cannot be used to derive the value of the encryption key or to decrypt the contents of the encrypted object. That means if you lose the encryption key, you lose the object.

When you retrieve an object, you must provide the same encryption key as part of your request. Amazon S3 first verifies that the encryption key you provided matches and then decrypts the object before returning the object data to you.

Hence, the valid consideration that the developer should keep in mind when implementing this architecture is: **You will lose access to the S3 object if you lose the encryption key**.

The option that says: **The salted HMAC value can be used to derive the value of the encryption key** is incorrect because the salted HMAC is just used to validate future encryption requests. It cannot be used to derive the value of the encryption key or to decrypt the contents of the encrypted object.

The option that says: **The salted HMAC value can be used to decrypt the contents of the encrypted object** is incorrect because, just as mentioned above, the HMAC cannot be used to derive the value of the encryption key or to decrypt the contents of the encrypted object.

The option that says: **The salted HMAC value can be used to decrypt the S3 object in the event that you lose the encryption key** is incorrect because if you lose the encryption key, you will also lose access to that object. You cannot use the salted HMAC value to decrypt the object.

References:
https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingKMSEncryption.html
https://docs.aws.amazon.com/AmazonS3/latest/API/RESTObjectPUT.html#RESTObjectPUT-responses-exam

ples
https://aws.amazon.com/blogs/security/how-to-prevent-uploads-of-unencrypted-objects-to-amazon-s3/

Check out these Amazon S3 and AWS KMS Cheat Sheets:
https://tutorialsdojo.com/amazon-s3/
https://tutorialsdojo.com/aws-key-management-service-aws-kms/

Click **here** for more **AWS Certified Security Specialty practice exam questions**.

Check out our other AWS practice test courses **here**:

With the growing number of security attacks each day, companies are now focusing their efforts in strengthening their digital security. This responsibility requires a team effort from both AWS engineers and industry professionals, which is why we have a shared responsibility model. Professionals will have to be equipped with the right tools and knowledge to protect what is valuable to them and to their company.

We hope that our guide has helped you achieve that goal, and we would love to hear back from you after your exam. Get some well-deserved rest, and we wish you the best of results.

Domain 1: Detection

Overview

The first domain of the AWS Certified Security Specialty (SCS-C03) exam focuses on your ability to design, implement, and troubleshoot detection mechanisms across your AWS infrastructure. This includes monitoring, alerting, and logging solutions that enable you to identify security events and anomalies. 16% of questions in the Security Specialty exam revolve around this topic.

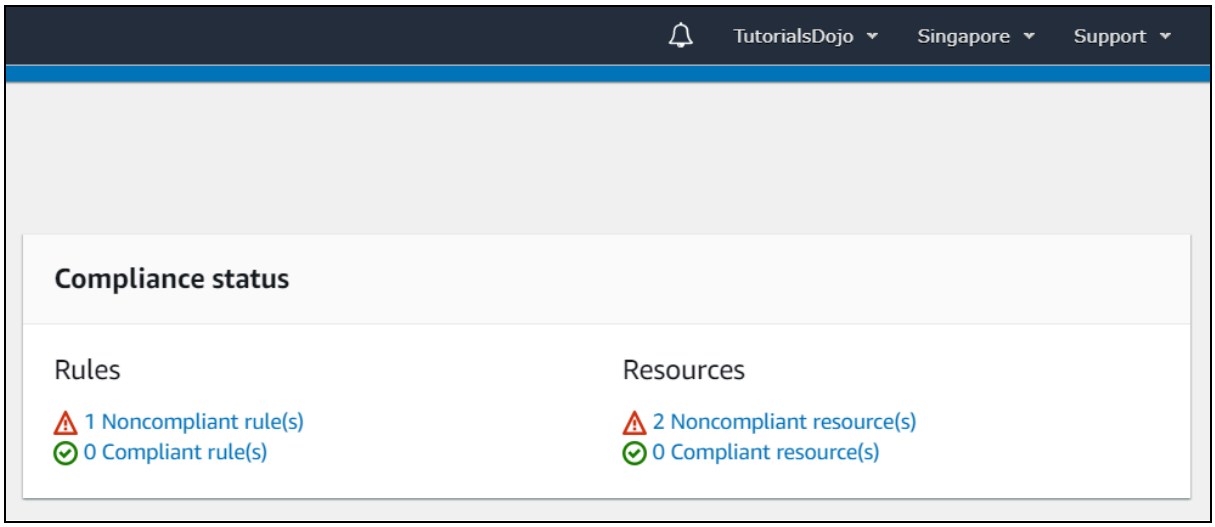
This domain will challenge your know-how in doing the following:

- Analyzing workloads to determine monitoring requirements
- Designing and implementing workload monitoring strategies (for example, configuring resource health checks)
- Aggregating security and monitoring events
- Creating metrics, alerts, and dashboards to detect anomalous data and events (for example, Amazon GuardDuty, Amazon Security Lake, AWS Security Hub, Amazon Macie)
- Creating and managing automations to perform regular assessments and investigations (for example, deploying AWS Config conformance packs, Security Hub, AWS Systems Manager State Manager)
- Identifying sources for log ingestion and storage based on requirements
- Configuring logging for AWS services and applications (for example, configuring an AWS CloudTrail trail for an organization, creating a dedicated Amazon CloudWatch logging account, configuring the Amazon CloudWatch Logs agent)
- Implementing log storage and log data lakes (for example, Security Lake) and integrating with third-party security tools
- Using AWS services to analyze logs (for example, CloudWatch Logs Insights, Amazon Athena, Security Hub findings)
- Using AWS services to normalize, parse, and correlate logs (for example, Amazon OpenSearch Service, AWS Lambda, Amazon Managed Grafana)
- Determining and configuring appropriate log sources based on network design, threats, and attacks (for example, VPC Flow Logs, transit gateway flow logs, Amazon Route 53 Resolver logs)
- Analyzing the functionality, permissions, and configuration of resources (for example, Lambda function logging, Amazon API Gateway logging, health checks, Amazon CloudFront logging)
- Remediating misconfiguration of resources (for example, troubleshooting CloudWatch Agent configurations, troubleshooting missing logs)

In this chapter, we will cover all of the related topics for Detection in AWS that will likely show up in your Security Specialty exam. Take note that this domain accounts for 16% of scored content, making it the third-highest weighted domain alongside Data Protection. The highest weighted domain is Identity and Access Management at 20%, while the lowest weighted domains are Incident Response and Security Foundations and Governance, both at 14%.

Using AWS Config Rules for Automated Checks and Remediation

AWS Config tracks and records configuration changes on your AWS resources. You can use this service to ensure your resources are properly configured and that they stay within the compliance requirements of your organization. AWS Config uses Config rules to evaluate whether a particular resource is compliant or not. You can view the compliance status of these evaluations on your AWS Config Dashboard.



AWS Config saves records of changes to an S3 bucket of your choosing or one that Config creates automatically. Optionally, you can set an SNS Topic to get alerts via email or SMS every time a change occurs.

There are two types of Config rules which you can check your compliance against:

- **AWS-Managed Rules** - These are rules that are defined and maintained by AWS. They require minimal to no configuration. You can use this to evaluate common security concerns such as:
    - *Do all of my security groups in use allow unrestricted incoming SSH traffic?*
    - *Is the account password policy for IAM users secure enough to meet the specified requirements?*
    - *Are all EBS volumes that are in an attached state encrypted?*

- **Custom Rules** - AWS-Managed rules are great, but it does not solve every problem as every business is built differently. If you want to check compliance against your company's internal security rules that are not readily defined in AWS, you can use Custom Rules. Custom rules are AWS Lambda functions that you create and maintain yourself. The Lambda function should contain the logic that will determine whether a resource is compliant or non-compliant.

AWS Config can take remediation actions against non-compliant resources using AWS Systems Manager Automation documents. You can use one of the prebuilt automation documents provided by AWS Config or write your own. An SSM automation document can be associated with a Config rule to take corrective actions automatically when a resource is evaluated as non-compliant. However, you may run the SSM automation document manually as well.

In other cases, it's also possible to create an EventBridge (CloudWatch Event) rule that triggers a Lambda function in response to a particular AWS Config event. Consider the following scenario:

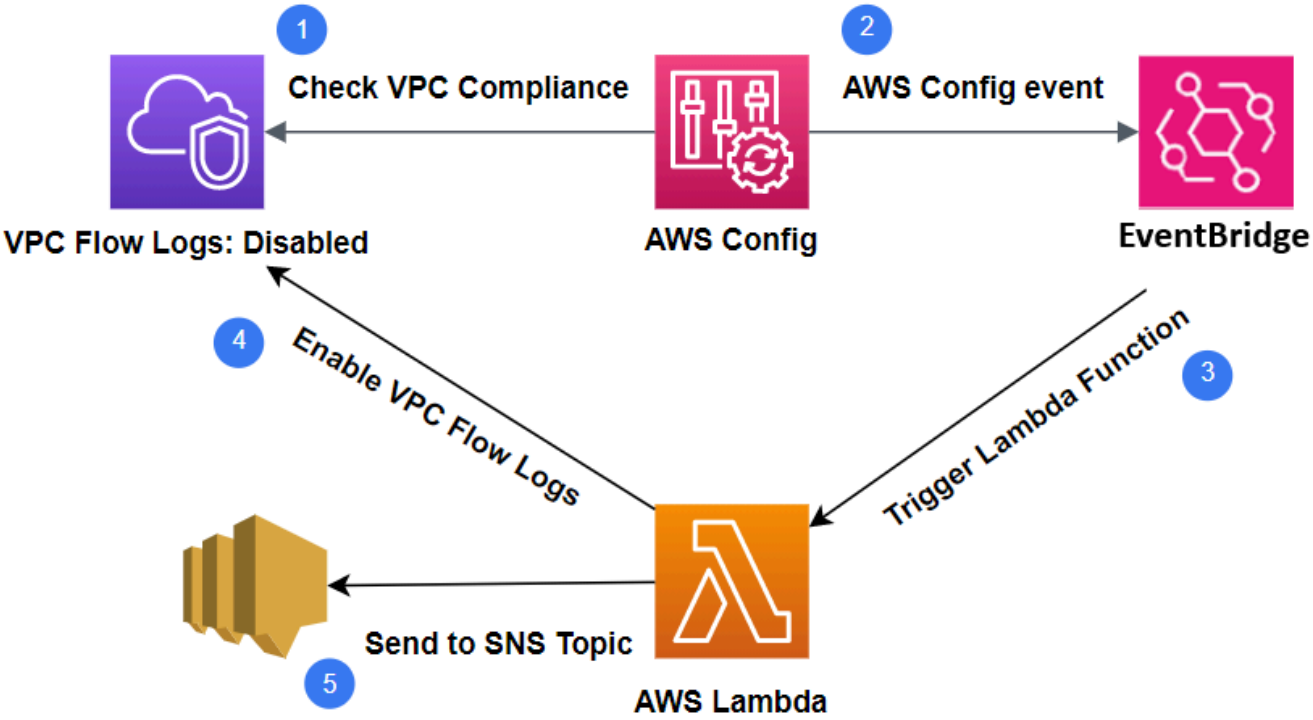
> *You are managing your development account where several teams are using it to test their applications. All VPCs created on the account should have VPC Flow Logs enabled, and the logs must be sent to a central S3 bucket for audit purposes. Since several teams created their own VPCs for testing, it is difficult for you to track all VPCs and manually enable VPC Flow Logs. The solution should easily detect non-compliant resources and automatically enable the VPC Flow Logs.*

The scenario addresses two problems: The first issue is about automatically tracking non-compliant VPCs (VPCs with disabled VPC Flow Logs). The second problem is about implementing the remedial action to take when Config detects a non-compliant VPC.

We can solve the problem by doing the following steps:

1. Create an AWS Config rule that will flag VPCs with disabled VPC Flow Logs.
2. Create a Lambda function that will enable the VPC Flow Logs on the non-compliant VPC.
3. Create an EventBridge(CloudWatch Event) event rule that triggers the Lambda function when AWS Config detects a non-compliant VPC.

You can optionally add an SNS topic to receive notifications for only that specific event. Below is the architecture diagram for the solution:

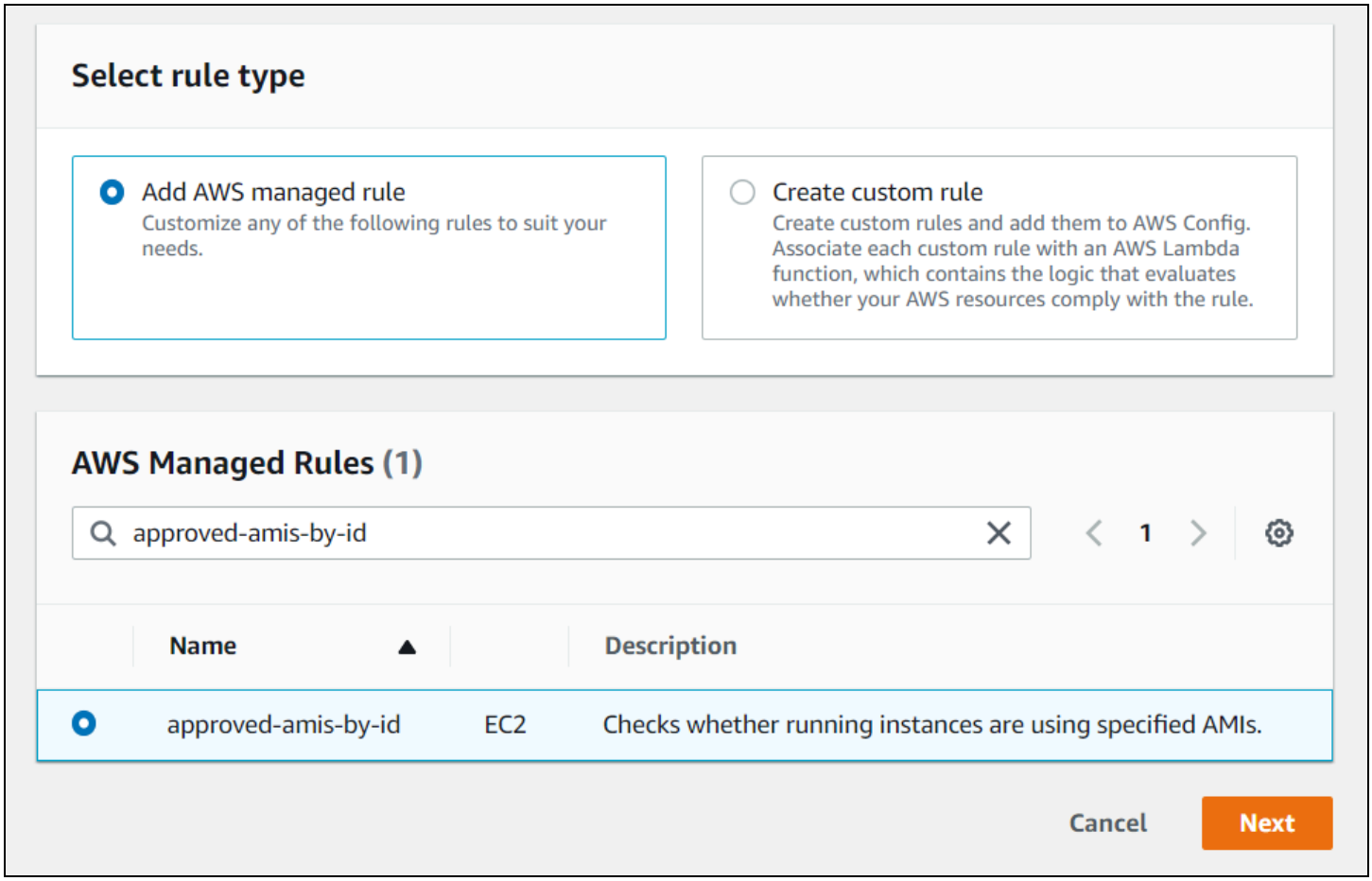


Using AWS Config to ensure that only approved AMIs are only launched

There are cases when you want to ensure that only the approved AMIs are used to launch EC2 instances for compliance and security reasons. Obviously, this could easily be done if you're involved in a small organization that manages two or more instances. You can just manually go through each of the instances and check their AMI IDs.

However, if you are managing hundreds of EC2 instances, that's a different situation. Let's say that you want to host a web application using a specific Linux distribution only. Manually inspecting individual instances would take you hours and is not very practical, especially now that we have access to modern tools that make our lives easier.

The better solution is to use the **`approve-amis-by-id`** managed rule in AWS Config. This is already pre-defined for you, so you don't have to create and maintain your own code to evaluate AMIs. This rule will detect any non-compliant AMIs, and you could automate the remediation action by following the same architecture that we previously discussed using Amazon EventBridge (Amazon CloudWatch Events).



AWS Config conformance packs

These are collections of Config rules and remediation actions that can be deployed as a single entity across accounts, regions, and organizations. These packs are created using YAML templates containing managed or custom rules with optional remediation actions, and can be stored in AWS Systems Manager documents for streamlined deployment. Organizations can leverage sample templates for quick starts covering operational best practices, security standards, and compliance frameworks, or create custom templates tailored to their specific governance requirements.

The conformance pack dashboard provides visibility into compliance scores, which represent the percentage of compliant rule-resource combinations, enabling teams to track remediation progress and assess the impact of changes on their compliance posture. For multi-account environments, conformance packs can be deployed across AWS Organizations using a management account or delegated administrator, automatically creating necessary Config rules and remediation actions without requiring additional permissions in member accounts.

Security Specialty Exam Notes:

You can use the AWS Config to monitor and assess non-compliant configurations in your resources. Send the event's status to Amazon EventBridge (Amazon CloudWatch Events) so it can trigger a Lambda Function that will fix the non-compliant configuration.

References:

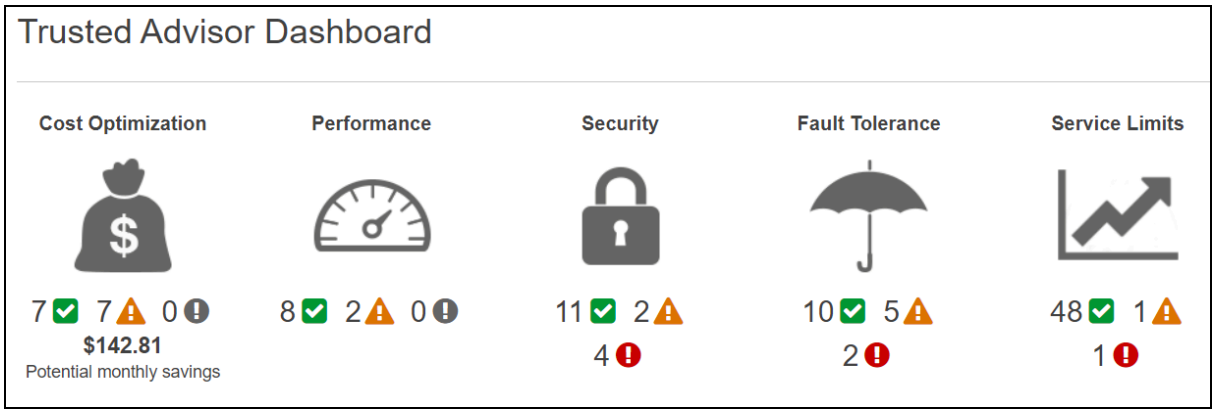
https://docs.aws.amazon.com/config/latest/developerguide/how-does-config-work.html
https://docs.aws.amazon.com/config/latest/developerguide/monitor-config-with-cloudwatchevents.html
https://docs.aws.amazon.com/config/latest/developerguide/approved-amis-by-id.html
https://docs.aws.amazon.com/config/latest/developerguide/conformance-packs.html
https://docs.aws.amazon.com/config/latest/developerguide/conformance-pack-dashboard.html
https://docs.aws.amazon.com/config/latest/APIReference/API_PutOrganizationConformancePack.html

Incident Response Management using Trusted Advisor

Tracking a few AWS resources in your account is no sweat, but as your business grows, the number of resources you manage will eventually add up. This could pose a severe security problem for you as it'll be challenging to keep tabs on various resources in terms of how permissive and restrictive they are.

Here is where Trusted Advisor comes into play. Trusted Advisor is a service that will analyze your AWS environment and provide best practice recommendations for you in five categories:

- Cost Optimization
- Performance
- Security
- Fault Tolerance
- Service Limits



Note: *Unlike AWS Config, Trusted Advisor does not allow users to specify resources to be assessed. AWS controls the resources and attributes to be inspected. You can think of it as a person who examines your AWS environment and gives you recommendations based on best practices.*

Since you're taking the AWS Certified Security Specialty Exam, our focus will be more on improving the security of your application on AWS. Below are some security best practices recommended to us by Trusted Advisor.

Multi-Factor Authentication on Root Account

AWS Trusted Advisor detects whether you have MFA enabled on your Root Account. It is advisable to use MFA when logging in as a root user for greater security. Note that an attacker will only need access to your account's email for him to reset your password. Without MFA, you're running the risk of only securing your

account with your email credentials. MFA requires users to type a secret code in addition to their password before they can access their account.

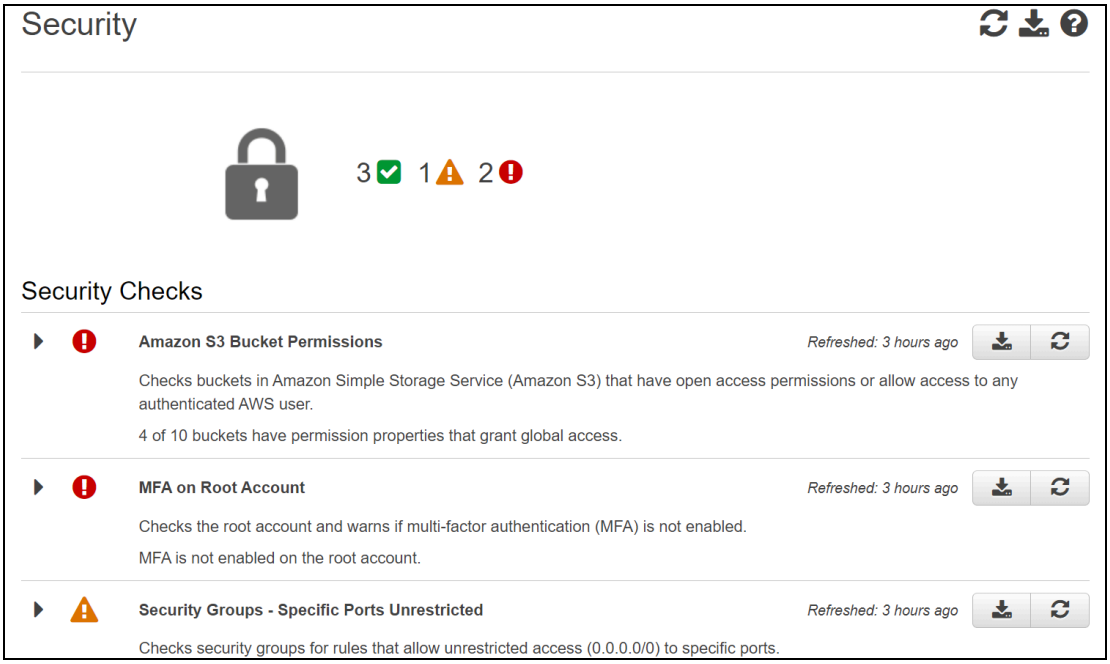
Amazon S3 Bucket Permissions

AWS Trusted Advisor checks if you have S3 buckets with open access permissions in your account. Confidential files stored on S3 buckets with open permissions can be exploited. Aside from storage, you also pay for requests against S3 buckets. Unauthorized users who spam S3 requests can lead to unexpected charges.

Security Groups - Unrestricted Access

AWS Trusted Advisor checks if you have security group rules that allow unrestricted access. Leaving your security groups open for other users increases your attack surface and is against the principle of least privilege access.

Here's how it looks on the Trusted Advisor Dashboard.



As you can see, there are three alert criteria shown (denoted by green, yellow, and red). These criteria reflect the current incident severity levels of your resources.

Green alert status means no problem is detected.

Yellow alert status indicates that an investigation should be done on the affected resources. You can think of this as an early warning for your resources that could create loopholes in your defense strategy.

Red alert status is the most critical alert criteria of the three. Resources tagged with red alerts have the most potential security vulnerabilities, which warrant an immediate solution.

Automated Monitoring of Trusted Advisor Security Checks

Like what we've discussed about AWS Config in the last section, we can also use Amazon EventBridge (Amazon CloudWatch Events) to detect the changes in the security check of Trusted Advisor based on an event rule. You can react to these changes by writing a function that will automatically take corrective action.

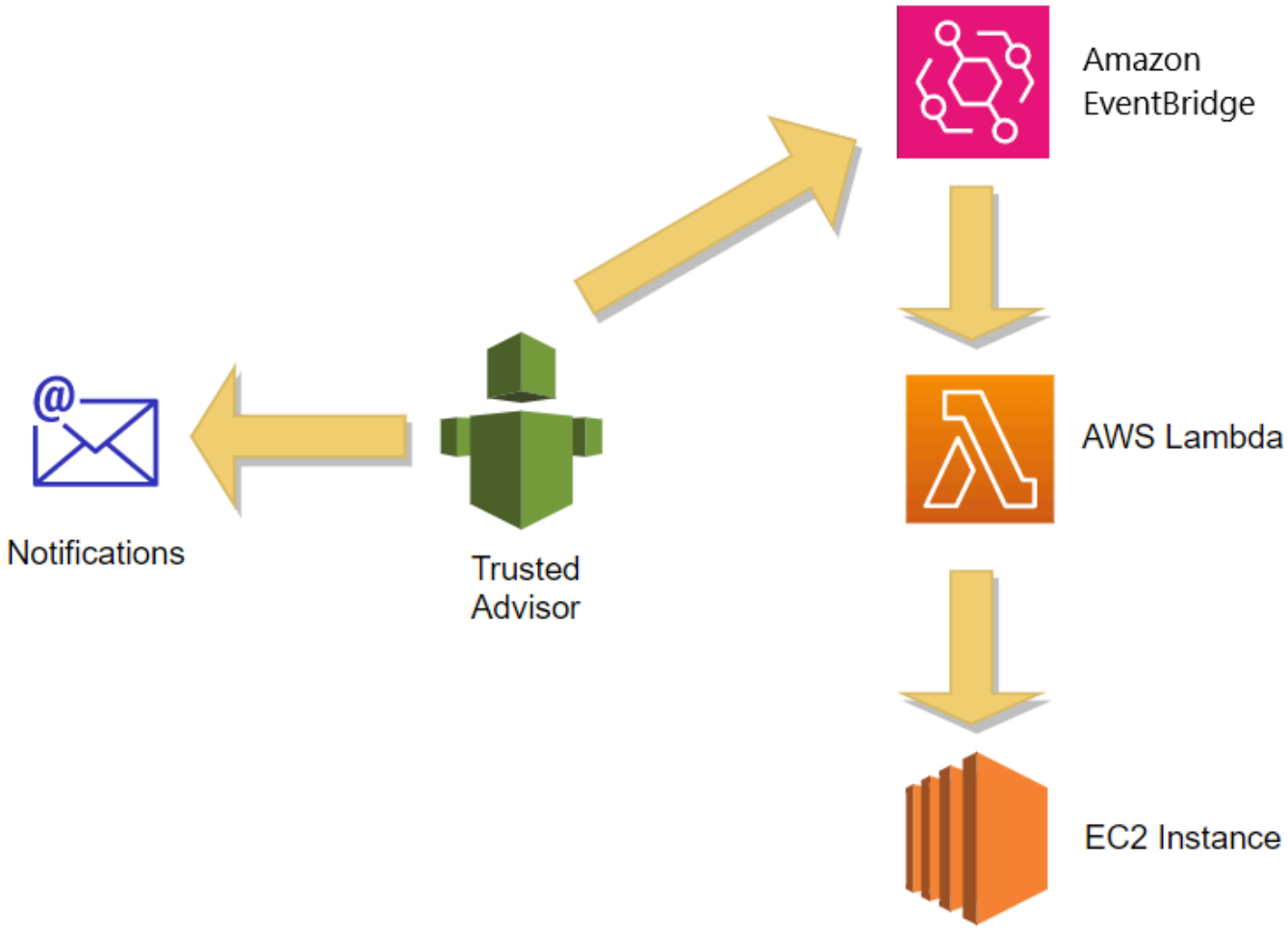
For example, let's say that you're managing an AWS account with different environments (Production, Development, UAT, etc.) with multiple EC2 instances. As part of your company's security policies, you need to restrict access to sensitive ports (e.g., port 22 for SSH, port 3306 for MySQL Database, etc.) by configuring the rules of security groups that are attached to active EC2 instances. It will be cumbersome to go through all the environments and check their security groups one by one. The Trusted Advisor can solve this dilemma.

By default, Trusted Advisor tags those security groups with access to sensitive ports as a red alert status, as you can see on the screenshot below:

Alert Criteria
Green: Access to port 80, 25, 443, or 465 is unrestricted.
Red: Access to port 20, 21, 1433, 1434, 3306, 3389, 4333, 5432, or 5500 is unrestricted.
Yellow: Access to any other port is unrestricted.

We can create an event pattern in Amazon EventBridge (Amazon CloudWatch Events) that will trigger a Lambda Function every time the Trusted Advisor identifies a security group with red alert status. The Lambda Function contains the logic that will call the appropriate EC2 API command — updating the security group's rule

to deny access to said vulnerable ports. Optionally, you can enable weekly email notifications from the Trusted Advisor to get information about your resources' status.



Security Specialty Exam Notes:

Trusted Advisor automatically checks your AWS environment against best practices. Similar to AWS Config, you can use **Amazon EventBridge (Amazon CloudWatch Events) + Lambda Function** for incident response.

References:
https://aws.amazon.com/premiumsupport/technology/trusted-advisor/best-practice-checklist/
https://docs.aws.amazon.com/awssupport/latest/user/cloudwatch-events-ta.html

Evaluating the Impact of an Exposed AWS Access Key

As more companies depend on Cloud to sustain and modernize operations, concerns revolving security also increases. There is a growing problem of publicly exposed IAM credentials being stored on a public repo that anyone could have access to. There are cases when somebody sends a text file containing information about his access keys to a colleague. There are also cases when a developer gets too complacent when building applications. His or her AWS credentials are stored on the source code, and that source code gets uploaded to a public repository. These "small" things could eventually lead to big problems in the future if not addressed.

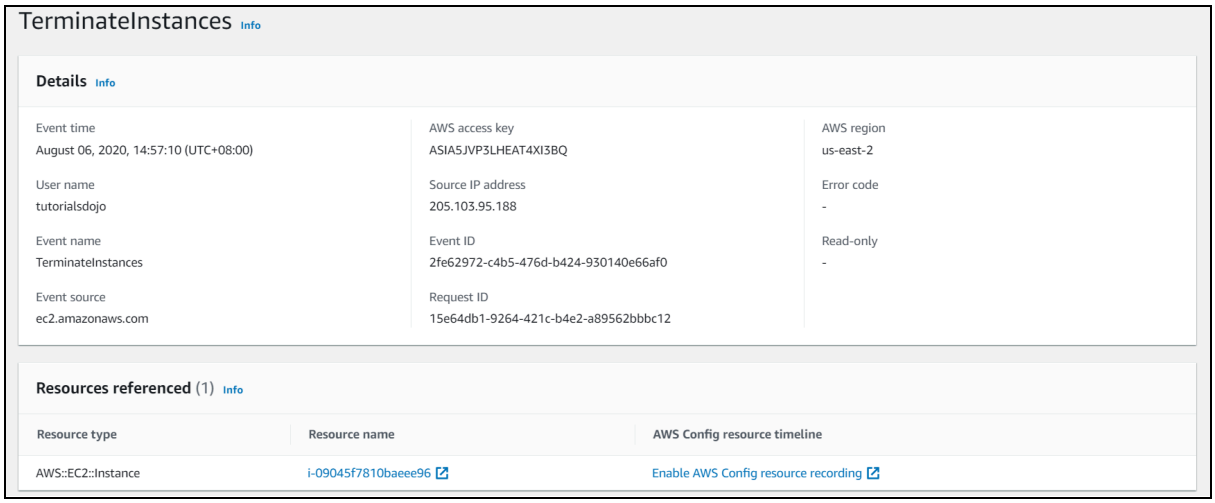
AWS access keys are powerful. They allow us to access AWS services programmatically. With just a few lines of code, the person who holds the compromised keys could terminate resources in seconds (if that individual has the necessary permissions). He or she could do anything that might jeopardize the projects everyone has been working on. And in the end, it is the careless user who will take the blame.

As a Security Specialist, it is your responsibility to assess the impact of compromised AWS access keys as early as possible and respond to it by doing the proper remedial action.

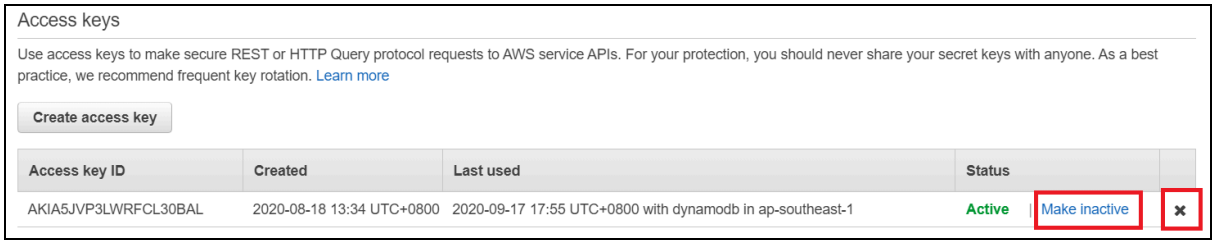
Using AWS CloudTrail

You can view and track the history of activities on your account for specific IAM users, roles, and AWS access keys through AWS CloudTrail event history. By doing this, you'll get an idea of how a particular credential was misused.

In the screenshot below, you can see different types of important information, such as the AWS access key used, the associated username, the date when the event happened, and the affected resource. You could also identify where the action was made by looking up the Source IP address on the internet.



After identifying the compromised access key, proceed to the IAM Console and find the user that has the exposed credentials. You could either **disable the exposed credentials** or **delete the exposed access keys.**



References:
https://aws.amazon.com/premiumsupport/knowledge-center/cloudtrail-search-for-activity/
https://aws.amazon.com/premiumsupport/knowledge-center/troubleshoot-iam-account-activity/

Incident Response Using Amazon GuardDuty

Amazon GuardDuty is a regional service used for intelligent threat detection. It analyzes billions of events across your AWS account so you can enhance the security of your resources by quickly responding to malicious and suspicious behaviors.

Data sources

- VPC Flow logs
  - Analyzes network traffic happening on Elastic Network Interfaces (ENIs).
  - You **DO NOT have** to turn on the VPC flow logs to generate findings. GuardDuty uses an independent stream on the AWS backend.

- DNS Logs
  - GuardDuty captures DNS logs derived from queries made from EC2 instances to known suspicious domains.
  - GuardDuty access DNS logs through AWS DNS resolvers (e.g., Route 53). Like VPC Flow Logs, you don't have to use Route 53 to generate DNS-based findings as GuardDuty uses an independent intern DNS resolver.
  - **DOES NOT WORK ON** 3rd Party DNS resolvers like OpenDNS, Google DNS, or Active Directory servers for domain servers.

- CloudTrail Events
  - View of the history of API calls made within your AWS Account and the user who made the call.

Finding Types

- **Backdoor** - indicates that your AWS resources are compromised and controlled by a command and control (C&C) server that is capable of doing malicious activities.
- **Behavior** - indicates that GuardDuty is detecting unusual activity patterns for a particular AWS resource.
- **Cryptocurrency** - indicates that GuardDuty is detecting a crypto mining-related activity in your AWS resource.
- **Pentest** - indicates that a penetration test is being carried out.

- **Persistence** - indicates that GuardDuty is detecting a principal in an AWS environment that is manifesting unusual behaviors.
- **Policy** - indicates that your AWS account is exhibiting behavior that goes against recommended security best practices.
- **PrivilegeEscalation** - indicates that a specific principal in your AWS environment is exhibiting behavior that can be indicative of a privilege escalation attack.
- **Recon** - indicates that there is an ongoing reconnaissance attack.
- **ResourceConsumption** - indicates when a principal in your AWS environment with no history of launching EC2 instances suddenly performs a RunInstances API action. This could be an indication of compromised IAM credentials.
- **Stealth** - indicates that an attack is attempting to hide its activities and tracks.
- **Trojan** - indicates a Trojan attack. A Trojan virus hides its real intentions by disguising itself as harmless software, making it easy for users to trust and install it on their computers.
- **UnauthorizedAccess** - indicates that GuardDuty is detecting a suspicious activity pattern by an unauthorized individual.

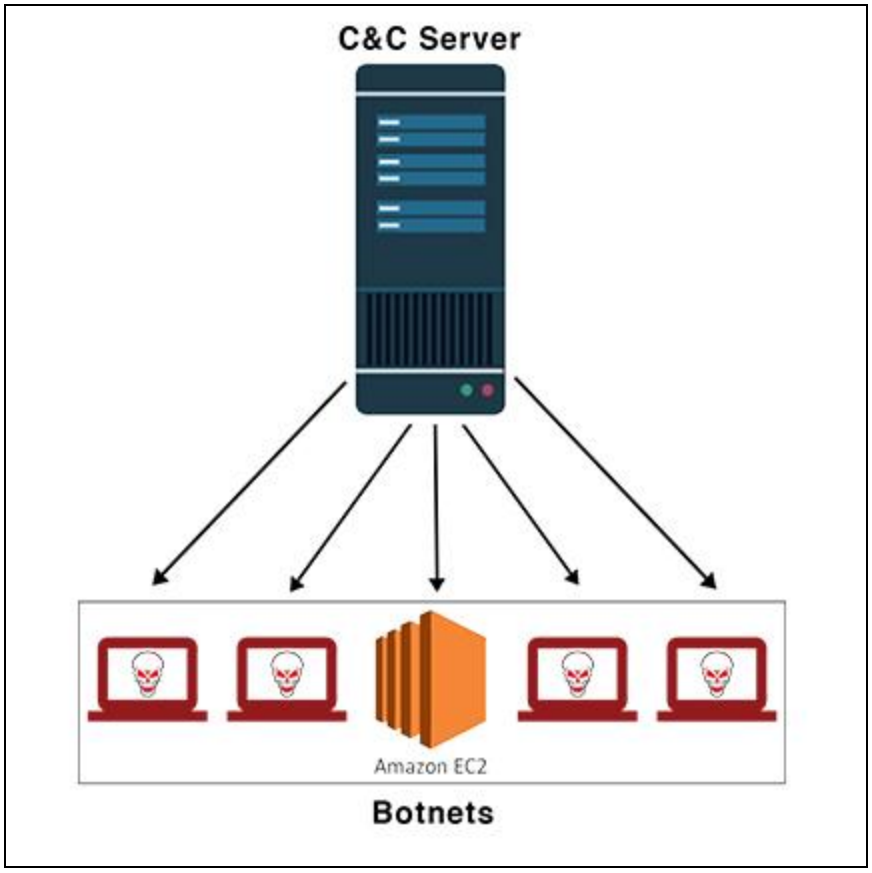
Using GuardDuty to detect EC2 instances that connect to known command and control (C&C) server

Command and Control (C&C) Server is a popular networking scheme responsible for dealing with malicious attacks such as Distributed Denial Of Service (DDOS), distributing malware that could mess up computer's registry entries, phishing attacks, or performing crypto mining without the user's consent.

C&C works by controlling a swarm of computers (usually called *botnets*) that perform unified, repeated harmful actions. Basically, they were built to search and destroy. They are similar to how zombies work; an infected computer infects another computer via malware.

As technology progresses, malware attacks also get smarter, which is harder to detect and deal with. It is only reasonable to leverage high-end security tools such as GuardDuty, which is easier to manage than provisioning your own "detector."

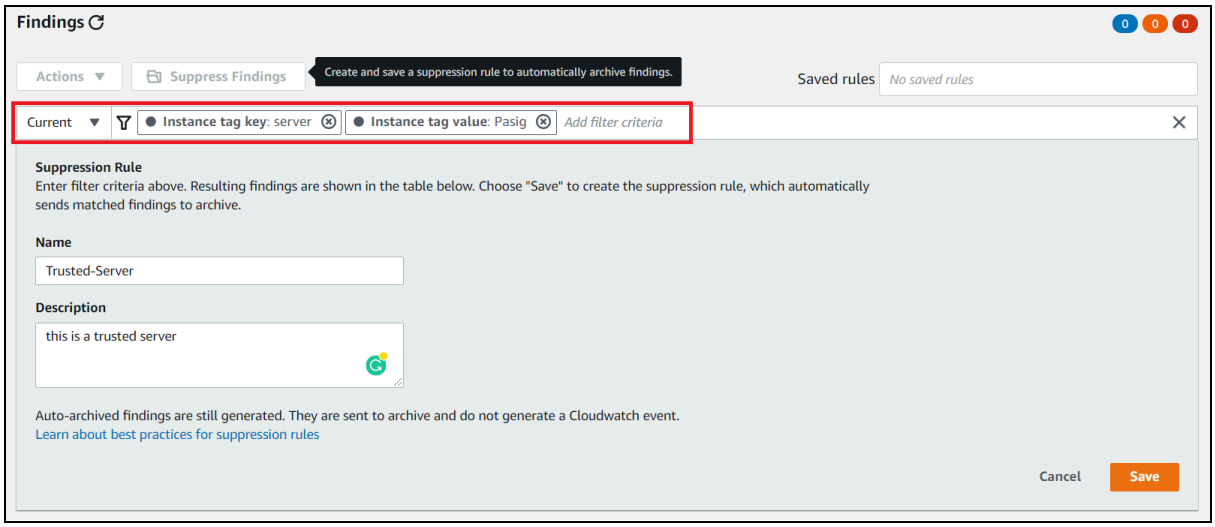
Let's say that you're managing a fleet of EC2 instances. The majority of them are infected by malware that consumes all of your CPU workloads to participate in crypto mining without you knowing. This is a high-level security threat because it could disrupt your service, which can lead to financial losses.



Enable GuardDuty to prepare your IT environment for this kind of scenario. With just one click, GuardDuty will continually monitor your AWS environment and report any findings to the AWS Console. Like the security assessment tools that we've discussed, GuardDuty can also be integrated with Amazon EventBridge (Amazon CloudWatch Events) to automate the response to an incident.

You could automate the sending of alerts to your security team or trigger a Lambda Function to programmatically isolate the affected instances.

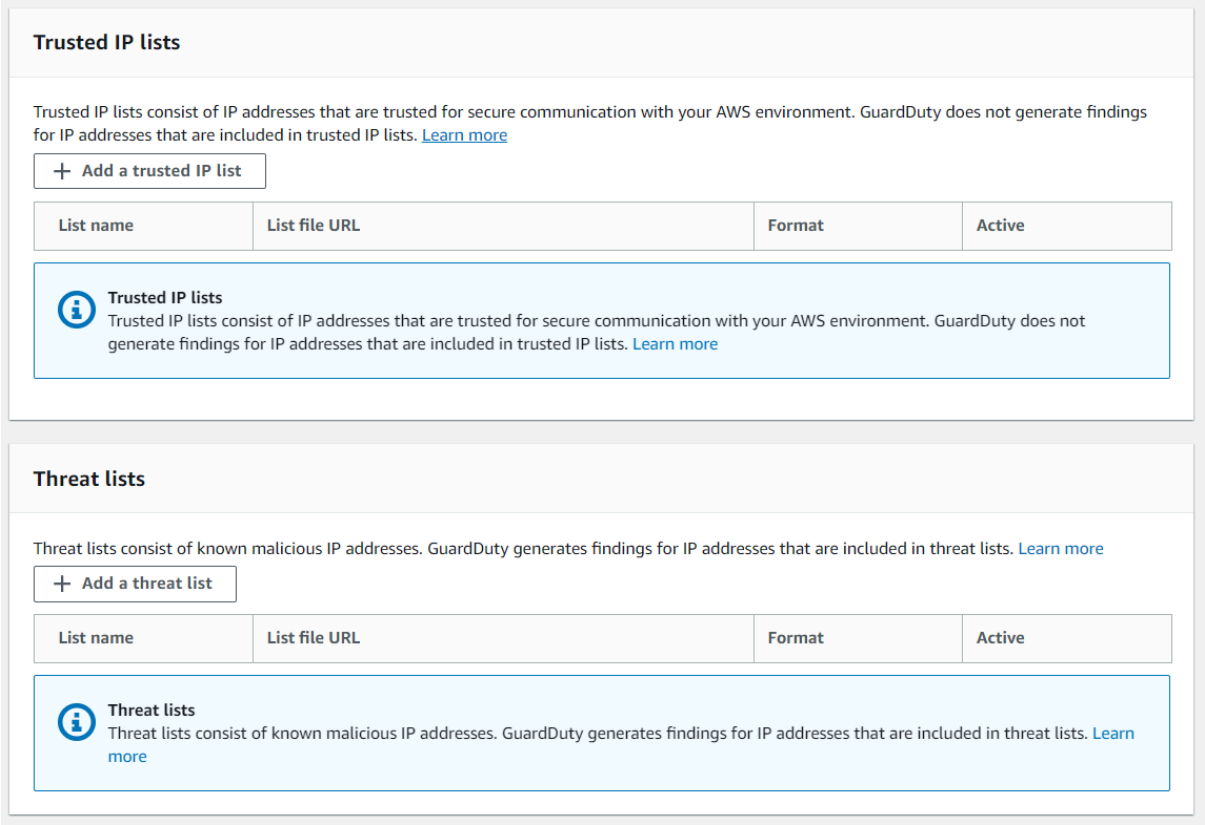
There are also cases when you want to identify false alarms when you're conducting internal security tests. You can achieve that by optionally adding a ***suppression rule*** that will filter new findings that match the filter criteria defined in the rule. The findings are automatically archived.



You can whitelist IP addresses using the suppression rule, but to further simplify the process, GuardDuty provides a feature where you can define a list of Trusted IP addresses and Threat lists.

Trusted IP lists refer to the list of whitelisted IP addresses. GuardDuty does not generate reports against activities that originate from the Trusted IP lists. You can only upload one trusted IP list per account per region.

Threat lists refer to the list of IP addresses that are known for exhibiting malicious behaviors. GuardDuty produces findings based on threat lists. You can upload six threat lists per account per region.



References:

https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_finding-types-active.html
https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_findings_cloudwatch.html
https://docs.aws.amazon.com/guardduty/latest/ug/findings_suppression-rule.html

AWS Personal Health Dashboard

AWS Personal Health Dashboard (PHD) offers a personalized view of service events that may be impacting your AWS resources. Events are grouped into three catergories:

1. Open issues - refers to operational issues such as increased latency for certain APIs, abnormal error rates, or a power loss in a datacenter.
- Scheduled changes - refers to upcoming maintenance activities that might have an impact on your resources.
- Notifications - refers to other events such as billing notifications, certificate rotations, internal changes, or something that may have an impact on security, such as the end of support for a specific runtime in AWS Lambda.

The dashboard requires no setup, and it is ready to use for authenticated AWS users. The Personal Health Dashboard keeps an event history as well so you can review the events that has happened over time.

How is it different from Trusted Advisor?

Short Answer:

PHD monitors AWS resources (the cause of the issue is on the AWS's side) while Trusted Advisor monitors your AWS environment (the cause of the issue is your fault).

Long Answer:

Have you ever experienced running into a problem with one of the AWS services, and you can't figure it out? Everything was working fine before, and all of your configuration settings are unchanged. Then, out of the blue, unexpected behavior of an AWS service suddenly occurs, which causes operational issues. You start debugging, hoping to unveil the root cause. You may never find it because it's highly likely that the problem comes from AWS's underlying infrastructure that you no longer have control over. It also means that other customers are possibly experiencing the same issue as you.

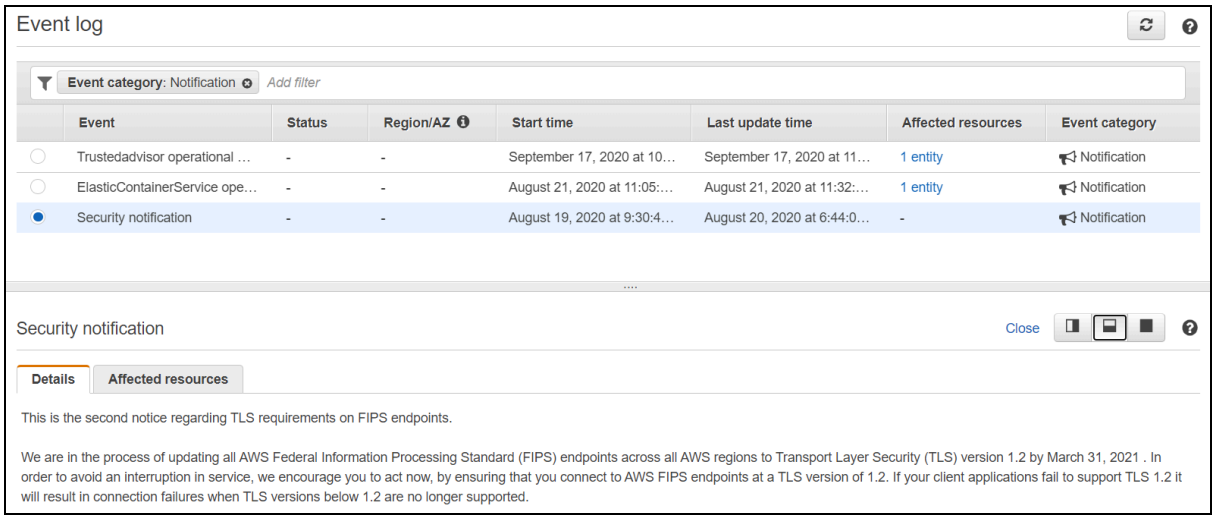
This is where PHD can help you. Instead of wasting time debugging an issue that is out of your hands, PHD can notify you and provide instructions that you can follow to alleviate the situation.

Trusted Advisor, on the other hand, analyzes your AWS environment and gives best practice recommendations. Trusted Advisor just monitors the AWS resources within your domain. It means that issues that may arise are unique to your account, and it is solely your responsibility to solve them.

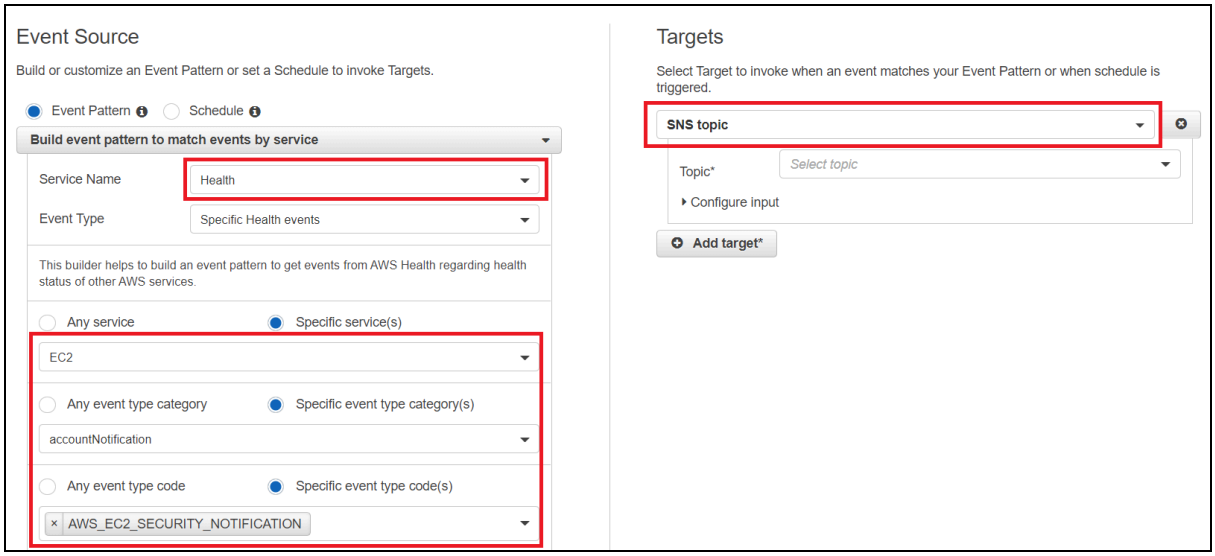
Adding Alerts For Security Notifications

Like other security assessment tools that we have discussed, we can integrate Personal Health Dashboard into Amazon EventBridge (Amazon CloudWatch Events) to automate our incident response plan.

You can view security issues that may come up, which could affect your resources in the Personal Health Dashboard. Here is an example of a security notification regarding TLS requirements on FIPS endpoints. You can also view the affected resources on the dashboard.



We don't have to check the dashboard every now and then to get information regarding a security issue. We can automate the notification process by setting an event rule that matches the event type and event code of interest. On the target, we choose an SNS Topic to which an Amazon EventBridge rule will deliver the information details regarding an event. SNS sends notifications if a security notification occurs.



References:
https://aws.amazon.com/premiumsupport/technology/personal-health-dashboard/
https://docs.aws.amazon.com/health/latest/ug/cloudwatch-events-health.html

Using Amazon S3 Object Lock for Preserving Forensic Evidence

S3 Object Lock is an Amazon S3 feature that enables Write Once, Read Many (WORM) protection for your stored objects. This means that once enabled, the data cannot be deleted or altered by any user, including those with administrative access, until the lock expires. This immutability is crucial for compliance, data retention policies, and for securing evidence in the form of forensic artifacts during threat detection and incident response.

Maintaining the integrity of forensic artifacts is crucial for analysis and potential legal proceedings. Imagine you've suffered a data breach. You've managed to isolate some suspicious files that might indicate how the attackers got in. These digital evidence are invaluable, and you need to make sure they remain exactly as they are for investigation. By using S3 Object Lock, you can preserve these artifacts in an unalterable state. Once you've identified these critical files, you can immediately apply Object Lock to ensure they can't be deleted or changed, thereby maintaining their integrity. This not only aids in a thorough and uncompromised investigation but also satisfies the regulatory requirements that often follow a security incident.

S3 Object Lock has two types of retaining objects:

1. Retention Periods - when you activate a retention period for an object, you specify a fixed time period during which the object cannot be modified or deleted. This is particularly useful for ensuring that forensic artifacts are kept intact for a predetermined period required for an investigation or compliance.

2. Legal Holds: A legal hold provides the same immutability benefits but has no predetermined time limit. The object remains immutable until you explicitly remove the legal hold. This is often used in situations where you're uncertain about the length of time the object needs to be retained.

References:
https://docs.aws.amazon.com/AmazonS3/latest/userguide/object-lock.html
https://aws.amazon.com/blogs/storage/protecting-data-with-amazon-s3-object-lock/

Using Amazon Inspector for Incident Management

Amazon Inspector plays a crucial role in incident management by proactively identifying security vulnerabilities, helping organizations detect and respond to potential threats before they can be exploited. It integrates with AWS Security Hub and Amazon EventBridge to automate security responses.

How Amazon Inspector Aids in Incident Management

- **Automated Threat Detection** – Continuously scans workloads to identify security vulnerabilities and misconfigurations.
- **Real-time Findings and Alerts** – Security issues are categorized based on severity levels (Critical, High, Medium, Low, Informational) and can trigger automated alerts.
- **Integration with AWS Security Hub** – Findings are automatically shared with Security Hub, enabling centralized incident monitoring.
- **Automated Remediation with EventBridge** – Findings can trigger automated workflows to mitigate security issues, such as patching vulnerabilities or isolating compromised resources.
- **Support for Compliance and Auditing** – Provides security assessments aligned with compliance frameworks, helping organizations maintain regulatory compliance.

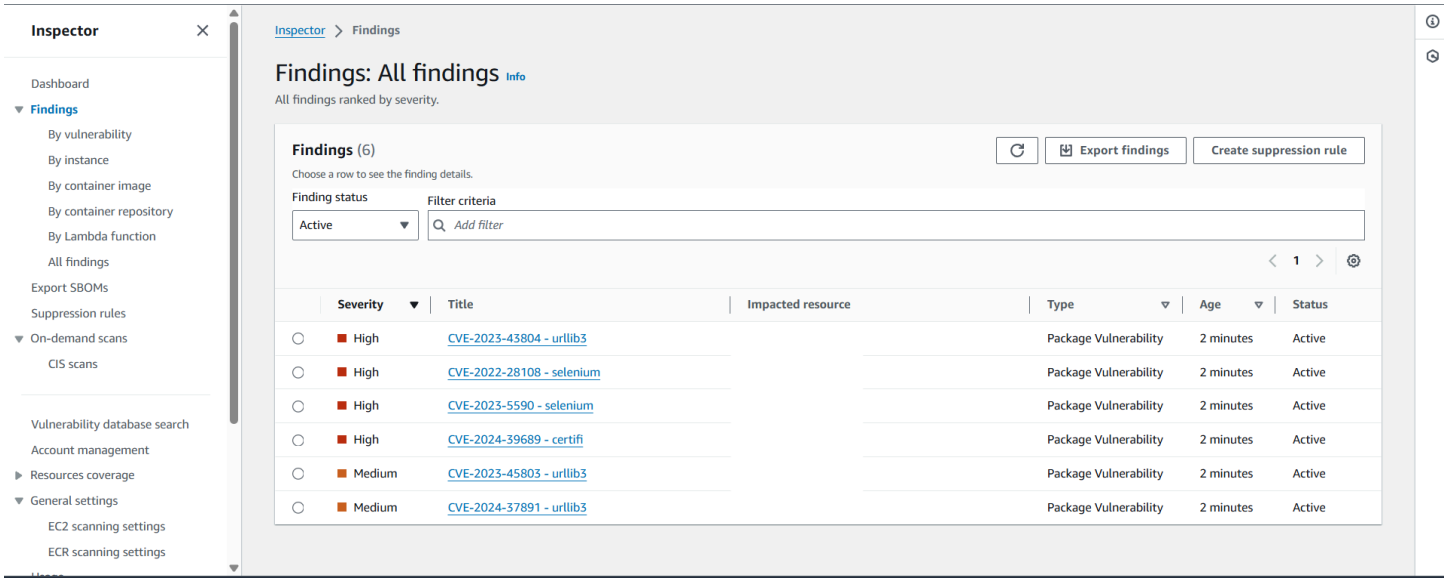
Features

- Amazon Inspector v2 automatically scans AWS workloads for known vulnerabilities and security risks by integrating with AWS Systems Manager, Security Hub, and EventBridge for continuous monitoring and response.

- Amazon Inspector v2 uses an AWS-managed vulnerability database, including the latest Common Vulnerabilities and Exposures (CVE) data, to assess EC2 instances, ECR images, and Lambda functions for security risks.

Two Types Of Assessments

Assessment Type	Assessments Performed	Inspector Agent	Pricing
EC2 Instance Scanning	Identifies vulnerabilities and missing security patches using CVE data	SSM Agent Required	Per instance scanned per month
ECR Image Scanning	Scans container images for known vulnerabilities at push and periodically	Required	Per repository scanned per month
Lambda Function Scanning	Scans deployed Lambda functions and their dependencies for vulnerabilities	Agentless	Per function scanned per month



- Each rule has an assigned severity level

Severity Levels	Indication	Recommended Action
Critical	Immediate security risk	Requires urgent remediation
High	Extremely Urgent	Should be addressed quickly
Medium	Somewhat Urgent	Fix the issue at the next possible opportunity (e.g. next update)

Low	Less urgent	Can be scheduled for future updates
Informational	Not an issue, but security insights	Can be reviewed for security improvements

- There is an additional severity level called *Informational*. This level simply highlights a security configuration detail of your assessment target. You can use this information to improve the security of your assessment target.
- Amazon Inspector v2 identifies security misconfigurations, including open ports, mismanaged security groups, and risky firewall settings, as part of its EC2 vulnerability assessments. Findings are integrated with AWS Security Hub for further analysis and response.
- In addition to vulnerability scanning, Amazon Inspector also provides network reachability findings, which analyze security group and network ACL configurations to determine whether resources are publicly accessible or exposed to unwanted inbound traffic.

Amazon Inspector v2 continuously monitors workloads and automatically generates security findings, which can be viewed in the AWS Console, Security Hub, or retrieved via AWS CLI and API. Findings include details such as affected resources, CVSS scores, and remediation recommendations.

References:
https://docs.aws.amazon.com/inspector/latest/user/what-is-inspector.html
https://docs.aws.amazon.com/inspector/latest/user/scanning-resources.html
https://docs.aws.amazon.com/inspector/latest/user/findings-understanding-severity.html

AWS Systems Manager Patch Manager

Importance Of Patching

You may find software updates and patches annoying, but they should not be left unattended. Software vendors are constantly releasing patches in an effort to find security flaws in their products, which may affect millions of their customers. However, the decision of whether to install the patch or not still depends on the customer's discretion.

Why Use Systems Manager Patch Manager?

Admit it or not, installing patches is a tedious and repetitive task (but we have to do it!). Why do all these mundane administrative tasks in a manual fashion if you can automate them?

Systems Manager Patch Manager can help you automate the process of patching your EC2 or on-premises instances. It enables you to scan instances for missing patches and apply them individually or to large groups of instances using EC2 instance tags. You can scan instances to see only a report of missing patches or check and automatically install all missing patches.

Patch Manager Concepts

- **Patch baselines**
    - Acts as rules for auto-approving patches within days of their release, as well as a list of approved and rejected patches. Basically, patch baselines define which patches are approved to be installed on your instances. Alternatively, you can create your own patch baselines for more control.
- **Patch groups**
    - Patch groups allow you to organize your instances for patching. You can create a patch group to separate instances with different operating systems or instances within different environments. This can be easily done by using EC2 tags.
- **Maintenance window**
    - Set up recurring schedules for installing patches and updates without interrupting business-critical operations to your instances.

References:
https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-patch.html

AWS Systems Manager State Manager

State Manager is a secure and scalable capability within AWS Systems Manager that automates the process of maintaining your managed nodes in a desired configuration state. This tool helps prevent configuration drift across your Amazon EC2 instances, edge devices, and hybrid infrastructure by continuously enforcing the configurations you define. State Manager operates through associations, which specify schedules and targets for applying configurations. For example, an association for antivirus software might run daily, automatically installing it if missing or starting the service if stopped. The service is available at no additional charge.

How State Manager Works Through Associations

An association defines your desired state by specifying an SSM document, parameters, targets, and a schedule. Key features include:

- Flexible targeting: Use tags, individual node IDs, AWS Resource Groups, or all managed nodes in your Region
- Preconfigured documents: Systems Manager includes dozens of ready-to-use SSM documents for common tasks like installing applications, configuring CloudWatch, running automations, and executing scripts
- Automated scheduling: Associations run immediately by default, then continue on your defined cron or rate schedule
- Compliance tracking: Monitor status through the console and store outputs in Amazon S3

Advanced Capabilities and Use Cases

State Manager integrates with Automation runbooks to manage various AWS resources, enabling you to:

- Attach IAM roles to EC2 instances and enforce security group rules
- Create or delete DynamoDB and EBS backups
- Control S3 bucket permissions and manage instance or RDS states
- Bootstrap nodes with software at startup and update agents on defined schedules

State Manager is ideal for organizations requiring strict compliance reporting, those using Auto Scaling groups, or teams maintaining consistent configurations across large resource fleets.

References:
https://docs.aws.amazon.com/systems-manager/latest/userguide/state-manager-about.html
https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-state.html
https://aws.amazon.com/blogs/mt/configure-amazon-ec2-instances-in-an-auto-scaling-group-using-state-manager/
https://docs.aws.amazon.com/systems-manager/latest/userguide/scheduling-automations-state-manager-associations.html

AWS Systems Manager OpsCenter

OpsCenter serves as a unified hub within AWS Systems Manager where IT operations teams can manage and resolve operational issues affecting their AWS infrastructure. The service consolidates operational work items, known as OpsItems, from various AWS services into a single interface, significantly reducing the time needed to diagnose and fix problems. When issues arise, OpsCenter automatically enriches each OpsItem with essential diagnostic information including resource identifiers, event histories, alarm trends, and visual timelines. Through its integration with CloudWatch and EventBridge, the system can autonomously generate OpsItems when monitoring thresholds are breached or when significant events occur across your AWS environment.

Key Features and Integration Capabilities

OpsCenter offers comprehensive features to streamline operational management:

- Automatic OpsItem creation: Integration with CloudWatch, CloudWatch Application Insights, EventBridge, AWS Config, and AWS Security Hub
- Related resources: Specify Amazon Resource Names (ARNs) of related resources to help avoid duplicate OpsItems and view details about similar issues
- Bulk editing: Select multiple OpsItems and edit fields such as Status, Priority, Severity, and Category simultaneously
- SNS notifications: Configure Amazon SNS topics to receive notifications whenever an OpsItem is changed or edited
- Comprehensive search: Search OpsItems by ID, title, last modified time, operational data value, source, and automation ID
- Summary reports: View automatically-generated reports showing OpsItems by status and sources with most open OpsItem

OpsCenter's integration with Systems Manager Automation provides a powerful remediation framework through pre-built runbooks. When investigating an OpsItem, operators can access a curated list of relevant automation workflows specifically applicable to the affected resource. Once a runbook is executed, the system maintains the association for future reference, building organizational knowledge over time. The platform retains a month-long history of automation executions for each OpsItem, enabling teams to track resolution patterns and refine their response procedures. These automated workflows can perform diverse operations including instance management, database operations, backup procedures, and security policy enforcement. For enterprise environments, OpsCenter extends its capabilities across organizational boundaries, enabling designated administrator accounts to orchestrate issue resolution across multiple AWS accounts without switching contexts, streamlining operations for complex multi-account architectures.

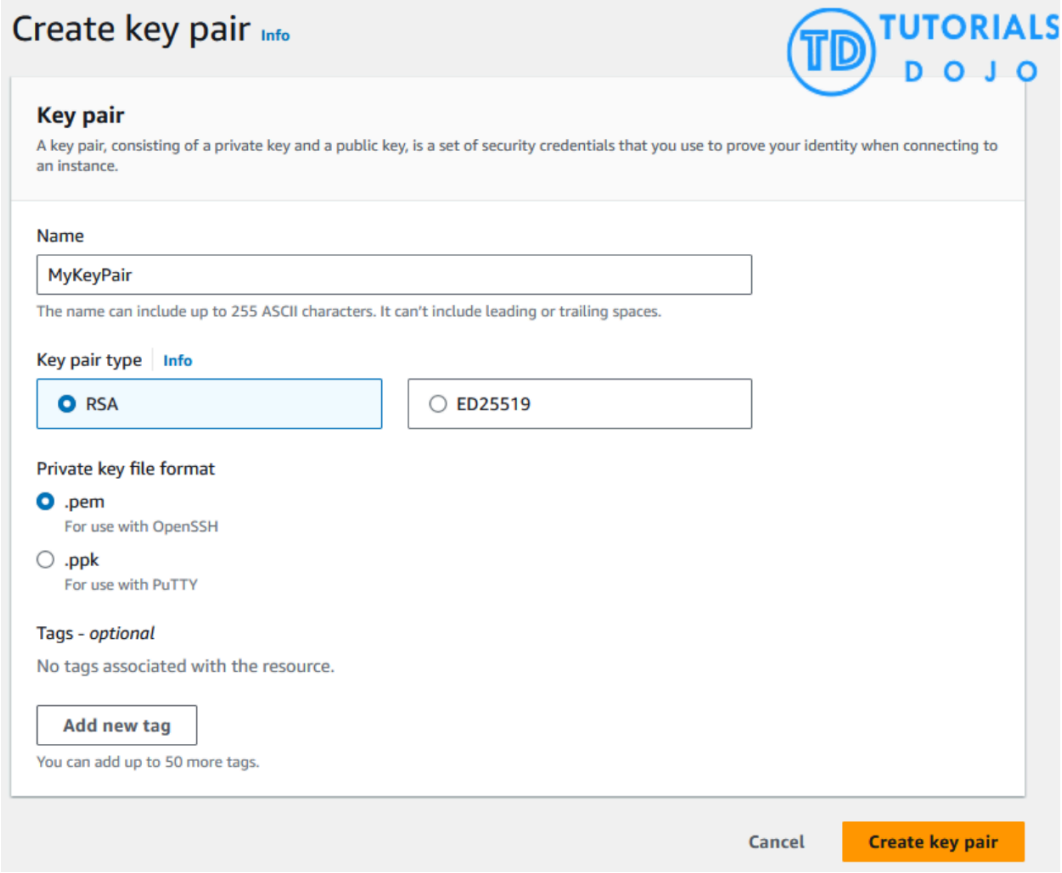
References:
https://docs.aws.amazon.com/systems-manager/latest/userguide/OpsCenter.html
https://docs.aws.amazon.com/systems-manager/latest/userguide/incident-manager.html
https://docs.aws.amazon.com/systems-manager/latest/userguide/OpsCenter-remediating.html
https://docs.aws.amazon.com/systems-manager/latest/userguide/OpsCenter-setting-up-cross-account.html

Securing Amazon EC2 Instances That Have Compromised SSH Private Keys

The most common way of connecting to EC2 instances from your computer is via the Secure Shell (SSH) protocol. SSH is the de-facto standard for securing a connection between two computers for remote administration.

SSH uses public-key cryptography (also called asymmetric cryptography) to encrypt and decrypt login information. Public key cryptography is a process that involves two keys: a public key for encryption and a private key for decryption. The public and private keys together are known as a *key pair*. Public key cryptography enables you to securely log in to your instances using a private key instead of a username/password.

Only the public key is stored in the EC2 instance. The private key is saved to your local computer. Anyone who can get hold of your keys can decrypt your login information and connect to your instance. It would be best if you kept the private keys securely by restricting other users from accessing it.



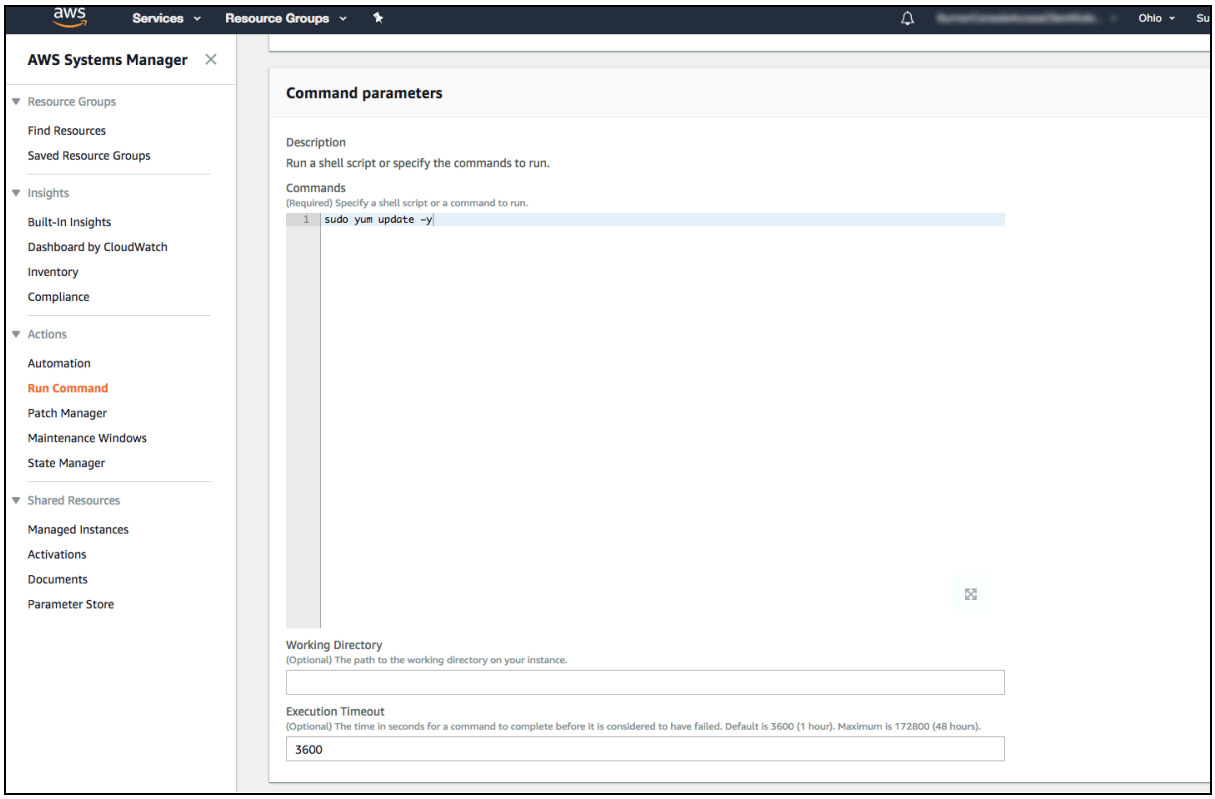
When connecting to an EC2 instance, you must specify a key pair along with the instance's hostname. At boot time, the public key content is placed on your Linux instance in an entry within *~/.ssh/authorized_keys*. When you connect to your instance using SSH, you must specify the private key that corresponds to the public key content to log in.

You can never recover a private key once you lose it. This could present a serious security issue to your managed instances. Imagine having your office laptop where dozens of SSH keys are stored get stolen. What would you do? You might be thinking, "*I could just shut the instances down and replace them with new ones with new key pairs.*" That is possible but imagine hosting an application in a production environment or across multiple instances that can't afford downtime. That approach is not feasible.

A better solution is to use AWS Systems Manager Run Command to protect your running instances by removing all public keys (associated with the compromised SSH private keys) stored in the *~/.ssh/authorized_keys* file of all affected EC2 instances. AWS Systems Manager Run Command lets you remotely and securely manage your instances' configuration without opening an inbound port in your security group's rule.

You can use the Run Command using the AWS console, the AWS Command Line Interface, AWS Tools for Windows PowerShell, or the AWS SDKs.

Below is an image of the Run Command from the AWS Console. Here, it is used to update the software packages of an instance.



Security Specialty Exam Notes:

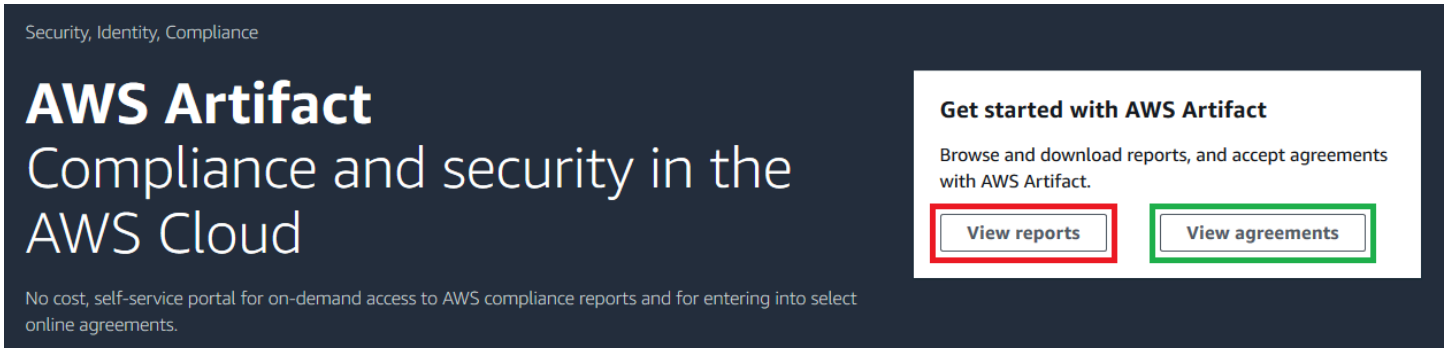
AWS Systems Manager Run Command doesn't require a key pair when connecting to an instance. You just need to associate an IAM role that will give Systems Manager permission to perform actions on your instances. With this, you would still be able to login and recover a compromised EC2 instance.

References:

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/replacing-lost-key-pair.html
https://docs.aws.amazon.com/systems-manager/latest/userguide/execute-remote-commands.html

AWS Artifact Security Reports and AWS Compliance-Related Information

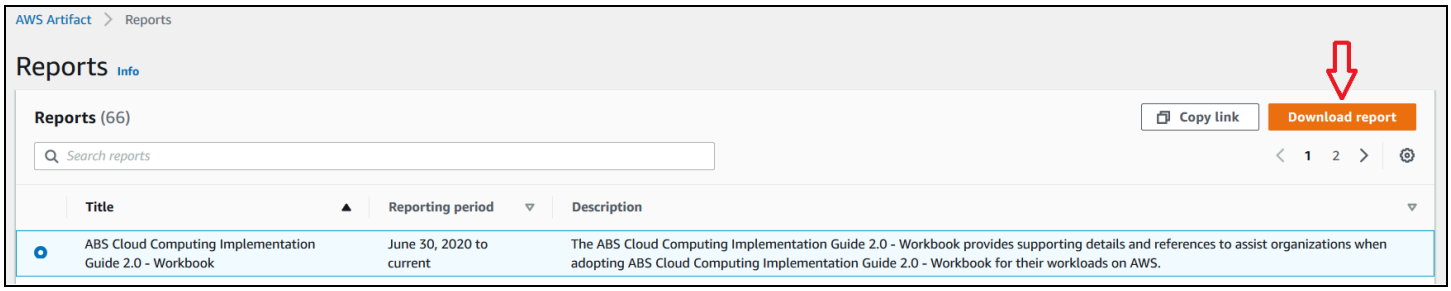
AWS Artifact is a self-service central repository of AWS' security and compliance reports and select online agreements. The security and compliance documents are called audit artifacts. An audit artifact is a piece of evidence that demonstrates that an organization is following a documented process or meeting a specific requirement (business compliant).



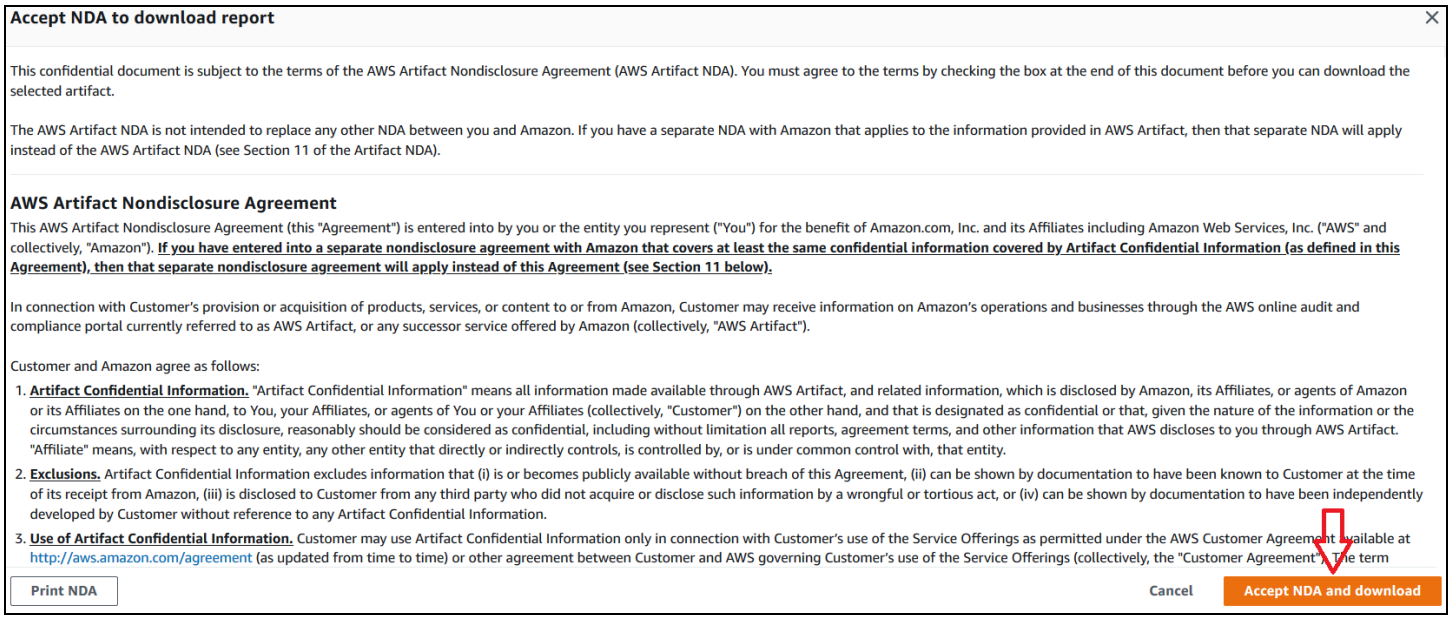
AWS Artifact Reports contain several compliance reports that are tested and verified by third-party auditors. It includes ISO, Service Organization Control (SOC) reports, Payment Card Industry (PCI) reports, and certifications that approve the execution and efficacy of AWS security controls.

To download a report, you can go to the AWS Artifact dashboard and click on view reports.

1. Select a report title and click on the **Download Report**.



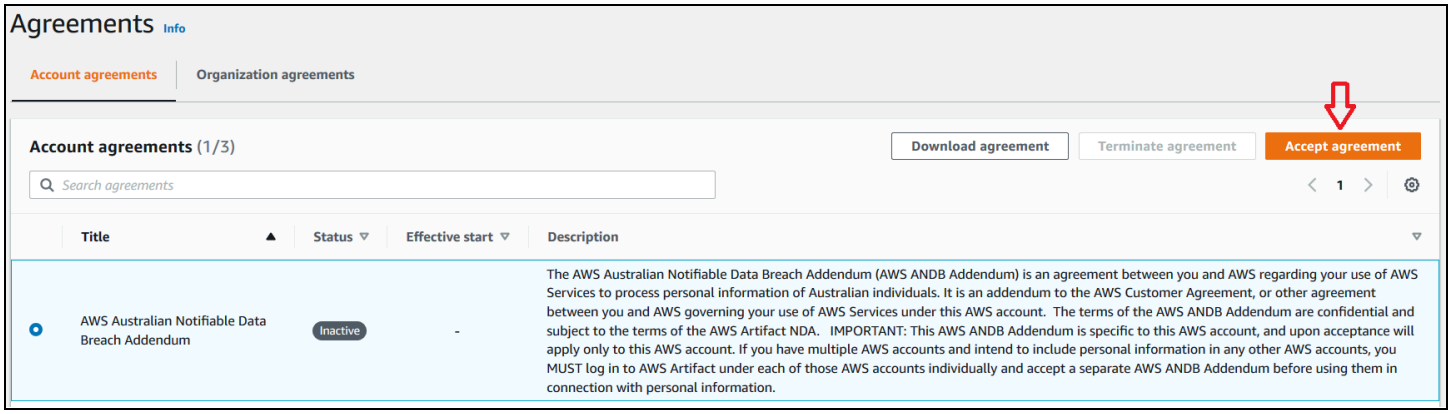
2. Read and agree to all the terms of the NDA and click the download button.



3. To open the downloaded report, you can use a PDF reader.

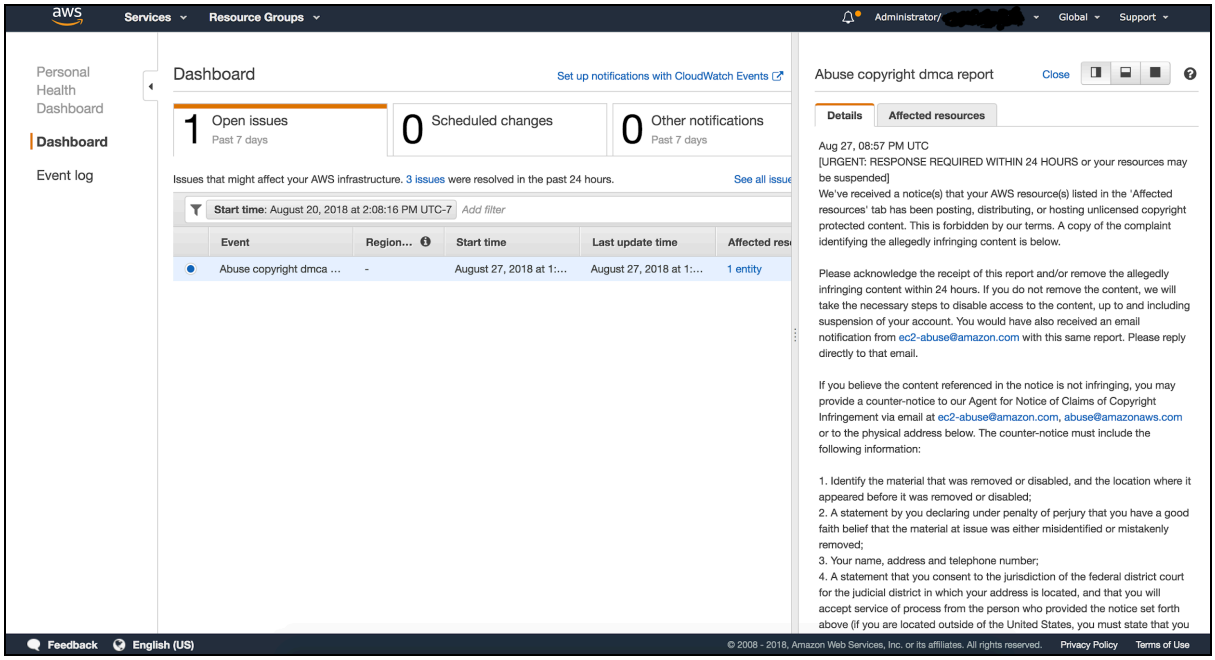
AWS Artifact Agreements allow you to review, accept, and manage agreements in your personal AWS account and all the accounts that are part of AWS Organizations. If you no longer need the accepted agreement, you can terminate it in the account agreements section.

- **Account Agreements** apply only to the individual account you used to sign into AWS.
- **Organization Agreements** apply to all accounts in an organization created through AWS Organizations, including its master account and all member accounts. Only the master account in an organization can accept agreements in AWS Artifact Organization Agreements.



AWS Abuse

AWS Abuse sends reports to customers about potentially abusive activities that are going on in their AWS environment.



The AWS Abuse team can assist you when AWS resources are used to engage in the following types of abusive behavior:

- Spam
- Port scanning
- Denial-of-service (DoS) attacks
- Intrusion attempts
- Hosting objectionable or copyrighted content
- Distributing malware

Response in AWS Abuse Notice

Note that AWS takes security seriously that if you fail to respond to an abuse notice, AWS might block your resources, or your account may get suspended.

The AWS Abuse Team will send abuse reports to your security contact (if there are any) or to the email address listed on your account.

Do the following if you receive an abuse notice:

1. Review the abuse notice and determine what type of abusive behavior was reported.
2. Reply directly to the email you received from the abuse team. Your message must contain the preventive measures that you're taking to avert abusive activity from recurring.
3. Closely monitor your email address for incoming messages, as the AWS Abuse Team might ask you for additional information.

References:
https://aws.amazon.com/premiumsupport/knowledge-center/aws-abuse-report/
https://aws.amazon.com/blogs/mt/automating-processes-for-handling-and-remediating-aws-abuse-alerts/

Amazon Route 53 Resolver logs

Amazon Route 53 Resolver query logging is a security and compliance feature that captures information about DNS queries that originate in your VPC and are answered by Route 53 Resolver. These logs provide visibility into DNS lookup patterns, enabling security teams to detect reconnaissance activities, data exfiltration attempts, command-and-control communications, and other DNS-based threats. Resolver query logs are a critical data source for security monitoring, threat hunting, and incident investigation.

DNS is often exploited by attackers for various malicious purposes including domain generation algorithms (DGA), DNS tunneling for data exfiltration, and lookups to command-and-control infrastructure. By analyzing Route 53 Resolver logs, security teams can identify these patterns and take appropriate action before significant damage occurs. These logs are particularly valuable when integrated with Amazon GuardDuty, AWS Security Hub, and centralized logging solutions like Amazon Security Lake.

What Are Route 53 Resolver Query Logs?

Route 53 Resolver query logs capture DNS queries that are forwarded by the Route 53 Resolver from your Amazon VPC. When a resource in your VPC makes a DNS query, the Route 53 Resolver handles that query and

can optionally log detailed information about it. These logs include the query name (domain being looked up), query type (A, AAAA, CNAME, etc.), response code, source IP address, and timestamp.

Key Characteristics

- Logs are generated for DNS queries originating from VPC resources
- Captures both successful and failed DNS lookups
- Includes queries to public domains and private hosted zones
- Does not log queries that bypass Route 53 Resolver (e.g., direct to external DNS)
- Can be configured per VPC with flexible destination options
- Near real-time log delivery for timely security analysis

What Gets Logged

- Query name (domain being queried, e.g., example.com)
- Query type (A, AAAA, CNAME, MX, TXT, etc.)
- Response code (NOERROR, NXDOMAIN, SERVFAIL, etc.)
- Query class (IN for Internet)
- Source IP address (EC2 instance, Lambda function, etc. making the query)
- VPC ID where the query originated
- Query timestamp (with millisecond precision)
- Resolver endpoint ID (if using Resolver endpoints)
- Transport protocol (UDP or TCP)
- EDNS client subnet information (if available)

What Does NOT Get Logged

- DNS queries that don't go through Route 53 Resolver (direct to 8.8.8.8, custom DNS)
- DNS responses or answer sections (only metadata, not the IP addresses returned)
- Queries from resources outside the VPC
- Route 53 public DNS queries (only Resolver queries from VPCs)

How Route 53 Resolver Query Logging Works

When you configure query logging for a VPC, Route 53 Resolver begins capturing metadata about DNS queries. The logging process is transparent to applications and has no impact on DNS resolution performance. Logs are delivered to your specified destination within seconds of the query being made, enabling near real-time security monitoring.

Configuration Process

1. Create a query logging configuration in Route 53 Resolver
2. Specify one or more VPCs to log queries from

3. Choose a destination (CloudWatch Logs, S3, Kinesis Data Firehose)
4. Route 53 Resolver begins capturing queries from specified VPCs
5. Logs are delivered to the configured destination

Destination Options

Amazon CloudWatch Logs

- Queries delivered to CloudWatch Logs log group
- Use CloudWatch Logs Insights for ad-hoc query analysis
- Set up metric filters and CloudWatch alarms for real-time alerting
- Retention policies control how long logs are stored
- Good for real-time monitoring and troubleshooting

Amazon S3

- Queries delivered to S3 bucket via Kinesis Data Firehose
- Cost-effective for long-term storage and compliance
- Use Amazon Athena for SQL-based analysis
- Integrate with third-party SIEM tools
- Apply S3 Lifecycle policies for cost optimization
- Good for compliance, audit trails, and historical analysis

Amazon Data Firehose

- Stream logs to S3, OpenSearch, or third-party destinations
- Transform logs using Lambda before delivery
- Buffer and compress logs for efficiency
- Supports delivery to Splunk, Datadog, New Relic, etc.
- Good for streaming to external security tools

References
https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resolver-query-logs.html
https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resolver-query-logs-choosing-target-resource.h

Creating Metrics, Alerts, and Dashboards to Detect Anomalous Data and Events

Detecting anomalous data and security events is a critical skill for AWS security professionals. AWS provides several native services that work together to monitor, detect, and alert on potential security threats across your cloud environment. The four primary services you need to master for the SCS-C03 exam are Amazon GuardDuty, Amazon Security Lake, AWS Security Hub, and Amazon Macie.

Understanding how to configure these services, interpret their findings, create meaningful alerts, and build dashboards for visibility is essential for both the exam and real-world security operations. This section provides comprehensive coverage of each service, including architecture, configuration, integration patterns, and exam-focused scenarios.

Amazon GuardDuty

Amazon GuardDuty is a managed threat detection service that continuously monitors your AWS accounts and workloads for malicious activity and unauthorized behavior. It uses machine learning, anomaly detection, and integrated threat intelligence to identify potential threats without requiring you to deploy any additional infrastructure.

GuardDuty operates as a regional service, meaning you must enable it in each AWS Region where you want threat detection. However, it supports multi-account and multi-region aggregation through a delegated administrator account.

How GuardDuty Works

GuardDuty analyzes billions of events across multiple AWS data sources to detect threats. The service establishes a baseline of normal activity for your account and then identifies deviations that may indicate malicious behavior. GuardDuty does not store or process your actual data; it only analyzes metadata and logs.

The detection engine combines three approaches:

- **Signature-based detection** – Matches known malicious IP addresses, domains, and patterns
- **Machine learning** – Identifies anomalies in API calls, network traffic, and user behavior
- **Threat intelligence** – Integrates feeds from AWS, CrowdStrike, and Proofpoint

Data Sources Analyzed by GuardDuty

GuardDuty independently pulls data from the following sources without requiring you to enable or configure them separately:

- **AWS CloudTrail Management Events** – Captures all API calls made to AWS services, including who made the call, when, from where, and what action was taken. GuardDuty analyzes these events to detect unauthorized API activity, credential compromise, and reconnaissance.
- **AWS CloudTrail S3 Data Events** – Monitors S3 object-level API operations such as GetObject, PutObject, and DeleteObject. This helps detect data exfiltration attempts and unauthorized access to sensitive buckets.
- **VPC Flow Logs** – Provides metadata about IP traffic going to and from network interfaces in your VPC. GuardDuty uses this to detect port scanning, DDoS attacks, and communication with known malicious IPs.

- **DNS Logs** – Captures DNS queries made by EC2 instances through Route 53 Resolver. GuardDuty analyzes these to detect communication with command-and-control servers and malicious domains.
- **EKS Audit Logs** – Monitors Kubernetes API server audit logs to detect suspicious activity in EKS clusters, such as unauthorized pod deployments or privilege escalation attempts.
- **RDS Login Activity** – Analyzes authentication events for RDS databases to detect brute force attacks, anomalous login patterns, and potentially compromised database credentials.
- **Lambda Network Activity** – Monitors network connections initiated by Lambda functions to detect functions communicating with malicious endpoints or exhibiting unusual network behavior.
- **EBS Malware Protection** – Scans EBS volumes attached to EC2 instances for the presence of malware, trojans, and other malicious files. This feature must be explicitly enabled and incurs additional costs.
- **Runtime Monitoring** – Provides visibility into OS-level and container-level activity for EC2 instances, ECS tasks, and EKS pods. Requires deploying a GuardDuty security agent.

GuardDuty Protection Plans

GuardDuty organizes its detection capabilities into protection plans that can be enabled or disabled independently:

- **S3 Protection** – Monitors CloudTrail S3 data events to detect suspicious access patterns
- **EKS Protection** – Includes EKS Audit Log Monitoring and EKS Runtime Monitoring
- **Malware Protection** – Scans EBS volumes for malware when triggered by findings
- **RDS Protection** – Monitors RDS and Aurora login activity
- **Lambda Protection** – Monitors Lambda function network activity
- **Runtime Monitoring** – Provides deep visibility into workload behavior

GuardDuty Finding Types

GuardDuty generates findings when it detects potential security issues. Each finding includes the finding type, severity, affected resources, and recommended remediation. Finding types are named using the format: ThreatPurpose:ResourceType/ThreatName.

Backdoor Findings:

- Backdoor:EC2/C&CActivity.B – EC2 instance querying a known command-and-control server
- Backdoor:EC2/C&CActivity.B!DNS – EC2 instance querying a C&C domain via DNS
- Backdoor:EC2/DenialOfService.Tcp – EC2 instance participating in a TCP DDoS attack
- Backdoor:EC2/DenialOfService.Udp – EC2 instance participating in a UDP DDoS attack
- Backdoor:EC2/Spambot – EC2 instance sending spam emails on port 25
- Backdoor:Lambda/C&CActivity.B – Lambda function communicating with C&C server
- Backdoor:Runtime/C&CActivity.B – Runtime activity indicating C&C communication

CryptoCurrency Findings:

- CryptoCurrency:EC2/BitcoinTool.B – EC2 instance communicating with Bitcoin-related IPs
- CryptoCurrency:EC2/BitcoinTool.B!DNS – EC2 instance querying Bitcoin mining pool domains
- CryptoCurrency:Lambda/BitcoinTool.B – Lambda function associated with cryptocurrency activity
- CryptoCurrency:Runtime/BitcoinTool.B – Container or instance runtime mining cryptocurrency

PenTest Findings:

- PenTest:IAMUser/KaliLinux – API calls from a Kali Linux machine
- PenTest:IAMUser/ParrotLinux – API calls from a Parrot Security OS machine
- PenTest:IAMUser/PentooLinux – API calls from a Pentoo Linux machine
- PenTest:S3/KaliLinux – S3 API calls from Kali Linux

Persistence Findings:

- Persistence:IAMUser/AnomalousBehavior – Unusual API calls to establish persistence
- Persistence:IAMUser/NetworkPermissions – User modifying network permissions unusually
- Persistence:IAMUser/ResourcePermissions – User modifying resource permissions unusually
- Persistence:IAMUser/UserPermissions – User modifying IAM permissions unusually

Policy Findings:

- Policy:IAMUser/RootCredentialUsage – Root account credentials were used
- Policy:S3/AccountBlockPublicAccessDisabled – Account-level S3 Block Public Access disabled
- Policy:S3/BucketAnonymousAccessGranted – Bucket policy grants anonymous access
- Policy:S3/BucketBlockPublicAccessDisabled – Bucket-level Block Public Access disabled
- Policy:S3/BucketPublicAccessGranted – Bucket made publicly accessible

PrivilegeEscalation Findings:

- PrivilegeEscalation:IAMUser/AdministrativePermissions – User granted admin permissions anomalously
- PrivilegeEscalation:Runtime/DockerSocketAccessed – Process accessed Docker socket in container
- PrivilegeEscalation:Runtime/RuncContainerEscape – Potential container escape detected
- PrivilegeEscalation:Runtime/UserfaultfdUsage – Userfaultfd syscall used (potential exploit)

Recon Findings:

- Recon:EC2/PortProbeEMRUnprotectedPort – EMR cluster probed on unprotected port
- Recon:EC2/PortProbeUnprotectedPort – EC2 instance probed on an unprotected port
- Recon:EC2/Portscan – EC2 instance performing outbound port scans
- Recon:IAMUser/MaliciousIPCaller – API calls from known malicious IP
- Recon:IAMUser/MaliciousIPCaller.Custom – API calls from IP on custom threat list
- Recon:IAMUser/NetworkPermissions – User performing reconnaissance on network permissions

- Recon:IAMUser/ResourcePermissions – User enumerating resource permissions
- Recon:IAMUser/TorIPCaller – API calls from Tor exit node
- Recon:IAMUser/UserPermissions – User enumerating IAM permissions

Stealth Findings:

- Stealth:IAMUser/CloudTrailLoggingDisabled – CloudTrail logging was disabled
- Stealth:IAMUser/LoggingConfigurationModified – Logging configuration was modified
- Stealth:IAMUser/PasswordPolicyChanged – Account password policy was weakened
- Stealth:S3/ServerAccessLoggingDisabled – S3 server access logging was disabled

Trojan Findings:

- Trojan:EC2/BlackholeTraffic – EC2 instance communicating with black hole IP
- Trojan:EC2/BlackholeTraffic!DNS – EC2 instance querying black hole domain
- Trojan:EC2/DGADomainRequest.B – EC2 instance querying algorithmically generated domain
- Trojan:EC2/DGADomainRequest.C!DNS – EC2 instance querying DGA domain via DNS
- Trojan:EC2/DNSDataExfiltration – Data exfiltration through DNS queries
- Trojan:EC2/DriveBySourceTraffic!DNS – EC2 instance querying drive-by download domain
- Trojan:EC2/DropPoint – EC2 instance communicating with known drop point
- Trojan:EC2/DropPoint!DNS – EC2 instance querying known drop point domain
- Trojan:EC2/PhishingDomainRequest!DNS – EC2 instance querying phishing domain
- Trojan:Lambda/BlackholeTraffic – Lambda function communicating with black hole
- Trojan:Lambda/DropPoint – Lambda function communicating with drop point
- Trojan:Runtime/BlackholeTraffic – Runtime activity communicating with black hole
- Trojan:Runtime/DropPoint – Runtime activity communicating with drop point

UnauthorizedAccess Findings:

- UnauthorizedAccess:EC2/MaliciousIPCaller.Custom – EC2 instance receiving connections from custom threat list IP
- UnauthorizedAccess:EC2/MetadataDNSRebind – EC2 instance performing DNS rebind to metadata service
- UnauthorizedAccess:EC2/RDPBruteForce – RDP brute force attack detected
- UnauthorizedAccess:EC2/SSHBruteForce – SSH brute force attack detected
- UnauthorizedAccess:EC2/TorClient – EC2 instance connecting to Tor network
- UnauthorizedAccess:EC2/TorRelay – EC2 instance acting as Tor relay
- UnauthorizedAccess:IAMUser/ConsoleLogin – Unusual console login detected
- UnauthorizedAccess:IAMUser/ConsoleLoginSuccess.B – Successful console login from unusual location
- UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration.InsideAWS – Instance credentials used from another AWS account

- UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration.OutsideAWS – Instance credentials used from outside AWS
- UnauthorizedAccess:IAMUser/MaliciousIPCaller – API calls from known malicious IP
- UnauthorizedAccess:IAMUser/MaliciousIPCaller.Custom – API calls from custom threat list IP
- UnauthorizedAccess:IAMUser/TorIPCaller – API calls from Tor exit node
- UnauthorizedAccess:Lambda/MaliciousIPCaller – Lambda invoked from malicious IP
- UnauthorizedAccess:Lambda/TorClient – Lambda function connecting to Tor
- UnauthorizedAccess:Lambda/TorRelay – Lambda function acting as Tor relay
- UnauthorizedAccess:S3/MaliciousIPCaller.Custom – S3 accessed from custom threat list IP
- UnauthorizedAccess:S3/TorIPCaller – S3 accessed from Tor exit node

Impact Findings:

- Impact:EC2/AbusedDomainRequest.Reputation – EC2 instance querying low-reputation domain
- Impact:EC2/BitcoinDomainRequest.Reputation – EC2 instance querying Bitcoin-related domain
- Impact:EC2/MaliciousDomainRequest.Reputation – EC2 instance querying malicious domain
- Impact:EC2/PortSweep – EC2 instance probing ports on many hosts
- Impact:EC2/SuspiciousDomainRequest.Reputation – EC2 instance querying suspicious domain
- Impact:EC2/WinRMBruteForce – WinRM brute force attack detected
- Impact:Runtime/AbusedDomainRequest.Reputation – Runtime activity with abused domain
- Impact:Runtime/BitcoinDomainRequest.Reputation – Runtime activity with Bitcoin domain
- Impact:Runtime/CryptoMinerExecuted – Cryptocurrency miner executed
- Impact:Runtime/MaliciousDomainRequest.Reputation – Runtime activity with malicious domain
- Impact:Runtime/SuspiciousDomainRequest.Reputation – Runtime activity with suspicious domain

Execution Findings:

- Execution:EC2/MaliciousFile – Malicious file detected on EC2 instance
- Execution:EC2/SuspiciousFile – Suspicious file detected on EC2 instance
- Execution:ECS/MaliciousFile – Malicious file detected in ECS container
- Execution:ECS/SuspiciousFile – Suspicious file detected in ECS container
- Execution:Kubernetes/MaliciousFile – Malicious file detected in Kubernetes pod
- Execution:Kubernetes/SuspiciousFile – Suspicious file detected in Kubernetes pod
- Execution:Runtime/MaliciousFileExecuted – Malicious file was executed
- Execution:Runtime/NewBinaryExecuted – New binary executed in container
- Execution:Runtime/NewLibraryLoaded – New library loaded in container
- Execution:Runtime/ReverseShell – Reverse shell detected
- Execution:Runtime/SuspiciousCommand – Suspicious command executed
- Execution:Runtime/SuspiciousTool – Suspicious tool executed

Discovery Findings:

- Discovery:Kubernetes/MaliciousIPCaller – Kubernetes API called from malicious IP
- Discovery:Kubernetes/MaliciousIPCaller.Custom – Kubernetes API called from custom threat list
- Discovery:Kubernetes/SuccessfulAnonymousAccess – Successful anonymous Kubernetes API access
- Discovery:Kubernetes/TorIPCaller – Kubernetes API called from Tor
- Discovery:S3/AnomalousBehavior – Anomalous S3 API activity detected
- Discovery:S3/MaliciousIPCaller – S3 accessed from malicious IP
- Discovery:S3/MaliciousIPCaller.Custom – S3 accessed from custom threat list
- Discovery:S3/TorIPCaller – S3 accessed from Tor

DefenseEvasion Findings:

- DefenseEvasion:Runtime/FilelessExecution – Fileless malware execution detected
- DefenseEvasion:Runtime/ProcessInjection.Proc – Process injection via /proc detected
- DefenseEvasion:Runtime/ProcessInjection.Ptrace – Process injection via ptrace detected
- DefenseEvasion:Runtime/ProcessInjection.VirtualMemoryWrite – Process injection via memory write
- DefenseEvasion:Runtime/SuspiciousCommand – Suspicious command for defense evasion

GuardDuty Severity Levels

GuardDuty assigns a severity level to each finding on a scale of 0.1 to 8.9. The severity helps you prioritize which findings to investigate first:

- **Low (0.1 - 3.9)** – Suspicious or potentially malicious activity that was blocked or hasn't yet compromised your resources. Examples include failed brute force attempts or reconnaissance activity that didn't result in access.
- **Medium (4.0 - 6.9)** – Suspicious activity that deviates from normally observed behavior and may indicate compromised resources. Examples include unusual API activity, unexpected data access patterns, or communication with suspicious domains.
- **High (7.0 - 8.9)** – Resource is actively compromised and being used for malicious purposes. Examples include cryptocurrency mining, data exfiltration, EC2 instances acting as part of a botnet, or active communication with command-and-control servers.

Configuring GuardDuty Alerts

GuardDuty automatically publishes all findings to Amazon EventBridge, enabling you to create automated responses and notifications. You can create EventBridge rules to filter findings based on various criteria and route them to different targets.

Common EventBridge Rule Patterns:

Filter by severity (high-severity only):

json

```
{
  "source": ["aws.guardduty"],
  "detail-type": ["GuardDuty Finding"],
  "detail": {
    "severity": [{ "numeric": [">=", 7] }]
  }
}
```

Filter by finding type:

```
json
{
  "source": ["aws.guardduty"],
  "detail-type": ["GuardDuty Finding"],
  "detail": {
    "type": [{ "prefix": "UnauthorizedAccess:" }]
  }
}
```

Filter by resource type:

```
json
{
  "source": ["aws.guardduty"],
  "detail-type": ["GuardDuty Finding"],
  "detail": {
    "resource": {
      "resourceType": ["Instance"]
    }
  }
}
```

Common Alert Destinations:

1. **Amazon SNS** – Send email or SMS notifications to security teams for immediate awareness. Create separate SNS topics for different severity levels.
2. **AWS Lambda** – Trigger automated remediation functions such as isolating compromised instances, revoking IAM credentials, or blocking malicious IPs in security groups.
3. **AWS Security Hub** – Aggregate GuardDuty findings with other security services for centralized visibility. This integration is automatic when both services are enabled.

4. **Amazon S3** – Archive findings for long-term storage, compliance, and forensic analysis. Use S3 Lifecycle policies to manage retention.
5. **AWS Step Functions** – Orchestrate complex remediation workflows that require multiple steps or human approval.
6. **Third-party SIEM solutions** – Forward findings to Splunk, Sumo Logic, Datadog, or other security tools via Lambda or Kinesis Data Firehose.

Example Lambda Function for Auto-Remediation:

A common use case is automatically isolating an EC2 instance when GuardDuty detects high-severity malicious activity:

1. EventBridge rule triggers Lambda on high-severity EC2 finding
2. Lambda function retrieves instance ID from the finding
3. Lambda creates a snapshot of the instance for forensic analysis
4. Lambda modifies security groups to isolate the instance (remove all ingress/egress except for forensic access)
5. Lambda sends notification to security team via SNS
6. Lambda creates a ticket in your incident management system

GuardDuty Multi-Account Management

For organizations with multiple AWS accounts, GuardDuty supports centralized management through two methods:

Method 1: AWS Organizations Integration (Recommended)

1. Enable trusted access for GuardDuty in AWS Organizations
2. Designate a delegated administrator account (not the management account)
3. The delegated administrator can enable GuardDuty for all member accounts
4. Enable auto-enable to automatically protect new accounts
5. All findings are visible in the delegated administrator account

Method 2: Invitation-Based

1. The administrator account invites member accounts
2. Member accounts accept the invitation
3. Findings from member accounts flow to the administrator
4. Less scalable than Organizations integration

Benefits of Centralized Management:

- View and manage findings from all accounts in one location

- Apply suppression rules across all accounts
- Manage threat lists and trusted IP lists centrally
- Ensure consistent protection across the organization
- Simplified compliance reporting

GuardDuty Suppression Rules

Suppression rules allow you to filter out findings that you've determined to be false positives or expected behavior. Suppressed findings are still generated but marked as archived.

Creating Suppression Rules:

1. Navigate to GuardDuty console → Settings → Suppression rules
2. Define filter criteria (finding type, severity, resource, etc.)
3. Provide a name and description for the rule
4. The rule applies to future findings matching the criteria

Common Suppression Scenarios:

- Penetration testing activities from known IP ranges
- Expected port scanning from security scanning tools
- API calls from authorized third-party security tools
- Known cryptocurrency-related activity for legitimate blockchain applications

References:
https://docs.aws.amazon.com/guardduty/latest/ug/what-is-guardduty.html
https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_finding-types-active.html
https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_findings_cloudwatch.html
https://docs.aws.amazon.com/guardduty/latest/ug/guardduty-accounts.html

Amazon Security Lake

Amazon Security Lake is a fully managed security data lake service that automatically centralizes security data from AWS services, SaaS providers, on-premises environments, and third-party sources into a purpose-built data lake stored in your AWS account.

Key Features:

- Centralizes security logs from multiple sources
- Automatically normalizes data to Open Cybersecurity Schema Framework (OCSF) format
- Stores data in S3 with optimized Parquet format
- Supports configurable retention and storage tiers

- Enables querying with Amazon Athena and integration with analytics tools

Supported AWS Log Sources:

- AWS CloudTrail (management and data events)
- Amazon VPC Flow Logs
- Amazon Route 53 Resolver query logs
- AWS Security Hub findings
- Amazon EKS audit logs
- AWS WAF logs

OCSF (Open Cybersecurity Schema Framework):

Security Lake normalizes all ingested data to OCSF, an open standard that:

- Provides a common schema for security events
- Enables correlation across different data sources
- Simplifies analytics and threat detection
- Supports interoperability with third-party security tools

Configuring Security Lake:

1. Enable Security Lake in your AWS account (creates an S3 bucket)
2. Select AWS log sources to collect
3. Configure rollup Regions for cross-region aggregation
4. Add subscriber access for analytics tools or third-party SIEMs
5. Set retention policies and storage class transitions

Subscriber Types:

- **Data Access Subscribers** – Query data directly using Athena or other tools
- **Query Access Subscribers** – Access via Lake Formation with granular permissions

Integration with Analytics:

- Amazon Athena for SQL queries
- Amazon OpenSearch Service for search and visualization
- Amazon QuickSight for dashboards
- Third-party SIEM tools (Splunk, IBM QRadar, Datadog)
- Amazon SageMaker for machine learning on security data

References:

https://docs.aws.amazon.com/security-lake/latest/userguide/what-is-security-lake.html
https://docs.aws.amazon.com/security-lake/latest/userguide/open-cybersecurity-schema-framework.html

AWS User Notifications

AWS User Notifications is a managed service that centralizes notifications from supported AWS services, allowing users to view and manage alerts in the AWS Console, mobile app, or through configured delivery channels.

Key Features:

- Centralized notification management across supported AWS services
- Delivery through email, mobile push notifications, and chat channels (e.g., Slack, Microsoft Teams)
- User-level notification preferences and filtering
- Integrates with services such as AWS Security Hub, Amazon GuardDuty, and Amazon Inspector

Supported Sources:

- AWS Security Hub findings
- Amazon GuardDuty alerts
- Amazon Inspector vulnerability findings
- AWS Health events

Delivery Channels:

- Email
- AWS Console Bell Notifications
- AWS Mobile App push
- Chat channels (Slack, Microsoft Teams)

References:

https://docs.aws.amazon.com/notifications/latest/userguide/what-is-service.html
https://docs.aws.amazon.com/notifications/latest/userguide/managing-org-notifications.html

Integration and Best Practices

How These Services Work Together:

- **GuardDuty** detects threats and generates findings
- **Macie** discovers sensitive data exposure in S3
- **Security Hub** aggregates findings from both services (plus others) into a unified dashboard
- **Security Lake** centralizes raw logs for advanced analytics and long-term retention

Best Practices for Detection:

- Enable GuardDuty in all accounts and regions
- Configure Macie discovery jobs for buckets containing sensitive data
- Enable Security Hub with relevant security standards
- Use Security Lake to centralize logs for correlation and hunting
- Create EventBridge rules for automated alerting on high-severity findings
- Establish custom Security Hub insights for your organization's priorities
- Integrate with your existing SIEM or SOAR platforms

CloudWatch Metrics and Dashboards:

Create CloudWatch dashboards to visualize security metrics:

- GuardDuty finding counts by severity
- Macie sensitive data discovery trends
- Security Hub compliance scores over time
- Security Lake data ingestion volumes

> **EXAM TIP:** For the exam, understand which service to use for each scenario: GuardDuty for threat detection, Macie for S3 sensitive data, Security Hub for aggregation and compliance, and Security Lake for centralized log storage and analytics.

References:
https://docs.aws.amazon.com/prescriptive-guidance/latest/security-reference-architecture/welcome.html
https://aws.amazon.com/blogs/security/

Domain 2: Incident Response

Overview

The second domain of the AWS Certified Security Specialty exam focuses on the topics related to logging and monitoring management in AWS. To be an effective AWS Security Specialist, it is important that you understand these key concepts. Roughly 14% of questions in the actual Security Specialty exam revolve around the topics of logging and monitoring in the AWS platform.

This is the second biggest domain in the exam, which is tied up with Exam Domain 5: Data Protection, which also has an 18% coverage. Therefore, you must allocate ample time learning the various AWS services, features and concepts in this knowledge area.

This domain will challenge your know-how in doing the following:

- Analyzing architectures to identify monitoring requirements and sources of data for security monitoring
- Analyzing environments and workloads to determine monitoring requirements
- Designing environment monitoring and workload monitoring based on business and security requirements
- Setting up automated tools and scripts to perform regular audits (for example, by creating custom insights in Security Hub)
- Defining the metrics and thresholds that generate alerts
- Analyzing the service functionality, permissions, and configuration of resources after an event that did not provide visibility or alerting
- Analyzing and remediating the configuration of a custom application that is not reporting its statistics
- Evaluating logging and monitoring services for alignment with security requirements
- Configuring logging for services and applications
- Identifying logging requirements and sources for log ingestion
- Implementing log storage and lifecycle management according to AWS best practices and organizational requirements
- Identifying misconfiguration and determining remediation steps for absent access permissions that are necessary for logging (for example, by managing read/write permissions, S3 bucket permissions, public access, and integrity)
- Determining the cause of missing logs and performing remediation steps
- Identifying patterns in logs to indicate anomalies and known threats
- Normalizing, parsing, and correlating logs

In this chapter, we will cover all of the related topics for logging and monitoring management in AWS that will likely show up in your Security Specialty exam.

Logging and Monitoring Services in AWS

Related but not similar — logging and monitoring are techniques implemented to achieve a common goal. They work hand in hand to ensure the system's performance baselines and security guidelines are always met.

Logging refers to the recording of events in files. Logs contain low-level details that can give you complete visibility of how your application or system performs under certain circumstances. From a security standpoint, logging helps security admins identify red flags easily overlooked in their system.

Monitoring is the process of analyzing and collecting metrics to ensure optimal performance is always in effect, unauthorized access is detected, and your services' usage is always aligned with organizational security policies.

Amazon CloudWatch is the primary logging and monitoring solution in AWS. It gives you system-wide visibility to your AWS resources and applications by monitoring *aws built-in* and *custom metrics*. These metrics give you insights into the operational health of your resources and how they are being used. You can create alarms, send notifications, or automatically make changes to the resources you are monitoring when a threshold is breached.

Basic CloudWatch Concepts:

- **CloudWatch Dashboard**
    - You can create customized dashboards to visualize and monitor the metric information of your resources.
    - All dashboards are **global**, not region-specific.
    - You can add, remove, resize, move, edit, or rename a graph. You can plot metrics manually in a graph.

- **Amazon EventBridge (Amazon CloudWatch Events)**
    - Deliver a near real-time stream of system events that describe changes in AWS resources.
    - Events respond to these operational changes and take corrective action as necessary by sending messages to respond to the environment, activating functions, making changes, and capturing state information.
    - Concepts

        - **Events** - indicates a change in your AWS environment.
        - **Targets** - processes events.
        - **Rules** - matches incoming events and routes them to targets for processing.

- **CloudWatch Logs**
    - Features
        - Monitor logs from EC2 instances in real-time
        - Monitor CloudTrail logged events
        - By default, logs are kept indefinitely and never expire
        - Archive log data
        - Log Route 53 DNS queries
    - **CloudWatch Logs Insights** enables you to interactively search and analyze your log data in CloudWatch Logs using queries.
    - After the CloudWatch Logs agent begins publishing log data to Amazon CloudWatch, you can search and filter the log data by creating one or more metric filters. **Metric filters** define the terms and patterns to look for in log data as it is sent to CloudWatch Logs.
- **CloudWatch Agent**
    - Collect more logs and system-level metrics (e.g., memory consumption, number of TCP connections) from EC2 instances and your on-premises servers.
    - Needs to be installed.
- **Authentication and Access Control**
    - Use IAM users or roles for authenticating who can access
    - Use Dashboard Permissions, IAM identity-based policies, and service-linked roles for managing access control.
    - A *permissions policy* describes who has access to what.
        - Identity-Based Policies
        - Resource-Based Policies

- There are no CloudWatch Amazon Resource Names (ARNs) for you to use in an IAM policy. Use an * (asterisk) instead as the resource when writing a policy to control access to CloudWatch actions.

Troubleshooting

- If your EC2 instance is not sending or stops sending logs, you can log in to that instance and check if the agent is still running.
- The CloudWatch agent sends data to the CloudWatch logs over the public Internet by default. When using CloudWatch agent, verify if your EC2 instance has route access to the Internet gateway/NAT gateway.
- Make sure that the IAM permissions used by the CloudWatch Logs agent allow putting log events as well as creating log groups and log streams in CloudWatch.
- If you don't see any CloudWatch logs after executing a Lambda Function, make sure its execution role has permission to write log data to CloudWatch Logs.

References:
https://aws.amazon.com/cloudwatch/faqs/
https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/troubleshooting-CloudWatch-Agent.html

AWS CloudTrail

AWS CloudTrail is also a monitoring tool. But instead of AWS resources, it records API activities that occur in your AWS account. CloudTrail delivers one free copy of management event logs for each AWS region.

CloudTrail Concepts

Events is the record of activity in an AWS account. This activity can be an action taken by a user, role, or service that is monitorable by CloudTrail.

There are three types of events that you can record:

- **Management events (***control plane operations***)** - events that include management operations that have occurred within your AWS account, such as user logins. Management events are enabled by default
- **Data events (***data plane operations)* - events that include resource operations performed on or within the resource itself, such as *S3 object-level API* activity or *Lambda function execution activity.* Logging data events are charged and, therefore, disabled by default.
- **Insights events** - events that include unusual activity and user behavior in your account.

Log Analysis of CloudTrail Logs

Event History

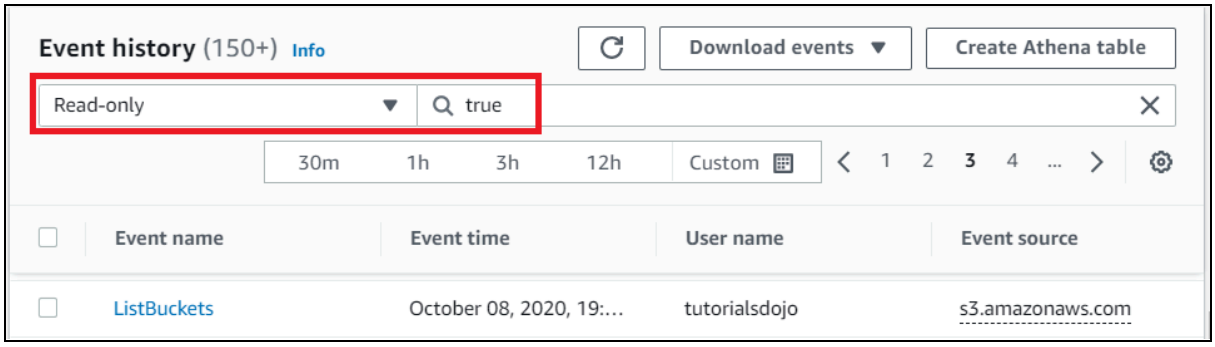
Even without creating a trail, you can still view the history of API activities in your account. You can view events in **Event History** to gain immediate visibility to API activities that occur in your AWS account in the past 90 days. Searching and downloading event records is also possible.

If you have a high API activity in a day, the number of CloudTrail events can be overwhelming. It would be difficult to look for specific events that may be helpful in troubleshooting a security incident.

In that case, you may use the event filter definitions to only view particular events (in Event history) that you're interested in:

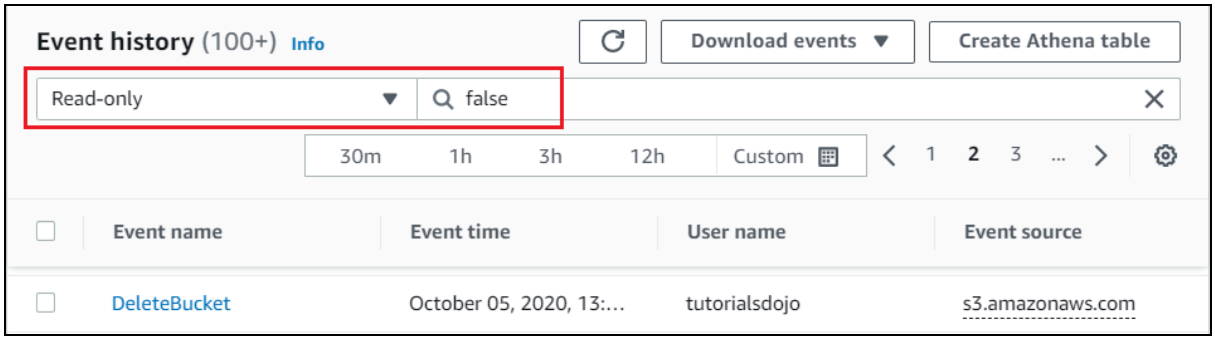
- **Read-only events** refer to API operations that read resources. Read operations are operations that don't cause modifications to a resource. Examples of these events are Amazon S3 *ListBuckets* and Amazon CloudWatch Logs *DescribeMetricsFilters* API actions.

In the CloudTrail console, set the value of Read-only events to "true" to view Read-only events.



- **Write-only events** refer to API operations that modify or may modify resources in your account. Examples are Amazon S3 *DeleteBucket* and Amazon EC2 *TerminateInstances* API operations.

To enable write-only events, set the value of Read-only to "false".



Working with AWS CloudTrail Lake

AWS CloudTrail Lake is a fully managed, immutable, security-focused data lake designed specifically for storing, querying, and analyzing CloudTrail and other activity events. It automatically ingests events into event data stores with a standardized schema and allows SQL-based querying directly within the CloudTrail console or via the CloudTrail Lake API. CloudTrail Lake simplifies security investigations by eliminating the need for custom ETL, external log analysis tools, or manual indexing. It also supports ingesting CloudTrail events, AWS service activity logs, and even custom application events, providing a centralized source for auditing and incident response. Retention can be configured from 7 days to 7 years, and AWS manages durability, scalability, and integrity of the stored events.

CloudWatch Logs

CloudTrail sends logs to an S3 bucket of your choice by default. However, you can configure a trail to send logs to CloudWatch Logs to monitor log data. CloudTrail events that are sent to CloudWatch Logs can trigger alarms according to the metric filters you define.

Amazon Athena

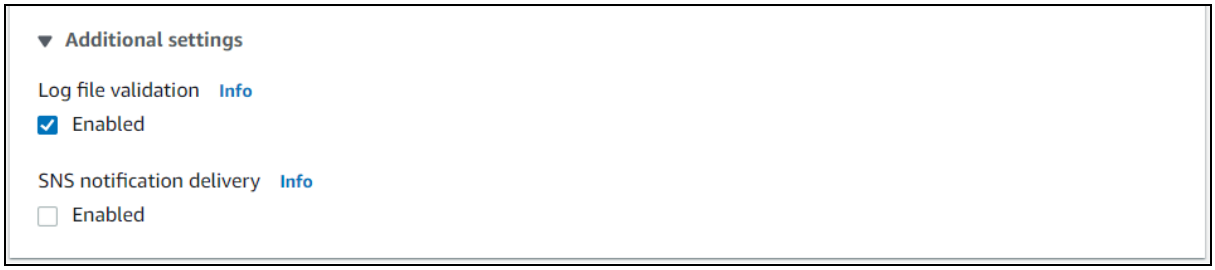
Amazon Athena is an interactive query service that makes it easy for you to analyze data stored in an S3 bucket using SQL commands. Since CloudTrail keeps its logs to an S3 bucket, we can leverage Athena to perform analysis on our CloudTrail logs. You no longer have to download the log files on a machine and analyze them through a third-party log analysis tool. Athena is serverless and is very easy to set up.

Log File Validation

File integrity validation is an important step in fortifying your security posture. It protects your system against data tampering or the altering/modifying/editing of data without authorization. Without file validation, you're inviting possible intrusions that could cause serious damage to your business/application.

CloudTrail makes it easy for us to protect the authenticity of log files with its feature: Log File Validation. CloudTrail uses SHA-256 for hashing and SHA-256 with RSA for digital signing, making data tampering virtually impossible (even with modern computers) to execute.

Log File Validation is enabled by default when you create a CloudTrail trail.



References:
https://aws.amazon.com/cloudtrail/faqs/
https://docs.aws.amazon.com/awscloudtrail/latest/userguide/logging-management-events-with-cloudtrail.html
https://docs.aws.amazon.com/awscloudtrail/latest/userguide/monitor-cloudtrail-log-files-with-cloudwatch-logs.html
https://docs.aws.amazon.com/athena/latest/ug/cloudtrail-logs.html

Central Logging Using AWS CloudTrail

Imagine auditing different accounts that have resources in different AWS regions. Auditing API activities in CloudTrail would be time-consuming and complex. One would have to log into each account for each region, who knows how many times. This approach doesn't really scale well, especially in big organizations. As it turns out, we can configure CloudTrail to enable us to analyze log files from a central location.

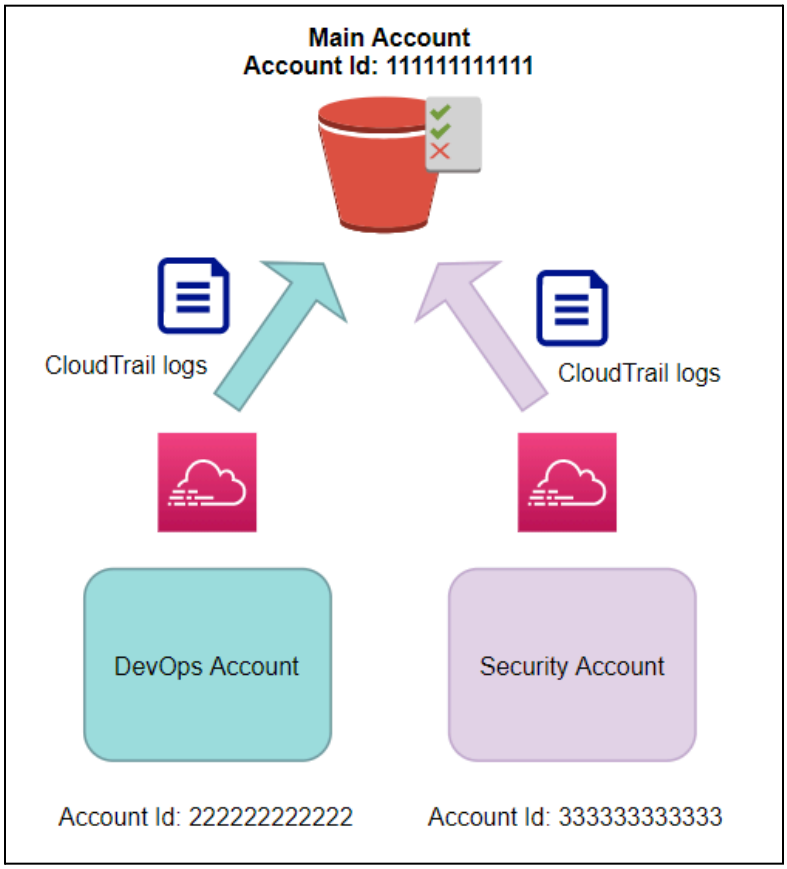
Multi-Region Trail

You can aggregate all log files from different regions into a single bucket using the multi-region trail feature. This feature is automatically enabled when you create a trail on the CloudTrail console. You can also change an existing single-region trail into multi-region by appending the `--is-multi-region-trail` parameter in the **`update-trail`** command.

```
aws cloudtrail update-trail --name main-trail --is-multi-region-trail
```

Multiple Accounts

Aside from multi-region, CloudTrail is also capable of accepting log files in a single S3 bucket from different accounts. Let's say you have three accounts: *main account, DevOps, and Security*. Each account has CloudTrail enabled, which generates a considerable amount of API event logs. Instead of analyzing log files separately, you can deliver the DevOps and Security account logs to the main account's bucket. This way, log analysis would be more manageable.



To enable logging from different accounts, you have to modify the default bucket policy that is created along with your trail (in the main account) upon its creation. As you can see on the sample bucket policy below, you must grant access to the DevOps and Security account by adding another S3 resource (it should be the same S3 bucket that is used by the main account) with their respective account IDs. Afterwhich, head over to the DevOps account and re-configure its trail by selecting the main account's bucket for storing log files. Do the same with the Security Account.

```
{
      "Version": "2012-10-17",
      "Statement": [{
                  "Sid": "AWSCloudTrailAclCheck20150319",
                  "Effect": "Allow",
                  "Principal": {
                        "Service": "cloudtrail.amazonaws.com"
                  },
                  "Action": "s3:GetBucketAcl",
                  "Resource": "arn:aws:s3:::main-cloudtrail-dojo"
            },
            {
                  "Sid": "AWSCloudTrailWrite20150319",
                  "Effect": "Allow",
                  "Principal": {
                        "Service": "cloudtrail.amazonaws.com"
                  },
                  "Action": "s3:PutObject",
                  "Resource":
["arn:aws:s3:::main-cloudtrail-tdojo/AWSLogs/111111111111/*",
 "arn:aws:s3:::main-cloudtrail-tdojo/AWSLogs/222222222222/*",
 "arn:aws:s3:::main-cloudtrail-tdojo/AWSLogs/333333333333/*"
                  ],
                  "Condition": {
                        "StringEquals": {
                              "s3:x-amz-acl": "bucket-owner-full-control"
                        }
                  }
            }
      ]
}
```

References:
https://docs.aws.amazon.com/awscloudtrail/latest/userguide/receive-cloudtrail-log-files-from-multiple-regions.html
https://docs.aws.amazon.com/awscloudtrail/latest/userguide/create-s3-bucket-policy-for-cloudtrail.html

Amazon CloudWatch Logs Agent Troubleshooting

Amazon CloudWatch Logs Agent (*awslogs agent*) is a service you can use to push log files from your EC2 instance to CloudWatch Logs. Using the awslogs agent is the older way of doing things. AWS recommends using the newer CloudWatch agent (unified agent), which is the successor of awslogs that offers more functionalities. Despite being old-fashioned, you still need to have an idea about the awslogs agent as it may appear on the exam.

You have to install the agent to your host before you can begin collecting log data. Since AWS has no control over your operating system, having basic knowledge of troubleshooting issues that might occur is important.

Coming into the exam, most of the problems regarding troubleshooting that you'll encounter will involve three things: **Misconfiguration, Insufficient permissions, and Connection problems**. So even if you can't exactly pinpoint the answer to an item, you would still be able to eliminate distractors as long as you remember these three things.

Misconfiguration

- Open the awslogs log file on ***/var/log/awslogs.log.*** Check for the following errors:
    - **NocredentialsError**
        - Make sure you attach an IAM role to your EC2 instance.
        - Alternatively, you can update the IAM user credentials in the */etc/awslogs/awscli.conf* file.
    - **AccessDeniedError**
        - Ensure that you have the right permissions for CloudWatch Logs.
- Check if your OS log rotation rules are supported.
- Check for duplicates in the *[logstream]* section of the agent configuration file.

Insufficient Permissions

- Check if you have the required permissions for the instance's IAM role:
    - `logs:CreateLogGroup` - creates a log group that contains the log stream.
    - `logs:CreateLogStream` - creates a log stream. The log stream is the sequence of log events generated from a resource.

- `logs:PutLogEvents` - uploads a batch of log events to the log stream.
- `logs:DescribeLogStreams` - this operation lists all the log streams for a particular log group.

Connection Problems

- Check your security group and network access control list's configuration and verify if it has access to the public Internet.

References:
https://aws.amazon.com/premiumsupport/knowledge-center/push-log-data-cloudwatch-awslogs/
https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/UsePreviousCloudWatchLogsAgent.html

Central Log Collection using CloudWatch Logs

In an on-premises setup, keeping log files in one place runs the risk of losing them. AWS helps alleviate this risk. Instead of having log data stored on your local disk, you can ship them to a cloud-based storage solution, which is far more durable and scalable. In CloudWatch Logs, you don't have to worry about setting up and maintaining an EBS or an S3 bucket to achieve this, as it uses Amazon S3 as its storage internally.

CloudWatch Logs enables you to collect log data from multiple sources and access them in a central location. The log files are not dumped as it is. They are systematically organized using CloudWatch's grouping convention. CloudWatch Logs uses the concept of *log groups* and *log streams* to compartmentalize log files conveniently. You choose the retention policy of your log data. By default, the retention setting is set to "never expire." But you can expire log data in 24 hours, up to 10 years.

Effective Application Logging using CloudWatch Logs:

- **Unified CloudWatch Agent**
    - The Unified CloudWatch agent is basically a program for collecting system-level metrics and logs from your EC2 instances and on-premises servers. The code for the CloudWatch agent is open-source and under MIT license.
    - AWS recommends using the Unified CloudWatch agent over awslogs agent or traditional shell scripts.
    - If you're hosting the agent on an on-premises server, you must configure that server to use an IAM user with programmatic access (access key & secret access). Make sure your IAM user has the correct CloudWatch Logs permissions.
    - Unified CloudWatch agents can retrieve custom metrics from your applications using the *StatsD* and *collectd* protocols.
    - You need to have permission to use the ***cloudwatch:putMetricData*** operation to publish metrics to Amazon CloudWatch.
- **Using CloudWatch Logs to view streaming logs**
    - Instead of using the "tail -f" command in your computer to print new lines in your log file as they get added, you'll get more when you view the streaming of log events on the CloudWatch Logs dashboard.

- Viewing logs on CloudWatch Logs enables you to perform interactive queries and analysis on log events through CloudWatch Logs Insights.

- **Log Rotation**

  - Log rotation is the process of archiving old log files. This approach makes the processing of log files more manageable and can help save disk space.

  - CloudWatch Logs supports the following log rotation rules:

    - Renaming of existing logs with a numerical suffix, then re-creating the original empty log file. For example, changing the name of the log file *syslog.log* to *syslog.log.1*.

    - Creating a new log file with the same pattern as the original file. For example, changing the name of the log file *syslog.log.2020-11-01* to *syslog.log.2020-11-02*.

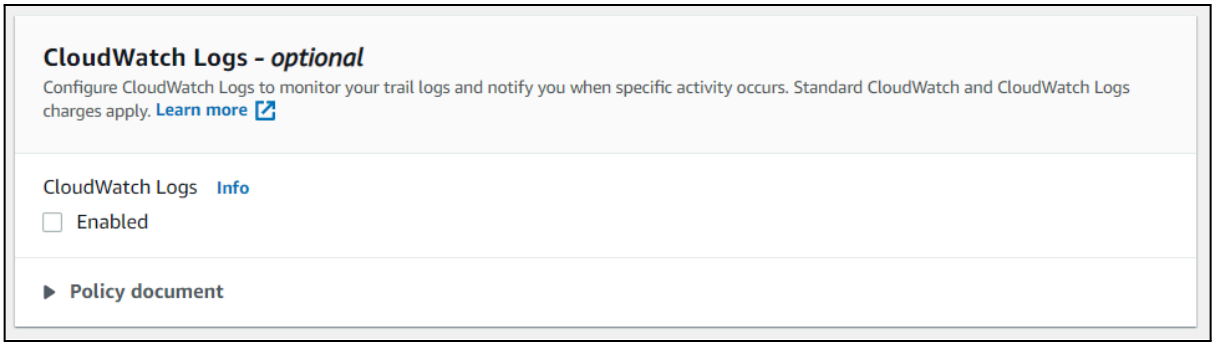
    - Trimming the original log by copying its contents to a new file. This is not recommended as it may introduce data loss.

References:
https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/Install-CloudWatch-Agent.html
https://aws.amazon.com/cloudwatch/faqs/
https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/WhatIsCloudWatchLogs.html

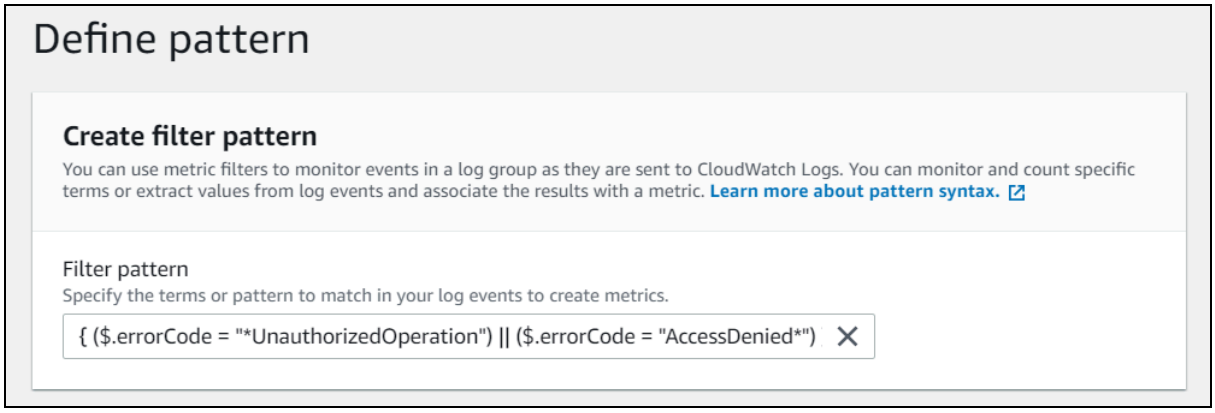
Using CloudWatch Metric Filter When Too Many Unauthorized AWS API Requests are Identified

We know that AWS CloudTrail monitors API calls that are being made within our account. Sure we can view who made the call and when it was made, but how can we know if a service is being spammed? Suppose we want to be alerted when a high volume of unauthorized API requests occurs — how do we do that?

CloudTrail delivers trail logs to an S3 bucket by default, but we can optionally configure CloudWatch Logs to monitor our trail logs so we can create an alarm when a certain activity happens.



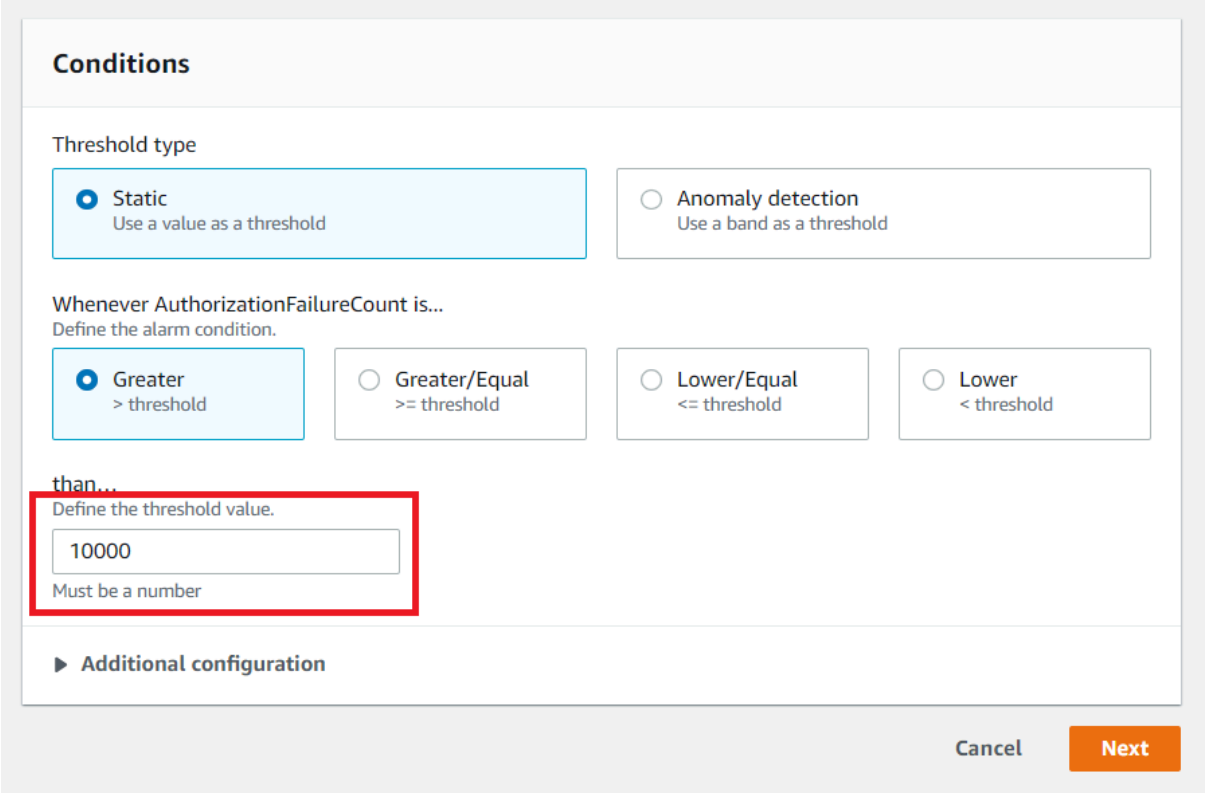
First, you must create a CloudWatch metric filter. The metric filter will search for the match terms, phrases, and values in your log events. In our case, we want CloudWatch to alarm when it detects a high number of unauthorized requests.



This filter pattern simply tells CloudWatch to detect error codes with an *UnauthorizedOperation* or *AccessDenied* error.

We said earlier that we could configure CloudWatch to alarm whenever a high number of unauthorized requests occurs, but there's no built-in feature in CloudWatch that will intelligently tell us if a certain number is deemed "high." So it means that the number is subjective, and it has to depend on your input.

You can set a ***threshold value***, which refers to the maximum number that must be breached for the alarm to go into the alarm state.



You could choose to trigger an SNS topic to notify you or your security team when an alarm state is in action.

Monitoring security group where the inbound rule is frequently updated

There are also cases when you want to monitor a security group where the inbound rule is always being updated and could be a security risk.

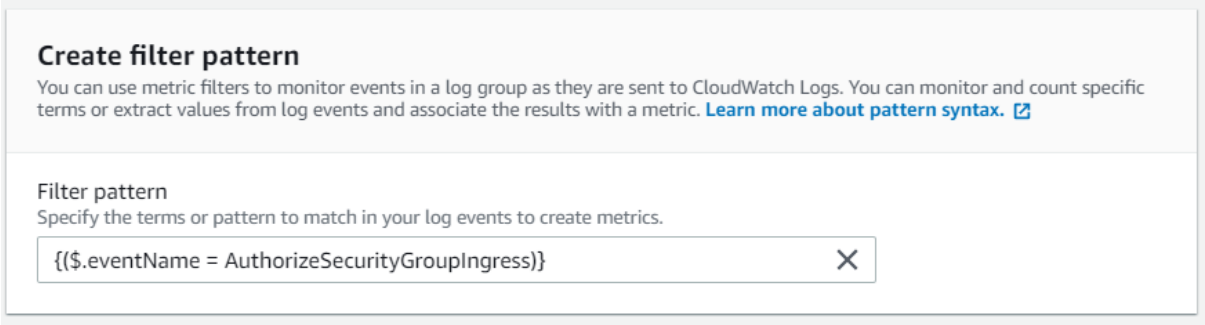
You can create a CloudWatch Alarm that is triggered when configuration changes that involve security groups happen. Below are the Security Group API call event types that you can add in the Filter Pattern of your CloudWatch Alarm:

- *CreateSecurityGroup* - Create a Security Group
- *DeleteSecurityGroup* - Delete a Security Group

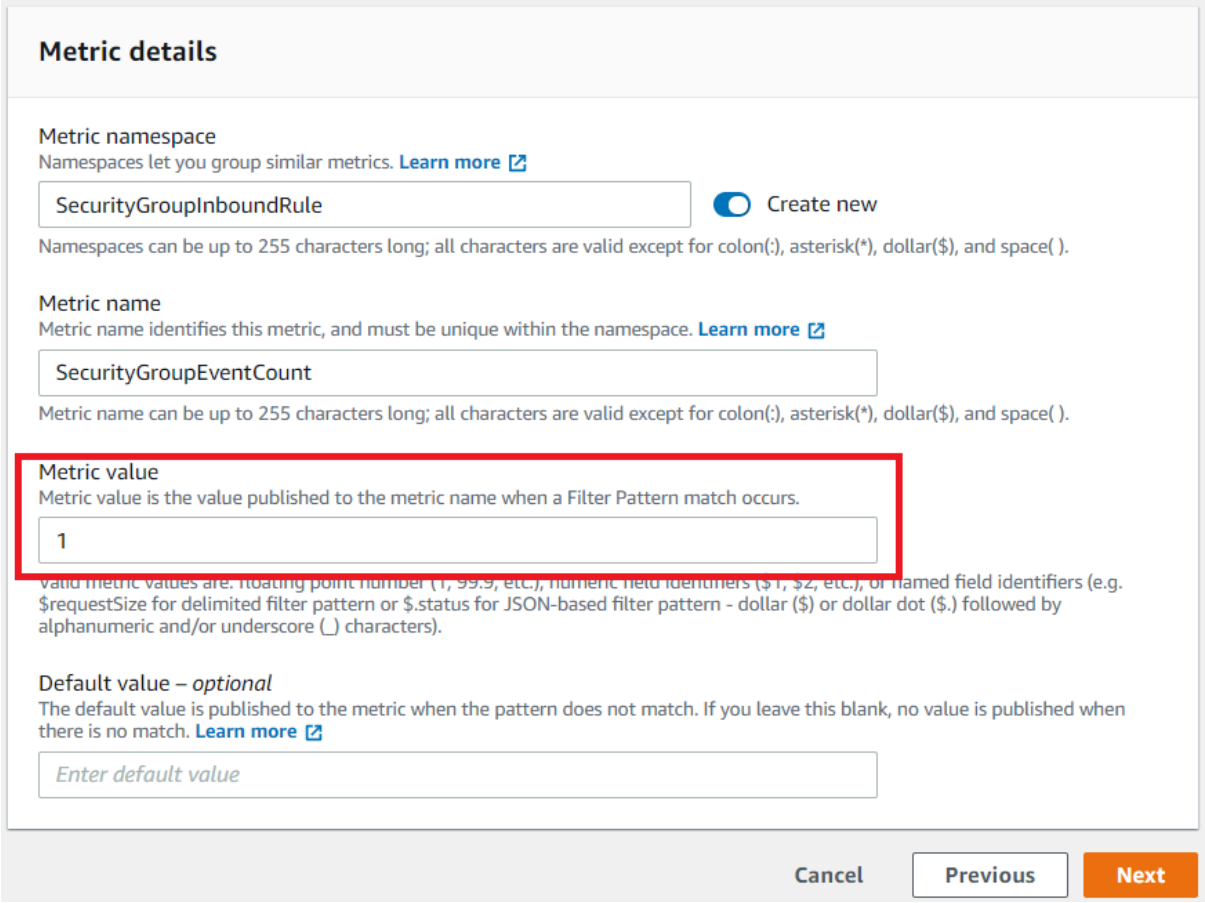
- *AuthorizeSecurityGroupEgress* - Add an Outbound Rule
- *AuthorizeSecurityGroupIngress* - Add an Inbound Rule
- *RevokeSecurityGroupEgress* - Remove an Outbound Rule
- *RevokeSecurityGroupIngress* - Remove an Inbound Rule

In this case, our point of interest is the security group's inbound rule. Again, we need to define a metric filter.

First, we need to ensure that the Filter Pattern includes the *AuthorizeSecurityGroupIngress* event in CloudWatch Alarm.



After that, we need to set the metric to 1, so we'll be notified immediately whenever a security group inbound rule changes.



Monitoring root account activities using the RootAccountUsage filter

Your root account has full access to all of its AWS resources. Unlike IAM users, you cannot disable or minimize the privileged access that comes with it. That is why AWS does not recommend logging in with a root account for everyday use unless necessary. Changing account settings, changing support plans, and restoring IAM permissions are some of the AWS activities that require root account access.

IAM best practices require you to enable MFA when signing in to your root account. You can use Trusted Advisor to detect disabled MFA for root accounts. On top of that, you can also get notified whenever your root account is used to access AWS by using CloudTrail and CloudWatch Alarms.

We learned in section 2 that CloudTrail produces management events which include IAM user and root user sign-in attempts. We use these events to trigger a CloudWatch Alarm. The method is similar to what we did in monitoring the security group inbound rule. You need to give permission to CloudWatch Logs to monitor your trail logs before creating a CloudWatch Alarm. This time, you have to input a different filter expression:

```
{ $.userIdentity.type = "Root" && $.userIdentity.invokedBy NOT EXISTS && $.eventType != "AwsServiceEvent"
}
```

References:
https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudwatch-alarms-for-cloudtrail.html#cloudwatch-alarms-for-cloudtrail-security-group
https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudwatch-alarms-for-cloudtrail.html#cloudwatch-alarms-for-cloudtrail-authorization-failures
https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudwatch-alarms-for-cloudtrail-additional-examples.html

Amazon Managed Grafana

Amazon Managed Grafana is a fully managed, enterprise-ready visualization and monitoring service based on the open-source Grafana platform. It allows security teams to create centralized dashboards for metrics, logs, and traces across multiple AWS accounts and Regions without managing any backend infrastructure. Amazon Managed Grafana natively integrates with services such as Amazon CloudWatch, AWS CloudTrail, Amazon Managed Service for Prometheus, AWS X-Ray, Amazon OpenSearch Service, AWS IoT SiteWise, and Amazon Timestream, offering a unified view of operational and security-related telemetry. AWS automatically handles provisioning, high availability, scaling, patching, and version upgrades for Grafana workspaces.

Security and Access Control

Amazon Managed Grafana includes built-in security controls aligned with AWS best practices for Identity and Access Management. Authentication and authorization for Grafana workspaces are managed through AWS IAM Identity Center (successor to AWS SSO) or external identity providers supporting SAML 2.0. Access can be restricted using fine-grained workspace roles (Administrator, Editor, Viewer), ensuring the principle of least privilege. Amazon Managed Grafana also supports secure private connectivity to data sources inside a VPC using AWS PrivateLink or VPC peering, preventing exposure of logs or metrics over the public internet. All data is encrypted at rest and in transit, and operational actions performed against AMG are logged in AWS CloudTrail, supporting auditing and forensic investigations.

Shared Responsibility Model

- **AWS Responsibility – "Security of the Cloud"**
  - AWS secures the underlying infrastructure that powers Amazon Managed Grafana, including hardware, networking, and foundational AWS services.
  - AWS ensures the service is built with strong security controls and undergoes regular independent audits.
  - Compliance validation is performed through AWS Compliance Programs; Amazon Managed Grafana inherits all applicable controls listed in *AWS Services in Scope by Compliance Program*.
- **Customer Responsibility – "Security in the Cloud"**
  - Customers must configure access controls, data permissions, network settings, and workspace roles according to their security requirements.
  - Responsibility levels vary depending on how the service is used and the sensitivity of the data being visualized.
  - Organizations must ensure compliance with internal policies, regulatory obligations, and industry standards relevant to their environment.

Monitoring, Dashboards, and Alerting

Amazon Managed Grafana enhances security operations by enabling teams to visualize CloudWatch metrics, CloudTrail events, Prometheus data, and OpenSearch logs in highly customizable dashboards that assist in threat detection, anomaly investigation, and operational monitoring. Its rich panel library allows deep correlation of signals across distributed systems, improving the speed and accuracy of incident response. Amazon Managed Grafana also supports Grafana's unified alerting system, letting analysts define rules that evaluate incoming telemetry and send notifications when suspicious thresholds or patterns are detected. These capabilities help security teams identify API anomalies, unauthorized access patterns, performance degradations, or unusual operational events across multiple AWS accounts.

References:
https://docs.aws.amazon.com/grafana/latest/userguide/what-is-Amazon-Managed-Service-Grafana.html
https://docs.aws.amazon.com/grafana/latest/userguide/security.html
https://docs.aws.amazon.com/grafana/latest/userguide/v10-alerting-overview.html

Amazon CloudWatch Managed Policies

A managed policy is a policy that is created and predefined by AWS for different services. This type of policy is intended for common use cases, so you don't have to write the policy from scratch. You can not modify the statements inside a managed policy. If managed policies do not support your application demands, you can create a customer-managed policy that you need to administer and write yourself.

Amazon CloudWatch has several managed policies, but you don't have to know them all. This section only covers the relevant policies that may appear in the actual exam.

Types

CloudWatchFullAccess - this type of managed policy allows the use of all CloudWatch operations. This should only be given to Administrators. This is not a good use case if you're strictly implementing the least privileged access for users in production.

CloudWatchReadOnlyAccess - this type of managed policy only allows read operations like getting metrics data or listing a dashboard's details. Any operations that don't modify configuration settings or data are under this policy.

CloudWatchActionsEC2Access - this type of managed policy gives read-only access to CloudWatch alarms and metrics. You can also view EC2 metadata and perform *stop*, *reboot*, and *terminate* to EC2 instances.

Reference:
https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/iam-identity-based-access-control-cw.html#managed-policies-cloudwatch

Real-time Logging Using Amazon Data Firehose and Amazon OpenSearch Service

Amazon Data Firehose is a fully managed service that allows you to load streaming data into data stores and analytics tools. It can batch, compress, and transform the data before sending it to its destination.

Security

- Amazon Data Firehose provides you the option to have your data automatically encrypted after it is uploaded to the destination.
- Manage resource access with IAM.

Amazon OpenSearch Service lets you search, analyze, and visualize your data in real-time. This service manages the capacity, scaling, patching, and administration of your OpenSearch clusters for you while still giving you direct access to the OpenSearch APIs.

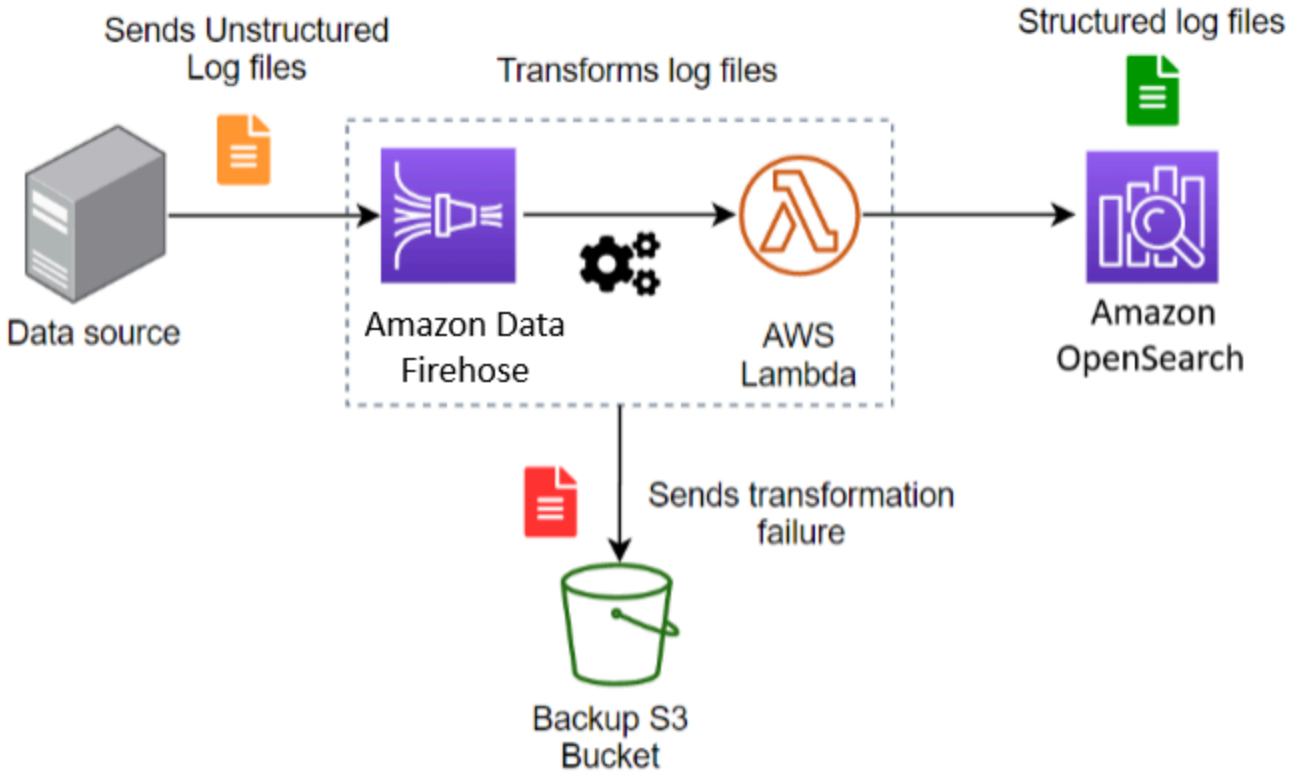
Security

- Amazon OpenSearch Service is **HIPAA eligible** and compliant with **PCI DSS, NOC and ISO** standards.
- You can securely connect your applications to your managed OpenSearch environment from your VPC or via the public Internet, configuring network access using VPC security groups or IP-based access policies.
- Securely authenticate your users and control access using Amazon Cognito and AWS IAM.
- Has built-in encryption of data-at-rest and in-transit to protect your data both when it is stored in your domain or in automated snapshots and when it is transferred between nodes in your domain.

Web server logs like Apache logs generate log data in an unstructured raw format, making it difficult to extract valuable information. Oftentimes, these schemaless log files need to be converted in a form that is easier to understand. A structured format will improve the indexing and searching of data, which is essential to log analysis.

If you need to set up a real-time logging solution, Amazon Data Firehose and Amazon OpenSearch are the way to go. These two services are both fully managed. It means that AWS abstracts all the operational workloads needed to maintain a server, allowing you to focus more on what is necessary for your application.

The architecture below illustrates the flow of data as it passes through Amazon Data Firehose down to OpenSearch.



A data source can be a web server, mobile device, or an IoT device. The data source continually pushes unstructured logs to the Amazon Data Firehose. The log files ingested through the Amazon Data Firehose are processed and transformed into a structured format (e.g., JSON, CSV) by a Lambda Function. After a successful data transformation, the new structured log file is delivered to OpenSearch. If something terrible happens to the delivery process, Amazon Data Firehose will send the data to a backup S3 bucket to prevent any data loss.

References:
https://docs.aws.amazon.com/opensearch-service/latest/developerguide/what-is.html
https://aws.amazon.com/firehose/
https://aws.amazon.com/opensearch-service/

Validate Findings from AWS Security Services to Assess the Scope and Impact of an Event

A critical skill in incident response is validating security findings to determine if they represent genuine threats and understanding their scope and impact. Not all findings indicate actual compromise – false positives must be distinguished from real incidents to prevent wasted resources on non-issues while ensuring real threats receive appropriate attention. Understanding scope (what was affected) and impact (severity of damage) guides response priorities and remediation efforts.

Understanding Finding Validation

Why Validation is Critical

Validation is essential because security services generate findings that include both false positives and false negatives. Legitimate activities can trigger security alerts, such as penetration testing, automation scripts, or approved third-party tools. Context matters significantly – the same finding may be benign in one environment but critical in another. Limited security team resources must focus on validated, high-impact incidents, and many compliance frameworks require documented incident validation and response.

The Validation Process

1. **Collect Finding Details** – Gather all available information from the security service
2. **Enrich Context** – Add organizational context (asset criticality, data classification, business function)
3. **Verify Indicators** – Confirm the finding's technical indicators are accurate
4. **Assess Legitimacy** – Determine if activity is authorized or malicious
5. **Determine Scope** – Identify all affected resources and accounts
6. **Evaluate Impact** – Assess potential or actual damage to CIA triad
7. **Prioritize Response** – Assign severity and urgency based on validation results

Validating GuardDuty Findings

Key Information in GuardDuty Findings

GuardDuty findings contain essential details needed for validation:

- **Finding Type –** Indicates threat nature (e.g., `UnauthorizedAccess:EC2/SSHBruteForce`)
- **Severity –** Low (0.1-3.9), Medium (4.0-6.9), High (7.0-8.9)
- **Resource Details –** Affected EC2 instance, IAM user, or S3 bucket
- **Action Details –** API calls, network connections, DNS queries
- **Actor Details –** Source IP, geolocation, organization (ISP/ASN)
- **First/Last Seen –** When activity started and most recently occurred

Validation Techniques

1. Verify Source IP Address

Check if the IP belongs to your organization (VPN, office, cloud infrastructure). Look up IP reputation in threat intelligence feeds and confirm geolocation matches expected user or service locations. Cross-reference with VPC Flow Logs for detailed connection information.

2. Correlate with CloudTrail

Find corresponding API calls in CloudTrail logs to verify user identity and authentication method. Check if MFA was used for sensitive actions. Review the user agent string for unexpected tools or automation. Examine error codes and success/failure patterns to understand the full context.

3. Check for Authorized Activity

Confirm with asset owners if activity was expected. Review change management tickets for scheduled maintenance. Check if your security team authorized penetration testing. Verify if third-party vendors have legitimate access. Review automated scripts and CI/CD pipelines that might generate alerts.

4. Assess Resource Criticality

Identify if the affected resource handles sensitive data. Determine the business impact of resource compromise. Check if the resource is in production or development environment. Review resource tags for classification and ownership information.

Validating Security Hub Findings

Security Hub Finding Categories

- **Software and Configuration Checks –** Misconfigurations and compliance violations
- **TTPs (Tactics, Techniques, Procedures) –** Evidence of attack techniques
- **Effects –** Confirmed security events with impact
- **Unusual Behaviors –** Anomalies requiring investigation
- **Sensitive Data Identifications –** Discovered sensitive data exposure

Validation by Security Standard

AWS Foundational Security Best Practices (FSBP)

Check the current resource configuration in the AWS Console to verify if the issue still exists. Some findings are point-in-time snapshots that may have been remediated. Review if the deviation is an approved exception that

should be documented in finding notes. Assess if the misconfiguration is exploitable in your environment or merely theoretical.

CIS AWS Foundations Benchmark

Validate against actual security impact rather than just compliance checkbox items. Some CIS controls may conflict with operational requirements and need documented exceptions. Prioritize controls that directly prevent compromise over purely procedural controls.

Validating Macie Findings

Policy Finding Validation

For `Policy:IAMUser/S3BucketPublic` findings, verify the bucket policy and ACL settings in the S3 Console. Check if public access is intentional (static websites, public datasets). Review bucket contents to assess data sensitivity. Confirm Block Public Access settings are configured appropriately.

Sensitive Data Finding Validation

For `SensitiveData:S3Object/Personal` findings, review the sample occurrences provided by Macie. Download the object and manually inspect it if authorized to understand the context. Verify if the data is production, test, or synthetic data. Check if the PII belongs to customers, employees, or test data. Assess if encryption and access controls are adequate for the data classification.

Validating Amazon Inspector Findings

Vulnerability Finding Validation

Check if the affected package or library is actually used by the application in a way that exposes the vulnerability. Verify if the vulnerability is exploitable in your specific environment – is it network accessible? Is authentication required? Review the CVSS score and attack complexity to understand exploitation difficulty. Check if compensating controls exist such as WAF rules or network isolation. Confirm if a patch is available and compatible with your environment.

Network Reachability Finding Validation

Verify if the network path is intended (e.g., bastion hosts should be accessible). Check if the exposed service is properly secured with authentication and encryption. Review if exposure is temporary for troubleshooting. Assess if the finding represents real risk or acceptable design.

Assessing Impact of Security Events

CIA Triad Impact Assessment

Confidentiality Impact

Assess what data was accessed or exfiltrated by reviewing CloudTrail for `GetObject`, `DescribeInstances`, and `ListBuckets` calls. Check S3 access logs for data retrieval patterns. Analyze VPC Flow Logs for large data transfers to external destinations. Determine data classification of accessed information.

Impact Levels:

- **HIGH –** Customer PII, financial data, or credentials accessed
- **MEDIUM –** Internal business data or non-sensitive customer data
- **LOW –** Public information or test data

Integrity Impact

Assess if data or configurations were modified by reviewing CloudTrail for `PutObject`, `ModifyDBInstance`, and `UpdateSecurityGroup` calls. Check AWS Config timeline for configuration changes. Investigate database modification logs. Verify resource state against known good baselines.

Impact Levels:

- **HIGH –** Production data modified or security controls disabled
- **MEDIUM –** Non-critical data changed or non-security configs modified
- **LOW –** Test data or temporary resources affected

Availability Impact

Assess service disruptions by checking for resource deletions (`TerminateInstances`, `DeleteBucket`). Review for crypto-mining consuming resources. Investigate denial-of-service indicators. Assess business impact of affected services.

Impact Levels:

- **HIGH –** Critical production services down, customer-facing impact
- **MEDIUM –** Internal services disrupted, degraded performance
- **LOW –** Development/test environments affected

Business Impact Considerations

- Regulatory compliance violations (GDPR, HIPAA, PCI DSS)

- Financial loss (fraud, ransom, service downtime)
- Reputational damage from breach disclosure
- Legal liability for data exposure
- Customer trust and retention impact

Common Exam Scenarios

- GuardDuty finding from known malicious IP during authorized pen test → Validate as false positive, add to trusted threat list
- Instance credentials used outside AWS → High severity, investigate CloudTrail for accessed resources, assess data exfiltration
- Security Hub compliance finding → Verify current state, assess if deviation is approved, prioritize based on exploitability
- Macie sensitive data finding → Verify data type, check encryption, assess access controls

References:
https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_findings.html
https://docs.aws.amazon.com/securityhub/latest/userguide/securityhub-findings-format.html
https://docs.aws.amazon.com/detective/latest/userguide/what-is-detective.html
https://docs.aws.amazon.com/macie/latest/user/findings-types.html

Automated Forensics Orchestrator for Amazon EC2 and EKS

Forensics Orchestrator for Amazon EC2 and EKS is a self-service AWS Guidance that enables Security Operations Centers to quickly configure automated forensics orchestration workflows for responding to potential security breaches. The Guidance follows digital forensics four-step process:

- Triaging: Assess threat severity and determine response
- Acquisition: Capture memory dumps and disk snapshots
- Investigation: Analyze forensic artifacts using automated tools
- Reporting: Generate findings and notifications

When threat detection engines identify malicious EC2 or EKS activities, findings are sent to Security Hub in the security account, which triggers EventBridge to initiate Step Functions orchestration workflows. Threat detection engines include:

- AWS Config
- Amazon GuardDuty
- Third-party tools

It supports EC2 instances and EKS clusters on EC2 instances distributed across multi-account and multi-region environments, providing capability to automatically capture and store targeted data for forensic examination while enabling SOCs to discover and analyze patterns of fraudulent activities.

The Guidance comprises five key components deployed via serverless architecture:

- **Forensic Triage Service:** Step Functions workflow receives EventBridge events from Security Hub custom actions (Forensic triage or Forensic isolation), retrieves instance information, determines isolation requirements based on Security Hub action, checks acquisition requirements based on instance tags, and stores triaging details in DynamoDB
- **Forensic Memory Acquisition Service:** Lambda functions leverage SSM Run Command to execute SSM documents on compromised instances, capturing memory dumps to S3 with metadata tags indicating OS and kernel version; post-acquisition isolation attaches new security group with no egress/ingress rules, detaching existing security groups to maintain chain of custody
- **Forensic Disk Acquisition Service:** Performs instance snapshots, shares EBS snapshots with forensic account using KMS keys, creates local copies in forensic account with forensic KMS keys for protection against account compromise or snapshot deletion
- **Forensic Investigation and Reporting Service:** Launches forensic EC2 instances from pre-built AMI (created via EC2 Image Builder) loaded with customer forensic tools and SSM Agent, attaches EBS volumes from snapshots, loads S3 memory dumps, executes SSM documents running forensic tools for timeline analysis, stores results in DynamoDB, sends investigation details via SNS
- **Forensic Image and SSM Document Builder Service:** Manages forensic AMI with required tooling

Memory and disk acquisition flows run in parallel. Step Functions orchestrates all workflows with DynamoDB storing state and results, enabling queries via AppSync API for forensic timeline review.

References:
https://docs.aws.amazon.com/solutions/latest/automated-forensics-orchestrator-for-amazon-ec2/welcome.html
https://docs.aws.amazon.com/solutions/latest/automated-forensics-orchestrator-for-amazon-ec2/architecture-overview.html
https://docs.aws.amazon.com/solutions/latest/automated-forensics-orchestrator-for-amazon-ec2/solution-components.html

AWS Fault Injection Service

AWS Fault Injection Service (FIS) is a fully managed service enabling fault injection experiments based on chaos engineering principles to test incident response effectiveness by creating controlled disruptive events. AWS strongly recommends completing planning phase and testing in pre-production environments since FIS performs real actions on real AWS resources.

Experiment templates contain:

- Actions: Fault injection activities like stopping instances, throttling APIs, increasing CPU
- Targets: Resources selected via IDs, tags, or states
- Stop conditions: CloudWatch alarms defining thresholds (maximum five per template)
- IAM role: Granting FIS execution permissions
- Logging configuration: CloudWatch Logs or S3 destinations

Actions run for specified durations sequentially or in parallel with post-action rollback returning targets to pre-experiment state. Tag-based targeting enables environment-specific experiments (e.g., "environment":"prod" with 10% instance subset). Stop conditions provide critical guardrails—experiments automatically halt if CloudWatch alarm thresholds are exceeded. Recommended stop condition metrics include:

- Steady-state workload metrics
- Component-level metrics receiving fault injection
- Synthetic monitors (user canaries)

FIS supports multi-account experiments targeting resources across different AWS accounts. CloudTrail logs document all FIS actions. Experiment report configuration captures CloudWatch dashboard data with pre/post-experiment windows, storing results in S3. Pricing is per-minute action duration multiplied by number of target accounts.

References:
https://aws.amazon.com/fis/features/
https://docs.aws.amazon.com/fis/latest/userguide/what-is.html

AWS Resilience Hub

AWS Resilience Hub provides a centralized location for managing and improving the resilience posture of applications on AWS. This service helps organizations proactively prepare and protect their AWS applications from disruptions by enabling them to define resilience goals, assess their current posture against those goals, and implement actionable recommendations based on the AWS Well-Architected Framework. The service integrates with AWS Fault Injection Service to create experiments that mimic real-life disruptions, helping organizations better understand application dependencies and uncover potential weaknesses. By consolidating all necessary AWS services and tools in one place, Resilience Hub enables continuous strengthening of resilience posture throughout the application lifecycle.

The platform delivers actionable recommendations to improve resilience and automatically generates code snippets that help create recovery procedures as AWS Systems Manager documents, referred to as Standard Operating Procedures (SOPs). During resilience assessments, organizations can evaluate their Recovery Time Objective (RTO) and Recovery Point Objective (RPO) targets under various disruption scenarios, including application failures, infrastructure issues, Availability Zone outages, and regional disruptions. AWS Resilience Hub generates recommended Amazon CloudWatch monitors and alarms that enable operators to quickly identify changes to an application's resilience posture once deployed. This comprehensive assessment approach analyzes application components using AWS Well-Architected Framework best practices to identify potential resilience weaknesses caused by incomplete infrastructure setup, misconfiguration, or areas requiring additional improvements.

Applications in AWS Resilience Hub are described by importing resources from various sources, including AWS CloudFormation stacks, Terraform state files, AWS Resource Groups, Amazon Elastic Kubernetes Service

clusters, or from applications already defined in myApplications. The service integrates with CI/CD pipelines through APIs, allowing resilience assessment and testing to be incorporated into continuous integration and delivery workflows. This integration ensures that infrastructure changes do not compromise application resilience before release into production. Organizations can run iterative assessments, compare results with previous reports, and continuously refine their applications until estimated workload RTO and RPO meet their defined targets. The service also maintains an audit trail of events during both planned and unplanned outages, helping organizations meet compliance and regulatory requirements.

References:
https://aws.amazon.com/resilience-hub/
https://docs.aws.amazon.com/resilience-hub/latest/userguide/what-is.html

Amazon Application Recovery Controller (ARC)

Amazon Application Recovery Controller helps you plan, orchestrate, and automate failover and recovery for applications running on AWS across Availability Zones (AZs) or Regions. It supports both multi-AZ and multi-Region disaster-recovery scenarios, reducing manual overhead while improving reliability and availability.

Key Features of Amazon ARC:

- **Multi-AZ recovery (Zonal Shift / Zonal Autoshift)** – Shift traffic away from an impaired AZ to healthy AZs in the same Region to mitigate single-AZ failures.
- **Multi-Region recovery (Region Switch + Routing Controls)** – Orchestrate cross-Region failover and failback plans, reroute traffic between Regions, and manage recovery even across multiple AWS accounts.
- **Readiness Checks & Monitoring** – Continuously validate that standby replicas (in other AZs or Regions) are ready to take over, by monitoring capacity, quotas, configuration, and routing policies.
- **Support for Load Balancers & EKS / EC2 / Auto Scaling / Container Resources** – ARC works with common compute and load-balancing resources such as ALB, NLB, Auto Scaling groups, Amazon EKS, EC2, ensuring broad support for application architectures.

Use Cases:

- Enable high availability and resilience for mission-critical applications that must remain online despite AZ or Region failures.
- Automate disaster recovery workflows and reduce operational overhead of manual failover scripts or procedures.
- Comply with regulatory or business requirements for rapid recovery and documented failover readiness.

References:
https://docs.aws.amazon.com/r53recovery/latest/dg/what-is-route53-recovery.html
https://docs.aws.amazon.com/r53recovery/latest/dg/compare-capabilities.html

Domain 3: Infrastructure Security

Overview

The third exam domain of the AWS Certified Security Specialty test is all about the process of securing your AWS infrastructure. About 18% of questions in the actual Security Specialty exam revolve around infrastructure security, which is the largest domain in the exam. Thus, you have to focus and spend more time studying these topics.

Infrastructure Security covers various concepts in designing and implementing your multitier workloads in the AWS Cloud. It involves designing and implementing security controls for edge services; designing and implementing network security controls; designing and implementing security controls for compute workloads as well as troubleshooting network security.

This domain will challenge your know-how in doing the following:

- Defining edge security strategies for common use cases (for example, public website, serverless app, mobile app backend)
- Selecting appropriate edge services based on anticipated threats and attacks (for example, OWASP Top 10, DDoS)
- Selecting appropriate protections based on anticipated vulnerabilities and risks (for example, vulnerable software, applications, libraries)
- Defining layers of defense by combining edge security services (for example, CloudFront with AWS WAF and load balancers)
- Applying restrictions at the edge based on various criteria (for example, geography, geolocation, rate limit)
- Activating logs, metrics, and monitoring around edge services to indicate attacks
- Implementing network segmentation based on security requirements (for example, public subnets, private subnets, sensitive VPCs, on-premises connectivity)
- Designing network controls to permit or prevent network traffic as required (for example, by using security groups, network ACLs, and Network Firewall)
- Designing network flows to keep data off the public internet (for example, by using Transit Gateway, VPC endpoints, and Lambda in VPCs)
- Determining which telemetry sources to monitor based on network design, threats, and attacks (for example, load balancer logs, VPC Flow Logs, Traffic Mirroring)
- Determining redundancy and security workload requirements for communication between on-premises environments and the AWS Cloud (for example, by using AWS VPN, AWS VPN over Direct Connect, and MACsec)
- Identifying and removing unnecessary network access
- Managing network configurations as requirements change (for example, by using AWS Firewall Manager)
- Creating hardened EC2 AMIs

- Applying instance roles and service roles as appropriate to authorize compute workloads
- Scanning EC2 instances and container images for known vulnerabilities
- Applying patches across a fleet of EC2 instances or container images
- Activating host-based security mechanisms (for example, host-based firewalls)
- Analyzing Amazon Inspector findings and determining appropriate mitigation techniques
- Passing secrets and credentials securely to compute workloads
- Identifying, interpreting, and prioritizing problems in network connectivity (for example, by using AWS VPC Reachability Analyzer or AWS Security Hub findings related to misconfigured security groups and network ACLs)
- Determining solutions to produce desired network behavior
- Analyzing log sources to identify problems
- Capturing traffic samples for problem analysis (for example, by using Traffic Mirroring)

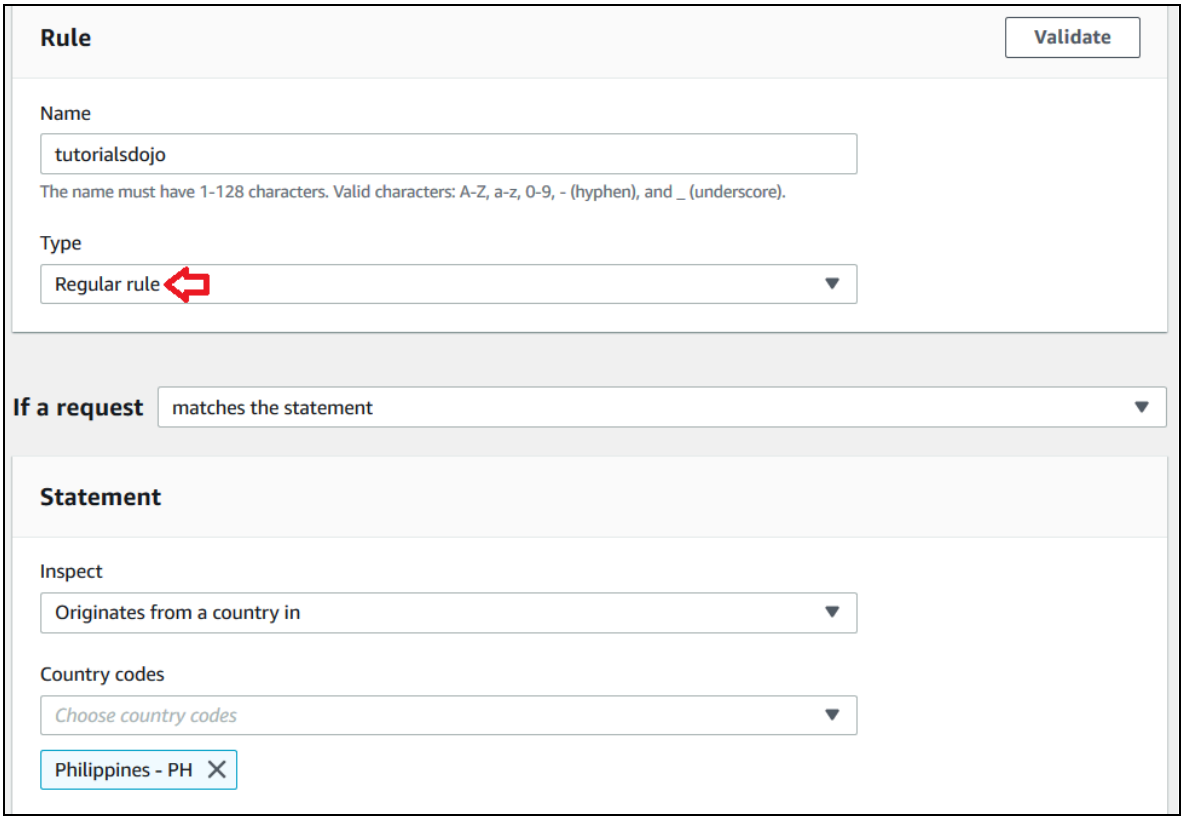
In this chapter, we will cover all of the related topics for edge, network, and server security in AWS that will likely show up in your Security Specialty exam.

AWS WAF and AWS Firewall Manager

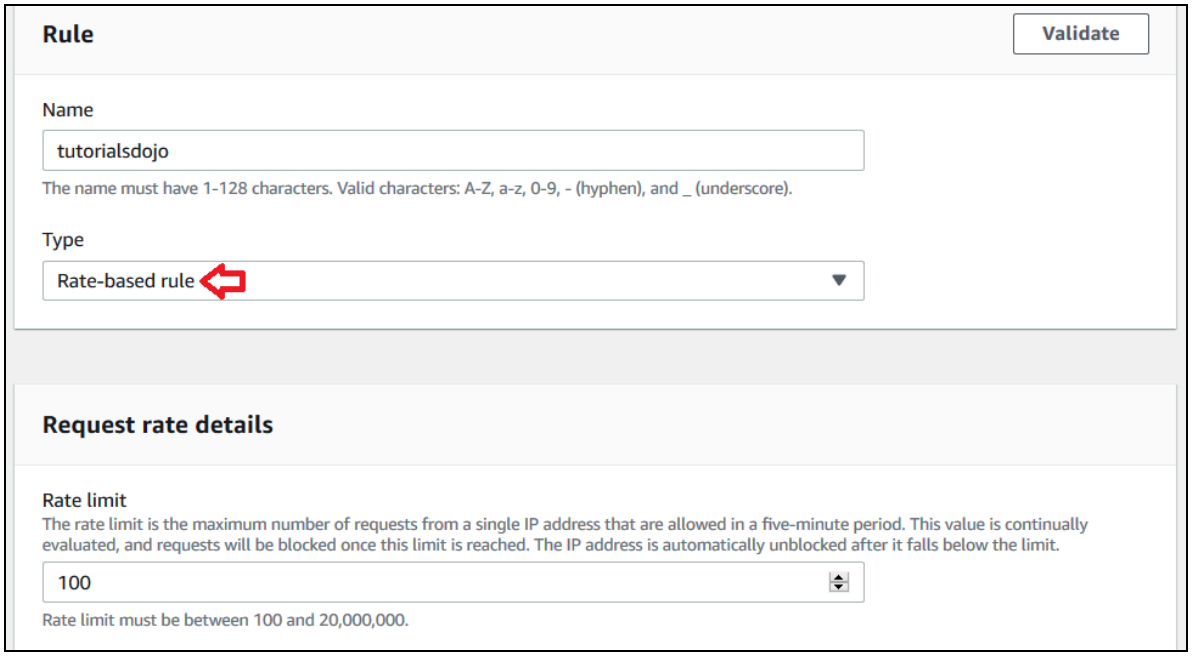
AWS Web Application Firewall (WAF) helps protect web applications from attacks by allowing you to configure rules that allow, block, or monitor (count) web requests based on the conditions that you define. These conditions filter web requests based on IP addresses, HTTP headers, HTTP body, or URI strings to block common attack patterns, such as SQL injection or cross-site scripting.

WAF Rules:

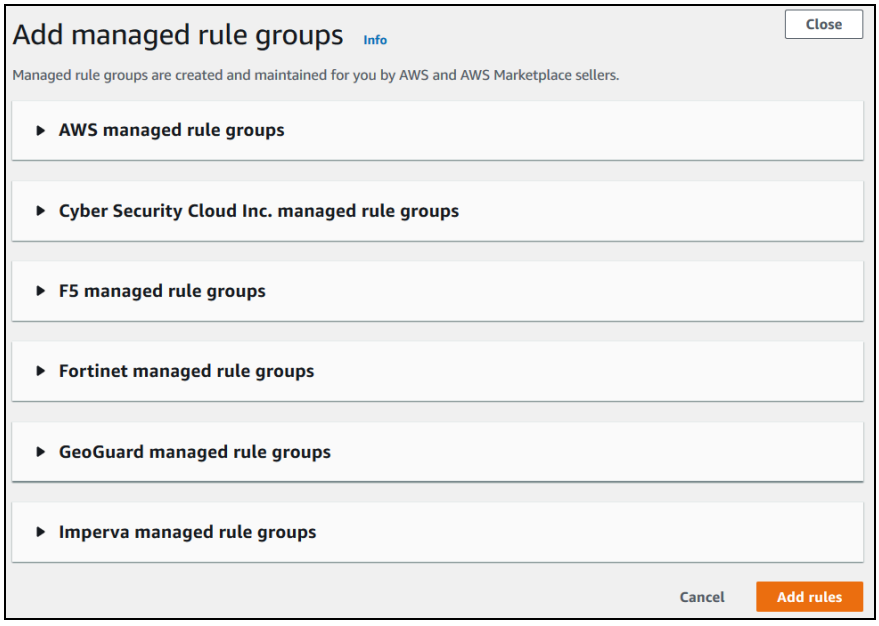
1. Custom Rules:
    - **Regular rules** - use only conditions to target specific requests.



    - **Rate-based rules** - are similar to regular rules, with a rate limit. Rate-based rules count the requests that arrive from a specified IP address every five minutes. The rule can trigger an action if the number of requests exceeds the rate limit.



2. **WAF Managed Rules** are an easy way to deploy pre-configured rules to protect your applications' common threats like application vulnerabilities. All Managed Rules are automatically updated by AWS Marketplace security Sellers.



AWS Firewall Manager allows organizations to centrally manage and enforce security policies across accounts and resources, even as requirements change. This is crucial in multi-account AWS environments, where each account may host different workloads with varying security needs. Firewall Manager integrates with AWS WAF

(Web Application Firewall), AWS Shield, and AWS Network Firewall, making it a comprehensive solution for varying types of network protection.

Imagine having a parent company that manages several subsidiary companies, each with its own AWS account. The parent company wants to enforce a unified security policy across all web applications deployed by the subsidiaries. These web applications may range from customer portals to internal dashboards, and they're all hosted in their own AWS accounts but fall under the corporate umbrella.

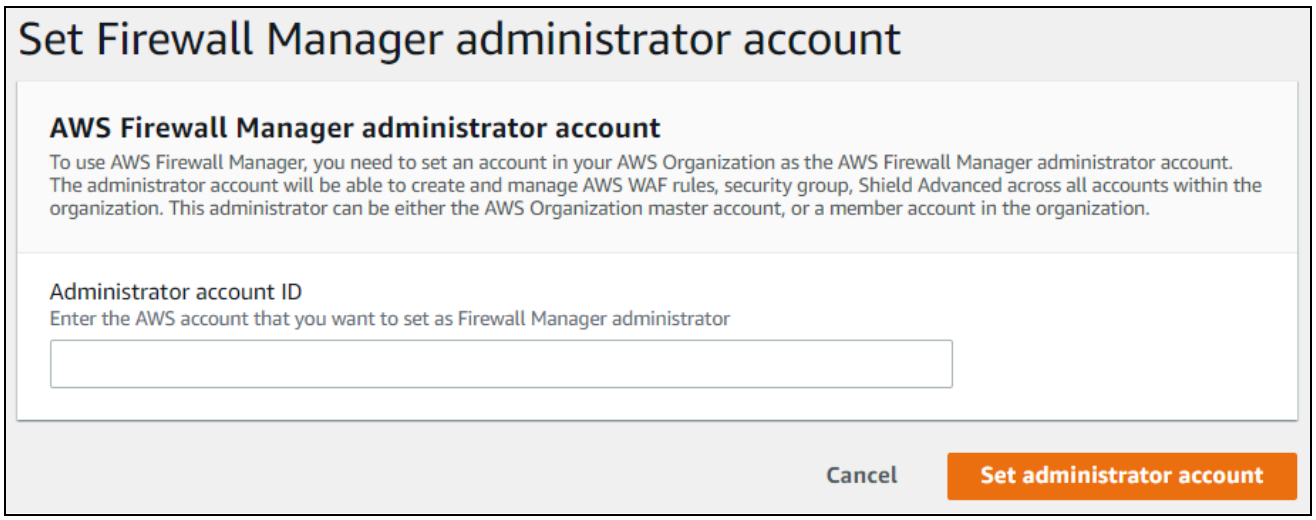
Problems:

- Manually configuring AWS WAF rules for each subsidiary's AWS account is time-consuming and error-prone.
- Subsidiaries may inadvertently misconfigure the WAF settings.
- There's a need for real-time updates and monitoring to ensure that all accounts are consistently secured.

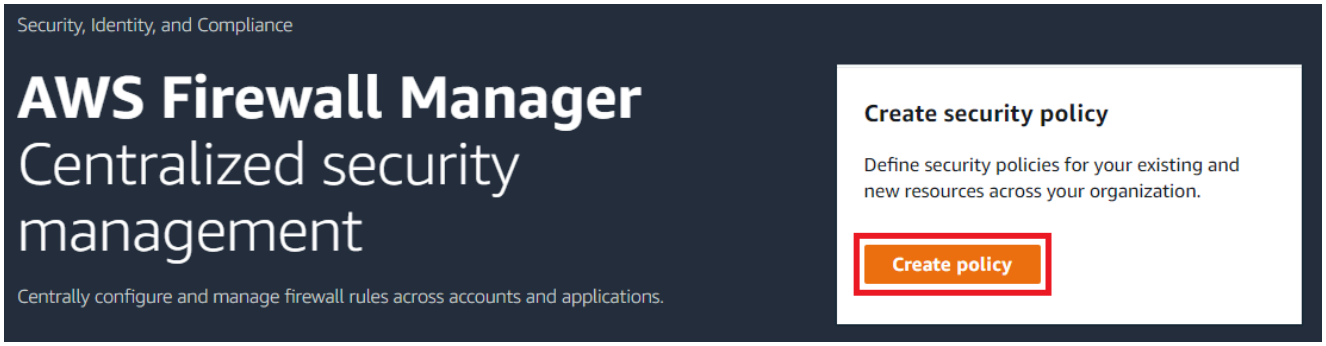
By using AWS Firewall Manager, you can centralize the configuration and management of AWS WAF rules. This allows the parent account to apply WAF rules uniformly across the subsidiary accounts and quickly roll out updates or changes to WAF rules.

Here are the steps to get started with AWS Firewall Manager:

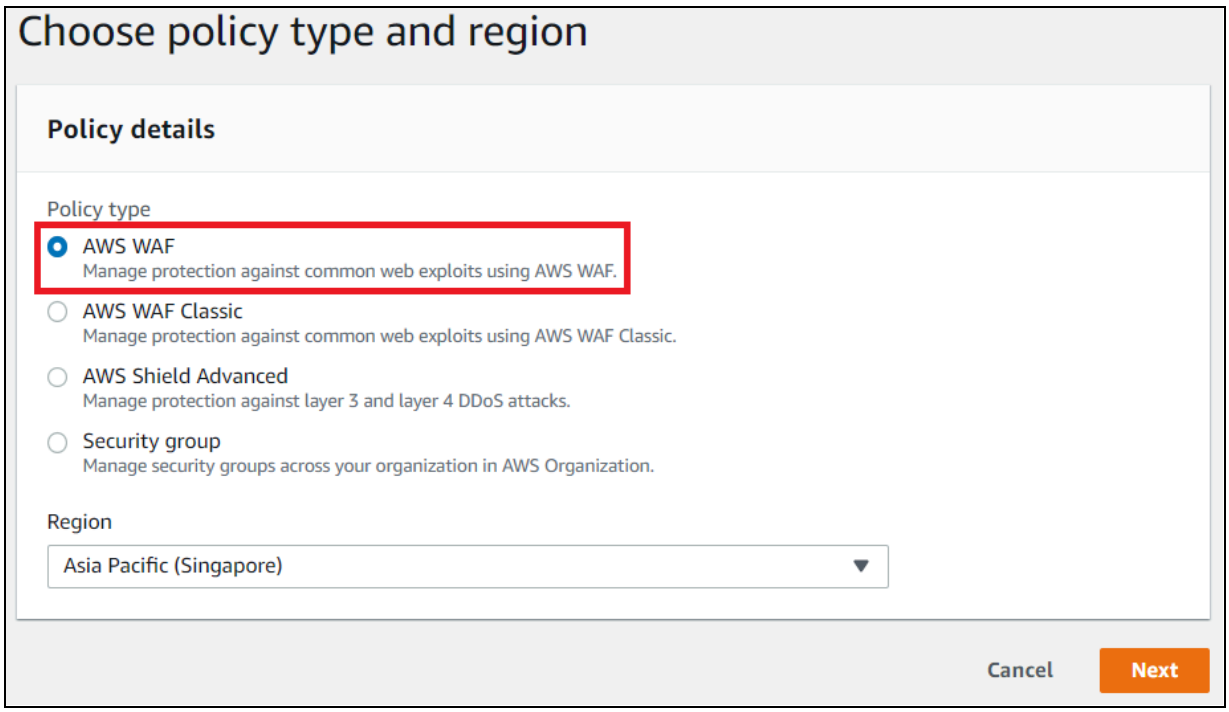
1. You need to set up an account in AWS Organizations first and enter the administrator account ID in AWS Firewall Manager.



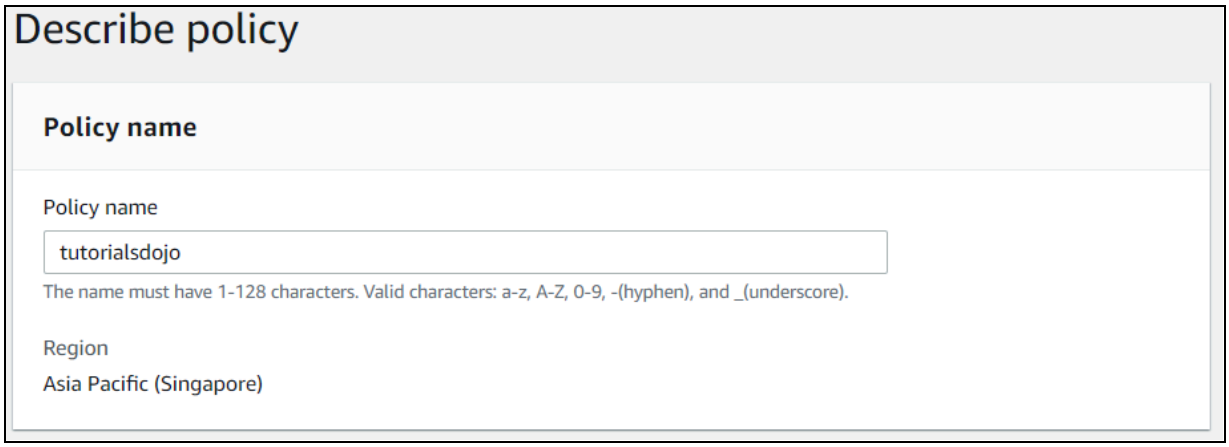
2. Go to the AWS Firewall Manager dashboard and click Create Policy.

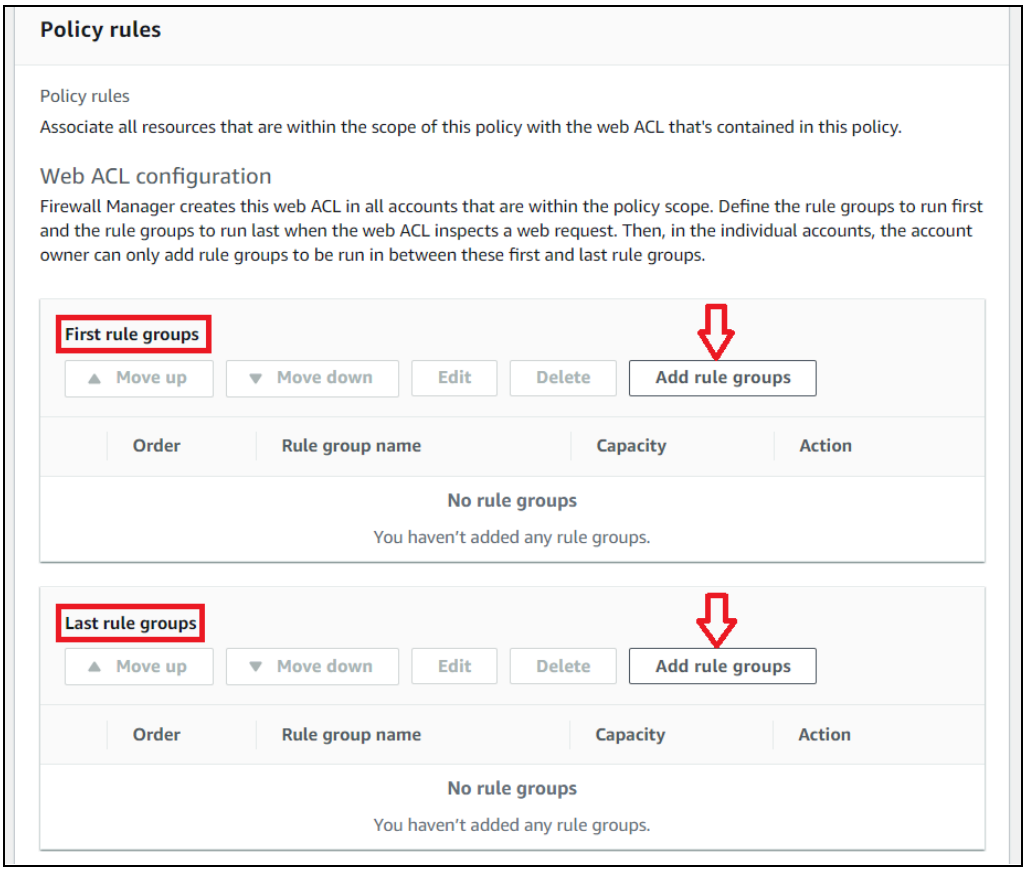


3. Select AWS WAF in the policy type and select your preferred region. Click next.

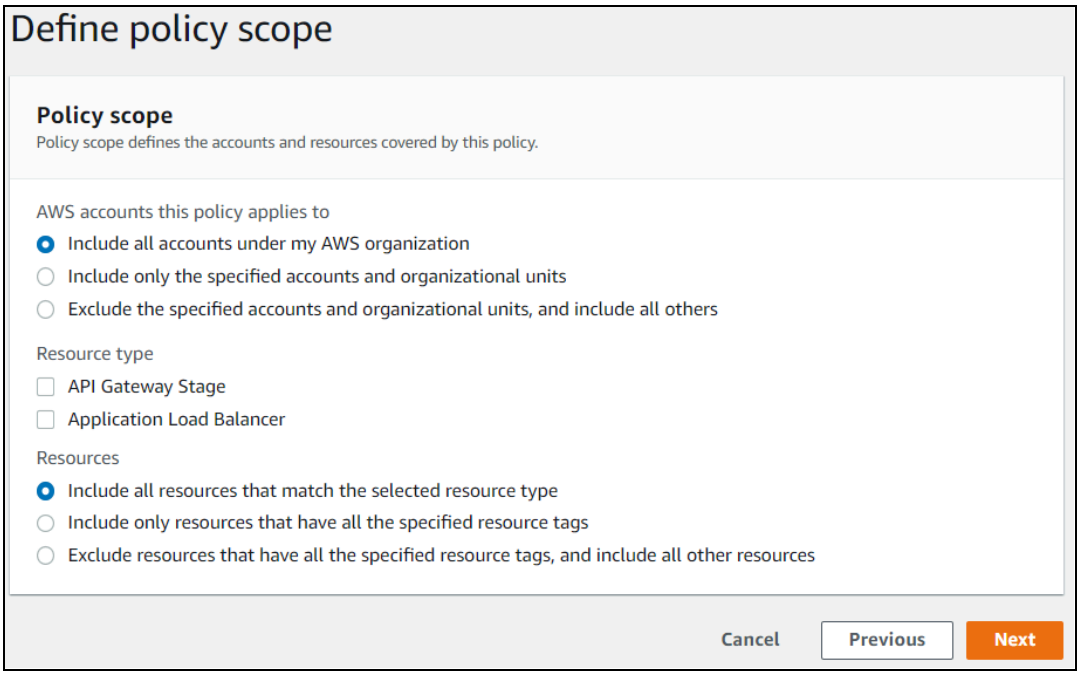


4. Fill out the policy name and add rule groups in the policy rules section. Select the type of policy action and click next.





5. Define the scope of your policy. Configure the policy tag (optional) and click create policy.



6. You can verify the created policy in the Security Policies section of AWS Firewall Manager.

References:
https://docs.aws.amazon.com/waf/latest/developerguide/how-aws-waf-works.html
https://docs.aws.amazon.com/waf/latest/developerguide/fms-chapter.html

Blocking User Requests Based On User-Agent HTTP Header

The User-Agent HTTP header is a request header that lets the server determine the browser, browser engine, application, device, program, and vendor of the requesting user. By knowing all of this information, the server will know which version of a web page or what data is compatible with the requesting device/application before sending. For example, some websites may have a different user experience if you're using an Android mobile phone than if you're browsing from an iPhone or a desktop computer.

An example of a user-agent HTTP header:

```
Mozilla/5.0 (iPhone; CPU iPhone OS 8_3 like Mac OS X) AppleWebKit/600.1.4 (KHTML, like
Gecko) FxiOS/1.0 Mobile/12F69 Safari/600.1.4
```

You can use the user agent string's information to block excessive requests coming from suspicious IP addresses. The AWS version of implementing such a filter mechanism is by using the rate-based rule of AWS WAF. The rate-based rule counts the rate of requests coming from a specific IP address for a 5-minute interval. A rule action is triggered to block abusive requests when the rate exceeds the configured threshold limit.

To look for a specific string of an agent-user HTTP header, we have to nest the ***string match statement*** inside the *rate-based statement* of the rate-based rule. This nested statement lets AWS WAF only count the requests that meet the specified string included.

Suppose you have a rate limit of 1,000. If the user-agent string of a request coming from a specific IP address passes the limit but doesn't match the value defined in the *string match statement*, AWS WAF will not block the request. However, if the user-agent header contains the string that you set in the string *match statement,* the request will be counted and blocked once it exceeds the rate limit.

References:
https://docs.aws.amazon.com/waf/latest/developerguide/waf-rule-statement-type-rate-based.html
https://docs.aws.amazon.com/waf/latest/developerguide/ddos-get-started-rate-based-rules.html

AWS Network Firewall

AWS Network Firewall is a managed network firewall service for VPCs. This network security service comes with intrusion prevention and detection capabilities and can filter traffic within the perimeter of your Amazon VPC.

This service is commonly used in various network security use cases such as inspecting VPC-to-VPC traffic, filtering outbound traffic, securing both AWS Direct Connect connection and VPN traffic as well as filtering the

Internet traffic. AWS Network Firewall also offers fine-grained network security controls for interconnected VPCs via the AWS Transit Gateway.

You can also use this to filter your outbound traffic to prevent unwanted data loss, block malware, and satisfy your strict network security compliance requirement. A single AWS Network Firewall can be configured with thousands of rules that can filter out network traffic routed to known bad IP addresses or suspicious domain names. It can also protect the AWS Direct Connect or VPN traffic that originates from client devices and your on-premises environments. The AWS Network Firewall can ensure that only authorized sources of traffic are granted access to your mission-critical VPC resources. It is also capable of performing the same activities as your Intrusion Detection Systems and Intrusion Prevention Systems or IDP/IPS. This is achieved by inspecting all inbound Internet traffic using features such as ACL rules, stateful inspection, protocol detection, intrusion prevention, et cetera.

The AWS Network Firewall has 3 basic components, namely the Firewall, the Firewall Policy, and the Rule Group.

A firewall is a resource that you create in AWS Network Firewall. This firewall is connected to the Amazon VPC of your choice, where a network filter will be implemented based on the behavior defined in your firewall policy. Your firewall can have one firewall policy only, which contains rule groups that you define.

You can deploy the firewall in multiple Availability Zones and to one subnet per zone. Each subnet that is associated with your firewall must have at least one available IP address. Afterward, you must update your VPC route tables to send incoming and outgoing traffic through the firewall endpoints.

The second component is the firewall policy. This is a policy that defines the behavior of your firewall using a collection of stateless and stateful rule groups. A firewall policy can be associated with one or more firewalls. However, a firewall can only have one firewall policy. Changing a firewall policy affects all other firewalls that reference it.

The third component in the AWS Network Firewall is called a rule group. This is basically a collection of stateless or stateful rules that define how to inspect and handle the network traffic in your VPC. A rules configuration includes 5-tuple network values and domain name filtering. The 5-tuple format includes the source IP address, source port, destination IP address, destination port, and protocol.

AWS Network Firewall has two types of rules which could be stateless or stateful. The first one is stateless in the sense that it does not have any context of the packet's traffic flow. A stateless rule just checks the packet itself. On the contrary, a stateful rule knows the context or the state of the packet, including its direction flow and other information that is not provided by the packet itself. Each rule in your stateful rule group has an associated order for its evaluation sequence. This concept is similar to network access control lists and security groups. A network ACL is stateless, while a security group is stateful.

You can also choose how you want the AWS Network Firewall to handle the packets that match your rule criteria. You can either pass or drop a packet that was filtered by the network firewall. A packet can also be forwarded to another stateful rule group for re-evaluation. Setting up custom actions is possible as well. A custom action can be configured to publish data to CloudWatch metrics for future network analysis.

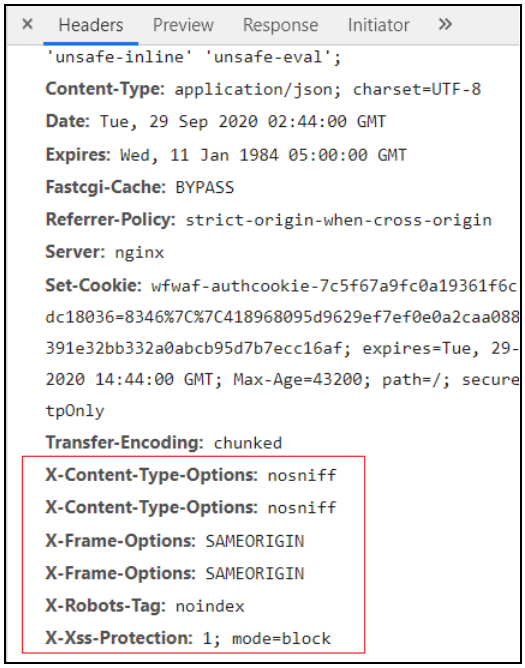
References:
https://aws.amazon.com/network-firewall
https://aws.amazon.com/blogs/networking-and-content-delivery/deployment-models-for-aws-network-firewall/

Adding HTTP Security Headers on the fly in CloudFront

HTTP security headers are a staple when it comes to securing websites. When implemented, they'll provide you with countermeasures and protection against common attacks like MIME sniffing, clickjacking, and cross-site scripting. Today, it's uncommon to find reputable websites that don't use security headers. These headers are a part of the server response that you receive from a request.

Some of the most commonly used security headers are:

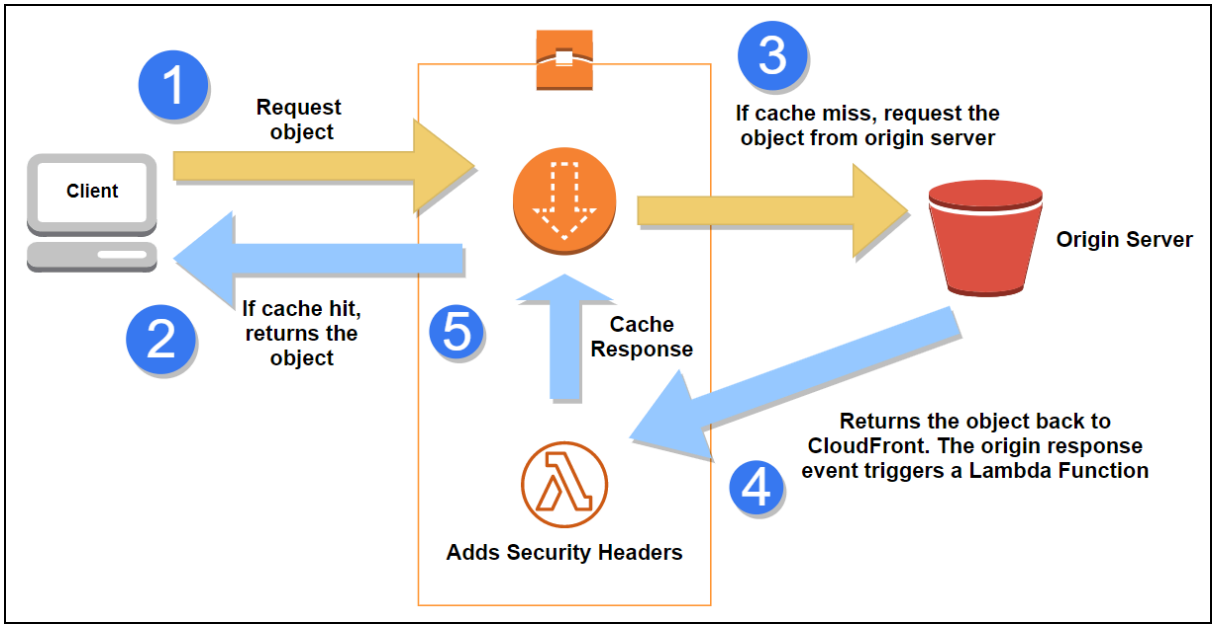
- **`X-Content-Type-Options`**
- **`X-Frame-Options`**
- **`X-XSS-Protection`**



If you're running your site with static files from S3 bucket, it is challenging to add security headers as Amazon S3 only supports adding custom headers with the prefix "x-amz-meta." A workaround for that is by using either a proxy server or Lambda@Edge.

Lambda@Edge is CloudFront's feature that lets you run Lambda functions closer to your end-users. This could improve your application's performance by reducing network latency. But aside from that added benefit, Lambda@Edge could also do intelligent processing of HTTP requests. We can use Lambda@Edge to add HTTP security headers to the response on the fly.

The diagram depicts the triggering of the Lambda function as CloudFront forwards requests to the origin server.



First, the client requests objects (e.g., HTML, CSS, JS Files, images, etc.) from CloudFront. If the requested items are found on the edge cache, the requested objects are immediately returned to the client. In the event of a cache miss, CloudFront will fetch the requested object from the origin server. The event by which the origin server responds to a request is called the origin response. With the event-driven nature of AWS Lambda, we can use the origin response event to trigger a Lambda function running on edge locations to generate and add the HTTP security headers needed. All of this process requires zero server administration.

References:
https://aws.amazon.com/blogs/networking-and-content-delivery/adding-http-security-headers-using-lambdaedge-and-amazon-cloudfront/
https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/lambda-requirements-limits.html#lambda-header-restrictions

Securing Amazon S3 and CloudFront Web Distributions

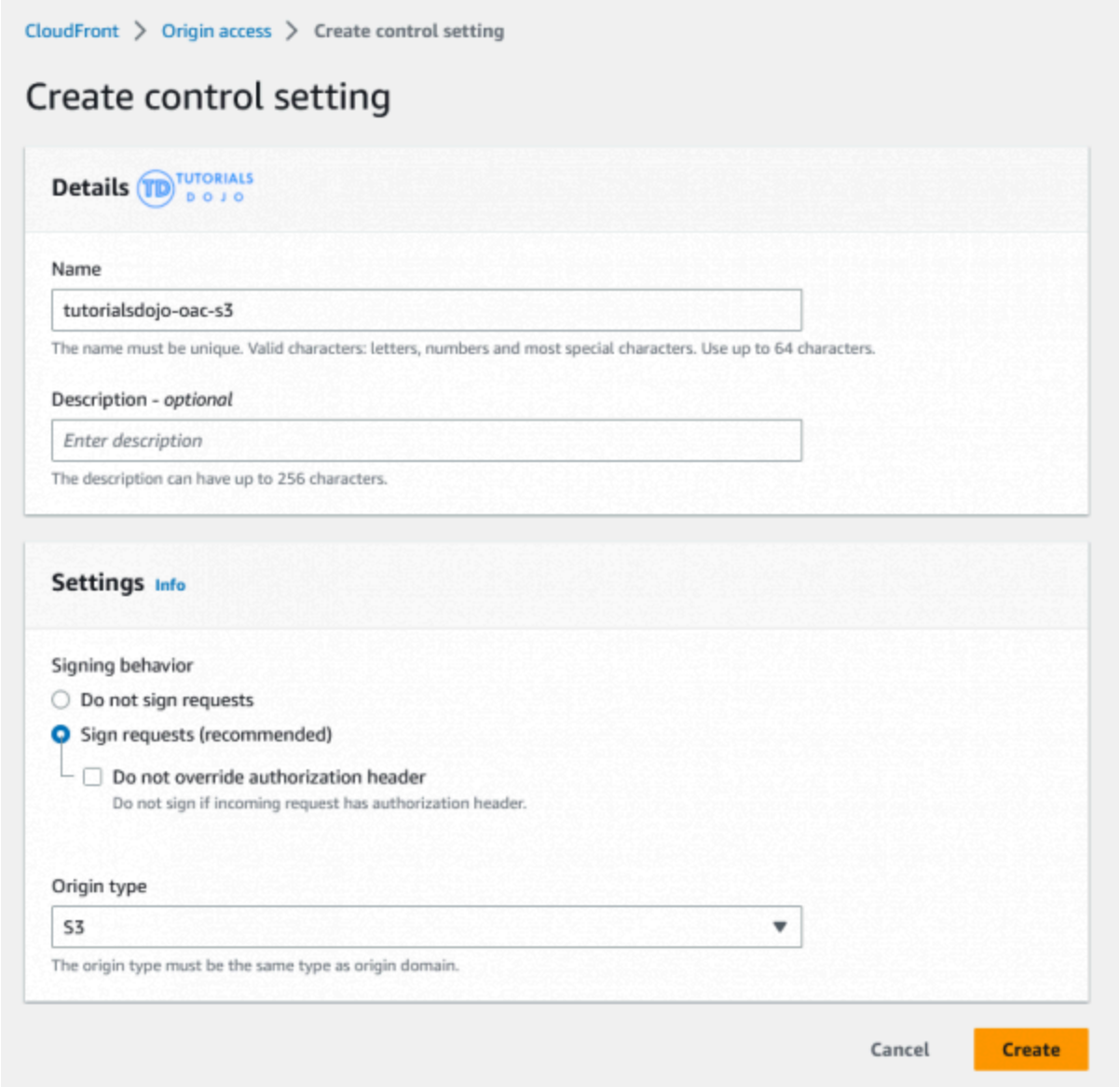
Amazon S3 and CloudFront gels together when it comes to distributing files over the Internet. If you have a web application that serves many static assets, picking S3 as your storage solution is a no-brainer.

An S3 bucket provides an S3 endpoint that you can use to distribute your files. That works, but in a typical web application set-up, you'll need more than that. Usually, you have to add an extra layer or two to make your application more performant and responsive. That can be achieved by caching the files or serving them from a content delivery network. It is good to know that CloudFront can do both.

CloudFront has two excellent features called *Signed URLs* and *Signed Cookies* that we can use to restrict access to sensitive files originating from an S3 bucket. For example, you have an application that sells high-definition photos. It is only right that paid users are the only ones that can view and download your images. And you can have that functionality by having your paid users access a Signed URL instead of the default CloudFront URL.

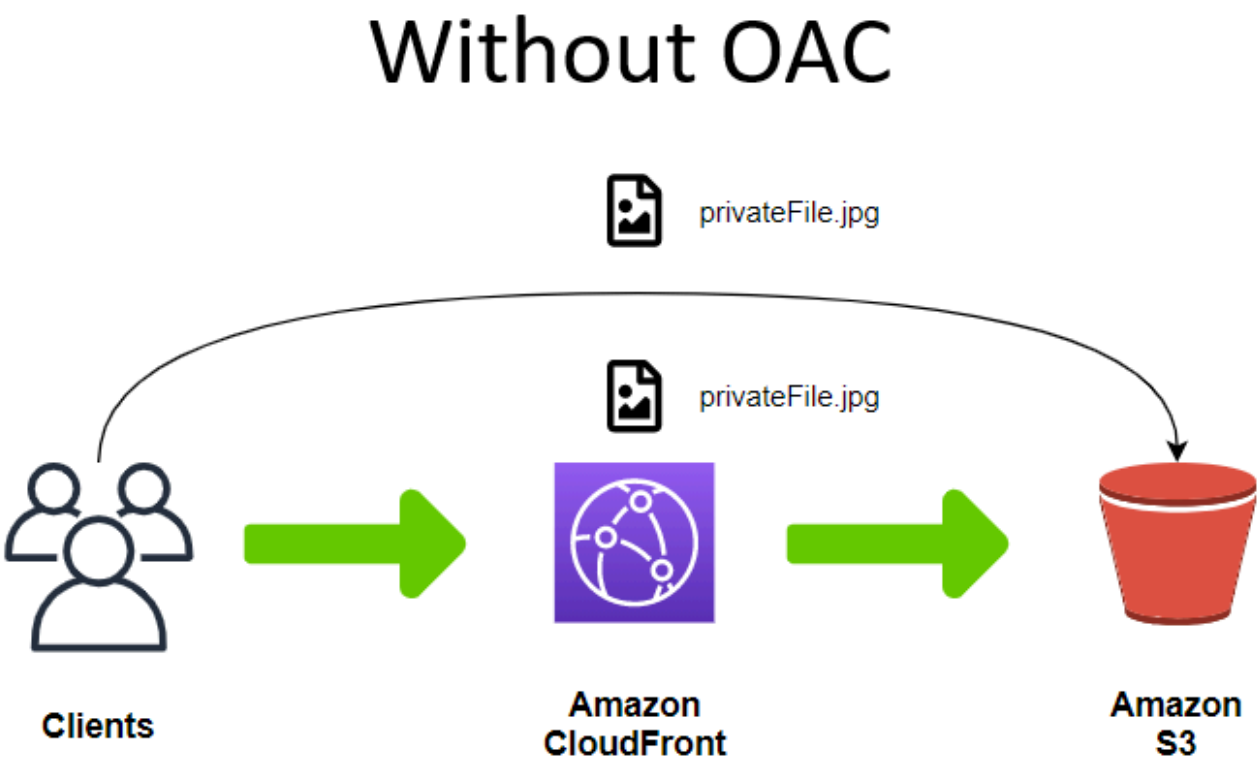
Why use Origin Access Control (OAC)?

I have mentioned earlier that you can retrieve objects from an S3 bucket using its S3 endpoint. That is the same endpoint that CloudFront uses as the origin domain name when creating a web distribution to serve files from S3. CloudFront provides you with a default domain name that you can use to access your content.



So if you have a file named *privateFile.jpg,* you can get that file by visiting either of the following endpoints:

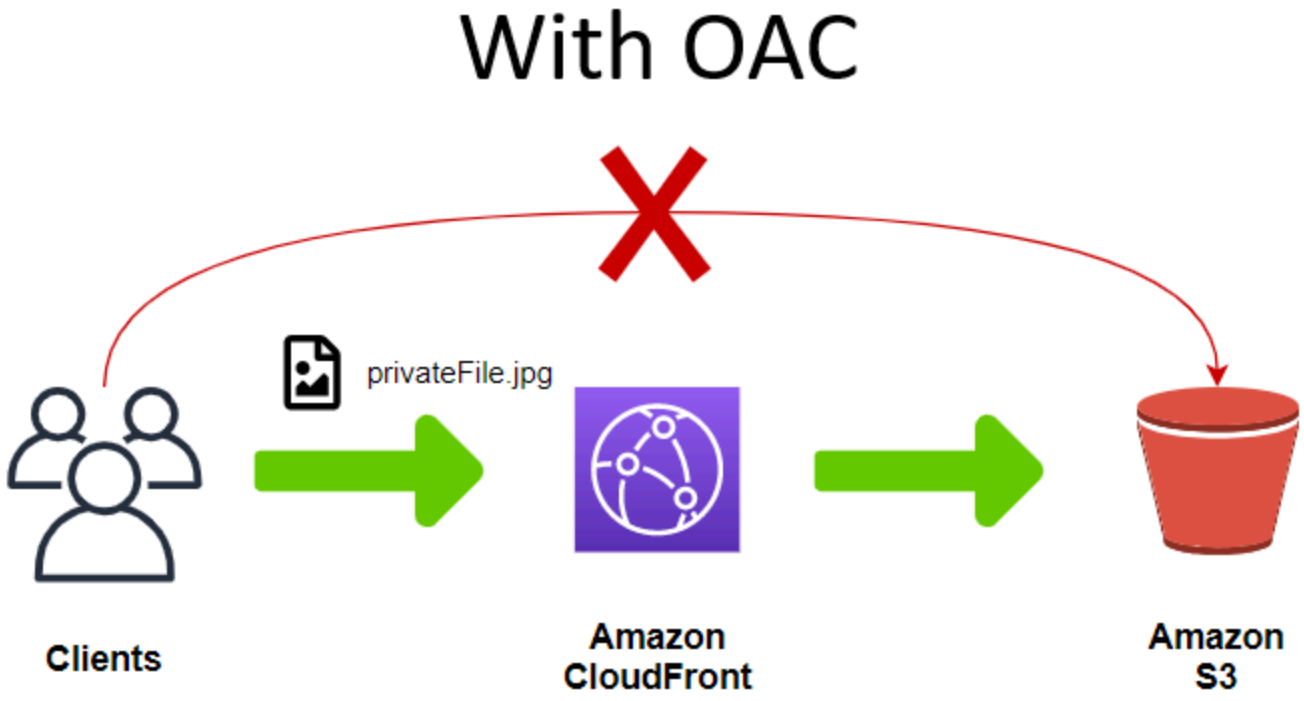
- *tutorialsdojo.S3.amazonaws.com/privateFile.jpg* (**S3 endpoint**)
- *https://d2908q01vomqc3.cloudfront.net/privateFile.jpg* (**CloudFront endpoint**)



This is a problem because:

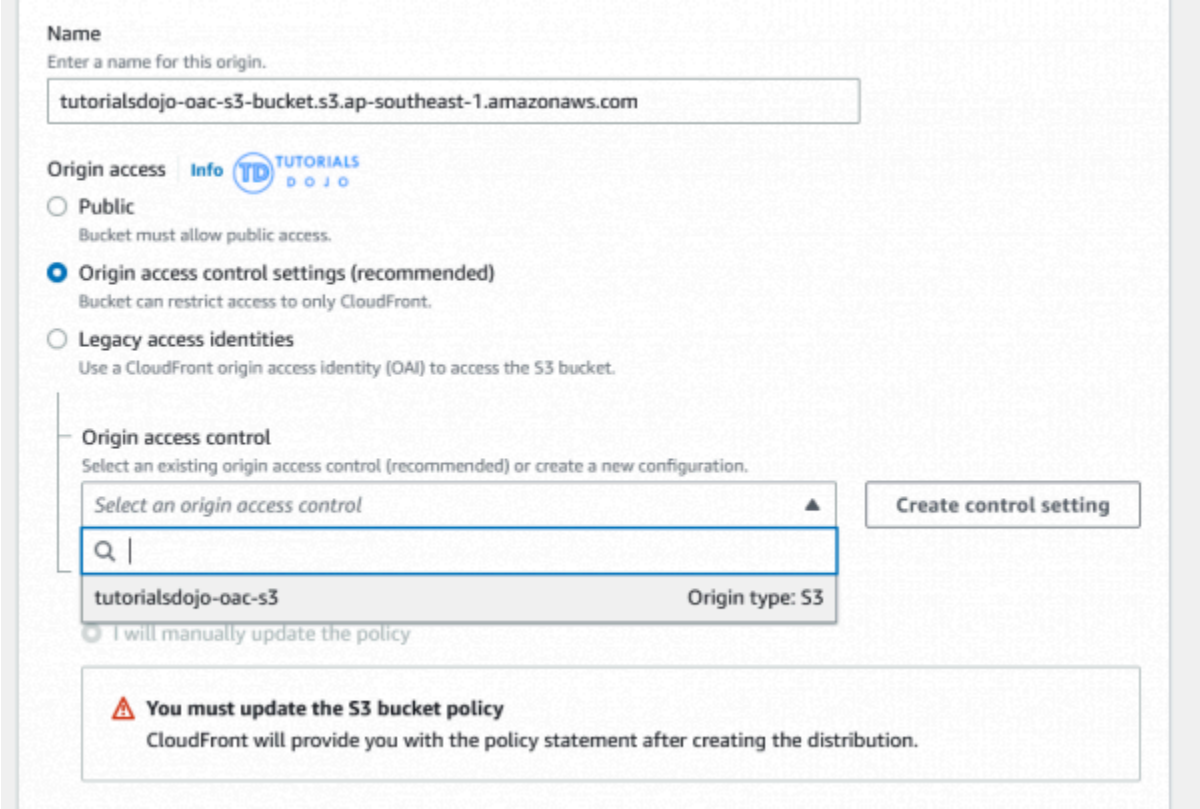
- Even if your Amazon S3 is already using CloudFront, the users can still bypass the CloudFront web distribution and access the S3 bucket directly.
- Users can bypass your Signed URL /Signed Cookies, effectively defeating its purpose.
- Users can directly request from the S3 bucket, and you'd have to pay for those GET requests. This can incur additional costs.

You can easily fix those aforementioned problems by using an Origin Access Control (OAC). Basically, OAC is used to restrict access only from CloudFront. By using OAC, CloudFront would completely deny any users from accessing your S3 bucket directly.



To implement OAC, follow these steps:

- Create OAC: In CloudFront, set up a new control setting with desired configurations.
- Set Up CloudFront Distribution: Create a CloudFront distribution, specifying the target S3 bucket and the previously created OAC.
- Update S3 Bucket Policy: Modify the S3 bucket policy to grant OAC read access.



References:
https://aws.amazon.com/blogs/networking-and-content-delivery/amazon-cloudfront-introduces-origin-access-control-oac/
https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/private-content-restricting-access-to-s3.html
https://tutorialsdojo.com/using-origin-access-control-to-secure-s3-object-access-to-cloudfront/

Amazon Route 53 Resolver Query Logs

Amazon Route 53 Resolver Query Logs provide visibility into DNS queries made from resources inside a VPC. The feature records information such as the domain queried, the query timestamp, the DNS response code, and the originating VPC or IP address. These logs help security teams monitor DNS activity, detect suspicious domains, investigate potential malware or data-exfiltration attempts, and maintain compliance with audit requirements. Logs can be sent to **Amazon CloudWatch Logs**, **Amazon S3**, or **Amazon Kinesis Data Firehose** for long-term storage, real-time analysis, or downstream security processing.

Security and Operational Considerations

- Helps detect indicators of compromise (IOCs) such as DNS queries to known malicious domains.
- Supports investigation of outbound traffic behavior and lateral movement patterns.
- Integrates with SIEM and security analytics tools using CloudWatch Logs, S3, or Firehose.
- Logging can be enabled at the VPC level, and each VPC can be associated with one or more query-logging configurations.
- DNS query logs do **not** capture traffic encrypted through DNS-over-HTTPS (DoH) or DNS-over-TLS (DoT).

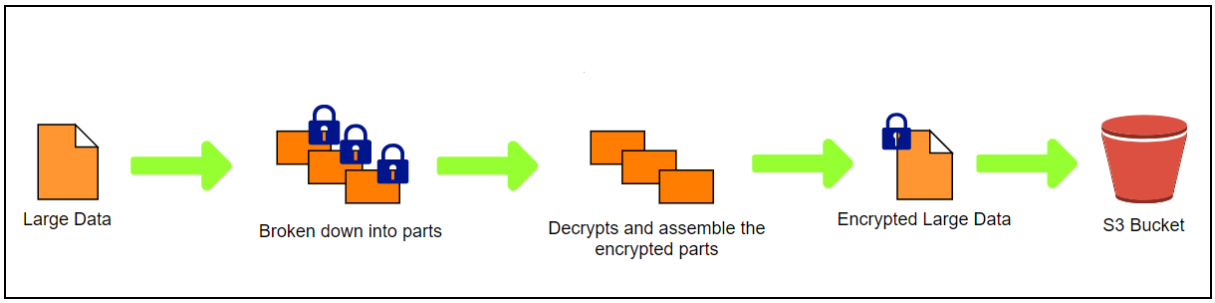
References:
https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resolver-query-logs.html

Uploading Large Encrypted Files in a SSE-KMS enabled S3 Bucket

When using the `aws s3 cp` to upload large files to Amazon S3, S3 implicitly uses the multipart upload feature. The multipart upload works by splitting up large data into individual parts that get uploaded in parallel, making the upload speed faster. Amazon S3 will then reconstruct the uploaded parts into the original data.

A KMS key can only encrypt files up to 4KB in size. To encrypt files larger than 4KB, you must use envelope encryption, which involves using a data key derived from the KMS key for encryption. In the case of SSE-KMS, envelope encryption is automatically handled by Amazon S3; you just need to have the necessary permissions for S3 to do its job. Let's say you initiate an upload of a 20MB file to an SSE-KMS-enabled S3 bucket. For the upload to succeed, you (the uploader) must have both ***kms:Decrypt*** and ***kms:GenerateDataKey**** permissions on the KMS key that S3 will use on your behalf.

The reason is that splitted parts uploaded to S3 are server-side encrypted. Amazon S3 needs to decrypt the parts first before they can be reassembled into their original form.



References:
https://repost.aws/knowledge-center/s3-large-file-encryption-kms-key
https://repost.aws/knowledge-center/s3-large-file-encryption-kms-key
https://aws.amazon.com/blogs/aws/amazon-s3-multipart-upload/

Protecting from DDOS Attacks through AWS Shield

AWS Shield is a managed DDoS protection service that safeguards applications running on AWS. This service has two tiers: Standard and Advanced.

- **Shield Standard** provides always-on network flow monitoring, which inspects incoming traffic to AWS and detects malicious traffic in real-time.
- **Shield Advanced** provides enhanced detection, inspecting network flows, and also monitoring application layer traffic to your Elastic IP address, Elastic Load Balancing, CloudFront, or Route 53 resources.

A **Distributed Denial of Service (DDoS)** attack comprises multiple systems that target a single system. The target system will be bombarded with packets from multiple locations. If the system cannot accommodate the large volume of requests, this will result in service unavailability.

DDoS attacks can be generated at different layers of the OSI model. The most common attacks are in the Network, Transport, Presentation, and Application Layers.

#	Layer	Application	Description	Vector Example
7	Application	Data	Network process to application	HTTP floods, DNS query floods
6	Presentation	Data	Data representation and encryption	SSL abuse
5	Session	Data	Interhost communication	N/A
4	Transport	Segments	End-to-end connections and reliability	SYN Floods, UDP Floods,

3	Network	Packets	Path determination and logical addressing	Smurf Attacks, ICMP Floods
2	Datalinks	Frames	Physical addressing	N/A
1	Physical	Bits	Media, signal, and binary transmission	N/A

To mitigate these kinds of attacks, we can classify DDoS attacks into two groups:

1. **Infrastructure Layer Attacks (Layers 3 and 4)**
    - The generated attacks in these layers are in a large volume of packets or requests to overload the servers' capacity. Examples of DDoS attacks in this category are smurf attacks, SYN and UDP floods.

2. **Application Layer Attacks (Layers 6 and 7)**
    - Unlike the attacks in the infrastructure layer, the generated attacks in the application layer are in a small volume. The focus of the generated attack in this group is to make your application unavailable to your users. An example of these attacks is the flooding of HTTP requests or search API and WordPress pingback attacks.

DDoS Protection Techniques

- **Reduce Attack Surface Area**
    - Exposure of application ports and protocols will result in possible points of attack. By minimizing the surface area, we can limit the attackers' options and focus on building protections in our infrastructure. We can implement this solution using Content Distribution Networks, Load Balancers, Firewalls, and Access Control Lists to restrict access directly to our servers or applications.

- **Plan for Scale**
    - **Transit capacity** is also known as bandwidth capacity. To prevent this kind of attack, your hosting provider must have redundant Internet access when handling large volumes of requests. The primary goal of a DDoS attack is to affect the availability of your applications. Using CDNs and smart DNS resolution services, we can create an additional layer of protection in our infrastructure. This will help us serve content and resolve DNS queries from locations that are closer to your users.
    - **Server capacity** is the most common type of attack that comes to our mind when we hear DDoS attacks. These are volumetric attacks that overwhelm the capacity of your resources. To resolve this problem, we need to have an elastic compute resource or a larger capacity to handle a large volume of attacks. You can also use a load balancer to shift the loads between resources to prevent overloading in one resource.

- **Know what is normal and abnormal traffic**
  - To protect us from different kinds of attacks, we must ensure that our traffic is always at the baseline level. This is known as rate limiting. Analyzing individual packets can help us accept legitimate traffic. To do this, we must understand the difference between normal and abnormal traffic.

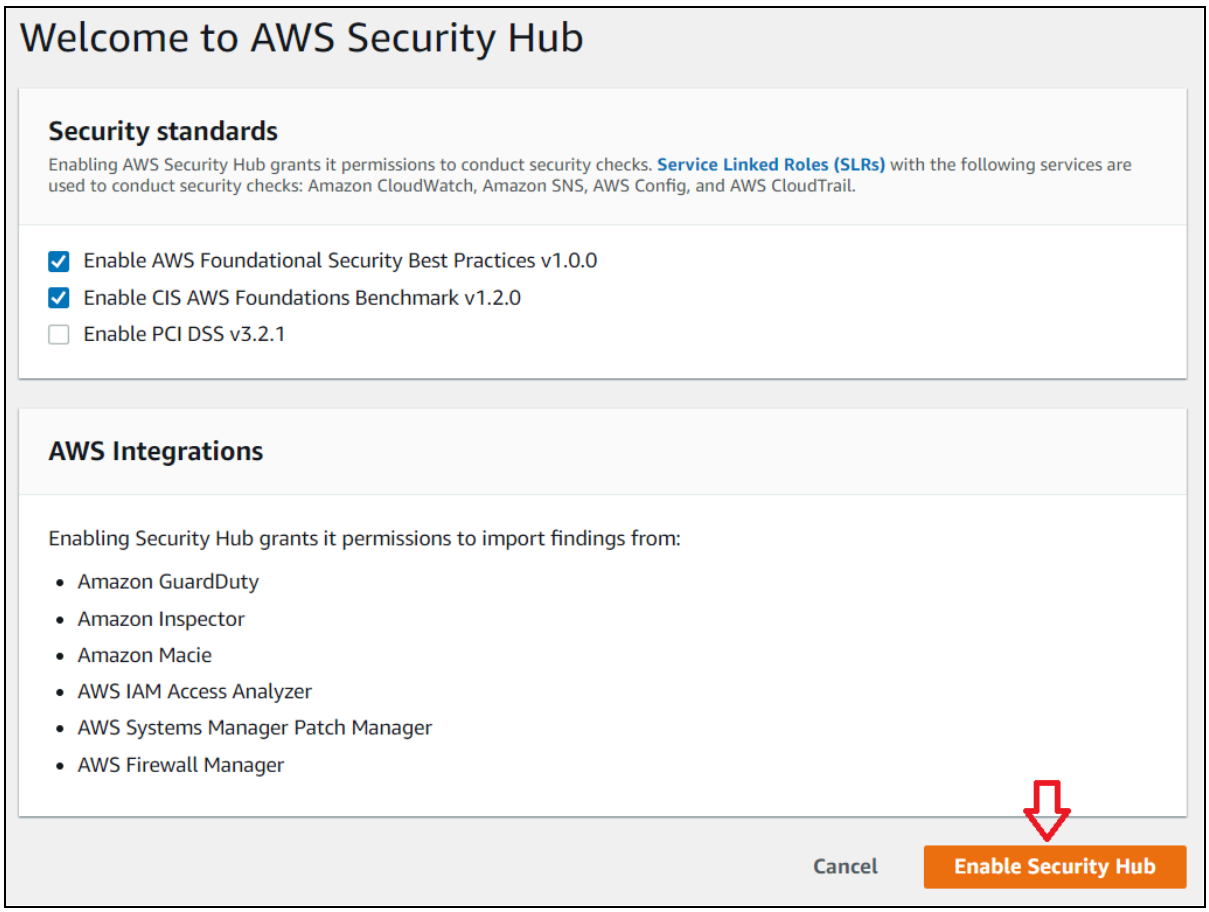
- **Deploy Firewalls for Sophisticated Application attacks**
  - Always remember: when deploying an application, you must associate it with a Web Application Firewall. This will mitigate the different kinds of attacks in the application layer, such as SQL injection and cross-site scripting, that will exploit our application's vulnerability. You can easily reduce these illegitimate requests by studying traffic patterns and creating customized protections.

Investigating AWS Resources that are Possibly Compromised

AWS Security Hub provides a comprehensive view of your security state within AWS and your compliance with security industry standards and best practices. It has integrated dashboards that consolidate your security findings across accounts to show you their current security and compliance status. All findings are stored for at least 90 days within AWS Security Hub.

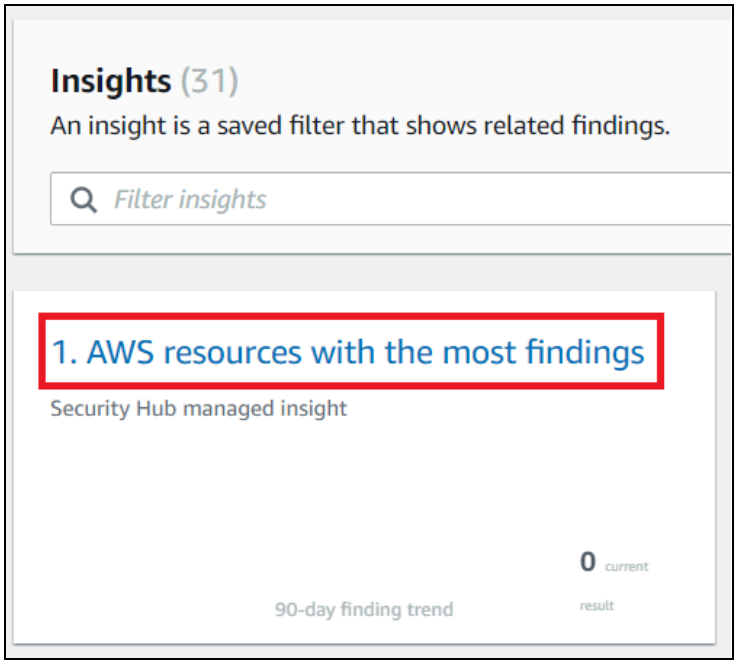
Here are the steps for this setup:

1. Go to the AWS Security Hub service and click on "Enable Security Hub".

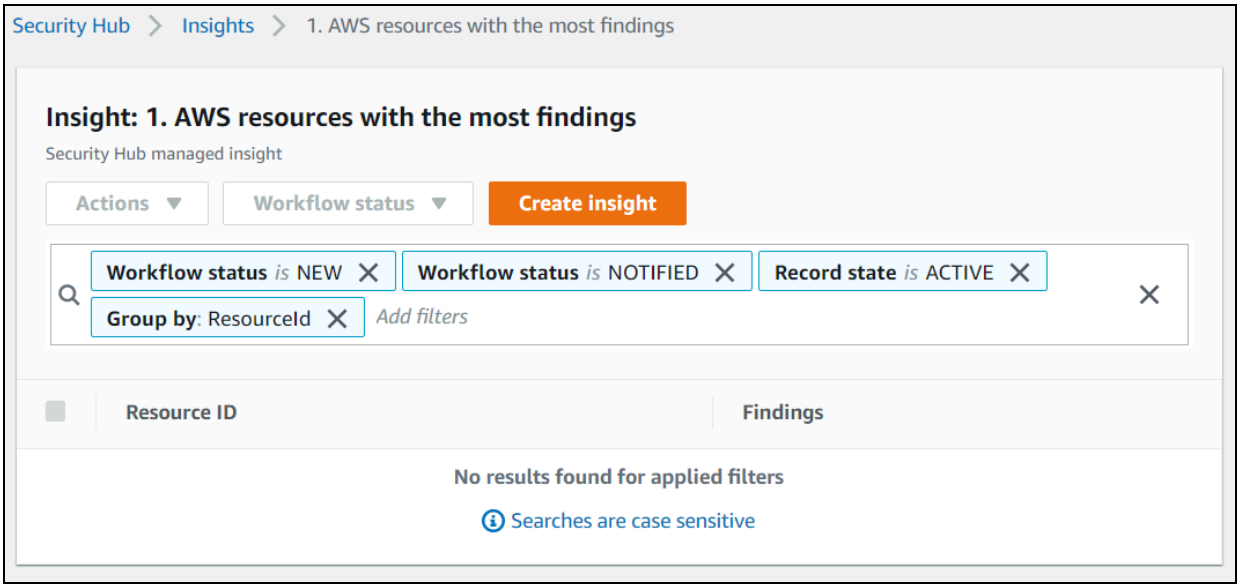


2. You will be redirected to the Security Hub summary page. Click Insights.

3. To view the findings in your AWS resources, click "AWS resources with the most findings".



4. We can now verify if there are any findings in our AWS resources. You can use filters to narrow down the match findings.



Reference:
https://docs.aws.amazon.com/securityhub/latest/userguide/securityhub-insights.html

Amazon Cognito

Amazon Cognito is a user management and authentication service that can be integrated into your web or mobile applications. Cognito allows you to authenticate users through an external identity provider and provides temporary security credentials to access your app's backend resources in AWS or any service behind Amazon API Gateway.

Amazon Cognito works with external identity providers that support:

- SAML or OpenID Connect
- Social identity providers (Facebook, Amazon, Google, Apple)
- Integration of your own identity provider

Instead of using AWS IAM for web identity federations, we suggest that you use Amazon Cognito since it can act as an identity broker. Cognito also provides additional features such as unauthenticated (guest) access and synchronization of user data across mobile devices and web applications.

For example, if you are not using Amazon Cognito as an authentication service, you need to write several lines of code to interact with a web identity provider just to call the **AssumeRoleWithWebIdentity** API. The token you got from the IdP will be traded for AWS temporary security credentials. The credentials you received from AWS will enable you to access AWS resources. But with Amazon Cognito, the service will handle all the federation work for you, and you only need to sign in using the supported external identity providers.

How Does Amazon Cognito Work?

1. A user started to use the application on a mobile device.
2. The user signs in using the supported identity provider.
3. The application will use a Cognito API to exchange the credentials for a Cognito token.
4. The application will exchange the Cognito token for temporary AWS security credentials.
5. The assigned policy in the temporary security credentials will determine what AWS resources can be accessed by the user.

Amazon Cognito User Pools vs Identity Pools

With the proliferation of smartphones in our connected world, more and more developers are quickly deploying their applications on the cloud. One of the first challenges in developing applications is allowing users to log in and authenticate on your applications. There are multiple stages involved in user verification, and most of these are not visible to the end user. AWS provides an easy solution for this situation.

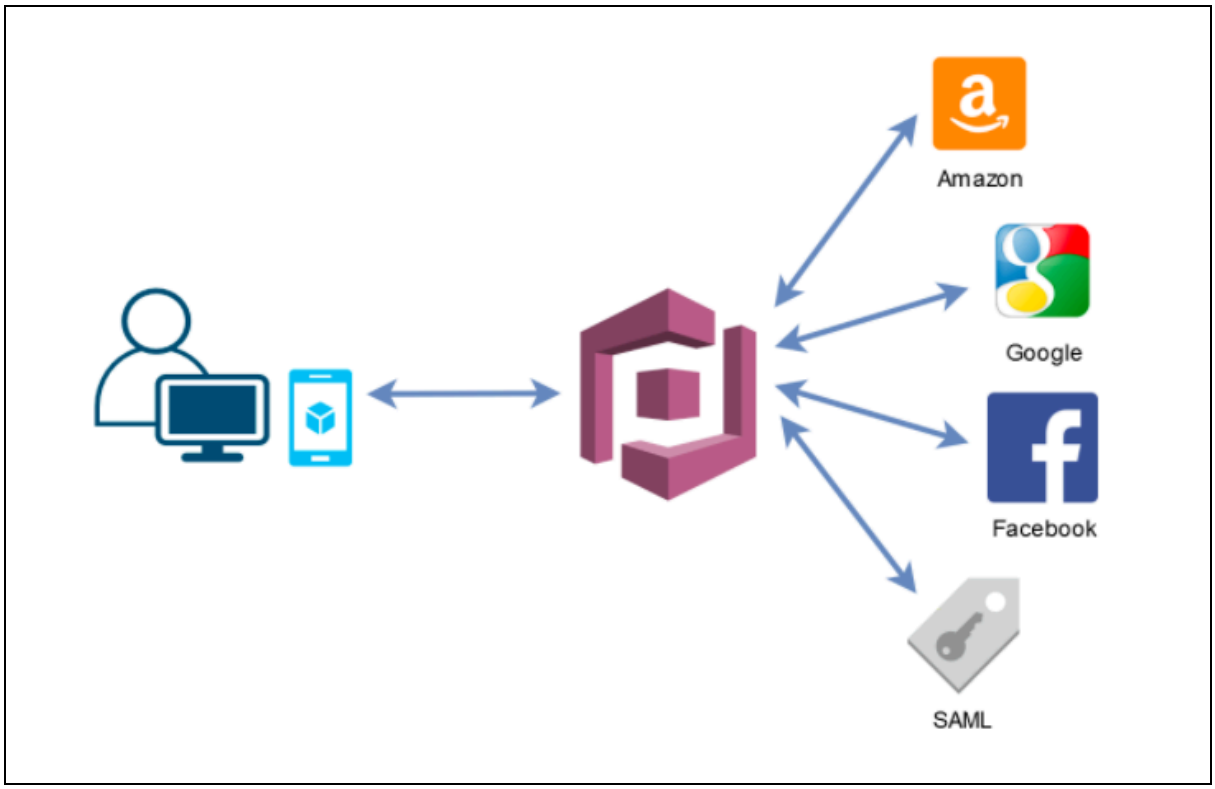
User Identity verification is at the core of Amazon Cognito. It provides solutions for three key areas of user identification:

1. **Authentication** – provides users sign-up and sign-in options. Enables support for federation with Enterprise Identities (Microsoft AD), or Social Identities (Amazon, Facebook, Google, etc.)
2. **Authorization** – sets of permission or operations allowed for a user. It provides fine-grained access control to resources.
3. **User Management** – allows management of user lifecycles, such as importing users, onboarding users, disabling users, and storing and managing user profiles.

In this article, we'll talk about Cognito User Pools and Identity Pools, including an overview of how they are used to provide authentication and authorization functionalities that can be integrated on your mobile app.

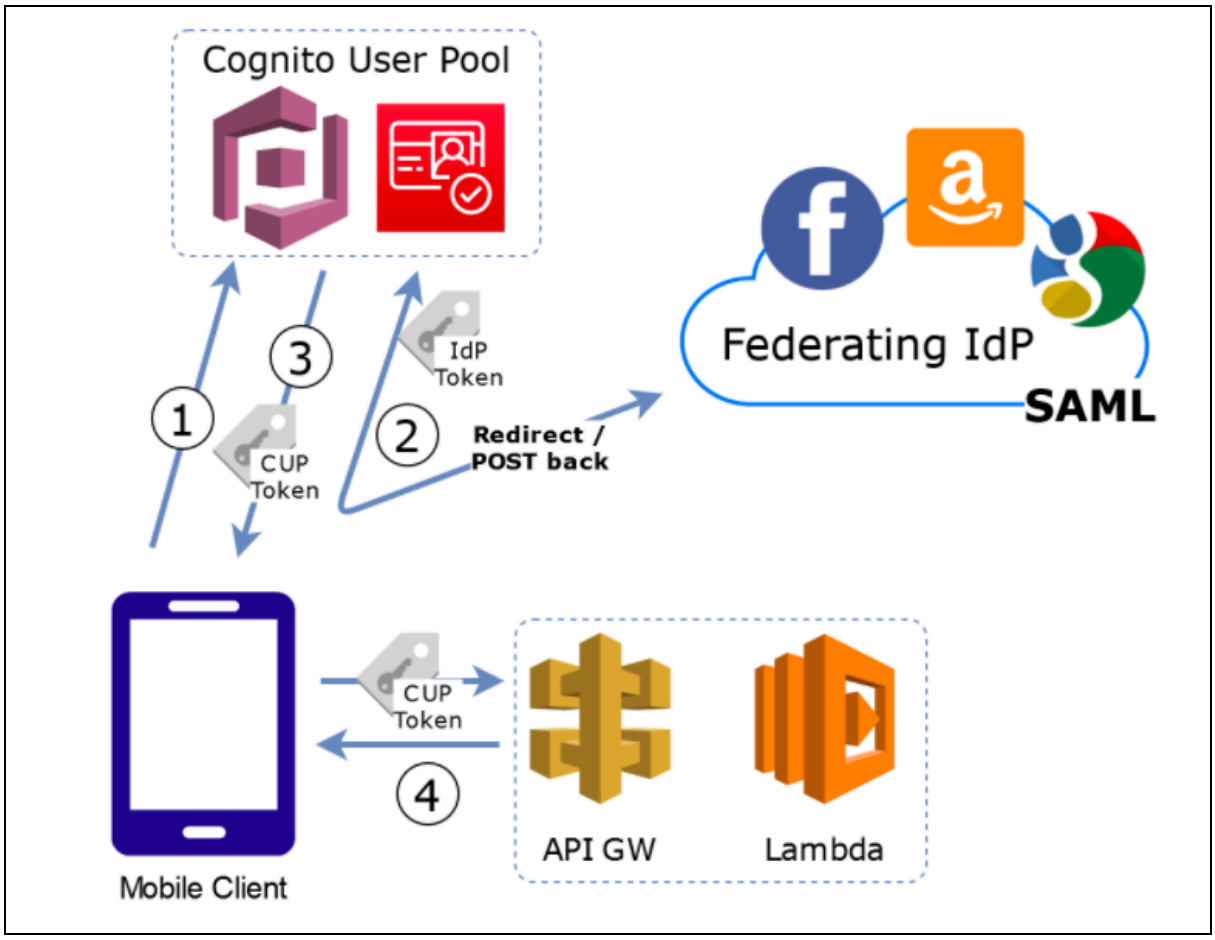
Amazon Cognito User Pools

Amazon Cognito User Pools are used for authentication. To verify your user's identity, you will want to have a way for them to login using usernames/passwords or federated login using Identity Providers such as Amazon, Facebook, or Google or a SAML supported authentication such as Microsoft Active Directory. You can configure these Identity Providers on Cognito, and it will handle the interactions with these providers so you only have to worry about handling the Authentication tokens on your app.



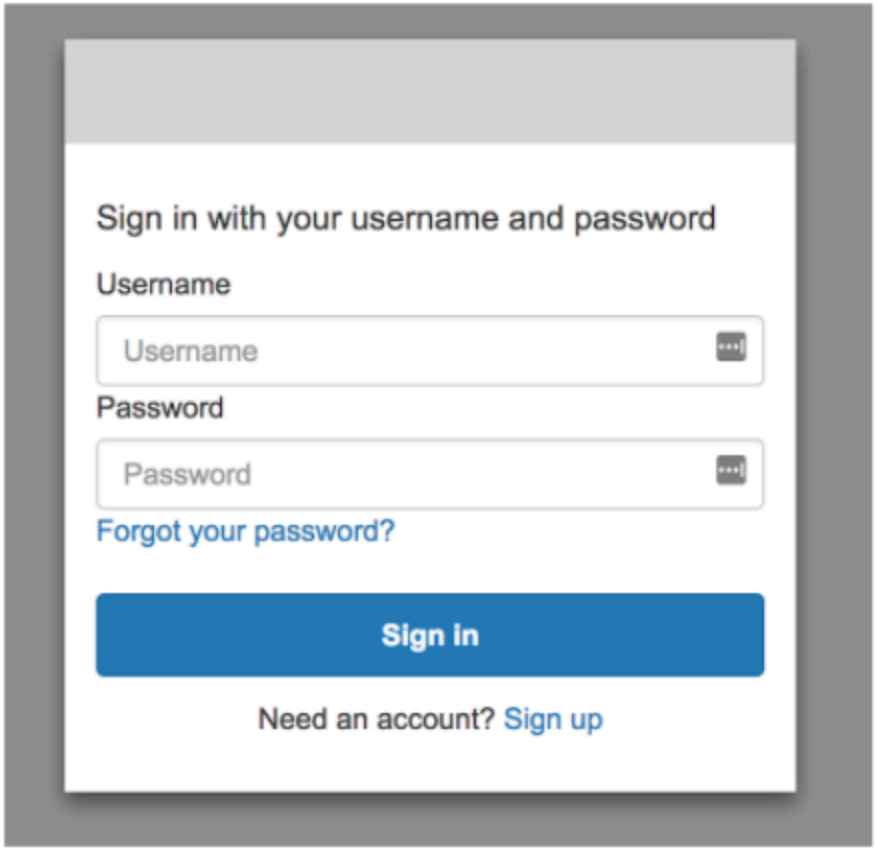
With Cognito User Pools, you can provide sign-up and sign-in functionality for your mobile or web app users. You don't have to build or maintain any server infrastructure on which users will authenticate.

This diagram shows how authentication is handled with Cognito User Pools:



1. Users send authentication requests to Cognito User Pools.
2. The Cognito user pool verifies the identity of the user or sends the request to Identity Providers such as Facebook, Google, Amazon, or SAML authentication (with Microsoft AD).
3. The Cognito User Pool Token is sent back to the user.
4. The person can then use this token to access your backend APIs hosted on your EC2 clusters or in API Gateway and Lambda.

If you want a quick login page, you can even use the pre-built login UI provided by Amazon Cognito which you just have to integrate on your application.



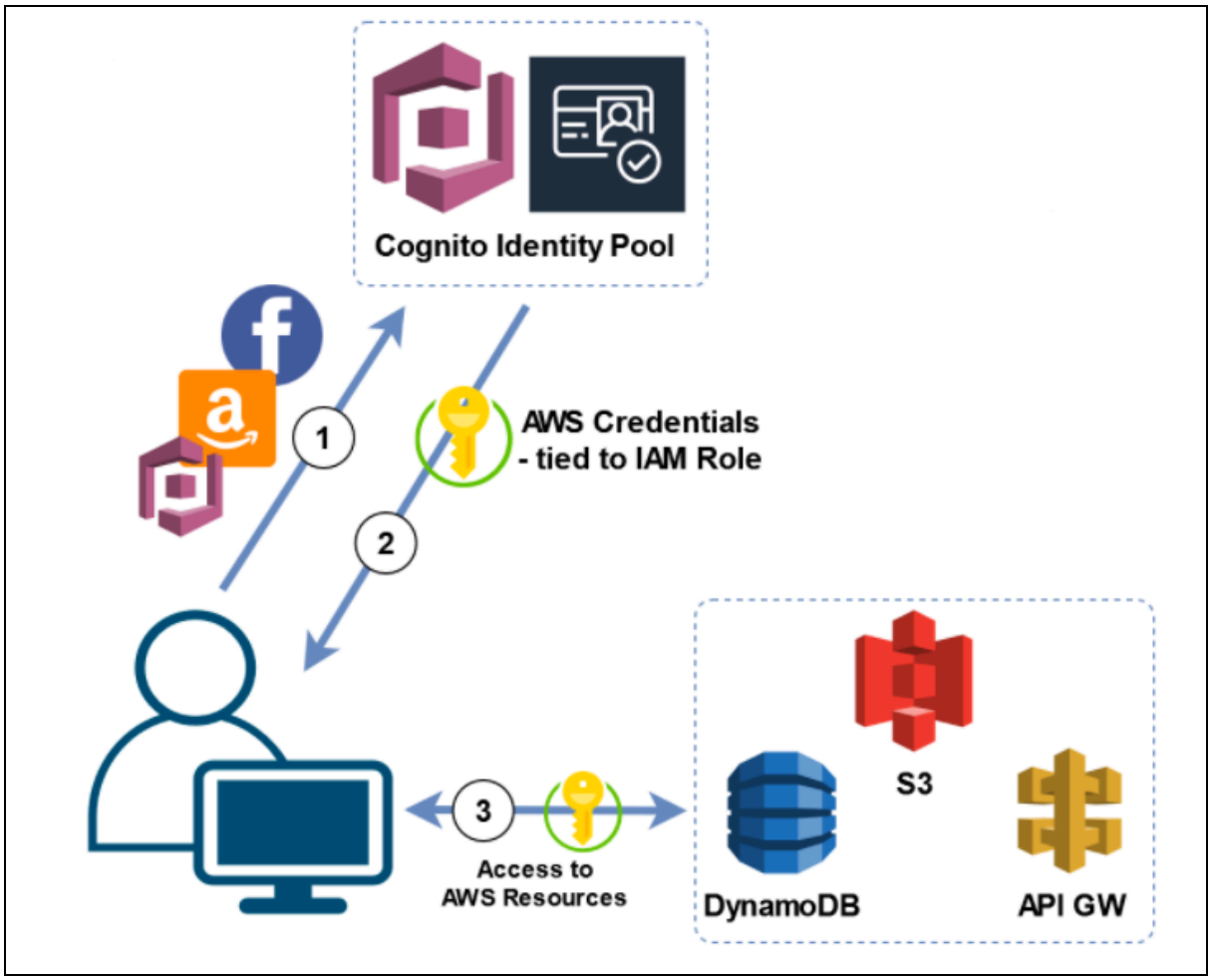
On the Amazon Cognito User Pool page, you can also manage users if you need to. You can reset the password, disable/enable users, and enroll/delete users or other actions needed for User Management.

Amazon Cognito Identity Pools

Cognito Identity Pools (Federated Identities) provide different functionality compared to User Pools. Identity Pools are used for User Authorization. You can create unique identities for your users and federate them with your identity providers. Using identity pools, users can obtain temporary AWS credentials to access other AWS services.

Identity Pools can be thought of as the actual mechanism authorizing access to AWS resources. When you create Identity Pools, think of it as defining who is allowed to get AWS credentials and use those credentials to access AWS resources.

This diagram shows how authorization is handled with Cognito Identity Pools:



1. The web app or mobile app sends its authentication token to Cognito Identity Pools. The token can come from a valid Identity Provider, like Cognito User Pools, Amazon, or Facebook.
2. Cognito Identity Pool exchanges the user authentication token for temporary AWS credentials to access resources such as S3 or DynamoDB. AWS credentials are sent back to the user.
3. The temporary AWS credentials will be used to access AWS resources.

You can define rules in Cognito Identity Pools for mapping users to different IAM roles to provide fine-grain permissions.

Here's a table summary describing Cognito User Pool and Identity Pool:

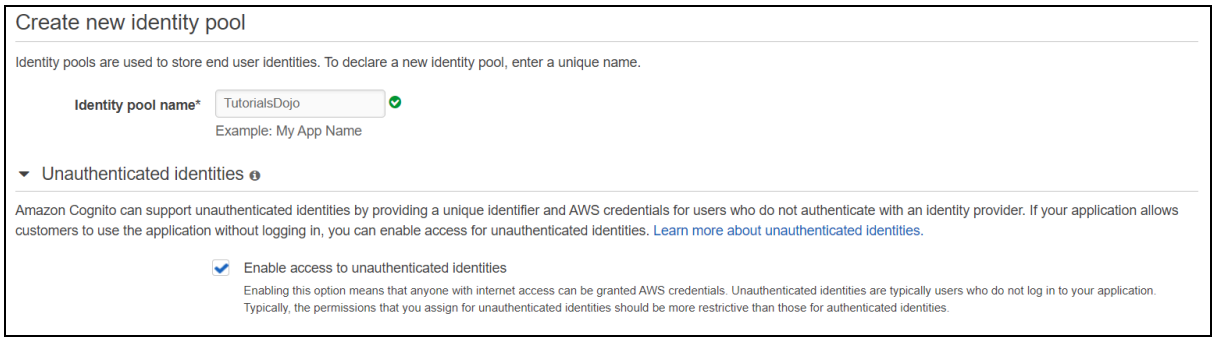
Cognito User Pools	Cognito Identity Pools
Handles the IdP interactions for you	Provides AWS credentials for accessing resources on behalf of users
Provides profiles to manage users	Supports rules to map users to different IAM roles
Provides OpenID Connect and OAuth standard tokens	Free
Priced per monthly active user	

Guest User Authentication

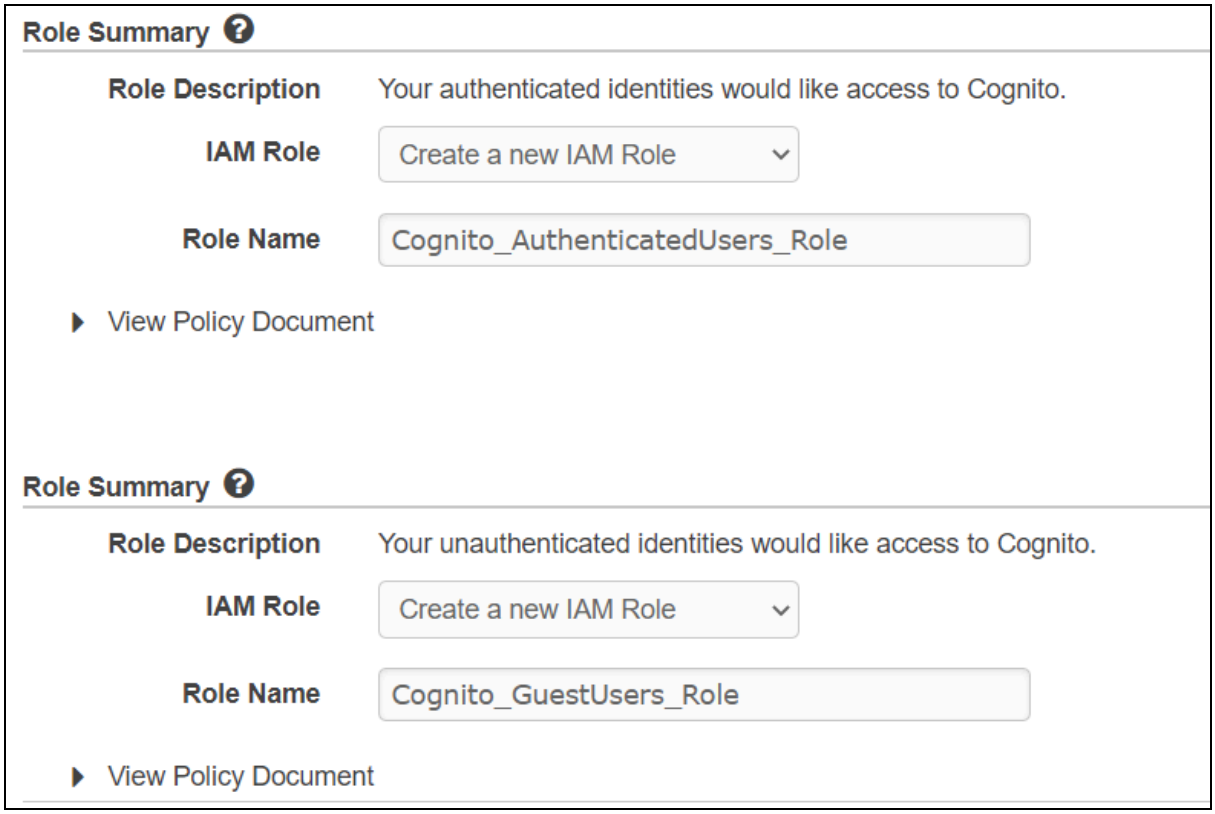
If you have an application that serves media files like images or videos from Amazon S3, you may want to have some control over what your users can and cannot do. For example, you want to allow registered users to upload their own photos so they can share it with their peers, while guest users are only allowed to view photos of a registered profile. You can achieve that functionality on your application via Identity Pool.

AWS Cognito Identity Pool supports two types of identities: *authenticated* and *unauthenticated*. Authenticated identities are users who are authenticated by a trusted public provider (e.g., Amazon, Facebook, Twitter, Cognito User Pool). Unauthenticated identities simply refer to "guest users" or users who don't have to be logged in to access your application.

You can enable unauthenticated identities upon the creation of an Identity Pool or by modifying the setting of an existing Identity Pool.



Cognito Identity Pool with unauthenticated access works by providing a unique identifier and AWS credentials for your guest users. You can control their permissions by defining the policy associated with your unauthenticated identities' role. For example, you can define a read-only access inside the role's policy so any guest users can only view media files from your S3 bucket.

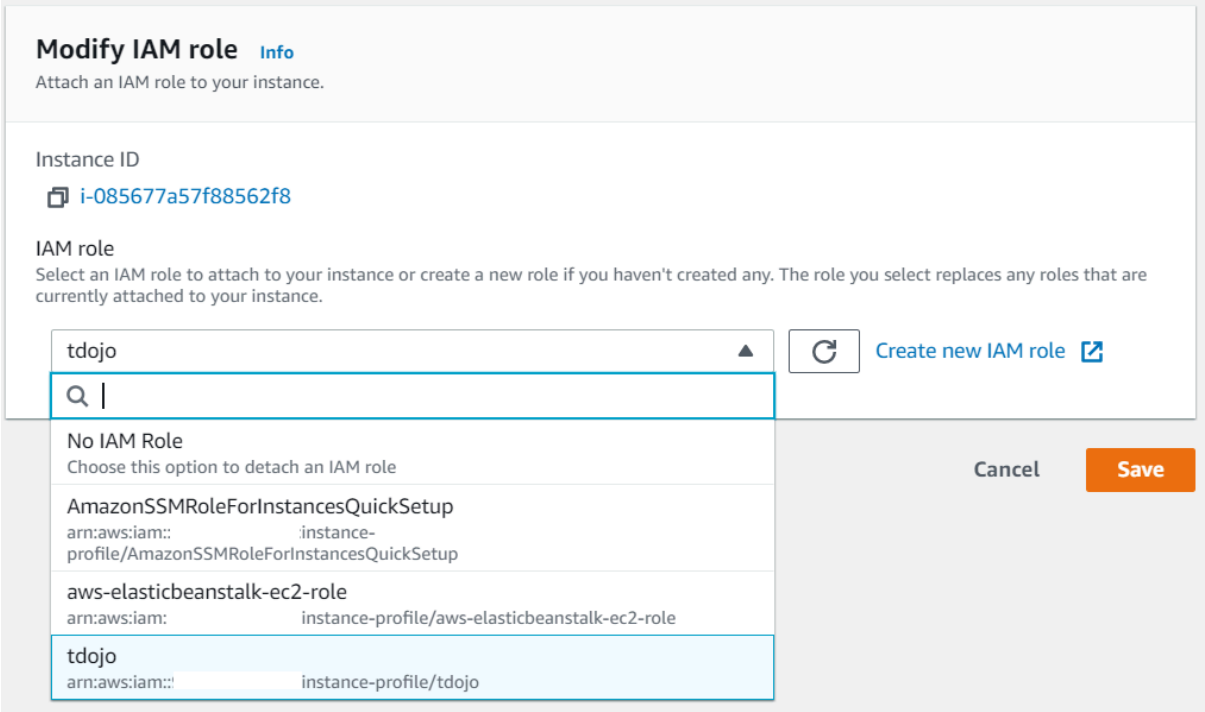


References:
https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-identity-pools.html
https://docs.aws.amazon.com/cognito/latest/developerguide/identity-pools.html#authenticated-and-unauthenticated-identities
https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-identity.html

Managing Instance Profile

IAM roles can be assumed by certain AWS resources to make requests on behalf of a user. Lambda functions that use execution roles (IAM roles) to call AWS APIs are an example of such resources. Unlike Lambda functions, an EC2 instance d. An instance profile is like a container for an IAM role that you want to associate with an EC2 instance so it can make API calls to other AWS services.

You can attach an instance profile to an instance via the EC2 console or AWS CLI. By doing this, we can make API calls without storing any AWS credentials on the EC2.



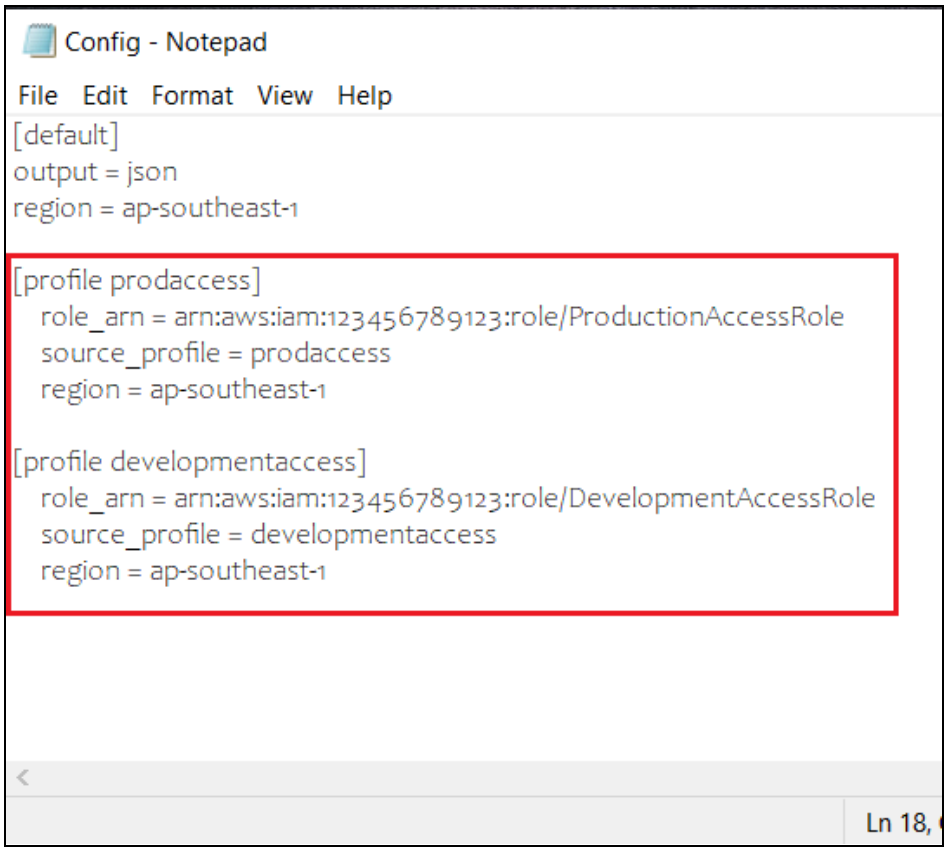
Switching IAM Roles

In some cases, you may want to use a combination of IAM roles and AWS key credentials to allow your EC2 instance to make requests to AWS services via AWS CLI. You might want to take this route to provide yourself with more flexibility in managing different environments with different policies.

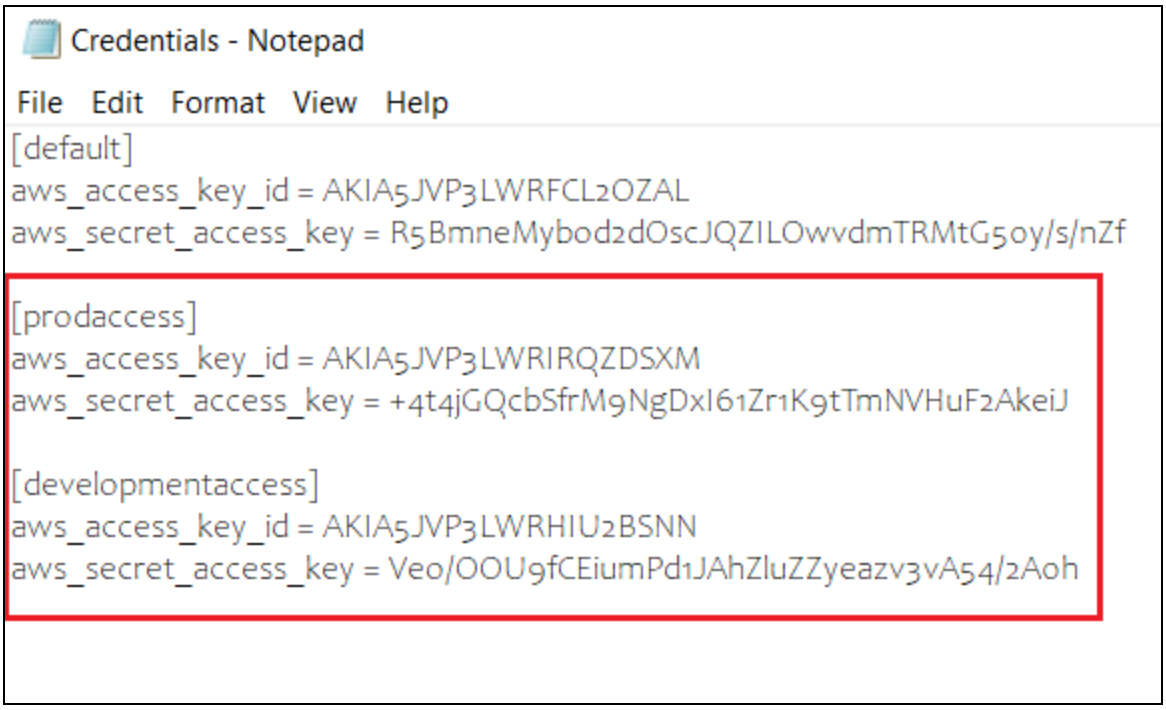
Consider the scenario below:

> *A developer is writing a shell script that calls the AWS CLI to upload objects in an S3 bucket of the development environment. The EC2 instance that is being used contains the access keys and the IAM role used to run the AWS CLI. The Security Administrator gave the developer a new set of the access key credentials with another IAM role that allows access to the production environment. How can the developer easily switch from one IAM role to another?*

In the scenario, we are given access to two environments (production and development) that have different permissions defined in their respective roles. In this case, we can easily switch from one role to another by adding the production profile in the config file of our CLI profile. You can find the file on *.aws/config* in Linux or *C:\Users\<your-user-name>\.aws\config* in Windows.



We will then define the new set of access key credentials for the production environment on the credentials file. The credentials file is located on the same directory as the config file.



Save the file. You can now run any AWS CLI command that is allowed within the `--profile` parameter. In this case, the prodaccess profile will inherit all permissions associated with the ProductionAccessRole.

```
aws s3 ls --profile prodaccess
```

It is important to note that we can only use one profile at a time. It means that whenever we use the *prodaccess* profile, we temporarily lose access to the permissions defined in the *developmentaccess* profile.

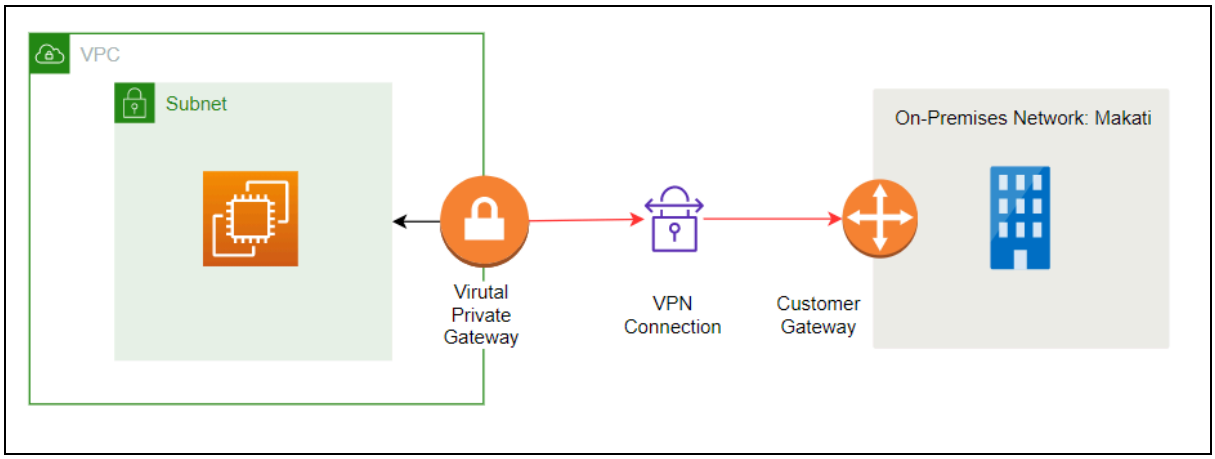
References:
https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-ec2.html
https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-ec2_instance-profiles.html

Using VPN to Protect Employees Who are Connecting to AWS Resources

A Virtual Private Network (VPN) encrypts the connection between computer networks through the public internet.

In AWS, you can establish a VPN connection between your VPC and remote servers in four (4) ways.

AWS Site-to-Site VPN

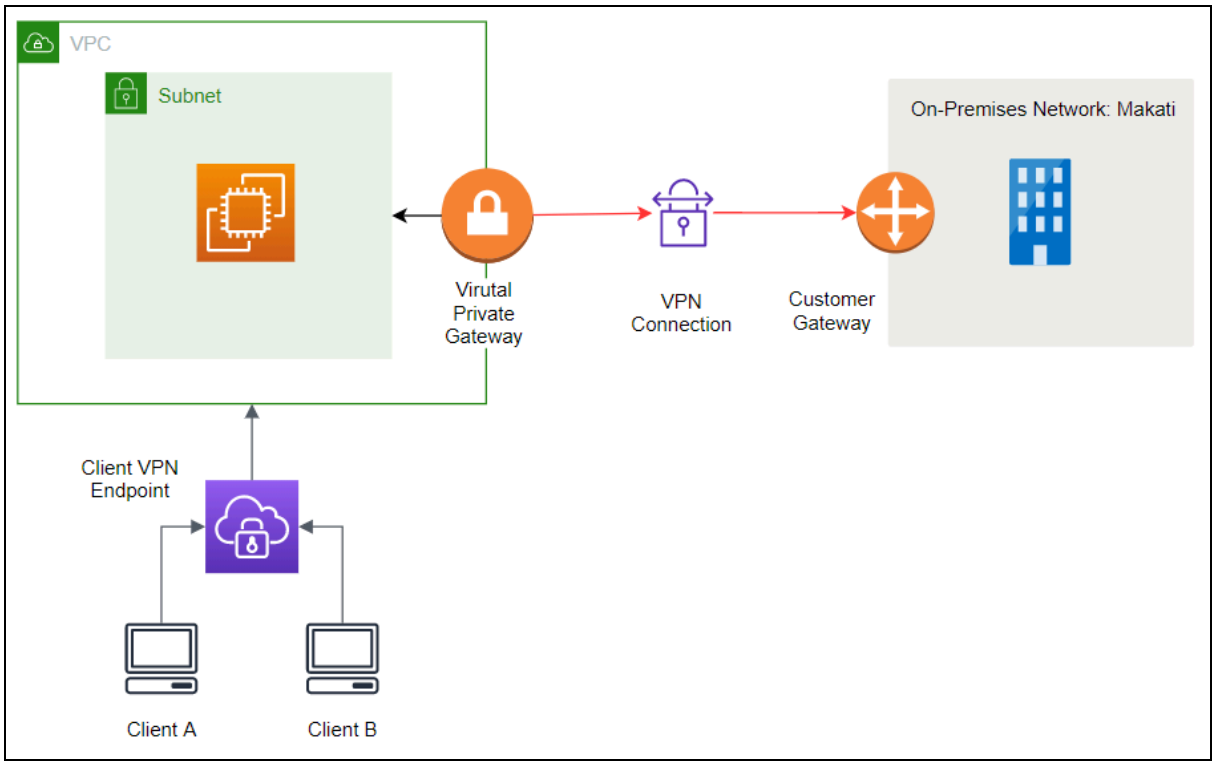


AWS Site-to-Site VPN provides an *IPsec VPN connection* between your VPC and your remote network. First, you need to create and set up a Virtual Private Gateway/Transit Gateway and a Customer Gateway on AWS. Then, you need to download the configuration file from AWS and use that to configure the settings of your customer gateway device (physical device on your end).

Use Case:

- Given the fact that IPsec is more complicated to set up, albeit more secure than SSL VPN, consider using AWS Site-to-Site VPN if a more secure connection is preferred.
- AWS Site-to-Site VPN is desirable in situations where users are working on-site. Note that users away from the office won't be able to connect to the VPC because they need to be connected to the Customer Gateway Device (corporate router).

AWS Client VPN

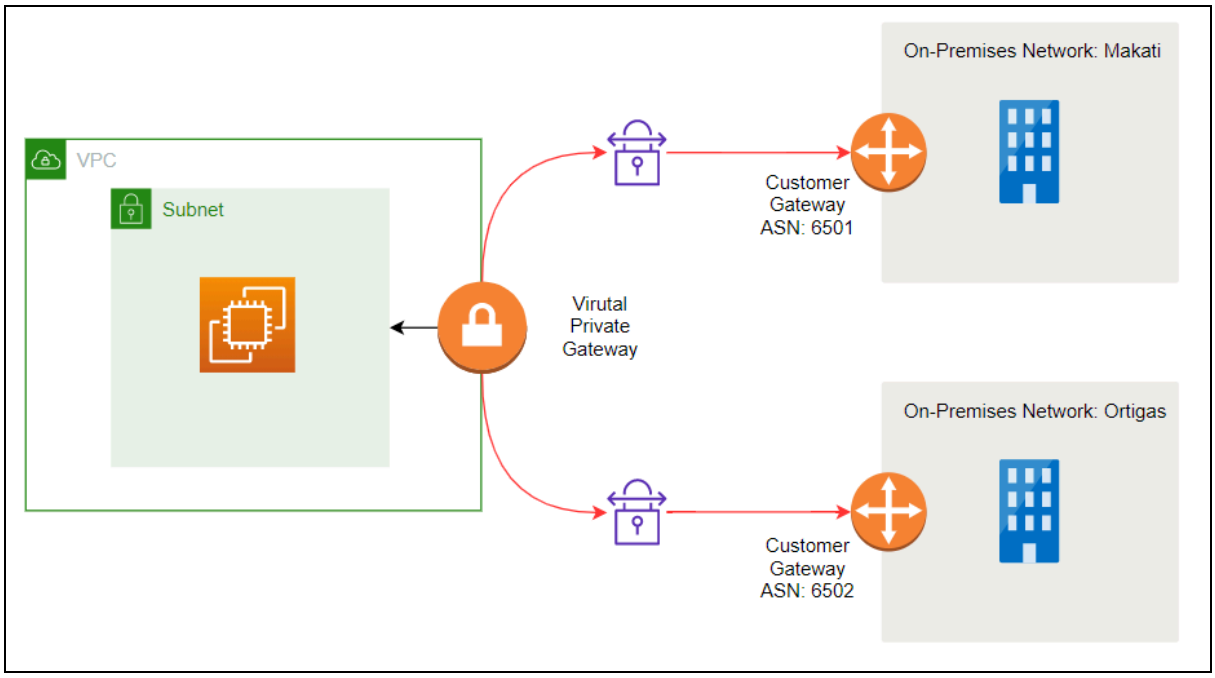


AWS Client VPN is a managed client-based VPN that is offered as a service. Like any managed service, things like high availability and scalability are already taken care of by AWS. To establish a secure TLS VPN session, you simply need to expose a VPN endpoint to which your users can connect.

Use Case:

- Unlike Site-to-Site VPN, users from any location can connect and access resources from your Amazon VPC or on-premises network using an OpenVPN-based VPN client. So, if you like to trade the more secure IPsec connection over the flexibility of having a connection from anywhere in the world then use AWS Client VPN.
- If you want less operational management.

AWS VPN CloudHub

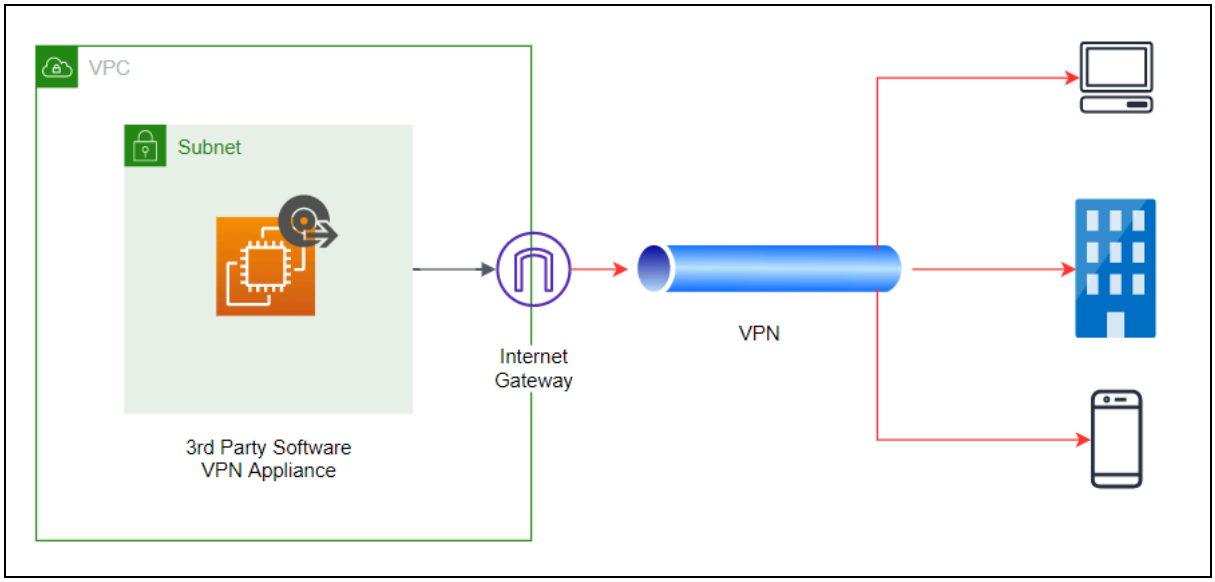


AWS VPN CloudHub is a hub-and-spoke VPN that allows communication between two or more on-premises networks situated at different locations. You must create multiple customer gateways with the public IP address of your gateway. And use a unique Border Gateway Protocol (BGP) Autonomous System Number (ASN) for each customer gateway.

Use case:

- If you want to establish a secure communication between multiple private networks that use Site-to-Site VPN connections via your Virtual Private Gateway.

Third Party Software VPN Appliance



Optionally, you can launch an EC2 instance to host a third-party software VPN appliance. Unlike AWS Client VPN, you have full control over the software, instance, and the responsibilities that come with it, like installing updates and patches. You can find a software VPN appliance from open source communities, AWS partners, or AWS Marketplace.

Use Case:

- If you prefer a particular software VPN appliance and want full control over it.
- If you have users who access your AWS resources from different locations (office, home, employees on travel).

References:
https://docs.aws.amazon.com/vpn/latest/s2svpn/VPC_VPN.html
https://docs.aws.amazon.com/vpc/latest/userguide/vpn-connections.html

AWS Direct Connect

Instead of relying on the public internet as the medium for linking your VPC and on-premises network, you may use AWS Direct Connect as an alternative. AWS Direct Connect is a service that lets you establish a dedicated network path between your local data center and VPC via a physical ethernet connection. This has a number of advantages over a public-based connection, including a more stable low-latency network experience, higher

throughput at a discounted transfer rate, and an inherently secure connection. As opposed to AWS VPN, Direct Connect is more arduous to implement, requires extensive planning, and is not something that can be set up overnight.

Encryption in AWS Direct Connect

Data sent over a Direct Connect connection are not encrypted by default. To encrypt traffic, you can either create an IPsec tunnel on top of Direct Connect or set up MacSec encryption.

IPsec VPN over Direct Connect (AWS VPN Site-to-Site)

- Protects at Layer 3 (Network layer) of the OSI model
- Must be set up over a public virtual interface of the Direct Connect
- A single VPN tunnel can reach a maximum throughput of 1.25 Gbps. To scale beyond this limit, you must aggregate multiple VPN tunnels, which can be difficult to set up and manage.

MAC Security (MACSec) over Direct Connect

- Protects at Layer 2 (Data link) of the OSI model
- MacSec encryption only works on **10Gbps and 100Gbps Dedicated Connections**.
- Suited for high-bandwidth workloads that require a near-line rate point-to-point encryption.
- All network circuits involved in the path of the traffic must have MACSec encryption support.

References:
https://docs.aws.amazon.com/directconnect/latest/UserGuide/MACsec.html
https://docs.aws.amazon.com/whitepapers/latest/aws-vpc-connectivity-options/aws-direct-connect-vpn.html

AWS Verified Access

AWS Verified Access is another service that enables secure, zero-trust access to AWS resources without relying on traditional VPNs. It continuously validates both the user and device before granting access, integrating with identity providers (SAML, OIDC, IAM Identity Center) to enforce authentication and authorization policies. Verified Access also checks device compliance and provides secure, conditional access for remote employees, contractors, and third-party users.

Key Features:

- **Zero-Trust Access** – Access is granted based on continuous verification of the user's identity, device posture, and network context, rather than solely relying on network location.

- **IdP Integration** – Works with AWS IAM Identity Center, SAML, and OIDC IdPs to enforce authentication and authorization policies across users and devices.
- **Device Posture Enforcement** – Checks device compliance (such as OS version, patch level, or security software) before granting access to AWS resources.
- **Secure Access Without VPN** – Reduces dependency on traditional VPNs by providing secure, identity- and device-aware access to AWS applications and services.

References:
https://docs.aws.amazon.com/verified-access/latest/ug/what-is-verified-access.html
https://docs.aws.amazon.com/verified-access/latest/ug/how-it-works.html

Penetration Testing in AWS

Penetration testing or pen testing is an authorized simulated cyberattack against a computer system. It is done to evaluate the system's security level and check for potential bugs or vulnerabilities that could harm the system.

Pen testing is vital to Security Specialists because it helps them expose the weak points of their newly implemented or existing security controls. By determining those weak points, they could right away tell what adjustments should be made and identify whether a security solution is viable or not for a particular event.

"Can I conduct penetration testing in my AWS environment?" - Yes, but it depends.

Pen testing in the Cloud might be slightly different than what you're used to in your on-premises environment. By now, you should already have a good grasp of the shared responsibility model, which basically explains the relationship between the Cloud Provider and its customer in terms of which of the two has control over what resources. Because of this shared responsibility, you are limited to security assessments/pen-testing strategies that you can carry out.

According to AWS, customers are **permitted** to perform penetration tests on these services **without prior approval** (*you do not need to submit any penetration test request form*):

- EC2 instances, NAT Gateways, and Elastic Load Balancers
- Amazon RDS
- Amazon CloudFront
- Amazon API Gateway
- Amazon Aurora
- AWS Lambda and Lambda@Edge Functions
- Amazon Lightsail resources
- Amazon Elastic Beanstalk environments

Customers are **prohibited** from doing the following tests:

- 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)

Note that the simulation of a DDoS attack is prohibited for regular customers. However, It is still possible to conduct DDoS tests as long as you're an AWS Partner Network (APN) Partner, and you have approval from AWS to conduct DDoS tests.

If you have other simulation events that you want to run, you can submit a request to AWS. A reply may take up to 7 days after a security team has reviewed your request. Lastly, it is imperative to know that any damages done to AWS or other AWS customers are under your responsibility.

Security Specialty Exam Notes:

You don't have to submit a request form to conduct penetration tests on permitted AWS resources.

Reference:
https://aws.amazon.com/security/penetration-testing/

AWS Billing Permissions

AWS Organizations enables you to set up a single payment method for all the AWS accounts in your organization through **consolidated billing**. With consolidated billing, you can see a combined view of charges incurred by all your accounts, as well as take advantage of pricing benefits from aggregated usage, such as volume discounts for EC2 and S3.

AWS Billing (service prefix: `aws-portal`) provides the service-specific resources, actions, and condition context keys for use in IAM permission policies. You can specify the following actions in the Action element of an IAM policy statement:

- **ModifyAccount** - Allow or deny IAM users permission to modify Account Settings.
- **ModifyBilling** - Allow or deny IAM users permission to modify billing settings.
- **ModifyPaymentMethods** - Allow or deny IAM users permission to modify payment methods.
- **ViewAccount** - Allow or deny IAM users permission to view account settings.

- **ViewBilling** - Allow or deny IAM users permission to view billing pages in the console.
- **ViewPaymentMethods** - Allow or deny IAM users permission to view payment methods.
- **ViewUsage** - Allow or deny IAM users permission to view AWS usage report

Suppose your company is handling multiple AWS accounts. In that case, it is preferable to use AWS Organizations so you can aggregate the bill of each account into a single master account that pays the bill of all its member accounts. Not only does it make the life of financial auditors easier, but implementing the principle of least privilege also becomes a whole lot simpler.

To understand what I mean, consider the following scenario:

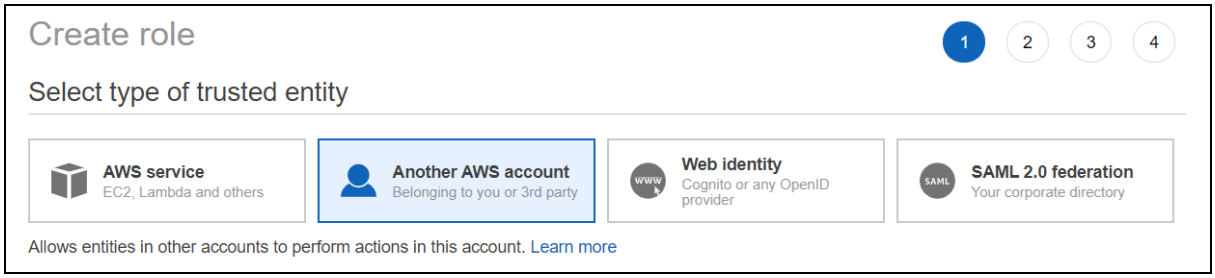
A company has 50 AWS accounts that are consolidated using AWS Organizations. The accountants from the finance department log in as IAM users in the AWS Finance account. The finance team members need to read the consolidated billing information in the AWS master account that pays the charges of all the member (linked) accounts. The required IAM access to the AWS billing services has already been provisioned in the master account. The Security Officer should ensure that the finance team is not able to view any other resources in the master account.

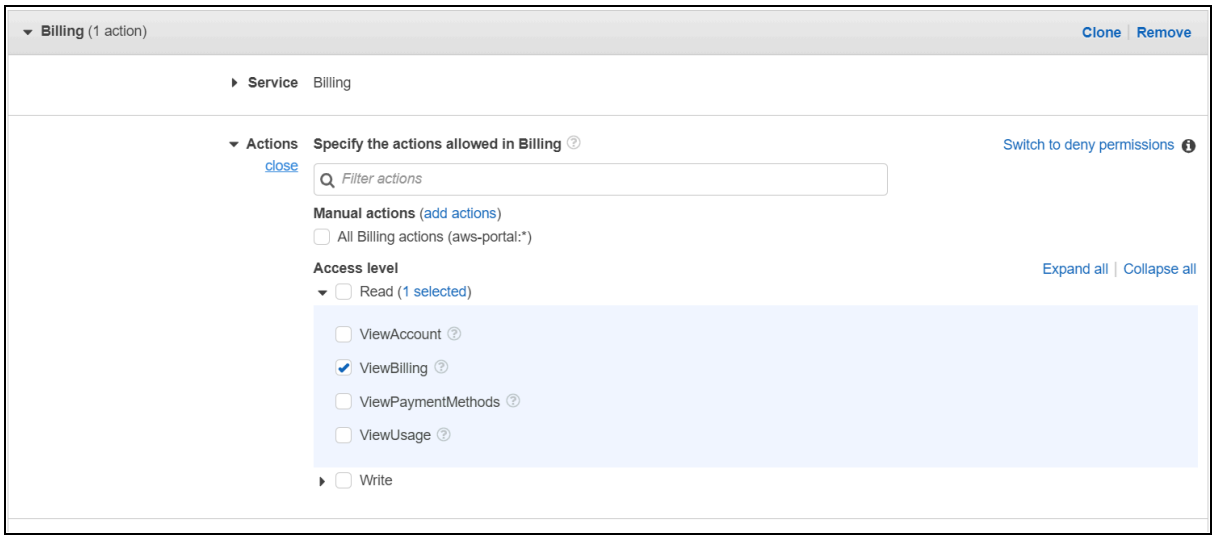
Essentially, the scenario requires the application of the principle of least privilege when granting access for the accountants from the finance department. To properly implement this, you must first need to determine the minimum set of actions needed for the accountants to do their job properly.

According to the scenario, the finance department should be allowed to read consolidated billing information in the Master Account. From the list of AWS Billing permissions that we saw earlier, the ***ViewBilling*** permission will fulfill the required permissions to do the task.

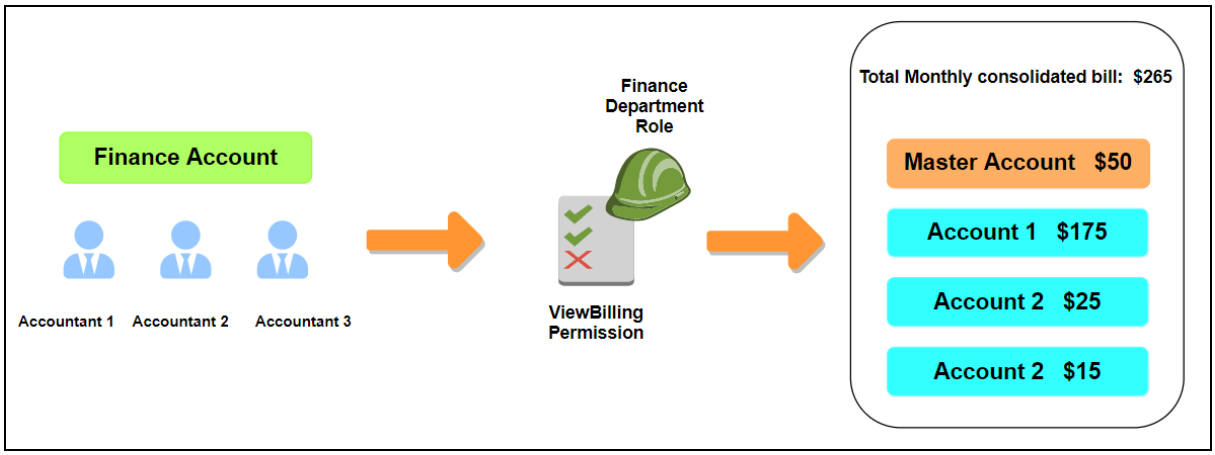
We can delegate access to the Finance Account by doing the following steps:

1. Set-up an IAM role **in the master account** with the ViewBilling permission.





2. Grant the finance users from the Finance account the permission to assume the role.



Reference:

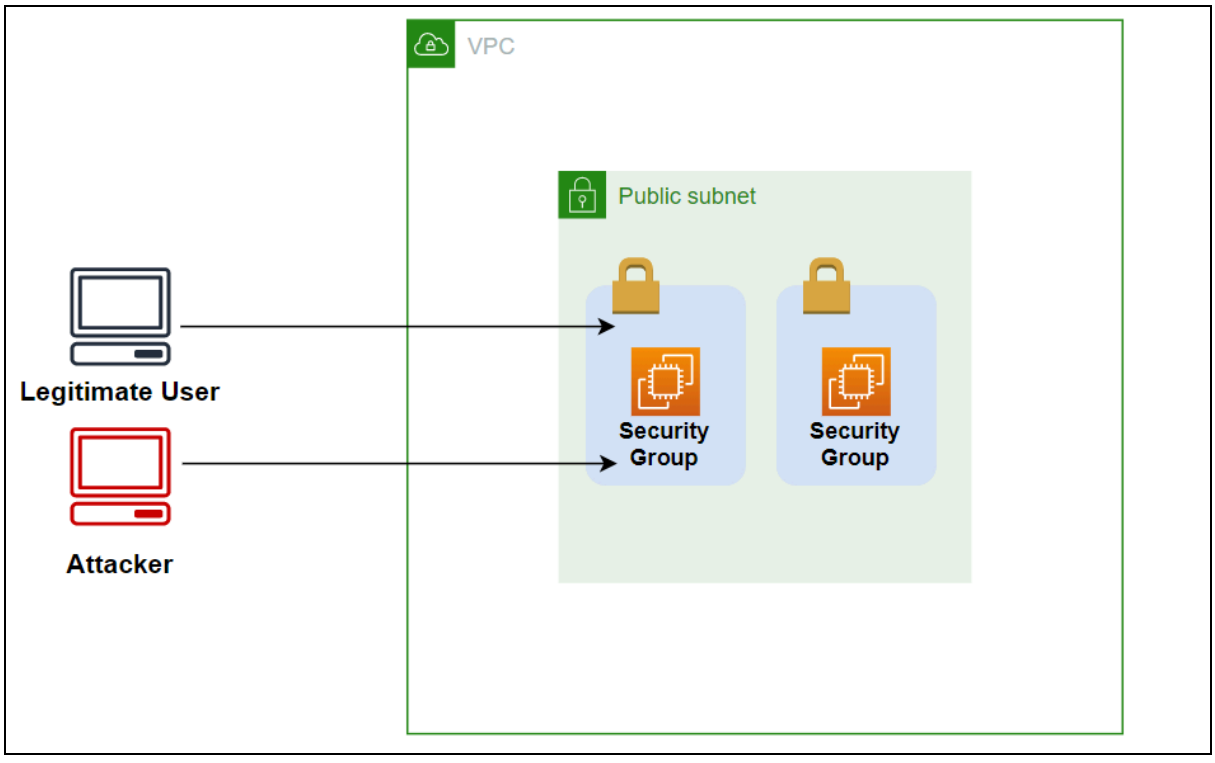
https://docs.aws.amazon.com/IAM/latest/UserGuide/list_awsbilling.html
https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/billing-permissions-ref.html
https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/billing-example-policies.html

Preventing Staff from Adding Rules to Security Groups without any Approval

In the absence of firewall software in your instance, security Groups act as the last line of defense of your application. For this reason, it is of utmost importance to restrict the connections to your instances to only authorized users/services. Security Groups are stateful. It means that return traffic is automatically allowed, regardless of what's specified in the outbound rule.

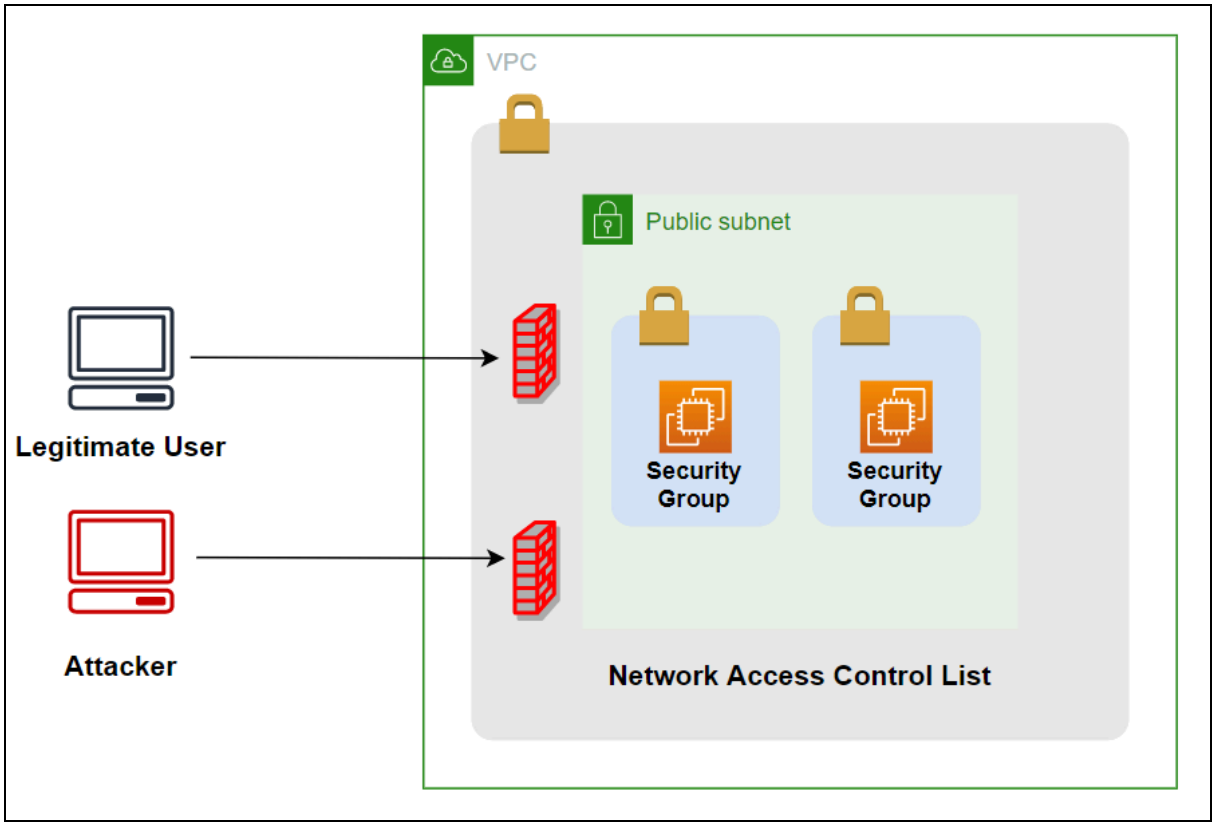
According to Shred-It, an information security company, employee negligence is the most common cause of security issues. As a case in point, a developer who needs to do some quick testing might carelessly add an inbound rule that allows SSH connection from anywhere. It might sound like a simple matter, but it could bring grave consequences if not addressed, especially if done habitually. As a Security Specialist, you have to anticipate that kind of scenario to mitigate the risks that come with it.

Attackers could take advantage of a port exposed to the Internet. For instance, leaving port 22 (SSH) open to public gives botnets the chance to exhaust your instance's resources through brute-force attacks. If they're lucky, they might even figure out your SSH credentials and have access to your instance.



Bad Way Of Fixing it

Since the problem is at the network level, it is tempting to fix the issue through direct network configuration. For example, the exam might trick you into choosing a Network Access Control List (NACL) to filter out source IP addresses matching the 0.0.0.0/0 CIDR range. Although it is possible, it is not practical. By doing so, you will block all incoming requests, including legitimate ones. Other applications hosted in your VPC might stop operating properly as well.

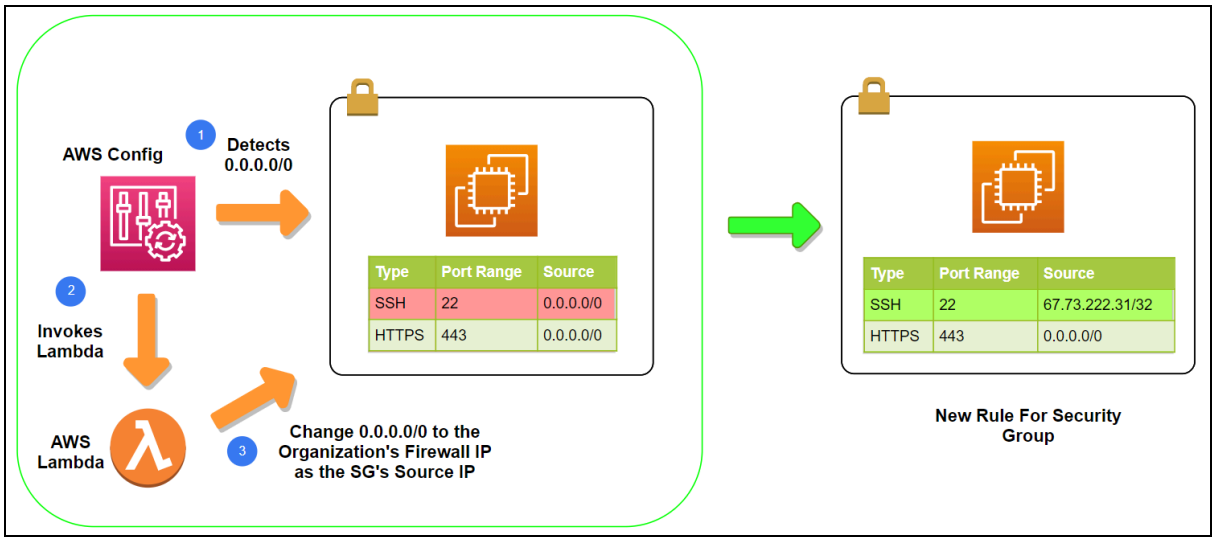


Right Way Of Fixing it

To solve this problem and to prevent any users from adding a 0.0.0.0/0 source IP address to a critical security group in the future, we can rely on AWS Config.

There is an AWS Config managed rule called ***vpc-sg-open-only-to-authorized-ports*** that checks whether the security group with 0.0.0.0/0 of any Amazon Virtual Private Cloud (Amazon VPC) allows only specific inbound TCP or UDP traffic. The rule is COMPLIANT if there is a security group with inbound 0.0.0.0/0 with no ports provided in the parameters.

The image below illustrates an overview of the solution. AWS Config detects security groups that allow 0.0.0.0/0 access to specific ports (in this case, port 22). A Lambda function is triggered when such an event happens. The function is programmed to update the security group's source IP for SSH to your organization's firewall IP address.



References:

https://docs.aws.amazon.com/config/latest/developerguide/vpc-sg-open-only-to-authorized-ports.html
https://docs.aws.amazon.com/config/latest/developerguide/evaluate-config_develop-rules_nodejs-sample.html
https://aws.amazon.com/blogs/security/how-to-monitor-aws-account-configuration-changes-and-api-calls-to-amazon-ec2-security-groups/
https://www.shredit.com/en-us/about/press-room/press-releases/sacking-employees-for-data-breach-negligence

Setting up Intrusion Prevention System (IPS) and Intrusion Detection System (IDS) Software

Intrusion Detection System (IDS) monitors network and system traffic for suspicious activity. The identified threats are reported to systems administrators or a security information and event management (SIEM) system.

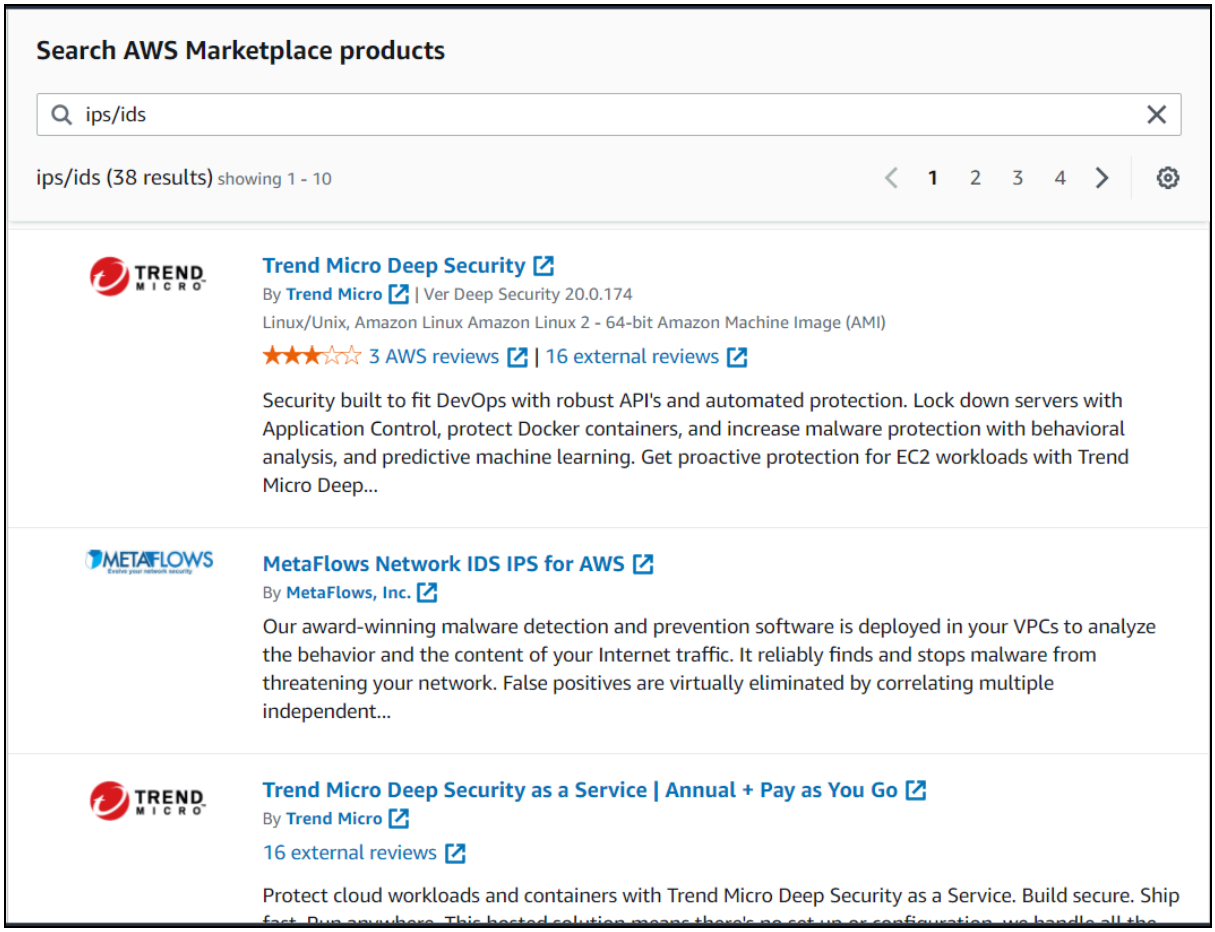
Intrusion Prevention System (IPS) provides an additional security layer by proactively denying network traffic from known security threats or dropping malicious packets.

IPS/IDS solutions help protect your EC2 instances by detecting vulnerabilities and responding to attacks. It could also protect your applications by using next-generation firewalls, which inspect and prevent malicious traffic at an application level.

How is it different from GuardDuty?

Amazon GuardDuty is only a threat detection service. Therefore, it is not a true IPS/IDS solution. However, you still could implement your own "IPS" with GuardDuty by triggering a Lambda Function that executes your own set of remedial actions. An IPS/IDS solution does a little bit more than that. Its main selling point is its automated response. You can save time stopping an attack since an IPD/IPS solution can carry out automatic mitigation. Also, it can help you prevent brute-force attacks from hundreds of sources. With configuration management tools like Ansible or AWS Systems Manager, you can easily manage the IPS/IDS software installation in several instances.

In AWS, you have the option to rent custom IPS/IDS software that fits your needs from the AWS MarketPlace. You can save on costs by leveraging on the pay-as-you-go pricing model of AWS.



Reference:
https://aws.amazon.com/mp/scenarios/security/ids/

Connecting On-premises Active Directory to AWS Managed Microsoft Active Directory

If you want to run AD-aware applications in AWS, such as a SQL server or .NET application and you wish to manage them from an Active Directory hosted on-premises, AWS Managed Microsoft Active Directory can help simplify the process. Integrating two active directories requires a trust relationship. Since the exam focuses on AWS Services, you'd likely encounter questions about setting up the trust relationship between an on-premises active directory and the AWS active directory.

Trust Relationship For Active Directory

The Trust Relationship describes how the users in one domain can access resources in another domain.

There are three trust relationship directions:

1. **One-way:incoming** - the trusting domain allows users from the trusted domain to access its resources. In other words, if Domain A trusts Domain B (incoming trust), then users from Domain B can access resources in Domain A.

2. **One-way:outgoing** - the trusting domain's users can access resources in the trusted domain. If Domain A has an outgoing trust with Domain B, users from Domain A can access resources in Domain B.

3. **Two-way (Bi-directional)** - users from each domain can access resources in the other domain. If Domain A and Domain B have a two-way trust, users from both domains can access resources in either domain.

Remember that the direction of trust is the opposite of the direction of access. When establishing trust, it's crucial to understand the roles of the 'trusting' and 'trusted' domains. The direction of trust and access can sometimes seem counterintuitive:

1. **Trusting Domain**: This is where the trust is placed. It's like a house that permits visitors.
2. **Trusted Domain**: This is the source of those visitors. It's like a neighbor being allowed into the house.

If you want users from an on-premises Active Directory (the neighbor) to access AWS resources (enter the house), the AWS Active Directory becomes the 'trusting' domain. It's essentially saying, 'I trust users from the on-premises AD to access my resources. Conversely, the on-premises AD is the 'trusted' domain because it provides the users that are granted access.

When you configure a one-way outgoing trust from the on-premises Active Directory to the AWS Active Directory, you are allowing users from your on-premises environment to authenticate and access resources protected by the AWS Active Directory. This setup ensures that while on-premises users can access AWS

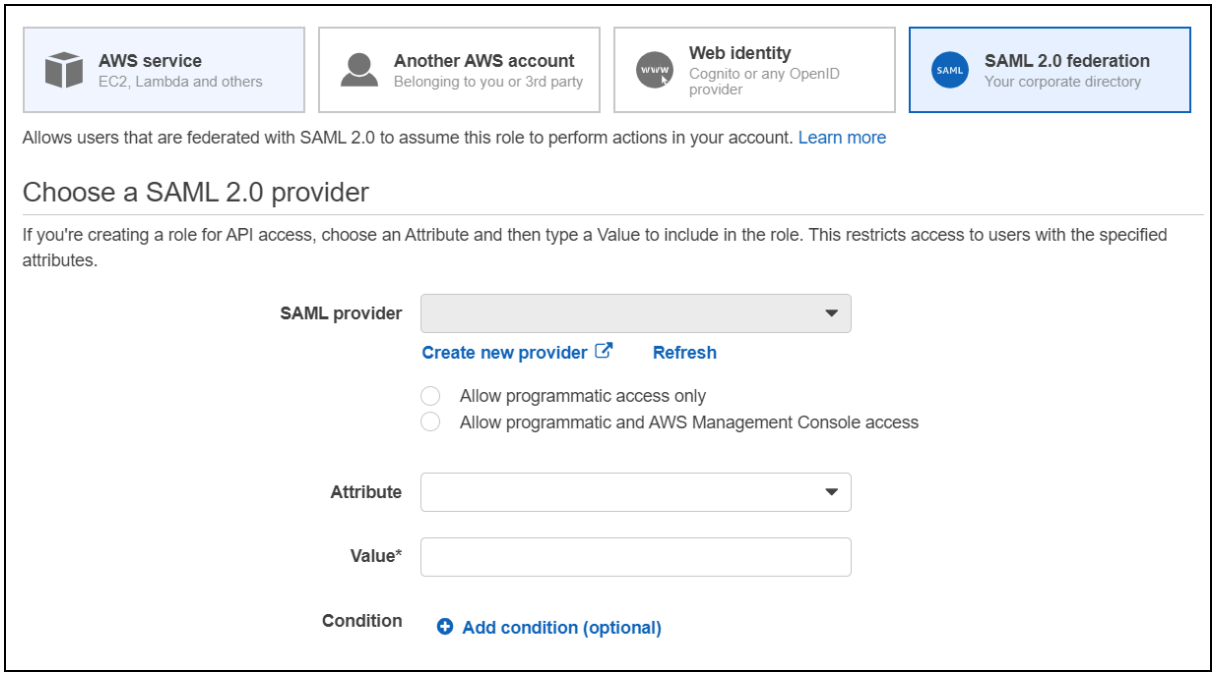
resources, users defined in the AWS Active Directory won't have permissions to access resources back in the on-premises AD, unless a separate trust is established in the opposite direction

References:
https://docs.aws.amazon.com/directoryservice/latest/admin-guide/ms_ad_setup_trust.html
https://aws.amazon.com/blogs/security/now-available-simplified-configuration-of-trust-relationships-in-the-aws-directory-service-console/
https://docs.aws.amazon.com/directoryservice/latest/admin-guide/ms_ad_tutorial_setup_trust.html

Setting up Single Sign-On Access Between On-premises Active Directory and AWS using ADFS and IAM Identity Center (previously known as AWS SSO)

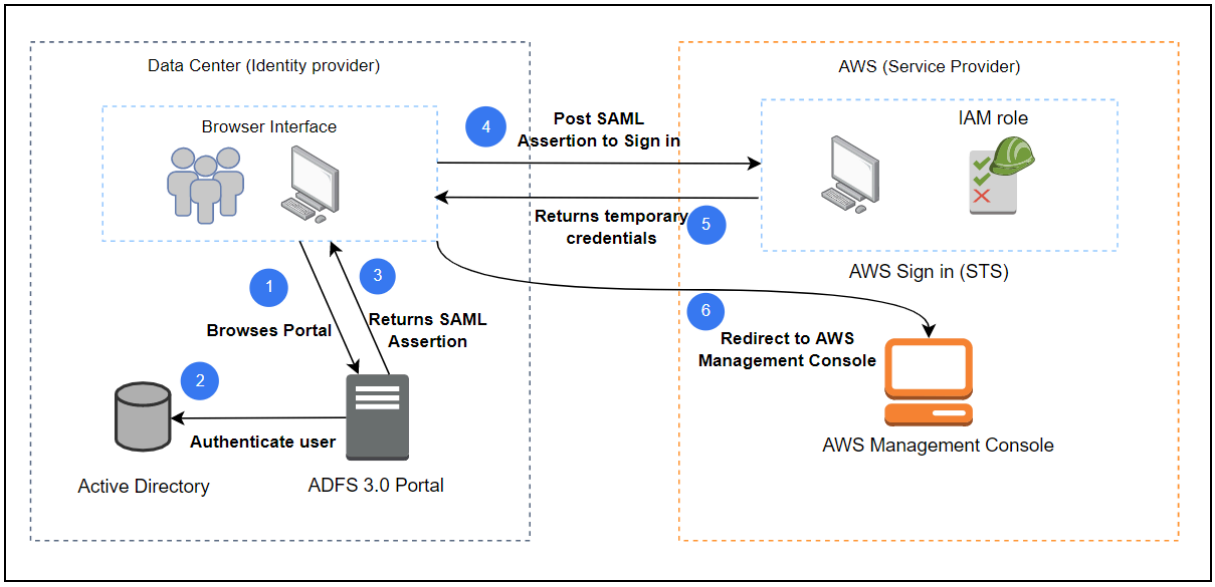
Setting up ADFS and IAM Identity Center (previously known as AWS SSO) provides a bridge between your on-premises Active Directory and cloud resources, enabling users to have a unified login experience. This is often referred to as Single Sign-On (SSO). The core idea of SSO is to authenticate once and gain access to multiple systems without needing to log in again.

If your organization has an existing user and policy scheme built in an on-premises Active Directory, you may use that instead of recreating it with IAM users. IAM supports external users authenticated by a SAML-compliant Idp to access the AWS Management Console or execute programmatic calls to AWS services. To establish federated access, you need to configure your on-premises Active Directory Federation Services (ADFS) to add a relying party trust between Active Directory and AWS. Then, you need to create an IAM role that will be assumed by the federated users. Attach the permissions needed by the federated users to the role plus the ***AssumeRoleWithSAML*** STS permission to generate temporary credentials. These credentials are what enable a federated user to make requests to AWS.



For example, suppose you have a Development and an Admin Group in your Active directory that needs access to AWS. You can create two IAM roles with the names: Development and Admin. The naming convention is arbitrary, but it would be convenient if you give them names similar to the ones in your Active Directory Group. You may add a full access permission to the Admin role and a more restrictive permission to the Development role. These two roles will be assumed by the Development and Admin groups, allowing them to inherit the permissions contained in the roles for a limited time.

The diagram below illustrates the process of federated authentication through an on-premises Active Directory.



1. **Portal Access**: The corporate user accesses the company's Active Directory Federation Services (ADFS) portal sign-in page. This is typically a web-based portal provided by ADFS for single sign-on (SSO) purposes.
2. **User Authentication**: The user enters their Active Directory authentication credentials into the ADFS portal. ADFS processes these credentials and authenticates the user against the on-premises Active Directory.
3. **Retrieve User Data**: Once authenticated, Active Directory provides ADFS with the user's attributes, including details like group memberships.
4. **SAML Assertion to User's Browser**:, ADFS dynamically constructs Amazon Resource Names (ARNs) using the user attributes. These ARNs represent IAM roles in AWS that the user is allowed to assume. Additionally, ADFS uses user attributes to determine the relevant AWS account IDs. ADFS then packages this information into a SAML (Security Assertion Markup Language) assertion. The user's browser automatically posts the SAML assertion to the `AssumeRoleWithSAML` endpoint of AWS STS, as per the received instructions.
5. **Request Temporary Credentials**: Upon validating the SAML assertion, AWS STS maps the user to the appropriate IAM role and assumes this role on their behalf. Once assumed, STS generates and returns temporary AWS security credentials to the user's browser.
6. **AWS Console Access**: With these temporary credentials in hand, the user is automatically redirected and logged into the AWS Management Console. The level of access and permissions in the console corresponds to those associated with the IAM role they've assumed. If the user needs to make programmatic requests to AWS services, they can use the provided temporary credentials.

Single Sign-On Access across multiple accounts

Navigating between different AWS accounts or applications via ADFS has its set of challenges. For each AWS account, ADFS demands its own unique trust relationship. This approach multiplies the configurations required, especially as an organization expands its AWS portfolio. Consequently, you end up with increased administrative overhead due to the ongoing maintenance of these relationships. To make matters more intricate, the varied configurations can produce an inconsistent login experience, potentially confusing users.

With IAM Identity Center , Active Directory users can leverage their directory credentials to achieve single sign-on access across various AWS accounts, simplifying the entire process. This not only ensures a consistent login experience but also reduces the administrative overhead. Moreover, users can directly access AWS applications and login into resources like Amazon EC2 Windows instances using the same directory credentials.

You can connect your On-premises AD to AWS using IAM Identity Center via two methods:

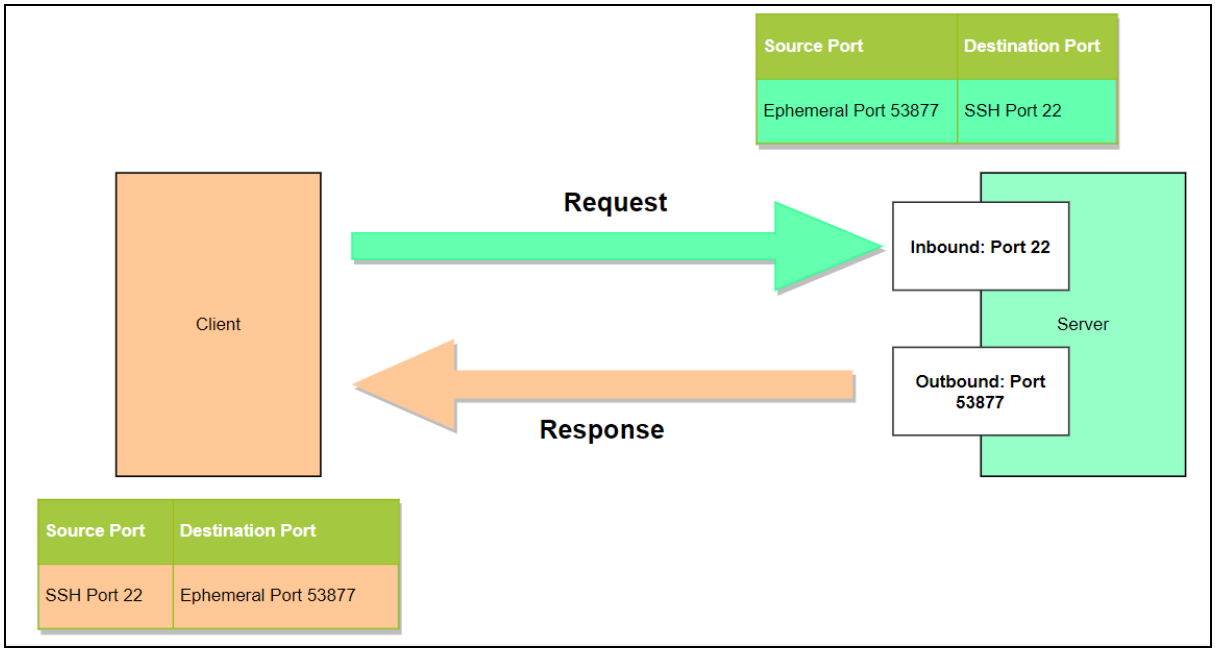
1. **AWS Managed Microsoft Active Directory**: IAM Identity Center requires a two-way trust extending from your AWS Managed Microsoft Active Directory to your on-prem AD to ensure authentication trust. Conversely, the trust flowing back permits the reading of user and group metadata. Note that One-way trusts do not work with IAM Identity Center.
2. **AD Connector**: As an alternative to the two-way trust, organizations can opt for the AD Connector. Think of it as a bridge: it's a directory gateway that funnels directory requests directly to your self-managed AD, ensuring real-time information without caching any data in the cloud.

Reference:
https://aws.amazon.com/blogs/security/aws-federated-authentication-with-active-directory-federation-services-ad-fs/
https://docs.aws.amazon.com/singlesignon/latest/userguide/connectonpremad.html

NACL - Ephemeral Port Range

Ephemeral ports are temporary ports that are generally used for outbound connections when a **client** makes a request to a server. These ports are automatically allocated by the TCP/IP stack upon establishing a connection. Since we can't control which port number will be used, we must allow an ephemeral port range if we're using a Network Access Control List.

Let's say that you want to connect to your EC2 instance via SSH from your local computer. In this case, your local computer is the client, and the EC2 instance is the server. When your computer initiates an SSH connection to the server, the client opens up a random port number from the available ephemeral port range and uses that as its source port. The server listening on port 22 will respond to the client's request using the ephemeral port as its destination port.



On AWS, the ephemeral port range for EC2 instances and ELBs is **1024-65535.** To implement the diagram above, we need to configure the NACL's inbound rule to allow traffic from the Client's IP on port 22.

Inbound Rule

Rule #	Type	Protocol	Port Range	Source	Allow / Deny
100	SSH (22)	TCP (6)	22	130.200.233.98/32	ALLOW

A range of ephemeral ports should be allowed in the outbound rule to allow outbound traffic to leave the subnet to which the NACL is applied.

Outbound

Rule #	Type	Protocol	Port Range	Destination	Allow / Deny
100	Custom TCP Rule	TCP (6)	1024 - 65535	130.200.233.98/32	ALLOW

References:
https://docs.aws.amazon.com/vpc/latest/userguide/vpc-network-acls.html
https://aws.amazon.com/premiumsupport/knowledge-center/resolve-connection-sg-acl-inbound/

Minimize Potential Attack Surface

Businesses primarily rely on Internet communication for fast financial transactions, customer feedback surveys, digital advertising, and many more. Because of the nature of activities on the Internet, it has become an attractive place for cybercriminals. It is full of predators always hunting for victims to prey on for their own gain. And failure to take preventive measures could cost you resources, time, and money.

One effective way of protecting your business from attackers is by reducing the attack surface of your application. The attack surface refers to the number of possible ways an attacker could enter your network. Attack surface reduction is a strategy that aims to limit the opportunities an attacker might have to target your application.

Ways To Reduce Attack Surface on AWS

- **Create a security group that only allows specific ports and authorized servers.**

  If you have a web application that uses an ELB and EC2 instance as its web server, instead of sharing a security group, create one security group for ELB and another for EC2. Add an inbound rule inside the EC2's security group. Configure the rule to only allow traffic from the ELB. You can achieve that by referencing the ELB's security group as the traffic source. Then, point the traffic coming from the public Internet to your ELB. By placing the ELB between the public Internet and your instance, you're preventing direct communication between the Internet users and your web server.

- **Set up Network Access Control Lists (ACLs) that only allow the required ports and network addresses in your network.**

  NACLs are useful in mitigating DDoS Attacks coming from known source IP addresses because it allows you to deny traffic to your subnet explicitly. If you only used TCP traffic, you can stop UDP flood attacks by denying all UDP traffic.

- **Protect your origin server by putting it behind a CloudFront web distribution.**

  The success of a DDoS attack depends on the number of compromised computer systems that target a specific server. You can protect your server from such attacks by putting it behind a CloudFront web distribution. CloudFront has multiple edge locations spread across the globe that serves content to users. Executing a DDoS attack on an extensive distributed system such as CloudFront is nearly impossible.

- **Enable AWS Shield Advanced for enhanced DDoS attack detection and monitoring for application-layer traffic to your AWS resources.**

  You can register an Elastic IP (EIP) Address as a protected resource when you enable AWS Shield Advanced. AWS resources attached to the registered EIP are automatically identified. AWS Shield Advanced detects protected resources quickly, which results in faster mitigation of infrastructure-layer DDoS attacks.

References:
https://docs.aws.amazon.com/whitepapers/latest/aws-best-practices-ddos-resiliency/attack-surface-reduction.html
https://aws.amazon.com/blogs/security/how-to-help-prepare-for-ddos-attacks-by-reducing-your-attack-surface

Using AWS Systems Manager Parameter Store and AWS Secrets Manager to Store System Credentials

Managing the security of your applications is an integral part of any organization, especially for infrastructures deployed in the cloud. One aspect of application security is how the parameters, such as environment variables, database passwords, API keys, product keys, etc. are stored and retrieved. As a best practice, secret information should not be stored in plain text and not be embedded inside your source code. It is also recommended to set up an automated system to rotate passwords or keys regularly (which is easy to forget when you manage keys manually).

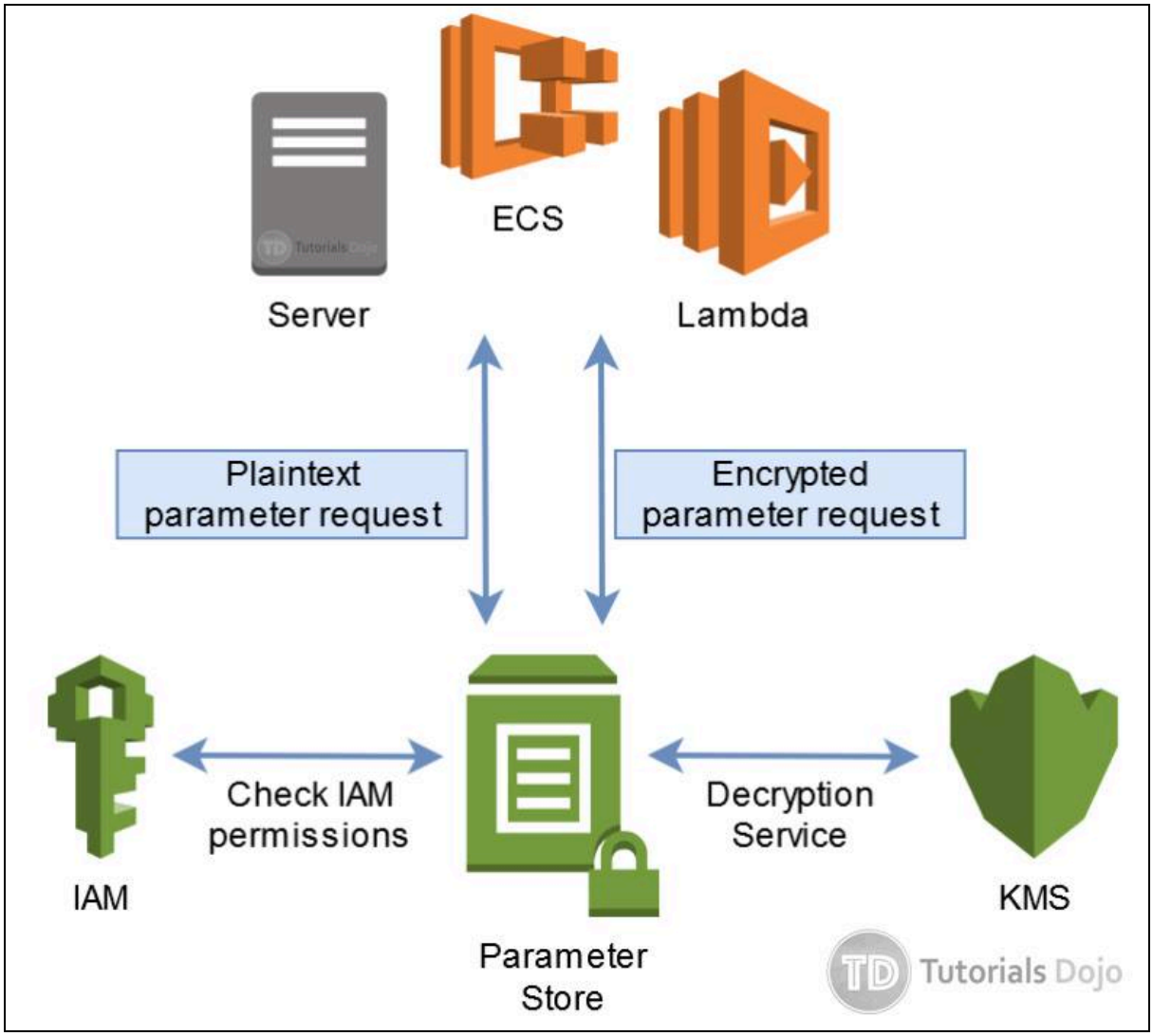
Managing and securing these types of data can be troublesome so Amazon provides the AWS Systems Manager Parameter Store and AWS Secrets Manager services for this purpose. Parameter Store and Secrets Manager are two distinct services but offer similar functionalities that allow you to centrally manage and secure your secret information.

AWS Systems Manager Parameter Store

Parameter Store is part of the application management tools offered by the AWS Systems Manager (SSM) service. Parameter Store allows you to create key-value parameters to save your application configurations, custom environment variables, product keys, and credentials on a single interface. It also allows you to secure your data by encryption, which is integrated with AWS KMS.

After you create your parameters in Parameter Store, you can then have these parameters retrieved by your SSM Run Command, SSM State Manager, or reference them on your application running on EC2, ECS, and Lambda or even on applications running your on-premises data center. This eliminates the need to hardcode variables or embed plain text credentials on your code. Parameter Store makes it easy to update these variables without modifying your source code, as well as eliminating the need to embed confidential information such as database passwords in your code.

Here's an overview of how applications can retrieve information on Parameter Store.



- Your application (on-premises servers, EC2, ECS, Lambda, etc.) sends a parameter request to SSM Parameter Store.
- If this is a plaintext parameter request, Parameter Store checks with IAM if the user/role is allowed to retrieve the parameter.
- If this is an encrypted parameter request, Parameter Store checks with IAM if the user/role is allowed to both retrieve and decrypt the parameter with AWS KMS. Decryption requires that the IAM has KMS Decrypt permission.
- If the IAM verification is successful, Parameter Store sends back the parameter value to the application.

Secure String Parameter

Parameter Store supports a lot of use cases, from saving unencrypted plaintext to more sensitive information, such as database passwords. You can also store configuration data and secure strings in hierarchies and track versions. During parameter creation, you specify the data type of your string:

- **String** - Any string value.
- **StringList** - Separate strings using commas.
- **SecureString** - Encrypt sensitive data using the KMS keys for your account

For parameters that should not be retrieved or referenced in plaintext, it is best to use the `SecureString` data type.

Sensitive information, such as passwords and secrets, should never be left exposed in plaintext. Parameter Store solves this problem by offering the `SecureString` data type, which uses AWS KMS to encrypt your information. AWS KMS uses either a customer-managed KMS key or an AWS-managed KMS key when encrypting the parameter value. Then in the application that references the parameter, you must set **`WithDecryption`** to "True" to use the original parameter value.

AWS Secrets Manager

AWS Secrets Manager enables you to rotate, manage, and retrieve database credentials, API keys, and other secrets throughout their lifecycle. It also makes it really easy for you to follow security best practices such as encrypting secrets and rotating these regularly.

If you are a security administrator responsible for storing and managing secrets, and ensuring that your organization follows regulatory and compliance requirements, you can use Secrets Manager to perform these tasks from one central location. Secrets Manager can offload the management of secrets from developers, such as database passwords or API keys, so they don't have to worry about where to store these credentials.

AWS Secret Manager also follows the same process flow like Parameter Store shown above. With descriptions laid out for both services, we'll take a look at their similarities and differences next.

Similarities and Differences

Both services offer similar web interfaces on which you can declare key-values pairs for your parameters and secrets.

Both services accept values of up to 4096 characters (4KB size) for each entry. And they both offer the option to encrypt these values. Encryption for both services is integrated into AWS KMS, so your application

referencing these parameters or secrets needs to have KMS Decrypt permission when retrieving encrypted values.

However, Parameter Store was designed to cater to a wider use case, not just secrets or passwords, but also application configuration variables like URLs, DB hostnames, custom settings, product keys, etc., which is why the default selection for creating a parameter is a plain text String value. You can enable encryption if you explicitly choose to.

Secrets Manager was designed specifically for confidential information that needs to be encrypted so the creation of a secret entry has encryption enabled by default. You can also choose to store in plaintext if you explicitly want to.

Secrets Manager also provides a built-in password generator through the use of AWS CLI. This can be helpful when you want to create an RDS instance with a CloudFormation template. You can create a randomly itemized password and later reference it on your RDS configuration.

Both services have a versioning feature. This allows you to view previous versions of your parameters in case you needed them. You can choose to restore the older version of the parameter. Parameter Store only allows one version of the parameter active at any given time. Secrets Manager on the other hand, allows you to have multiple items active at the same time. Secrets Manager distinguishes between different versions by the staging labels. You can check out staging labels here.

The next point of difference is the ability to rotate the secret. AWS Secrets Manager offers the ability to switch secrets at any given time and can be configured to regularly rotate depending on your requirements.

Another feature available for Secrets Manager is cross-account access. Secrets can be accessed from another AWS account. For example, you can have an application with an IAM role to retrieve secrets from another AWS account. This is useful if your secrets are centrally managed from another AWS account.

One advantage of the SSM Parameter is that it costs nothing to use it. You can store up to 10,000 parameters without getting billed. However, if you need more functionality in managing your parameters, you can use Advanced Parameters, which comes at a cost. You can set Parameter policies to an Advanced parameter, allowing you to set an expiration date or time-to-live to a parameter.

SSM Parameter offers three types of parameter policies:

1. **Expiration policy** – allows you to automatically delete a parameter after a specified period of time.
2. **ExpirationNotification policy** – allows you to receive a notification from Amazon EventBridge when a parameter is about to expire within a specified number of days.
3. **NoChangeNotification policy** – allows you to receive a notification from Amazon EventBridge when a parameter has not been modified for a specified period of time. This is helpful in cases where you have to proactively identify passwords that may need to be updated.

	SSM Parameter Store	AWS Secrets Manager
Store values up to 4096 Characters	Yes	Yes
Values can be encrypted with KMS	Yes	Yes
Can be referenced in CloudFormation	Yes	Yes
Built-in password generator		Yes
Automated secret rotation		Yes
Cross-account access		Yes
Additional Cost	Free	Yes

As an additional note, Parameter Store is now integrated with Secrets Manager so that you can retrieve Secrets Manager secrets when using other AWS services that already support references to Parameter Store parameters. This is helpful if your application is configured to use Parameter Store APIs, but you want your secrets to be stored in Secrets Manager.

Security Specialty Exam Notes:

If you're specifically asked to store **database credentials**, always choose Secrets Manager over SSM Parameter Store.

When asked for storing general application parameters such as API keys or software licenses, choose SSM Parameter store.

References:

https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-parameter-store.html
https://docs.aws.amazon.com/systems-manager/latest/userguide/integration-ps-secretsmanager.html

Hardening AMIs using EC2 Image Builder

Imagine you have an Auto Scaling group with a launch template that references a custom AMI. Since EC2 instances in the group are only temporary, they don't receive security patches or software updates during their runtime. If you want to apply these updates, you'd have to launch an instance from the current AMI, install the updates there, take an AMI snapshot of that instance, and then reconfigure your ASG's launch template to reference the new AMI. Such tasks can be error-prone and difficult to replicate across accounts or regions. To simplify this process, you can use an EC2 Image Builder pipeline to automate the entire image creation process, allowing you to enforce consistent security configurations on your AMIs, which can be distributed across different regions and AWS accounts.

When you create an Image Builder pipeline, you must select a 'recipe' that serves as the blueprint for creating the customed image. Image Builder launches temporary EC2 instances where it runs the provided recipe and executes validation tests. Thus, It's important that these instances have sufficient permissions to be able to do their tasks successfully. The permissions the EC2 instances have will depend on the profile instance associated with them.

AWS provides a set of predefined IAM policies that you can attach to the instance profile:

- **EC2InstanceProfileForImageBuilder** - contains the minimum set of permissions to build an AMI.
- **EC2InstanceProfileForImageBuilderECRContainerBuilds** - apart from building AMIs, EC2 Image Builder can also automate the creation of container images and upload them to Amazon ECR. Hence, you may use this predefined policy if you're building container images. This permission contains the minimum set of permissions for working with Amazon ECR.
- **AmazonSSMManagedInstanceCore** - EC2 Image Builder uses the help of Systems Manager Automation in the building process. This policy grants all the SSM actions that are needed by the Image Builder service.

In addition to the three policies, you may also grant the instance profile the `s3:PutObject` so the Image Builder can output logs to an Amazon S3 bucket.

References:
https://docs.aws.amazon.com/imagebuilder/latest/userguide/image-builder-setting-up.html
https://docs.aws.amazon.com/imagebuilder/latest/userguide/start-build-container-pipeline.html

Scanning vulnerabilities in container images using Amazon ECR

Amazon Elastic Container Registry (ECR) is a fully-managed Docker container registry service that allows developers to store, manage, and deploy Docker images. One of the key features of ECR that enhances security is its integrated vulnerability scanning.

When you push a Docker image to an ECR repository, you have the option to scan the image for vulnerabilities. Amazon ECR uses the Common Vulnerabilities and Exposures (CVE) database to report any known security weaknesses in the container image. The scan checks for operating system vulnerabilities, software flaws, and configuration issues. The results are accessible directly from the AWS Management Console, and you can integrate them into your CI/CD pipeline for automated alerting or blocking deployments based on the findings.

Let's say you need to ensure that all container images deployed in production are free of known vulnerabilities, and you need an automated way to enforce this requirement. Here's a sample solution of how you can use Amazon ECR image scanning.

1. **Automate Scans** - configure your ECR repositories to automatically scan images on push. This ensures that every new or updated image goes through a security check.
2. **CI/CD Integration** - integrate the vulnerability scan results into your CI/CD pipeline. For example, you can configure your pipeline to halt deployments if critical vulnerabilities are found.
3. **Compliance Reporting** - use the reports generated by ECR to correct your internal and external security auditing requirements.
4. **Alerting** - set up CloudWatch Alarms or SNS notifications for team members responsible for security to act upon any critical vulnerabilities discovered.

References:
https://docs.aws.amazon.com/AmazonECR/latest/userguide/image-scanning.html
https://aws.amazon.com/blogs/containers/amazon-ecr-native-container-image-scanning/

Amazon VPC Flows Logs

VPC Flow Logs is a feature in AWS that allows you to capture information about the incoming and outgoing IP traffic of the network interfaces in your Amazon VPC. Flow logs can assist you in properly monitor and log the activities in your VPC. It can diagnose an overly restrictive security group or network ACL rules, monitor the incoming traffic to your EC2 instances, and determine the flow of traffic to and from the network interfaces. After you've created a flow log, you can retrieve and view its data in the chosen destination.

A flow log data can be published to these destinations:

- Amazon CloudWatch Logs
- Amazon S3

Storing all the log data to Amazon S3 is a strategy that you can adopt to consolidate every flow log from across all VPCs that you own. The flow logs of your VPC can be published to an Amazon S3 bucket that you specify. The collected flow log records for all of the monitored network interfaces are sent to a series of log file objects stored in the S3 bucket. In this way, all of your logs are in one place which lessens the management overhead.

Both VPC Flow Logs and Traffic Mirroring are used to monitor the traffic in your VPC. VPC Flow Logs enables you to collect, store, and analyze network flow logs, while Traffic Mirroring gives you deeper insights into the network traffic.

VPC Flow Logs	Traffic Mirroring
Captured information: • Allowed and denied traffic • IP addresses (Source and Destination) • Ports • Protocol number • Packet and byte counts • Action taken (Accept or Reject)	**Use case:** • Analyze the packets to perform a root-cause analysis. • Reverse engineering a network attack • Detect and stop compromised workloads

How to know if the issue is in the network ACL or in the security group?

You can easily differentiate the issues between the two by reviewing the VPC flow log display.

- If the network ACL permits the outbound ICMP traffic, the flow logs will display two **ACCEPT** records (originating ping and response ping).
- But if the security group denies an inbound ICMP traffic, the flow logs will display a **REJECT** record. This means that the traffic did not reach the instance.

Exploring AWS Security: Network Access Analyzer in VPC

In the realm of AWS security, Network Access Analyzer stands as a crucial tool for fortifying Virtual Private Clouds (VPCs). Network Access Analyzer meticulously identifies unintended network access, empowering users to define and enforce network security requirements effectively.

Understanding Network Access Analyzer

Network Access Analyzer serves two primary functions:

1. Enhanced Security Posture: By pinpointing unintended network access, Network Access Analyzer bolsters network security posture in alignment with compliance standards.
2. Compliance Demonstration: Network Access Analyzer assists in showcasing compliance by ensuring adherence to regulatory requirements.

Key Features

Network Access Analyzer aids in verifying critical network access requirements:

- Network Segmentation: Ensures isolation between VPCs, maintaining separation for sensitive systems.
- Internet Accessibility Management: Identifies and restricts resources accessible via internet gateways.
- Trusted Network Paths: Validates the presence of essential network controls across paths.
- Trusted Network Access: Confirms access from trusted IP address ranges, ports, and protocols.

Operational Concepts

Network Access Analyzer operates on two core concepts:

1. Network Access Scopes: Define network access requirements, generating findings based on specified criteria.
2. Findings: Highlight potential network path violations based on defined criteria.

Accessing Network Access Analyzer

Network Access Analyzer can be accessed through:

- AWS Management Console: User-friendly interface for Network Access Analyzer resource management.
- AWS CLI and CloudFormation: Command-line tools and templates for programmatic access.
- AWS SDKs and Query API: Language-specific APIs and low-level API actions for integration.

In summary, Network Access Analyzer serves as a cornerstone for securing AWS VPCs. By leveraging its capabilities, AWS Security Specialists can fortify network integrity, demonstrate compliance, and ensure secure resource access effectively.

Reference:

https://docs.aws.amazon.com/vpc/latest/network-access-analyzer/what-is-network-access-analyzer.html

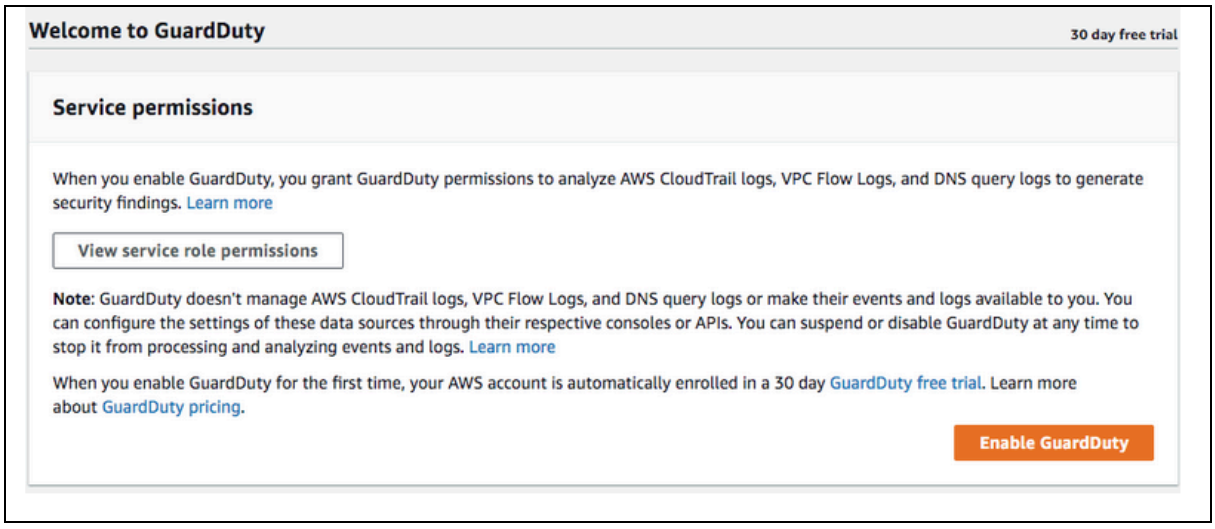
Amazon GuardDuty Multi-account Aggregation

Imagine running an organization with multiple AWS accounts for different workloads, teams, and projects. With every account, you need to monitor GuardDuty findings individually. It will be quite difficult for your security team to monitor these findings with their constant switching between AWS accounts.

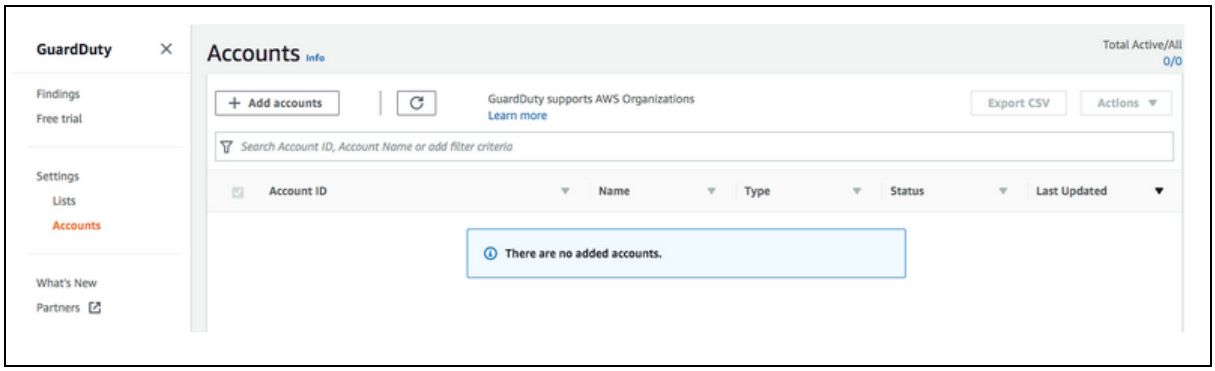
Amazon GuardDuty supports the consolidation of these findings into one AWS account. For example, your organization has 10 AWS accounts. All you have to do is create a "GuardDuty" AWS account with the sole purpose of ingesting all the findings from the 10 AWS accounts. With the help of this article, you should be able to aggregate your GuardDuty findings from multiple AWS accounts to a single AWS account.

In this scenario, we'll be using two AWS accounts: first is the master account, where all the findings will be sent to, and a secondary AWS account which will send its findings to the master account.

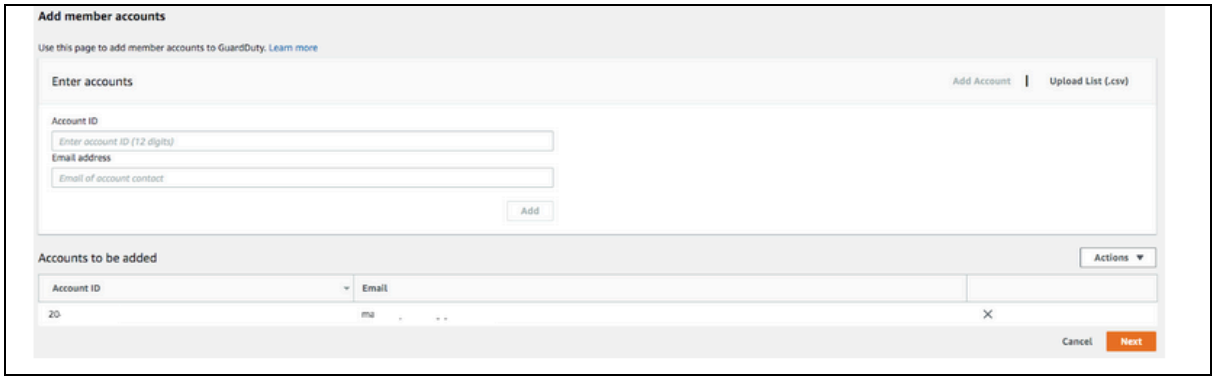
1. To start, we need to "Enable GuardDuty" for both the master and secondary accounts.



2. Once enabled, you will be redirected to the GuardDuty console. Head over to the "Accounts" section and click "Add accounts". For multiple accounts, you can add accounts by using the "Upload List (.csv)"

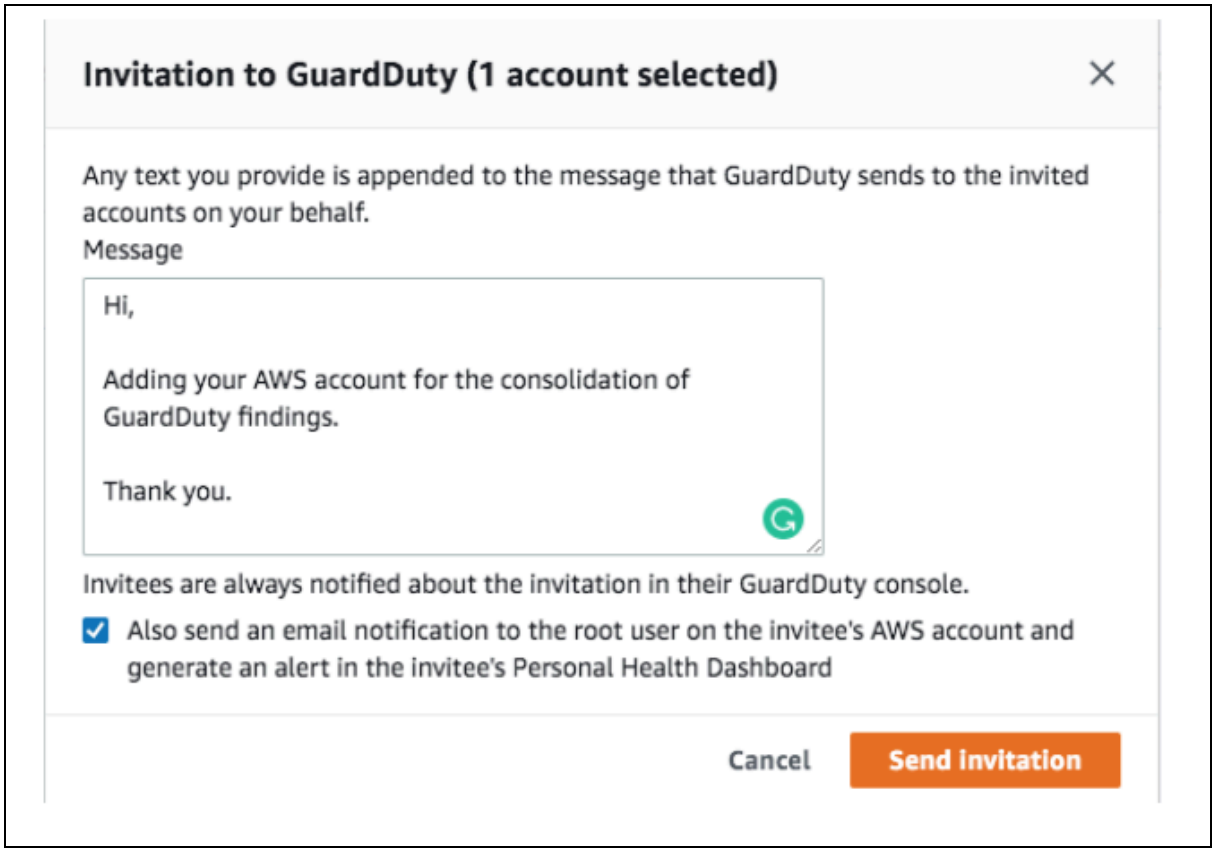


3. Enter the 12-digit account number and the email address associated with the secondary account. Click "Add" then "Next"



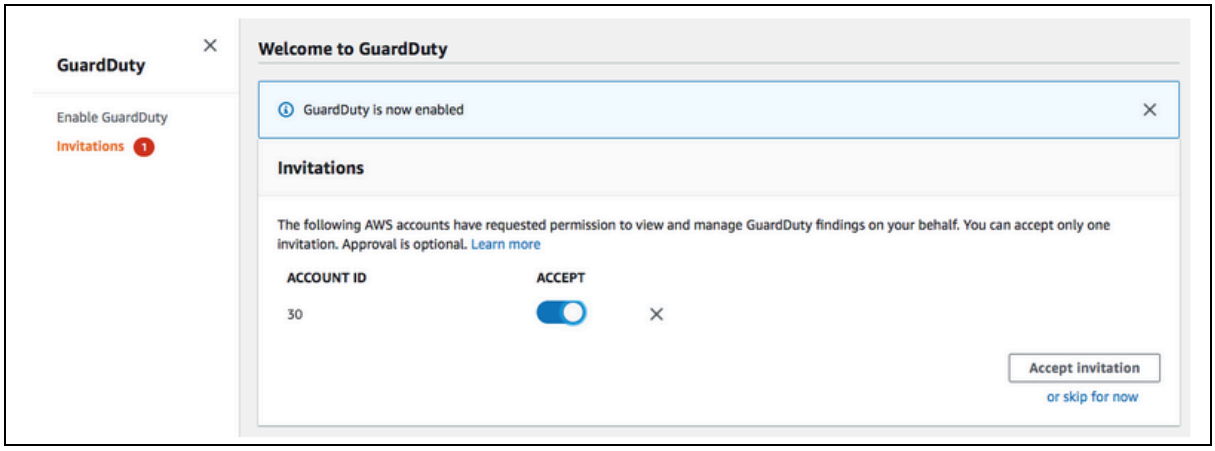
4. Once you have filled in the details of the secondary account, you should see it under the accounts tab. During this stage, the status of the account is "invite". Click on the invite and a pop up message will appear.

I. You can send an optional message to the receiver.
II. Tick the "also send an email notification" to ensure that the associated email of the secondary account will receive the email.
III. Once done, click "Send Invitation"
IV. During the invitation process, AWS will check if the account ID and the email address associated with the account is valid.

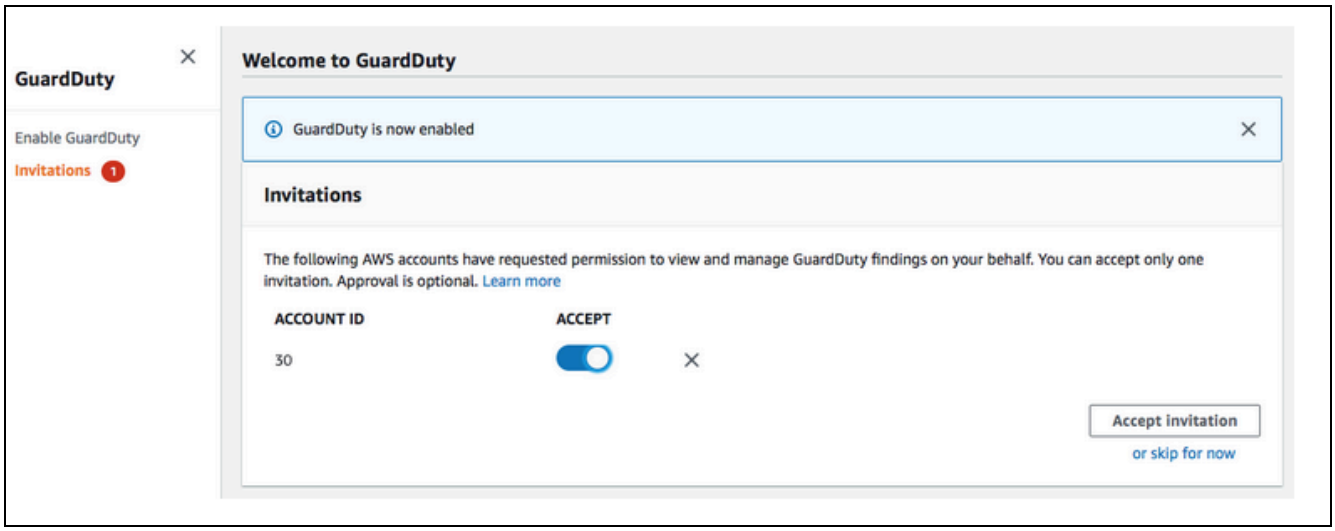


5. You have two options to accept the invitation:

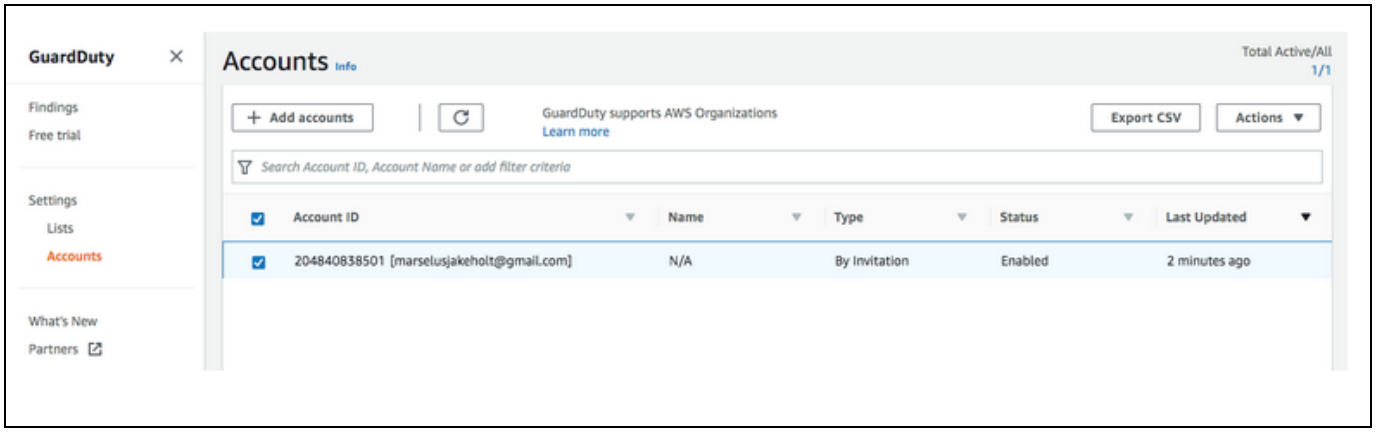
- Head over to your secondary account's GuardDuty and accept the invitation.
- Click the URL sent by AWS over the email.
  ***Note:** Remember that you need to enable GuardDuty on the secondary account before accepting the invitation.*



6. Once you have accepted the invitation, all of the findings in the secondary account will now be sent to the master account.



7. Once the secondary account has accepted the invitation, the status will now be "Enabled"



References:
https://docs.aws.amazon.com/guardduty/latest/ug/what-is-guardduty.html
https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_accounts.html

Testing network connectivity using VPC Reachability Analyzer

AWS VPC Reachability Analyzer is a feature that helps you perform network diagnostics to troubleshoot reachability between two resources within a VPC or peered VPCs. The feature allows you to understand why communication is failing between two endpoints, helping you to resolve network issues more quickly.

With the VPC Reachability Analyzer, you specify a source and a destination that can be:

- An EC2 instance
- A Network Interface
- A subnet A VPC peering connection
- VPN connections
- Transit Gateway connections

Once you have specified the source and destination, the tool analyzes the routes, security groups, network access control lists (NACLs), and more, to determine if network traffic can successfully flow between the source and destination. It then provides a detailed report of the analysis, including the reason for any blockage and suggestions for how to resolve it.

Use Cases

Network Troubleshooting - understanding why certain network communications are failing.
Security Audits - you can use it to validate that your security groups and NACLs are configured as you intend, ensuring that only the desired traffic is allowed and all other traffic is blocked.
Network Design - before deploying a new service, you can run analyses to ensure that network paths are set up correctly.

References:
https://docs.aws.amazon.com/vpc/latest/reachability/what-is-reachability-analyzer.html
https://aws.amazon.com/blogs/networking-and-content-delivery/automating-connectivity-assessments-with-vpc-reachability-analyzer/

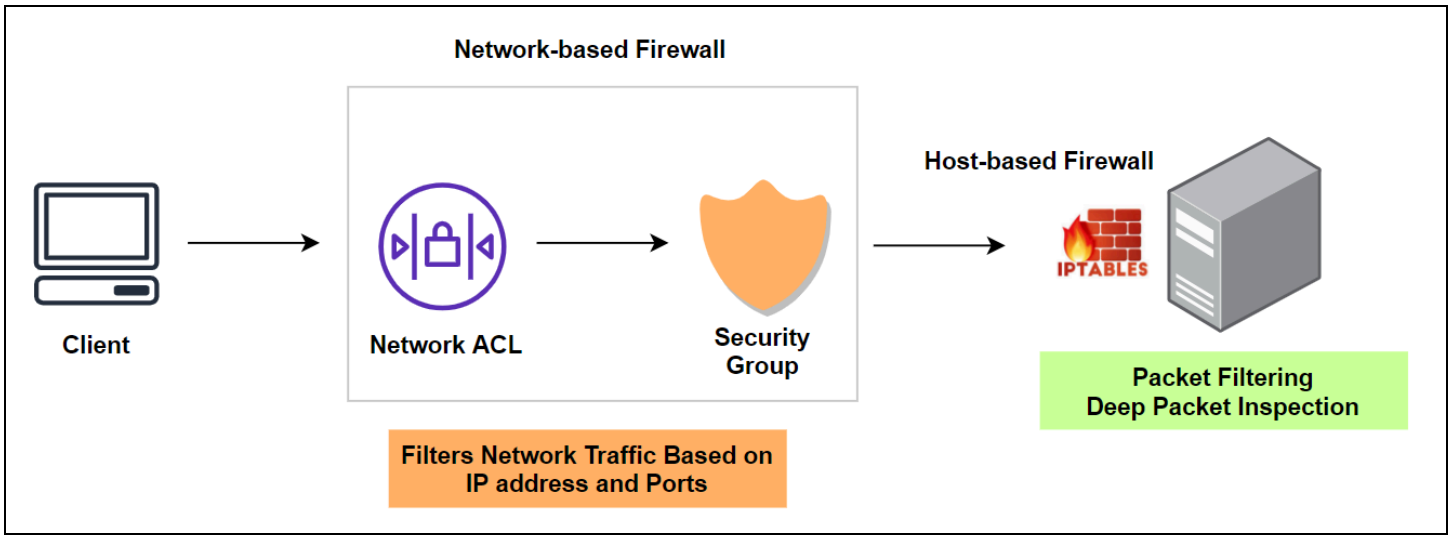
Host-Based Security

Security Groups and Network Access Control Lists (ACLs) are network-based virtual firewalls that AWS provides to help us manage and control the network traffic that reaches our instances. Security Groups operate at the instance-level, while Network ACLs operate at the subnet level. With their combined efforts, you can design and achieve the same granularity level that a traditional firewall gives.

Because of the AWS Shared Responsibility Model, some of the security burdens are offloaded from the AWS customers. However, AWS users are still encouraged to have host-based firewalls installed as there may be cases where security groups and Network ACLs might not be enough to cater to the needs of a large complicated network.

Using Network-based and Host-based Firewall For Ultimate Protection

A network-based firewall allows or denies traffic based on IP addresses and ports. Oftentimes, this level of "protection" isn't enough since requests containing malicious code can be disguised as legitimate. Therefore, knowing the type of traffic as well as keeping an eye on the data that is sent to your network, is essential.



OS-level software like IPtables, Windows Firewall, and third-party IPS/IDS solutions provide customized protection and functionality such as threat detection and deep packet inspection (DPI). Because they operate on the host level, they can also control access at an application level — something that a network-based firewall can't do. It is recommended to implement a security solution that comprises a network-based firewall and a host-based firewall.

Packet Data Inspection

Imagine you are in a library. You are instructed to weed out children's books from adult ones. It should be an easy task. You just need to take a quick look at the book cover or the book title to know what it is for. But imagine if someone has decided to write an inappropriate book with a misleading title and cover — one that could be easily mistaken for a playful kids' book?

The book title is not enough; you should also examine its contents.

The packets are the books, and its content is the packet's payload. The Security Groups and Network ACLs work similarly to how you filter books by title. They can only make a policy decision based on the packet's header: *source and destination IP address, port number, and protocol type*. These are just surface-level of filtering.

To augment the level of your security, you need to have a way to examine the contents of the packets that enter your network. This is the job of a deep packet inspection software — to protect your applications against sophisticated attacks such as injection of malware or shellcode on the packet's payload. In AWS, you could set up your own DPI solution by installing a host-based agent or a proxy-based deep packet inspection software.

Detecting whether a file stored on an Amazon EC2 instance has been modified.

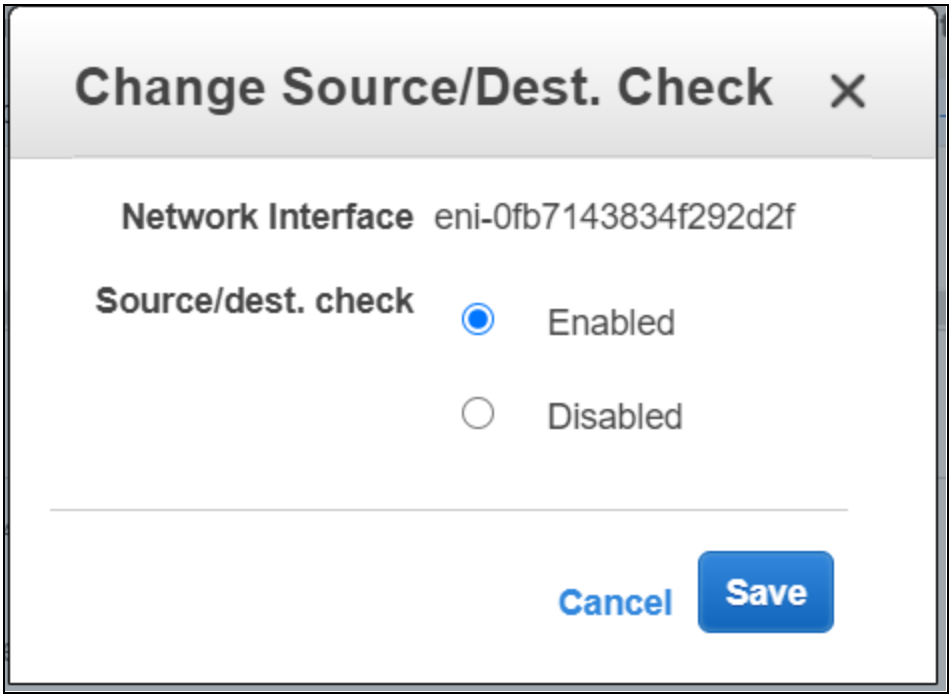
Sometimes, even a restrictive permission to an instance is not enough as human nature bounds us to make mistakes. Modifications of files, accidental deletion of important data, and mistyping of commands that may cause harm — these are just some of the risks that come with privileged access. It can't be helped, though. Every administrator in an organization needs privileged access to do their job. That is why we need a solution that can help detect such scenarios so that immediate action can be carried out when they happen.

In AWS, we rely on host-based IDS software for detecting modified files on an EC2 instance. This software can analyze and check the integrity of important system files by comparing their current state to a trusted state. You can configure the software to alert your security team whenever it detects irregularities or modifications to the files.

Disabling Source/Destination Check

When launching an EC2 instance, an elastic network interface is automatically allotted to it by AWS as *eth0*. However, if you spin up an instance from a virtual security appliance AMI, you might need additional elastic network interfaces. You can create and attach secondary ENIs with their own IP addresses (public and private) and security groups to your instance.

An ENI has an attribute called Source/Destination check which is enabled by default. Instances attached with ENI that have default configuration can only be a source or a destination. It means that it can only receive or send traffic.



Since a virtual security appliance is neither a source nor destination, you need to disable the Source/Destination check to allow both ingress and egress traffic to pass through.

Protecting your EC2 Instance Metadata Service

Instance metadata is a list of information that can be used to configure your instance. Some examples of them are IP address, hostname, security group, instance-id, and security credentials attached to the instance. You can only retrieve instance metadata through a unique IP address (*http://169.254.169.254/latest/meta-data/*) within the AWS network.

Let's say that you have an application running on an instance that communicates to a private S3 bucket. That instance has an IAM role that permits it to do S3 API calls. When the application has to perform an S3 API action, it retrieves the IAM role's credentials from the instance metadata item *iam/security-credentials/role-name*.

This is a security risk because anyone who can log in to your instance can now exploit the sensitive data stored on your S3 bucket. What you can do as a mitigation technique is to restrict metadata services access using IPtables. You can restrict instance metadata access for specific applications, users, or groups on your instance.

References:
https://aws.amazon.com/answers/networking/vpc-security-capabilities/
https://docs.aws.amazon.com/vpc/latest/userguide/security.html
https://aws.amazon.com/security/
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instancedata-data-retrieval.html
https://d1.awsstatic.com/whitepapers/Security/AWS_Security_Best_Practices.pdf

Domain Whitelisting Using Proxy Servers

Domain whitelisting is a security strategy wherein an authorized person (i.e., Security Administrator, Security Head) explicitly permits egress traffic from a network, devices, or applications to specific domains that were deemed trustworthy by an organization. This is done to reduce the chance of a computer downloading a virus/malware from the Internet, communicating with malicious IP addresses, or being a DDoS target.

Whitelisting is a stricter and more secure policy than blacklisting, as it denies all traffic by default. Whitelisting is a trust-centric approach — the reliability of protection wholly depends on the level of trust given by an organization to the networks being whitelisted.

In AWS, we can use HTTP proxies such as Squid or Varnish for whitelisting approved domains or DNS blocking. To implement this, do the following:

1. Install and run the proxy software to an EC2 instance in a public subnet.
2. Open the inbound port 80 and 443 to allow HTTP and HTTPS traffic.
3. Configure the route table of the private subnet where your web application resides to use the proxy EC2 instance when communicating to the public Internet.
4. Since the EC2 instance is acting as an inline proxy server, don't forget to disable its source/destination check attribute.

References:
https://aws.amazon.com/blogs/security/how-to-set-up-an-outbound-vpc-proxy-with-domain-whitelisting-and-content-filtering/
https://aws.amazon.com/blogs/security/how-to-add-dns-filtering-to-your-nat-instance-with-squid/

Certificate Management Using AWS Certificate Manager

AWS Certificate Manager (ACM) is a service that lets you easily create and manage public domain validation (DV) certificates that you use to establish an SSL/TLS encrypted connection between a client and a server. You can request a public certificate directly from ACM, free of charge, or import your own. The managed renewal of certificates is only limited to ACM certificates and does not apply to imported certificates.

You can not export and install ACM certificates to an EC2 instance, as you would have with free SSL/TLS certificates like Let's Encrypt. Instead, you would have to integrate it with other AWS services like *Amazon CloudFront* or *AWS Elastic Load Balancer*. Then, use these services to interface with your backend server.

You must issue a certificate in the **US East (N. Virginia)** region if you want to set up an HTTPS connection between **viewers and CloudFront**. However, you can issue a certificate **in any region** if you wish to create an encrypted communication between **CloudFront and an ELB origin**.

ACM Private Certificate Authority (CA) is a managed private CA service that enables you to procure a Public Key Infrastructure (PKI) without the operational expenses of running your own infrastructure, like buying a dedicated HSM for storing CA keys.

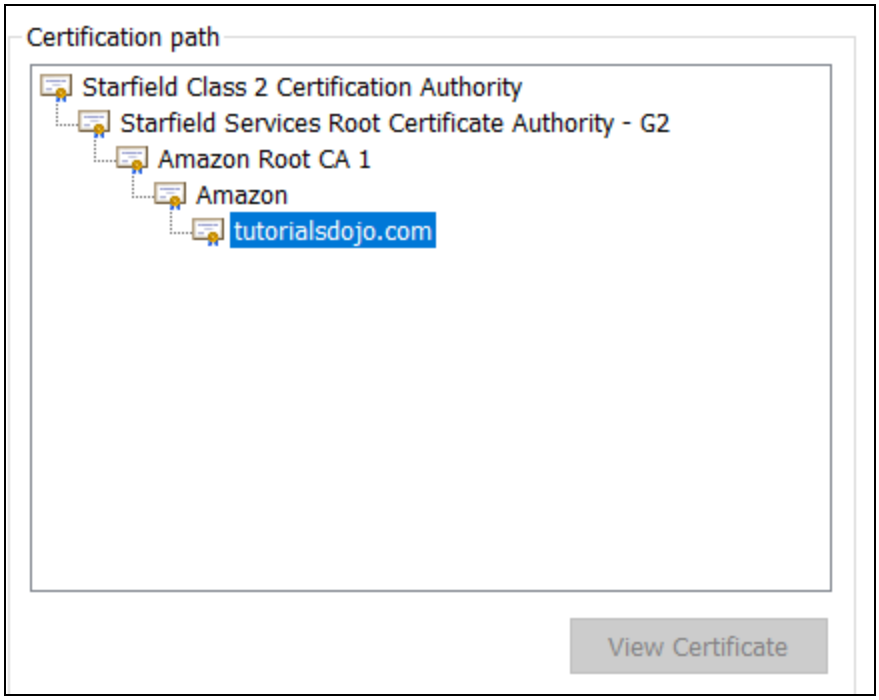
A private CA handles the issuance, validation, and revocation of private certificates within a private network. It comprises two major components: The first is the **CA certificate**, a cryptographic building block upon which certificates can be issued. The second is **a set of run-time services** for maintaining revocation information through the **Certificate Revocation List** (CRL).

Unlike ACM-issued certificates, ACM Private CA can customize the usage of certificates. You can create *certificate extensions* to modify the purpose of a digital certificate that caters to your needs. These extensions are used for applications like *code signing certificates*, *OCSP signing certificates*, and *mutual authentication*.

CA Hierarchy

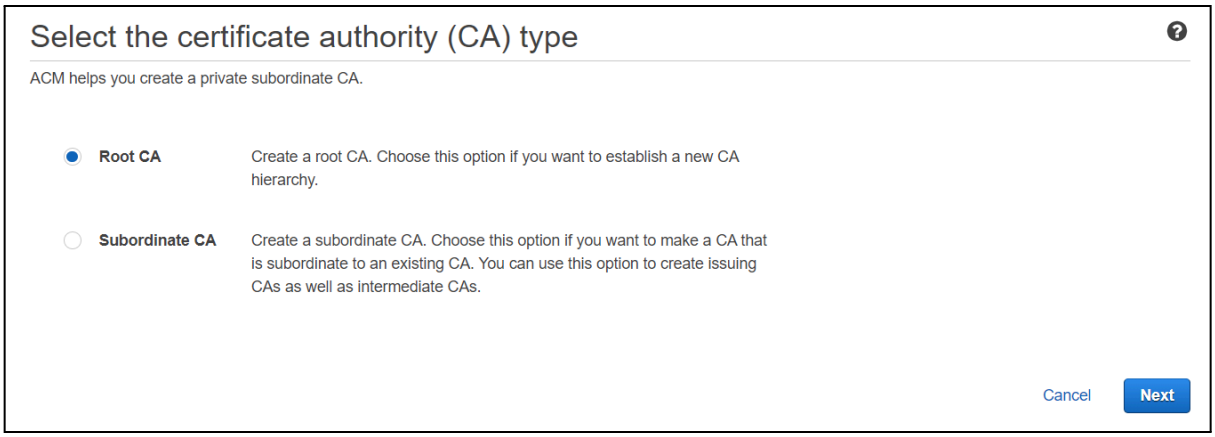
A CA hierarchy provides strong security and restrictive access controls for the most-trusted root CA at the top of the trust chain, while allowing more permissive access and bulk certificate issuance for subordinate CAs lower in the chain.

The image below is an example of a public CA hierarchy. We can quickly determine the root CA by merely looking at the top-level value of the certification path. In this case, the "**Starfield Class 2 Certification Authority**" is the root CA. The certificates below it are its subordinate CAs. The bottommost "**tutorialsdojo.com**" certificate, which is also the domain name that it protects, is the end-entity certificate.



When you enter google.com into your browser, a google server and your browser negotiate which certificate should be used to start an HTTPS connection. Your browser will search for the server's certificate from its list of public trusted certificate authorities (e.g., Comodo SSL, GeoTrust SSL, Go Daddy) like the one shown in the screenshot above. If your browser fails to recognize the server's certificate, it will report an insecure connection even if it uses a valid certificate. This usually happens for servers that use self-signed certificates.

With private certificates, the hierarchy doesn't change except for the fact that you're self-hosting the root CA and subordinate CA. In this case, the trusted authority is the entity that issued the root CA.



Mutual (two-way) Authentication in AWS

Most websites that you're familiar with use one-way SSL. One-way SSL begins when a client sends a request to a server. The server then sends its public certificate to the client. Next, the client verifies the certificate's authenticity based on its known trusted authorities. Both parties will start communicating via SSL/TLS connection after a successful verification of the client. In this case, the client is the only one responsible for authenticating the request.

On the other hand, mutual authentication is when the client and the server authenticate each other's requests by exchanging their public certificates. They can begin exchanging information in an encrypted channel once certificates have been successfully verified at both ends. This authentication type is popular with the Internet Of Things (IoT) and business-to-business (B2B) applications.

In AWS, you can use the AWS ACM Private CA to implement a mutual authentication. Let's say that you want to create a mutual authentication between two containers in a VPC. You can begin by first creating a subordinate CA using the root CA. Then, you can use OpenSSL to generate a CSR and a private key in the machines. After generating, sign them by using the `aws acm-pca issue-certificate` command. The output returns the certificate ARN. You can retrieve the certificate by specifying that certificate ARN when calling the `aws acm-pca get-certificate` command.

References:
https://aws.amazon.com/certificate-manager/
https://docs.aws.amazon.com/acm-pca/latest/userguide/PcaWelcome.html
https://aws.amazon.com/blogs/security/how-to-create-certificates-with-custom-extensions-using-aws-certificate-manager-private-ca/
https://docs.aws.amazon.com/cli/latest/reference/acm-pca/issue-certificate.html

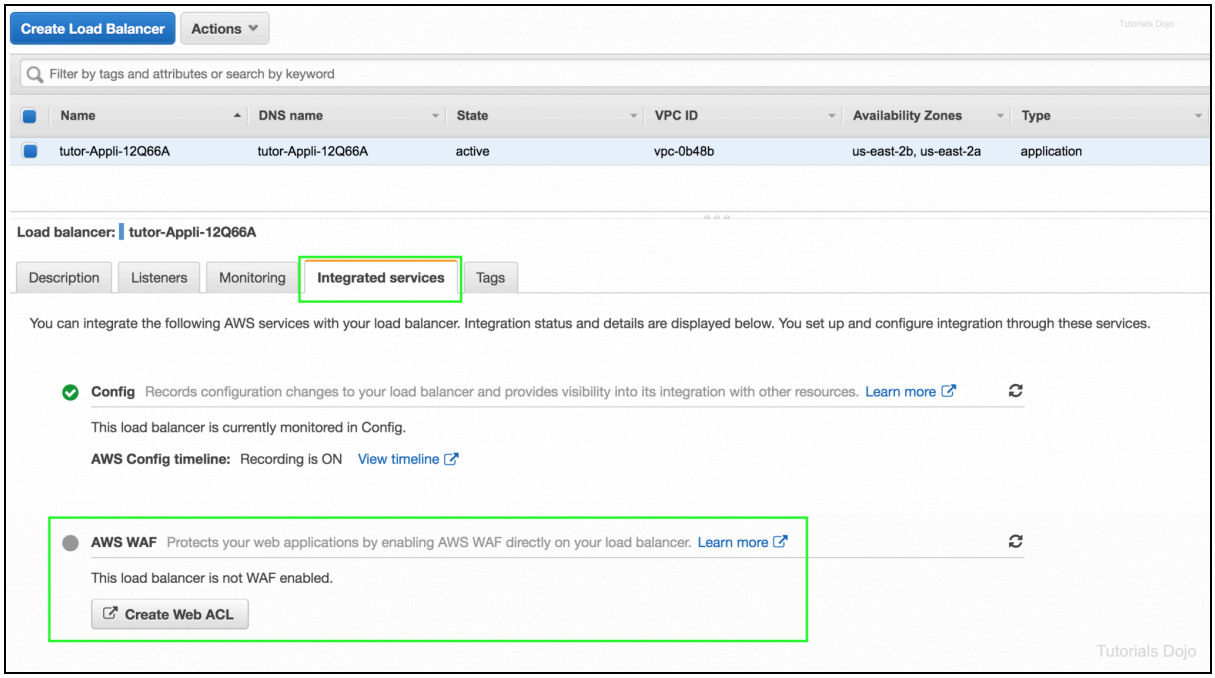
Elastic Load Balancer Security

The Elastic Load Balancer (ELB) is a service that automatically distributes traffic across multiple targets. But aside from load balancing, it has security features that we can use to enhance network security; or at least make it easier to implement secured connections.

Integration with WAF

Suppose you want to use AWS WAF to protect your EC2 instances against common web exploits. In that case, you're going to have to use the Application Load Balancer (ALB) in front of your instances. The reason for that is because Amazon EC2 does not directly integrate with AWS WAF. In this type of set-up, the ALB sits in between your web application and AWS WAF.

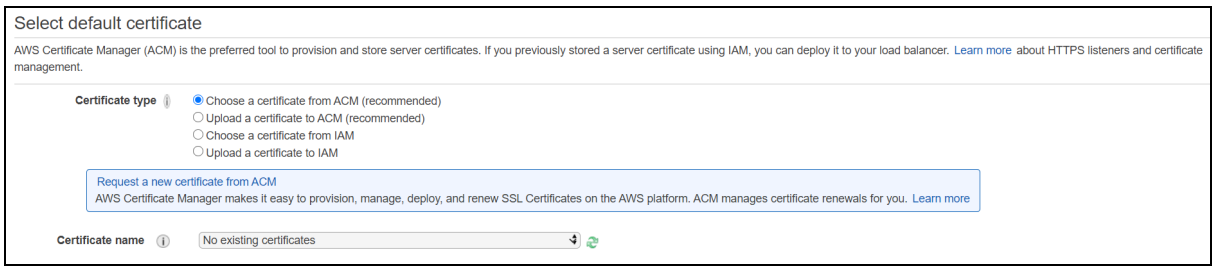
AWS WAF filters incoming web requests (according to rules that you set) that goes to your ALB. Then, the ALB load balances the traffic to your web application.



SAN/SNI

Subject Alternative Name (SAN) allows you to issue a single SSL/TLS certificate for different subdomains. Suppose I have a root domain called mywebsite.com and two other subdomains, namely blog.mywebsite.com, and mail.mywebsite.com. Instead of issuing three separate certificates for each of them, I can secure these

domains under a single SAN certificate. To use SAN with ALB, you simply have to choose the certificate to be used from IAM or ACM when creating your load balancer.



Server Name Indication (SNI) is a TLS protocol that lets you consolidate multiple certificates to handle different domains in a single location. These domains are served from a single IP address — in our case, it is the ALB's endpoint. Let's say that we have two domains: `domain1.com` and `domain2.com`. Naturally, we want to securely serve these domains via HTTPS. We provision two certificates for each of the domains. Now, instead of using two load balancers to serve the domain, we can just add the domain's certificates in a single ALB. The ALB will intelligently choose the correct certificate to use.

Perfect Forward Secrecy

Perfect Forward Secrecy (PFS) is a protocol that uses an ephemeral key to encrypt data for each communication exchange. These ephemeral keys are discarded once the session ends. By introducing PFS, even if an attacker has managed to steal the key that was used to encrypt a session, the stolen data can only be decrypted from the data inthatt that was encrypted by that key — effectively reducing the blast radius of a breach.

To implement PFS in ELB, you need to configure your load balancer to use **Elliptic Curve Cryptography (ECDHE) cipher suites**. This suite of algorithms is what allows the client and server to create shared ephemeral keys.

Deregistering compromised EC2 instances

If a computer has been compromised or is suspected of being hacked, the first step to go about it is to isolate the affected computer. This may be in the form of shutting it down to stop the attack right away or disconnecting it from any network. In AWS, we can't physically unplug a compromised EC2 machine.

You can do the following actions to isolate a compromised EC2 instance:

1. If you have multiple EC2 instances running on an ALB target group, you must first capture the metadata from the instance before making any changes.
2. Detach the instance from an auto scaling group if it belongs to any. Then, deregister the compromised instance from its target group.
3. Change its security group rule to a more restrictive one, only allowing authorized users such as the forensic team to access it. This way, you can safely isolate it from other instances, preventing further damage to it.

References:
https://aws.amazon.com/blogs/aws/aws-web-application-firewall-waf-for-application-load-balancers/
https://aws.amazon.com/blogs/aws/new-application-load-balancer-sni/
https://aws.amazon.com/about-aws/whats-new/2014/02/19/elastic-load-balancing-perfect-forward-secrecy-and-more-new-security-features

DynamoDB Security

You can implement data encryption of a DynamoDB database in two ways:

DynamoDB Encryption Client

- DynamoDB Encryption Client is a software development kit that provides end-to-end encryption (EE2E) — data is encrypted at both ends of a secure communication channel (source & destination.)
- DynamoDB Encryption Client uses signing to protect your data against unauthorized changes. It transparently encrypts data as you add them to your table and decrypts data as you retrieve them after verifying.
- **Only** selected attribute values in an item can be encrypted. The table name, partition key, sort key, and some attributes that you didn't encrypt remain in plaintext.

  Example:

  For the sake of simplicity, imagine we have a DynamoDB table called "*Song Table*." And for some reason, you'd like to encrypt an entire item. DynamoDB Encrypted Client will not allow you to do that. However, you can encrypt the value of the "Song_Price" attribute since it is not a primary key. DynamoDB needs the primary keys to remain in plaintext to look for an item so it'll not waste resources scanning the entire table.

  Song Table

Artist (*Partition Key*)	Song_Title(*Sort Key*)	Song_Price
Beethoven	Für Elise	$1.00

  After encryption, the result will look something like this:

  Song Table

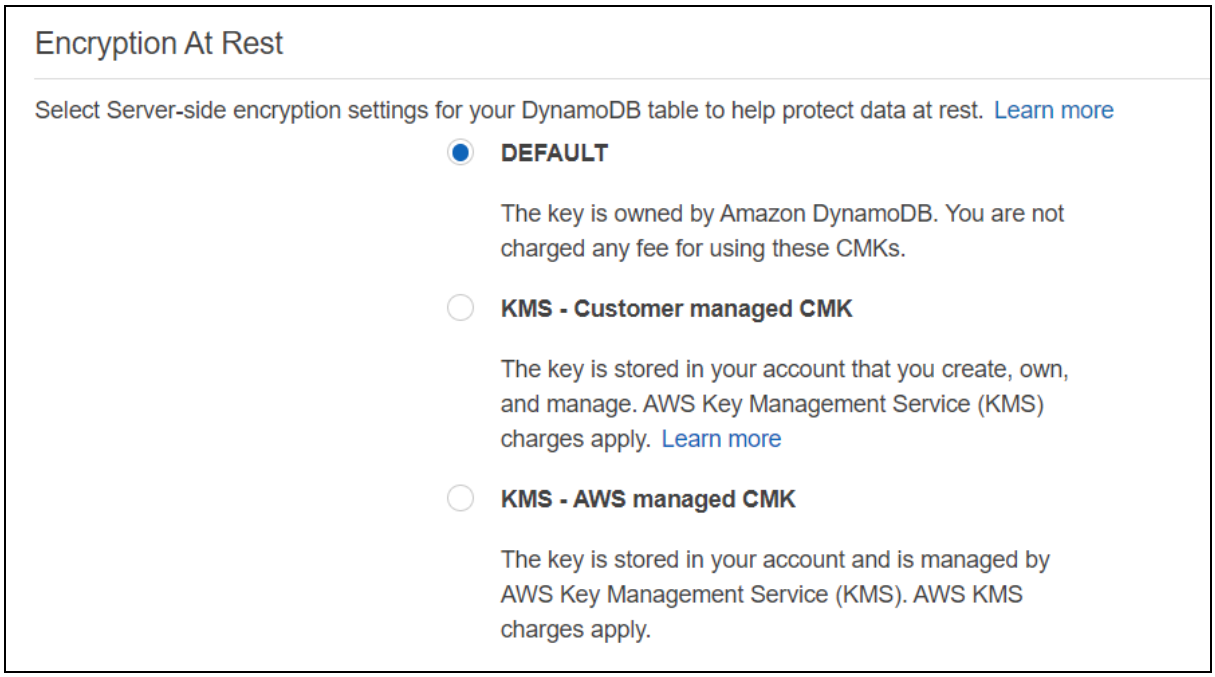
Artist (Partition Key)	Song_Title(Sort Key)	Song_Price
Beethoven	Für Elise	Binary(b" 'b\xd3\xb9+e\xf1\\")

- DynamoDB Encryption Client can be used with encryption keys from AWS KMS, AWS CloudHSM, or any cryptographic service that you prefer.

- AWS or any third-party services cannot view your data.

Server-side encryption at rest

- Server-side encryption is a DynamoDB feature that transparently encrypts data as it saves to disk and automatically decrypts them when you access your table.
- You can use an Amazon DynamoDB owned key, your own KMS key, or an AWS managed key.



- Unlike DynamoDB Encryption Client, **all table data is encrypted** in server-side encryption.

References:
https://docs.aws.amazon.com/dynamodb-encryption-client/latest/devguide/client-server-side.html
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/EncryptionAtRest.html

Amazon S3 Cross-Origin Resource Sharing (CORS)

- A security mechanism that controls how web applications loaded from one domain can access S3 resources from another domain.
- Browser-enforced security feature preventing unauthorized cross-origin requests.
- Configured through CORS rules attached to S3 buckets.
- Essential for web applications that need to access S3 objects from JavaScript in browsers.

What is CORS?

Cross-Origin Resource Sharing (CORS) is a browser security feature that prevents JavaScript code from one domain (e.g., example.com) from accessing resources on another domain (e.g., mybucket.s3.amazonaws.com). By default, browsers enforce the "Same-Origin Policy" which blocks these requests. CORS rules tell the browser which cross-origin requests should be allowed.

Without CORS configuration, a web application hosted at `https://www.example.com` cannot make AJAX requests to retrieve objects from `https://mybucket.s3.amazonaws.com` because the browser blocks the cross-origin request.

How S3 CORS Works

When a browser makes a cross-origin request to S3:

1. **Preflight Request (OPTIONS)** – For certain requests, browser sends OPTIONS to check permissions
2. **S3 Evaluates CORS Rules** – S3 checks if origin matches any CORS rule
3. **CORS Response Headers** – S3 includes Access-Control-Allow-* headers if matched
4. **Browser Allows/Blocks** – Browser permits or denies request based on headers

CORS Configuration Structure

S3 CORS configuration is XML with one or more CORS rules:

```
xml
<CORSConfiguration>
 <CORSRule>
  <AllowedOrigin>https://www.example.com</AllowedOrigin>
  <AllowedMethod>GET</AllowedMethod>
  <AllowedMethod>PUT</AllowedMethod>
  <AllowedHeader>*</AllowedHeader>
  <ExposeHeader>ETag</ExposeHeader>
  <MaxAgeSeconds>3600</MaxAgeSeconds>
 </CORSRule>
</CORSConfiguration>
```

CORS Rule Elements

- **AllowedOrigin** (required)
  - Specifies which origin(s) can access the bucket
  - Exact match: `https://www.example.com`
  - Wildcard: `*` (allow all origins - security risk)

    - Must include protocol (https:// or http://)
    - Does NOT support partial wildcards like `https://*.example.com`
- **AllowedMethod** (required)
    - HTTP methods allowed: GET, PUT, POST, DELETE, HEAD
    - Must specify at least one method
- **AllowedHeader** (optional)
    - Request headers allowed in preflight request
    - Use * to allow all headers (common for flexibility)
    - Specify exact headers: Content-Type, Authorization, x-amz-*
- **ExposeHeader** (optional)
    - Response headers browser can expose to JavaScript
    - Common: ETag, x-amz-request-id, x-amz-server-side-encryption
- **MaxAgeSeconds** (optional)
    - How long (seconds) browser caches preflight response
    - Typical: 3600 (1 hour) to 86400 (24 hours)

Example CORS Configurations

Single Origin - Web Application:

xml
```
<CORSConfiguration>
 <CORSRule>
  <AllowedOrigin>https://www.myapp.com</AllowedOrigin>
  <AllowedMethod>GET</AllowedMethod>
  <AllowedMethod>POST</AllowedMethod>
  <AllowedHeader>*</AllowedHeader>
  <MaxAgeSeconds>3600</MaxAgeSeconds>
 </CORSRule>
</CORSConfiguration>
```

Multiple Origins:

xml
```
<CORSConfiguration>
 <CORSRule>
  <AllowedOrigin>https://www.example.com</AllowedOrigin>
  <AllowedOrigin>https://dev.example.com</AllowedOrigin>
  <AllowedMethod>GET</AllowedMethod>
  <AllowedMethod>PUT</AllowedMethod>
  <AllowedHeader>*</AllowedHeader>
```

```
  </CORSRule>
</CORSConfiguration>
```

Public Access (Not Recommended):

```
xml
<CORSConfiguration>
  <CORSRule>
    <AllowedOrigin>*</AllowedOrigin>
    <AllowedMethod>GET</AllowedMethod>
    <AllowedHeader>*</AllowedHeader>
  </CORSRule>
</CORSConfiguration>
```

Security Considerations

Avoid Wildcard Origins in Production

- `<AllowedOrigin>*</AllowedOrigin>` allows ANY website to access resources
- Security risk: malicious sites can read bucket contents (if publicly accessible)
- Only use for truly public data (images, fonts, open datasets)
- Never use with sensitive data in production

Specify Exact Origins

- Always use specific origins: `https://www.example.com` (not `*`)
- Include protocol (https:// preferred)
- Include subdomain and port if needed
- `www.example.com` ≠ `example.com` (different origins)

CORS Does Not Replace Access Control

- CORS only controls browser behavior (not API access)
- Non-browser clients (curl, mobile apps) ignore CORS
- Must still use bucket policies, IAM policies for access control
- CORS + bucket policy both required for secure browser access

Use HTTPS Origins

- Prefer `https://` over `http://`
- Modern browsers block mixed content (HTTPS → HTTP)

- Prevents man-in-the-middle attacks

Common Use Cases

- **Static Website Hosting** – Web app at example.com loading resources from S3
- **Single-Page Applications** – React/Angular/Vue apps accessing S3 data
- **Direct Browser Uploads** – Upload files directly to S3 using presigned URLs
- **CloudFront + S3** – CloudFront serving S3 content with CORS headers
- **Mobile Web Apps** – Progressive Web Apps accessing S3 resources

Troubleshooting CORS Issues

"No 'Access-Control-Allow-Origin' header" Error:

- CORS not configured on bucket → Add CORS configuration
- Origin doesn't match AllowedOrigin exactly → Check protocol, subdomain, port
- Bucket policy blocks request → Verify bucket policy allows access

"Method not allowed" Error:

- HTTP method not in AllowedMethod → Add required method (PUT, POST, DELETE)
- AllowedHeader doesn't include custom headers → Add headers or use `*`

Preflight Request Fails:

- Missing AllowedHeader for custom headers → Add specific headers or `*`
- Bucket policy blocking OPTIONS → Ensure policy allows s3:GetObject

CORS and CloudFront

When using CloudFront with S3 origin:

- Enable "Cache based on selected request headers" → Whitelist
- Add `Origin` to whitelisted headers
- CloudFront forwards Origin header to S3
- S3 evaluates CORS and returns appropriate headers
- Alternative: Configure CORS headers directly in CloudFront

References:
https://docs.aws.amazon.com/AmazonS3/latest/userguide/cors.html
https://docs.aws.amazon.com/AmazonS3/latest/userguide/ManageCorsUsing.html

Configure Integrations with AWS Edge Services and Third-Party Services

AWS enables comprehensive security data integration through the Open Cybersecurity Schema Framework (OCSF), a collaborative open-source standard co-founded by AWS and industry partners including Splunk, Broadcom, and CrowdStrike. Amazon Security Lake automatically converts logs from natively supported AWS services to OCSF format and stores them in customer-owned S3 buckets as Apache Parquet files. For custom sources, organizations must transform raw security data into OCSF-compliant format with specific requirements: partitioning by region, account ID, and event day; Parquet files limited to 1 MB data page size and 256 MB row group size with zstandard compression; and delivery intervals between 5 minutes and 1 event day. AWS AppFabric complements this by fully managing integrations with over 25 SaaS applications including Slack, Microsoft 365, ServiceNow, and Zoom, automatically normalizing audit logs to OCSF every two minutes and delivering them via Amazon S3 or Kinesis Data Firehose. Third-party security vendors such as Cribl, NETSCOUT, Lacework, Tanium, and Trend Micro provide pre-built connectors that transform their security telemetry into OCSF format, enabling seamless ingestion into Security Lake through Lambda-based transformation pipelines or direct API integrations.

AWS edge services provide layered security through tight integration of Amazon CloudFront content delivery network with AWS WAF and AWS Shield. CloudFront distributions support both standard and multi-tenant configurations for web ACL associations, with multi-tenant distributions enabling baseline protection across multiple domains while allowing tenant-level security customization through protection pack inheritance and override capabilities. Organizations deploy third-party WAF rules through AWS Marketplace managed rule groups, which are curated by security vendors including Imperva, F5, Fortinet, and Palo Alto Networks. These rule groups integrate directly into AWS WAF web ACLs with subscription-based pricing and automatic updates as new vulnerabilities emerge. AWS Managed Rules provide no-additional-cost protection covering OWASP Top 10 vulnerabilities, while intelligent threat integration APIs enable advanced protections through the Bot Control, Account Takeover Prevention, and Account Creation Fraud Prevention managed rule groups. Best practices for CloudFront origin protection include configuring custom headers with rotating secrets managed by AWS Secrets Manager, implementing origin access control for S3 buckets, and creating WAF rules at both the CloudFront edge and origin Application Load Balancer layers to validate custom header presence before allowing requests to reach backend resources.

AWS Shield provides automatic DDoS mitigation operating across the globally distributed edge location network, with Shield Standard included at no cost for all AWS customers and Shield Advanced offering enhanced protections for mission-critical applications. Shield Advanced integrates with AWS WAF to provide application-layer DDoS defense through the automatically updated DDoS protection managed rule group, supporting up to 50 billion requests per month with included WAF costs for protected resources. AWS Firewall Manager centralizes security policy management across AWS Organizations, automatically applying Shield Advanced and WAF protections to new accounts and resources while auditing compliance through continuous monitoring. For comprehensive defense-in-depth architecture, organizations implement edge security at CloudFront with geographic and rate-based rules, application-layer protection at ALB with managed rule groups and custom rules for application-specific vulnerabilities, and integrate with Amazon Security Lake to centralize

security logs in OCSF format for unified analysis across AWS services, SaaS applications, and third-party security tools. This multi-layered approach combines edge-based traffic filtering, stateful inspection, threat intelligence integration, and normalized security data aggregation to provide comprehensive protection while maintaining centralized visibility through standardized schema frameworks.

References:
https://docs.aws.amazon.com/security-lake/latest/userguide/custom-sources.html
https://docs.aws.amazon.com/appfabric/latest/adminguide/ocsf-schema.html
https://docs.aws.amazon.com/waf/latest/developerguide/what-is-aws-waf.html
https://aws.amazon.com/blogs/security/how-to-enhance-amazon-cloudfront-origin-security-with-aws-waf-and-aws-secrets-manager/
https://aws.amazon.com/shield/features/

Implementing Protections and Guardrails for Generative AI Applications

The OWASP Top 10 for LLM Applications 2025 identifies critical security vulnerabilities including prompt injection, sensitive information disclosure, supply chain risks, data and model poisoning, improper output handling, excessive agency, system prompt leakage, vector and embedding weaknesses, misinformation, and unbounded consumption. Amazon Bedrock Guardrails addresses LLM01 Prompt Injection through multi-layered prompt attack detection that identifies jailbreaks bypassing native safety controls, prompt injections overriding developer instructions, and prompt leakage attempts extracting system prompts, with configurable filter strengths from Low to High and support for both text and code-based attacks. For LLM02 Sensitive Information Disclosure, Guardrails provides sensitive information filters that detect and either block or mask personally identifiable information including names, email addresses, phone numbers, and custom regex patterns in both user inputs and model responses, with support for configurable policies applied to prompts, responses, or both. LLM09 Misinformation is mitigated through contextual grounding checks that evaluate responses for factual accuracy based on reference sources and relevance to user queries, filtering over 75% of hallucinated responses in RAG and summarization workloads, plus Automated Reasoning checks using formal mathematical logic to validate outputs against defined rules with up to 99% accuracy.

Organizations must implement defense-in-depth strategies combining AWS security services to address risks beyond Guardrails capabilities. For LLM06 Excessive Agency, apply least privilege IAM roles to Amazon Bedrock Agents action group Lambda functions that interact with external systems, implement role-based access control using AWS IAM Identity Center or Amazon Cognito with rule-based claim-to-role mapping, and enable user confirmation requirements for mutating actions before execution. LLM08 Vector and Embedding Weaknesses in RAG architectures require permission-aware vector databases with fine-grained access controls to prevent cross-tenant data leakage, encryption of knowledge base data using AWS KMS customer managed keys, and user-level authorization ensuring retrieved embeddings respect individual access permissions. LLM03 Supply Chain vulnerabilities demand validation of foundation model sources through Amazon Bedrock's curated model catalog or Amazon SageMaker's model governance features, implementation of AWS Config rules to enforce approved model usage, and Service Control Policies restricting

Bedrock access to authorized accounts within AWS Organizations. LLM05 Improper Output Handling requires sanitizing and validating all LLM outputs before passing to downstream systems, implementing AWS WAF to detect malicious payloads at the application layer, and using Lambda parsers in Bedrock Agents for granular control over output processing.

Comprehensive implementation requires layering AWS security services across the generative AI stack: AWS WAF for filtering suspicious input patterns and rate limiting to address LLM10 Unbounded Consumption, AWS CloudTrail for logging all API interactions to detect LLM07 System Prompt Leakage attempts, Amazon CloudWatch for monitoring usage metrics and setting alarms on anomalous behavior, and AWS Secrets Manager for secure credential management preventing exposure through LLM outputs. Organizations should use the AWS Generative AI Security Scoping Matrix to determine shared responsibility boundaries across different application types, from fully managed services like PartyRock to custom implementations requiring full model training oversight, ensuring appropriate controls for LLM04 Data and Model Poisoning through data validation pipelines, Amazon S3 versioning with object-level immutability, and SageMaker Data Wrangler for detecting potential bias during data preparation. Security teams must establish organizational resiliency by socializing AI security as a core business requirement, implementing governance frameworks that enforce least privilege access using IAM Access Analyzer and encrypting all data at rest with AWS KMS including consideration of customer managed keys.

References:

https://owasp.org/www-project-top-10-for-large-language-model-applications/assets/PDF/OWASP-Top-10-for-LLMs-v2025.pdf
https://aws.amazon.com/bedrock/guardrails/
https://aws.amazon.com/blogs/machine-learning/architect-defense-in-depth-security-for-generative-ai-applications-using-the-owasp-top-10-for-llms/
https://docs.aws.amazon.com/bedrock/latest/userguide/lambda-parser.html

Domain 4: Identity and Access Management

Overview

The fourth exam domain of the AWS Certified Security Specialty exam deals with Identity and Access Management in AWS. Expect to see a lot of advanced concepts in AWS IAM and its various IAM Entities. The concept of Identity and Access Management (IAM) covers the design, implementation, and troubleshooting of the authentication and authorization process of your various AWS resources.

The "identity" aspect of IAM helps you identify the genuine users or entities in your organization. This is also known as the process of authentication. The IAM service has an "Authorization" function to access and control your AWS resources. A policy can be created that contains multiple IAM permissions that authorize the IAM User to do a set of actions via the AWS Management Console or via AWS API. This is the "access management" aspect of the IAM service.

This domain will test your knowledge on the following:

- Establishing identity through an authentication system, based on requirements
- Setting up multi-factor authentication (MFA)
- Determining when to use AWS Security Token Service (AWS STS) to issue temporary credentials
- Constructing attribute-based access control (ABAC) and role-based access control (RBAC) strategies
- Evaluating IAM policy types for given requirements and workloads
- Interpreting an IAM policy's effect on environments and workloads
- Applying the principle of least privilege across an environment
- Enforcing proper separation of duties

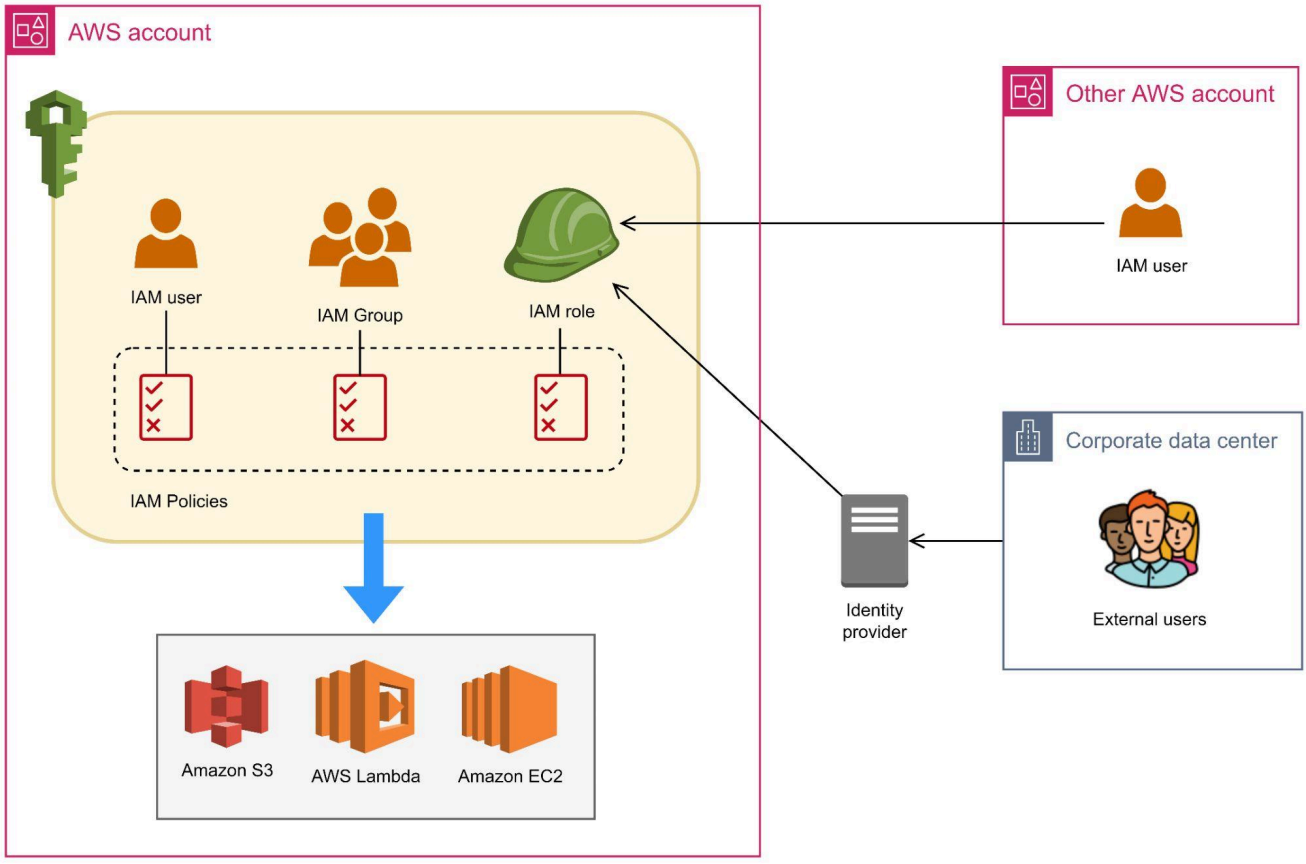
We will cover a lot of IAM policies and troubleshooting topics in this chapter. This section just cover 16% of the exam, so you have to manage your time reviewing the concepts/services under this domain.

AWS Identity Access Management (IAM)

IAM is the primary tool for controlling and managing access to an AWS account. It sits at the core of AWS security; everything you do with AWS, whether it's creating a Lambda function, uploading a file to an S3 bucket, or something mundane as viewing EC2 instances on the Console, is governed by IAM. It allows you to specify who, which AWS resources, as well as what actions they can and cannot do. These are also known as *authentication* and *authorization*.

IAM Identities

IAM identities handle the authentication aspect of your AWS account. It pertains to any user, application, or group that belongs to your organization.



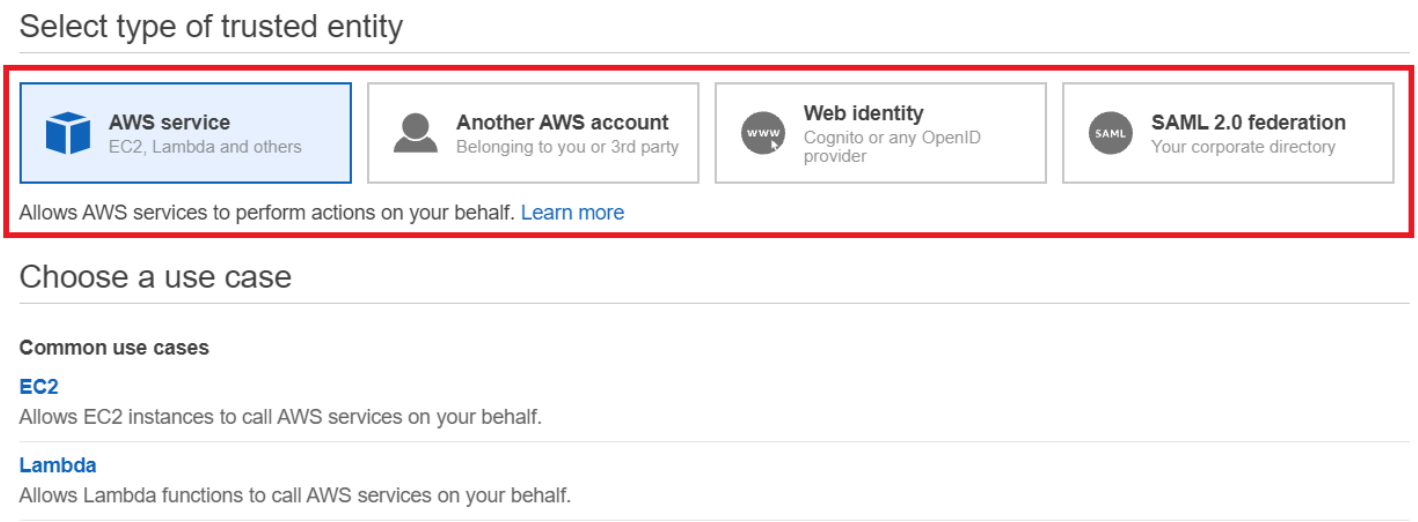
An IAM identity can be an **IAM User**, **IAM role**, or **IAM Group**.

An **IAM User** represents the user or application who interacts with AWS services. Take note that an IAM user is different from the root user. Although full admin permissions can be given to an IAM user, there are certain

actions that only a root user can carry out, such as deleting an AWS account. IAM users are given long-term credentials for accessing AWS services. These credentials can be in the form of a username and password (for console access) or an access key and secret key (for programmatic access). The former is primarily used when logging into the AWS Console, whereas the latter is used when interacting with AWS Services via CLI/API commands.

An **IAM role** isn't intended to be associated with a unique user, rather, it is meant to be assumed by certain AWS Services or a group of external users with common business functions. Unlike IAM users, roles use time-limited security credentials for accessing AWS. When creating a role, you choose a trusted entity. The trusted entity determines who is authorized to assume the IAM role.

An IAM role can be assumed by AWS Services to perform actions on your behalf, an IAM user within your account or from an external one, and users federated by identity providers that AWS trusts. Examples of these identity providers are Microsoft Active Directory, Facebook, Google, Amazon, or any IdP that is compatible with OpenID Connect (OIDC) and SAML 2.0.



IAM Group is simply a collection of IAM users. The policies attached to an IAM Group are inherited by the IAM users under it. It's possible to assign an IAM user to multiple groups, but groups cannot be nested (group within a group). IAM group offers an easy way of managing users in your account. For example, if you have a team of developers that needs access to AWS Lambda, instead of attaching the necessary permission for them individually, you can put the developers together in an IAM Group called 'Developers' and associate the Lambda permissions to that group. If a new member joins the team, simply add him/her to the 'Developers' IAM group.

IAM Policy

An IAM identity cannot perform AWS actions without an IAM Policy attached to it unless the resource being accessed allows the IAM Identity to do so. An IAM Policy is what authorizes an IAM user or role to control and access your AWS resources. There are three types of IAM Policies to choose from:

- AWS Managed Policies - these are built-in policies that have been constructed to conform to common use cases and job roles. For example, let's say you're working as a system administrator, and you have to give your new database administrator the necessary permissions to do his/her job. Rather than figuring out the right permissions, you may simply use the `DatabaseAdministrator` managed policy, which grants complete access to AWS services necessary to set up and configure AWS database services. AWS Managed Policies cannot be deleted or modified.

- Customer-managed Policy - this refers to the policies that you manually create in your account.

- Inline Policy - a policy that is embedded in an IAM Identity. Unlike AWS Managed Policies and Customer-managed Policies, an inline policy does not have its own ARN; thus, it can't be referenced by other IAM identities. It is scoped to a specific IAM user or role.

References:
https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html
https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html
https://docs.aws.amazon.com/IAM/latest/UserGuide/id_groups.html

IAM Policy Structure

An IAM Policy is expressed as a JSON document. It is made up of two components: policy-wide information and one or more statements. The policy-wide information is an optional element that is placed on top of the document. It's usually just a Version element pertaining to the AWS policy language version being used. This element usually has a static value of `2012-10-17`, referring to the release date of the current AWS policy access language.

The Statement block is where you add the permissions you need for accessing various AWS services. A policy can have single or multiple statements where each statement is enclosed within a bracket.

The following is an example of an IAM Policy.

```
{
    "Version": "2012-10-17",
    "Statement": [{
                "Sid": "AllowFullEC2AccessFromMyNetwork",
                 "Effect": "Allow",
                 "Action": ["ec2:DescribeInstance*"],
                 "Resource": "*",
                 "Condition": {"IpAddress": {"aws:SourceIp": "180.0.111.0/24"}}
            }]
}
```

An individual statement has the following elements:

- `Statement ID (Sid)` - this is just a label that you give to a statement. It's helpful for quickly identifying what a statement is for rather than reviewing what's in the `Action`, `Effect`, or `Resource`. This element is optional; however, it might be useful when modifying or debugging policies with multiple statements.

- `Effect` - explicitly dictates whether the policy allows or denies access to a given resource. It only accepts an `Allow` or a `Deny`.

- `Action` - this element contains the list of actions that the policy allows or denies. You can use the (*) wildcard to grant access to all or a subset of AWS operations. In the example above, the `ec2:DescribeInstance*` refers to all actions that start with the string "DescribeInstance". So this can be `DescribeInstanceAttribute`, `DescribeInstances`, `DescribeInstancesStatus`, and so on.

- `Resource` - the list of AWS resources to which the `Action` element is applied. The example policy above uses a (*) wildcard alone to match all characters. This implies that Action is applicable to all resources. You can be more restrictive to the resources that you want to work with by specifying their Amazon Resource Names (ARN).

- `Condition` - an optional element that you can use to apply logic to your policy when it's in effect. For instance, the sample policy above only allows requests originating from the `180.0.111.0/24` network.

Some conditions that you should be aware of are:

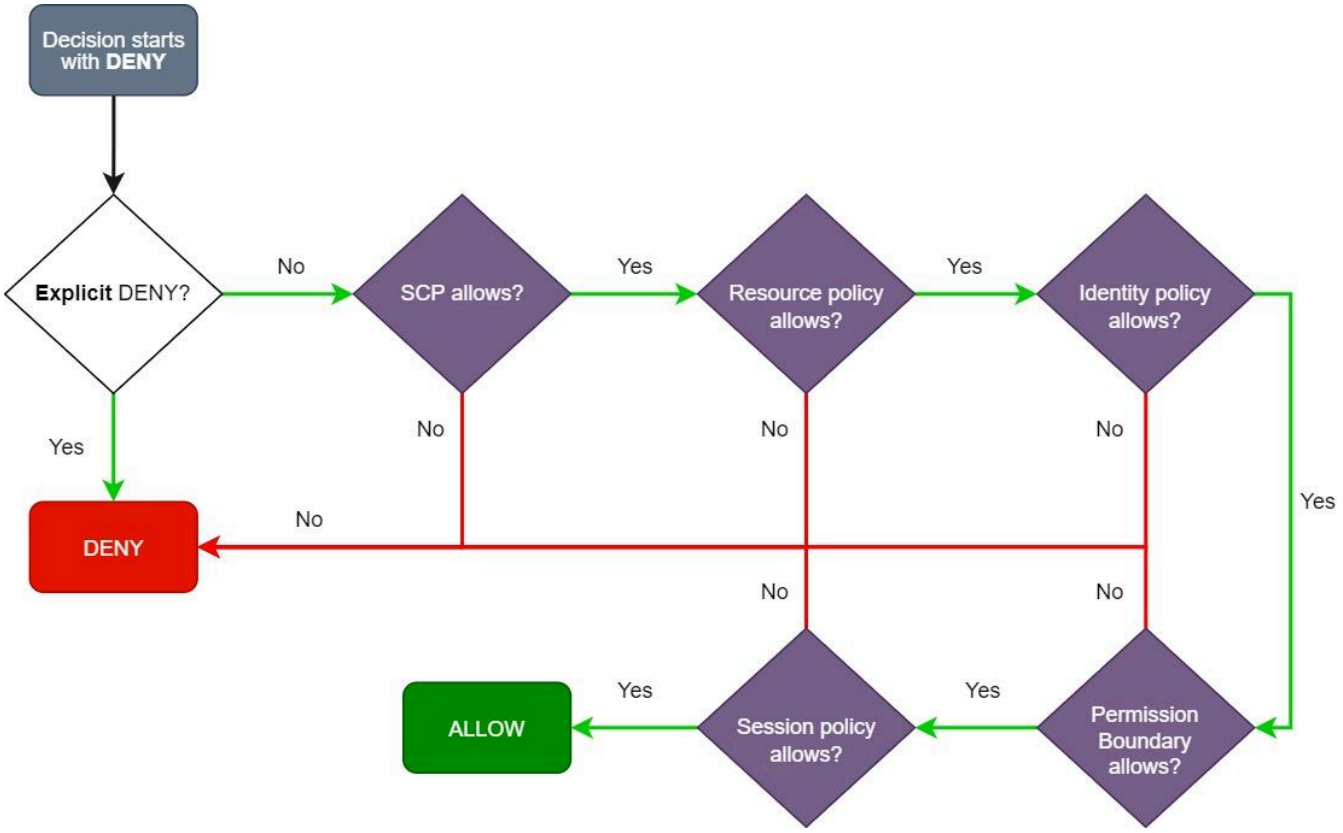
- `StringEquals` - Exact string matching and case-sensitive
- `StringNotEquals` - Negated matching
- `StringLike` - Exact matching but ignoring case
- `StringNotLike` - Negated matching
- `Bool` - Lets you construct Condition elements that restrict access based on true or false values.
- `IpAddress` - Matching specified IP address or range.
- `NotIpAddress` - All IP addresses except the specified IP address or range
- `ArnEquals, ArnLike`
- `ArnNotEquals, ArnNotLike`
- Use a `Null` condition operator to check if a condition key is present at the time of authorization.
- You can add `IfExists` to the end of any condition operator name (except the `Null` condition)—for example, `StringLikeIfExists`.

References:
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/iam-policy-structure.html
https://aws.amazon.com/blogs/security/back-to-school-understanding-the-iam-policy-grammar/

How AWS IAM Handles Conflicting IAM Policies Attached to a Resource

When someone tries to do something in AWS, like launching an EC2 instance or listing S3 buckets, AWS evaluates all the involved IAM policies to determine if the request should granted. Since IAM policies can be associated with different types of identities, the hierarchy of these identities influences the final permissions for a user.

Let's break down the process of how AWS evaluates a request using the simplified version of the AWS flow chart below.



1. First, AWS denies a request by default. For instance, if your IAM user has no policies attached to it and you attempt to read an S3 object, your request is implicitly denied.
2. If there's an explicit deny in any of the policies involved in the request, IAM denies the request.
3. When a request is made in an AWS account with Organizations' SCPs in place, IAM first examines the SCPs to determine if the action is allowed. If the SCPs don't allow the action, IAM denies the request.

4. If there's no SCP to evaluate or there are no conflicts in the SCP, IAM proceeds to check the resource policy. Note that not all AWS services have a resource policy. Some examples of resource policy are S3 bucket policy, KMS key policy, Lambda function resource policy, SQS queue policy, etc.
5. IAM then checks the policies attached to the IAM user or role making the request. The request is denied if no policy matches the actions being requested. IAM also implicitly denies if there are no identity-based policies present in the request.
6. IAM checks whether there are any statements in the permissions boundary that would prevent the requested action. If there is, the request is denied. Otherwise, the evaluation process continues.
7. Finally, IAM checks whether there's a session policy that was passed in the request. A session policy is an inline permission that can be given to a session principal (federated users). IAM grants the request if the session policy allows the action. If there are no session policies involved, IAM checks if the principal is a role session. If it is, then the request is allowed.

It's important to understand that the policy evaluation for **same-account access differs from cross-account access**. The diagram above depicts the policy evaluation for cross-account access.

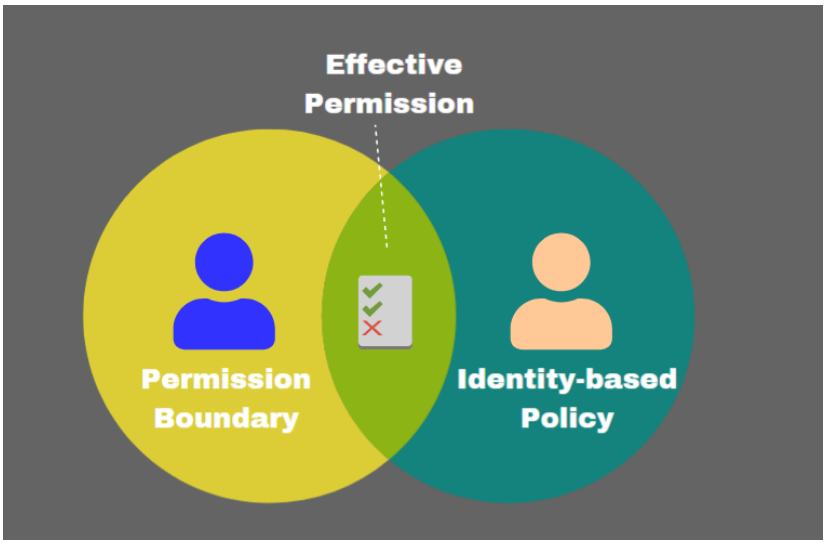
In the **same account access** setting**,** it's possible to grant permissions either via the Resource policy or Identity policy. For instance, even if an IAM user has no policies assigned, the user can still be granted access to an S3 bucket if authorized in the bucket policy. Similarly, an S3 bucket with no bucket policy can be accessed by an IAM user as long as its identity policy allows him to do so. However, this logic does not apply to the **IAM role trust policy** and **KMS key policy.** For this two resource-policy, permissions must be explicitly given to the Identity policies of the principal as well.

Reference:
https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_evaluation-logic.html

IAM Permissions Boundary

AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources. This service allows you to manage your IAM identities: users, groups, roles, and permissions. You manage access in AWS by creating policies and attaching them to IAM identities or AWS resources.

A **permissions boundary** is an IAM feature that AWS introduced to set the maximum permissions that an IAM entity (user or role) could have. You can use an AWS managed policy or a customer managed policy to set the boundary for your IAM entities. When you select a permissions boundary for an entity, the entity can perform only the actions that are allowed by both its identity-based policies and its permissions boundaries. These actions allowed by both identity-based policies and its permissions boundaries are called *effective permissions*.



Within an account, the permissions for an entity can be affected by identity-based policies, resource-based policies, permissions boundaries, Organizations SCPs, or session policies.

Note: The permissions boundary doesn't grant permission to an IAM identity; it only sets the maximum permissions that the entity can have. You can define one or more permissions boundaries using managed policies.

Permission Boundary on IAM User Account

The following steps will show you how to set a permission boundary for new and existing IAM users.

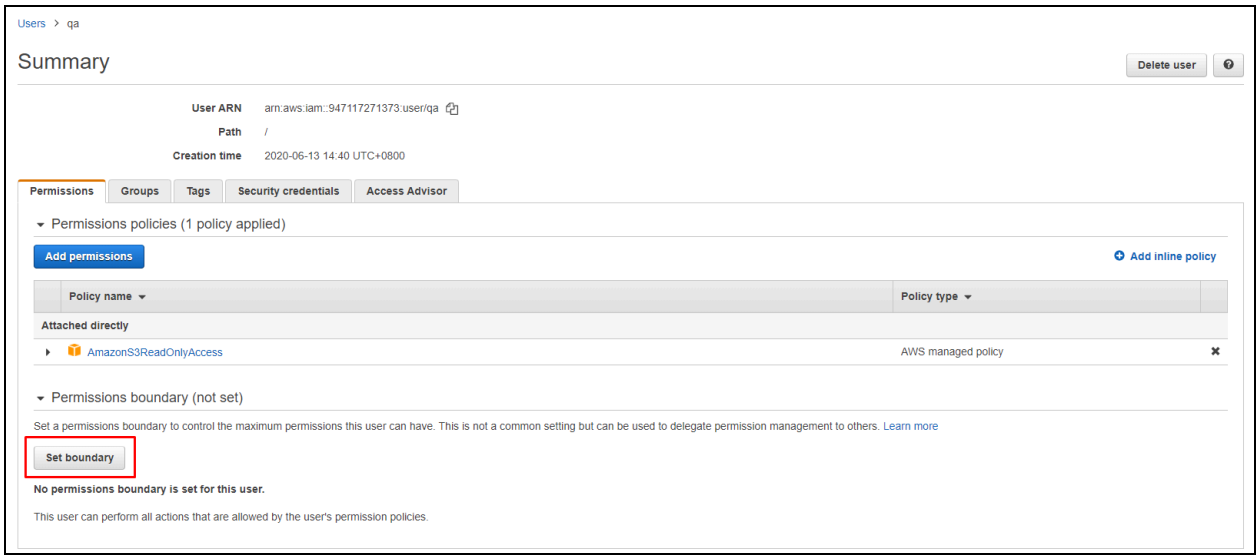
1. Define a permission boundary. Below is a sample permission boundary on JSON format. It provides full access to AWS Lambda and read access to Amazon EC2 and S3 on all resources when attached to an entity.

IAMBoundaryDeveloper

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "ec2:Describe*",
                "s3:Get*",
                "lambda:*",
                "s3:Describe*",
                "ec2:Get*"
            ],
            "Resource": "*"
        }
    ]
}
```

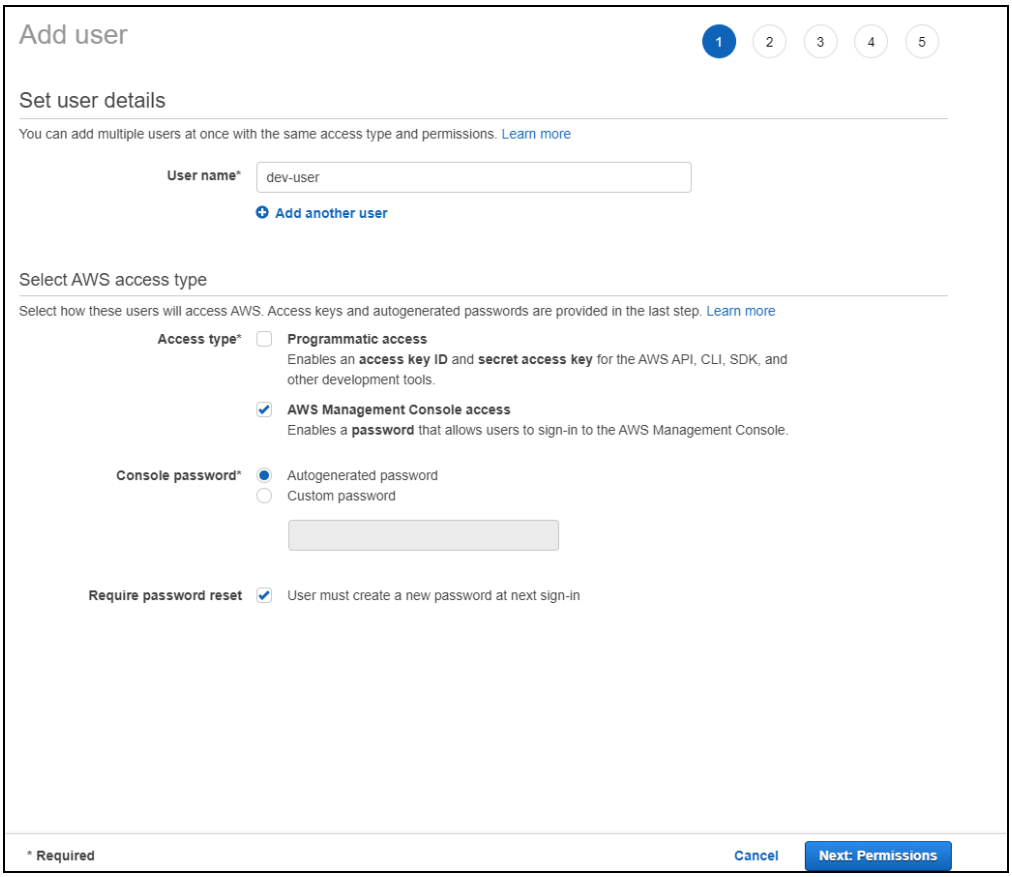
2. You can then attach this policy on a new or existing IAM user to set its permission boundary.

   For existing IAM User, go to IAM → Select a User → Permissions → Set Boundary.

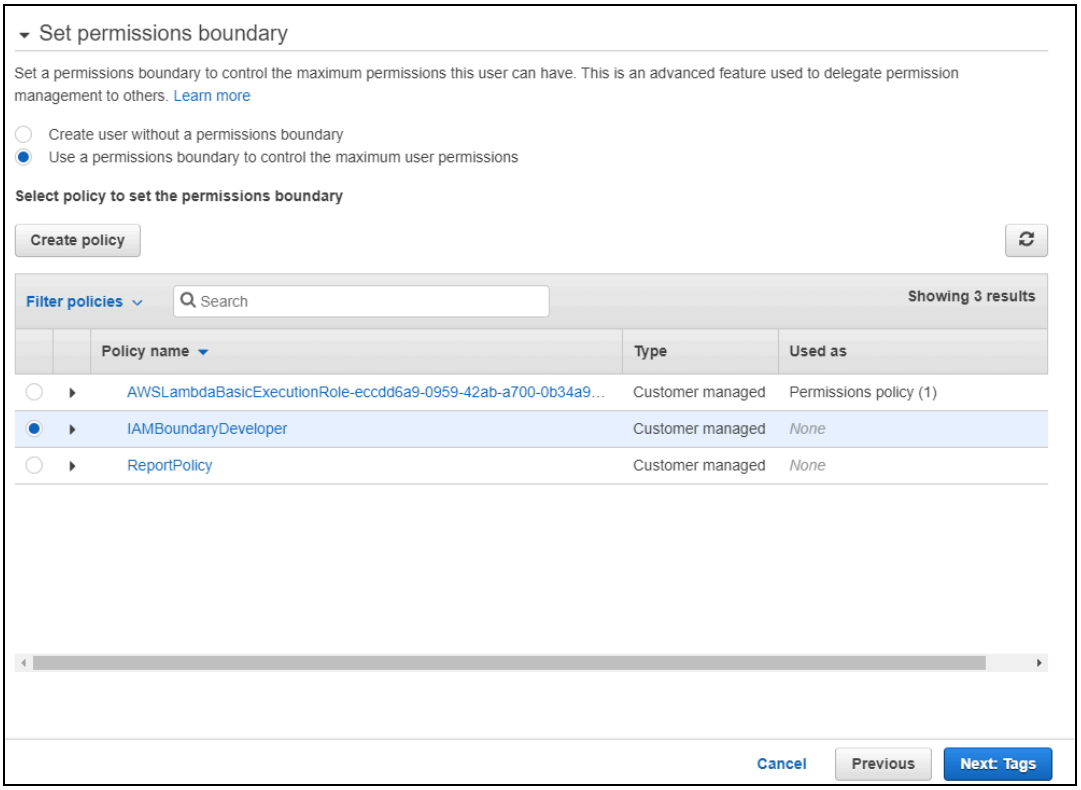


Permission Boundaries for new IAM user

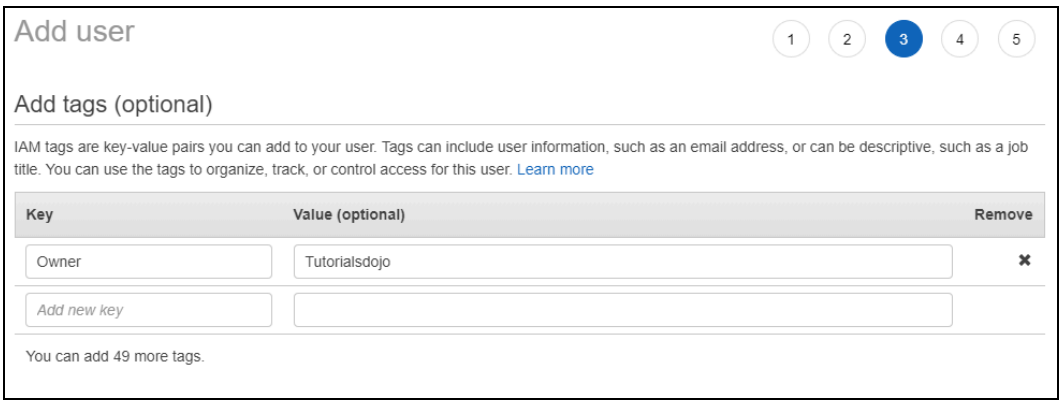
1. Go to IAM → Users → Add user to create a new IAM user. Click *Next: Permissions.*



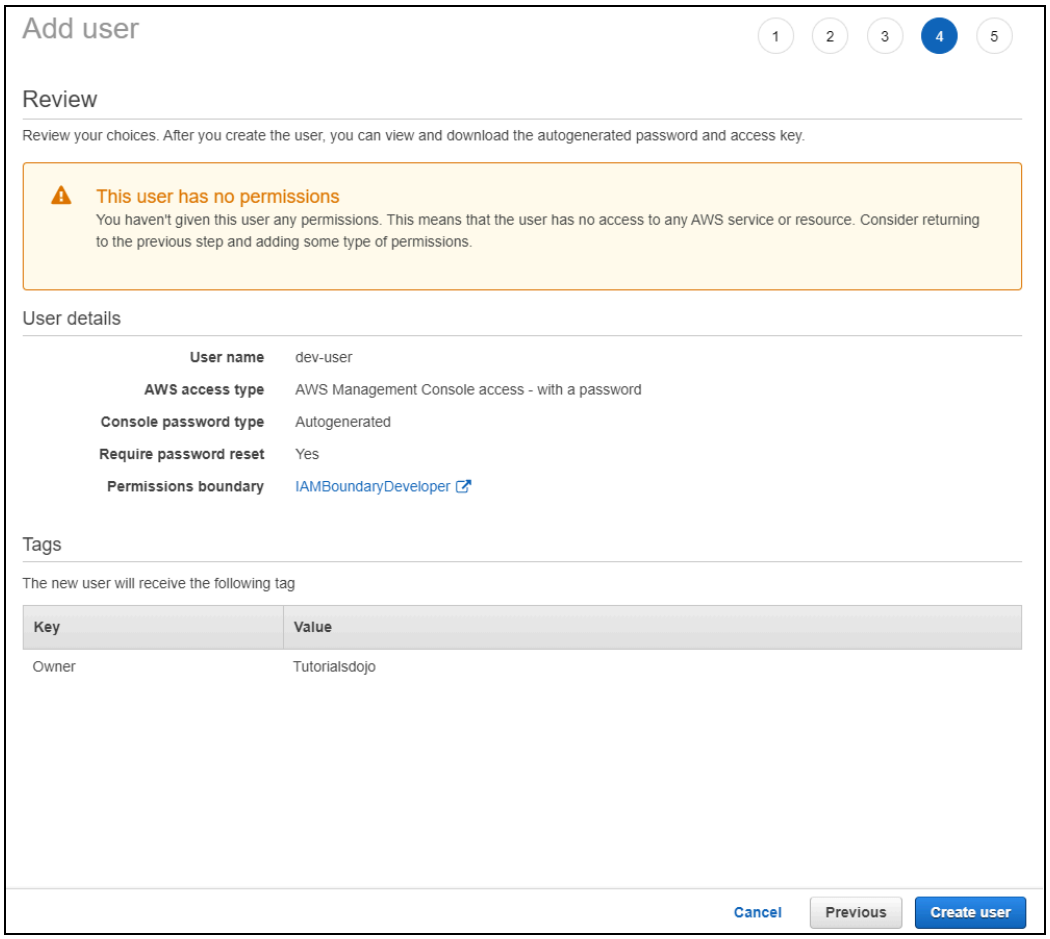
2. On the Permission Boundary, select Customer Managed on the filter, then select the policy we just created. Leave the User Permissions empty for now. Click Next: Tags.



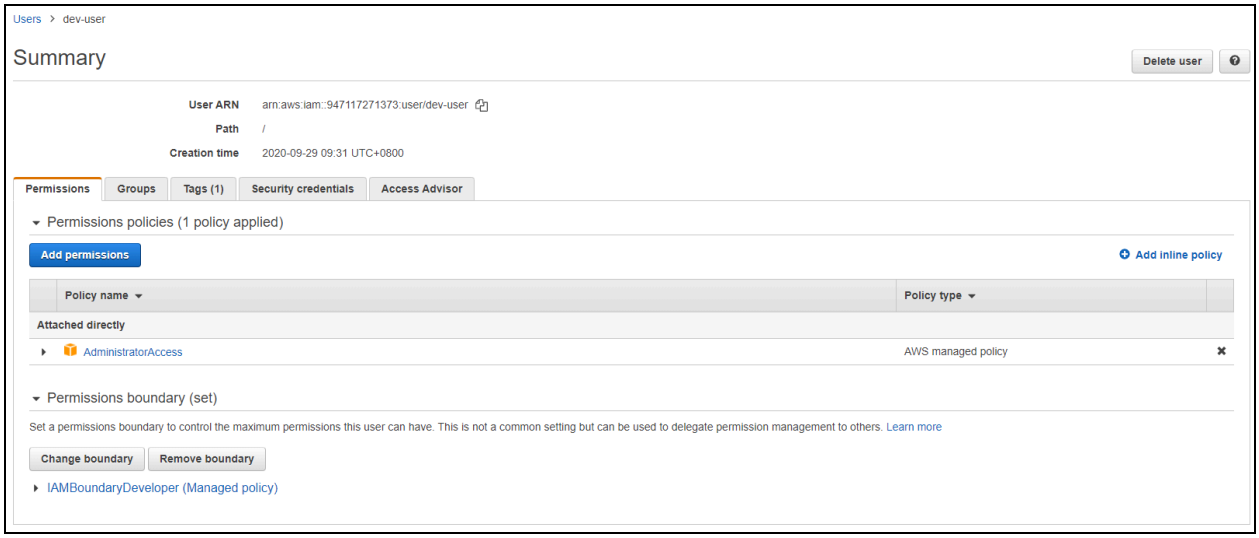
3. Add tags.



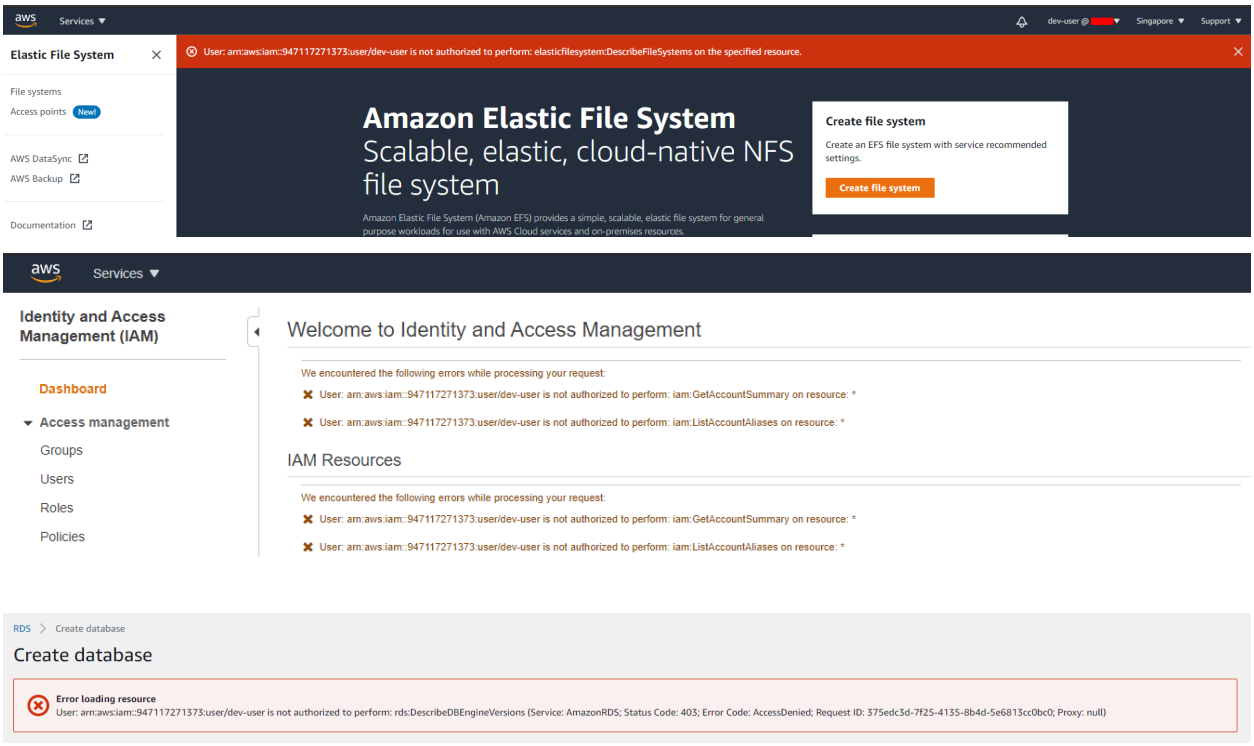
4. Review the details, then click Create User.



To see the permission boundary in action, we attached the AdministratorAccess policy to the IAM user *dev-user*.



Log in on the AWS Console using the *dev-user* account and access different AWS Services. Notice that even though the *dev-user* has an admin-level policy attached to it, access is still bounded within the services we defined on its permission policy. Accessing other services will give us permission errors.



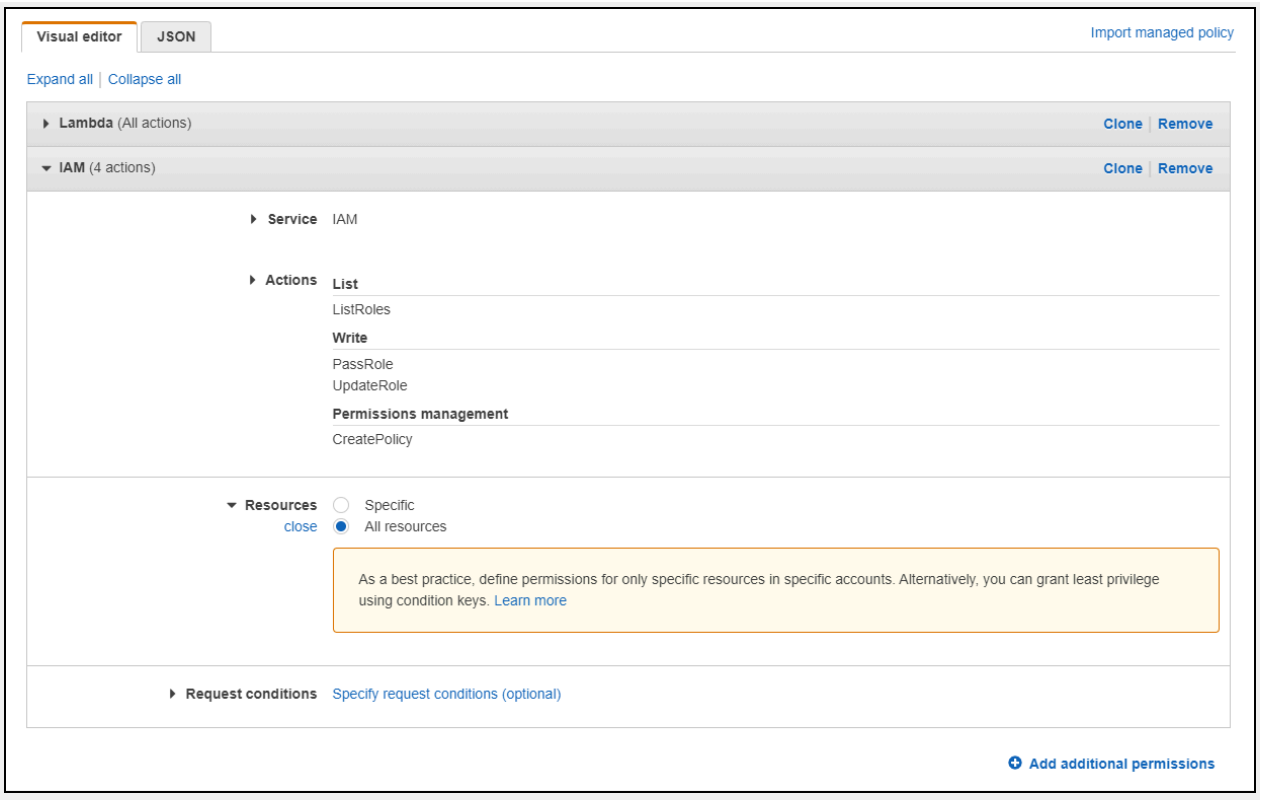
Using Permissions Boundaries as Request Condition

Another way to utilize permission boundaries is through request conditions on permission policy. Let's say Jon, one of the developers, needs to create an IAM role and policies for his AWS Lambda functions. You then give him access using the below policy, but it also allows him to create any role policies, including Administrator Access. This is a management risk since you don't want any developer to grant admin-level access to a role.

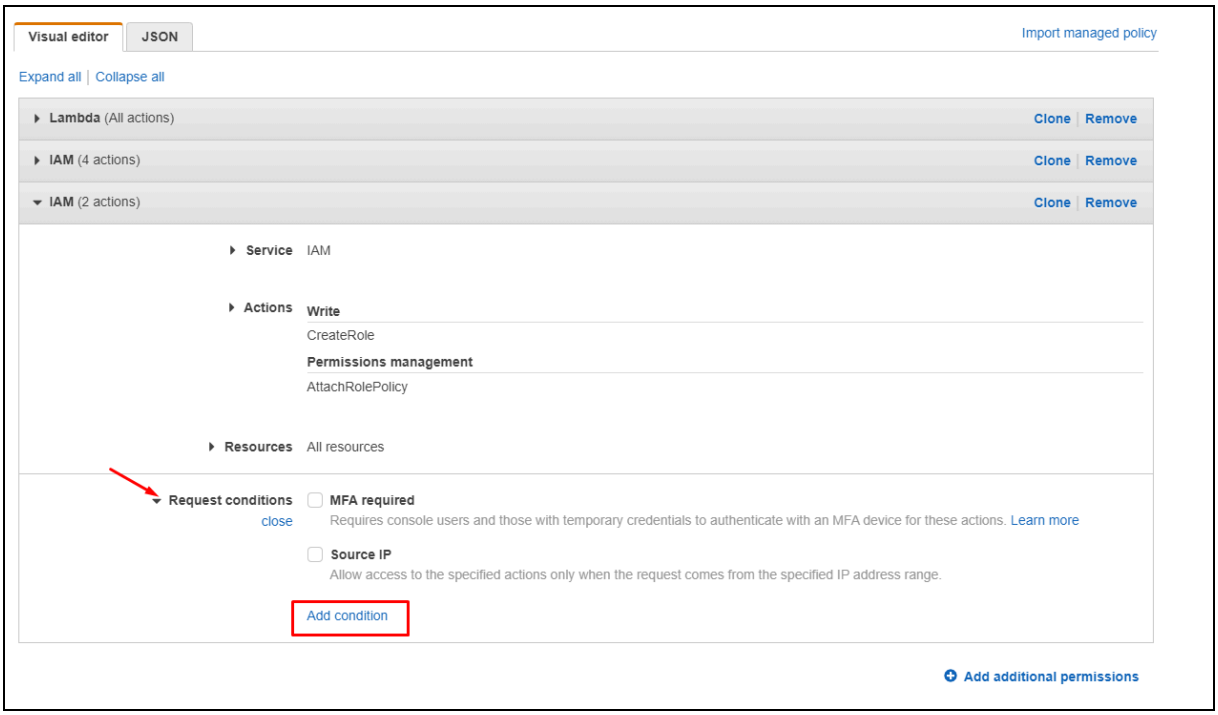
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "iam:CreatePolicy",
                "iam:Get*",
                "iam:List*",
                "iam:PassRole",
                "lambda:*",
                "iam:CreateRole",
                "iam:UpdateRole",
                "iam:AttachRolePolicy"
            ],
            "Resource": "*"
        }
    ]
}
```

To eliminate this risk, you can create a permission boundary and use this as a request condition for creating and updating roles. Let's use the permission boundary *IAMBoundaryDeveloper* we created in the previous demonstration.

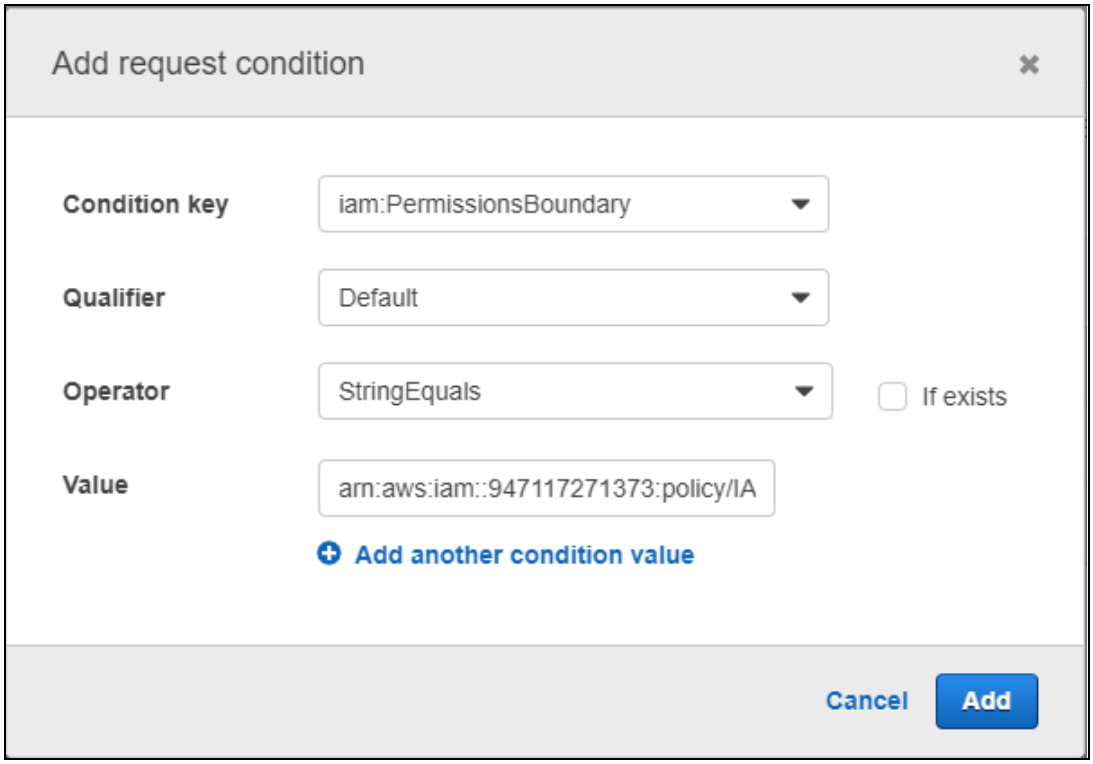
To use the request condition, let's go to Jon's user policy and remove the ***CreateRole*** and ***AttachRolePolicy***.



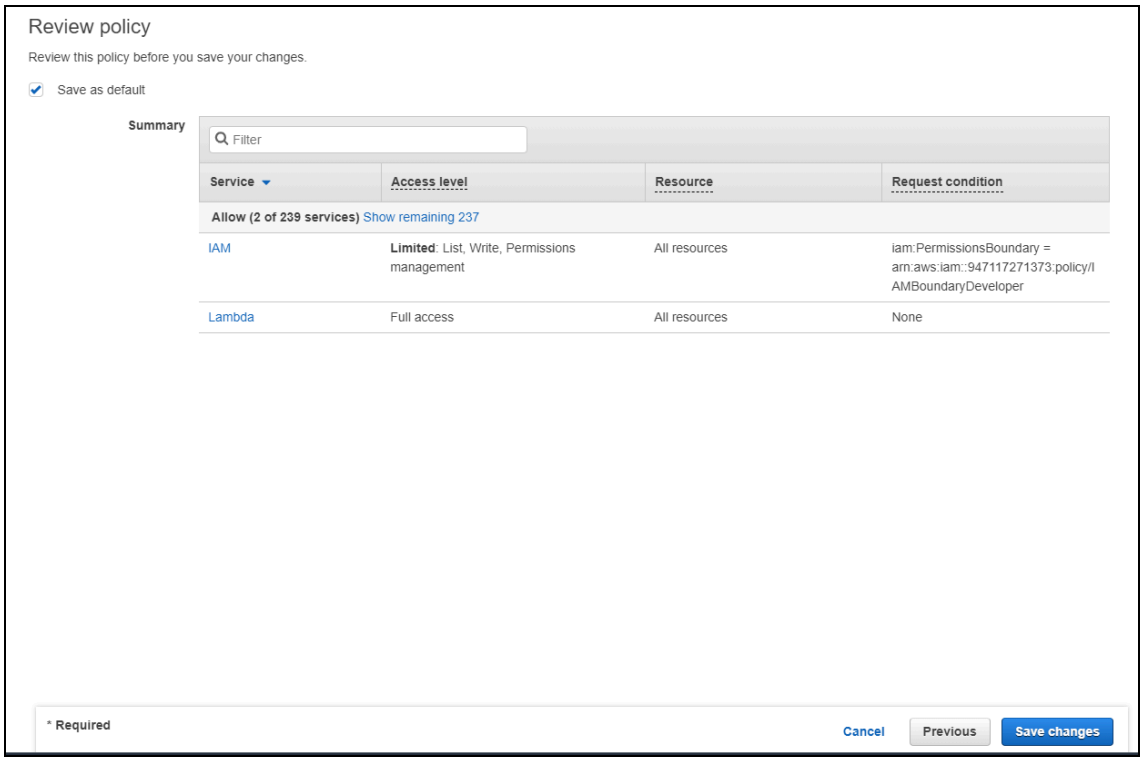
Add the ***CreateRole*** and ***AttachRolePolicy*** again on separate permission. This time we set request conditions for these operations. Click request conditions, then click Add Condition.



Select the following values for condition key, qualifier, and operator. For the value, paste the ARN of the *IAMBoundaryDeveloper*. Click Add.



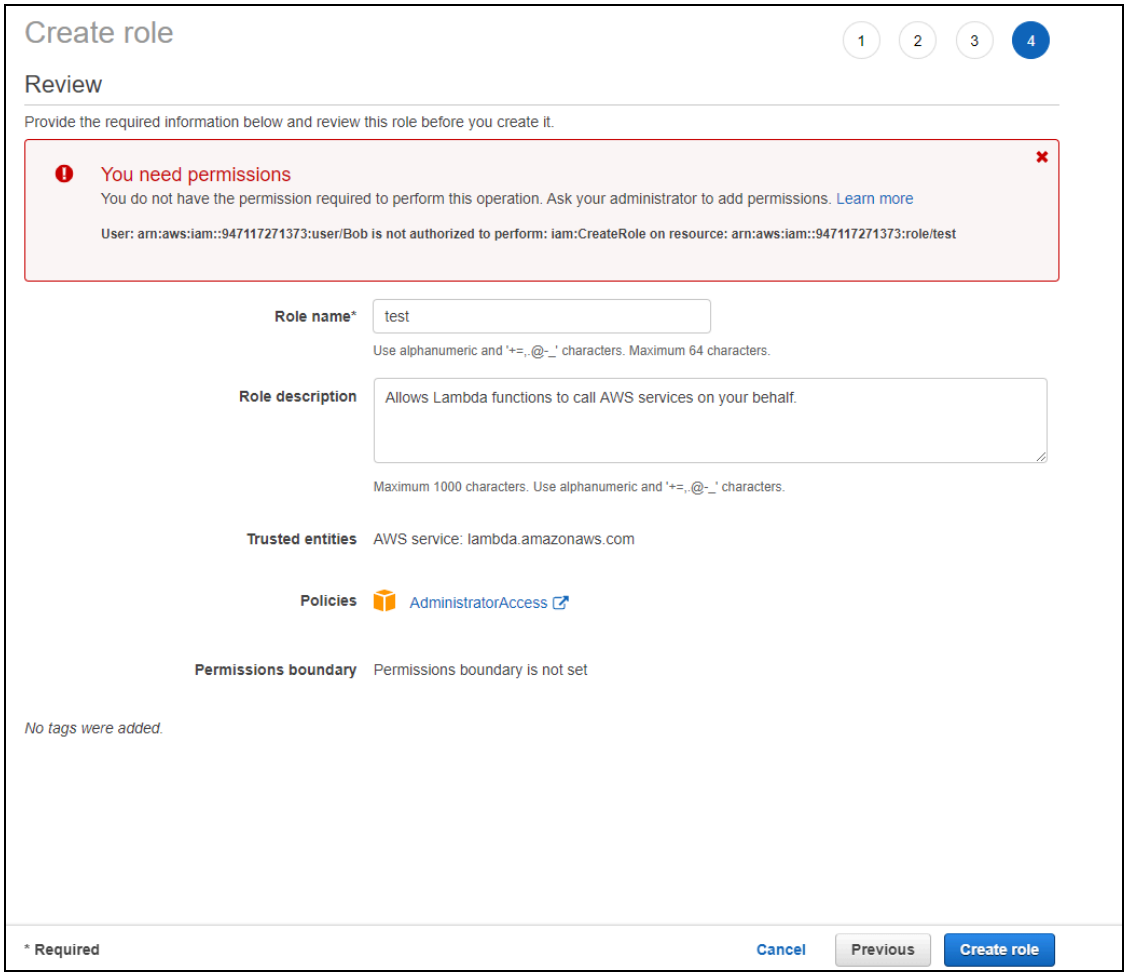
Review your policy then click Save Changes.



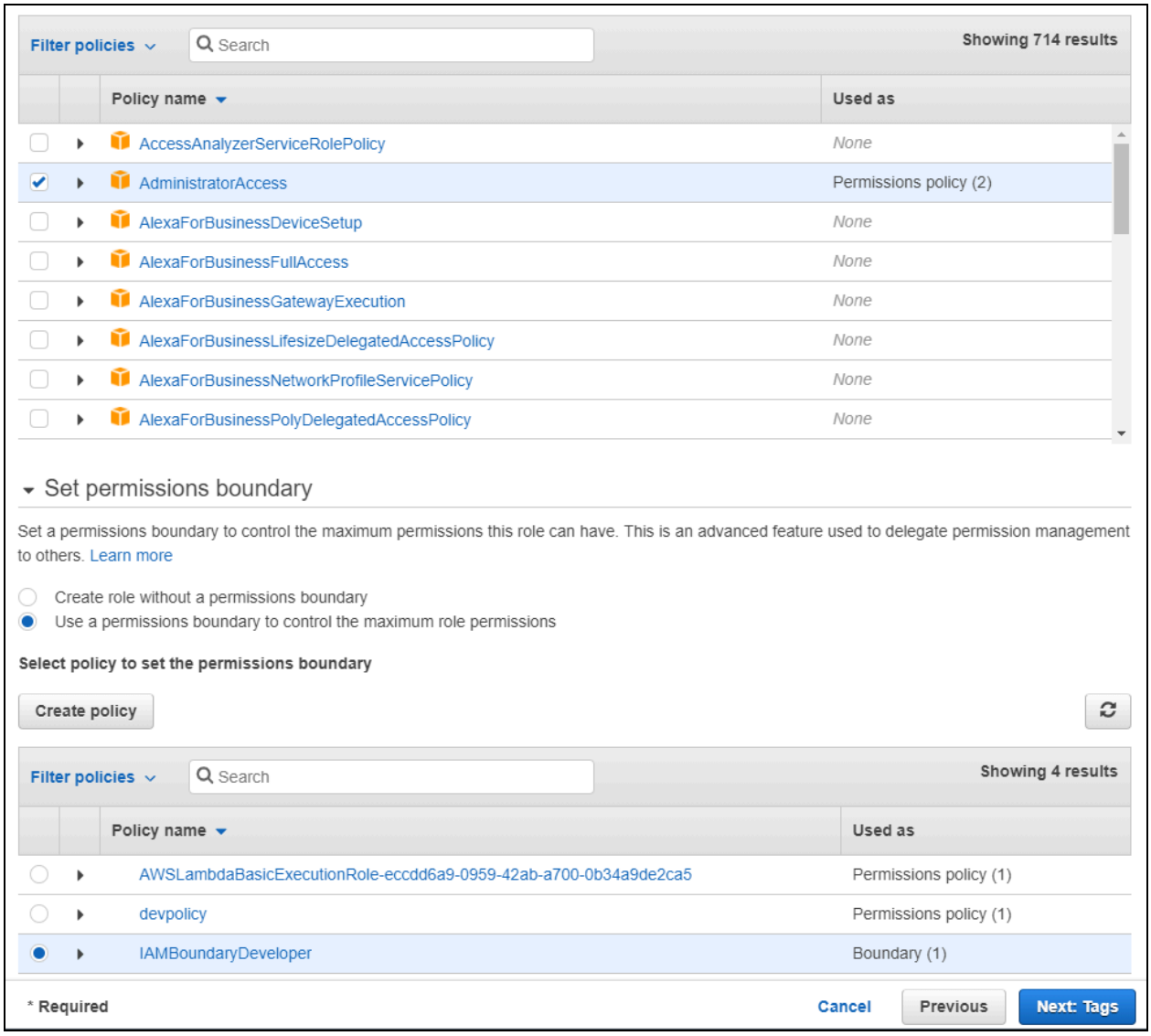
Here's the policy on JSON format. Notice the Condition for ***CreateRole*** and ***AttachRolePolicy*** operations.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "iam:CreateRole",
                "iam:AttachRolePolicy"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PermissionsBoundary":
"arn:aws:iam::947117271373:policy/IAMBoundaryDeveloper"
                }
            }
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "iam:CreatePolicy",
                "iam:Get*",
                "iam:List*",
                "iam:PassRole",
                "lambda:*",
                "iam:UpdateRole"
            ],
            "Resource": "*"
        }
    ]
}
```

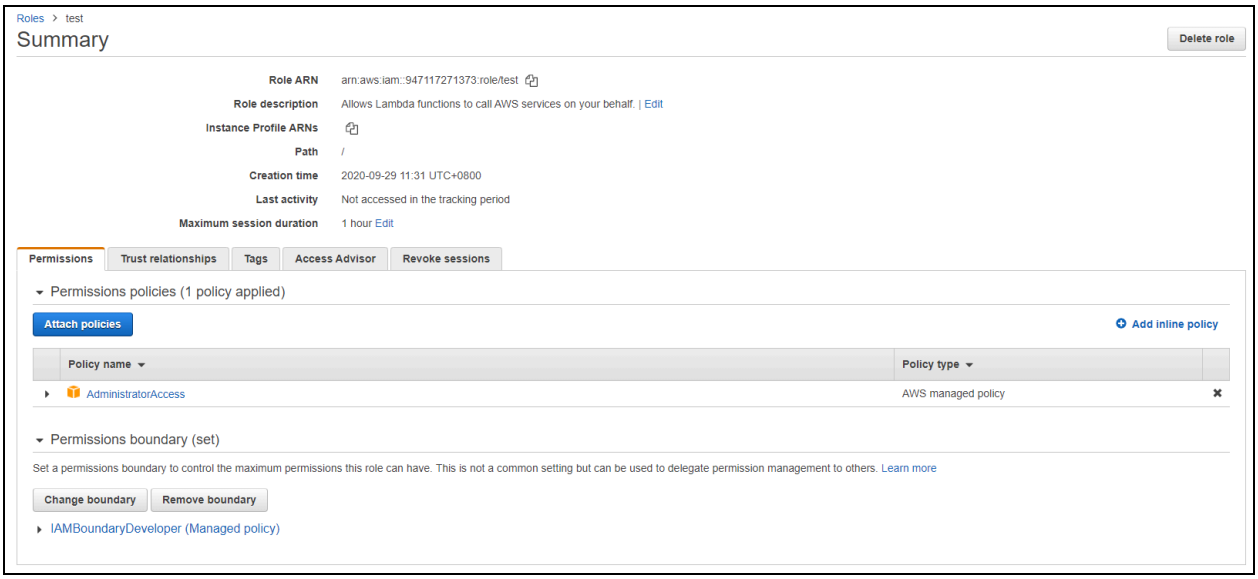
Let's login to the AWS Console and try to create some IAM Roles. As expected, we can't create an IAM Role without adding a permission boundary.



Let's try to create a Role with *AdministratorAccess* but now defining a permission boundary using the *IAMBoundaryDeveloper* policy.



Notice that Role creation is successful. But because a permission boundary is set, its access is only limited to the services we define on *IAMBoundaryDeveloper* though it has an admin-level policy.



References:

https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html#policies_bound
https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_boundaries.html
https://aws.amazon.com/blogs/security/delegate-permission-management-to-developers-using-iam-permissions-boundaries/

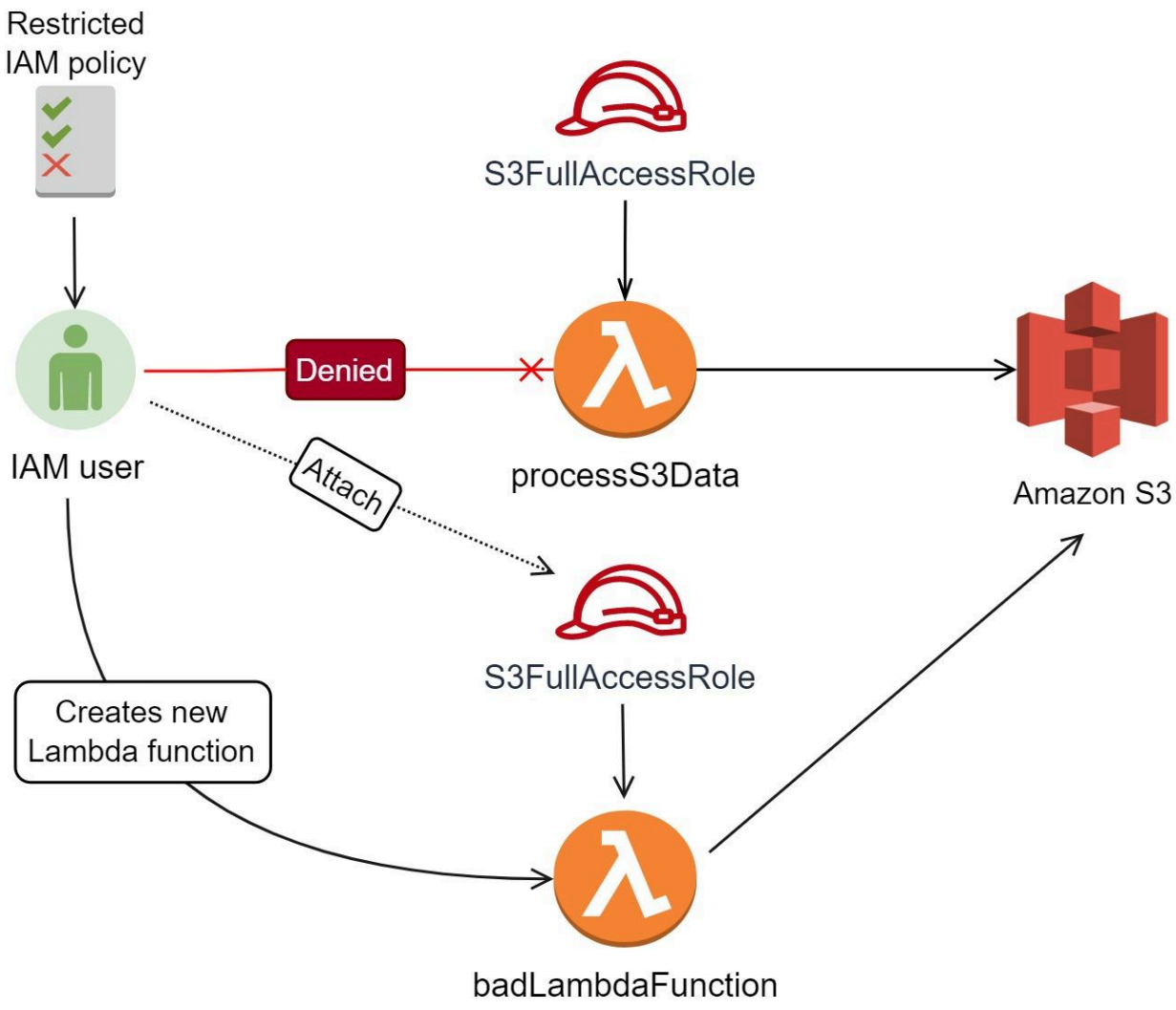
Using the iam:PassRole permission to pass an IAM Role to AWS Services

The **`iam:PassRole`** action grants a user the permission to attach IAM roles to an AWS resource. This is simple yet powerful permission that warrants due scrutiny when constructing IAM Policies. You can, for example, use the `PassRole` permission to prohibit certain IAM users from passing roles that have greater permissions than the user is allowed to have. Let me paint a scenario for you to explain this.

Say your company has a Lambda function that processes data stored on Amazon S3. An execution role with full S3 access is attached to the Lambda function. The Lambda function runs per schedule and only admins are authorized to make code changes to it. Even though you don't have access to the Lambda function, you can bypass the permissions given to you if the following policy is attached to your IAM user.

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

The preceding IAM policy allows you to pass any IAM roles that exist on your company's AWS account. This creates a security hole that can be exploited to elevate your access. Given you have enough permissions to create Lambda functions, all you have to do is launch a fresh Lambda function and attach the execution role that has full S3 access. Doing so, even if your IAM user lacks S3 permissions, you'd still be able to access Amazon S3 using the Lambda function.


Restricted
IAM policy
S3FullAccessRole
IAM user
Denied
processS3Data
Amazon S3
Attach
S3FullAccessRole
Creates new
Lambda function
badLambdaFunction

For better security, be more granular when it comes to providing access. For example, list the specific IAM roles that a user can pass rather than just using a wildcard, like what's shown below:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [
                "arn:aws:iam::123456789123:role/td-cloudwatch-agent-role",
                "arn:aws:iam::123456789123:role/td-s3-role"
            ]
        }
    ]
}
```

References:
https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_passrole.html
https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_examples_iam-passrole-service.html

Mapping Permissions of Active Directory User Attributes to AWS Services

AD Connector is a proxy service that provides an easy way to connect compatible AWS applications to your existing on-premises Microsoft Active Directory. It also enables you to streamline identity management and reuse your existing security policies from your Active Directory.

After you configured the AD Connector, the service will allow you to:

- **Sign in to AWS applications** such as Amazon WorkSpaces, Amazon QuickSight, and Amazon EC2 for Windows Server instances using your Active Directory credentials.
- Join your EC2 Windows instances to your on-premises Active Directory domain through AD Connector using a **seamless domain join**.
- Use **Federated sign-in** to log in to the AWS applications. AD Connector forwards sign-in requests to your on-premises Active Directory domain controllers for authentication.

Remember that AD Connector is designed to establish a trusted relationship between your Active Directory and AWS. With IAM Role, you can customize trust relationships that will allow the role to access AWS resources. You must have a trust relationship with AWS Directory Service first before assigning an IAM role to your users.

After the configuration between on-premises AD and AWS, you can now assign a user or group to an IAM role that will allow your directory users to access the AWS Management Console. The role assignment that you created will define what service a user or group can access. If you successfully created a role assignment, you can now verify the user's role and Id. The next time the user signs in to the AWS Management Console, the user will be signed in under the assigned role.

In short, role mapping will define what AWS services can be accessed by Active Directory users.

References:
https://aws.amazon.com/blogs/security/how-to-connect-your-on-premises-active-directory-to-aws-using-ad-connector/
https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_identifiers.html
https://aws.amazon.com/premiumsupport/knowledge-center/enable-active-directory-console-access/

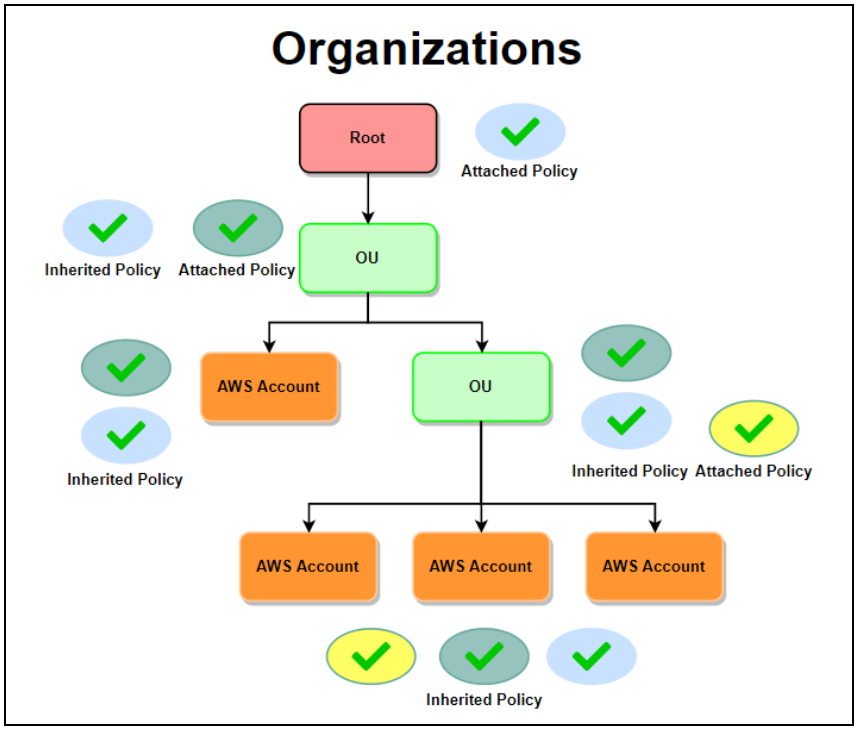
Using AWS Organizations to Provide Access to Third-Party AWS Accounts

AWS Organizations was introduced by AWS to ease the process of managing multiple AWS Accounts. AWS defines this service as an account management service that enables customers to consolidate multiple AWS accounts into an organization that they create and centrally manage.

Customers mainly benefit from AWS Organizations in terms of billing and account management. Instead of paying bills from each AWS Account, customers can consolidate and pay all bills on their AWS Master Account. This makes the billing process simple and saves money by leveraging *Pay less by using more* pricing principles.

You can centrally manage all of your AWS Accounts from your Master Account. The master account is the account used to create the Organization. You can then create an AWS account or invite an existing AWS Account under your Organization, which is considered a member account.

You can also have a hierarchical grouping on your AWS Organization. You can do this by creating an Organizational Units (OU), a logical group or container for your accounts. OUs may either have an AWS Account or another OU into it. You can attach policies to control permissions on any level on your Organizational hierarchy; these policies are called Service Control Policies (SCPs). When you attach a policy to one of the hierarchy nodes, it flows down and affects all the branches (OUs) and leaves (accounts) beneath it.



Using Service Control Policies (SCPs) to Restrict Permissions of Root Users in Member Accounts

Service control policies (SCPs) are a type of organization policy that you can use to manage your organization's permissions. SCPs are similar to IAM permissions policies except that they don't grant any permissions. Instead, SCPs specify the maximum permissions for an AWS Organizations entity (root, organizational unit, or account). When you attach an SCP to your organization root or an OU, the SCP limits permissions for entities in member accounts.

Since SCPs don't grant any permissions, you still need to attach identity-based or resource-based policies to IAM users, roles, or the resources in your organization's accounts to grant permissions.

SCPs affect only IAM users and roles that are managed by accounts which are part of the organization. SCPs don't affect resource-based policies directly; an example of this is Amazon S3 bucket policies. SCP also affects the root user of any member account within the organization.

Service Control Policy Limits

You can't use SCPs to restrict the following tasks:

- Any action performed by the master account
- Any action performed using permissions that are attached to a service-linked role
- Register for the Enterprise support plan as the root user
- Change the AWS support level as the root user
- Manage Amazon CloudFront keys
- Provide trusted signer functionality for CloudFront private content
- Modify AWS account email allowance/reverse DNS
- Tasks on some AWS-related services:
    - Alexa Top Sites
    - Alexa Web Information Service
    - Amazon Product Marketing API

Exceptions for only member accounts created before September 15, 2017

For some accounts created before September 15, 2017, you can't use SCPs to prevent the root user in those member accounts from performing the following tasks:

- Enable or disable multi-factor authentication on the root user
- Create, update, or delete x.509 keys for the root user
- Change the root user's password
- Create, update, or delete root access keys

Note: *For all accounts created after September 15, 2017, these exceptions don't apply, and you can use SCPs to prevent the root user in those member accounts from performing these tasks.*

References:
https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_policies_about-scps.html
https://aws.amazon.com/organizations/faqs/
https://docs.aws.amazon.com/organizations/latest/userguide/orgs_getting-started_concepts.html
https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_policies_scps.html

Using Resource Control Policies (RCPs) to Restrict Access to Resources in Member Accounts

RCPs are complementary to service control policies (SCPs) but focus on resource-centric controls rather than principal-centric controls. While SCPs limit what IAM principals within your organization can do, RCPs limit who can access resources in your organization, regardless of whether those principals belong to your organization or not. This makes RCPs particularly valuable for establishing data perimeters and preventing external access to sensitive resources. When you attach an RCP to your organization root, organizational unit (OU), or individual account, it restricts access to resources in those entities even when accessed by principals outside your organization. Like SCPs, RCPs do not grant any permissions but only define maximum permissions, meaning you must still attach identity-based or resource-based policies to actually grant access.

RCPs impact the effective permissions of all principals trying to access resources in member accounts with applicable RCPs, including root users and external principals, with the exception of service-linked roles which cannot be restricted by RCPs. At launch, RCPs support a limited set of AWS services including Amazon S3, AWS Security Token Service (STS), AWS Key Management Service (KMS), Amazon Simple Queue Service (SQS), and AWS Secrets Manager, with additional services expected to be added over time. RCPs are evaluated in conjunction with other policies—the effective permissions are the logical intersection of the RCP and identity-based or resource-based policies. A typical use case is to enforce that only principals within your organization can access S3 buckets, preventing data exfiltration even if developers inadvertently configure overly permissive bucket policies. Organizations can use AWS IAM Access Analyzer to identify resources with external access and assess the impact of RCPs before enforcement, as IAM Access Analyzer now includes a "Resource control policy restriction" field indicating whether an RCP influences access findings.

Resource Control Policy Limits

RCPs have several important constraints and limitations that organizations must consider:

- Size and Structure Limits:
  - An RCP cannot exceed 5,120 characters in total size
  - An RCP can contain a maximum of five individual statements
  - The Principal field must always be set to "*" with actual restrictions implemented through the Condition element
  - RCPs can only use "Deny" effects and cannot grant permissions
- RCPs do not apply to:
  - Any action on resources in the management account
  - Any action performed by service-linked roles

References:
https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_policies_rcps.html
https://aws.amazon.com/blogs/aws/introducing-resource-control-policies-rcps-a-new-authorization-policy/
https://aws.amazon.com/about-aws/whats-new/2024/11/resource-control-policies-restrict-access-aws-resources/
https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_policies_rcps_examples.html
https://docs.aws.amazon.com/IAM/latest/UserGuide/access-analyzer-findings-filter.html

Declarative policies - Service-level Configuration Enforcement

Unlike authorization policies (SCPs and RCPs) that regulate API access, declarative policies are enforced directly in the service's control plane to enforce durable intent. This distinction ensures baseline configurations remain enforced even when services introduce new features or APIs. Key benefits include "set once and forget" maintenance, configurations automatically apply to new accounts joining the organization, new principals created, and new resources provisioned without requiring policy updates. Declarative policies prevent non-compliant actions regardless of invocation method: IAM roles, root users, or service-linked roles (service-linked roles are exempt from most restrictions but subject to declarative policy enforcement). When a declarative policy is detached, attribute states roll back to previous configurations before policy attachment. The effective policy inherits rules from the organization root and OUs combined with directly attached account policies, determining final enforcement.

Declarative policies support Amazon EC2, Amazon VPC, and Amazon EBS with additional services planned. Supported attributes include:

- **VPC Block Public Access:** Controls internet gateway communication with modes (off, block_ingress, block_bidirectional); enables or disables exclusion creation
- **Image Block Public Access:** Controls AMI sharing with states (unblocked, block_new_sharing)

- **Snapshot Block Public Access:** Controls EBS snapshot public access with states (unblocked, block_new_sharing, block_all_sharing)
- **Allowed Image Settings:** Restricts AMI usage to specific providers (amazon, aws_marketplace, aws_backup_vault, or 12-digit account IDs); supports up to 200 providers and up to 10 criteria named criteria_1 through criteria_10; operates in enabled, disabled, or audit_mode states
- **Instance Metadata Defaults:** Configures http_tokens for IMDSv2 enforcement, http_put_response_hop_limit for token travel distance, http_endpoint accessibility, instance_metadata_tags access
- **EC2 Serial Console Access:** Controls serial console accessibility with enabled or disabled status

Policy syntax uses JSON structure starting with `ec2_attributes` fixed key name, supporting operators like `@@assign` and `@@append` for value assignment and list manipulation. Custom error messages configure via exception_message attribute, enabling administrators to redirect users to internal wikis, ticketing systems, or provide actionable remediation guidance. AWS recommends avoiding personally identifiable information (PII) in custom messages. Default message when unspecified: "This action is denied due to an organizational policy in effect."

Account status reports provide visibility into current configuration state for all declarative policy attributes across specified scope (individual accounts, OUs, or entire organization root). Reports generated asynchronously via StartDeclarativePoliciesReport API (management account or delegated administrator only), saved to S3 using path structure: s3://bucket/prefix/ec2_TARGETID_REPORTID_YYYYMMDDTHHMMZ.csv. Reports display:

- numberOfMatchedAccounts: Uniform configuration across accounts
- numberOfUnmatchedAccounts: Inconsistent configuration
- Most frequent value: Most commonly observed configuration
- Region breakdown: Configuration status by Region

Only one report per organization can generate at a time; concurrent attempts return errors.

References:
https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_policies_declarative.html
https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_policies_declarative_syntax.html
https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_policies_declarative_best-practices.html
https://docs.aws.amazon.com/cli/latest/reference/ec2/start-declarative-policies-report.html
https://docs.aws.amazon.com/controltower/latest/controlreference/declarative-controls.html

AI services opt-out policies
AWS AI services typically store and use customer content for service improvement (operational issues, performance evaluation, debugging, model training), potentially in Regions outside where services are used. Organizations use AI services opt-out policies to control data collection across all accounts.

Opting out deletes all associated historical content previously shared with AWS (limited to service improvement content, not content required for service functions). Policies can target individual AI services or all services using "default" key, which represents all current AI services and automatically includes future AI services AWS might add.

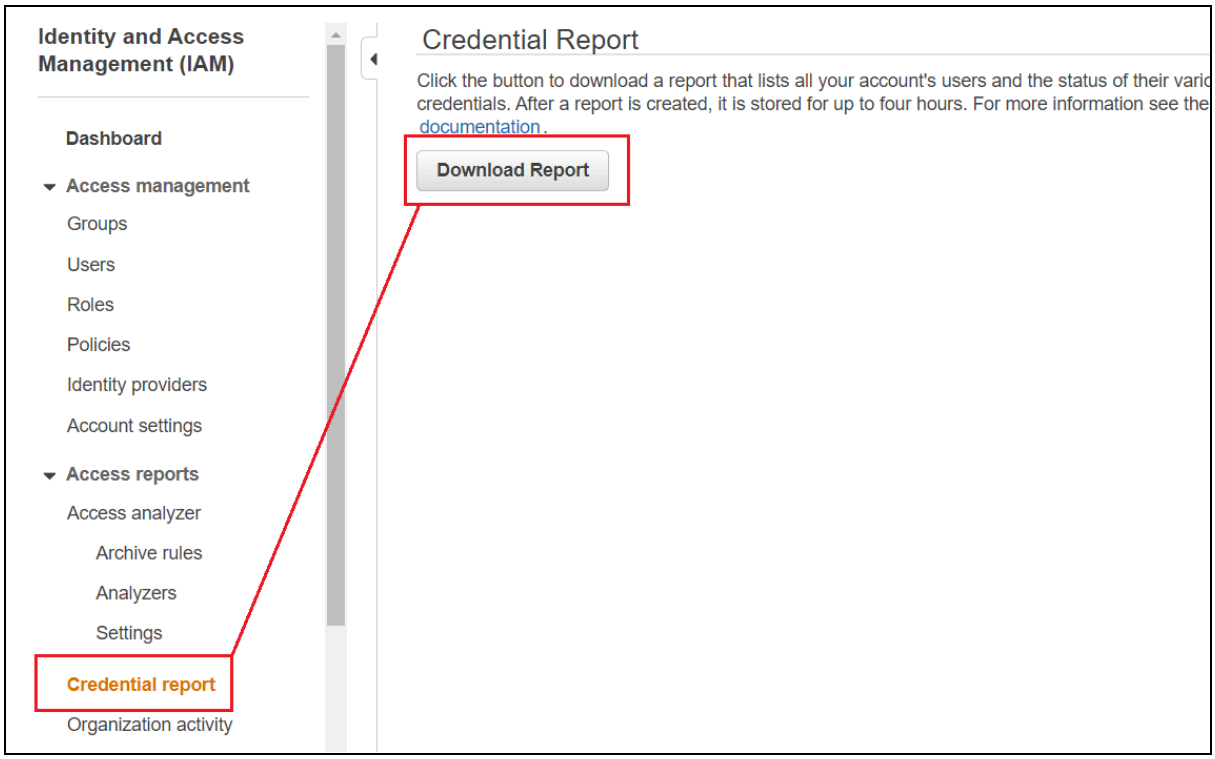
AI services opt-out policies use JSON structure with services element containing service name keys. The opt_out_policy key supports only @@assign operator with two values: optOut (opt out of content use) or optIn (opt in to content use). Inheritance control uses @@operators_allowed_for_child_policies with @@assign (child policies can override) or @@none (prevents child policy overrides). Operators @@append, @@remove, and @@enforced_for are not supported.

References:
https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_policies_ai-opt-out_syntax.html
https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_policies_ai-opt-out.html

Generating Credential Reports for your AWS Account Using AWS IAM

A Credential report is a CSV file that contains different information about all of the users in an AWS account. Some of the information includes the user creation date, ARN, the last time a key was rotated, and the last time a password or an access key was used or changed.

You can generate and download a credential report programmatically or through the IAM Console. To download a report, scroll down and find the "Credential report" on the left pane of IAM. Click the "Download Report" button.



The manual method is good enough for audit purposes. However, if you need to automate a process to meet a compliance requirement like disabling an access key after a certain amount of days, you must use IAM APIs.

Let's say you want to invalidate access keys that are more than 90 days old. First, you'll need the required information from the credential report. The access key age can be found under the "*access_key_1_last_rotated*" column. The N/A values are for those IAM users that don't have programmatic access.

	G	H	I	J	K
1	password_next_rotation	mfa_active	access_key_1_active	access_key_1_last_rotated	access_key_1_last_used_date
2	not_supported	false	false	N/A	N/A
3	N/A	false	false	N/A	N/A
4	N/A	false	true	2020-08-18T05:34:10+00:00	2020-10-08T17:11:00+00:00
5	N/A	false	true	2020-07-14T06:46:10+00:00	N/A
6	N/A	false	false	N/A	N/A
7	N/A	false	true	2020-07-09T09:22:32+00:00	2020-08-21T05:11:00+00:00
8	N/A	false	true	2020-09-30T06:58:31+00:00	2020-09-30T07:18:00+00:00
9	N/A	false	false	N/A	N/A
10					

You can automate the process by writing a script or a Lambda Function that will call the **GenerateCredentialReport**, **GetCredentialReport**, and **UpdateAccessKey APIs**. The first API command is used to tell IAM to generate a report. After generating, you execute the second API to download the report. Write a logic that will parse the CSV file to get the user with an access key age greater than 90 days. Use the third API command to modify the status of the access key to "**Inactive**."

References:
https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_getting-report.html
https://docs.aws.amazon.com/IAM/latest/APIReference/API_GenerateCredentialReport.html

Choosing the Most Suitable AWS STS API for Authentication

AWS Security Token Service (AWS STS) is a global web service that allows you to generate temporary access for IAM users or federated users to gain access to your AWS resources. These temporary credentials are limited-privilege and are for short-term use only. Once expired, these temporary credentials can no longer be used to access your AWS resources.

AWS STS can't be accessed on the AWS console; it is only accessible through API. All STS requests go to a single endpoint at *https://sts.amazonaws.com/*, and logs are then recorded to AWS CloudTrail.

AWS STS API Operations

AssumeRole - *cross-account delegation and federation through a custom identity broker*

The `AssumeRole` API operation lets an IAM user assume an IAM role belonging to your account or to an external one (cross-account access). For the request to succeed, you have to specify a valid IAM user to the trusted entities of the IAM role. Once the request is successful, AWS generates and returns temporary credentials consisting of an access key ID, a secret access key, and a security token. These credentials can then be used by the IAM user to make requests to AWS services.

AssumeRoleWithWebIdentity - *federation through a web-based identity provider*

The `AssumeRoleWithWebIdentity` API operation returns temporary security credentials for federated users who are authenticated through a public identity provider (e.g., Amazon Cognito, Login with Amazon, Facebook, Google, or any OpenID Connect-compatible identity provider). The temporary credentials can then be used by your application to establish a session with AWS. Just like `AssumeRole`, trusted entities who will be assuming the role must be specified. This time, instead of IAM users, it'll be an identity provider.

`AssumeRoleWithWebIdentity` does not require IAM Identities credentials, making it suitable for mobile applications that require access to AWS. The `AssumeRoleWithWebIdentity` is one of the APIs that Amazon Cognito uses under the hood to facilitate the exchange of token and credentials on your behalf. Because Amazon Cognito abstracts the hassles associated with user authentication, it is recommended that you use Amazon Cognito when providing AWS access to application-based users. However, you may just use `AssumeRoleWithWebIdentity` as a standalone operation.

AssumeRoleWithSAML - *federation through an enterprise Identity Provider compatible with SAML 2.0*

The `AssumeRoleWithSAML` API operation returns a set of temporary security credentials for federated users who are authenticated by enterprise Identity Providers compatible with SAML 2.0. The users must also use SAML 2.0 (Security Assertion Markup Language) to pass authentication and authorization information to AWS. This API operation is useful for organizations that with existing identity systems (such as Windows Active Directory or OpenLDAP) that can produce SAML assertions.

GetFederationToken - *federation through a custom identity broker*

The `GetFederationToken` API operation returns a set of temporary security credentials consisting of a security token, access key, secret key, and expiration for a federated user. This API is usually used for federating access to users authenticated by a custom identity broker and can only be called using the programmatic credentials of an IAM user (IAM Roles are not supported). Although you can use the security credentials of a root user to call `GetFederationToken`, it is not recommended for security reasons. Instead, AWS advises creating an IAM user specifically for a proxy application that does the authentication process.

The default expiration period for this API is significantly longer than `AssumeRole` (12 hours instead of one hour). Since you do not need to obtain new credentials as frequently, the longer expiration period can help reduce the number of calls to AWS.

You can use the temporary credentials created by `GetFederationToken` in any AWS service except the following:

- You cannot call any IAM operations using the AWS CLI or the AWS API.
- You cannot call any STS operations except `GetCallerIdentity`.

GetSessionToken - *MFA-protect API calls to AWS*

The *GetSessionToken* API operation returns a set of temporary security credentials to an existing IAM user. The temporary security credentials consist of an access key ID, a secret access key, and a security token. You can set the operation to allow AWS requests only when MFA is enabled for the IAM user to provide enhanced security. IAM users can then make programmatic calls to API operations that require MFA authentication. If users provided the wrong MFA code, the API returns an access denied error. Since the credentials are temporary, you also have security from IAM users accessing your AWS resources through a less secure environment.

DecodeAuthorizationMessage

`DecodeAuthorizationMessage` decodes additional information about the authorization status of a request from an encoded message returned in response to an AWS request.

For example, a user calling an API to which he or she does not have access will encounter the `Client.UnauthorizedOperation` error. Some AWS operations additionally return an encoded message that gives details about a certain authorization failure. The message is encoded so that privilege details about the authorization are hidden from the user who requested the operation. To decode an authorization status message, one must be granted permission via an IAM policy to request the `DecodeAuthorizationMessage` action.

The decoded message includes the following types of information:

- Whether the request was denied due to an explicit deny or due to the absence of an explicit allow.
- The principal who made the request.
- The requested action.
- The requested resource.
- The values of condition keys in the context of the user's request.

GetAccessKeyInfo

`GetAccessKeyInfo` returns the AWS account's ID for the specified access key ID. This operation also identifies if the access key IDs are long-term credentials or temporary credentials from AWS STS. Access key IDs beginning with AKIA are long-term credentials, while access key IDs beginning with `ASIA` are temporary credentials. If you have access to the AWS account, which `GetAccessKeyInfo` returns, you can review your root user access keys and pull a credential report from your root user account to learn which IAM user owns the keys.

Example request

```
https://sts.amazonaws.com/
?Version=2011-06-15
&Action=GetAccessKeyInfo
&AccessKeyId=AKIAI44QH8DHBEXAMPLE
```

Example response

```
<GetAccessKeyInfoResponse xmlns="https://sts.amazonaws.com/doc/2011-06-15/">
  <GetAccessKeyInfoResult>
```

```
    <Account>123456789012</Account>
  </GetAccessKeyInfoResult>
 <ResponseMetadata>
    <RequestId>6a28c46d-368e-4fc4-9143-6e8eEXAMPLE</RequestId>
 </ResponseMetadata>
</GetAccessKeyInfoResponse>
```

GetCallerIdentity

`GetCallerIdentity` returns details about the IAM user or role that is used to call the operation. These details consist of the AWS account ID number, AWS ARN, and UserID.

Example 1 - Called by an IAM user.

This is an example of a request and response made using the credentials of user Bob in the AWS account 123456789012.

Example request

```
POST / HTTP/1.1
Host: sts.amazonaws.com
Accept-Encoding: identity
Content-Length: 32
Content-Type: application/x-www-form-urlencoded
Authorization: AWS4-HMAC-SHA256
Credential=AKIAI44QH8DHBEXAMPLE/20160126/us-east-1/sts/aws4_request,
        SignedHeaders=host;user-agent;x-amz-date,
        Signature=1122334455abcdef1122334455abcdef1122334455abcdef1122334455abcdef
X-Amz-Date: 20160126T215751Z
User-Agent: aws-cli/1.10.0 Python/2.7.3 Linux/3.13.0-76-generic botocore/1.3.22

Action=GetCallerIdentity&Version=2011-06-15
```

Example response

```
HTTP/1.1 200 OK
x-amzn-RequestId: 01234567-89ab-cdef-0123-456789abcdef
Content-Type: text/xml
Content-Length: 357
```

```
Date: Tue, 26 Jan 2016 21:57:47 GMT

<GetCallerIdentityResponse xmlns="https://sts.amazonaws.com/doc/2011-06-15/">
  <GetCallerIdentityResult>
   <Arn>arn:aws:iam::123456789012:user/Bob</Arn>
    <UserId>FDASG34G3DG43VEXAMPLE</UserId>
    <Account>123456789012</Account>
  </GetCallerIdentityResult>
  <ResponseMetadata>
    <RequestId>01234567-89ab-cdef-0123-456789abcdef</RequestId>
  </ResponseMetadata>
</GetCallerIdentityResponse>
```

Example 2 - Called by federated user created with AssumeRole.

This is an example of a request and response made with temporary credentials created by *AssumeRole*. The assumed role and the `RoleSessionName` are returned as shown on the example response.

Example request

```
POST / HTTP/1.1
Host: sts.amazonaws.com
Accept-Encoding: identity
Content-Length: 43
X-Amz-Date: 20160301T213302Z
User-Agent: aws-cli/1.10.0 Python/2.7.3 Linux/3.13.0-79-generic botocore/1.3.22
X-Amz-Security-Token:<REDACTED>
Content-Type: application/x-www-form-urlencoded
Authorization: AWS4-HMAC-SHA256
Credential=AKIAI44QH8DHBEXAMPLE/20160301/us-east-1/sts/aws4_request,
        SignedHeaders=host;user-agent;x-amz-date;x-amz-security-token,
        Signature=1122334455abcdef1122334455abcdef1122334455abcdef1122334455abcdef

Action=GetCallerIdentity&Version=2011-06-15
```

Example response

```
HTTP/1.1 200 OK
x-amzn-RequestId: 01234567-89ab-cdef-0123-456789abcdef
Content-Type: text/xml
```

```
Content-Length: 438
Date: Fri, 16 Oct 2020 20:12:47 GMT

<GetCallerIdentityResponse xmlns="https://sts.amazonaws.com/doc/2011-06-15/">
  <GetCallerIdentityResult>
    <Arn>arn:aws:sts::123456789012:assumed-role/sampleRole/sampleRoleSessionName</Arn>
    <UserId>FAS343DG43VEXAMPLE:sampleRoleSessionName</UserId>
    <Account>123456789012</Account>
  </GetCallerIdentityResult>
  <ResponseMetadata>
    <RequestId>01234567-89ab-cdef-0123-456789abcdef</RequestId>
  </ResponseMetadata>
</GetCallerIdentityResponse>
```

Example 3 - Called by a federated user created with GetFederationToken.

This is an example of a request and response made with temporary credentials created by using `GetFederationToken`. The operation returned the Name parameter with the value `sampleFederatedUser`.

Example Request

```
POST / HTTP/1.1
Host: sts.amazonaws.com
Accept-Encoding: identity
Content-Length: 43
X-Amz-Date: 20160301T215108Z
User-Agent: aws-cli/1.10.0 Python/2.7.3 Linux/3.13.0-79-generic botocore/1.3.22
X-Amz-Security-Token:<REDACTED>
Content-Type: application/x-www-form-urlencoded
Authorization: AWS4-HMAC-SHA256
Credential=AKIAI44QH8DHBEXAMPLE/20160301/us-east-1/sts/aws4_request,
        SignedHeaders=host;user-agent;x-amz-date;x-amz-security-token,
        Signature=1122334455abcdef1122334455abcdef1122334455abcdef1122334455abcdef

Action=GetCallerIdentity&Version=2011-06-15
```

Example Response

```
HTTP/1.1 200 OK
x-amzn-RequestId: 01234567-89ab-cdef-0123-456789abcdef
```

```
Content-Type: text/xml
Content-Length: 437
Date: Fri, 16 Oct 2020 20:12:47 GMT

<GetCallerIdentityResponse xmlns="https://sts.amazonaws.com/doc/2011-06-15/">
  <GetCallerIdentityResult>
    <Arn>arn:aws:sts::123456789012:federated-user/my-federated-user-name</Arn>
    <UserId>123456789012:sampleFederatedUser</UserId>
    <Account>123456789012</Account>
  </GetCallerIdentityResult>
  <ResponseMetadata>
    <RequestId>01234567-89ab-cdef-0123-456789abcdef</RequestId>
  </ResponseMetadata>
</GetCallerIdentityResponse>
```

References:
https://docs.aws.amazon.com/STS/latest/APIReference/API_Operations.html
https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_temp_request.html

IAM Policy Simulator

The IAM Policy Simulator is a testing and validation tool within AWS Identity and Access Management that enables administrators to evaluate policy effectiveness before deployment to production environments. This tool provides a safe sandbox where you can examine how various policy types, including identity-based policies, permissions boundaries, and resource-based policies, would affect access to AWS services and resources. The simulator evaluates whether specific API actions would be permitted or denied without actually executing those actions or impacting your live infrastructure. You can access the tool through a web console interface or programmatically via AWS CLI and API calls, making it adaptable to different operational workflows. One particularly valuable feature is the ability to test draft policies that haven't been attached to any IAM entities yet, allowing for iterative refinement before committing changes to your security configuration.

Key Features and Testing Capabilities

The Policy Simulator delivers comprehensive functionality for policy validation and troubleshooting:

- Flexible policy evaluation: Test individual policies or evaluate multiple policies simultaneously to understand their combined effect on permissions
- Contextual testing: Incorporate condition elements such as source IP addresses, request timestamps, or MFA status to simulate realistic access scenarios

- Boundary testing: Examine how permissions boundaries constrain the maximum available permissions for IAM entities
- Resource-specific simulations: Test policies against particular AWS resources like S3 buckets, SQS queues, SNS topics, and Glacier vaults to verify resource-level access controls
- Organizational policy impact: For accounts within AWS Organizations, assess how service control policies influence identity-based policy effectiveness
- Infrastructure scenario modeling: Configure specific deployment scenarios such as VPC-based EC2 environments or storage configurations to match your architecture

Important Limitations and Best Practices

While the IAM Policy Simulator is highly effective for policy testing, understanding its constraints ensures appropriate usage. The tool cannot evaluate service control policies containing conditional logic, nor does it support resource control policies or cross-account access patterns for roles and users. Additionally, resource-based policy simulation is unavailable for IAM roles. The simulator operates in isolation, meaning it may not capture every nuance of how policies interact in your actual production environment with all its variables and dependencies. Any modifications made within the simulator remain confined to the testing session and never alter your actual account policies. AWS recommends using the simulator as a first-line validation tool, then confirming policy behavior in your live environment before broad deployment. This approach minimizes the risk of overly permissive or restrictive policies while helping you maintain least-privilege access principles across your organization.

Reference:
https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_testing-policies.html

Leveraging IAM Access Analyzer to grant least privilege of access

Implementing the least privilege of access is certainly easier said than done. It's challenging to accomplish, as it's not set-and-forget but rather an ongoing process. This process demands someone with a thorough understanding of the services involved, the critical permissions for each service, and a commitment to continuous monitoring and updating. As the workload and permission requirements for a user change over time, it's essential to adapt and maintain the least privilege of access accordingly.

IAM Access Analyzer is a tool that you can use as a guide towards enforcing the least privilege of access.

Three key features of the IAM Access Analyzer:

1. **Policy generation based on access patterns**: IAM Access Analyzer can generate IAM policies based on the historical access patterns of an IAM user or role. This makes it easier to create policies that adhere to the principle of least privilege access.
2. **Continuous monitoring**: IAM Access Analyzer continuously monitors for new or updated resource permissions to help you identify permissions that grant public and cross-account access. For example, if an Amazon S3 bucket policy were to change, IAM Access Analyzer would alert you that the bucket is accessible by users from outside the account.
3. **Last-accessed information**: IAM Access Analyzer provides data about when AWS services were last used, which helps you identify opportunities to tighten your permissions. With this information, you can compare the permissions that have been granted with when those permissions were last accessed to remove unused access and further refine your permissions.

References:
https://docs.aws.amazon.com/IAM/latest/UserGuide/what-is-access-analyzer.html
https://aws.amazon.com/iam/features/
https://aws.amazon.com/blogs/security/use-iam-access-analyzer-to-generate-iam-policies-based-on-access-activity-found-in-your-organization-trail/

Implementing Role-Based Access Control (RBAC) and Attribute-Based Access Control (ABAC)

RBAC is one of the most commonly used access control paradigms. In this model, permissions are associated with roles, and roles are assigned to users. A role outlines the set of permissions that should be granted to any user who assumes that role. In AWS, RBAC is implemented through IAM roles, where specific permissions are attached to a role, and that role is then assigned to users or services.

When to use RBAC?

Use RBAC when you have clear job functions or tasks that map directly to roles. For example, in a company where there are a limited number of roles: Customer Support, Inventory Manager, and DevOps. Each role has a clear, well-defined set of responsibilities.

- **Customer Support -** needs read-only access to customer data and order histories.
- **Inventory Manager -** needs read-write access to the inventory database but no access to customer data.
- **DevOps** - needs broad access to AWS services for management and maintenance tasks. In this relatively straightforward environment, RBAC would be sufficient and easier to manage.

In this relatively straightforward environment, RBAC would be sufficient and easier to manage. One significant downside of using the traditional RBAC model in AWS is the manual intervention required when new resources are introduced. For example, if your company launches a new S3 bucket to store customer data, you'll need to manually update the IAM roles for groups like the data analytics team or customer support staff who need access to this new bucket. This can inadvertently lead to violations of the principle of least privilege. When hurriedly updating policies to include new resources, there's a risk of granting broader access than necessary.

ABAC is more flexible and granular compared to RBAC. In ABAC, permissions are granted or denied based on attributes associated with the user, the resource to be accessed, and other contextual information. In AWS, these attributes are often represented through resource tags.

Let's dive into the following scenario to understand how ABAC is implemented in AWS.

A large hospital system has multiple departments like Radiology, Cardiology, and General Medicine. Doctors from different departments need access to patient records. However, they should only have access to the records of patients who are actually under their care or consultation, and not all patients in the hospital system.

- **Attributes**: Every doctor and patient record can have multiple attributes, such as `["Department": "Cardiology"], ["Patient_Type": "Outpatient"]`.
- **Policy**: Access is granted based on these dynamic attributes.

AWS Implementation

1. Tag patient records in an Amazon S3 bucket with attributes like `{"Department": "Radiology", "Patient_Type": "Inpatient"}`.
2. Tag IAM roles assigned to doctors with similar attributes like `{"Department": "Radiology"}`.
3. Create a policy that allows access based on these tags.

The policy for the doctors might look like:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::HospitalRecords/*",
      "Condition": {
        "StringEquals": {
          "s3:ExistingObjectTag/Department": "${aws:PrincipalTag/Department}"
        }
      }
    }
  ]
}
```

In this policy, doctors will only be able to access S3 objects (patient records, in this case) that have the same Department tag as the IAM role that is assumed by them.

Advantages of ABAC over RBAC:

- **Reduced Administrative Overhead**
  - Imagine your team continuously launching new features, each requiring its own set of AWS resources like Lambda functions, S3 buckets, and RDS databases. With RBAC, each new feature launch would require you to manually update IAM roles to grant the necessary permissions for these new resources. In contrast, using ABAC with attribute-based policies, you can tag these new resources as "Feature_X," for example. Then, you can create a policy that grants access to any resource tagged as such. As new resources for Feature_X are rolled out, they automatically inherit the right permissions without requiring manual policy updates.
- **Fine-Grained Permissions**
  - ABAC allows for much more nuanced access control, down to specific attributes of a resource or user. This is particularly useful in complex or highly secure environments.

- **Fewer Policies Needed**
  - Because ABAC can incorporate multiple attributes into a single policy, you often need fewer policies compared to RBAC. This not only simplifies management but also reduces the chance of errors that can occur when dealing with a large number of more simplistic policies, as is often the case with RBAC.

References:

https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction_attribute-based-access-control.html
https://docs.aws.amazon.com/prescriptive-guidance/latest/saas-multitenant-api-access-authorization/abac-rbac-examples.html

Centralized Fine-Grained Authorization with Amazon Verified Permissions

Amazon Verified Permissions is a managed, fine-grained authorization service for applications, enabling centralized policy-based control rather than scattering authorization logic across code. It uses the open-source Cedar policy language to define permissions in terms of principals, resources, actions, and optional attributes ("tags," roles, context, etc.).

Key features:

- AVP supports both role-based access control (RBAC) and attribute-based access control (ABAC) — you can grant access based on a user's role or on attributes (e.g., tags, resource attributes, session/context data).
- It decouples authorization from application logic: your app calls AVP APIs to evaluate whether an action is allowed instead of embedding access checks throughout your code.
- Policies can be centrally managed, versioned, and audited. AVP can log authorization decisions to AWS CloudTrail for compliance and traceability.
- AVP integrates with identity providers (for example via Amazon Cognito or other OIDC providers), letting you use existing authentication systems while offloading authorization to AVP.

References:
https://docs.aws.amazon.com/verifiedpermissions/latest/userguide/what-is-avp.html
https://docs.aws.amazon.com/verifiedpermissions/latest/userguide/security_iam_service-with-iam.html

Domain 5: Data Protection

Overview

The fifth exam domain of the AWS Certified Security Specialty test focuses on data protection of your AWS infrastructure. It has the second biggest percentage representing approximately 18% of the overall exam. You should have the skill to design and implement controls that provide confidentiality and integrity for data in transit; design and implement controls that provide confidentiality and integrity for data at rest; design and implement controls to manage the lifecycle of data at rest as wel as to design and implement controls to protect credentials, secrets, and cryptographic key materials. This domain will test your knowledge and skills on the following:

- Designing secure connectivity between AWS and on-premises networks (for example, by usingDirect Connect and VPN gateways)
- Designing mechanisms to require encryption when connecting to resources (for example, Amazon RDS, Amazon Redshift, CloudFront, Amazon S3, Amazon DynamoDB, load balancers, Amazon Elastic File System [Amazon EFS], Amazon API Gateway)
- Requiring TLS for AWS API calls (for example, with Amazon S3)
- Designing mechanisms to forward traffic over secure connections (for example, by using Systems Manager and EC2 Instance Connect)
- Designing cross-Region networking by using private VIFs and public VIFs
- Designing resource policies to restrict access to authorized users (for example, S3 bucket policies, DynamoDB policies)
- Designing mechanisms to prevent unauthorized public access (for example, S3 Block Public
- Access, prevention of public snapshots and public AMIs)
- Configuring services to activate encryption of data at rest (for example, Amazon S3, Amazon RDS, DynamoDB, Amazon Simple Queue Service [Amazon SQS], Amazon EBS, Amazon EFS)
- Designing mechanisms to protect data integrity by preventing modifications (for example, by using S3 Object Lock, KMS key policies, S3 Glacier Vault Lock, and AWS Backup Vault Lock)
- Designing encryption at rest by using AWS CloudHSM for relational databases (for example, Amazon RDS, RDS Custom, databases on EC2 instances)
- Choosing encryption techniques based on business requirements
- Designing S3 Lifecycle mechanisms to retain data for required retention periods (for example, S3 Object Lock, S3 Glacier Vault Lock, S3 Lifecycle policy)
- Designing automatic lifecycle management for AWS services and resources (for example, Amazon S3, EBS volume snapshots, RDS volume snapshots, AMIs, container images, CloudWatch log groups, Amazon Data Lifecycle Manager [Amazon DLM])
- Establishing schedules and retention for AWS Backup across AWS services
- Designing management and rotation of secrets for workloads (for example, database access credentials, API keys, IAM access keys, AWS KMS customer managed keys)
- Designing KMS key policies to limit key usage to authorized users
- Establishing mechanisms to import and remove customer-provided key material

AWS Key Management Service (AWS KMS)

AWS KMS Key

The KMS Key is the most basic resource in AWS KMS. A KMS key includes metadata, such as the key ID, creation date, description, and key state. The KMS key also contains the key material used to encrypt and decrypt data.

AWS KMS supports the following key types:

1. **Symmetric** - a 256-bit key that is used for encryption and decryption.
2. **Asymmetric** - an RSA key pair that is used for encryption and decryption or signing and verification (but not both), or an elliptic curve (ECC) key pair that is used for signing and verification.
3. **Multi-region keys** - symmetric or asymmetric keys that can be used across regions.
4. **Hash-Based Message Authentication Code (HMAC) KMS keys** - generate and verify hash-based message authentication codes

Features

- KMS is integrated with CloudTrail, which provides you the ability to audit who used which keys, on which resources, and when.
- Customer master keys (KMS keys) are used to control access to data encryption keys that encrypt and decrypt your data.
- You can choose to have KMS automatically rotate master keys created within KMS once per year without the need to re-encrypt data that has already been encrypted with your master key.
- To help ensure that your keys and your data are highly available, KMS stores multiple copies of encrypted versions of your keys in systems that are designed for 99.999999999% durability.

Three types of KMS Key:

1. **Customer-managed** - You have complete control over these KMS keys. You are in charge of key management, which includes key policy configuration, key material rotation, and key lifecycle management.
2. **AWS-managed** - These are KMS keys in your account that are created, managed, and used on your behalf by an AWS service. You cannot manage, rotate, and change the key policies of AWS-managed keys. You also cannot use them in cryptographic operations directly; the service that creates them uses them on your behalf.

3. **AWS-owned** - These are KMS keys that an AWS service creates, owns, and manages for use in multiple AWS accounts. You cannot view, use, track, or audit these KMS keys.

Type of KMS key	Can view metadata	Can manage KMS Key	Used only for your AWS account	Automatic rotation
Customer managed	Yes	Yes	Yes	Optional. Every **365** days
AWS-managed	Yes	No	Yes	Required. Every **365** days
AWS-owned	No	No	No	Varies

KMS generates key material for all keys that it creates by default. Default key materials cannot be extracted, exported, viewed, or managed. A key material by itself cannot be deleted; if you want to get rid of it you must delete the KMS Key it's associated with. You may also import your own key material into a customer-managed KMS key. When you import a key material, you can specify a date upon which the key material expires. KMS keys generated under an expired key material become unusable. As opposed to You can also delete key material on demand as opposed to the scheduled deletion.

References:
https://docs.aws.amazon.com/kms/latest/developerguide/concepts.html#master_keys
https://tutorialsdojo.com/aws-key-management-service-aws-kms/

Envelope Encryption

A KMS key is capable of encrypting data up to 4 KB in size, making it not suitable for the encryption of large data sizes. To encrypt data more than 4KB, as you most likely will, you'll need to use a method called envelope encryption.

Envelope encryption is not specific to KMS. But when we talk about envelope encryption in the context of KMS, it refers to the process of using a KMS key to generate a data key. You then use that data key to encrypt a file or create another data key; the depth of encryption is up to you. Multiple layers of security can be achieved by generating a secondary data key under the top-level data key, then encrypting that key under another key, and so on. The catch is there should always be a master key that remains in plaintext so you can eventually decrypt all data keys. In KMS, the master key is the KMS key.

As a security best practice, always see to it that the plaintext data key is deleted from memory after use.

KMS keys are securely stored in hardware security modules (HSM) and don't leave unencrypted. In AWS KMS, you share tenancy access to HSM with other customers. If you wish to procure an exclusive, single-tenant access to HSM, use AWS CloudHSM instead.

KMS key alias

When you perform cryptographic operations programmatically, you inform KMS of which key to use by passing a key id. A key id is a long string of characters that uniquely identifies a KMS key. Unless you have a knack for memorizing things, we generally do not want to remember key IDs. Also, as the keys that you manage increase, you'll find it more challenging to keep track of which keys are used for what purpose. Enter alias. An *alias* is an optional display name for a KMS key. Each KMS key can have multiple aliases, but an alias can only point to a single key. The alias name must be unique in your AWS account and region.

Consider the Key ID below. Isn't that too much for us to remember? Would it be great if we can have a nickname for this ID that we can reference to?

```
Key ID = 2e3ef989-a491-417d-98d9-3682156cae1f
```

Luckily, we can create an alias for this ID by using the *CreateAlias* command.

```
aws kms create-alias --alias-name alias/dojo-key --target-key-id
2e3ef989-a491-417d-98d9-3682156cae1f
```

Now, instead of specifying a KMS key by its key id, we can pass its alias to the *--key-id* parameter. By doing this, managing different keys would be easier as you could easily determine the owner of the keys or the purpose it serves by creating a descriptive alias.

```
aws kms encrypt --key-id alias/dojo-key --plaintext fileb://ExamplePlaintextFile
```

Note: Not all KMS API operations support alias. For the list of the supported KMS API operations, refer to the KMS API documentation

References:
https://aws.amazon.com/kms/faqs/
https://docs.aws.amazon.com/kms/latest/developerguide/rotate-keys.html#rotate-keys-how-it-works
https://docs.aws.amazon.com/kms/latest/developerguide/kms-alias.html

AWS KMS API

Users and developers who manage security can interact with AWS KMS programmatically via the CLI or SDK. These utilize the AWS KMS API for all of the transactions. You also do not use your standard username and password when interacting with the KMS API, so be sure to enter your access keys and secret access keys instead. Once you have configured your AWS profile on your local machine, you can start executing API calls. If you encounter any errors during API calls, check if your IAM User has been granted the necessary permissions to perform that action.

List of Commonly Used AWS KMS APIs

Below are the important KMS API commands that you should know when going to the exam:

- ***Encrypt***
- ***Decrypt***
- ***GenerateDataKey***
- ***GenerateDataKeyWithoutPlaintext***

Encrypt

To encrypt plaintext into ciphertext using your KMS key, run the *Encrypt* command. Enter the key ID that you'd like to use and the data you'd like to encrypt. After running it, the API will return your ciphertext in blob format, the key ID used and the encryption algorithm used.

```
aws kms encrypt --key-id alias/dojo-key --plaintext fileb://ExamplePlaintextFile
```

Decrypt

To decrypt your ciphertext, run the *Decrypt* command and enter your ciphertext blob. If you used an asymmetric KMS key to encrypt this text, then you need to specify a key ID parameter. You should also specify the encryption algorithm and encryption context if the defaults were not used. After running it, the API will return your plaintext, key ID used for decryption, and the encryption algorithm used.

```
aws kms decrypt \
  --key-id alias/dojo-key \
  --ciphertext-blob fileb://ExampleCiphertextFile
```

GenerateDataKey

We talked about envelope encryption and how it works in the previous article. And I have mentioned about using a data key to encrypt a file instead of encrypting directly using the KMS key. AWS KMS simplifies the process of envelope encryption for us by providing an API that will generate a data key that we can use to encrypt a file or another encryption key.

You can specify your data key's length by using either the *--key-spec* or *--number-of-bytes* parameter.

In this example, we used the `AES_256` parameter to generate a 256-bit symmetric key.

```
aws kms generate-data-key --key-id alias/dojo-key --key-spec AES_256
```

The resulting output returns the encrypted data key (`CiphertextBlob`), `Plaintext` data key, and the `KeyId` used to generate the data key.

```
{
    "CiphertextBlob":
"AQIBAHgMxXGERpLXTIIM54OPUp/dXeRYW2ALjX6EVz3skLXeBwG6AEIFFTyHrw6EXSuZxf7gAAAAfjB8Bgkqhk
iG9w0BBwagbzBtAgEAMGgGCSqGSIb3DQEHATAeBglghkgBZQMEAS4wEQQMqsMiCfXkoxHsHbxfAgEQgDunMIdAh
gNqLaI6QtKnw5UrqQhrPezpLSE0fvkUD4yVpkJp1594C8DV6wBohptgrmSVA8B16xU9VK+cWA==",

    "Plaintext": "s7hbvvuIm0Dg2ZMNpXPWqZq5cKjv1bPj23HYA4d/syM=",

    "KeyId":
"arn:aws:kms:ap-southeast-1:123456789123:key/c7181401-5441-447e-a019-4dd4b5f39aba"
}
```

Use the plaintext data key when encrypting and decrypting data. After an encryption or decryption operation, you should delete the plaintext data key and only keep the encrypted data key. When the time comes where you have to decrypt data, you must call the *Decrypt* command to decrypt the encrypted data key first. The output will return the plaintext data key that you can use to decrypt the data finally.

GenerateDataKeyWithoutPlaintext

This KMS API command is almost similar to the previous API command that we have discussed. Their aim is to generate a data key that we can use for envelope encryption. Their only difference is that *GenerateDataKeyWithoutPlaintext*, as its name implies, will only generate a data key that returns the encrypted data key.

```
aws kms generate-data-key-without-plaintext \
  --key-id alias/dojo-key \
  --key-spec AES_256
```

The output returns the encrypted data key and the key ID used to generate the data key.

```
{
    "CiphertextBlob":
"AQIDAHgMxXGERpLXTIIM54OPUp/dXeRYW2ALjX6EVz3skLXeBwGsYnYEmxiS9joF09WJDLioAAAAfjB8Bgkqhk
iG9w0BBwagbzBtAgEAMGgGCSqGSIb3DQEHATAeBglghkgBZQMEAS4wEQQM9g2SrPclDG0c52LqAgEQgDt6SKU3W
p4R8+qDvNBjq6IWbrXmxUfhnLOaTqAZBeYZ8UtaHf2kWoPOVA4jBjmsGaOtakTQvsrMLKAbNw==",

    "KeyId":
"arn:aws:kms:ap-southeast-1:123456789123:key/c7181401-5441-447e-a019-4dd4b5f39ebe"
}
```

Note that we can't use this encrypted data key for data encryption. We must first decrypt the data key by calling the *Decrypt* command. The output will return the plaintext version of the data key. Only then can we proceed by encrypting data with this plaintext data key.

Why use GenerateDataKeyWithoutPlaintext?

It is better to use *GenerateDataKeyWithoutPlaintext* over *GenerateDataKey* when dealing with a distributed system of containers that store encrypted data along with data keys in an encrypted form. Imagine one of your services creates and saves data to each of your containers. That service could simply call the *Decrypt* command to decrypt the encrypted data key on a container, use the resulting plaintext data key to decrypt the data, and delete the plaintext data key. In this case, the service never sees the plaintext data key.

References:
https://docs.aws.amazon.com/kms/latest/developerguide/programming-keys.htmlhttps://docs.aws.amazon.com/kms/latest/APIReference/API_Operations.htmlhttps://docs.aws.amazon.com/cli/latest/reference/kms/

Delegating Permissions and KMS Actions in AWS KMS

In AWS KMS, you can control access to your master keys by using ***key policies*** and ***grants***. Key policies are JSON-based documents that describe who is allowed to do what KMS operations (e.g., Encrypt, Decrypt) to your master keys.

Key Policy

Key policies help protect your KMS keys by defining specific requirements that must be fulfilled before an action is permitted. The policy structure is similar to IAM policies, and it also uses JSON formatting. Here is a basic example of a key policy:

```
{
  "Sid": "Allows Encrypted Multipart Upload",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::123456789123:user/ExampleUser"},
  "Action": ["kms:GenerateDataKey*", "kms:Decrypt"],
  "Resource": "*",
  "Condition": {"Bool": {"kms:GrantIsForAWSResource": true}}
}
```

To create your key policy, you must first indicate the policy version that you will be using. Then in the statement body, you must include the following parameters:

- Effect - indicates whether the policy will allow or deny actions
- Principal - the identity to which the policy will grant or deny permissions to
- Action - the permissions that you want to grant/deny to the principal
- Resource - the list of objects that your policy will be applied to

You can also include the following optional parameters in your statement body:

- Sid - a unique identifier for your policy
- Conditions - conditions that need to be met before your policy takes effect

When you create a KMS key with the AWS Management Console, you can choose the IAM users, IAM roles, and AWS accounts that should be given access to the KMS key, and these will be added to a default key policy that the console creates for you. On the other hand, when you create a KMS key via the *CreateKey* command and you do not provide a key policy in the parameters, AWS automatically creates and assigns a default key policy for your KMS key.

Either way, you should see the first statement of your default key policy similar to this:

```
{
  "Sid": "Enable IAM User Permissions",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::123456789123:root"},
  "Action": "kms:*",
  "Resource": "*"
}
```

Take a look at the value of the Principal. At first glance, one could naturally assume that the policy is granting access to the *root* user. However, this is not the case. Keep in mind that the "root" does **NOT** pertain to the root user of the account. What it actually implies is that it allows IAM entities (e.g., IAM Users, IAM Roles) in *123456789123* AWS Account to gain full access to the KMS key. Simply put, the first statement is allowing IAM to further manage the permissions to your KMS key.

kms:ViaService Condition Key

In situations where you need to limit the use of your KMS key for requests from specific AWS services, you can add the *kms:ViaService* condition key to your key policy statement as follows:

```
{
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::123456789123:user/Carlo"},
  "Action": [
      "kms:Encrypt",
      "kms:Decrypt"
 ],
  "Resource": "*",
  "Condition":{
      "StringEquals": {
        "kms:ViaService":
         ["ec2.ap-southeast-1.amazonaws.com",
          "ses.ap-southeast-1.amazonaws.com"
        ]
      }
   }
}
```

The preceding policy means that the KMS key can be used for encryption and decryption if the requester is IAM User Carlo, and the request comes from either Amazon EC2 and Amazon SES in the ap-southeast-1 region. So even if you're using the correct principal to use a certain KMS key, you still need to ensure that the service and region of the request match the values defined in the *kms:ViaService* condition key. Otherwise, the request will be denied.

Grants

There can only be one key policy for each KMS key, which means that if you manage hundreds of KMS keys, you're also responsible for managing hundreds of key policies. As one would expect, the complexity level of editing each key policy increases with the number of master keys you're administering.

Fortunately, we can associate KMS permissions to a KMS key without even touching the key policy associated with it. We can programmatically delegate permissions to a principal by using *grants*. To create a grant, we call the ***CreateGrant*** operation.

```
aws kms create-grant --key-id [KEY_ID or KEY ARN] \
  --grant-principal arn:aws:iam::123456789123:user/josephine \
  --retiring-principal arn:aws:iam::123456789123:user/rizal \
  --operations Decrypt
```

The *CreateGrant* operation does not support the use of aliases, so you must pass the key ID or the key ARN instead. The `--grant-principal` parameter defines the grantee or the person to whom you are delegating access to. The `--retiring-principal` parameter designates the principal that can retire or revoke the access to the KMS key. The `--operations` defines the operations the grantee is allowed to do.

The *CreateGrant* operation returns a grant token and grant ID. You can retire a grant anytime when you're done using it. Simply call the *RetireGrant* command with the grant token that you received when you created the grant. Note that a grant can only allow access, but not deny.

```
aws kms retire-grant --grant-token [GRANT TOKEN]
```

Security Specialty Exam Notes:

The "***root***" in the `"Principal": {"AWS": "arn:aws:iam::123456789123:root"}` statement of a key policy pertains to any IAM user in the specified account and **NOT** the root user.

You can use the `kms:ViaService` permission if you want to limit KMS key access against requests coming from specific AWS resources.

Always think of *grants* when you are required to programmatically create and revoke KMS key access.

References:
https://docs.aws.amazon.com/kms/latest/developerguide/determining-access-key-policy.html
https://docs.aws.amazon.com/kms/latest/developerguide/policy-conditions.html
https://docs.aws.amazon.com/kms/latest/developerguide/grants.html
https://docs.aws.amazon.com/kms/latest/developerguide/policy-conditions.html

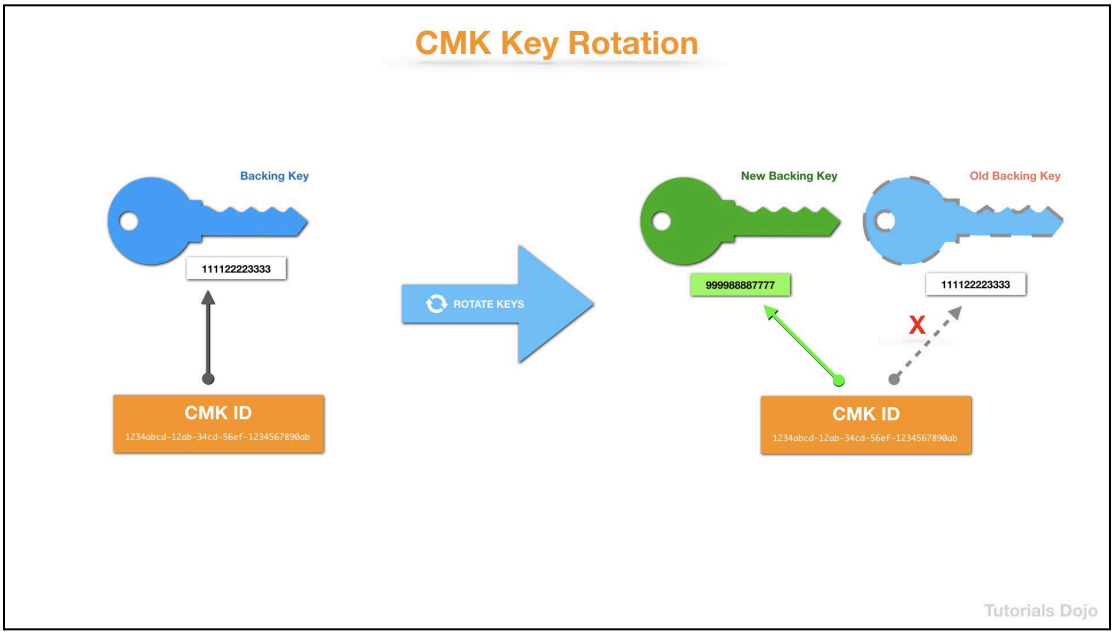
AWS KMS Key Rotation

An AWS Managed Key is automatically rotated every 365 days. This feature is optional for Customer Managed Key. However, automatic key rotation is not supported by the following:

- Asymmetric KMS keys where public and private key pair are used to encrypt/decrypt or sign/verify operations
- KMS keys in custom key stores where the key material is stored on AWS CloudHSM cluster
- KMS keys that have imported key material which is supported only for symmetric KMS keys.

Automatic Key Rotation

The diagram below illustrates the process of automatic KMS Key Rotation.



When automatic key rotation is enabled, the properties of the KMS key (e.g., key id, key ARN, aliases, region, etc.) do not change when the key is rotated. Only the *backing key,* the cryptographic material that is used in encryption operations, is altered in the event of key rotation. AWS saves the old key versions so that data encrypted under a previous backing key can still be decrypted. As a result, you do not have to update the aliases or KMS key IDs that were previously included in your scripts or application code.

Manual Key Rotation

If you need a custom rotation period, you can manually replace your current KMS key with a newly created one. As a result, when the time comes to rotate keys, you'd also need to update the KMS key ID references on your application, which can be a lot of work. Aliases simplify this process. Instead of referencing KMS keys by its ID or ARN in your code, use an alias. This way, all you have to do is update the KMS key that the alias is pointing to using the `UpdateAlias` API.

AWS Managed KMS key Rotation

You can't manage the key rotation for AWS Managed Keys. The automatic key rotation is handled by AWS KMS and is automatically rotated every three years.

References:
https://docs.aws.amazon.com/kms/latest/developerguide/concepts.html
https://docs.aws.amazon.com/kms/latest/developerguide/rotate-keys.html
https://docs.aws.amazon.com/kms/latest/developerguide/rotate-keys.html#rotate-keys-manually

Using Encryption Context for Additional Authenticated Data (AAD) to Support Authenticated Encryption

Additional Authenticated Data (ADD) is a string that you pass as a part of a KMS ***Encrypt*** and ***Decrypt*** request for the purpose of preventing data tampering. On AWS KMS, to implement AAD, we pass the encryption context parameter when initiating an *Encrypt* API call. The encryption context is a set of name-value pairs that cryptographically bind to the encryption's resulting ciphertext. You can think of the encryption context as your digital signature for your requests. Make sure that you create them as unique as you can. By doing this, we can prevent confused deputy attacks.

```
aws kms encrypt \
  --plaintext fileb://ExamplePlaintext.txt \
  --key-id alias/dojo-key \
  --encryption-context password=mysecurepassword123,Department=IT
```

To decrypt the ciphertext data, you must pass the exact name-value pairs that you've used for encryption. Failure to do so results in an ***InvalidCiphertextException*** error.

References:
https://docs.aws.amazon.com/kms/latest/developerguide/concepts.html#encrypt_context
https://docs.aws.amazon.com/crypto/latest/userguide/cryptography-concepts.html#term-aad

Storing Encryption Keys Using AWS CloudHSM

Like any other "as-a-service" Cloud Services, AWS CloudHSM is basically an HSM that lives in the Cloud or an HSM-as-a-Service model. Unlike KMS, you **do not share** tenancy with other AWS customers. In addition, CloudHSM lets you run and manage a dedicated FIPS 140-2 Level 3 validated. You have full control over your encryption keys, including their lifecycle management. To interact with your HSM, you need to use the CloudHSM Client — a software package that lets you communicate directly to CloudHSM over a secured, mutually authenticated channel. There are also available CloudHSM API actions that you can use, but they're limited in terms of what they can do.

Advantages Of Using AWS CloudHSM

- **Highly Available**
  - Instead of using HSM in a single physical location, AWS CloudHSM manages HSMs in a cluster across availability zones to achieve high availability. You can create a cluster of HSM in one subnet for each availability zone per region. AWS CloudHSM keeps the individual HSM in the cluster in sync, so if you change a configuration of an HSM inside AZ- 1, the change will be reflected on the HSM inside AZ -2.
- **Only Accessible By through a VPC**
  - AWS CloudHSM only allows creating a CloudHSM cluster in a VPC to isolate your AWS CloudHSM from other customers. It is **important to note** that you **can only** operate the CloudHSM Client on an EC2 instance inside a VPC. You can securely connect to the EC2 instance from on-premises via VPN connection, or Direct Connect.
- **Tamper-evident Control**
  - HSMs are purposely built to be tamper-resistant. If you use AWS CloudHSM, you are just essentially using HSMs in a remote data center — the Cloud, instead of on-premises. The nature and physical properties of an HSM do not change.
- **Has access logging available**
  - You can obtain the AWS CloudHSM Client's logs on the EC2 instance where you run the CloudHSM Client. The path of the log files varies depending on the operating system used. AWS CloudHSM API calls are captured by CloudTrail.

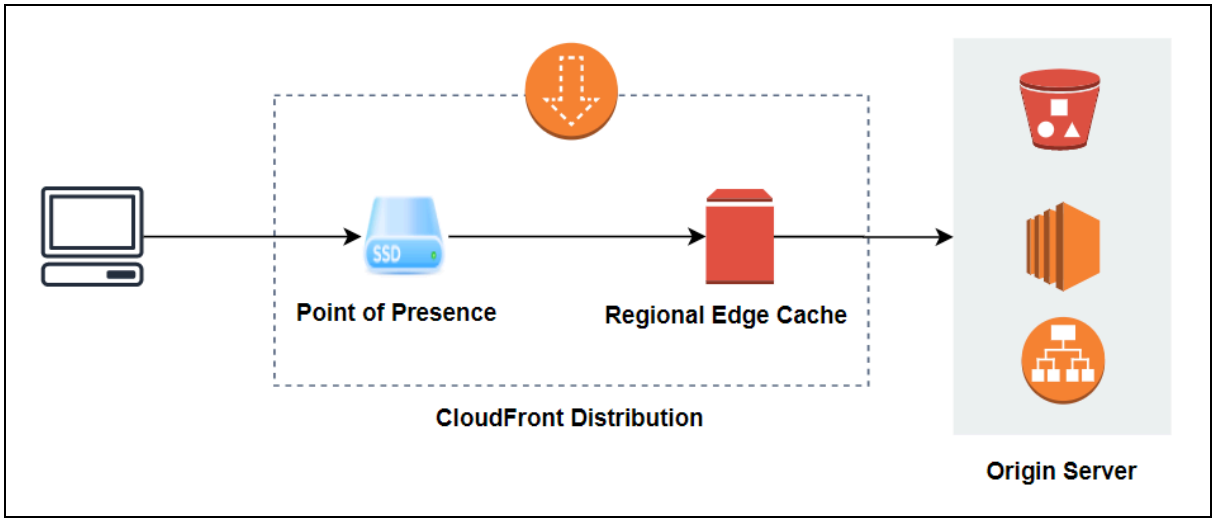
Securing Data In AWS CloudFront

CloudFront accelerates content delivery by serving them directly from distributed servers called *Points of Presence (POPs)*. In AWS, we refer to POPs as ***edge locations***. These edge locations are stationed in different places around the globe — providing low-latency access to the end-users.

We have briefly discussed how CloudFront caches data in edge locations back in Domain 3 under the "*Adding HTTP Security Headers Using Lambda@Edge and CloudFront*" section. What we know so far is that whenever a user retrieves files from a CloudFront distribution, CloudFront will check first if that file exists in its edge location. If not, it will get the data back from the origin server, cache that data on the edge location, and return it to the user. The same file will be fetched from the edge location the next time it is requested.

That explanation is kind of incomplete. As a matter of fact, there is another cache layer in addition to the CloudFront POPs. That extra cache layer is called *Regional Edge Cache* (REC). Files stored on POPs eventually get less popular and need to be removed to make space for popular files. This is where REC comes into play. It acts as a fallback for files that become less popular. With this setup, when the less popular files are requested again, the data retrieval latency will remain relatively small since there is no need for CloudFront to request files from the origin server.

AWS needs two "cache layers" because of the technology it uses. As you see on the diagram, POPs use Solid State Drive (SSD) while RECs use Elastic Block Storage. SSDs have faster read and write throughputs than EBS since EBS is a network-attached storage. This is why SSDs are the preferred choice for serving files directly to users.

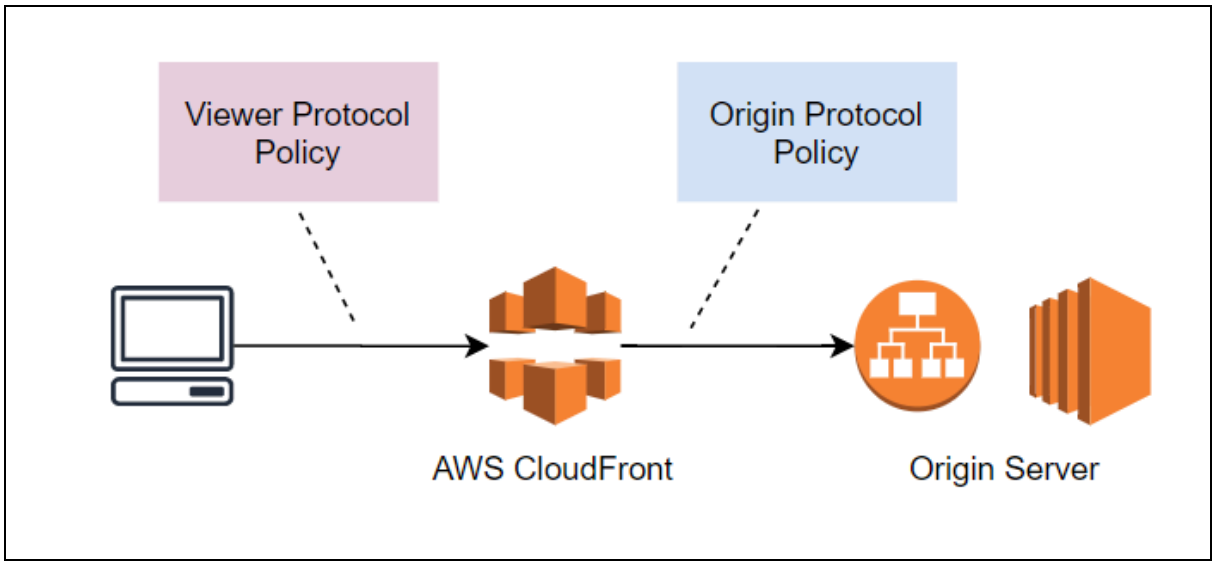


How to secure data at rest on POPs and RECs?

There is no need to configure CloudFront.The SSD and EBS that AWS uses for POP and REC are encrypted. It means that your data at rest is encrypted as well.

How about Encryption in transit?

There are two methods in enforcing HTTPS connections on CloudFront: via **Viewer Protocol Policy** or **Origin Protocol Policy.**



Viewer Protocol Policy

- Describes the connection protocol and policy **between the client (viewer) and CloudFront**.
- It supports
  - HTTP and HTTPS
  - Redirects HTTP to HTTPS
  - HTTPS Only.

Origin Protocol Policy

- describes the connection protocol policy **between CloudFront and the Origin Server**.
- It supports
  - HTTP Only
  - HTTPS Only
  - Match Viewer
- Selecting **`Match viewer`** will prompt CloudFront to to use the same protocol as the viewer. This can be useful when you have a mix of HTTP and HTTPS content, and you want to avoid protocol mismatches that could affect the delivery of your content.

- Origin Protocol Policy **does not** apply at S3 website endpoints. This is because S3 website endpoints only support HTTP connections. In order to enforce HTTPS communication between CloudFront and S3, you must change Viewer Protocol Policy setting to "HTTP and HTTPS" or "Redirect HTTP to HTTPS."

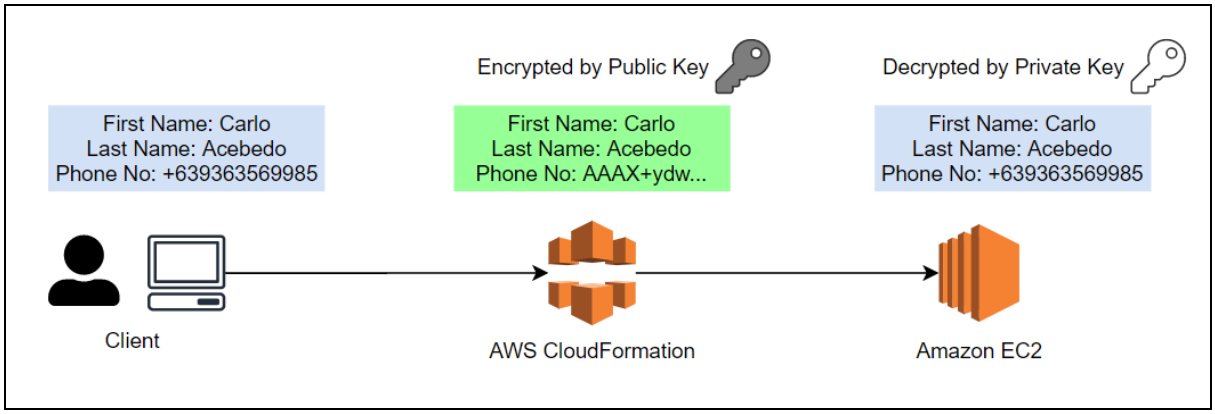
Custom Domain Name

CloudFront uses its own SSL certificate to provide an HTTPS CloudFront domain name by default. If you wish to use a custom domain name, you must use your own SSL certificate. Make sure that the domain listed on the certificate matches the alternate domain name that you want to use. You can request one from the ACM in the N. Virginia Region or use an imported certificate stored on IAM.

Field Encryption

Field Encryption refers to the process of encrypting data on certain data fields. Field Encryption adds an extra protection layer specifically for confidential data like security number, card number, phone number, or any personally identifiable information.

You can configure CloudFront to encrypt specific data as soon as it reaches the edge location. The Field Encryption uses public-key cryptography, which needs two separate keys: *public* and *private keys*. You must generate your own RSA key pair. Upload the public key on CloudFront and specify the data field that you want to encrypt.



Geo Restriction

CloudFront also supports Geo Restriction in case you want to deliver your content to specific locations only. You can either blacklist or whitelist countries on Geo Restriction settings.

Security Specialty Exam Notes:

CloudFront uses encrypted SSD and EBS volumes for POP and REC respectively.

If you need a custom domain name, use a custom SSL certificate. Ensure that the domain name listed on the certificate matches the custom domain name that you will use.

Enable HTTPS on Viewer Protocol Policy and Origin Protocol Policy if you want end-to-end encryption.

To use Field Encryption, create an RSA key pair and upload the public key to CloudFront. Specify the data field that you want to encrypt.

Use Geo Restriction to restrict access to your content for users in specific countries.

References:
https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/SecurityAndPrivateContent.html
https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/data-protection-summary.html
https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/data-protection-summary.html#data-protection-sum

Using Elastic Load Balancer For Encrypting Traffic

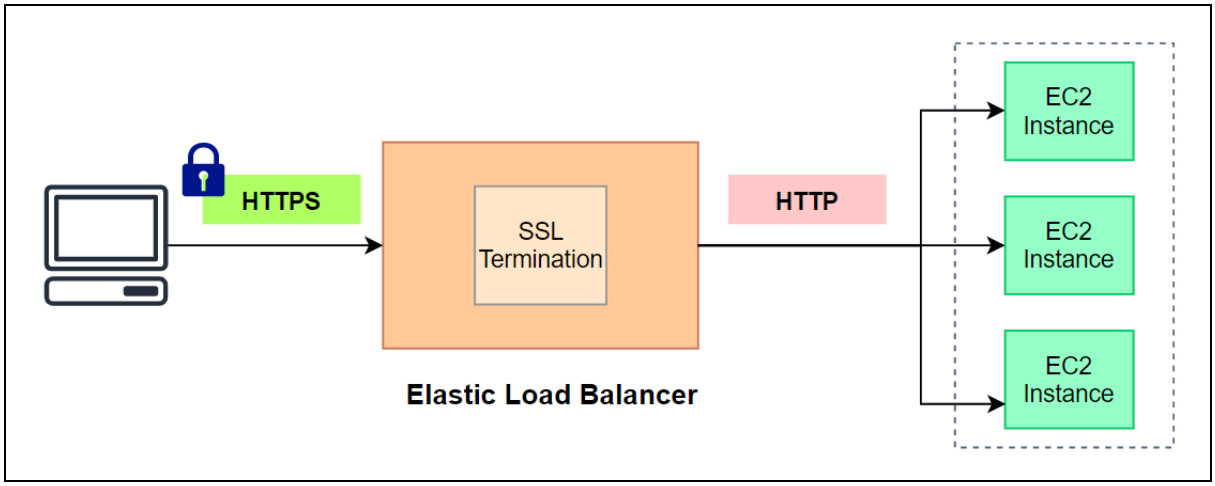
There are three ways to use the Elastic Load Balancer to protect the data coming from a source to its destination:

SSL Termination / SSL Offloading

SSL Termination or SSL offloading refers to the process of offloading the encryption/decryption of data sent via SSL traffic from a web server to a dedicated server. In AWS, instead of burdening your backend server to process SSL traffic, we can move that responsibility to an Elastic load balancer. By doing so, you'd reduce the workload of your EC2 instances and save on compute resources.

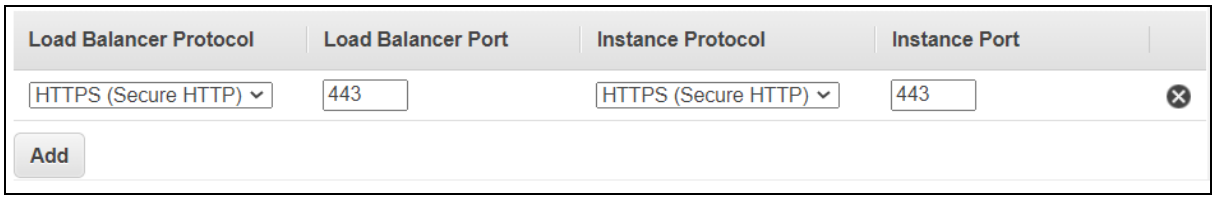
To begin, make sure you install an SSL server certification on your load balancer. Then, configure your ELB to listen to a secure protocol (HTTPS/SSL). Client's request will now be decrypted at the load balancer. The ELB

will send back the unencrypted data across the EC2 instances on the port specified on the ELB's listener configuration.

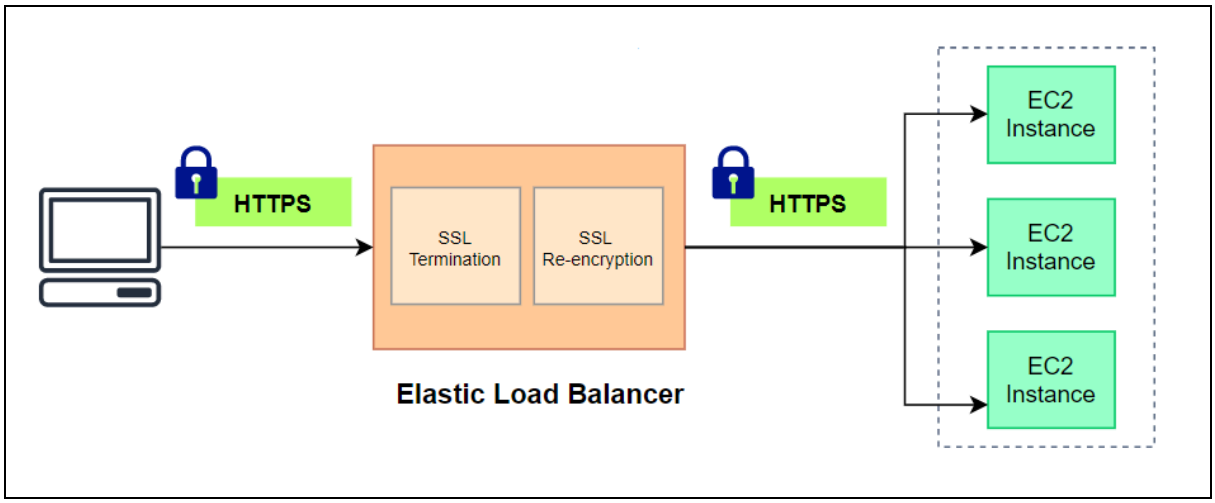


SSL Termination and Re-encryption

In cases where you are required to send all data in transit via HTTPS, you can optionally configure your ELB to terminate requests and re-encrypt them before sending them to your back-end servers. To employ this method, you have to install an SSL certificate first to your ELB. The ELB uses the SSL certificate to authenticate both the client and back-end server's requests. Modify your load balancer protocol to listen to HTTPS traffic on port 443. Likewise, change your instance protocol to listen to HTTPS traffic.

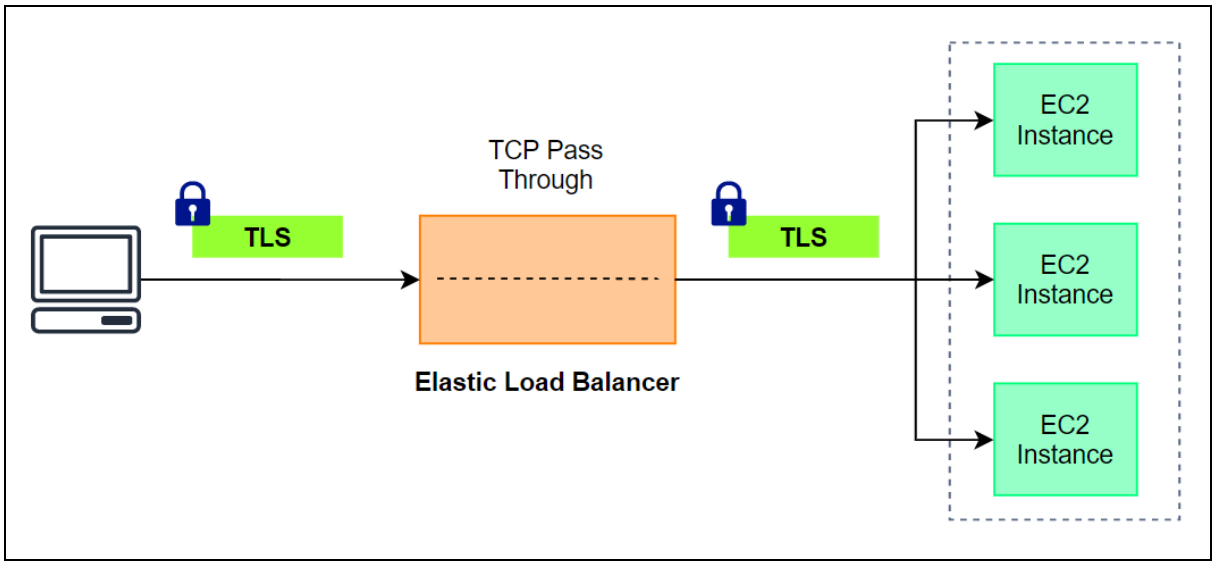


By doing this, the communication from your ELB to your back-end server is now automatically encrypted.



End-to-End Encryption

End-to-End Encryption is the process of securing data traffic from both ends of a communication channel, preventing any third-parties (e.g. load balancers) from reading the data unencrypted. To do this in AWS, we simply have to set both the load balancer and instance protocol to TCP. We will not install any TLS/SSL certificate on the load balancer since the requests are not terminated nor decrypted at the load balancer. The ELB lets the requests pass through as it is. To enable encrypted communication between the back-end server and the client, we have to install the TLS/SSL certificate on the back-end server.



Security Specialty Exam Notes:

Note that the ALB can only listen to HTTP and HTTPS traffic. If you're using a custom security protocol that requires a different port, you should consider using the Network Load Balancer.

References:
https://aws.amazon.com/blogs/aws/elastic-load-balancer-ssl-support-options/
https://aws.amazon.com/blogs/compute/maintaining-transport-layer-security-all-the-way-to-your-container-using-the-network-load-balancer-with-amazon-ecs/
https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/elb-listener-config.html
https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/elb-add-or-delete-listeners.html

Recovering the Data of an Encrypted Amazon EBS Volume if You Lose the KMS key

Consider this scenario: *A colleague had inadvertently scheduled a deletion for a KMS key used to encrypt a production EBS volume. What can you do to recover the data from the encrypted volume?*

In order to solve this problem, we need to know some basic concepts first about EBS encryption and how it works.

EBS Encryption

- Data stored at rest on an encrypted volume, disk I/O, and snapshots created from it are all encrypted.
- Provides encryption for data in-transit between EC2 and the EBS volume attached to it since encryption occurs on the servers that host EC2 instances.
- The following types of data are encrypted:
    - Data at rest inside the volume
    - All data moving between the volume and the instance
    - All snapshots created from the volume
    - All volumes created from those snapshots
- Uses AWS Key Management Service (AWS KMS) master keys when creating encrypted volumes and any snapshots created from your encrypted volumes.
- Volumes restored from encrypted snapshots are automatically encrypted.

How does EBS encryption work?

Amazon Elastic Block Store (EBS) uses a data key derived from a KMS key to encrypt data as it gets stored on the disk. The good thing about this is that you don't have to worry about generating and managing those data keys since EBS already supervises these operations behind the scenes.

EBS stores the data key within its volume metadata. When you detach an EBS volume from an instance, the data key remains in encrypted form. Now, when you attach an encrypted volume to an EC2 instance, Amazon EC2 sends a **`CreateGrant`** request to KMS so that it can use the specified KMS key to decrypt the data key back to its plaintext form. This plaintext data key is then stored in the instance's hypervisor memory for as long as the EBS volume is attached to the instance. Amazon EC2 utilizes this data key to encrypt disk I/O to the volume. Keep in mind, once the KMS key is deleted, it cannot be recovered. Additionally, if you try to launch an instance with an encrypted volume and you don't have access to the KMS key, the instance will stay in the PENDING state and eventually be terminated.

So, how do we retrieve data from an encrypted EBS volume that we can no longer decrypt?

First, determine if the EBS volume is still attached to an instance. If it is, what you can do is to create a new EBS volume, and mount it to the same instance where your encrypted EBS is attached. Then, transfer the data from the old EBS to the new one.

This kind of situation highlights the importance of detection and alerting systems and why they should not be taken for granted. Imagine if you learned about the deleted KMS key months after. In the worst-case scenario, if a system failure hits your EBS, you won't be able to restore the snapshots taken from your encrypted EBS volume. They're as good as gone.

Security Specialty Exam Notes:

You can't recover a deleted KMS key.

To recover files from the encrypted EBS volume, create a new EBS volume and mount it to the same instance where the encrypted EBS is attached. Then, migrate the files from the old EBS to the new EBS.

If you create an EC2 instance with an encrypted EBS volume but lack sufficient KMS key permissions, the instance creation proceeds but the instance remains in a pending state briefly before being terminated.

References:
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSEncryption.html#migrate-data-encrypted-unencrypted
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AmazonEBS.html

Vault Locking in Amazon S3 Glacier

A Glacier Vault can be described as a container for your archived objects in S3 Glacier. To begin using Amazon S3 Glacier, you need a vault. Creating and deleting vaults can be easily done in the AWS Management Console, but interacting with them requires you to use the APIs. For example, let's say you want to upload images or log files to your vault. To do so, you would either use the AWS CLI or write code that would upload these objects.

Large corporations often have compliance requirements with how they store their data. To meet these requirements, you can use a feature in S3 Glacier called a Vault Lock. S3 Glacier Vault Lock allows you to create a vault lock policy that specifies how your archives will be handled. You can specify controls such as "write once read many" (WORM) in a vault lock policy and lock the policy from future edits. Once locked, the policy can no longer be changed.

You can include a bunch of controls in a vault lock policy, such as data retention based on duration or tags. These policies are written similarly as IAM Policies, which follow JSON formatting. You can set one vault lock policy for each vault.

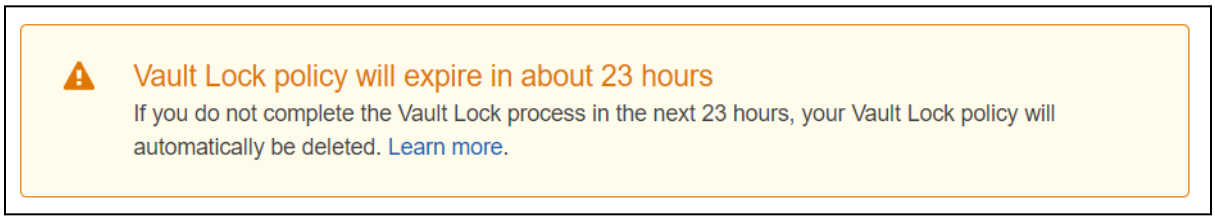
How To Lock Your Glacier Vault Using Glacier CLI command

1. ***Initiate the lock by attaching a vault lock policy to your vault.***

   To begin, call the `initiate-vault-lock` command and pass the necessary parameters. You need to pass your AWS Account Id and vault name after the `--account-id` and `--vault-name` parameter respectively. Both parameters take a string value. The `--policy` parameter takes a json value. Save your vault policy as a json file. Enter the file's path after the `--policy` parameter.

```
aws glacier initiate-vault-lock --account-id [AccountId] --vault-name
[VaultName] --policy file://examplePolicy.json
```

This command will set the lock to an in-progress state and return a lock ID. While in the in-progress state, **you have 24 hours to validate your vault lock policy before the lock ID expires.**

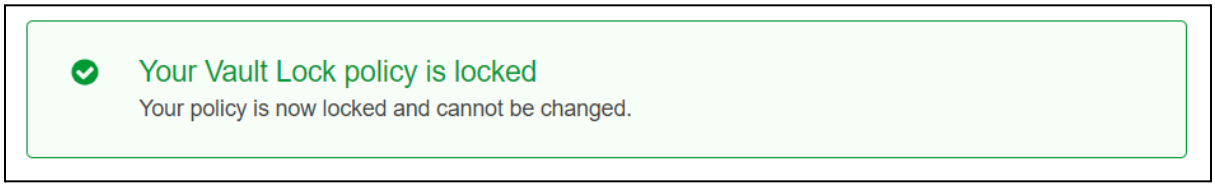


2. ***Use the lock ID to complete the locking process.***

To do so, call the `complete-vault-lock` and pass the necessary parameters. The first two parameters are identical to the first step. This time, we're passing the lock Id on the `--lock-id` parameter.

```
aws glacier complete-vault-lock --account-id [AccountId] --vault-name
[VaultName] --lock-id [LockId]
```

You can verify the lock state of your vault on the Glacier Console.



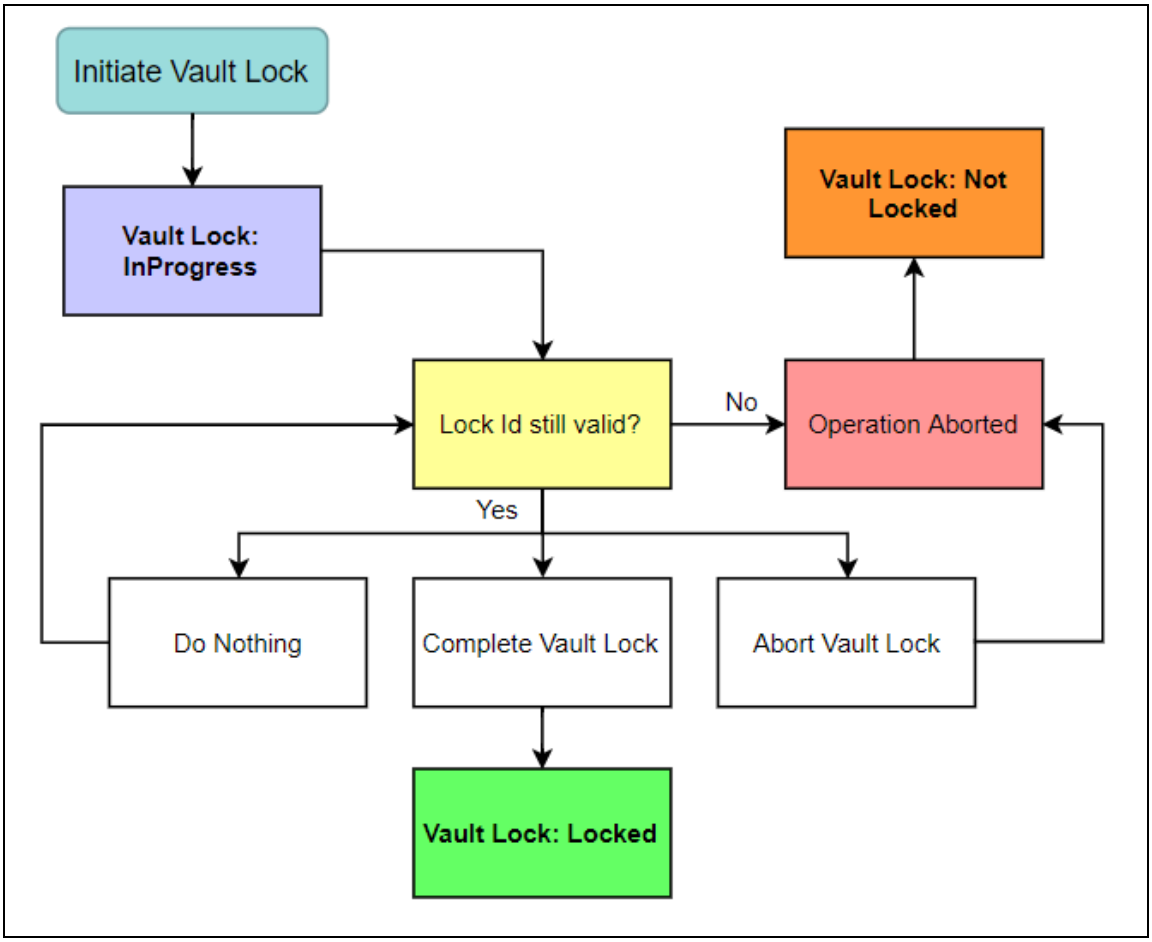
How To Abort Your Vault Lock Operation

In cases where you need to revise your vault lock policy, you can abort the pending operation. A small caveat though: **the revision must be done within the 24-hour timeframe.**

Call the `abort-vault-lock` command. This time, you only have to pass your AWS account id and the name of the vault associated with your vault lock policy. As a result, the state of your vault lock will revert to "Not locked". You can now start revising your vault policy. After making changes, you need to call the `initiate-vault-lock` operation again.

```
aws glacier abort-vault-lock --account-id [AccountId] --vault-name [VaultName]
```

You can use the vault locking's simplified workflow below for your reference.



References:
https://docs.aws.amazon.com/amazonglacier/latest/dev/vault-lock.html
https://docs.aws.amazon.com/amazonglacier/latest/dev/api-InitiateVaultLock.html

Drift detection

As you manage your infrastructure, changes can occur between your stack's actual configuration and the configuration defined in the CloudFormation templates. These discrepancies, known as "drift," can potentially lead to unexpected behavior or security vulnerabilities.

CloudFormation comes with a built-in Drift Detection feature that enables you to monitor and detect when a stack drifts from its expected configuration. Drift detection compares the current stack resources with the expected configurations defined in the corresponding CloudFormation template. If the stack's live configuration deviates from the template, AWS CloudFormation will flag this as drift. This might include modifications to resource properties that were not made using CloudFormation, or resources that have been added or deleted outside of CloudFormation.

Security implications of drifts

Drift can lead to security risks and compliance issues, especially when the misalignment involves security-critical resources such as IAM roles, security groups, or S3 bucket policies. For instance, consider a scenario where an S3 bucket's policy is modified directly through the AWS console, resulting in a drift from the original CloudFormation template configuration. This modification may inadvertently relax the access restrictions, allowing unauthorized access to the bucket's data.

Remediating drifts

To handle drifts effectively, you can employ a proactive approach that uses EventBridge and AWS Lambda. You can configure EventBridge to listen for CloudFormation drift detection status changes. Upon detecting a drift, EventBridge can trigger a Lambda function designed to automatically remediate the drift. This function may reset configurations to align with the CloudFormation template, ensuring your resources remain consistent with their desired state. This automation not only reduces the manual labor involved in drift rectification but also significantly decreases the time window in which your infrastructure may be at risk.

Data Protection in Amazon Managed Service for Apache Flink

Amazon Managed Service for Apache Flink offers a fully managed Apache Flink service to efficiently process and analyze streaming data, enabling real-time insights and responses to business demands. Leveraging this serverless service, users can swiftly construct SQL queries and Java applications utilizing pre-built templates and operators for diverse processing tasks, including data organization, transformation, aggregation, and

analysis at any scale.

Important Data Protection Concepts:

- **Data Sources and Encryption:**
  Amazon Managed Service for Apache Flink integrates seamlessly with services supporting data encryption, such as Amazon Data Firehose and Amazon S3, ensuring robust data protection throughout the data lifecycle.

- **Encryption at Rest:**
  - Incoming data from Kinesis data streams can be encrypted using the StartStreamEncryption API.
  - Output data can be securely stored in an encrypted Amazon S3 bucket via Amzaon Data Firehose, with the option to specify the encryption key for enhanced security.
  - Application code, durable application storage, and running application storage are all encrypted at rest, providing comprehensive protection for sensitive data and application resources.

- **Encryption In Transit:**
  - Managed Service for Apache Flink guarantees encryption for all data in transit, with encryption enabled by default for all applications. This encryption covers data flowing from Amazon Data Streams to Managed Service for Apache Flink, between internal components within the service, and between Managed Service for Apache Flink and Amazon Data Firehose, ensuring data confidentiality during transmission.

- **Key Management:**
  - Data encryption in Managed Service for Apache Flink utilizes service-managed keys without support for customer-managed keys. This approach simplifies key management while maintaining high levels of security for encrypted data.

References:

https://docs.aws.amazon.com/managed-flink/latest/java/data-protection.html
https://aws.amazon.com/managed-service-apache-flink/

Protecting Your S3 Bucket

Amazon S3 is a versatile object storage solution that boasts virtually unlimited storage capacity. You can expect that your files will be durably stored in S3 given that AWS provides an SLA for this service. When creating your S3 bucket, AWS provides you with a unique bucket URL that you can use to access your S3 bucket directly from the public Internet if you have public access enabled.

Amazon S3 is a service that is not used within a VPC. This means that traffic does not pass through VPC resources such as Internet gateways or NAT gateways. This also means that, for security, we cannot use security groups and network access control lists to control who can access what objects in our bucket.

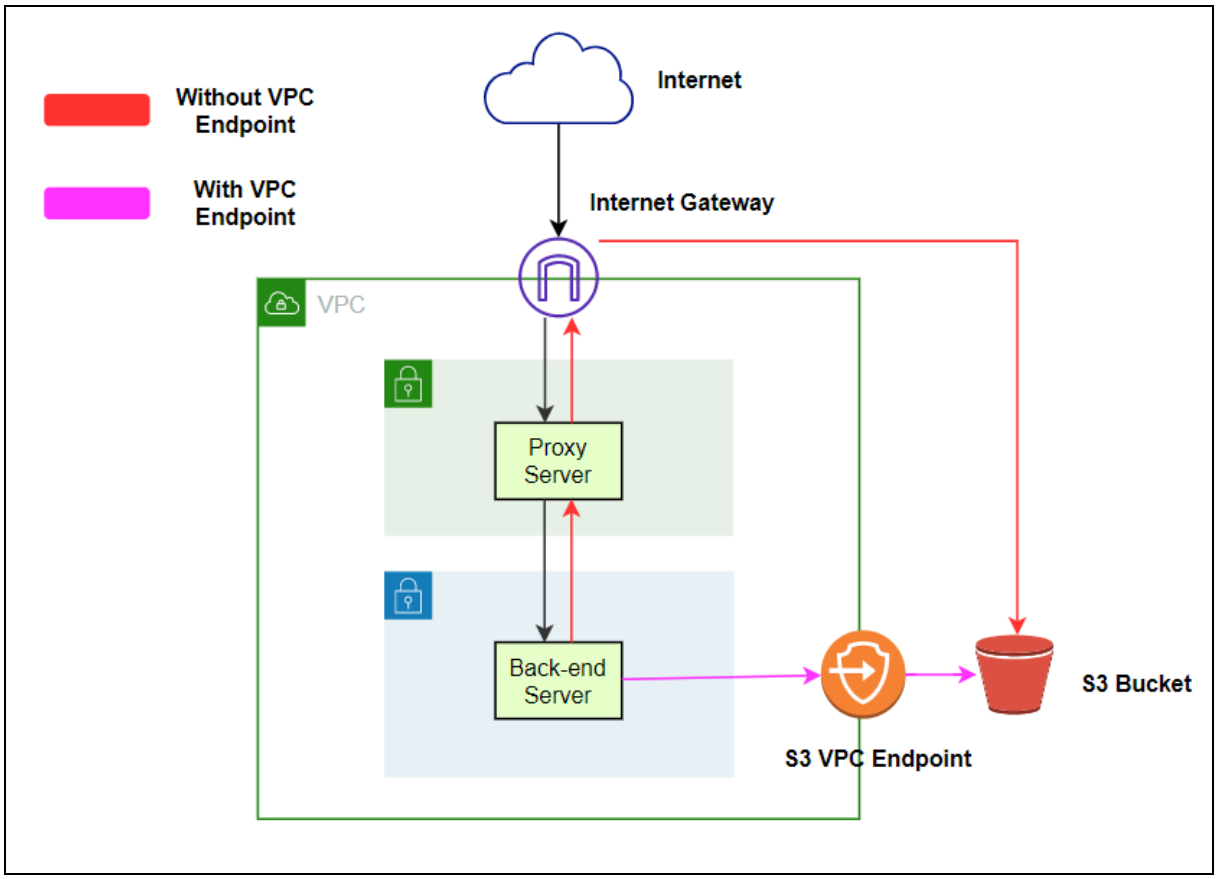
This section looks into implementing security in Amazon S3.

Using VPC endpoints to whitelists sources

In the other chapters, we talked about restricting access to private EC2 instances by whitelisting trusted sources using a proxy server like Squid. This time, we'll see how whitelisting can be done on an S3 bucket.

Let's say you're hosting a web application on an EC2 instance that downloads files from an S3 bucket based on the client's requests. By default, your download requests are routed through an Internet gateway and use the public Internet to connect to Amazon S3. With the VPC endpoint, you can privately connect your VPC to Amazon S3. You add a gateway entry in your VPC's route table to communicate between your AWS resources, such as Amazon EC2 instances. Your S3 request passes through the gateway instead of the public Internet. The VPC endpoint is a regional service. You must create the endpoint in the same region as the VPC you want to link it to.

As you can see in the diagram below, it would be difficult to whitelist sources without a VPC endpoint. The response from Amazon S3 would have to travel back to the Internet gateway then to the proxy server and finally back to your back-end server. Then, you would need to send the requested file back to the client. With the VPC endpoint, you have a direct internal connection to the S3 bucket, and the traffic flow is much simpler and more straightforward.



VPC endpoints for S3 are secured through VPC endpoint access policies, which allows you to set which S3 buckets the endpoints should and should not have access to. By default, any user or service within the VPC using credentials from any AWS account has access to any Amazon S3 resource. Use these together with S3 bucket policies to further refine access control over your buckets and objects.

We can whitelist sources on Amazon S3 by using the "**Condition**" block on the bucket policy. For example, we want to explicitly allow the "**12.11.12.11/32**" IP address and the "**vpce-5555666**" vpc endpoint to perform all S3 operations on all the objects inside the "**examplebucket**" bucket. We can achieve that by adding the Condition operators: "**StringEquals**" & "**IpAddress**" and plugging their corresponding values.

```
{
      "Statement": [{
            "Sid": "VPCe and SourceIP",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": [
                  "arn:aws:s3:::examplebucket/*"
            ],
            "Condition": {
                  "StringEquals": {
                        "aws:sourceVpce": "vpce-5555666"
                  },
                  "IpAddress": {
                        "aws:SourceIp": "12.11.12.11/32"
                  }
            }
      }]
}
```

Encryption at rest

Encryption at rest refers to the encryption of objects as it gets stored on an S3 bucket. In S3, we can either enable the default encryption or write our own code by calling the correct KMS API commands. You can enable the default encryption of a bucket on the S3 console. Default encryption supports SSE-S3 and SSE-KMS. Once enabled, all objects sent to your bucket are encrypted automatically.

Encryption in transit

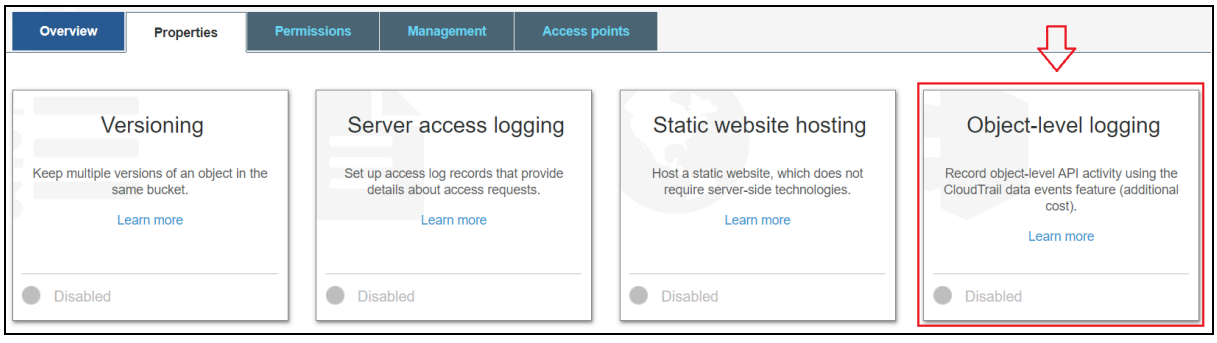
Amazon S3 permits HTTP and HTTPS requests by default. HTTP, as we all know, is an insecure protocol and is rarely used nowadays. Naturally, you'd want to protect your data not only at rest but also in transit. In order to do so, we use a similar method in whitelisting through the "**Conditional**" block. But this time we'll make use of the "**aws:SecureTransport**" Conditional operator. This operator is a boolean type which only accepts 2 values (*true or false*).

```
{
      "Statement": [{
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": [
                  "arn:aws:s3:::examplebucket/*"
            ],
            "Condition": {
                  "Bool": {
                        "aws:SecureTransport": "True"
                  }
            }
      }]
}
```

This bucket policy will only allow HTTPS requests to all the objects inside the "examplebucket" bucket. All HTTP requests will be denied.

S3 API Object-level logging

By default, CloudTrail captures bucket-level API operations and sends logs to an S3 bucket of your choice. In most cases, this won't be of much help in debugging an issue, as most data events come from object-level APIs. You can find object-level logging on the S3 console under the properties tab. There, you can find different selections for S3 bucket properties including object-level logging. Enabling this will allow CloudTrail to record object-level API actions such as ***GetObject*** and ***PutObject***. Take note that this will incur additional costs.



Amazon Macie for Personally Identifiable Information (PII)

Amazon Macie is a fully managed service that uses machine learning to classify sensitive data such as personally identifiable information (PII) or intellectual property, regulatory documents, API keys, and secret keys.

Amazon Macie allows you to achieve the following:

- Identify and protect various data types, including PII, PHI, regulatory documents, API keys, and secret keys.
- Verify compliance with automated logs that allow for instant auditing.
- Identify changes to policies and access control lists.
- Observe changes in user behavior and receive actionable alerts.
- Receive notifications when data and account credentials leave protected zones.
- Detect when large quantities of business-critical documents are shared internally and externally.

You can use Macie to analyze S3 buckets in your account to discover usage patterns on any sensitive files that you have. Macie will give alerts if it detects unauthorized access or accidental data leaks.

References:
https://aws.amazon.com/premiumsupport/knowledge-center/block-s3-traffic-vpc-ip/
https://docs.aws.amazon.com/AmazonS3/latest/dev/bucket-encryption.html
https://aws.amazon.com/premiumsupport/knowledge-center/s3-bucket-policy-for-config-rule/
https://aws.amazon.com/about-aws/whats-new/2017/08/introducing-amazon-macie/

Amazon S3 Encryption

1. Server-side encryption-KMS (SSE-KMS):
    - Amazon S3 facilitates the encryption for you
    - You can use the AWS-Managed KMS key for S3 or a customer-managed KMS key to encrypt S3 objects.
    - To encrypt with a customer-managed KMS key, make sure to add the kms:GenerateDataKey and kms:Decrypt permissions to its key policy.
    - To enforce the encryption of objects as they get uploaded, include a condition in your bucket policy that denies requests that do not include the `x-amz-server-side-encryption` header with `aws:kms` as the value.

2. Server-side encryption-S3 (SSE-S3)
    - Amazon S3 facilitates the encryption for you
    - Amazon S3 uses an AES-256 KMS key that it owns and manages for encryption.
    - To enforce the encryption of objects, include a condition in your bucket policy that denies requests that do not include the **`x-amz-server-side-encryption`** header with **`AES256`** as the value.

3. Server-side encryption-Customer-provided encryption keys (SSE-C)
    - Amazon S3 facilitates the encryption, but you have to provide the encryption key along with the data as part of your request.
    - To retrieve encrypted data, you must include the same encryption key in your request, so Amazon S3 can decrypt it.
    - Amazon S3 does not store your encryption keys. You're responsible for managing it, meaning, if you lose your encryption keys, your data is as good as gone.
    - You must include the following encryption key information in your request:
        - x-amz-server-side-encryption-customer-algorithm
        - x-amz-server-side-encryption-customer-key
        - x-amz-server-side-encryption-customer-key-MD5
    - SSE-C only accepts AES-256 keys and rejects requests made over HTTP. It only accepts HTTPS requests, thus, it enforces both encryptions in transit and encryption at rest.

4. Client-side encryption
    - Data is encrypted by the sender before it reaches Amazon S3
    - The management of keys and encryption/decryption of data is the sender's responsibility.
    - You may use a client-side cryptographic library like OpenSSL or AWS Encryption SDK to help you with the encryption and decryption of data as you send and retrieve them.

Reference:
https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingEncryption.html

Redacting PII in an S3 object on the fly using S3 Object Lambda

Traditionally, if you needed to process or modify data stored in an S3 bucket, you would have to create a separate copy of it. For example, you're storing CSV files containing personally identifiable information (PII), such as names and bank account numbers. To share a version of the file with users while only allowing access to certain attributes, such as the last four digits of a bank account, you would need to create a censored copy of the file.

With S3 object lambda, you don't have to do this. S3 Object Lambda enables you to process data **as it is being retrieved** from S3 using Lambda functions. You can transform or augment data on-the-fly without modifying your source data.

To use S3 Object Lambda to redact PII on the fly, you can create a Lambda function that parses the contents of the object as it is being requested and replaces any PII with a placeholder value, such as "REDACTED" or "***". You can use regular expressions on the function to detect PIIs or leverage AI services like Amazon Comprehend to automatically detect PII on the data being requested.

References:
https://aws.amazon.com/s3/features/object-lambda/
https://aws.amazon.com/blogs/aws/introducing-amazon-s3-object-lambda-use-your-code-to-process-data-as-it-is-being-retrieved-from-s3/

Design and Configure Secure Data Replication and Backup Solutions

Secure data replication and backup strategies protect against accidental deletion, corruption, and ransomware attacks. Effective backup solutions follow the 3-2-1 rule: three copies of data, on two different media types, with one copy offsite. In AWS, this means production data, automated backups in the same region, and cross-region or cross-account copies for geographic and administrative separation.

AWS Backup

- Fully managed backup service that centralizes and automates data protection across AWS services.
- Policy-based backup management eliminates custom scripts.
- Supports cross-region and cross-account backup copies for disaster recovery.
- Enforces compliance through backup policies and audit capabilities.

Supported Services

- **Compute:** EC2 instances (including EBS volumes and metadata)
- **Storage:** EBS, EFS, FSx (Windows, Lustre, NetApp ONTAP, OpenZFS)

- **Databases:** RDS, Aurora, DynamoDB, Neptune, DocumentDB
- **Hybrid:** Storage Gateway volumes, VMware VMs

Key Features

Backup Plans

- Define backup frequency, retention, and lifecycle policies
- Schedule-based rules (hourly, daily, weekly, monthly)
- Continuous backups (point-in-time recovery) for supported services
- Automatic transition to cold storage for cost optimization
- Assign plans to resources using tags or resource IDs

Backup Vault

- Logical container for organizing and securing backups
- Encryption with AWS KMS keys (required for all backups)
- Resource-based access policies control who can access backups
- Vault Lock for immutable backups (WORM - Write Once Read Many)
- Supports cross-account and cross-region copies

Backup Vault Lock (Immutable Backups)

- Prevents backup deletion for ransomware protection
- **Governance mode:** Prevents deletion unless user has special permissions
- **Compliance mode:** Nobody can delete, even root user (true immutability)
- Supports minimum and maximum retention periods
- Cannot be disabled once enabled in compliance mode
- Critical for SEC 17a-4, FINRA, HIPAA compliance

Cross-Region and Cross-Account Copies

- Automatically copy backups to different regions for disaster recovery
- Copy to separate AWS accounts for administrative isolation
- Protects against regional failures and account compromise
- Independent retention policies for copies
- Re-encryption with different KMS keys in destination

Point-in-Time Recovery (PITR)

- Continuous backup for RDS, Aurora, DynamoDB
- Restore to any point within retention window

- Minimal data loss (typically seconds of RPO)

Lifecycle Management

- Transition backups to cold storage after specified days (90% cost savings)
- Minimum cold storage retention: 90 days
- Restore from cold storage takes longer (hours vs minutes)

Security Features

Encryption

- All backups encrypted at rest using AWS KMS
- Choose AWS-managed or customer-managed KMS keys
- Different encryption keys for backup copies across accounts/regions
- In-transit encryption using TLS for all API calls

Access Control

- IAM policies control who can create, modify, or delete backup plans
- Vault access policies control access to backups themselves
- Service-linked role grants AWS Backup permissions
- Separate permissions for backup creation vs. restoration (least privilege)

Audit and Compliance

- AWS Backup Audit Manager provides compliance reporting
- Pre-built frameworks (PCI, HIPAA, GDPR)
- Custom frameworks for organization-specific requirements
- Continuous monitoring of backup policies and practices
- Integration with AWS Config for compliance rules
- CloudTrail logs all backup and restore operations

AWS Organizations Integration

- Centrally manage backup policies across all accounts
- Backup policies deployed via service control policies (SCPs)
- Delegated administrator for backup management
- Consolidated billing and organization-wide compliance reporting

Amazon Data Lifecycle Manager (DLM) - Security Perspective

Cross-Account Snapshot Copies for Ransomware Protection:

- Copy EBS snapshots to separate AWS account for isolation
- Prevents attackers in production account from deleting backups
- Backup account uses different IAM credentials and MFA
- SCPs prevent snapshot deletion in backup account
- Regular testing of cross-account restore procedures

Snapshot Encryption

- Snapshots inherit encryption from source EBS volume
- Can re-encrypt with different KMS key during cross-account copy
- Destination account controls KMS key permissions
- Supports compliance with data residency requirements

Fast Snapshot Restore (FSR)

- Pre-warms EBS volumes for instant performance
- Critical for RTO requirements in disaster recovery
- Eliminates I/O latency after restore
- Enable on DR snapshots in alternate region
- Higher cost but rapid recovery after ransomware attack

AMI Lifecycle

- Automate AMI creation from tagged EC2 instances
- Automated security patching through AMI rebuilds
- Immutable infrastructure pattern for security
- Cross-region AMI copies for disaster recovery
- Automatic deregistration of old AMIs

AWS DataSync

- Automated data transfer service for moving data between on-premises and AWS, or between AWS services.
- Accelerates migration, replication, and disaster recovery workflows.
- Built-in encryption, integrity verification, and access control.

Data Transfer Capabilities

- On-premises to AWS (NFS, SMB to S3, EFS, FSx)
- AWS to AWS (S3 to S3, EFS to EFS, cross-region)
- AWS to on-premises (reverse sync for disaster recovery)

- Incremental transfers (only changed data)
- Automatic retries and bandwidth throttling
- Up to 10 Gbps per task

Supported Sources and Destinations

- **Sources:** NFS, SMB, HDFS, S3, EFS, FSx (all types)
- **Destinations:** S3 (all storage classes), EFS, FSx (all types)
- Self-managed object storage via S3 API

Security Features

Encryption

- TLS 1.2 encryption for all data transfers (in transit)
- Destination data encrypted using service's native encryption
- S3: SSE-S3, SSE-KMS, or SSE-C
- EFS/FSx: Encryption at rest with KMS

Data Integrity

- End-to-end checksums validate data integrity
- Automatic verification at source and destination
- Detects corruption during transfer
- Failed transfers automatically retried

Access Control

- IAM roles control DataSync permissions
- VPC endpoints for private connectivity (no internet)
- Security groups restrict network access
- CloudTrail logs all DataSync API operations

DataSync Agent

- VM deployed on-premises (VMware, Hyper-V, KVM) or EC2
- Establishes secure TLS-encrypted connection to DataSync service
- Activated using unique activation key
- Site-to-Site VPN or Direct Connect for secure connectivity

Use Cases

- **Hybrid Backup:** Replicate on-premises data to S3 for backup
- **Disaster Recovery:** Continuous replication to AWS for DR readiness

- **Data Migration:** Secure migration of large datasets to AWS
- **Archive:** Transfer compliance data to S3 Glacier with Object Lock

Ransomware Protection Strategies

Immutable Backups

- AWS Backup Vault Lock in compliance mode prevents deletion
- S3 Object Lock in compliance mode for archived backups
- Minimum retention period ensures backups survive attacks
- Even root user cannot delete locked backups

Administrative Separation

- Store backups in separate AWS account
- Different IAM credentials and MFA for backup account
- SCPs prevent deletion in backup account
- Backup account not accessible from production

Multi-Region Backups

- Cross-region copies protect against regional compromise
- Separate encryption keys in each region
- Independent retention policies
- Regional isolation limits blast radius

Access Controls

- Least privilege IAM policies for backup operations
- Separate roles for backup creation vs. restoration
- MFA required for backup deletion or vault modification
- Condition keys restrict access by source IP or time

Monitoring and Alerting

- CloudWatch alarms for backup failures
- SNS notifications for vault lock modifications
- GuardDuty detects unusual API activity
- EventBridge rules for automated incident response

Testing and Validation

- Regular restore testing validates backup integrity
- Automated restore to test environment

- Conduct tabletop exercises for ransomware scenarios

Versioning and Point-in-Time Recovery

- S3 Versioning preserves all object versions
- DynamoDB PITR enables recovery to any second
- RDS automated backups with 35-day retention
- Multiple restore points increase recovery options

Common Exam Scenarios

Scenario 1: Ransomware Protection

- **Requirement:** Protect backups from deletion by attackers who compromise AWS account
- **Solution:** AWS Backup with Vault Lock in compliance mode; cross-account copies to separate account; SCPs preventing deletion; MFA for backup account

Scenario 2: Regulatory Compliance

- **Requirement:** Retain backups for 7 years, prevent early deletion (SEC 17a-4)
- **Solution:** AWS Backup Vault Lock compliance mode with 7-year minimum retention; S3 Object Lock; Backup Audit Manager with PCI/FINRA framework; CloudTrail logging

Scenario 3: Disaster Recovery

- **Requirement:** RTO of 4 hours, RPO of 15 minutes
- **Solution:** AWS Backup with continuous backups (PITR) for RDS; cross-region copies; DLM with Fast Snapshot Restore; automated restore testing

Scenario 4: Hybrid Cloud Backup

- **Requirement:** Securely replicate on-premises file server to AWS
- **Solution:** DataSync agent on-premises; scheduled transfers to S3 with SSE-KMS; S3 Versioning; Site-to-Site VPN; VPC endpoints

Scenario 5: Multi-Account Architecture

- **Requirement:** Centralize backup management across 50 accounts
- **Solution:** AWS Backup with Organizations integration; backup policies via SCPs; delegated administrator; cross-account vault access

References:
https://docs.aws.amazon.com/aws-backup/latest/devguide/whatisbackup.html

https://docs.aws.amazon.com/aws-backup/latest/devguide/vault-lock.html
https://docs.aws.amazon.com/datasync/latest/userguide/what-is-datasync.html
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/snapshot-lifecycle.html

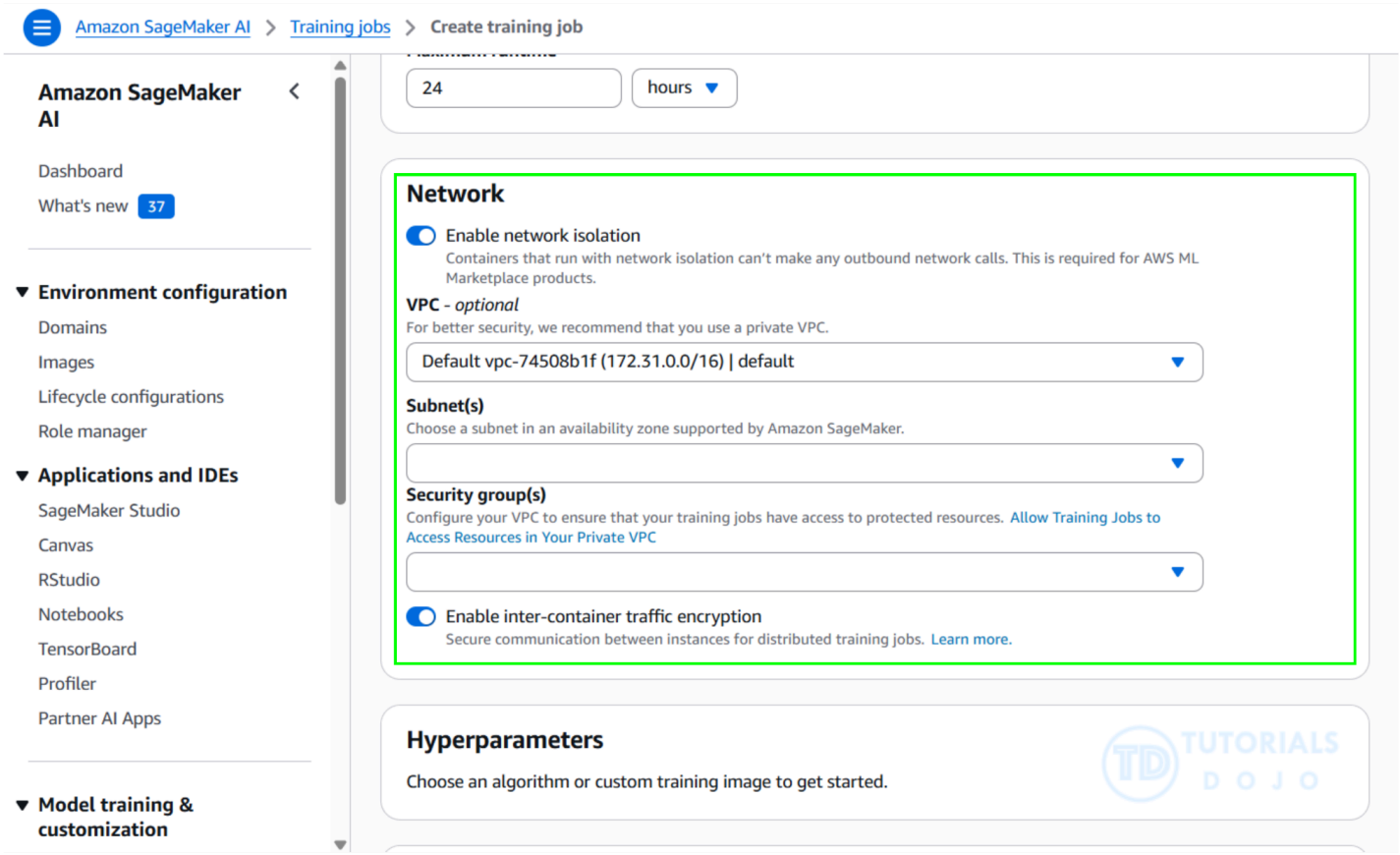
Design and Configure Inter-Resource Encryption In-Transit

Inter-resource encryption in-transit protects data as it moves between compute nodes within distributed systems on AWS.

Amazon EMR provides comprehensive encryption options through security configurations, enabling in-transit encryption for open-source frameworks including Hadoop MapReduce, Apache Spark, Presto, and Tez. When enabled, EMR automatically configures TLS encryption for data exchanges between cluster nodes, including HDFS communications between NameNode and DataNode, MapReduce encrypted shuffle operations, and Presto internal communications between coordinators and workers. Organizations can specify TLS certificates through PEM files stored in Amazon S3 or implement custom certificate providers as Java classes, with EMR automatically distributing these artifacts to each cluster node. For maximum security coverage, AWS recommends enabling both in-transit encryption and Kerberos authentication, as certain network endpoints only support TLS when both features are activated.

Amazon EKS takes a different approach to encryption in-transit, as Kubernetes lacks native encryption features for pod-to-pod communication. Traffic between Nitro-based instances is automatically encrypted at the hardware layer using AES-256 when instances are in the same Region and VPC or peered VPCs, though this encryption is bypassed when traffic passes through intermediate hops like transit gateways or load balancers. Organizations can implement additional encryption layers using service meshes such as AWS App Mesh, Istio, or Linkerd, which support mutual TLS authentication for encrypting traffic between services, or deploy network plugins like Cilium with WireGuard for transparent node-to-node encryption. For storage-level encryption, Amazon EFS supports transport encryption that can be enabled by adding the TLS parameter to mount options in Kubernetes persistent volumes.

Amazon SageMaker AI enables inter-container traffic encryption for distributed training jobs through a simple configuration option that encrypts data transmitted between training instances. When enabled through the `EnableInterContainerTrafficEncryption` parameter, SageMaker automatically establishes encrypted tunnels between nodes participating in distributed machine learning workloads. While this encryption increases security for sensitive healthcare and regulated workloads, it may impact training time for distributed deep learning algorithms, though most built-in algorithms like XGBoost, DeepAR, and linear learner are not impacted.



The **AWS Nitro System** provides automatic encryption at the hardware level for supported instance types using AES-256-GCM authenticated encryption, encrypting traffic transparently between instances in the same Region and VPC with zero performance impact, as the encryption is offloaded to specialized Nitro Card hardware rather than consuming CPU resources. This automatic encryption is available across numerous modern instance families including M7g, M8g, C5n, and R5n series, requiring no configuration or key management from customers.

References:
https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-data-encryption-options.html
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/data-protection.html
https://docs.aws.amazon.com/eks/latest/best-practices/network-security.html
https://aws.amazon.com/blogs/containers/transparent-encryption-of-node-to-node-traffic-on-amazon-eks-using-wireguard-and-cilium/
https://docs.aws.amazon.com/sagemaker/latest/dg/encryption-in-transit.html
https://docs.aws.amazon.com/whitepapers/latest/security-design-of-aws-nitro-system/the-components-of-the-nitro-system.html

Describe the Differences Between Imported Key Material and AWS Generated Key Material

AWS Key Management Service (KMS) allows you to create customer managed keys for encrypting your data. When creating a KMS key, you have the choice between letting AWS generate the cryptographic key material or importing your own key material from an external source. Understanding the differences between these two approaches is critical for the SCS-C03 exam and for making informed decisions about key management in production environments.

Key material is the actual cryptographic material used to encrypt and decrypt data. It consists of the secret bits that make up the encryption key. In AWS KMS, this key material never leaves the Hardware Security Modules (HSMs) in plaintext, regardless of whether it's generated by AWS or imported by you.

AWS Generated Key Material

What is AWS Generated Key Material?

AWS generated key material is cryptographic material created by AWS KMS within FIPS 140-2 validated Hardware Security Modules (HSMs). When you create a KMS key without specifying an origin, AWS automatically generates the key material using cryptographically secure random number generators within the HSMs.

Characteristics of AWS Generated Key Material

- **Automatic Generation** – AWS creates the key material when you create the KMS key
- **Permanent Lifecycle** – Key material exists for the lifetime of the KMS key
- **Cannot Be Deleted Independently** – Only deleted when the entire KMS key is deleted (after waiting period)
- **Automatic Rotation** – Supports automatic annual key rotation (cryptographic best practice)
- **Multi-Region Support** – Can be replicated to other AWS Regions as multi-Region keys
- **No Expiration** – Key material never expires and remains available indefinitely
- **HSM Protection** – Generated and stored in FIPS 140-2 Level 2 validated HSMs (Level 3 for CloudHSM)

Key Management Operations

AWS handles all aspects of key material lifecycle management. The key material is generated within the HSM boundary and never exists in plaintext outside the HSM. AWS manages the durability and availability of the key material by automatically replicating it across multiple HSMs within the same Region. You cannot export or view the actual key material, maintaining its confidentiality.

Automatic Key Rotation

One of the most significant advantages of AWS generated key material is automatic key rotation. When enabled, AWS KMS automatically rotates the key material annually (every 365 days). During rotation, AWS generates new cryptographic material but keeps the same logical key ID and key ARN. Previous versions of key material are retained and used automatically to decrypt data encrypted with older versions. This provides seamless backward compatibility – you never need to re-encrypt existing data. The rotation is completely transparent to your applications, requiring no code changes or operational intervention.

Best Practices for AWS Generated Keys

- Enable automatic key rotation for compliance and security best practices
- Use AWS generated keys unless you have specific requirements for imported keys
- Leverage multi-Region keys when you need cross-Region encryption with the same key
- Take advantage of AWS's built-in durability and availability guarantees
- Monitor key usage with CloudWatch metrics and CloudTrail logs

Imported Key Material

What is Imported Key Material?

Imported key material is cryptographic material that you generate outside of AWS KMS and then import into a KMS key. This approach gives you complete control over the key generation process and the source of your cryptographic material. You might import key material to meet compliance requirements, use existing keys, or maintain a root of trust outside AWS.

Characteristics of Imported Key Material

- **External Generation** – You create the key material outside AWS using your own tools or HSMs
- **Optional Expiration** – You can set an expiration time for the key material
- **Manual Deletion** – You can delete key material independently from the KMS key
- **No Automatic Rotation** – AWS cannot automatically rotate imported key material
- **Single-Region Only** – Cannot be replicated as multi-Region keys
- **Your Responsibility** – You're responsible for key material durability, backup, and rotation
- **Manual Reimport** – You must manually reimport key material if deleted or expired

Import Process Overview

Importing key material is a multi-step process that ensures security throughout. First, you create a KMS key with the origin set to "EXTERNAL" without key material. Next, you download a wrapping public key and import token from AWS KMS. Then, you generate your own 256-bit symmetric key material outside AWS using a cryptographically secure random number generator. You encrypt your key material using the wrapping public key (RSA with either RSAES_OAEP_SHA_256 or RSAES_OAEP_SHA_1). Finally, you upload the encrypted key material and import token to AWS KMS, where it's decrypted inside the HSM and becomes available for use.

Key Material Expiration

When importing key material, you can optionally specify an expiration time. This feature allows you to implement time-based key rotation policies or ensure keys are only valid for specific periods. When key material expires, the KMS key becomes unusable for encryption and decryption operations, but the key itself is not deleted. You can reimport the same key material or import new key material to reactivate the key. AWS KMS automatically publishes a CloudWatch event 45 days before expiration, giving you time to plan for rotation or reimport. This feature is particularly useful for compliance requirements that mandate regular key rotation or for temporary encryption needs where keys should automatically become unusable after a certain period.

Use Cases for Imported Key Material

Several scenarios justify the additional complexity of importing key material:

Regulatory Compliance – Some regulations or security frameworks require that cryptographic keys be generated within specific HSMs or jurisdictions. Importing key material allows you to generate keys in your own certified HSMs and then import them to AWS KMS for use with AWS services.

Root of Trust Outside AWS – Organizations with existing key management infrastructure may want to maintain their root of trust outside AWS. By generating keys in their own HSMs and importing them, they ensure that AWS never has access to the key generation process.

Key Backup and Portability – Imported key material can be backed up in your own secure storage before import. This provides an additional layer of protection and allows you to maintain copies of your keys outside AWS. You can also import the same key material into multiple AWS accounts or Regions (though not as multi-Region keys).

Migration from External Systems – When migrating encrypted data to AWS, you may need to import existing encryption keys so that you can decrypt legacy data without re-encryption.

Temporary Keys – Using the expiration feature, you can create keys that automatically become unusable after a specific time, useful for time-limited projects or temporary access scenarios.

Key Rotation Control – Some organizations want complete control over when and how keys are rotated. Imported key material allows you to implement custom rotation schedules by deleting old key material and importing new material.

Comparison Table

Feature	AWS Generated Key Material	Imported Key Material

Generation Location	AWS KMS HSMs	External (your HSMs/systems)
Lifecycle	Permanent (tied to KMS key)	Flexible (can delete/reimport)
Automatic Rotation	Supported (annual)	Not supported
Multi-Region Keys	Supported	Not supported
Expiration	No expiration	Optional expiration time
Deletion	Only with entire key (waiting period)	Can delete material independently
Durability Responsibility	AWS fully responsible	Shared (you must backup)
Operational Complexity	Low (fully managed)	High (manual processes)
Root of Trust	Within AWS	Can be external
Reimport Capability	N/A	Can reimport anytime
CloudWatch Expiration Events	N/A	45-day advance notification
Use with Multi-Account	Normal CMK sharing	Must import separately per account
Compliance Scenarios	Standard AWS compliance	Custom compliance requirements

References:
https://docs.aws.amazon.com/kms/latest/developerguide/importing-keys.html
https://docs.aws.amazon.com/kms/latest/developerguide/concepts.html#key-origin
https://docs.aws.amazon.com/kms/latest/developerguide/rotate-keys.html
https://docs.aws.amazon.com/kms/latest/developerguide/multi-region-keys-overview.html

Mask sensitive data

Masking sensitive data in transit and at rest is a critical security control to prevent accidental exposure of personally identifiable information (PII), financial data, credentials, and other confidential information. AWS provides built-in data protection capabilities in CloudWatch Logs and Amazon SNS that automatically scan for sensitive data and can audit, redact, or block messages containing such data. These features help organizations maintain compliance with regulations like GDPR, HIPAA, and PCI DSS by preventing sensitive data from being logged or transmitted to unauthorized destinations.

Data protection works by scanning data in real-time using managed data identifiers (patterns for common sensitive data types) and custom data identifiers (organization-specific patterns). When sensitive data is

detected, you can configure policies to audit the finding, mask (redact) the sensitive data, or block the entire message. This automated approach reduces the risk of developers or applications accidentally logging sensitive information that could be exposed through log analysis tools, third-party integrations, or security breaches.

CloudWatch Logs Data Protection

What is CloudWatch Logs Data Protection?

CloudWatch Logs data protection is a feature that helps you discover and protect sensitive data within your log streams. It continuously scans log events as they're ingested into CloudWatch Logs, identifies sensitive data using pattern matching, and takes action based on your configured policies. This prevents sensitive data from being inadvertently stored in logs where it might be accessed by developers, operators, or third-party log analysis tools.

How CloudWatch Logs Data Protection Works

When you enable data protection on a log group, CloudWatch Logs scans every log event as it arrives. The service uses managed data identifiers to detect common sensitive data types like credit card numbers, Social Security numbers, AWS credentials, and other PII. You can also create custom data identifiers using regex patterns to detect organization-specific sensitive data such as employee IDs, internal account numbers, or proprietary identifiers.

Once sensitive data is detected, CloudWatch Logs takes action based on your policy configuration. In audit mode, the service creates findings in CloudWatch Logs but leaves the original log data intact. This allows you to understand what sensitive data is being logged without disrupting existing workflows. In de-identify mode, the service masks (redacts) the sensitive data by replacing it with asterisks or a hash value, while preserving the rest of the log message. This maintains log utility for troubleshooting while protecting sensitive information.

Creating a Data Protection Policy

To implement data protection in CloudWatch Logs, you create a data protection policy and attach it to one or more log groups. The policy is a JSON document that defines which data identifiers to use and what action to take when sensitive data is detected.

A basic policy structure includes the name and version, a statement that specifies the data identifiers and operations. You define which managed data identifiers to use by category (such as Credentials, Financial, or all categories). You can include custom data identifiers by specifying their ARNs. You specify the operation as either Audit (find and log but don't modify) or Deidentify (find and mask the data).

For example, a policy to mask all credentials and financial data would scan for sensitive data in those categories and replace any matches with a masked value. A policy for audit-only mode would detect sensitive

data but not modify the logs, useful when you first want to understand what's being logged before implementing masking.

Findings and Monitoring

When CloudWatch Logs detects sensitive data, it publishes findings to a separate audit log. These findings include details about what type of sensitive data was detected, which data identifier matched, how many instances were found, and metadata about the log group and stream. The findings themselves do not contain the actual sensitive data, maintaining security.

You can query these findings using CloudWatch Logs Insights to understand patterns of sensitive data logging. Common queries include counting findings by data identifier type to see what types of sensitive data are most commonly logged, identifying log groups with the highest frequency of sensitive data to prioritize remediation, tracking findings over time to measure improvement, and finding specific applications or services that are logging sensitive data.

Use Cases for CloudWatch Logs Data Protection

Several scenarios benefit from CloudWatch Logs data protection:

Preventing Credential Exposure: Applications that log error messages or debug information may inadvertently include API keys, access tokens, or passwords. Data protection automatically detects and masks these credentials, preventing them from being exposed in logs where they could be accessed by unauthorized users or leaked through log exports.

PCI DSS Compliance: Payment Card Industry Data Security Standard (PCI DSS) strictly prohibits storing full credit card numbers, CVV codes, or PINs in logs. CloudWatch Logs data protection helps maintain compliance by automatically masking payment card data if it's accidentally logged.

HIPAA Compliance: Healthcare applications must protect patient information including health insurance IDs, DEA numbers, and medical record numbers. Data protection policies can detect and mask this protected health information (PHI) in application logs.

GDPR Compliance: European regulations require organizations to protect personal data including names, addresses, and identification numbers. Data protection helps prevent unauthorized storage and processing of personal data in logs.

Third-Party Log Analysis: When using third-party log analysis tools or SIEM solutions, you may need to send logs outside your direct control. Data protection ensures sensitive data is masked before logs are exported to these external systems.

Amazon SNS Message Data Protection

What is SNS Message Data Protection?

Amazon SNS message data protection scans messages published to SNS topics for sensitive data and can audit or deny the message based on your policy configuration. This prevents sensitive data from being transmitted through SNS to subscribers, which may include email addresses, SMS numbers, HTTP endpoints, Lambda functions, or SQS queues. By enforcing data protection at the messaging layer, you ensure that sensitive data doesn't propagate to multiple downstream systems.

How SNS Message Data Protection Works

When you enable data protection on an SNS topic, every message published to that topic is scanned before delivery. SNS uses the same managed data identifiers as CloudWatch Logs to detect sensitive data patterns. The service examines both the message body and message attributes for sensitive data.

Based on your policy configuration, SNS takes different actions when sensitive data is detected. In audit mode, SNS allows the message to be delivered normally but publishes an audit finding to CloudWatch Logs documenting what sensitive data was detected. This helps you understand what data is flowing through your topics without disrupting existing workflows. In deny mode, SNS blocks the entire message from being delivered to any subscribers and publishes a finding. The publisher receives an error indicating the message was denied due to data protection policy.

Creating an SNS Data Protection Policy

An SNS data protection policy is similar in structure to CloudWatch Logs policies but with different operations. The policy defines which data identifiers to scan for and whether to audit or deny messages containing sensitive data.

For example, a policy to block messages containing credentials would deny any message that contains AWS secret keys, private keys, or other credential types. A policy for auditing financial data would allow messages to flow through but log findings when credit card numbers or bank account numbers are detected. You can create policies that deny based on some data types while auditing others, such as blocking credentials immediately while auditing PII to understand usage patterns before enforcement.

Audit Findings and Monitoring

When SNS detects sensitive data, it publishes findings to CloudWatch Logs. Each finding includes the topic ARN, message ID, which data identifier matched, number of findings, and timestamp. The findings do not include the actual sensitive data or the full message content for security purposes.

You can create CloudWatch alarms based on these findings to alert when sensitive data is detected. For example, you might create an alarm that triggers when more than 10 messages per hour contain credentials, indicating a possible misconfiguration or compromised application. You can also use CloudWatch Logs

Insights to analyze finding patterns, such as identifying which applications are sending sensitive data most frequently or tracking whether finding rates decrease after remediation efforts.

Use Cases for SNS Message Data Protection

SNS data protection addresses several important security scenarios:

Multi-Subscriber Protection: When an SNS topic has multiple subscribers including email addresses, HTTP endpoints, and third-party integrations, you need to ensure sensitive data doesn't reach any of these potentially less secure endpoints. Data protection policies ensure no subscriber receives sensitive data, regardless of their individual security posture.

Email and SMS Security: SNS topics that send notifications via email or SMS are particularly vulnerable to data exposure. Email and SMS are not encrypted end-to-end and may be stored insecurely. Blocking sensitive data from these channels prevents accidental exposure through unencrypted communication methods.

Webhook Protection: Applications often use SNS to send notifications to external webhooks for integrations with third-party services. Without data protection, sensitive data could be transmitted to external services without proper vetting. Policies can prevent this by blocking or auditing messages before they reach external endpoints.

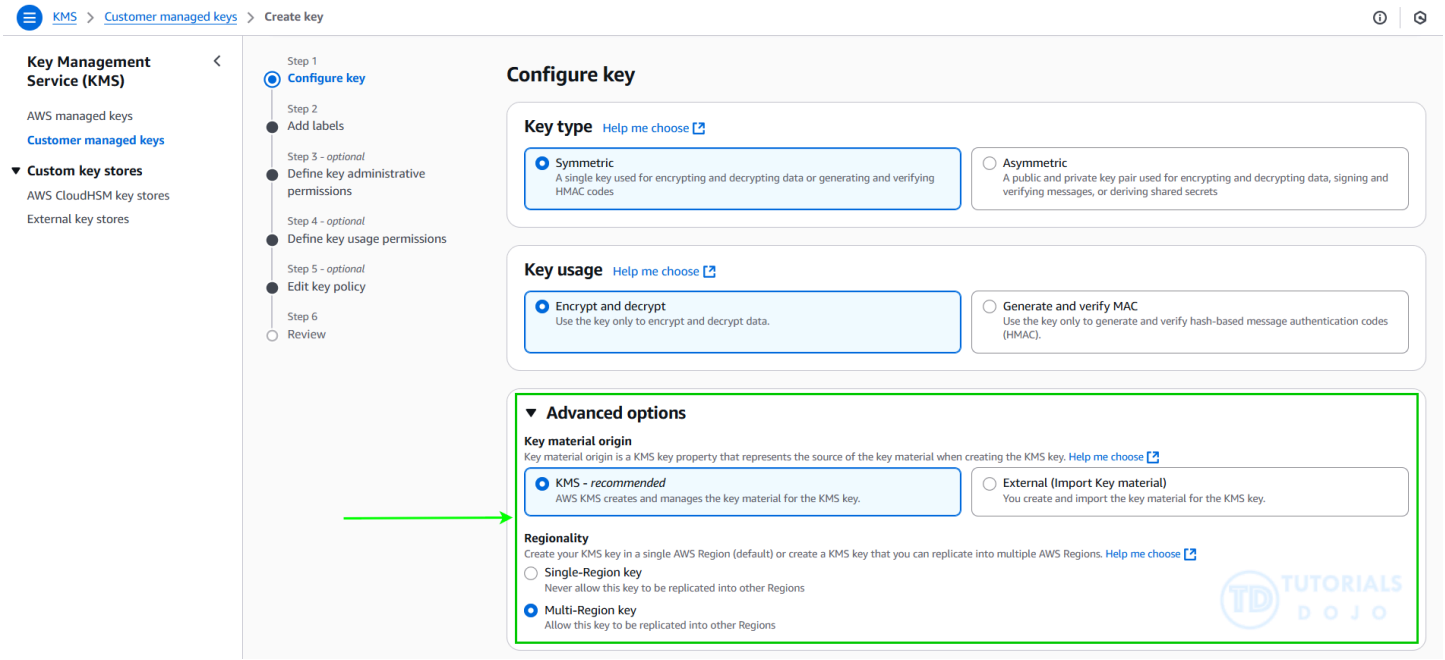
Fan-out Architecture Security: SNS is commonly used for fan-out patterns where one message is delivered to multiple SQS queues or Lambda functions. If any downstream component is compromised or misconfigured, sensitive data could be exposed. Data protection provides a centralized control point to prevent sensitive data from entering the fan-out pattern.

Development and Testing: Development teams may use production-like data in test environments that send notifications through SNS. Data protection policies can prevent real customer data from being sent to test notification endpoints, maintaining separation between production and non-production environments.

References:
https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/protect-sensitive-log-data.html
https://docs.aws.amazon.com/sns/latest/dg/sns-message-data-protection.html
https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/protect-sensitive-log-data-types.html
https://docs.aws.amazon.com/sns/latest/dg/message-data-protection-policies.html

Create and Manage Encryption Keys and Certificates Across Single or Multiple AWS Regions

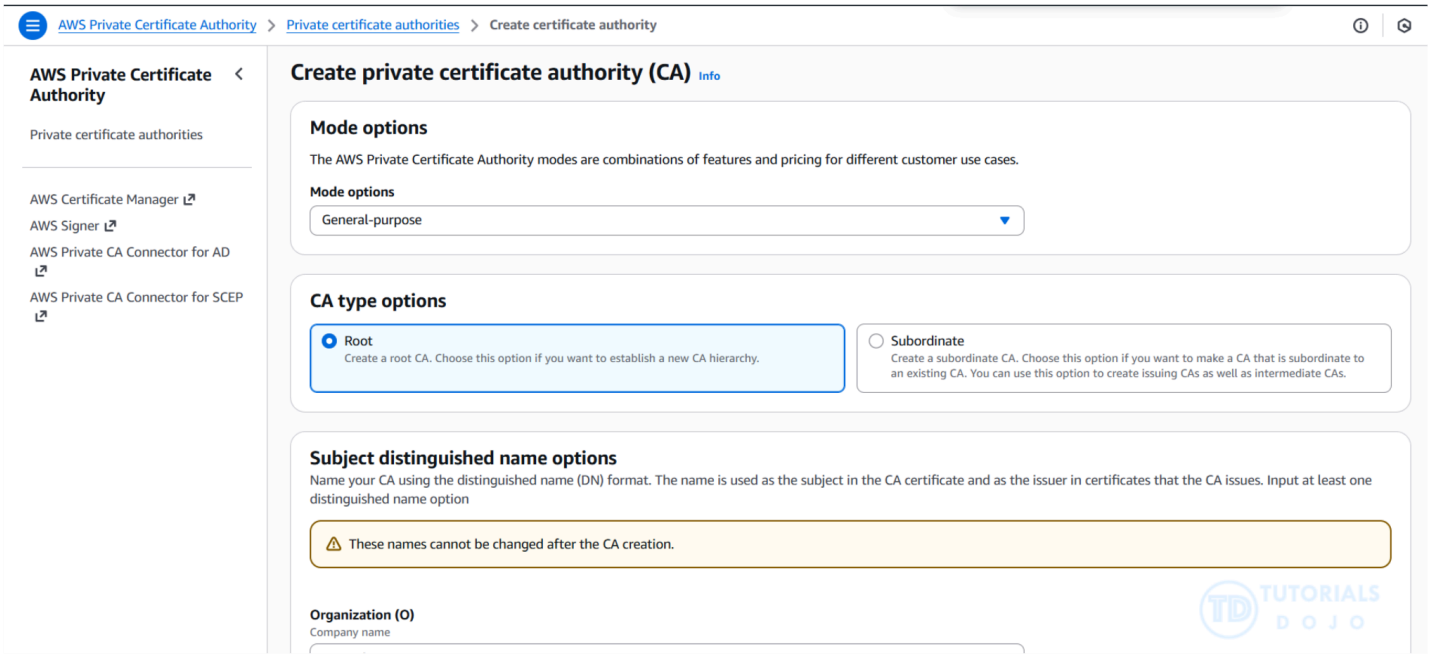
AWS Key Management Service enables organizations to create and manage customer managed KMS keys for cryptographic operations with full control over key lifecycle, policies, and usage permissions. Customer managed KMS keys support both single-region and multi-region configurations, with multi-region keys sharing identical key material across multiple Regions while maintaining independent key policies, grants, aliases, and tags in each Region. Multi-region primary keys can be replicated to other Regions within the same AWS partition using the AWS KMS console or `ReplicateKey` API, creating replica keys with distinctive key IDs beginning with the mrk- prefix that enable interoperable cryptographic operations across Regions without re-encrypting data. Organizations can update the primary Region through the `UpdatePrimaryRegion` operation, converting the current primary key to a replica and promoting a specified replica to primary, which is useful for disaster recovery scenarios or relocating key administration closer to operational centers. Each KMS key costs one dollar per month regardless of rotation history, with multi-region keys priced individually per Region, and organizations must manage key policies, grants, and access controls independently for each key even when they are related multi-region keys.



Automatic key rotation enhances security by generating new cryptographic material while maintaining backward compatibility for decryption operations using previous key material versions. AWS KMS supports configurable rotation periods between 90 and 7 years (2560 days) with a default annual rotation schedule, and organizations can perform on-demand rotation at any time using the `RotateKeyOnDemand` operation without affecting existing automatic rotation schedules. When automatic key rotation is enabled for multi-region primary keys, the rotation configuration automatically propagates to all replica keys, ensuring synchronized key material across all related multi-region keys with access to all previous versions for decrypting older

ciphertext. AWS managed keys automatically rotate annually and cannot be disabled, while customer managed symmetric encryption keys with AWS KMS-generated key material support both automatic and on-demand rotation, with each newly generated backing key costing an additional dollar per month to retain all versions for decryption purposes. Organizations should monitor key rotation events through AWS CloudTrail logs and Amazon CloudWatch metrics, implementing automated responses such as AWS Lambda functions that trigger notifications or incident response actions when specific key management operations are detected, while using the `kms:RotationPeriodInDays` condition key in key policies to enforce organizational requirements for rotation frequency.

AWS Private Certificate Authority provides managed CA services for creating hierarchical certificate infrastructures across multiple Regions and AWS accounts, with CAs existing as Regional resources that must be explicitly created in each desired Region since AWS never automatically replicates CAs. Organizations can architect resilient CA hierarchies by deploying redundant root CAs in different Regions for disaster recovery, or by creating redundant subordinate CAs across Regions chained to a single root CA, with each approach offering different operational complexity and trust store distribution requirements.



Cross-account certificate issuance is facilitated through AWS Resource Access Manager, enabling centralized CA management where a primary account shares CA access with secondary accounts through resource-based policies that permit specific IAM roles to issue, revoke, and export certificates without duplicating CA infrastructure in each account. AWS Certificate Manager integrates with Private CA to automate certificate lifecycle management, providing 13-month validity periods for managed certificates with automatic renewal, while certificates issued directly through Private CA APIs support custom validity periods tailored to specific workload requirements such as short-lived certificates for ephemeral containers. Organizations should

implement event-driven workflows using Amazon EventBridge to monitor ACM certificate lifecycle events including expiration warnings, available certificates, and renewal status, triggering automated responses through AWS Lambda functions that export and distribute certificates to non-ACM-integrated resources, publish findings to AWS Security Hub for centralized visibility, or send notifications through Amazon SNS to stakeholders requiring action. For comprehensive certificate monitoring, organizations can leverage Private CA audit reports generated through scheduled EventBridge rules that invoke Lambda functions to analyze certificate expirations, store reports in Amazon S3, and deliver proactive alerts before certificates expire, particularly critical for certificates issued outside of ACM that lack automatic tracking and renewal capabilities.

References:
https://docs.aws.amazon.com/kms/latest/developerguide/overview.html
https://docs.aws.amazon.com/prescriptive-guidance/latest/encryption-best-practices/kms.html
https://docs.aws.amazon.com/privateca/latest/userguide/PcaRegions.html
https://aws.amazon.com/blogs/security/automating-aws-private-ca-audit-reports-and-certificate-expiration-alerts/
https://aws.amazon.com/blogs/security/using-acm-private-certificate-authority-multi-account-environment-using-iam-roles/

Amazon FSx for Lustre is a fully managed, high-performance file system designed for compute-intensive and data-parallel workloads that require fast, shared storage. It provides sub-millisecond latencies, high throughput, and millions of IOPS—making it suitable for machine learning, analytics, media processing, and HPC workloads. FSx for Lustre integrates with Amazon S3, enabling large datasets to be accessed and processed in parallel from a Lustre file system. Security features include encryption at rest using AWS KMS, encryption in transit, integration with VPC security controls, and fine-grained access management through POSIX permissions and Amazon FSx access control mechanisms.

References:
https://docs.aws.amazon.com/fsx/latest/LustreGuide/what-is.html
https://docs.aws.amazon.com/fsx/latest/LustreGuide/security.html

Domain 6: Security Foundations and Governance

Overview

The 6th exam domain of the AWS Certified Security Specialty test focuses on the management and security governance of your AWS infrastructure. It represents approximately 14% of the overall exam. You should have the skill to design and implement controls that provide confidentiality and integrity for data in transit; design and implement controls that provide confidentiality and integrity for data at rest; design and implement controls to manage the lifecycle of data at rest as wel as to design and implement controls to protect credentials, secrets, and cryptographic key materials.

This domain will test your knowledge and skills on the following:

- Deploying and configuring AWS Organizations
- Determining when and how to deploy AWS Control Tower (for example, which services must be deactivated for successful deployment)
- Implementing SCPs as a technical solution to enforce a policy (for example, limitations on the use of a root account, implementation of guardrails in Control Tower)
- Centrally managing security services and aggregating findings (for example, by using delegated administration and AWS Config aggregators)
- Securing AWS account root user credentials
- Using CloudFormation to deploy cloud resources consistently and securely
- Implementing and enforcing multi-account tagging strategies
- Configuring and deploying portfolios of approved AWS services (for example, by using AWS Service Catalog)
- Organizing AWS resources into different groups for management
- Deploying Firewall Manager to enforce policies
- Securely sharing resources across AWS accounts (for example, by using AWS Resource Access Manager [AWS RAM])
- Identifying sensitive data by using Macie
- Creating AWS Config rules for detection of noncompliant AWS resources
- Collecting and organizing evidence by using Security Hub and AWS Audit Manager
- Identifying anomalies based on resource utilization and trends
- Identifying unused resources by using AWS services and tools (for example, AWS Trusted Advisor, AWS Cost Explorer)
- Using the AWS Well-Architected Tool to identify security gaps

Management and Governance in AWS

AWS provides a wide range of management and governance services that you can use for your organization. These services can help you provision, manage, govern, and operate your cloud environments more effectively. The time-consuming manual tasks such as account creation, license provisioning, configuration management, and the like can now be fully automated by using AWS.

To manage your cloud resources, you can use the AWS Management Console, AWS Command Line Interface, AWS Console Mobile Application, AWS Resource Access Manager, AWS Systems Manager, and many other services. You can also govern your cloud infrastructure using AWS Config, AWS Organizations, AWS Service Catalog, AWS Control Tower, and other services to enforce operational standards across all your AWS resources and comply with your corporate IT policies.

There are a couple of differences between managing and governing your cloud architecture. Both of them have a certain level of control over your AWS resources, but the latter is focused on implementing strict guardrails that your staff and underlying systems must follow.

In AWS, you can manage existing resources using a wide selection of web consoles, APIs, and management services. For example, you can start, stop, and hibernate an Amazon EC2 instance using the AWS Management Console or programmatically via the EC2 API. The scope of control is limited to the available commands that can be issued on your cloud resource. So in this situation, you can only control when the instance can be started, stopped, or put into hibernation.

In governance, you are expected to create policies that will be followed by your systems and AWS services. Any deviations from the policy would not be allowed. Let's say you want to implement a policy in which Amazon EC2 instances can only be launched from pre-approved AMIs. In this way, your organization can ensure that all of its applications are running on a tested machine image with no security vulnerabilities. No one should be able to launch an instance from an AMI that is not pre-approved by your platform team. This is similar to a real-world government where we as its citizens, are bound by its laws, decrees, city ordinances, et cetera.

As part of governance planning, organizations should also define breakglass procedures, which include emergency access accounts and workflows to be used only during critical incidents. These procedures ensure that breakglass accounts have limited, auditable privileges, are used strictly in emergencies, and that all actions are logged for compliance. Properly implemented breakglass procedures allow administrators to respond to urgent issues without violating standard access policies or compromising security controls.

Multi-Account Strategies in AWS

Each company has its own set of technical needs and compliance requirements as well as a unique setup that fits it organizational structure. A small startup with a handful of employees may only need a single AWS account while a multinational company with thousands of employees, several business units, and various regional headquarters around the world needs multiple AWS accounts to properly organize their enterprise architecture.

A company may have multiple teams and departments with different security and compliance controls that need to be isolated from each another. Others might have entirely different business processes or could be part of different business units that requires billing consolidation. Explicit security boundaries may also be needed, which provide a mechanism to have direct control and visibility of the company's account limits, billing and service quotas.

Organizations can use multiple AWS accounts to isolate and manage their enterprise applications and proprietary data with ease. This isolation of designated AWS resources can improve the security posture of the business while removing the manual overhead of managing the large number of AWS resources. Doing so can adhere to the best practices mentioned in the AWS Well-Architected Framework pillars such as operational excellence, security, reliability, and cost optimization. As part of a multi-account strategy, organizations should also define emergency "breakglass" accounts with highly restricted privileges. These accounts are used only during critical incidents, ensuring urgent access while maintaining auditability and compliance.

There are various management and governance services available in AWS that you can utilize in setting up your multi-account architecture. You can AWS Organizations, AWS Control Tower, AWS Service Catalog and many more.

AWS Organizations is a service in AWS that allows you to centrally govern your cloud environment as you grow and scale your AWS workloads. It can centrally provision both AWS accounts and AWS resources automatically – removing the hassle of manually doing this task yourself. AWS Organizations also allows companies to secure and audit their environment for compliance; share resources; control access to accounts, regions, and services. Consolidated Billing is also provided to simplify the cost management of your multiple AWS accounts. In addition, AWS Organizations also supports aggregation of health events, centralized management of backups, tagging for multi-account environments as well as consolidated data on use of access permissions.

AWS Control Tower is a service that helps you set up and govern a secure multi-account AWS environment. It automates the setup of your multi-account AWS environment with just a few clicks. The setup uses blueprints that follow AWS best practices for security and management. Control Tower provides mandatory high-level rules, called guardrails, that help enforce your policies using service control policies, or detect policy violations using AWS Config rules.

AWS Well-Architected Framework

A building is only as strong as its foundation. This fundamental principle is not just applicable to masonry but also to cloud architecture as well. There's a particular framework that can guide you in designing a secure, high-performing, resilient, and efficient architecture for your various applications and workloads in the AWS Cloud. This framework is called the AWS Well-Architected Framework.

What is the AWS Well-Architected Framework?

Basically, the AWS Well-Architected Framework is a body of knowledge that describes the key concepts, design principles, and architectural best practices to help you design and run efficient workloads in AWS. You can ensure that your cloud architecture aligns with the AWS best practices by answering several foundational questions provided by this framework. It also comes with related services and tools that you can use to measure the overall efficiency of your design. This will empower you to improve your existing IT infrastructure in terms of operations, security, reliability, efficiency, cost optimization, and sustainability.

The AWS Well-Architected Framework is categorized into several pillars, which reflect the various components of your architecture. Each pillar has its own roster of key topics that revolve around a specific subject matter. For every key topic, there's a list of common design patterns and anti-patterns for your architecture. A design pattern is a common blueprint of what's actually being used by thousands of companies to fulfill a particular use case more effectively, while an anti-pattern is its complete opposite which often produces entirely ineffective or subpar results. These design patterns and anti-patterns reflect the associated technical concepts, design principles, and strategies for that knowledge area.
The key topics in each pillar have a corresponding implementation guide that you can follow along. The risk level is also described if this recommendation is not established in your architecture. There's also a list of benefits included if you apply the recommended best practice on your own cloud infrastructure.

You'll discover a whole range of industry best practices in the AWS Well-Architected Framework. Essentially, a best practice is a proven method or technique that has been generally accepted as the best or the superior way to other known alternatives, as it often produces better results compared with other means. This distinct practice is considered the standard way of doing things in the status quo.

How does it Work?

So how does the AWS Well-Architected Framework improve the design of your cloud solutions?

Suppose you are developing an online solution that handles sensitive financial information. Your application has passed all the integration tests and is now ready for production deployment. However, you still want to verify that your cloud infrastructure is indeed secure as part of your corporate compliance.

You can check the security pillar of the AWS Well-Architected Framework that focuses on protecting your data, files, and overall systems. This includes key topics on data integrity, managing user permissions, and establishing controls to detect security incidents.

In essence, you can improve your cloud designs by simply answering the evaluation questions and following the best practices provided by this framework. These questions will shed light on your existing or new architecture in the AWS Cloud. It has questions like:

- "How do you protect your data at rest?"
- "How do you protect your data in transit?"
- "How do you manage identities for people and machines?"
- …and so on and so forth.

Your answer to these questions can show if your cloud architecture is secure or not. If you responded "I don't know" in the "How do you protect your data at rest?" question, then that means your architecture is not secure and has a high number of security vulnerabilities. This signifies that you don't employ encryption and tokenization schemes in your system.

The same goes for the "How do you protect your data in transit?" query. If you answer that you do not protect your data in transit, then that indicates your architecture has no firewall rules, network authentication, secure key management, and other mechanisms to keep your sensitive data safe as it traverses through different systems and networks. With this realization, you can now resolve the deficiencies in your system by following the prescriptive guidance provided by the AWS Well-Architected Framework.

Your team can start incorporating the various security best practices in your AWS workloads. So for 'protecting your data at rest', you will need to implement secure key management, enforce encryption at rest, automate data at rest protection, enforce access control, and use mechanisms to keep unauthorized people away from your data.

Conversely, you can protect your data in transit by authenticating network communications, automating the detection of unintended data access, implementing secure key and certificate management, enforcing

encryption in transit via SSL, plus other recommendations that are mentioned in the security pillar of AWS Well-Architected Framework.

And there you have it! We have successfully exposed the fatal flaws of your cloud architecture, like opening a can of worms by merely asking the right architectural questions. Your designs will truly be well-architected because you adhere to the industry-standard best practices of the AWS Well-Architected Framework, which is quite time-consuming to do if you just blindly audit your own systems yourself.

AWS Well-Architected Tool (WA Tool)

As mentioned previously, the AWS Well-Architected Framework is simply a body of knowledge that describes the key concepts, design principles, and architectural best practices for designing and running efficient workloads in AWS. This framework is actually just a document in its raw form which means you have to manually do the architectural checks on your cloud architecture yourself to determine if you are using the recommended AWS best practices.

Suppose you want to measure the security compliance of your AWS infrastructure. You have no choice but to read the entire Security Pillar section of the AWS Well-Architected Framework and manually inspect your cloud stack to see if the design principles are incorporated or not. It's very tedious to track all your components one by one if they conform to the best practices recommended by AWS. This task is quite difficult to accomplish and consumes much of your time. The combination of a steep learning curve and management overhead makes it unsustainable in the long run, which is why most companies give up on adopting this framework.

The good thing is that AWS provides a way to easily incorporate the design principles and best practices at your fingertips. This is possible through the AWS Well-Architected Tool or WA Tool for short. The AWS Well-Architected Tool is nothing but a self-service console that you can use to check your cloud architectures against the AWS Well-Architected Framework. It also is available at absolutely no charge in the AWS Management Console, so you can use this as often as you want.

This tool is designed to help you review the state of your applications and cloud workloads against architectural best practices in the AWS Cloud. It also helps you to easily identify opportunities for improvement and track your progress over time. The AWS WA Tool can be integrated with other AWS services, such as the AWS Trusted Advisor, AWS Compute Optimizer, and AWS Service Catalog, among others. For example, you can use the AppRegistry module of the AWS Service Catalog to register your custom application and its associated resource collection. This application can be connected to the Well-Architected Tool via the AppRegistry for easier workload discovery. Since the task of discovering the different components of your workload is automated, you will be able to save a lot of time since you won't have to manually search your resources as part of assessing your architecture. You can even enable the AWS Trusted Advisor to simplify your workload reviews by providing automated context for supported questions.

Furthermore, the AWS Well-Architected Tool provides APIs that you can use to extend the AWS Well-Architected functionality into your existing applications, architecture governance processes, and workflows. These APIs provide programmatic access to the AWS Well-Architected Tool without using the AWS Management Console. You can create custom programs using these APIs to fetch the workloads, best practices, and measurements programmatically. The AWS Command Line Interface, or the AWS CLI, also has available commands that you use to invoke the Well-Architected Tool APIs.

There are three primary steps in using the AWS Well-Architected Tool. The first step is to define your workload, and second, conduct an architectural review of your cloud systems. The last step is to apply the best practices and improvement plans that are identified in the review.

Hardening CloudFormation templates

When writing CloudFormation templates, it's a security best practice to avoid hardcoding sensitive info, like client secrets, API keys, or passwords. Sharing templates with embedded credentials by mistake can put your infrastructure and data at risk.

Passing Parameters Securely

Using dynamic references

The most recommended way of passing sensitive parameters to a CloudFormation template is through Dynamic references. Instead of specifying sensitive info during stack creation, you may opt to store them first in AWS Secrets Manager or Systems Manager Parameter Store. Then you can use a dynamic reference so CloudFormation can retrieve and resolve them at runtime.

A dynamic reference follows a simple syntax - `{{resolve:service:reference-key}}`, where the service denotes the AWS service from which the secret is being fetched, and `reference-key` identifies the specific secret.

CloudFormation supports the following reference key names:

1. `ssm` (SSM Parameter Store plaintext parameter)
2. `ssm-secure` (SSM Parameter Store secure string parameter)
3. `secretsmanager` (Secrets Manager secret)

Suppose you are building a stack for an application that is made up of a Lambda function and an RDS database. To protect the database credentials from being exposed, you shouldn't store them in plain text in the

template, function code, or the function's environment variables. Instead, you can store the credentials as a secret in AWS Secrets Manager and then reference it in your template via the built-in `secretsmanager` reference key.

```
MyRDSInstance:
    Type: 'AWS::RDS::DBInstance'
    Properties:
      DBName: MyRDSInstance
      AllocatedStorage: '20'
      DBInstanceClass: db.t2.micro
      Engine: mysql
      MasterUsername: '{{resolve:secretsmanager:DBcreds:SecretString:username}}'
      MasterUserPassword: '{{resolve:secretsmanager:DBcreds:SecretString:password}}'
```

You can pass the secret name (in this example, "DBcreds") as an environment variable to the Lambda function.

```
AWSTemplateFormatVersion: '2010-09-09'
Resources:
  MyLambdaFunction:
    Type: 'AWS::Lambda::Function'
    Properties:
      Code:
        S3Bucket: name-of-your-s3-bucket
        S3Key: my-lambda-function.zip
      Handler: index.handler
      Role: !GetAtt MyLambdaExecutionRole.Arn
      Runtime: python3.9
      Environment:
        Variables:
          SECRET_NAME: 'DBCreds'
```

The Lambda function can then retrieve the secret value from AWS Secrets Manager at runtime using the `GetSecretValue` API.

```
import json
import boto3
import os
```

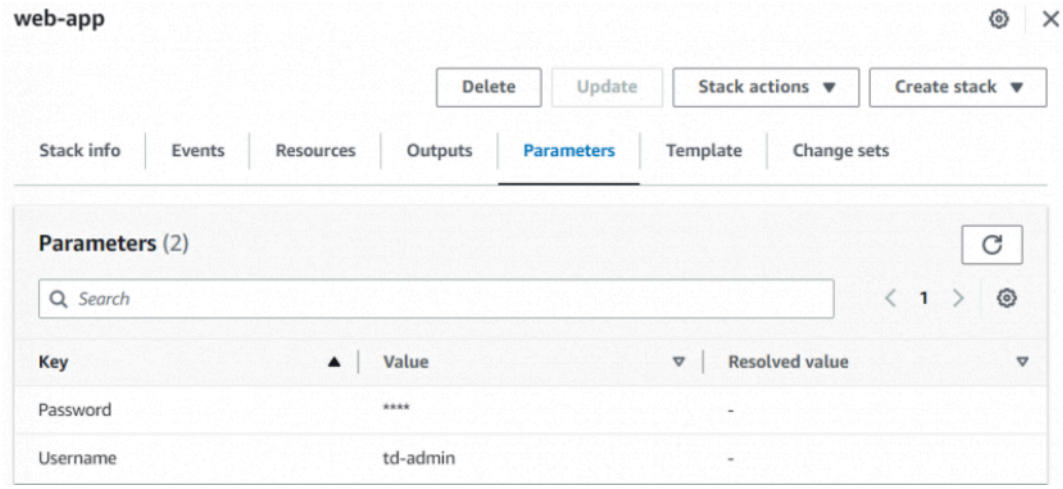
```
def lambda_handler(event, context):
    # Get the name of the secret from an environment variable
    secret_name = os.environ['SECRET_NAME']

    # Create a Secrets Manager client
    secretsmanager = boto3.client('secretsmanager')

    # Get the username and password from the secret
    secret_value = secretsmanager.get_secret_value(SecretId=secret_name)
    secret_dict = json.loads(secret_value['SecretString'])
    username = secret_dict['username']
    password = secret_dict['password']
```

Using the NoEcho Attribute

There could be certain scenarios where you prefer to pass credentials using parameters. In these cases, using the `NoEcho` property for parameters in CloudFormation templates can mask the sensitive values in the console.

```
Parameters:
  MyParameter:
    Type: String
    NoEcho: true
Resources:
# ...
```

When you set the `NoEcho` attribute to true, CloudFormation will redact the parameter value by displaying asterisks (***) in the Console or any API responses.



While `NoEcho` hides the values in the console output and CloudFormation events, other methods might still expose them. For example, if these credentials are used as environment variables in a Lambda function (let's say lambda), any user who has permissions to read the function configuration will be able to view these environment variables, potentially exposing sensitive information. Hence, it is essential to apply another layer of security by restricting IAM permissions. AWS provides fine-grained access control to ensure that only authorized personnel have the necessary access to Lambda function configurations. By setting stringent IAM policies, you can restrict who can view the environment variables of your Lambda functions.

AWS CloudFormation Guard

AWS CloudFormation Guard is an open-source policy-as-code evaluation tool that provides a declarative domain-specific language for defining and enforcing compliance policies across infrastructure deployments. Guard validates CloudFormation templates, Terraform JSON configurations, Kubernetes manifests, and AWS Config configuration items against custom policy rules, enabling organizations to enforce encryption requirements, tagging standards, network isolation policies, and IAM least privilege configurations before resources are provisioned. The Guard Rules Registry provides pre-built managed rule sets implementing AWS Config Managed Rules and compliance frameworks including CIS AWS Foundations Benchmark, PCI-DSS, HIPAA, and NIST, allowing rapid adoption of industry-standard security controls. Guard rules use queries with filter expressions and validation through operators, regular expressions, and named-rule blocks that enable reuse across complex policy hierarchies.

Integration of CloudFormation Guard into CI/CD pipelines enables shift-left security by validating infrastructure templates before deployment. Guard CLI integrates with AWS CodePipeline through CodeBuild projects that execute validation commands against templates and fail builds when policy violations are detected. Organizations implement Guard validation at multiple stages: developer workstations using IDE plugins or pre-commit hooks, pull request validation through GitHub Actions that block merges of non-compliant

templates, central CI/CD pipelines performing comprehensive policy checks before deployment authorization, and detective controls using AWS Config for continuous compliance monitoring. Multi-account deployment strategies leverage CloudFormation StackSets to distribute standardized security policies across organizational units with Guard validation ensuring consistent compliance enforcement before stack instances are created in target accounts.

CloudFormation Linter (cfn-lint) serves as a static analysis tool that validates CloudFormation templates against the AWS CloudFormation Registry Resource Provider Schemas, providing comprehensive syntax checking and best practice enforcement before deployment. The v1 release migrated from CloudFormation specifications to registry schemas, incorporating data from AWS pricing APIs and botocore to enhance validation accuracy across resource types and their region-specific configurations. With over 263 validation rules, cfn-lint examines intrinsic functions, parameter constraints, IAM policy syntax, resource dependencies, and property relationships to identify configuration errors that would otherwise result in stack deployment failures. Organizations integrate cfn-lint into development workflows through IDE extensions for real-time feedback, pre-commit hooks that prevent invalid templates from entering version control, CI/CD pipeline stages that gate deployments based on linting results, and AWS Systems Manager automation runbooks for centralized validation across multi-account environments.

CloudFormation Hooks extend Guard capabilities from client-side validation to server-side enforcement, providing proactive controls that automatically evaluate resources during CloudFormation operations and AWS Cloud Control API operations. Guard Hooks download rules from Amazon S3, evaluate resource configurations, and either block non-compliant operations in FAIL mode or issue warnings in WARN mode, with support for stack-level, resource-level, and change set operations. Organizations activate Guard Hooks using the `AWS::CloudFormation::GuardHook` resource, configuring execution roles, target operations, resource type filters, and stack filters to control when validation occurs. Best practices include versioning Guard rules in S3, implementing graduated enforcement starting with WARN mode before switching to FAIL mode, maintaining environment-specific rule sets with stricter controls in production, and integrating hook logs with AWS Security Hub for centralized compliance visibility.

References:
https://docs.aws.amazon.com/cfn-guard/latest/ug/what-is-guard.html
https://github.com/aws-cloudformation
https://aws.amazon.com/blogs/devops/
https://docs.aws.amazon.com/systems-manager-automation-runbooks/latest/userguide/automation-aws-runcfnlint.html

AWS CHEAT SHEETS

AWS Analytics Services

Amazon Athena

- A serverless, interactive query service that analyzes data directly in Amazon S3 using standard SQL.
- Pay only for the queries you run – no infrastructure to manage.
- Commonly used for analyzing CloudTrail logs, VPC Flow Logs, Security Hub findings, and Security Lake data.

Features

- Query data directly in S3 without loading it into a database.
- Supports CSV, JSON, Parquet, ORC, and Avro formats.
- Integrates with AWS Glue Data Catalog for schema management.
- Results stored in S3 bucket you specify.
- Supports partitioning to improve performance and reduce costs.
- Built on Presto distributed SQL engine.

Concepts

- **Databases and Tables**
  - Tables define schema for data in S3.
  - Schema stored in AWS Glue Data Catalog or internal Athena catalog.
  - Tables created manually or automatically using Glue crawlers.
- **Partitioning**
  - Divides data based on column values (e.g., date, region, account).
  - Improves query performance by scanning only relevant data.
  - Common patterns: year/month/day or account/region/service.
- **Workgroups**
  - Separate users, teams, or workloads.
  - Each workgroup has separate query result locations and cost controls.
- **Query Results**
  - Stored in specified S3 bucket in CSV format.
  - Retained for 45 days by default.
  - Can be encrypted using SSE-S3 or SSE-KMS.

Common Security Use Cases

- Analyzing CloudTrail logs for API activity and unauthorized access.

- Analyzing VPC Flow Logs for network traffic patterns and suspicious connections.
- Querying Security Hub findings by severity and compliance status.
- Querying Security Lake OCSF-formatted data for threat hunting.
- Analyzing ALB and WAF logs for attack patterns.

Amazon Athena Integration

- **AWS Glue Data Catalog** – Centralized metadata repository shared across analytics services.
- **AWS CloudTrail** – Direct query of CloudTrail logs; auto-create tables from CloudTrail console.
- **Amazon Security Lake** – Primary query engine for Security Lake OCSF data.
- **AWS Lake Formation** – Fine-grained access control with column/row-level security.

Amazon Athena Security

- Uses IAM policies to control query execution and data access.
- Query results encrypted using SSE-S3 or SSE-KMS.
- All communication encrypted in transit using TLS.
- Integrates with Lake Formation for fine-grained permissions.
- VPC endpoints available for private access via AWS PrivateLink.
- Required IAM permissions:
  - `athena:StartQueryExecution` – Run queries
  - `athena:GetQueryResults` – Retrieve results
  - `s3:GetObject` – Read source data
  - `s3:PutObject` – Write query results
  - `glue:GetTable` – Access metadata

Query Optimization Best Practices

- Use columnar formats (Parquet, ORC) for 30-90% cost reduction.
- Partition data by commonly filtered columns.
- Compress data using Snappy, GZIP, or ZSTD.
- Optimize file sizes to 128 MB - 1 GB after compression.
- Include partition filters in WHERE clauses.
- Use CTAS (CREATE TABLE AS SELECT) to create optimized tables.

Amazon Athena Pricing

- Charged based on amount of data scanned per query.
- $5.00 per TB of data scanned (standard pricing).
- No charges for DDL statements or failed queries.
- Query result storage charged at standard S3 rates.
- Columnar formats and partitioning reduce costs significantly.

References:
https://docs.aws.amazon.com/athena/latest/ug/what-is.html
https://docs.aws.amazon.com/athena/latest/ug/security.html https://aws.amazon.com/athena/pricing/

Amazon OpenSearch Service

- A managed service for deploying, operating, and scaling OpenSearch clusters for search, log analytics, and real-time application monitoring.
- Fork of Elasticsearch and Kibana, offering both OpenSearch and legacy Elasticsearch versions.
- Commonly used for security log analysis, SIEM implementations, and real-time threat detection.

Features

- Full-text search with relevance scoring and filtering.
- Real-time data ingestion and near real-time search.
- OpenSearch Dashboards (formerly Kibana) for visualization and exploration.
- Alerting and anomaly detection for security monitoring.
- SQL query support for familiar query syntax.
- Machine learning features for anomaly detection and forecasting.
- Built-in integration with AWS services for log ingestion.
- Multi-AZ deployment with automated snapshots to S3.

Concepts

- **Clusters and Nodes**
  - Cluster consists of one or more nodes (EC2 instances).
  - Node types: data nodes, master nodes, UltraWarm nodes, cold storage nodes.
  - Master nodes manage cluster state; data nodes store data and handle queries.
- **Indices and Documents**
  - Index is a collection of documents (similar to database table).
  - Documents are JSON objects stored and searchable.
  - Index patterns used to search across multiple indices.
- **Shards and Replicas**
  - Primary shards divide index data across nodes for scalability.
  - Replica shards provide redundancy and improve read performance.
  - Recommended: 1 replica per primary shard for high availability.
- **Domains**
  - An OpenSearch domain is a cluster with settings, access policies, and network configuration.
  - Domain endpoint used to interact with cluster.
- **Storage Tiers**

    - **Hot storage** – Standard data nodes for active, frequently accessed data.
    - **UltraWarm** – S3-backed storage for read-only data, up to 90% cost reduction.
    - **Cold storage** – S3-based storage for infrequently accessed data, lowest cost.
- **Snapshots**
    - Automated daily snapshots stored in S3 (retained for 14 days).
    - Manual snapshots can be taken and retained indefinitely.
    - Snapshots used for backup, recovery, and cluster migration.

Common Security Use Cases

- Centralized log aggregation from CloudTrail, VPC Flow Logs, WAF, and GuardDuty.
- Real-time security event monitoring and alerting.
- SIEM (Security Information and Event Management) implementation.
- Threat hunting and incident investigation.
- Compliance reporting and audit trail analysis.
- Anomaly detection using built-in machine learning features.
- Visualization of security metrics and KPIs in OpenSearch Dashboards.

Amazon OpenSearch Integration

- **AWS Lambda** – Process and ingest data from various sources into OpenSearch.
- **Amazon Kinesis Data Firehose** – Stream logs directly to OpenSearch with transformation.
- **AWS CloudWatch Logs** – Stream CloudWatch log groups to OpenSearch for analysis.
- **Amazon S3** – Load data from S3 using Lambda or manual ingestion.
- **Amazon Security Lake** – Query Security Lake data and visualize in OpenSearch.
- **AWS IoT Core** – Ingest IoT device data for real-time analytics.
- **Amazon Cognito** – User authentication for OpenSearch Dashboards.
- **AWS CloudTrail** – Ingest API activity logs for security auditing.

Amazon OpenSearch Security

- **Network Security**
    - VPC deployment for private access (recommended for security workloads).
    - Public endpoint with IP-based access policies (less secure).
    - Security groups control network access to domain.
- **Encryption**
    - Encryption at rest using AWS KMS keys (required for sensitive data).
    - Encryption in transit (node-to-node and client-to-node) using TLS.
    - Must be enabled at domain creation – cannot be enabled later.
- **Access Control**
    - **IAM-based access** – Uses IAM policies for API-level permissions.
    - **Fine-grained access control (FGAC)** – Index, document, and field-level permissions.

    - **SAML authentication** – Integrate with enterprise identity providers.
    - **Amazon Cognito** – User authentication for OpenSearch Dashboards.
    - Resource-based policies for domain-level access control.
- **Audit Logging**
    - Logs authentication attempts, index operations, and search queries.
    - Audit logs published to CloudWatch Logs for monitoring.
    - Helps track who accessed what data and when.
- **Required IAM Permissions**
    - `es:ESHttpGet` – Read data from indices
    - `es:ESHttpPost` – Write data to indices
    - `es:ESHttpPut` – Create/update indices and documents
    - `es:ESHttpDelete` – Delete indices and documents

Alerting and Monitoring

- Create monitors to watch for specific conditions in data.
- Triggers send notifications via SNS, Slack, webhook, or custom destination.
- Common security alerts:
    - High rate of failed authentication attempts
    - Critical GuardDuty findings
    - Unusual API activity patterns
    - Access from suspicious IP addresses
- Anomaly detection identifies unusual patterns without defining thresholds.
- Integration with CloudWatch for cluster health metrics.

Best Practices

- Deploy in VPC for production security workloads.
- Enable encryption at rest and in transit.
- Use fine-grained access control for multi-tenant environments.
- Implement dedicated master nodes for cluster stability (3 recommended).
- Use UltraWarm for older, less frequently accessed logs.
- Configure automated snapshots to S3 for disaster recovery.
- Monitor cluster health and set CloudWatch alarms.
- Right-size instance types based on workload characteristics.

Amazon OpenSearch Service Pricing

- Charged for instance hours (by instance type and size).
- Charged for EBS storage attached to data nodes.
- UltraWarm storage charged at lower S3-based rate.
- Cold storage charged at S3 rates.

- Data transfer charges apply for data leaving AWS.
- No charge for data transfer within same AZ.
- Snapshot storage charged at standard S3 rates.

References:
https://docs.aws.amazon.com/opensearch-service/latest/developerguide/what-is.html
https://docs.aws.amazon.com/opensearch-service/latest/developerguide/security.html
https://aws.amazon.com/opensearch-service/pricing/

AWS Application Integration Services

Amazon Simple Notification Service (Amazon SNS)

- A fully managed pub/sub messaging service for application-to-application (A2A) and application-to-person (A2P) communication.
- Enables fanout messaging to multiple subscribers simultaneously.
- Commonly used for security alerting, automated notifications, and distributing security findings to multiple destinations.

Features

- Message fanout to multiple subscribers (SQS, Lambda, HTTP/S, email, SMS, mobile push).
- Message filtering to deliver subset of messages to specific subscribers.
- Message encryption at rest and in transit.
- Dead-letter queues (DLQ) for failed message deliveries.
- FIFO topics for ordered message delivery.
- Message archiving and replay using Kinesis Data Firehose.
- Cross-region message delivery.
- Server-side encryption using AWS KMS.

Concepts

- **Topics**
  - Communication channel where publishers send messages.
  - Two types: Standard (best-effort ordering, at-least-once delivery) and FIFO (strict ordering, exactly-once delivery).
  - Topic names must be unique within an AWS account and region.
- **Publishers**
  - Applications, services, or users that send messages to topics.

    - Common AWS publishers: CloudWatch Alarms, EventBridge, Lambda, S3, Security Hub, GuardDuty.
- **Subscriptions**
    - Endpoint that receives messages published to a topic.
    - Subscription types: SQS, Lambda, HTTP/S, Email, Email-JSON, SMS, mobile push notifications.
    - Subscriptions must be confirmed before receiving messages (except SQS and Lambda).
- **Message Filtering**
    - Subscribers receive only messages matching their filter policy.
    - Reduces unnecessary processing and costs.
    - Filter based on message attributes or message body content.
- **Message Attributes**
    - Metadata sent with messages (key-value pairs).
    - Used for filtering and routing messages.
    - Examples: severity level, event type, resource ID.

Common Security Use Cases

- Alerting security teams on high-severity GuardDuty findings.
- Notifying on Security Hub compliance failures.
- Sending notifications for CloudWatch security metric alarms.
- Distributing security events to multiple SIEM systems simultaneously.
- Alerting on AWS Config rule violations.
- Notifying on IAM policy changes or unauthorized access attempts.
- Sending incident response notifications to on-call teams.
- Triggering automated remediation workflows via Lambda subscriptions.

Amazon SNS Integration

- **Amazon EventBridge** – Route events from AWS services to SNS topics.
- **AWS Lambda** – Trigger Lambda functions for automated response.
- **Amazon SQS** – Fanout pattern: one message to multiple SQS queues.
- **AWS CloudWatch** – Send alarm notifications to SNS.
- **AWS Security Hub** – Forward findings to SNS for alerting.
- **Amazon GuardDuty** – Send threat detection alerts via EventBridge to SNS.
- **AWS Config** – Notify on configuration changes and compliance violations.
- **Amazon Macie** – Alert on sensitive data findings.
- **AWS CloudFormation** – Send stack event notifications.
- **Amazon S3** – Trigger notifications on object events.

Amazon SNS Security

- **Encryption at Rest**

    - Server-side encryption (SSE) using AWS KMS keys.
    - Messages encrypted before being written to disk.
    - Transparent decryption when delivered to subscribers.
- **Encryption in Transit**
    - All API calls encrypted using TLS.
    - HTTPS endpoints required for HTTP/S subscriptions.
- **Access Control**
    - **IAM policies** – Control who can publish to topics and manage subscriptions.
    - **Topic policies** – Resource-based policies defining who can access the topic.
    - **Subscription policies** – Control which topics can send to specific endpoints.
    - Cross-account access supported via topic policies.
- **Message Data Protection**
    - Scan and mask sensitive data (PII, credentials) in messages.
    - Audit data in transit to detect compliance violations.
    - Redact or block messages containing sensitive data.
    - Uses pattern matching to identify sensitive data types.
- **VPC Endpoints**
    - Access SNS privately without internet gateway using AWS PrivateLink.
    - Keep traffic within AWS network.
- **Required IAM Permissions**
    - `sns:Publish` – Send messages to topics
    - `sns:Subscribe` – Create subscriptions
    - `sns:CreateTopic` – Create new topics
    - `sns:SetTopicAttributes` – Modify topic settings

Message Delivery and Retry

- Standard topics: best-effort ordering, at-least-once delivery.
- FIFO topics: strict ordering, exactly-once delivery (limited to SQS FIFO subscriptions).
- Automatic retries for failed deliveries with exponential backoff.
- Dead-letter queues (DLQ) capture messages that fail all retry attempts.
- Delivery status logging tracks successful and failed deliveries.
- Delivery protocols: SQS (most reliable), Lambda, HTTP/S, Email, SMS.

SNS Fanout Pattern

- One SNS message distributed to multiple SQS queues or Lambda functions simultaneously.
- Enables parallel processing of same event by different systems.
- Common pattern: Security finding → SNS → [SIEM queue, ticketing queue, remediation Lambda].
- Provides loose coupling between producers and consumers.

Best Practices

- Use message filtering to reduce unnecessary deliveries.
- Enable encryption at rest for sensitive security data.
- Implement dead-letter queues to capture failed deliveries.
- Use FIFO topics when message ordering is critical.
- Configure delivery status logging for monitoring.
- Set appropriate retry policies for HTTP/S endpoints.
- Use SNS + SQS fanout for reliable, parallel processing.
- Apply least privilege IAM and topic policies.
- Enable message data protection for PII scanning.

Amazon SNS Pricing

- Charged per published message (first 1 million requests/month free tier).
- Different pricing for different protocols:
    - SNS to SQS/Lambda: lowest cost
    - Email/Email-JSON: moderate cost
    - SMS: highest cost (varies by destination country)
- Message filtering incurs no additional cost.
- Data transfer charges apply for messages leaving AWS.
- KMS encryption incurs standard KMS API usage charges.
- No charge for message delivery failures.

References:
https://docs.aws.amazon.com/sns/latest/dg/welcome.html
https://docs.aws.amazon.com/sns/latest/dg/sns-security.html https://aws.amazon.com/sns/pricing/

AWS Step Functions

- A serverless orchestration service that lets you coordinate multiple AWS services into serverless workflows using visual workflows.
- Define workflows as state machines using Amazon States Language (JSON-based).
- Commonly used for automating security incident response, orchestrating compliance remediation, and coordinating multi-step security investigations.

Features

- Visual workflow designer for building and monitoring workflows.
- Built-in error handling, retry logic, and compensation workflows.
- Parallel execution of multiple tasks simultaneously.

- Integration with 220+ AWS services and 10,000+ API actions.
- Execution history and logging for audit and debugging.
- Two workflow types: Standard (long-running, exactly-once execution) and Express (high-volume, at-least-once execution).
- Event-driven execution via EventBridge, API Gateway, or SDK.
- Built-in wait states for human approvals or time delays.

Concepts

- **State Machine**
  - Workflow definition written in Amazon States Language (ASL).
  - Collection of states that perform tasks, make decisions, or wait.
  - Two types: Standard (up to 1 year execution) and Express (up to 5 minutes).
- **States**
  - **Task** – Perform work (invoke Lambda, call API, run ECS task).
  - **Choice** – Conditional branching based on input.
  - **Parallel** – Execute multiple branches simultaneously.
  - **Wait** – Delay execution for specified time.
  - **Pass** – Pass input to output, optionally transforming data.
  - **Succeed/Fail** – Terminal states marking completion.
  - **Map** – Iterate over array items, processing each.
- **Executions**
  - Instance of a state machine running with specific input.
  - Each execution has unique execution ARN.
  - Execution history tracks state transitions and outputs.
- **Standard vs Express Workflows**
  - **Standard**: Exactly-once execution, up to 1 year duration, full execution history, higher cost.
  - **Express**: At-least-once execution, up to 5 minutes, limited history, lower cost (14x cheaper).

Common Security Use Cases

- **Automated Incident Response**
  - GuardDuty finding → EventBridge → Step Functions → [Isolate instance → Snapshot EBS → Notify team → Create ticket].
- **Multi-Step Remediation**
  - Security Hub finding → Step Functions → [Verify violation → Apply fix → Validate → Update finding status].
- **Security Orchestration (SOAR)**
  - Coordinate multiple security tools and actions in response to threats.
  - Implement playbooks for common security scenarios.
- **Compliance Remediation**

    - Config rule violation → Step Functions → [Assess impact → Get approval → Remediate → Verify compliance].
- **Forensic Data Collection**
    - Suspicious activity → Step Functions → [Capture memory → Snapshot volumes → Preserve logs → Package evidence].
- **Credential Rotation**
    - Scheduled event → Step Functions → [Rotate secrets → Update applications → Validate → Notify].
- **Security Assessment Workflows**
    - Trigger scans → Aggregate results → Prioritize findings → Generate reports → Distribute.

AWS Step Functions Integration

- **AWS Lambda** – Execute custom code for security logic.
- **Amazon EventBridge** – Trigger workflows from AWS service events.
- **AWS Security Hub** – Update finding workflow status and add notes.
- **AWS Systems Manager** – Execute automation documents and runbooks.
- **Amazon SNS** – Send notifications at workflow stages.
- **AWS Secrets Manager** – Rotate credentials as part of workflow.
- **Amazon ECS/EKS** – Run containerized security tools.
- **AWS Glue** – Run data processing jobs for security analytics.
- **Amazon Athena** – Execute queries for investigation.
- **AWS IAM** – Assume roles for cross-account operations.
- **DynamoDB** – Store workflow state and metadata.

AWS Step Functions Security

- **Encryption**
    - Data encrypted at rest using AWS-owned keys or customer-managed KMS keys.
    - Data in transit encrypted using TLS.
    - Sensitive data in state machine definitions should be stored in Secrets Manager.
- **Access Control**
    - IAM policies control who can create, execute, and manage state machines.
    - State machines assume IAM roles to access AWS services.
    - Resource-based policies for cross-account execution.
    - Use least privilege for state machine execution roles.
- **Audit and Compliance**
    - All API calls logged in CloudTrail.
    - Execution history provides audit trail of workflow steps.
    - CloudWatch Logs integration for detailed logging.
    - X-Ray integration for distributed tracing.
- **VPC Integration**

    - Express workflows can access resources in VPC.
    - Use VPC endpoints for private service access.
- **Required IAM Permissions**
    - `states:StartExecution` – Start workflow executions
    - `states:DescribeExecution` – View execution details
    - `states:StopExecution` – Stop running executions
    - State machine execution role needs permissions for invoked services

Error Handling and Retry

- **Retry** – Automatically retry failed states with exponential backoff.
    - Configure max attempts, backoff rate, and interval.
    - Specific error matching (e.g., `States.Timeout`, `Lambda.ServiceException`).
- **Catch** – Handle errors and transition to error handling states.
    - Implement compensation logic or alternative workflows.
    - Add context about failure to output.
- **Timeouts** – Set maximum execution time for tasks and workflows.
- **Fallback** – Define fallback states for graceful degradation.

Workflow Patterns for Security

- **Sequential Processing** – Execute steps in order (investigate → contain → eradicate → recover).
- **Parallel Processing** – Perform multiple actions simultaneously (snapshot volume + capture logs + notify team).
- **Human Approval** – Wait for approval before proceeding with high-risk actions.
- **Conditional Branching** – Different paths based on severity, environment, or resource type.
- **Error Recovery** – Automatic retry with fallback to manual intervention.
- **Scheduled Execution** – EventBridge schedule triggers routine security tasks.

Monitoring and Observability

- CloudWatch metrics for execution success/failure rates.
- CloudWatch Logs for detailed execution logging.
- EventBridge events for execution status changes.
- X-Ray tracing for performance analysis.
- Execution history provides complete audit trail.
- Set CloudWatch alarms for failed executions.

Best Practices

- Use Standard workflows for mission-critical security responses.
- Use Express workflows for high-volume, short-duration tasks.
- Implement error handling and retry logic for all tasks.

- Store sensitive data in Secrets Manager, not state definitions.
- Use human approval tasks for high-impact remediation.
- Design idempotent workflows that can safely retry.
- Apply least privilege to execution role permissions.
- Enable CloudWatch Logs for debugging and audit.
- Use Map state for processing multiple resources.
- Implement timeouts to prevent indefinite execution.

AWS Step Functions Pricing

- **Standard Workflows**
  - Charged per state transition ($0.025 per 1,000 transitions).
  - First 4,000 state transitions per month free tier.
  - Includes all execution history and logging.
- **Express Workflows**
  - Charged by number of executions and duration.
  - Approximately 14x cheaper than Standard for high-volume workloads.
  - $1.00 per 1 million requests + duration charges.
- Additional charges for invoked services (Lambda, SNS, etc.).
- No charge for wait states.

References:

https://docs.aws.amazon.com/step-functions/latest/dg/welcome.html
https://docs.aws.amazon.com/step-functions/latest/dg/security.html
https://aws.amazon.com/step-functions/pricing/

AWS Compute Services

Amazon API Gateway

- A fully managed service for creating, publishing, maintaining, monitoring, and securing REST, HTTP, and WebSocket APIs at any scale.
- Acts as a "front door" for applications to access data, business logic, or functionality from backend services.
- Commonly used to expose Lambda functions, EC2 instances, or other AWS services as RESTful APIs.

Features

- Support for REST APIs, HTTP APIs, and WebSocket APIs.
- Request/response transformation and validation.
- API throttling and quota management per API key or stage.
- Integration with AWS WAF for protection against common web exploits.

- Built-in DDoS protection via AWS Shield.
- API caching to reduce backend load and improve latency.
- Custom domain names with SSL/TLS certificates from ACM.
- API versioning through stages (dev, staging, prod).

API Types

- **REST API**
  - Feature-rich API type with full request/response transformation.
  - Supports API keys, usage plans, and request validation.
  - Higher latency and cost compared to HTTP API.
- **HTTP API**
  - Lower latency and cost (up to 71% cheaper than REST API).
  - Simplified feature set focused on proxying to backends.
  - Built-in OIDC and OAuth 2.0 authorization.
- **WebSocket API**
  - Two-way real-time communication for chat applications, streaming, and IoT.
  - Maintains persistent connections between clients and backend.

Security Features

- **Authentication and Authorization**
  - AWS IAM roles and policies for API access control.
  - Amazon Cognito User Pools for user authentication.
  - Lambda authorizers (custom authorizers) for token-based auth.
  - API keys for simple identification (not for authorization).
- **Encryption**
  - TLS 1.2 encryption for data in transit.
  - Custom domain names with ACM certificates.
  - End-to-end encryption when integrated with encrypted backends.
- **Access Control**
  - Resource policies to control API access by IP, VPC, or AWS account.
  - CORS configuration for browser-based applications.
  - Private APIs accessible only from VPC via VPC endpoints (PrivateLink).
- **Throttling and Quotas**
  - Default account-level limit: 10,000 requests per second.
  - Per-method throttling to protect specific endpoints.
  - Usage plans with API keys to set quotas per customer.
  - Burst capacity to handle traffic spikes.
- **Logging and Monitoring**
  - CloudWatch Logs for execution logs and access logs.
  - CloudWatch metrics for monitoring API performance.

  - AWS X-Ray for distributed tracing and debugging.
  - CloudTrail for API management activity auditing.

Request Validation

- Validate request body against JSON schema before reaching backend.
- Validate request parameters (headers, query strings, path parameters).
- Return 400 Bad Request for invalid requests, reducing backend load.
- Prevents malformed requests from reaching Lambda or EC2.

Integration with AWS WAF

- Attach WAF Web ACLs to REST APIs to protect against:
  - SQL injection attacks.
  - Cross-site scripting (XSS).
  - IP-based blocking or whitelisting.
  - Rate limiting beyond API Gateway's built-in throttling.
  - Geo-blocking to restrict access by country.

References:
https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html
https://docs.aws.amazon.com/apigateway/latest/developerguide/security.html

Amazon Elastic Compute Cloud (Amazon EC2)

- Provides secure, resizable compute capacity in the cloud as virtual servers (instances).
- Offers complete control over the virtual computing environment including OS, network, and storage.
- Foundation for many AWS services and customer applications.

Security Features

- **Instance Security**
  - Security groups act as virtual firewalls (stateful, allow rules only).
  - Network ACLs provide subnet-level traffic control (stateless, allow and deny rules).
  - IAM roles for EC2 provide temporary credentials for AWS API access.
  - Instance metadata service (IMDS) v2 provides secure access to instance metadata.
  - Nitro System provides hardware-based security and isolation.
- **Encryption**
  - EBS encryption using AWS KMS keys (transparent encryption at rest).
  - In-transit encryption between instance and EBS volumes (Nitro instances).
  - Instance store volumes encrypted on Nitro instances.

- **Access Control**
  - Key pairs (SSH) for Linux instance access.
  - RDP with Windows password or key pair for Windows instances.
  - EC2 Instance Connect for browser-based SSH (temporary keys).
  - Systems Manager Session Manager for shell access without SSH/RDP.
- **Monitoring and Logging**
  - CloudWatch metrics for instance monitoring (CPU, network, disk).
  - VPC Flow Logs for network traffic analysis.
  - CloudTrail for API activity logging.
  - Systems Manager for inventory, patching, and compliance.

IMDSv2 (Instance Metadata Service Version 2)

- Requires session-oriented requests using PUT request for token.
- Token has TTL (time-to-live) between 1 second and 6 hours.
- Mitigates SSRF (Server-Side Request Forgery) attacks.
- Should be enforced on all instances for security best practices.
- Configure IMDSv2 required mode to prevent IMDSv1 access.

EC2 Image Builder

- Automates the creation, management, and deployment of customized, secure AMIs.
- Features:
  - Automated pipeline for building hardened AMIs.
  - Built-in security tests to validate images.
  - Automatic patching and updates on schedule.
  - Versioning and distribution to multiple regions/accounts.
  - Integration with Systems Manager for component management.
  - Compliance validation before AMI distribution.
- Security Benefits:
  - Embed security controls directly in AMIs (CIS benchmarks, STIG).
  - Automate removal of unnecessary software and services.
  - Ensure consistent security baseline across all instances.
  - Test images for vulnerabilities before production deployment.
  - Reduce attack surface through minimal, hardened images.
- Use Cases:
  - Creating golden AMIs with organizational security standards.
  - Implementing immutable infrastructure patterns.
  - Automating OS and application patching through image rebuilds.
  - Ensuring compliance with security frameworks (PCI DSS, HIPAA).

EC2 Instance Connect

- Browser-based SSH access to Linux instances without managing SSH keys.
- Temporary public key pushed to instance metadata for 60 seconds.
- Uses IAM permissions to control who can connect to instances.
- Audit trail in CloudTrail for all connection attempts.
- Security Benefits:
    - No long-lived SSH keys to manage or rotate.
    - Centralized access control through IAM policies.
    - No need to open port 22 to internet (can use with Session Manager).
    - Automatic key rotation (60-second validity).
    - Integration with IAM for MFA enforcement.
- Requirements:
    - Amazon Linux 2, Ubuntu 16.04 or later, or supported OS.
    - Instance must have public IP or use Instance Connect Endpoint.
    - Security group must allow SSH from AWS IP ranges (or use Endpoint).
    - IAM permissions: ec2-instance-connect:SendSSHPublicKey.

Systems Manager Session Manager

- Shell access to EC2 instances without SSH/RDP or bastion hosts.
- No inbound ports required (agent connects outbound to Systems Manager).
- Fully audited sessions logged to CloudWatch Logs or S3.
- Support for MFA before session initiation.
- Can restrict commands through IAM session documents.
- Port forwarding for accessing RDS, Redshift, or other internal resources.

References:
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-security.html
https://docs.aws.amazon.com/imagebuilder/latest/userguide/what-is-image-builder.html
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-connect-methods.html

Amazon Elastic Kubernetes Service (Amazon EKS)

- Managed Kubernetes service that runs Kubernetes control plane across multiple AZs.
- Eliminates need to install, operate, and maintain your own Kubernetes control plane.
- Certified Kubernetes conformant, ensuring compatibility with existing tooling.

Features

- Managed control plane with automatic scaling and high availability.
- Integration with AWS services (IAM, VPC, ELB, EBS, CloudWatch).
- Support for Kubernetes add-ons (CoreDNS, kube-proxy, VPC CNI).
- EKS Anywhere for running Kubernetes on-premises with AWS support.
- Fargate integration for serverless pod execution.
- Support for EC2 and Fargate worker nodes in same cluster.
- Automatic patching and version upgrades for control plane.

Security Features

- **Control Plane Security**
  - Control plane runs in AWS-managed VPC (isolated from customer account).
  - TLS encryption for all communication with control plane.
  - API server endpoints can be public, private, or both.
  - Private endpoint accessible only from within VPC.
  - Public endpoint can be restricted by IP CIDR.
- **IAM Integration**
  - IAM Roles for Service Accounts (IRSA) for pod-level IAM permissions.
  - Maps IAM principals to Kubernetes RBAC groups (aws-auth ConfigMap).
  - Service account tokens automatically injected into pods.
  - No need to store AWS credentials in pods or ConfigMaps.
- **Pod Security**
  - Pod Security Standards (Restricted, Baseline, Privileged).
  - Network policies for pod-to-pod traffic control.
  - Security groups for pods (individual pod-level network isolation).
  - Secrets encryption using AWS KMS.
  - Container image scanning with Amazon ECR image scanning.
- **Network Security**
  - Pods can have security groups directly attached (not just node-level).
  - Network policies control pod-to-pod and pod-to-service communication.
  - VPC CNI plugin for native VPC networking (pods get VPC IP addresses).
  - Private clusters with no internet gateway (all traffic through NAT).
- **Audit Logging**

    - Control plane logs sent to CloudWatch Logs (API, audit, authenticator, scheduler).
    - Kubernetes audit logs capture all API requests to control plane.
    - Integration with CloudTrail for EKS API management activity.

IAM Roles for Service Accounts (IRSA)

- Associates Kubernetes service account with IAM role.
- Pods using service account automatically receive temporary IAM credentials.
- Credentials obtained from STS via OIDC provider.
- Eliminates need for instance profile permissions (least privilege).
- Each pod/service can have different IAM permissions.
- Credentials automatically rotate by STS.

Secrets Encryption

- Kubernetes secrets stored in etcd can be encrypted using AWS KMS.
- Envelope encryption: etcd data encrypted with data key, data key encrypted with KMS key.
- Must be enabled during cluster creation (cannot enable later without migration).
- Provides encryption at rest for sensitive configuration data.

EKS Security Best Practices

- Use private endpoint or restrict public endpoint by IP CIDR.
- Enable control plane logging (especially audit logs) to CloudWatch.
- Encrypt Kubernetes secrets with AWS KMS.
- Use IRSA instead of node instance profiles for pod permissions.
- Implement pod security standards (start with Baseline, move to Restricted).
- Use security groups for pods for fine-grained network control.
- Scan container images in ECR before deployment.
- Implement network policies to control pod-to-pod traffic.
- Use Amazon GuardDuty EKS Protection to detect threats.
- Regularly update to latest Kubernetes version for security patches.

GuardDuty EKS Protection

- Analyzes EKS audit logs to detect suspicious activity.
- Detects privilege escalation, compromised credentials, and pod exploitation.
- Finding examples:
    - Execution of binary in sensitive container.
    - Anonymous access to Kubernetes API.
    - Privilege escalation to cluster-admin role.
    - Pod created with privileged mode or host network.

References:
https://docs.aws.amazon.com/eks/latest/userguide/security.html
https://docs.aws.amazon.com/eks/latest/userguide/security-iam.html
https://docs.aws.amazon.com/eks/latest/userguide/pod-security-policy.html

Amazon EMR (Elastic MapReduce)

- Cloud big data platform for processing vast amounts of data using open-source tools.
- Supports Apache Spark, Hadoop, HBase, Presto, Flink, and other frameworks.
- Used for log analysis, data transformation, machine learning, and analytics workloads.

Features

- Managed Hadoop framework that handles provisioning, configuration, and tuning.
- Auto-scaling to add or remove instances based on workload.
- Integration with S3 for data storage (separation of compute and storage).
- Support for spot instances to reduce costs.
- EMR Notebooks for interactive data exploration and development.
- EMR Studio for collaborative development environment.

Security Features

- **Encryption**
  - Encryption at rest for HDFS, EMRFS (S3), and local disks.
  - In-transit encryption between nodes using TLS certificates.
  - S3 encryption (SSE-S3, SSE-KMS, CSE-KMS, CSE-Custom).
  - EBS volume encryption using KMS keys.
- **Authentication and Authorization**
  - IAM roles for EMR service and EC2 instances.
  - Kerberos authentication for cluster components.
  - LDAP integration for user authentication.
  - Apache Ranger for fine-grained authorization.
  - Lake Formation integration for data lake permissions.
- **Network Security**
  - Security groups for master and worker nodes.
  - Launch EMR clusters in private subnets.
  - S3 VPC endpoints to keep traffic within AWS network.
  - Support for EMR on EKS for containerized workloads.

- **Auditing and Logging**
  - CloudTrail logs for EMR API calls.
  - S3 logs for cluster bootstrap actions and step execution.
  - CloudWatch Logs for application and system logs.
  - Integration with AWS Security Hub for findings.

Encryption Configuration

- **At-Rest Encryption:**
  - Local disk encryption (EBS volumes on cluster nodes).
  - EMRFS encryption for data in S3.
  - Encryption must be enabled at cluster launch (cannot enable later).
- **In-Transit Encryption:**
  - TLS certificates for inter-node communication.
  - Applied to Hadoop components, Spark, Tez, Presto.
  - Certificate can be provided or auto-generated.

Kerberos Authentication

- Provides strong authentication for Hadoop ecosystem components.
- Can use cluster-dedicated KDC or external KDC (Active Directory).
- Protects against replay attacks and man-in-the-middle attacks.
- Required for multi-tenant environments and compliance requirements.
- Works with data access controls in Apache Ranger or Lake Formation.

Apache Ranger Integration

- Provides centralized security administration for Hadoop components.
- Fine-grained access control for HDFS, Hive, Spark SQL.
- Policy-based authorization (user, group, role-based).
- Audit logging for all data access.
- Use case: Enforce column-level and row-level security in data lake.

EMR Security Configurations

- JSON document that specifies encryption, authentication, and authorization settings.
- Created once and reused across multiple clusters.
- Cannot be modified after cluster launch (immutable).
- Includes settings for:
  - Encryption at rest and in transit.
  - Kerberos authentication.
  - IAM roles for EMRFS requests to S3.

- Lake Formation integration.

Security Best Practices

- Enable encryption at rest and in transit for all clusters processing sensitive data.
- Use security configurations for consistent security settings across clusters.
- Launch clusters in private subnets with NAT for internet access.
- Use IAM roles for S3 access instead of hardcoding credentials.
- Enable Kerberos for multi-tenant environments.
- Use Apache Ranger or Lake Formation for fine-grained data access control.
- Configure security groups to restrict inbound access to master node.
- Use S3 VPC endpoints to keep S3 traffic within AWS network.
- Monitor cluster activity with CloudWatch and CloudTrail.
- Terminate clusters when not in use (ephemeral clusters for cost and security).

References:
https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-security.html
https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-data-encryption.html
https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-kerberos.html

AWS Lambda

- Serverless compute service that runs code in response to events without provisioning servers.
- Automatically scales from a few requests per day to thousands per second.
- Pay only for compute time consumed (billed per millisecond).

Features

- Event-driven execution from 200+ AWS services and SaaS applications.
- Automatic scaling and high availability across multiple AZs.
- Support for multiple languages (Python, Node.js, Java, Go, .NET, Ruby, custom runtimes).
- Built-in fault tolerance and automatic retries.
- Container image support (up to 10 GB).
- Integration with AWS services (S3, DynamoDB, API Gateway, EventBridge, etc.).

Security Features

- **Execution Role (IAM Role)**
  - Grants function permissions to access AWS services.
  - Attached to function at creation time.
  - Should follow least privilege principle.

    - Temporary credentials provided via environment variables.
- **Resource-Based Policies**
    - Control which services and accounts can invoke the function.
    - Allows cross-account invocation.
    - Example: Allow API Gateway or S3 to invoke function.
- **VPC Access**
    - Functions can access resources in VPC (RDS, ElastiCache, internal APIs).
    - Lambda creates ENIs in customer VPC subnets.
    - Security groups and NACLs control traffic.
    - Can access internet via NAT gateway in public subnet.
- **Encryption**
    - Environment variables encrypted at rest using AWS KMS.
    - Sensitive data should be encrypted before storing in environment variables.
    - Integration with Secrets Manager or Parameter Store for secrets.
    - In-transit encryption via TLS for all AWS API calls.
- **Code Security**
    - Code signing ensures only trusted code runs in production.
    - Signing profile with AWS Signer ensures code integrity.
    - Prevents deployment of unsigned or tampered code.
    - Enforced through function configuration.
- **Monitoring and Logging**
    - CloudWatch Logs for function execution logs.
    - CloudWatch metrics for invocations, errors, duration, throttles.
    - AWS X-Ray for distributed tracing and debugging.
    - CloudTrail for function configuration changes.

Environment Variables and Secrets

- Environment variables passed to function code at runtime.
- Automatically encrypted at rest with AWS-managed key or customer KMS key.
- Should NOT store sensitive data in plaintext environment variables.
- Best practice: Store secrets in AWS Secrets Manager or Parameter Store.
- Lambda can retrieve secrets at runtime using SDK.
- Use Lambda Extensions or environment variables with KMS decryption.

Lambda Function URLs

- Dedicated HTTPS endpoint for Lambda function (simple alternative to API Gateway).
- Can be public (no authentication) or IAM-authenticated.
- Supports CORS configuration.
- Use case: Webhooks, simple HTTP APIs, serverless functions without API Gateway.

- Security consideration: Public URLs are accessible to internet (implement authentication in code).

Lambda Layers

- Shared code and dependencies packaged separately from function code.
- Reduces deployment package size and promotes code reuse.
- Can include libraries, custom runtimes, or security tools.
- Security consideration: Verify integrity of third-party layers before use.

VPC Configuration

- Lambda functions outside VPC by default (access public AWS services only).
- VPC configuration required to access private resources (RDS, ElastiCache).
- Function assumes ENIs are available in specified subnets.
- Security groups attached to ENIs control outbound traffic.
- Cold start latency higher for VPC functions (ENI creation time).
- Best practice: Use VPC endpoints for AWS services to avoid NAT gateway costs.

Code Signing for Lambda

- Ensures only trusted code runs in Lambda function.
- Signing profile created in AWS Signer with signing certificate.
- Code package signed before upload to Lambda.
- Lambda verifies signature before execution.
- If signature invalid or missing, function fails to deploy.
- Use case: Enforce code integrity in production environments.

References:

https://docs.aws.amazon.com/lambda/latest/dg/lambda-security.html
https://docs.aws.amazon.com/lambda/latest/dg/configuration-vpc.html
https://docs.aws.amazon.com/lambda/latest/dg/configuration-codesigning.html

Amazon Data Lifecycle Manager (DLM)

- Automates the creation, retention, and deletion of EBS snapshots and EBS-backed AMIs.
- Eliminates manual snapshot management and ensures consistent backup schedules.
- Reduces storage costs by automatically deleting old snapshots.

Features

- Automated snapshot lifecycle policies based on tags.
- Cross-account snapshot copying for DR and backup.
- Cross-Region snapshot copying for geographic redundancy.
- Fast snapshot restore (FSR) for instant volume restoration.
- Snapshot archiving to lower-cost storage tier.
- Support for EBS volumes and EBS-backed AMIs.
- Integration with CloudWatch Events for policy execution.

Lifecycle Policies

- **EBS Snapshot Policies**
  - Automate creation of snapshots for tagged EBS volumes.
  - Schedule-based (hourly, daily, weekly, monthly, yearly).
  - Retention rules (count-based or age-based).
  - Snapshot tagging for cost allocation and organization.
- **EBS-Backed AMI Policies**
  - Automate creation of AMIs from tagged EC2 instances.
  - Include associated EBS snapshots automatically.
  - Useful for golden image creation and updates.
  - Can deregister old AMIs and delete associated snapshots.
- **Cross-Account Copy Policies**
  - Copy snapshots to different AWS accounts for isolation.
  - Separate production and backup accounts for security.
  - Useful for ransomware protection (immutable backups).
- **Cross-Region Copy Policies**
  - Copy snapshots to different regions for disaster recovery.
  - Geographic redundancy for business continuity.
  - Can combine with cross-account for maximum isolation.

Security Features

- **Snapshot Encryption**
  - Snapshots inherit encryption from source volume.
  - Can re-encrypt with different KMS key during cross-account/region copy.
  - Supports both AWS-managed and customer-managed KMS keys.
  - Encrypted snapshots cannot be shared publicly.
- **Access Control**
  - IAM policies control who can create and manage DLM policies.
  - Resource-based policies for cross-account snapshot sharing.
  - KMS key policies must grant permissions for cross-account access.
  - CloudTrail logs all DLM API activity for auditing.
- **Tagging**

- Policies target resources by tags (e.g., Backup=true).
- Snapshots automatically tagged with policy ID and creation timestamp.
- Tags used for cost allocation and compliance tracking.
- Can use tag-based IAM policies for additional access control.

Snapshot Archiving

- Move infrequently accessed snapshots to archive tier (75% cost reduction).
- Minimum retention in archive: 90 days.
- Restoration from archive takes 24-72 hours.
- Use case: Long-term compliance backups that rarely need restoration.
- DLM can automate archiving based on snapshot age.

Fast Snapshot Restore (FSR)

- Pre-warms EBS volumes created from snapshots for instant performance.
- Eliminates I/O latency during first access (no lazy loading).
- Billed per snapshot per AZ per hour (expensive).
- DLM can automatically enable FSR on snapshots.
- Use case: Production database restores where performance is critical.

Cross-Account Snapshot Copies for Ransomware Protection

- Copy snapshots to separate AWS account not accessible to production.
- Prevents attackers in production account from deleting backups.
- Backup account uses separate IAM credentials and SCPs.
- Enable MFA delete on S3 buckets if archiving to S3.
- Implement least privilege access to backup account.

DLM Policy Example Workflow

1. Tag EBS volumes with `Backup=Daily`.
2. Create DLM policy targeting volumes with `Backup=Daily` tag.
3. Schedule: Create snapshot daily at 2:00 AM UTC.
4. Retention: Keep 7 daily snapshots.
5. Copy to `us-west-2` for DR and `backup-account` for isolation.
6. Archive snapshots older than 30 days.
7. Delete snapshots older than 365 days.

Security Best Practices

- Encrypt all EBS volumes and snapshots with customer-managed KMS keys.
- Use cross-account snapshot copies for ransomware protection.

- Implement cross-Region copies for disaster recovery.
- Tag volumes consistently for DLM policy targeting.
- Use IAM policies to restrict who can modify or delete DLM policies.
- Enable CloudTrail logging for DLM API activity.
- Test snapshot restoration regularly to validate backup integrity.
- Use separate AWS accounts for production and backup isolation.
- Configure KMS key policies for cross-account snapshot sharing.
- Implement SCPs to prevent deletion of backups in backup accounts.

References:
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/snapshot-lifecycle.html
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/automating-snapshots.html
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-snapshot-archive.html

AWS Developer Tools

AWS Fault Injection Service (FIS)

- A fully managed chaos engineering service for running controlled fault injection experiments on AWS workloads.
- Tests application resilience by deliberately introducing failures and disruptions.
- Validates monitoring, observability, and incident response procedures.
- Used to find weaknesses before they cause real outages.

Features

- Pre-built experiment templates for common failure scenarios.
- Support for AWS services (EC2, ECS, EKS, RDS, Lambda, etc.).
- Stop conditions to halt experiments automatically if impacts are too severe.
- Integration with CloudWatch alarms to monitor experiment impact.
- Gradual rollout of experiments with percentage-based targeting.
- Automatic rollback capabilities to restore original state.
- Real-time experiment monitoring and logs.

Common Experiment Actions

- **EC2 Actions**
  - Stop, terminate, or reboot instances.
  - Inject CPU or memory stress.
  - Network disruptions (latency, packet loss, bandwidth throttling).

    - API throttling errors.
- **ECS/EKS Actions**
    - Stop or delete tasks/pods.
    - Inject CPU or memory stress in containers.
    - Network chaos for containerized applications.
- **RDS Actions**
    - Reboot database instances.
    - Failover to standby instance (Multi-AZ testing).
- **Lambda Actions**
    - Inject latency or errors into function invocations.
- **Systems Actions**
    - IAM service quota errors (simulate throttling).
    - Network blackhole (drop all traffic to/from resources).

Use Cases

- **Resilience Testing**
    - Verify application auto-scaling works correctly under load.
    - Test failover mechanisms (Multi-AZ RDS, ALB health checks).
    - Validate circuit breaker patterns and retry logic.
    - Ensure graceful degradation when dependencies fail.
- **Disaster Recovery Validation**
    - Test backup and restore procedures under failure conditions.
    - Validate cross-Region failover capabilities.
    - Ensure RTO and RPO targets are achievable.
- **Incident Response Testing**
    - Verify monitoring and alerting systems detect issues.
    - Test runbooks and incident response procedures.
    - Train teams to respond to production-like failures.
    - Validate that security teams are notified of anomalies.
- **Security Incident Simulation**
    - Test detection capabilities for GuardDuty, Security Hub.
    - Simulate instance compromise by stopping security agents.
    - Test automated response and remediation workflows.
    - Validate incident response plans and escalation procedures.

Security Features

- **IAM Permissions**
    - Requires explicit IAM permissions to create and run experiments.
    - Service-linked role (AWSServiceRoleForFIS) for AWS API actions.
    - Separate permissions for experiment templates vs. running experiments.

    - Can restrict experiments by resource tags.
- **Stop Conditions**
    - CloudWatch alarms that automatically stop experiments if breached.
    - Prevents experiments from causing excessive damage.
    - Examples: CPU > 90%, Error rate > 5%, Latency > 2s.
    - Multiple stop conditions can be configured (logical OR).
- **Resource Targeting**
    - Target resources by tags, resource IDs, or filters.
    - Supports percentage-based selection (e.g., 20% of instances).
    - Count-based selection (e.g., 3 out of 10 instances).
    - Prevents accidentally targeting all production resources.
- **Audit and Logging**
    - CloudTrail logs all FIS API activity (experiment creation, start, stop).
    - Experiment execution logs stored in CloudWatch Logs.
    - Tracks which user started experiments and when.
    - Action logs show exactly what actions were performed.
- **Safeguards**
    - Dry-run mode to validate experiment configuration without executing.
    - Manual approval required to start experiments (not automated).
    - Stop condition protection (cannot disable once experiment starts).
    - AWS-managed stop conditions for extreme situations.

Experiment Template Structure

- **Description** – What the experiment tests and expected outcome.
- **Targets** – Which resources to affect (tags, IDs, filters, selection mode).
- **Actions** – What to do to targets (stop EC2, inject latency, reboot RDS).
- **Stop Conditions** – CloudWatch alarms that halt the experiment.
- **IAM Role** – Role that FIS assumes to perform actions.

Stop Conditions

- CloudWatch alarms that monitor key metrics during experiments.
- If alarm breaches, experiment automatically stops.
- All actions are immediately rolled back where possible.
- Common stop conditions:
    - Application error rate exceeds threshold.
    - Request latency exceeds acceptable limits.
    - CPU or memory utilization too high.
    - Availability drops below minimum threshold.
- Stop conditions should be configured conservatively to prevent real outages.

Security Considerations for Chaos Engineering

- **Pre-Production Testing First**
  - Start experiments in non-production environments.
  - Gradually increase scope and severity.
  - Never run experiments in production without thorough testing.
- **Notification and Approval**
  - Notify security and operations teams before experiments.
  - Schedule experiments during maintenance windows.
  - Require management approval for production experiments.
  - Communicate blast radius and rollback plans.
- **Blast Radius Limitation**
  - Start with single instance or small percentage.
  - Use canary deployment pattern (test on subset first).
  - Target non-critical resources initially.
  - Never target 100% of resources in critical services.
- **Monitoring and Observability**
  - Ensure monitoring is comprehensive before experiments.
  - Have dashboards ready to observe experiment impact.
  - Configure stop conditions based on SLOs/SLAs.
  - Record baseline metrics before starting experiments.

Integration with Security Tools

- **GuardDuty Integration**
  - FIS can trigger GuardDuty findings during experiments.
  - Validates that GuardDuty detects unusual behavior.
  - Tests security team response to findings.
- **Security Hub Integration**
  - Experiments may generate Security Hub findings.
  - Validates centralized finding aggregation works correctly.
  - Tests automated remediation workflows triggered by findings.
- **CloudWatch Alarms**
  - Stop conditions prevent experiments from causing real incidents.
  - Validates that alarms are properly configured.
  - Tests that on-call teams receive notifications.
- **EventBridge Integration**
  - Experiment state changes sent to EventBridge.
  - Can trigger Lambda for custom logic or notifications.
  - Integrates experiments into broader orchestration workflows.

References:
https://docs.aws.amazon.com/fis/latest/userguide/what-is.html
https://docs.aws.amazon.com/fis/latest/userguide/security.html
https://docs.aws.amazon.com/fis/latest/userguide/experiments.html

AWS Internet of Things

AWS IoT Core

- A managed cloud service for connecting IoT devices to AWS and other devices.
- Enables secure bidirectional communication between devices and AWS cloud.
- Supports billions of devices and trillions of messages.
- Provides device gateway, message broker, rules engine, and device shadows.

Features

- Device gateway for secure device connections (MQTT, HTTPS, WebSockets).
- Message broker routes messages between devices and AWS services.
- Rules engine transforms and routes messages to other AWS services.
- Device shadows (persistent virtual representations of devices).
- Device Defender for security auditing and anomaly detection.
- Fleet provisioning for automated device onboarding.
- Jobs service for remote device management and updates.
- Support for multiple communication protocols (MQTT, MQTT over WSS, HTTPS).

Components

- **Device Gateway**
  - Entry point for device connections to AWS IoT Core.
  - Handles protocol translation (MQTT, HTTPS, WebSockets).
  - Scales automatically to support concurrent connections.
  - Each account has unique endpoint (xxxxxx.iot.region.amazonaws.com).
- **Message Broker**
  - Pub/sub message broker using MQTT protocol.
  - Routes messages between devices and AWS services.
  - QoS levels: 0 (at most once) and 1 (at least once).
  - Retains messages for 1 hour if device is disconnected.
- **Rules Engine**
  - SQL-based queries to filter and transform messages.
  - Routes messages to AWS services (Lambda, S3, DynamoDB, Kinesis, SNS).
  - Enables real-time data processing and analytics.
- **Device Shadow**

    - JSON document representing device state.
    - Syncs state between device and cloud.
    - Allows applications to interact with device even when offline.

Security Features

Authentication

- **X.509 Certificates (Recommended)**
    - Mutual TLS authentication between device and IoT Core.
    - Each device has unique certificate and private key.
    - Certificates can be AWS-generated or customer-provided.
    - Certificate rotation supported for security.
    - Certificates stored in AWS IoT certificate registry.
- **IAM Users, Groups, and Roles**
    - For AWS services and mobile applications accessing IoT Core.
    - Not typically used for IoT devices themselves.
    - IAM policies control access to IoT Core APIs.
- **Amazon Cognito Identity**
    - For mobile applications requiring temporary credentials.
    - Provides federated authentication with external identity providers.
- **Custom Authentication**
    - Custom authorizers using Lambda for non-standard auth.
    - Supports token-based authentication schemes.
    - Useful for integrating existing authentication systems.

Authorization (IoT Policies)

- JSON documents attached to X.509 certificates or Cognito identities.
- Define which actions a device can perform (connect, publish, subscribe).
- Specify allowed topics and topic patterns using wildcards.
- Policy variables enable dynamic authorization based on certificate attributes.
- More granular than IAM policies, designed specifically for IoT devices.

Encryption

- **In Transit**
    - All communication encrypted using TLS 1.2 or TLS 1.3.
    - Mutual TLS for device-to-cloud communication.
    - X.509 certificates verify both device and IoT Core identities.
- **At Rest**
    - Device shadow documents encrypted in DynamoDB.

  - Certificates stored encrypted in IoT certificate registry.
  - Messages stored in S3 (via Rules Engine) encrypted with SSE.

AWS IoT Device Defender

- Continuous security auditing and monitoring for IoT fleets.
- Two main capabilities: Audit and Detect.

Audit

- Checks IoT configurations against security best practices.
- Audit checks include:
  - Certificates expiring soon or revoked but still active.
  - IoT policies that are overly permissive (wildcards in resources).
  - Devices sharing certificates (security risk).
  - Unauthenticated Cognito identities with IoT policies.
  - CA certificates expiring or inactive but still in use.
  - Device certificates with same clientID (causes connection issues).
- Scheduled audits (daily, weekly, monthly) or on-demand.
- Results published to SNS topic for alerting.
- Integration with Security Hub for centralized findings.

Detect (Behavior Anomaly Detection)

- Machine learning models learn normal device behavior.
- Detects anomalies such as:
  - Unusual message rates (too many or too few).
  - Unusual connection patterns.
  - Unusual ports or IPs contacted.
  - High volume of authorization failures.
  - Traffic to unauthorized endpoints.
- CloudWatch metrics track behavior baselines.
- Alarms trigger when deviations exceed thresholds.
- Helps identify compromised devices or malware.

Security Profiles

- Define expected behaviors for device groups.
- Metrics tracked: messages sent/received, connections, authorization failures.
- Static thresholds or ML-based anomaly detection.
- Applied to thing groups or all things.
- Violations publish to SNS for automated response.

AWS IoT Device Defender Mitigation Actions

- Automated responses to security violations.
- Actions include:
  - Update device certificate to quarantine device.
  - Add things to quarantine thing group.
  - Replace overly permissive policy with restrictive policy.
  - Enable logging for affected devices.
  - Publish notification to SNS topic.
- Triggered automatically when violations detected.

Thing Groups and Thing Types

- **Thing Groups**
  - Logical grouping of devices (e.g., by location, device type, environment).
  - Hierarchical structure (up to 7 levels deep).
  - Apply policies and security profiles to entire groups.
  - Use for fleet management and access control.
- **Thing Types**
  - Define searchable attributes for device categorization.
  - Not used for access control, mainly for organization.

Fleet Provisioning

- Automated device registration and certificate provisioning.
- Two methods:
  - **Just-In-Time Provisioning (JITP)** – Device uses temporary certificate, replaced with permanent certificate on first connection.
  - **Bulk Provisioning** – Pre-register devices and certificates before deployment.
- Provisioning templates define device creation parameters.
- Reduces manual certificate management overhead.
- Pre-provisioning hooks (Lambda) can validate devices before registration.

IoT Jobs

- Remote execution of operations on devices.
- Use cases:
  - Software/firmware updates.
  - Security patches.
  - Certificate rotation.
  - Configuration changes.
- Jobs can target specific devices, thing groups, or all devices.

- Devices report job execution status.
- Rollout configuration controls deployment speed.

Logging and Monitoring

- **CloudWatch Logs**
  - IoT Core publishes logs for connections, publishes, subscribes, errors.
  - Log levels: ERROR, WARN, INFO, DEBUG (configurable).
  - Separate log streams per device for troubleshooting.
- **CloudWatch Metrics**
  - Metrics for message broker (messages published/delivered, connections).
  - Rule execution metrics (success, failure, action attempts).
  - Device Defender metrics for behavior monitoring.
- **CloudTrail**
  - Logs all IoT Core API calls (management operations).
  - Tracks who created policies, registered devices, modified rules.
  - Does not log data plane operations (publish, subscribe).

AWS IoT Policies

- JSON documents that control what actions an authenticated identity can perform in AWS IoT Core.
- Attached to X.509 certificates, Cognito identities, or thing groups.
- Provide fine-grained, device-specific authorization for IoT operations.
- Different from IAM policies - designed specifically for IoT device authorization.

Key Characteristics

- Use JSON format similar to IAM policies but with IoT-specific actions and resources.
- Support policy variables for dynamic, certificate-based authorization.
- Can use wildcards in resource ARNs for flexible topic/topic filter permissions.
- Evaluated in addition to IAM policies when both apply.
- More granular than IAM - operate at topic and clientId level.
- Required for devices to connect, publish, subscribe, or receive messages.

IoT Policy Structure

Basic policy contains:

- **Version** – Policy language version (typically "2012-10-17")
- **Statement** – Array of permission statements containing:
  - **Effect** – "Allow" or "Deny"
  - **Action** – IoT operations (iot:Connect, iot:Publish, iot:Subscribe, iot:Receive)
  - **Resource** – ARN of the resource (client, topic, topicfilter)

- **Condition** (optional) – Additional constraints on when policy applies

IoT Policy Actions

- **iot:Connect** – Allows device to connect to AWS IoT message broker
  - Resource: Client ID ARN (arn:aws:iot:region:account:client/clientId)
  - Required for any device communication
- **iot:Publish** – Allows publishing messages to MQTT topics
  - Resource: Topic ARN (arn:aws:iot:region:account:topic/topicName)
  - Controls which topics device can send messages to
- **iot:Subscribe** – Allows subscribing to MQTT topic filters
  - Resource: Topic filter ARN (arn:aws:iot:region:account:topicfilter/topicFilter)
  - Controls which topics device can listen to
- **iot:Receive** – Allows receiving messages on subscribed topics
  - Resource: Topic ARN (arn:aws:iot:region:account:topic/topicName)
  - Required in addition to Subscribe to actually receive messages
- **iot:GetThingShadow** – Read device shadow
  - Resource: Thing ARN
- **iot:UpdateThingShadow** – Update device shadow
  - Resource: Thing ARN
- **iot:DeleteThingShadow** – Delete device shadow
  - Resource: Thing ARN

Policy Variables

Policy variables enable dynamic authorization based on certificate attributes:

- **${iot:Connection.Thing.ThingName}** – Name of thing making the connection
- **${iot:Connection.Thing.ThingTypeName}** – Thing type of the connecting thing
- **${iot:ClientId}** – Client ID used in the connection
- **${iot:Certificate.Subject.CommonName}** – Common name from certificate
- **${iot:Certificate.Subject.Country}** – Country from certificate subject
- **${iot:Certificate.SerialNumber}** – Certificate serial number
- **${iot:Certificate.Issuer.CommonName}** – Certificate issuer common name

Policy variables allow single policy to work for many devices by substituting device-specific values at runtime.

References:
https://docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot.html
https://docs.aws.amazon.com/iot/latest/developerguide/security.html
https://docs.aws.amazon.com/iot/latest/developerguide/device-defender.html

https://docs.aws.amazon.com/iot/latest/developerguide/iot-policies.html
https://docs.aws.amazon.com/iot/latest/developerguide/thing-policy-variables.html

AWS Machine Learning Services

Amazon Bedrock

- A fully managed service providing access to foundation models (FMs) from leading AI companies through a single API.
- Enables building and scaling generative AI applications without managing infrastructure.
- Supports models from Anthropic (Claude), Meta (Llama), Cohere, AI21 Labs, Stability AI, and Amazon Titan.

Features

- Access to multiple foundation models via unified API.
- Model customization with fine-tuning and continued pre-training.
- Knowledge bases with Retrieval-Augmented Generation (RAG).
- Agents that can execute tasks and call APIs automatically.
- Guardrails to implement safety and privacy controls.
- Model evaluation to compare model performance.
- Provisioned throughput for consistent performance and lower latency.

Security Features

- **Data Privacy**
  - Customer data not used to train base foundation models.
  - No data shared between customers.
  - Data encrypted in transit (TLS) and at rest (KMS).
  - Data residency controls (data stays in selected region).
  - Optional VPC endpoints for private connectivity.
- **Access Control**
  - IAM policies control access to Bedrock APIs and models.
  - Service Control Policies (SCPs) can restrict model access.
  - Resource-based policies for cross-account access.
  - Support for IAM conditions (source IP, VPC, tags).
- **Encryption**
  - All data encrypted at rest using AWS KMS.
  - Customer-managed KMS keys supported.
  - In-transit encryption using TLS 1.2+.
- **Audit and Compliance**
  - CloudTrail logs all Bedrock API calls.

    - CloudWatch metrics for model invocations and errors.
    - Integration with AWS Config for compliance monitoring.
    - Supports compliance frameworks (SOC, ISO, HIPAA eligible).

Bedrock Guardrails

- Content filtering and safety controls for generative AI applications.
- Protection categories:
    - **Content Filters** – Block harmful content (hate, violence, sexual, misconduct).
    - **Denied Topics** – Prevent discussion of specific subjects.
    - **Word Filters** – Block profanity or custom word lists.
    - **Sensitive Information Filters** – Detect and redact PII (SSN, credit cards, emails).
- Configurable filter strength: None, Low, Medium, High.
- Applied to both input prompts and model responses.
- Supports custom regex patterns for organization-specific content.

Sensitive Information Protection

- Automatically detects PII in prompts and responses.
- Supports managed identifiers:
    - Names, addresses, email addresses, phone numbers.
    - Social Security Numbers, credit card numbers.
    - Driver's licenses, passport numbers.
    - Health insurance IDs, medical record numbers.
- Actions when PII detected:
    - **Block** – Reject entire request/response.
    - **Anonymize** – Replace PII with placeholder (e.g., [NAME], [SSN]).
- Custom regex patterns for organization-specific sensitive data.

Knowledge Bases (RAG)

- Retrieval-Augmented Generation connects FMs to data sources.
- Data sources: Amazon S3, Amazon OpenSearch, Amazon Aurora, Pinecone, Redis.
- Vector embeddings stored in vector databases.
- Security considerations:
    - IAM role required for Bedrock to access data sources.
    - Data encrypted in vector databases.
    - Access logs track which data is retrieved for responses.
    - Implement least privilege for data source access.

Bedrock Agents

- Autonomous agents that plan and execute tasks using APIs and knowledge bases.

- Can invoke Lambda functions to perform actions.
- Security considerations:
    - IAM role controls agent's permissions to invoke functions.
    - Action groups define allowed API calls.
    - Guardrails apply to agent inputs and outputs.
    - Agent activity logged in CloudTrail.

Model Customization Security

- Fine-tuning and continued pre-training use customer data.
- Training data encrypted at rest (S3 with SSE-S3 or SSE-KMS).
- Custom models private to customer account.
- Training jobs run in isolated environment.
- Training data not used for base model improvements.

VPC Endpoints (AWS PrivateLink)

- Private connectivity to Bedrock without internet gateway.
- Traffic stays within AWS network.
- Use case: Compliance requirements for data not traversing public internet.
- Endpoint policies control which models can be accessed.

References:
https://docs.aws.amazon.com/bedrock/latest/userguide/what-is-bedrock.html
https://docs.aws.amazon.com/bedrock/latest/userguide/security.html
https://docs.aws.amazon.com/bedrock/latest/userguide/guardrails.html

Amazon CodeGuru Security

- An automated security scanning tool that detects security vulnerabilities in application code.
- Uses machine learning and automated reasoning to identify security issues.
- Provides remediation recommendations and code fixes.

Features

- Static application security testing (SAST) for code.
- Detects vulnerabilities including OWASP Top 10.
- Supports Java, Python, JavaScript, TypeScript, C#.
- Integration with CI/CD pipelines (GitHub, GitLab, Bitbucket, Jenkins).
- Scans code repositories, pull requests, and local code.
- Generates security findings with severity ratings.

- Provides remediation guidance and code snippets.
- Secrets detection (hardcoded credentials, API keys).

Security Vulnerability Detection

- **Injection Flaws**
  - SQL injection, command injection, LDAP injection.
  - NoSQL injection, XML injection.
- **Broken Authentication**
  - Weak password requirements.
  - Improper session management.
  - Insecure credential storage.
- **Sensitive Data Exposure**
  - Hardcoded secrets (passwords, API keys, tokens).
  - Logging sensitive information.
  - Unencrypted data transmission.
- **Broken Access Control**
  - Missing authorization checks.
  - Insecure direct object references.
  - Path traversal vulnerabilities.
- **Security Misconfiguration**
  - Default credentials.
  - Overly permissive permissions.
  - Insecure cryptographic practices.
- **Cross-Site Scripting (XSS)**
  - Reflected XSS, stored XSS, DOM-based XSS.
  - Improper input validation.
- **Insecure Deserialization**
  - Untrusted data deserialization.
  - Remote code execution risks.
- **Using Components with Known Vulnerabilities**
  - Outdated libraries with CVEs.
  - Vulnerable dependencies.
- **Insufficient Logging and Monitoring**
  - Missing security event logging.
  - Inadequate error handling.

Secrets Detection

- Scans code for hardcoded secrets before they reach repositories.
- Detects:
  - AWS credentials (access keys, secret keys).

    - API keys (Google, Stripe, Twilio, SendGrid, etc.).
    - Database credentials (MySQL, PostgreSQL, MongoDB).
    - Private keys (RSA, SSH, PGP).
    - OAuth tokens, JWT tokens.
    - Generic passwords in code.
- Reduces risk of credential exposure in public repositories.

Finding Severity Levels

- **Critical** – Immediate risk of exploitation (e.g., SQL injection in authentication).
- **High** – Significant security risk requiring prompt attention.
- **Medium** – Moderate risk that should be addressed.
- **Low** – Minor issues or best practice violations.
- **Info** – Informational findings, not security issues.

Integration with CI/CD

- Scan code automatically on commits, pull requests, merges.
- Fail builds if critical vulnerabilities detected.
- GitHub Actions, GitLab CI, Jenkins, AWS CodePipeline support.
- API available for custom integrations.
- Results published to Security Hub for centralized visibility.

References:
https://docs.aws.amazon.com/codeguru/latest/security-ug/what-is-codeguru-security.html
https://docs.aws.amazon.com/codeguru/latest/security-ug/security.html

Amazon Q Business

- A generative AI-powered assistant for enterprise business use.
- Connects to company data sources to answer questions and generate content.
- Provides conversational interface for employees to access information.

Features

- Natural language Q&A over enterprise data.
- Connects to 40+ data sources (S3, SharePoint, Salesforce, ServiceNow, etc.).
- Respects data source permissions and access controls.
- Web experience and chat interface.
- Admin controls and guardrails.

- Usage analytics and monitoring.

Security Features

- **Data Source Permissions**
  - Respects source system permissions (user sees only authorized data).
  - Identity-aware responses based on user's access level.
  - Supports SAML 2.0 and OIDC for authentication.
  - Integration with enterprise identity providers (Okta, Azure AD).
- **Access Control**
  - IAM policies control Q Business administration.
  - Application-level access controls for user groups.
  - Admin can restrict which data sources are accessible.
- **Data Privacy**
  - Customer data not used to improve base models.
  - Data encrypted in transit (TLS) and at rest (KMS).
  - Data processing stays within customer's AWS account.
  - Supports VPC deployment for private connectivity.
- **Content Filtering**
  - Built-in guardrails to block inappropriate content.
  - Admin-configurable content policies.
  - Prevents generation of harmful or biased responses.
- **Audit and Monitoring**
  - CloudTrail logs administrative actions.
  - Usage metrics in CloudWatch.
  - Conversation logs for compliance (optional).

Data Source Connectors

- Amazon S3, SharePoint, Google Drive, OneDrive.
- Salesforce, ServiceNow, Zendesk, Slack, Jira.
- Database connectors (RDS, Redshift, Snowflake).
- Custom connectors via API.
- Each connector requires IAM role with data source access.

References:
https://docs.aws.amazon.com/amazonq/latest/business-use-dg/what-is.html
https://docs.aws.amazon.com/amazonq/latest/business-use-dg/security.html

Amazon Q Developer

- An AI-powered coding assistant for software developers.
- Provides code suggestions, explanations, and security scanning.
- Integrates with IDEs (VS Code, JetBrains, Visual Studio, AWS Cloud9).

Features

- Real-time code suggestions and completions.
- Natural language to code generation.
- Code explanations and documentation.
- Security vulnerability scanning in IDE.
- Code transformation (language upgrades, framework migrations).
- Command-line chat for AWS CLI and infrastructure questions.
- Integration with AWS services for context-aware suggestions.

Security Features

- **Security Scanning**
  - Real-time security vulnerability detection in IDE.
  - Detects same issues as CodeGuru Security (OWASP Top 10).
  - Hardcoded secret detection before commit.
  - Immediate feedback to developers during coding.
- **Code Reference Tracking**
  - Identifies when suggestions match public code.
  - Provides attribution and license information.
  - Helps avoid license compliance issues.
- **Data Privacy**
  - Code snippets sent to AWS for analysis.
  - Customer code not used to train base models.
  - Data encrypted in transit and at rest.
  - Admins can disable code sharing for analysis.
- **Access Control**
  - AWS Builder ID or IAM Identity Center for authentication.
  - Organization-level controls via IAM Identity Center.
  - Individual developer or team licensing.

Security Vulnerability Scanning

- Scans code as developers type in IDE.
- Detects:

    - SQL injection, command injection.
    - Cross-site scripting (XSS).
    - Hardcoded credentials and secrets.
    - Insecure cryptography.
    - Path traversal vulnerabilities.
    - Insecure deserialization.
- Provides inline suggestions for remediation.
- Severity ratings: Critical, High, Medium, Low, Info.

Code Transformation

- Automates code upgrades (Java 8 to 17, Python 2 to 3).
- Migrates applications between frameworks.
- Security benefit: Upgrades remove deprecated insecure APIs.
- Automated testing validates transformations.

References:
https://docs.aws.amazon.com/amazonq/latest/qdeveloper-ug/what-is.html
https://docs.aws.amazon.com/amazonq/latest/qdeveloper-ug/security-scans.html

Amazon SageMaker AI

- A fully managed service for building, training, and deploying machine learning models at scale.
- Provides tools for entire ML lifecycle from data preparation to model deployment.
- Used for fraud detection, recommendation systems, predictive maintenance, anomaly detection.

Features

- Notebooks for data exploration and model development.
- Data Wrangler for data preparation and feature engineering.
- Built-in algorithms and pre-trained models.
- Distributed training on GPU/CPU clusters.
- Hyperparameter tuning and AutoML.
- Model registry and versioning.
- Real-time and batch inference endpoints.
- Model monitoring and drift detection.
- MLOps capabilities (pipelines, CI/CD for ML).

Security Features

- **Access Control**

    - IAM policies control access to SageMaker resources.
    - IAM roles for notebook instances, training jobs, endpoints.
    - Resource-based policies for cross-account model sharing.
    - Fine-grained permissions per SageMaker action.
- **Network Isolation**
    - VPC deployment for notebook instances and training jobs.
    - Network isolation mode prevents internet access.
    - VPC endpoints for private API access.
    - Private endpoints for model inference (no internet).
- **Encryption**
    - Data at rest encrypted with KMS (S3, EBS, EFS).
    - Customer-managed KMS keys supported.
    - Inter-node encryption for distributed training.
    - In-transit encryption for model endpoints (HTTPS).
- **Monitoring and Logging**
    - CloudWatch metrics for training and inference.
    - CloudWatch Logs for container logs.
    - CloudTrail for API activity auditing.
    - Model Monitor detects data drift and quality issues.
- **Model Security**
    - Model artifacts stored encrypted in S3.
    - Model registry tracks model lineage and approvals.
    - Inference endpoints support authentication via IAM.
    - Model explainability with SageMaker Clarify.

SageMaker Notebook Security

- Jupyter notebooks run on EC2 instances (managed by SageMaker).
- Can be launched in VPC for access to private resources.
- Internet access can be disabled for isolation.
- IAM role attached provides AWS API permissions.
- Root access can be disabled for security hardening.
- Lifecycle configurations for automated security setup.
- Pre-signed URLs for notebook access (time-limited).

Training Job Security

- Training containers run in isolated environments.
- VPC configuration keeps training traffic private.
- Network isolation mode prevents internet and VPC access.
- Inter-container traffic encryption for distributed training.
- Training data encrypted in S3 and decrypted in memory.

- IAM role grants access to training data and output locations.
- Spot instances supported but not recommended for sensitive data.

Inference Endpoint Security

- Real-time endpoints behind HTTPS endpoints.
- Authentication via IAM SigV4 request signing.
- VPC configuration for private endpoints (no internet).
- Autoscaling based on traffic (no manual capacity management).
- Model artifacts encrypted in S3 and at rest on endpoint.
- Endpoint logs captured in CloudWatch Logs.
- Data capture for model monitoring (optional, consider PII).

SageMaker Model Monitor

- Continuous monitoring of model quality in production.
- Detects data drift (distribution changes in inputs).
- Model quality monitoring (accuracy degradation).
- Bias drift detection (fairness over time).
- Feature attribution drift (explainability changes).
- Alerts via CloudWatch alarms when issues detected.
- Security relevance: Detect adversarial inputs, model poisoning.

SageMaker Clarify

- Detects bias in training data and model predictions.
- Provides model explainability (feature importance).
- Generates reports for compliance and governance.
- Security relevance: Ensure models don't discriminate, meet fairness requirements.

SageMaker Pipelines (MLOps)

- CI/CD for machine learning workflows.
- Automate data processing, training, evaluation, deployment.
- Version control for pipelines and models.
- Approval gates before production deployment.
- IAM integration for pipeline execution permissions.
- Security benefit: Consistent, auditable ML workflows.

SageMaker Feature Store

- Centralized repository for ML features.
- Online store (low latency) and offline store (S3).

- Encryption at rest and in transit.
- IAM-based access control.
- Audit trail with CloudTrail.
- Security consideration: Ensure features don't leak sensitive data.

Security Best Practices

- Deploy notebook instances and training jobs in VPC.
- Use network isolation mode for highly sensitive workloads.
- Encrypt all data at rest with customer-managed KMS keys.
- Enable inter-container traffic encryption for distributed training.
- Use IAM least privilege for SageMaker execution roles.
- Disable internet access on notebook instances when not needed.
- Implement model approval workflows before production deployment.
- Use SageMaker Model Monitor to detect drift and anomalies.
- Enable CloudTrail logging for all SageMaker API activity.
- Scan container images for vulnerabilities before use.
- Use VPC endpoints for private connectivity to AWS services.
- Implement data anonymization before training on sensitive data.
- Use SageMaker Clarify to detect and mitigate bias.
- Version models in Model Registry for auditability.
- Secure inference endpoints with IAM authentication.

Common Security Use Cases

- **Fraud Detection** – Train models to detect fraudulent transactions, account takeovers.
- **Anomaly Detection** – Identify unusual patterns in logs, network traffic, user behavior.
- **Threat Intelligence** – ML models to classify malware, detect phishing.
- **Access Pattern Analysis** – Detect compromised credentials based on login patterns.
- **Security Operations** – Automate incident triage and prioritization.

References:
https://docs.aws.amazon.com/sagemaker/latest/dg/security.html
https://docs.aws.amazon.com/sagemaker/latest/dg/security-iam.html
https://docs.aws.amazon.com/sagemaker/latest/dg/best-practices-security.html

AWS Security & Identity Services

AWS Audit Manager

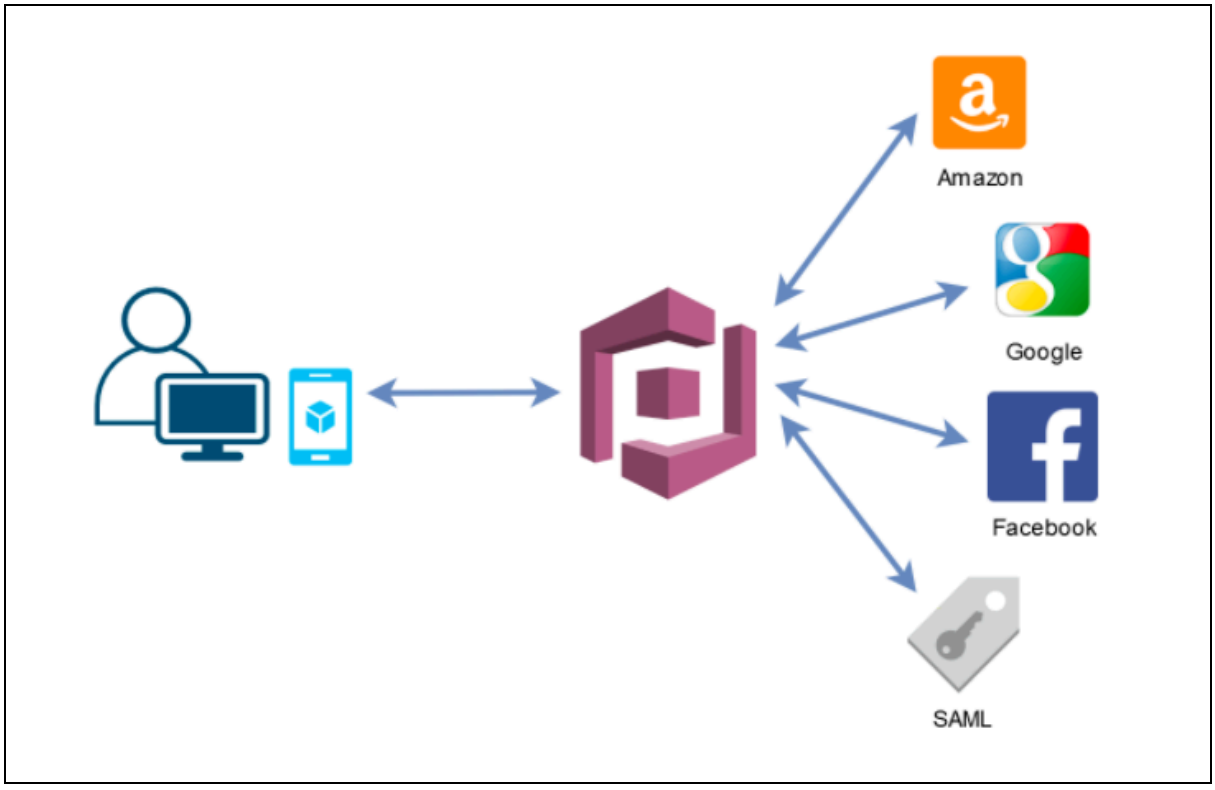
- A service that will help you audit your AWS usage on a regular basis in order to simplify risk management and compliance with regulations and industry standards.
- Automates evidence collection for policies, procedures, and activities, as well as the creation of audit reports.
- Features
    - Centrally manage and upload evidence from on-premises or multi-cloud environments.
    - View analytics data for active assessments on the Audit Manager dashboard and quickly identify non-compliant evidence that needs to be remedied.
    - Creation of frameworks with standard or custom controls based on your specific internal audit requirements.
    - Custom frameworks can also be shared with another AWS account or replicated into another AWS Region under your own account.
    - Supports control set delegation to team members to assist you in reviewing related evidence, adding comments, and updating the status of each control.
- Concepts
    - Assessments
        - An assessment is based on a framework, which is a collection of controls.
        - When you create an assessment, continuous collection of evidence begins.
        - For audit, you or a delegate can review this evidence and add it to an assessment report.
        - An assessment has two states:
            - Active – currently collecting evidence.
            - Inactive – stops collecting evidence.
    - Assessment reports
        - Summarizes the evidence that was gathered from an assessment.
        - It also includes links to evidence PDF files.
        - Assessment reports are placed in an S3 bucket.
    - Delegations
        - Allows you to delegate a control set to a subject matter expert for review and validation of evidence.
        - In different AWS Regions, an account can be:
            - Audit owner
            - Delegate
        - Delegates are asked by audit owners to review the evidence associated with a control set.
    - Framework library

        - Defines the controls and data source mappings for a given compliance standard or regulation
            - Standard Frameworks – prebuilt AWS frameworks
            - Custom Frameworks – frameworks that you own.
        - By creating a share request, a recipient can use your custom framework to create assessments.
    - Control library
        - Standard Controls
            - AWS predefined controls.
            - Editing or deleting standard controls is not allowed.
            - You can customize any standard control to meet your specific requirements.
        - Custom Controls
            - Customized controls that you own.
            - Allows you to define which data sources you want to collect evidence from.
        - The data source types for automated evidence:
            - AWS API calls
            - AWS Config
            - AWS Security Hub
            - AWS CloudTrail
- AWS Audit Manager Monitoring
    - You can capture snapshots of your resource security posture by reporting:
        - Results of security checks directly from AWS Security Hub.
        - Findings to AWS Config.
    - Collects log data from AWS CloudTrail and converts processed logs into evidence of user activity.
    - Audit Manager includes a License Manager framework to help you prepare for audits.
    - You can use the following services to help you prepare for your audit:
        - AWS License Manager framework
        - AWS Control Tower Guardrails framework
    - Using Amazon SNS, you can send a notification to a user when one of the following events occurs:
        - The audit owner delegates a control set for review.
        - The audit owner has finished reviewing a control set.
        - The delegate submits a control set that has been reviewed to the audit owner.
- AWS Audit Manager Security
    - Uses AWS IAM service-linked roles to connect to data sources.
    - Data is encrypted using the AWS KMS key.
- AWS Audit Manager Pricing
    - You are charged based on the number of resource assessments performed.
    - You are charged for assessment reports stored in Amazon S3.

References:
https://aws.amazon.com/audit-manager/
https://docs.aws.amazon.com/audit-manager/latest/userguide/what-is.html

Amazon Cognito

- A user management and authentication service that can be integrated to your **web or mobile applications**. Amazon Cognito also enables you to authenticate users through an **external identity provider** and provides **temporary security credential**s to access your app's backend resources in AWS or any service behind Amazon API Gateway. Amazon Cognito works with external identity providers that support SAML or OpenID Connect, social identity providers (Facebook, Twitter, Amazon, Google, Apple) and you can also integrate your own identity provider.
- An Amazon Cognito ID token is represented as a **JSON Web Token** (JWT). Amazon Cognito uses JSON Web Tokens for token authentication.
- How It Works



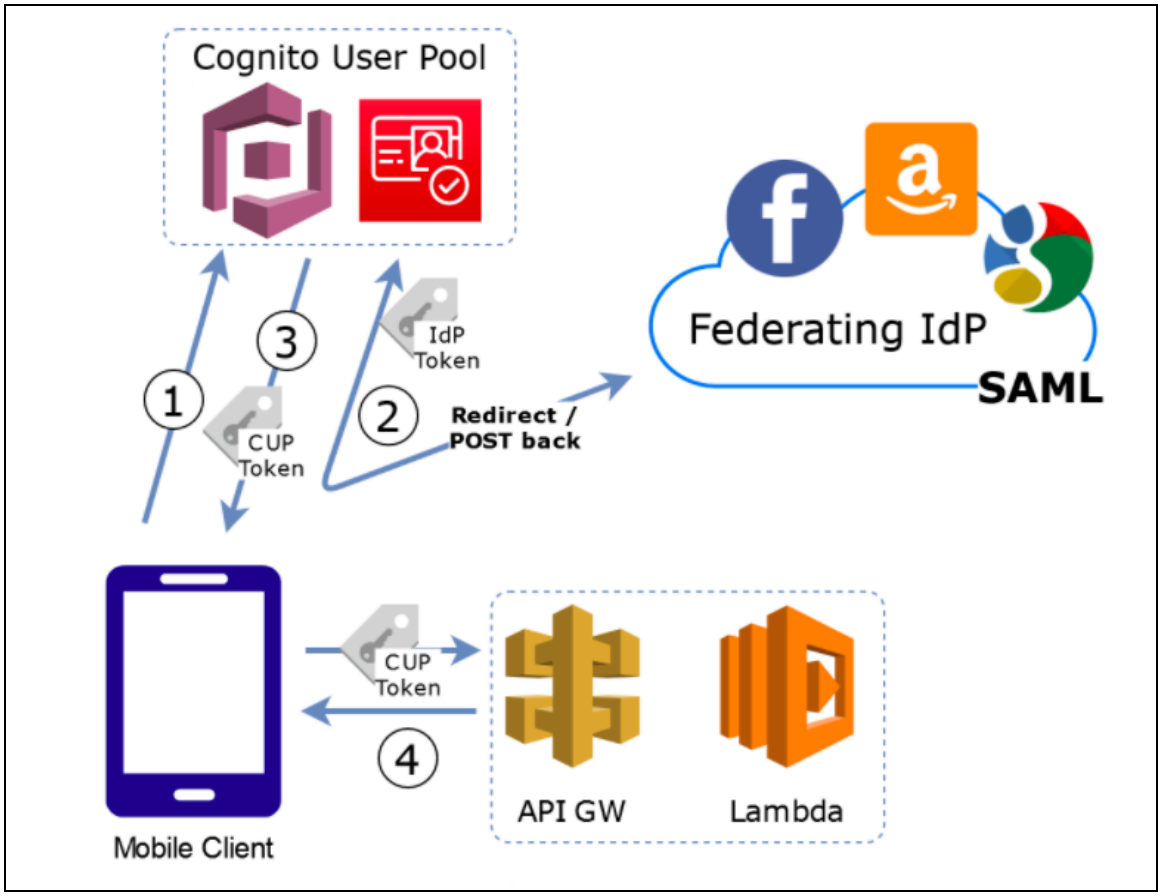
User Pools

- User pools are user directories that provide sign-up and sign-in options for your app users.
- Users can sign in to your web or mobile app through Amazon Cognito, or federate through a third-party identity provider (IdP).

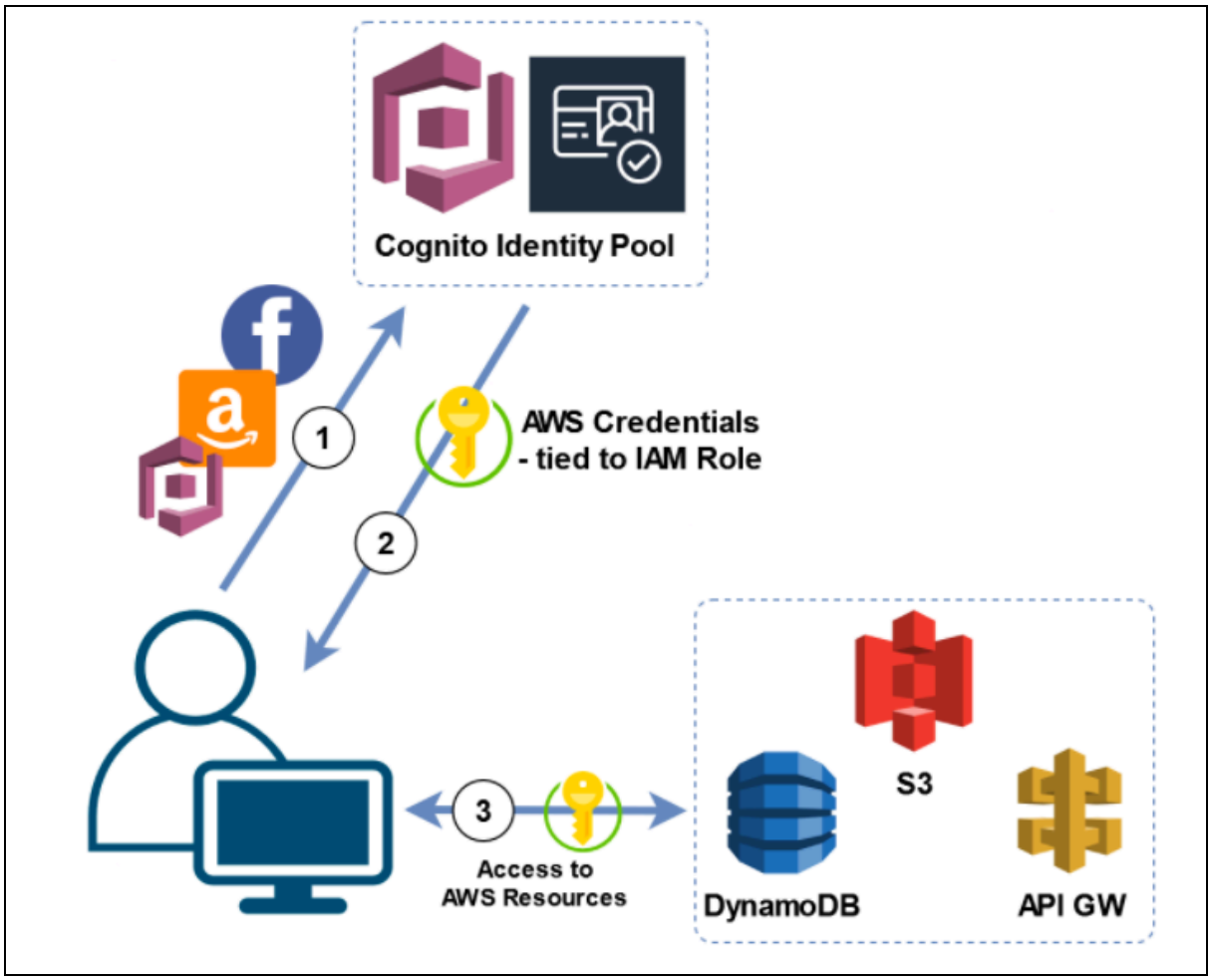
- You can use the aliasing feature to enable your users to sign up or sign in with an email address and a password or a phone number and a password.
- User pools are each created in one AWS Region, and they store the user profile data only in that region. You can also send user data to a different AWS Region.
- A User Pool is like a *directory* of users.
- Manage Users
  - After you create a user pool, you can create, confirm, and manage users accounts.
  - Amazon Cognito User Pools groups lets you manage your users and their access to resources by mapping IAM roles to groups.
  - User accounts are added to your user pool in one of the following ways:
  - The user signs up in your user pool's client app, which can be a mobile or web app.
  - You can import the user's account into your user pool.
  - You can create the user's account in your user pool and invite the user to sign in.
  - Sign up authflow below

- **Identity Pools**
  - Use this feature if you want to federate users to your AWS services.
  - Identity pools enable you to grant your users temporary AWS credentials to access AWS services, such as Amazon S3 and DynamoDB.
  - Identity pools support anonymous guest users, as well as the following identity providers:
    - Amazon Cognito user pools
    - Social sign-in with Facebook, Google, and Login with Amazon
    - OpenID Connect (OIDC) providers
    - SAML identity providers
    - Developer authenticated identities
  - To save user profile information, your identity pool needs to be integrated with a user pool.
  - Amazon Cognito Identity Pools can support unauthenticated identities by providing a unique identifier and AWS credentials for users who do not authenticate with an identity provider.
  - The permissions for each authenticated and non-authenticated user are controlled through IAM roles that you create.
  - Once you have an OpenID Connect token, you can then trade this for temporary AWS credentials via the *AssumeRoleWithWebIdentity* API call in AWS Security Token Service (STS). This call is no different than if you were using Facebook, Google+, or Login with Amazon directly, except that you are passing an Amazon Cognito token instead of a token from one of the other public providers.
- Common Use Cases
  - Enable your users to authenticate with a user pool.
  - After a successful user pool sign-in, your web or mobile app will receive user pool tokens from Amazon Cognito. You can use those tokens to control access to your server-side resources.

- Access resources with API Gateway and Lambda with a User Pool. API Gateway validates the tokens from a successful user pool authentication, and uses them to grant your users access to resources, including Lambda functions, or your own API.



- After a successful user pool authentication, your app will receive user pool tokens from Amazon Cognito. You can exchange them for temporary access to other AWS services with an identity pool.



- Enable your users access to AWS services through an identity pool. In exchange, the identity pool grants temporary AWS credentials that you can use to access other AWS services.
- Grant your users access to AWS AppSync resources with tokens from a successful Amazon Cognito authentication (from a user pool or an identity pool).
- Amazon Cognito is also commonly used together with AWS Amplify, a framework for developing web and mobile applications with AWS services.

Amazon Cognito Sync

- Store and sync data across devices using Cognito Sync.
- You can programmatically trigger the sync of data sets between client devices and the Amazon Cognito sync store by using the synchronize() method in the AWS Mobile SDK. The synchronize() method reads the **latest version** of the data available in the Amazon Cognito sync store and compares it to the local, cached copy. After comparison, the synchronize() method writes the latest updates as necessary to the local data store and the Amazon Cognito sync store.

    - The Amazon Cognito Sync store is a key/value pair store linked to an Amazon Cognito identity. There is no limit to the number of identities you can create in your identity pools and sync store.
    - Each user information store can have a maximum size of 20MB. Each data set within the user information store can contain up to 1MB of data. Within a data set you can have up to 1024 keys.
    - With Cognito Streams, you can push sync store data to a Kinesis stream in your AWS account.
- Advanced Security Features
    - When Amazon Cognito detects unusual sign-in activity, such as sign-in attempts from new locations and devices, it assigns a risk score to the activity and lets you choose to either prompt users for additional verification or block the sign-in request.
    - Users can verify their identities using SMS or a Time-based One-time Password (TOTP) generator.
    - When Amazon Cognito detects users have entered credentials that have been compromised elsewhere, it prompts a password change.
- Integration with AWS Lambda
    - You can create an AWS Lambda function and then trigger that function during user pool operations such as user sign-up, confirmation, and sign-in (authentication) with a Lambda trigger.
    - Amazon Cognito invokes Lambda functions synchronously. When called, your Lambda function must respond within 5 seconds. If it does not, Amazon Cognito retries the call. After 3 unsuccessful attempts, the function times out.
    - You can create a Lambda function as a backend to Cognito that serves auth challenges to users signing in.
- Pricing
    - If you are using Cognito Identity to create a User Pool, you pay based on your monthly active users (MAUs) only. A user is counted as a MAU if, within a calendar month, there is an identity operation related to that user, such as sign-up, sign-in, token refresh or password change.
        - The Cognito Your User Pool feature has a free tier of 50,000 MAUs for users who sign in directly to Cognito User Pools or through social identity providers and 50 MAUs for users federated through SAML 2.0 based identity providers.
    - You pay an additional fee when you enable advanced security features for Amazon Cognito.
    - Amazon Cognito uses Amazon SNS for sending SMS messages for Multi-Factor Authentication (MFA) and phone number verification, so there are associated SNS costs as well.

References:
https://aws.amazon.com/cognito/
https://aws.amazon.com/cognito/faqs/
https://docs.aws.amazon.com/cognito/latest/developerguide/what-is-amazon-cognito.html
Overview of Amazon Cognito User Pools and Federated Identities

Amazon Detective

- The service automatically collects log data from your AWS resources and uses machine learning, statistical analysis, and graph theory to build a linked set of data that enables you to easily conduct faster and more efficient security investigations.
- Can be integrated with AWS security services like Amazon GuardDuty, Amazon Macie, and AWS Security Hub as well as partner security products to identify potential security issues or findings.
- Amazon Detective can analyze trillions of events from multiple data sources, such as VPC Flow Logs, AWS CloudTrail, and Amazon GuardDuty, and automatically creates a unified, interactive view of your resources, users, and the interactions between them over time. This allows you to identify the underlying reasons for the findings, drill down into relevant historical activities, and quickly determine the root cause of a security concern.
- Amazon Detective's prebuilt data aggregations, summaries, and context help you to quickly analyze and determine the nature and extent of possible security issues.
- Concepts
    - Investigation - The process of performing triage on suspicious or interesting activity, determining the scope, getting to its underlying source or cause, and then determining how to proceed.
    - Behavior graph - A linked set of data generated from incoming source data that is associated with one or more AWS accounts. Each behavior graph uses the same structure of findings, entities, and relationships.
    - Management account - The AWS account that owns a behavior graph and that uses the behavior graph for investigation. The management account invites member accounts to contribute their data to the behavior graph. Management accounts can also view data usage for the behavior graph, and remove member accounts from the behavior graph.
    - Member account - An AWS account that a management account invited to contribute data to a behavior graph. Member accounts can respond to the behavior graph invitation and remove their account from the behavior graph. They have no other access to the behavior graph.
    - **Finding** - A security issue detected by Amazon GuardDuty.
    - **Entity** - An item extracted from the incoming data. Each entity has a type, which identifies the type of object it represents. Examples include IP addresses, Amazon EC2 instances, and AWS users.
        - For each entity, the source data is also used to populate entity properties. Property values can be extracted directly from source records or aggregated across multiple records.
    - **Relationship** - Activity that occurs between individual entities. Relationships are also extracted from the incoming source data.
        - Similar to an entity, a relationship has a type, which identifies the types of entities involved and the direction of the connection. An example of a relationship type is an IP address connecting to an Amazon EC2 instance.

    - Profile - For a finding or an entity, a single page that provides a collection of data visualizations plus supporting guidance.
        - For findings, profiles help analysts to determine whether the finding is of genuine concern or a false positive.
        - For entities, profiles provide supporting details for an investigation into a finding or for a general hunt for suspicious activity.
    - Scope time - The time window that is used to scope the data displayed on finding and entity profiles. The default scope time for a finding profile reflects the first and last times when the suspicious activity was observed. The default scope time for an entity profile is the previous 24 hours.
- Amazon Detective needs to be enabled on a **per region** basis and enables you to quickly analyze activity across all your accounts within each region.
- Amazon Detective is a **multi-account service** that aggregates data from monitored member accounts under a single management account within the same region. You can configure multi-account monitoring deployments in the same way that you configure management and member accounts in Amazon GuardDuty and AWS Security Hub.
    - If you cannot use the same management accounts across all of the services, then after you enable Detective, you can optionally create a cross-account role.
- If you are using Amazon GuardDuty, Amazon Detective will automatically ingest and process two weeks of historical log data upon activation.
- The management account for a behavior graph can disable Amazon Detective. When you disable Detective, the behavior graph and its associated Detective data are deleted. Deleted behavior graphs cannot be restored.
- Amazon Detective is able to analyze IAM role sessions by processing VPC flow records and CloudTrail management events from across a customer's enabled accounts, collating data about activity performed under an IAM Role into role sessions. This lets you visualize and understand the actions that users and apps have performed using the assumed roles.
- Amazon Detective vs Amazon GuardDuty vs AWS Security Hub
    - Amazon GuardDuty is a **threat detection service** that continuously monitors for malicious activity and unauthorized behavior to protect your AWS accounts and workloads.
    - With Security Hub, you have a **single place that aggregates, organizes, and prioritizes your security alerts, or findings**, from multiple AWS services, such as Amazon GuardDuty, Amazon Inspector, and Amazon Macie, as well as from AWS Partner solutions.
    - Amazon Detective simplifies the **process of investigating security findings and identifying the root cause**.
- Common Use Cases
    - Triage security findings
    - Incident investigation
    - Hunting for hidden security threats

Amazon GuardDuty

- An intelligent threat detection service. It analyzes billions of events across your AWS accounts from AWS CloudTrail (AWS user and API activity in your accounts), Amazon VPC Flow Logs (network traffic data), and DNS Logs (name query patterns).
- GuardDuty is a regional service.
- Threat detection categories
    - **Reconnaissance** -- Activity suggesting reconnaissance by an attacker, such as unusual API activity, intra-VPC port scanning, unusual patterns of failed login requests, or unblocked port probing from a known bad IP.
    - **Instance compromise** -- Activity indicating an instance compromise, such as cryptocurrency mining, backdoor command, and control activity, malware using domain generation algorithms, outbound denial of service activity, unusually high volume of network traffic, unusual network protocols, outbound instance communication with a known malicious IP, temporary Amazon EC2 credentials used by an external IP address, and data exfiltration using DNS.
    - **Account compromise** -- Common patterns indicative of account compromise include API calls from an unusual geolocation or anonymizing proxy, attempts to disable AWS CloudTrail logging, changes that weaken the account password policy, unusual instance or infrastructure launches, infrastructure deployments in an unusual region, and API calls from known malicious IP addresses.
- Amazon GuardDuty provides three severity levels (Low, Medium, and High) to allow you to prioritize response to potential threats.
- CloudTrail Event Source
    - Currently, GuardDuty only analyzes CloudTrail management events. (Read about types of CloudTrail trails for more information)
    - GuardDuty processes all CloudTrail events that come into a region, including global events that CloudTrail sends to all regions, such as AWS IAM, AWS STS, Amazon CloudFront, and Route 53.
- VPC Flow Logs Event Source
    - VPC Flow Logs capture information about the IP traffic going to and from Amazon EC2 network interfaces in your VPC.
- DNS Logs Event Source
    - If you use AWS DNS resolvers for your EC2 instances (the default setting), then GuardDuty can access and process your request and response DNS logs through the internal AWS DNS resolvers. Using other DNS resolvers will not provide GuardDuty access to its DNS logs.
- GuardDuty vs Macie
    - Amazon GuardDuty provides broad protection of your AWS accounts, workloads, and data by helping to identify threats such as attacker reconnaissance, instance compromise, and account compromise. Amazon Macie helps you protect your data in Amazon S3 by helping you classify what data you have, the value that data has to the business, and the behavior associated with access to that data.

- GuardDuty Findings
  - GuardDuty generates **findings** when it detects unexpected and potentially malicious activity in your AWS environment. These are viewable via Console, GuardDuty CLI or API operations.
  - A Finding's summary includes:
    - **Finding type** – a concise yet readable description of the potential security issue.
    - **Severity** – a finding's assigned severity level of either High, Medium, or Low.
    - **Region** – the AWS region in which the finding was generated.
    - **Count** – the number of times GuardDuty generated the finding after you enabled GuardDuty in your AWS account.
    - **Account ID** – the ID of the AWS account in which the activity took place that prompted GuardDuty to generate this finding.
    - **Resource ID** – the ID of the AWS resource against which the activity took place that prompted GuardDuty to generate this finding.
    - **Threat list name** - the name of the threat list that includes the IP address or the domain name involved in the activity that prompted GuardDuty to generate the finding.
    - **Last seen** – the time (your local timezone if checked through the console, and UTC if checked through CLI or API) at which the activity took place that prompted GuardDuty to generate this finding.
  - A finding's **Resource affected** section includes:
    - **Resource role** – a value that usually is set to **Target** because the affected resource can be a potential target of an attack.
    - **Resource type** – the type of the affected resource. This value is either **AccessKey** or **Instance**.
    - **Instance ID** – the ID of the EC2 instance involved in the activity that prompted GuardDuty to generate the finding.
    - **Port** – the port number for the connection used during the activity that prompted GuardDuty to generate the finding.
    - **Access key ID** – access key ID of the user engaged in the activity that prompted GuardDuty to generate the finding.
    - **Principal ID** – the principal ID of the user engaged in the activity that prompted GuardDuty to generate the finding.
    - **User type** – the type of user engaged in the activity that prompted GuardDuty to generate the finding.
    - **User name** – The name of the user engaged in the activity that prompted GuardDuty to generate the finding.
  - A finding's **Action** section includes:
    - **Action type** – the finding activity type. This value can be one of the following: NETWORK_CONNECTION, AWS_API_CALL, PORT_PROBE, or DNS_REQUEST.
    - **API** – the name of the API operation that was invoked and thus prompted GuardDuty to generate this finding.
    - **Service name** – the name of the AWS service (GuardDuty) that generated the finding.

- **Connection direction** – the network connection direction observed in the activity that prompted GuardDuty to generate the finding. The values can be INBOUND, OUTBOUND, and UNKNOWN.
- **Protocol** – the network connection protocol observed in the activity that prompted GuardDuty to generate the finding.

- A finding's **Actor** section includes:
  - **Location** – location information of the IP address involved in the activity that prompted GuardDuty to generate the finding.
  - **Organization** – ISP organization information of the IP address involved in the activity that prompted GuardDuty to generate the finding.
  - **IP address** – the IP address involved in the activity that prompted GuardDuty to generate the finding.
  - **Port** – the port number involved in the activity that prompted GuardDuty to generate the finding.
  - **Domain** – the domain involved in the activity that prompted GuardDuty to generate the finding.
- A finding's **Details** section includes:
  - **ThreatPurpose** - describes the primary purpose of a threat or a potential attack. Can have the following values:
    - **Backdoor** - this value indicates that the attack has compromised an AWS resource and is capable of contacting its home command and control (C&C) server to receive further instructions for malicious activity.
    - **Behavior** - this value indicates that GuardDuty is detecting activity or activity patterns that are different from the established baseline for a particular AWS resource.
    - **Cryptocurrency** - this value indicates that GuardDuty is detecting software that is associated with cryptocurrencies.
    - **Pentest** - sometimes owners of AWS resources or their authorized representatives intentionally run tests against AWS applications to find vulnerabilities, like open security groups or access keys that are overly permissive. These pen tests are done in an attempt to identify and lock down vulnerable resources before they are discovered by attackers.
    - **Persistence** - this value indicates that a principal in your AWS environment is exhibiting behavior that is different from the established baseline. Such as, a principal has no prior history of updating network configuration settings, or updating policies or permissions attached to AWS users or resources.
    - **Policy** - this value indicates that your AWS account is exhibiting behavior that goes against recommended security best practices.
    - **PrivilegeEscalation** - this value informs you that a specific principal in your AWS environment is exhibiting behavior that can be indicative of a privilege escalation attack.

        - **Recon** - this value indicates that a reconnaissance attack is underway, scoping out vulnerabilities in your AWS environment by probing ports, listing users, database tables, and so on.
        - **ResourceConsumption** - this value indicates that a principal in your AWS environment is exhibiting behavior that is different from the established baseline. Such as a principal has no prior history of launching EC2 instances.
        - **Stealth** - this value indicates that an attack is actively trying to hide its actions and its tracks.
        - **Trojan** - this value indicates that an attack is using Trojan programs that silently carry out malicious activity. Sometimes this software takes on the appearance of a legitimate program. Sometimes users accidentally run this software. Other times this software might run automatically by exploiting a vulnerability.
        - **UnauthorizedAccess** - this value indicates that GuardDuty is detecting suspicious activity or a suspicious activity pattern by an unauthorized individual.
      - **ResourceTypeAffected** - describes which AWS resource is identified in this finding as the potential target of an attack. Currently, only EC2 instances and principals (and their credentials) can be identified as affected resources in GuardDuty findings.
      - **ThreatFamilyName** - describes the overall threat or potential malicious activity that GuardDuty is detecting.
      - **ThreatFamilyVariant** - describes the specific variant of the **ThreatFamily** that GuardDuty is detecting. Attackers often slightly modify the functionality of the attack, thus creating new variants.
      - **Artifact** - describes a specific resource that is owned by a tool that is used in the attack.
    - You can create filters for your GuardDuty findings.
      - A *suppression rule* is a filter used to automatically archive new findings. After you create a suppression rule, new findings that match the criteria defined in the rule are automatically archived.
    - GuardDuty supports exporting active findings to Amazon EventBridge (Amazon CloudWatch Events) and, optionally, to an Amazon S3 bucket. New Active findings that GuardDuty generates are automatically exported within about 5 minutes after the finding is generated.
- Trusted IP Lists and Threat Lists
    - **Trusted IP lists** consist of IP addresses that you have **whitelisted for secure communication** with your AWS infrastructure and applications. GuardDuty does not generate findings for IP addresses on trusted IP lists.
    - At any given time, you can have only one uploaded trusted IP list per AWS account per region.
    - **Threat lists** consist of known **malicious** IP addresses. GuardDuty generates findings based on threat lists.
    - At any given time, you can have up to six uploaded threat lists per AWS account per region.

- Pricing

Pricing is based on the quantity of AWS CloudTrail Events analyzed (per 1,000,000 events) and the volume of Amazon VPC Flow Log and DNS Log data analyzed (per GB).

References:
https://aws.amazon.com/guardduty/
https://aws.amazon.com/guardduty/faqs/
https://docs.aws.amazon.com/guardduty/latest/ug/what-is-guardduty.html
https://www.youtube.com/watch?time_continue=7&v=o2YaIsps5LY

Amazon Inspector

- Amazon Inspector is an automated security assessment service that continuously scans AWS workloads to identify and remediate security risks. It integrates with AWS services to provide security assessments for EC2 instances, Amazon Elastic Container Registry (ECR) images, and AWS Lambda functions without requiring manual intervention.
- The inspector uses IAM *service-linked roles*.

Features

- **Continuous and Automated Scanning** – Unlike Inspector Classic, which requires manual assessment runs, Inspector v2 automatically scans workloads as soon as they are deployed.
- **Agentless for ECR and Lambda** – No need to install an agent for container (ECR) and Lambda function scanning.
- **EC2 Scanning with SSM Agent** – For EC2 instances, an AWS Systems Manager (SSM) Agent is used instead of a dedicated Inspector agent.
- **Integration with AWS Security Hub** – Findings from Inspector v2 are automatically sent to AWS Security Hub, making it easier to manage vulnerabilities across multiple AWS accounts.
- **Support for AWS Organizations** – Security teams can centrally manage vulnerability scanning across multiple accounts.
- **Risk Scoring and Prioritization** – Findings are assigned Common Vulnerability Scoring System (CVSS) scores to prioritize risks effectively.
- Amazon Inspector v2 continuously scans workloads using an AWS-managed vulnerability database, identifying security risks based on the latest CVE information and AWS best practices.
- **Automation** – Amazon Inspector v2 automates security vulnerability assessments throughout the development and deployment pipeline by continuously scanning workloads, detecting vulnerabilities, and integrating with AWS Security Hub for proactive risk management.

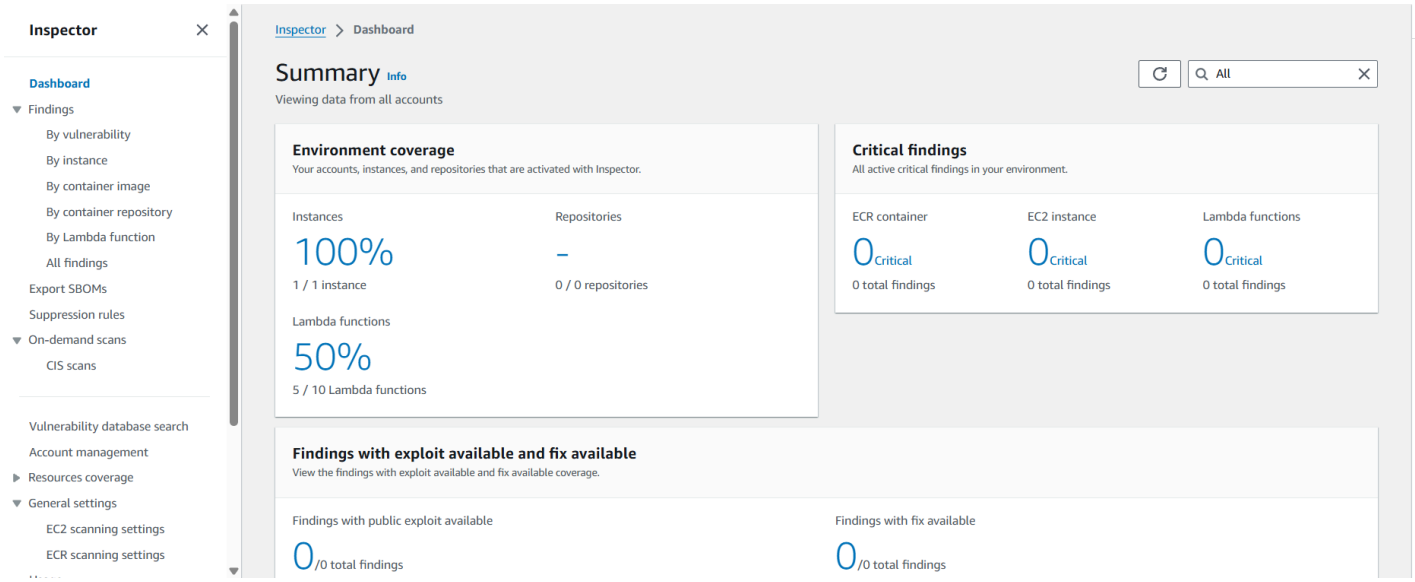
Scanning Capabilities

1. **EC2 Instance Scanning**
    a. Amazon Inspector continuously scans EC2 instances for vulnerabilities. This includes checking for common vulnerabilities and exposures (CVEs), operating system and programming language package vulnerabilities, and issues related to network reachability and unintended network exposure.
    b. Scanning is performed using the Systems Manager (SSM) Agent installed on the EC2 instance or through Amazon Elastic Block Store (EBS) snapshots of instances.
    c. By default, EC2 scanning is enabled in a hybrid mode, which means it can scan instances both with and without an agent.
2. **ECR Image Scanning**

   a. Amazon Inspector provides automatic vulnerability scanning for container images stored in the Amazon Elastic Container Registry (ECR).
   b. Once you activate ECR scanning, all images pushed within the last 30 days or pulled within the last 90 days are initially scanned.
   c. The inspector continues to monitor these images for vulnerabilities over a 90-day period, although this duration can be adjusted. You can configure the scanning behavior to run on push or periodically for selected repositories.
3. **Lambda Function Scanning**
   a. AWS Lambda functions, including their dependencies, are automatically scanned by Amazon Inspector for security vulnerabilities.
   b. Lambda function scanning happens in two stages:
      i. It scans newly deployed functions and rescans them when updated or
      ii. when new CVEs are published.
   c. There is also an option to activate **Lambda Code Scanning**, which scans for vulnerabilities within the custom application code in your Lambda functions. When both scanning options are enabled, they work together to provide comprehensive vulnerability assessments for Lambda functions and their code.

How Amazon Inspector Works

- **EC2 Instances** - When EC2 instances are launched, Amazon Inspector automatically scans them if they have the SSM Agent installed.
- **ECR Containers** - When new container images are pushed to Amazon ECR, Inspector scans them immediately to detect vulnerabilities.
- **Lambda Functions** - If Lambda functions are created or updated, Inspector checks for security issues in their dependencies.
- **Findings** - All findings are sent to the AWS Security Hub and can trigger Amazon EventBridge alerts for automated responses.



Comparison: Inspector v1 vs. Inspector v2

Feature	Inspector Classic (v1)	Inspector v2
Scanning Approach	**Manual assessment runs**	**Continuous, automated scanning**
Agent Requirement	**Requires Inspector Agent**	**Uses SSM Agent (for EC2 only)**
Supported Resources	**EC2 only**	**EC2, ECR, Lambda**
Findings Management	**No integration with Security Hub**	**Integrated with AWS Security Hub**
Organization-Wide Scanning	**No**	**Supports AWS Organizations**
ECR Image Scanning	**No**	**Yes**
Lambda Function Scanning	**No**	**Yes**

Viewing and Managing Findings

- **Severity Levels** - Findings are categorized based on severity: Critical, High, Medium, Low, and Informational.
- **Access** - They are accessible through the **Amazon Inspector Console, AWS Security Hub, or AWS CLI**.
- **Remediation** - The Inspector provides remediation recommendations and can trigger notifications via **Amazon EventBridge**.

Amazon Inspector pricing is based on:

- **EC2 instance assessments** – Charged per instance scanned per month.
- **ECR image assessments** – Charged per repository scanned per month.
- **Lambda function assessments** – Charged per function scanned per month.
- Free tier available for **first 15 days** after enabling Inspector.

References:
https://docs.aws.amazon.com/inspector/latest/user/what-is-inspector.html
https://docs.aws.amazon.com/inspector/latest/user/scanning-resources.html
https://aws.amazon.com/inspector/pricing/
https://aws.amazon.com/inspector/faqs/

Amazon Macie

- A security service that uses machine learning to automatically discover, classify, and protect sensitive data in AWS. Macie recognizes sensitive data such as personally identifiable information (PII) or intellectual property.
- Amazon Macie allows you to achieve the following:
    - Identify and protect various data types, including PII, PHI, regulatory documents, API keys, and secret keys
    - Verify compliance with automated logs that allow for instant auditing
    - Identify changes to policies and access control lists
    - Observe changes in user behavior and receive actionable alerts
    - Receive notifications when data and account credentials leave protected zones
    - Detect when large quantities of business-critical documents are shared internally and externally
- Concepts
    - An **Alert** is a notification about a potential security issue that Macie discovers. Alerts appear on the Macie console and provide a comprehensive narrative about all activity that occurred over the last 24 hours.
        - Basic alerts – Alerts that are generated by the security checks that Macie performs. There are two types of basic alerts in Macie:
            - Managed (curated by Macie) basic alerts that you can't modify. You can only enable or disable the existing managed basic alerts.
            - Custom basic alerts that you can create and modify to your exact specifications.
        - Predictive alerts – Automatic alerts based on activity in your AWS infrastructure that deviates from the established normal activity baseline. More specifically, Macie continuously monitors IAM user and role activity in your AWS infrastructure and builds a model of the normal behavior. It then looks for deviations from that normal baseline, and when it detects such activity, it generates automatic predictive alerts.
    - **Data source** is the origin or location of a set of data.
        - AWS CloudTrail event logs and errors, including Amazon S3 object-level API activity. You can't modify existing or add new CloudTrail events to the list that Macie manages. You can enable or disable the supported CloudTrail events, thus instructing Macie to either include or exclude them in its data security process.
        - Amazon S3 objects. You can integrate Macie with your S3 buckets and/or specify S3 prefixes
    - **User**, in the context of Macie, a user is the AWS Identity and Access Management (IAM) identity that makes the request.
- There are certain file formats that Macie does not support, such as wav files.
- Once Macie begins monitoring your data, it uses several **automatic content classification methods** to identify and prioritize your sensitive and critical data and to accurately assign business value to your

data. Each classification has a designated risk level between 1 and 10, with 10 being the highest risk and 1 being the lowest. These methods include:

- **Content Type Classification** - Macie uses an identifier that is embedded in the file header of your data objects. Macie can assign only one content type to an object. You can't modify existing or add new content types. You can only enable or disable any existing content types, thus enabling or disabling Macie to assign them to your objects during the classification process.
- **File Extension Classification** - Macie offers a set of managed file extensions. Macie can assign only one file extension to an object. You can't modify existing or add new file extensions. You can enable or disable any existing file extensions, thus enabling or disabling Macie to assign them to your objects during the classification process.
- **Theme Classification** - Object classification by theme is based on keywords that Macie searches for as it examines the contents of data objects. Macie can assign one or more themes to an object. You can't modify existing or add new themes. You can enable or disable any existing themes, thus enabling or disabling Macie to assign them to your objects during the classification process.
- **Regex Classification** - Macie offers a set of managed regexes. Object classification by regex is based on specific data or data patterns that Macie searches for as it examines the contents of data objects. Macie can assign one or more regexes to an object. You can't modify existing or add new regexes. You can enable or disable any existing regexes, thus enabling or disabling Macie to assign them to your objects during the classification process.
- **PII Classification** - Object classification by personally identifiable information (PII) is based on recognizing any personally identifiable artifacts based on industry standards such as NIST-80-122 and FIPS 199.
- **Support Vector Machine–Based Classifier** - It classifies content inside your S3 objects (text, token n-grams, and character n-grams) that Macie monitors and their metadata features (document length, extension, encoding, headers) to accurately classify documents based on content.

- You can use the **Research** tab in the Macie console to construct and run queries in the query parser and conduct in-depth investigative research of your data and activity that Macie monitors.
- If you disable Macie, the following actions occur:
  - It no longer has access to the resources in the management account and all member accounts. You must add member accounts again if you decide to reenable Macie.
  - It stops processing the resources in the management account and all member accounts. After Macie is disabled, the metadata that Macie collected while monitoring the data in your management and member accounts is deleted. Within 90 days from disabling Macie, all of this metadata is expired from the Macie system backups.

- Other Additional Features
  - You can scan Amazon S3 buckets across multiple AWS accounts and perform scoping of scans by object prefix.
  - An estimation of the costs of these job runs is sent to you for review before you run them.
  - Once a job is submitted, findings are generated in the Amazon Macie console and sent out through Amazon EventBridge, where sensitive data location information is included in the findings. This allows for the identification of sensitive data within objects using detail such as line numbers, page numbers, record index, or column and row numbers.
- Pricing
  - You are charged based on the amount of content classified, and the amount of AWS CloudTrail events assessed by Amazon Macie for anomalies (both Management API activity and Amazon S3 object-level API activity).
  - Amazon Macie stores the generated metadata of classified S3 objects for 30 days at no additional cost. Additional monthly fees will be incurred if you choose the optional Extended Data Retention feature.

References:
https://aws.amazon.com/macie/
https://docs.aws.amazon.com/macie/latest/userguide/what-is-macie.html
https://aws.amazon.com/macie/faq/
https://www.youtube.com/watch?v=LCjX2rsQ2wA

AWS Artifact

- A self-service central repository of AWS' security and compliance reports and select online agreements.
- An **audit artifact** is a piece of evidence that demonstrates that an organization is following a documented process or meeting a specific requirement (business compliant).
- **AWS Artifact Reports** include the following:
    - ISO,
    - Service Organization Control (SOC) reports,
    - Payment Card Industry (PCI) reports,
    - and certifications that validate the implementation and operating effectiveness of AWS security controls.
- **AWS Artifacts Agreements** include
    - the Nondisclosure Agreement (NDA)
    - the Business Associate Addendum (BAA), which typically is required for companies that are subject to the HIPAA Act to ensure that protected health information (PHI) is appropriately safeguarded.
- **All AWS Accounts with AWS Artifact IAM permissions have access to AWS Artifact**. Root users and IAM users with admin permissions can download all audit artifacts available to their account by agreeing to the associated terms and conditions. You will need to grant IAM users with non-admin permissions access to AWS Artifact.
- To use organization agreements in AWS Artifact, your organization must be enabled for **all features**.
- AWS Artifact Agreements
    - AWS Artifact Account Agreements apply only to the individual account you used to sign into AWS.
    - AWS Artifact Organization Agreements apply to all accounts in an organization created through AWS Organizations, including the organization's management account and all member accounts. Only the management account in an organization can accept agreements in AWS Artifact Organization Agreements.
    - Management accounts and member accounts of an Organization can have AWS Artifact Account Agreements and AWS Artifact Organization Agreements of the same type in place at the same time.
    - If you have accounts in separate organizations that you want covered by an agreement, you must log in to each organization's management account and accept the relevant agreements through AWS Artifact Organization Agreements.
    - Terminating the organization agreement does not terminate the account agreement.
    - When a member account is removed from an organization (e.g. by leaving the organization, or by being removed from the organization by the management account), any organization agreements accepted on its behalf will no longer apply to that member account.
- Business Associate Addendum (BAA)
    - You can accept the AWS BAA for your individual account, or if you are a management account in an organization, you can accept the AWS BAA on behalf of all accounts in your organization.

    - Upon accepting the AWS BAA in AWS Artifact Agreements, you will instantly designate your AWS account(s) for use in connection with protected health information (PHI) and HIPAA.
    - If you terminate an online BAA under the Account agreements tab in AWS Artifact, the account you used to sign into AWS will immediately cease to be a HIPAA Account, unless it was also covered by an organization BAA.
    - If you are a user of a management account and terminate an online BAA in AWS Artifact, all accounts within your organization will immediately be removed as HIPAA Accounts, unless they were covered by individual account BAAs.
    - If you have both an account BAA and an organization BAA in place at the same time, the terms of the organization BAA will apply instead of the terms of the account BAA.
- AWS Australian Notifiable Data Breach Addendum (ANDB Addendum)
    - Using the management account of your organization, you can use the Organization agreements tab in AWS Artifact Agreements to accept an ANDB Addendum on behalf of all existing and future member accounts in your organization.
    - When both the account ANDB Addendum and organizations ANDB Addendum are accepted, the organizations ANDB Addendum will apply instead of the account ANDB Addendum.
    - If you terminate an account ANDB Addendum under the Account agreements tab in AWS Artifact, the AWS account you used to sign into AWS Artifact will not be covered by an ANDB Addendum with AWS, unless it is also covered by an organizations ANDB Addendum.
    - If you are a user of a management account and terminate an organizations ANDB Addendum within the Organization agreements tab in AWS Artifact, the AWS accounts in that AWS organization will not be covered by an ANDB Addendum with AWS, unless they are covered by an account ANDB Addendum
- Most errors you receive from AWS Artifact can be resolved by adding the necessary IAM permissions.

References:
https://aws.amazon.com/artifact/
https://docs.aws.amazon.com/artifact/latest/ug/what-is-aws-artifact.html
https://aws.amazon.com/artifact/faq/

AWS Certificate Manager

- A service that lets you easily provision, manage, and deploy public and private SSL/TLS certificates for use with AWS services and your internal connected resources. SSL/TLS certificates are used to secure network communications and establish the identity of websites over the Internet as well as resources on private networks.
- ACM is integrated with the following services:
    - Elastic Load Balancing
    - Amazon CloudFront - To use an ACM certificate with CloudFront, you must request or import the certificate in the US East (N. Virginia) region.
    - AWS Elastic Beanstalk
    - Amazon API Gateway
    - AWS CloudFormation
- AWS Certificate Manager manages the renewal process for the certificates managed in ACM and used with ACM-integrated services.
- You can import your own certificates into ACM, however, you have to renew these yourself.
- Concepts
    - ACM Certificate are X.509 version 3 certificates. Each is valid for **13 months**.
    - When you request an ACM certificate, you must validate that you own or control all of the domains that you specify in your request.
    - **Each ACM Certificate must include at least one fully qualified domain name (FQDN)**. You can add additional names if you want to.
    - You can create an ACM Certificate containing a wildcard name (*.example.com) that can protect several sites in the same domain (subdomains).
    - You cannot download the private key for an ACM Certificate.
    - The first time you request or import a certificate in an AWS region, ACM creates an AWS managed key (KMS key) in AWS KMS with the alias aws/acm. This KMS key is unique in each AWS account and each AWS region. ACM uses this KMS key to encrypt the certificate's private key.
    - You cannot add or remove domain names from an existing ACM Certificate. Instead, you must request a new certificate with the revised list of domain names.
    - You cannot delete an ACM Certificate that is being used by another AWS service. To delete a certificate that is in use, you must first remove the certificate association.
    - Applications and browsers trust public certificates automatically by default, whereas an administrator must explicitly configure applications to trust private certificates.
- Types of Certificates For Use With ACM
    - **Public certificates**
        - ACM manages the renewal and deployment of public certificates used with ACM-integrated services.

        - You cannot install public ACM certificates directly on your website or application, only for integrated services.
    - **Private certificates**
        - ACM Private CA provides three ways to create and manage private certificates. 1) You can choose to delegate private certificate management to ACM. When used in this way, ACM can automatically renew and deploy private certificates used with ACM-integrated services. 2) You can export private certificates from ACM and use them with EC2 instances, containers, on-premises servers, and IoT devices. ACM Private CA automatically renews these certificates and sends an Amazon CloudWatch notification when the renewal is completed. You can write client-side code to download renewed certificates and private keys and deploy them with your application. 3) ACM Private CA gives you the flexibility to create your own private keys, generate a certificate signing request (CSR), issue private certificates from your ACM Private CA, and manage the keys and certificates yourself. You are responsible for renewing and deploying these private certificates.
    - **Imported certificates**
        - If you want to use a third-party certificate with ACM-integrated services, you may import it into ACM using the AWS Management Console, AWS CLI, or ACM APIs. ACM does not manage the renewal process for imported certificates. You are responsible for monitoring the expiration date of your imported certificates and for renewing them before they expire. You can use the AWS Management Console to monitor the expiration dates of imported certificates and import a new third-party certificate to replace an expiring one.
    - **CA certificates**
        - ACM private CA can issue certificates to identify private certificate authorities. These certificates allow CA administrators to create a private CA hierarchy, which provides strong security and restrictive access controls for the most-trusted root CA at the top of the trust chain, while allowing more permissive access and bulk certificate issuance for subordinate CAs lower in the chain.

ACM Private Certificate Authority

- ACM PCA allows you to create a private certificate authority (CA) and then use ACM to issue private certificates.
- With ACM Private CA, you can create complete CA hierarchies, including root and subordinate CAs. A CA hierarchy provides strong security and restrictive access controls for the most-trusted root CA at the top of the trust chain, while allowing more permissive access and bulk certificate issuance for subordinate CAs lower in the chain.
- A private CA handles the issuance, validation, and revocation of private certificates within a private network. It is comprised of two major components: The first is the **CA certificate**, a cryptographic building block upon which certificates can be issued. The second is **a set of run-time services** for maintaining revocation information through the **Certificate Revocation List** (CRL).

- Benefits of a Private CA
  - Create certificates with any subject name you want.
  - Create certificates with any expiration date you want.
  - Use any supported private key algorithm and key length.
  - Use any supported signing algorithm.
  - Configure certificates in bulk using templates.
- Automatic renewal is not available for ACM Private CA certificates for which ACM does not create the private key and certificate signing request (CSR).
- You cannot copy private CAs between Regions. To use private CAs in more than one Region, you must create your CAs in those Regions.

Domain Verification for Certificates

- Before the Amazon certificate authority can issue a certificate for your site, AWS Certificate Manager must verify that you own or control all of the domain names that you specified in your request. You can choose either **email validation** or **DNS validation** when you request a certificate.
- For DNS validation, ACM uses **CNAME (Canonical Name) records to validate** that you own or control a domain.
- In the DNS validation console page, ACM will provide you a CNAME record that you must add to your DNS database, whether it be Route 53 or other hosts.
- For email validation, ACM sends email to the 3 contact addresses listed in WHOIS and to 5 common system addresses for each domain that you specify. To validate it, one of the recipients must click on the approval link.

Pricing

- There is no additional charge for provisioning public or private SSL/TLS certificates you use with ACM-integrated services, such as Elastic Load Balancing and API Gateway.
- You are billed for each active ACM Private CA per month pro-rated
- For private certificates, ACM Private CA allows you to pay monthly for the service and certificates you create. You pay less per certificate as you create more private certificates.

References:
https://aws.amazon.com/certificate-manager/
https://aws.amazon.com/certificate-manager/faqs/
https://docs.aws.amazon.com/acm/latest/userguide/acm-overview.html
https://docs.aws.amazon.com/acm-pca/latest/userguide/PcaWelcome.html

AWS Directory Service

For Microsoft Active Directory

- Also known as **AWS Managed Microsoft AD**, the service enables your directory-aware workloads and AWS resources to use **managed Active Directory** in the AWS Cloud.
- The service is built on actual Microsoft Active Directory and powered by Windows Server 2012 R2.
- AWS Managed Microsoft AD is your best choice if you need actual Active Directory features to support AWS applications or Windows workloads, including Amazon RDS for Microsoft SQL Server. It's also best if you want a standalone AD in the Cloud that supports Office 365 or you need an LDAP directory to support your Linux applications.
- Concepts
    - AWS Managed Microsoft AD provides multiple directory choices for customers who want to use existing Microsoft AD or Lightweight Directory Access Protocol (LDAP)–aware applications in the cloud.
    - When you create a directory, AWS Directory Service creates two domain controllers and adds the DNS service on your behalf. The domain controllers are created in different subnets in a VPC
    - When creating a directory, you need to provide some basic information such as a fully qualified domain name (FQDN) for your directory, Administrator account name, and password, and the VPC you want the directory to be attached to.
    - AWS does not provide Windows PowerShell access to directory instances, and it restricts access to directory objects, roles, and groups that require elevated privileges.
    - AWS Managed Microsoft AD does not allow direct host access to domain controllers via Telnet, Secure Shell (SSH), or Windows Remote Desktop Connection.
    - When you create an AWS Managed Microsoft AD directory, you are assigned an organizational unit (OU) and an administrative account with delegated administrative rights for the OU.
    - AWS Managed Microsoft AD directories are deployed across **two Availability Zones in a region** by default and connected to your Amazon VPC.
    - You cannot configure the storage, CPU, or memory parameters of your AWS Managed Microsoft AD directory.
- Active Directory Schema
    - A **schema** is the definition of attributes and classes that are part of a distributed directory and is similar to fields and tables in a database. Schemas include a set of rules which determine the type and format of data that can be added or included in the database.
    - Attributes, classes, and objects are the basic elements that are used to build object definitions in the schema.
        - Each schema attribute, which is similar to a field in a database, has several properties that define the characteristics of the attribute.
        - The classes are analogous to tables in a database and also have several properties to be defined.

        - Each class and attribute must have an Object ID that is unique for all of your objects. Software vendors must obtain their own Object ID to ensure uniqueness.
        - Some attributes are linked between two classes with forward and back links, such as groups. A group shows you the members of the group; while a member shows what groups it belongs to.
- Features
    - AWS Managed Microsoft AD is deployed in HA and across multiple Availability Zones. You can also scale out your directory by deploying additional domain controllers.
    - AWS Managed Microsoft AD runs on AWS managed infrastructure with monitoring that automatically detects and replaces domain controllers that fail.
    - Data replication and automated daily snapshots are configured for you.
    - You can integrate AWS Managed Microsoft AD easily with your existing Active Directory by using **Active Directory trust relationships**.
    - Allows seamless domain join for new and existing Amazon EC2 for Windows Server instances.
    - AWS Managed Microsoft AD can also provide a single directory for all kinds of workloads (EC2, RDS, WorkSpaces, etc).
    - The service supports schema extensions that you submit to the service in the form of a LDAP Data Interchange Format (LDIF) file.
    - You can configure Amazon SNS to receive email and text messages when the status of your AWS Directory Service changes.
    - You can configure SAML 2.0–based authentication with cloud applications using AWS Directory Service.
    - You can use AWS Managed Microsoft AD as a resource forest that contains primarily computers and groups with trust relationships to your on-premises directory. This enables your users to access AWS applications and resources with their on-premises AD credentials.
- Microsoft AD Prerequisites
    - A VPC with at least two subnets. Each of the subnets must be in a different Availability Zone.
    - The necessary ports for the domain controllers that AWS Directory Service creates for you should be open to allow them to communicate with each other.
    - The VPC must have default hardware tenancy.
    - AWS Directory Service does not support using NAT with Active Directory.
- Two Editions of AWS Managed Microsoft AD
    - Both Standard Edition and Enterprise Edition can be used as your organization's primary directory to manage users, devices, and computers.
    - You also can use both editions to create resource forests and extend your on-premises AD to the AWS Cloud. **Resource forests** use a trust relationship with your on-premises AD to enable you to access AWS applications and resources with your on-premises AD credentials.
    - Both editions also support the creation of additional domain controllers to improve the redundancy and performance of your managed directory.
    - Unique to Standard Edition

        - Optimized to be a primary directory for small and midsize businesses with up to 5,000 employees.
        - Provides you with enough storage capacity to support up to approximately 30,000 directory objects, such as users, groups, and computers.
    - Unique to Enterprise Edition
        - Designed to support enterprise organizations with up to approximately 500,000 directory objects.
- Seamless Domain Joins
    - **Seamless domain join** is a feature that allows you to join your Amazon EC2 for Windows Server instances seamlessly to a domain, at the time of launch and from the AWS Management Console. You can join instances to AWS Managed Microsoft AD that you launch in the AWS Cloud.
    - You cannot use the seamless domain join feature from the AWS Management Console for **existing EC2 for Windows Server** instances, but you can join existing instances to a domain using the EC2 API or by using PowerShell on the instance.
- Security and Monitoring
    - AWS Managed Microsoft AD is both HIPAA and PCI DSS compliant.
    - Manage users and devices by using native Active Directory Group Policy objects (GPOs).
    - AWS Managed Microsoft AD uses the same Kerberos-based authentication as Active Directory to deliver Single Sign-On (SSO).
    - AWS Managed Microsoft AD supports federation access for users and groups to the AWS Management Console.
    - Amazon EBS volumes used in the directory service are encrypted.
- Pricing
    - You pay only for the type and size of the managed directory that you use.
    - AWS Managed Microsoft AD allows you to use a directory in one account and share it with multiple accounts and VPCs. There is an hourly sharing charge for each additional account to which you share a directory.

Active Directory Connector

- A **proxy service** that provides an easy way to connect compatible AWS applications, such as Amazon WorkSpaces, Amazon QuickSight, and Amazon EC2 for Windows Server instances, to your existing on-premises Microsoft Active Directory.
- AD Connector is your best choice when you want to use your existing on-premises directory with compatible AWS services.
- Features
    - When users log in to the AWS applications, AD Connector forwards sign-in requests to your on-premises Active Directory domain controllers for authentication.

    - You can also join your EC2 Windows instances to your on-premises Active Directory domain through AD Connector using seamless domain join.
    - AD Connector is NOT compatible with RDS SQL Server.
    - AD Connector comes in two sizes, small and large.
    - You can spread application loads across multiple AD Connectors to scale to your performance needs. There are no enforced user or connection limits.
- AD Connector Prerequisites
    - You need to have a VPC with at least two subnets. Each of the subnets must be in a different Availability Zone.
    - The VPC must be connected to your existing network through a virtual private network (VPN) connection or AWS Direct Connect.
    - The VPC must have default hardware tenancy.
    - Your user accounts must have Kerberos pre-authentication enabled.

Simple AD

- A **standalone Microsoft Active Directory–compatible** directory from AWS Directory Service that is powered by **Samba 4**.
- You can use Simple AD as a standalone directory in the cloud to support Windows workloads that need basic AD features, compatible AWS applications, or to support Linux workloads that need LDAP service.
- Features
    - Simple AD supports basic Active Directory features such as user accounts, group memberships, joining a Linux domain or Windows based EC2 instances, Kerberos-based SSO, and group policies.
    - AWS provides monitoring, daily snapshots, and recovery as part of the service.
    - Simple AD is compatible with the following AWS applications: Amazon WorkSpaces, Amazon WorkDocs, Amazon QuickSight, and Amazon WorkMail.
    - You can also sign in to the AWS Management Console with Simple AD user accounts.
    - Simple AD does NOT support multi-factor authentication, trust relationships, DNS dynamic update, schema extensions, communication over LDAPS, PowerShell AD cmdlets, or FSMO role transfer.
    - Simple AD is NOT compatible with RDS SQL Server.
    - Simple AD is available in two sizes:
        - Small - Supports up to 500 users
        - Large - Supports up to 5,000 users
- Simple AD Prerequisites
    - Your VPC should have at least two subnets. For Simple AD to install correctly, you must install your two domain controllers in separate subnets that must be in a different Availability Zone. In addition, the subnets must be in the same Classless Inter-Domain Routing (CIDR) range.

    - The necessary ports for the domain controllers that AWS Directory Service creates for you should be open to allow them to communicate with each other.
    - The VPC must have default hardware tenancy.
- When you create a directory with Simple AD, AWS Directory Service performs the following tasks on your behalf:
    - Sets up a Samba-based directory within the VPC.
    - Creates a directory administrator account with the user name '*Administrator*' and the specified password. You use this account to manage your directory.
    - Creates a security group for the directory controllers.
    - Creates an account that has domain admin privileges.
- Simple AD forwards DNS requests to the IP address of the Amazon-provided DNS servers for your VPC. These DNS servers will resolve names configured in your Route 53 private hosted zones

Amazon Cloud Directory

- A **cloud-native directory** that can store hundreds of millions of application-specific objects with multiple relationships and schemas. Use Amazon Cloud Directory if you need a **highly scalable directory store** for your application's **hierarchical data**.
- You can organize directory objects into multiple hierarchies to support many organizational pivots and relationships across directory information.
- Concepts
    - A schema is a collection of facets that define what objects can be created in a directory and how they are organized.
    - A schema also enforces data integrity and interoperability.
    - A single schema can be applied to more than one directory at a time.
    - Amazon Cloud Directory supports uploading of a compliant **JSON file for schema creation**.
    - A directory is a schema-based data store that contains specific types of objects organized in a multi-hierarchical structure.
    - Before you can create a directory in Amazon Cloud Directory, AWS Directory Service requires that you first apply a schema to it. A directory cannot be created without a schema and typically has one schema applied to it.

References:
https://aws.amazon.com/directoryservice/features/?nc=sn&loc=2
https://docs.aws.amazon.com/clouddirectory/latest/developerguide/what_is_cloud_directory.html
https://docs.aws.amazon.com/directoryservice/latest/admin-guide/what_is.html
https://docs.aws.amazon.com/clouddirectory/latest/developerguide/what_is_cloud_directory.html

AWS Fargate

- A serverless compute engine for containers that work with both Amazon Elastic Container Service (ECS) and Amazon Elastic Kubernetes Service (EKS).
- With Fargate, no manual provisioning, patching, cluster capacity management, or any infrastructure management required.
- Use Case
  - Launching containers without having to provision or manage EC2 instances.
  - If you want a managed service for container cluster management.
- Configurations
  - Amazon ECS task definitions for Fargate require that you specify CPU and memory at the task level (task definition).
  - Amazon ECS task definitions for Fargate support the limits parameter to define the resource limits to set for a container.
  - Amazon ECS task definitions for Fargate support the awslogs, splunk, firelens, and fluentd log drivers for the log configuration.
  - When provisioned, each Fargate task receives the following storage:
    - 10 GB of Docker layer storage
    - An additional 4 GB for volume mounts.
  - Task storage is ephemeral.
  - If you have a service with running tasks and want to update their platform version, you can update your service, specify a new platform version, and choose Force new deployment. Your tasks are redeployed with the **latest** platform version.
  - If your service is scaled up without updating the platform version, those tasks receive the platform version that was specified on the service's current deployment.
- Network
  - Amazon ECS task definitions for Fargate require that the network mode is set to awsvpc. The awsvpc network mode provides each task with its own elastic network interface.
- Compliance
  - PCI DSS Level 1, ISO 9001, ISO 27001, ISO 27017, ISO 27018, SOC 1, SOC 2, SOC 3, and HIPAA
  - AWS Fargate is not yet available in AWS GovCloud.
- Pricing
  - You pay for the amount of vCPU and memory resources consumed by your containerized applications.

References:
https://aws.amazon.com/fargate/
https://aws.amazon.com/fargate/faqs/
https://docs.aws.amazon.com/AmazonECS/latest/developerguide/AWS_Fargate.html

AWS Identity and Access Management (AWS IAM)

- Control who is authenticated (signed in) and authorized (has permissions) to use resources.
- AWS account **root user** is a single sign-in identity that has complete access to all AWS services and resources in the account.
- **Features**
  - You can grant other people permission to administer and use resources in your AWS account without having to share your password or access key.
  - You can grant different permissions to different people for different resources.
  - You can use IAM features to securely provide credentials for applications that run on EC2 instances which provide permissions for your applications to access other AWS resources.
  - You can add two-factor authentication to your account and to individual users for extra security.
  - You can allow users to use **identity federation** to get temporary access to your AWS account.
  - You receive AWS CloudTrail log records that include information about **IAM identities** who made requests for resources in your account.
  - You use an **access key** (an access key ID and secret access key) to make programmatic requests to AWS. An Access Key ID and Secret Access Key can only be uniquely generated once and must be regenerated if lost.
  - IAM has been validated as being compliant with Payment Card Industry (PCI) Data Security Standard (DSS).
  - IAM is *eventually consistent*. IAM achieves high availability by replicating data across multiple servers within Amazon's data centers around the world.
  - IAM and AWS Security Token Service (STS) are offered at no additional charge.
  - Your unique account sign-in page URL: https://*My_AWS_Account_ID*.signin.aws.amazon.com/console/
  - You can use IAM tags to add custom attributes to an IAM user or role using a tag key–value pair.
  - You can generate and download a credential report that lists all users on your AWS account. The report also shows the status of passwords, access keys, and MFA devices.
- **Infrastructure Elements**
  - **Principal**
    - An entity that can make a request for an action or operation on an AWS resource. Users, roles, federated users, and applications are all AWS principals.
    - Your AWS account root user is your *first principal*.
  - **Request**
    - When a principal tries to use the AWS Management Console, the AWS API, or the AWS CLI, that principal sends a *request* to AWS.
    - Requests includes the following information:
      - **Actions or operations** – the actions or operations that the principal wants to perform.

        - **Resources** – the AWS resource object upon which the actions or operations are performed.
        - **Principal** – the user, role, federated user, or application that sent the request. Information about the principal includes the policies that are associated with that principal.
        - **Environment data** – information about the IP address, user agent, SSL enabled status, or the time of day.
        - **Resource data** – data related to the resource that is being requested.
    - **Authentication**
        - To authenticate from the console as a user, you must sign in with your user name and password.
        - To authenticate from the API or AWS CLI, you must provide your access key and secret key.
    - **Authorization**
        - AWS uses values from the *request context* to check for policies that apply to the request. It then uses the policies to determine whether to allow or deny the request.
        - Policies types can be categorized as *permissions policies* or *permissions boundaries*.
            - *Permissions policies* define the permissions for the object to which they're attached. These include identity-based policies, resource-based policies, and ACLs.
            - *Permissions boundary* is an advanced feature that allows you to use policies to limit the maximum permissions that a principal can have.
        - To provide your users with permissions to access the AWS resources in their own account, you need **identity-based policies**.
        - **Resource-based policies** are for granting cross-account access.
        - Evaluation logic rules for policies:
            - By default, **all requests are denied**.
            - An *explicit allow* in a permissions policy overrides this default.
            - A *permissions boundary* overrides the allow. If there is a permissions boundary that applies, that boundary must allow the request. Otherwise, it is implicitly denied.
            - An explicit deny in any policy overrides any allows.
    - **Actions or Operations**
        - Operations are defined by a service, and include things that you can do to a resource, such as viewing, creating, editing, and deleting that resource.
    - **Resource**
        - An object that exists within a service. The service defines a set of actions that can be performed on each resource.
- **Users**
    - **IAM Users**

        - Instead of sharing your root user credentials with others, you can create individual **IAM users** within your account that correspond to users in your organization. IAM users are not separate accounts; they are users within your account.
        - Each user can have its own password for access to the AWS Management Console. You can also create an individual access key for each user so that the user can make programmatic requests to work with resources in your account.
        - By default, a brand new IAM user has **NO permissions** to do anything.
        - Users are global entities.
    - **Federated Users**
        - If the users in your organization already have a way to be authenticated, you can federate those user identities into AWS.
    - **IAM Groups**
        - An IAM group is a collection of IAM users.
        - You can organize IAM users into IAM groups and attach access control policies to a group.
        - A user can belong to multiple groups.
        - Groups cannot belong to other groups.
        - Groups do not have security credentials, and cannot access web services directly.
    - **IAM Role**
        - A role does not have any credentials associated with it.
        - An IAM user can assume a role to temporarily take on different permissions for a specific task. A role can be assigned to a federated user who signs in by using an external identity provider instead of IAM.
        - **AWS service role** is a role that a service assumes to perform actions in your account on your behalf. This service role must include all the permissions required for the service to access the AWS resources that it needs.
            - **AWS service role for an EC2 instance** is a special type of service role that a service assumes to launch an EC2 instance that runs your application. This role is assigned to the EC2 instance when it is launched.
            - **AWS service-linked role** is a unique type of service role that is linked directly to an AWS service. Service-linked roles are predefined by the service and include all the permissions that the service requires to call other AWS services on your behalf.
        - An instance profile is a container for an IAM role that you can use to pass role information to an EC2 instance when the instance starts.
    - Users or groups can have multiple policies attached to them that grant different permissions.

When to Create IAM User	When to Create an IAM Role
You created an AWS account and you're the only person who works in your account.	You're creating an application that runs on an Amazon EC2 instance and that application makes requests to AWS.
Other people in your group need to work in your AWS account, and your group is using no other identity mechanism.	You're creating an app that runs on a mobile phone and that makes requests to AWS.
You want to use the command-line interface to work with AWS.	Users in your company are authenticated in your corporate network and want to be able to use AWS without having no sign in again (federate into AWS).

- **Policies**
  - Most permission policies are JSON policy documents.
  - The IAM console includes *policy summary tables* that describe the access level, resources, and conditions that are allowed or denied for each service in a policy.
  - The *policy summary table* includes a list of services. Choose a service there to see the *service summary*.
  - This *summary table* includes a list of the actions and associated permissions for the chosen service. You can choose an action from that table to view the action *summary*.
  - To assign permissions to federated users, you can create an entity referred to as a **role** and define permissions for the **role**.
  - **Identity-Based Policies**
    - Permissions policies that you attach to a principal or identity.
    - **Managed policies** are standalone policies that you can attach to multiple users, groups, and roles in your AWS account.
    - **Inline policies** are policies that you create and manage and that are embedded directly into a single user, group, or role.

    Resource-based Policies
    - Permissions policies that you attach to a resource such as an Amazon S3 bucket.
    - Resource-based policies are only inline policies.
    - **Trust policies** - resource-based policies that are attached to a role and define which principals can assume the role.
- **AWS Security Token Service (STS)**

    - Create and provide trusted users with temporary security credentials that can control access to your AWS resources.
    - Temporary security credentials are short-term and are not stored with the user but are generated dynamically and provided to the user when requested.
    - By default, AWS STS is a global service with a single endpoint at https://sts.amazonaws.com.
- **Assume Role Options**
    - AssumeRole - Returns a set of temporary security credentials that you can use to access AWS resources that you might not normally have access to. These temporary credentials consist of an access key ID, a secret access key, and a security token. Typically, you use *AssumeRole* within your account or for cross-account access.
        - You can include multi-factor authentication (MFA) information when you call *AssumeRole*. This is useful for cross-account scenarios to ensure that the user that assumes the role has been authenticated with an AWS MFA device.
    - AssumeRoleWithSAML - Returns a set of temporary security credentials for users who have been authenticated via a SAML authentication response. This allows you to link your enterprise identity store or directory to role-based AWS access without user-specific credentials or configuration.
    - AssumeRoleWithWebIdentity - Returns a set of temporary security credentials for users who have been authenticated in a mobile or web application with a web identity provider. Example providers include Amazon Cognito, Login with Amazon, Facebook, Google, or any OpenID Connect-compatible identity provider.
- **STS Get Tokens**
    - GetFederationToken - Returns a set of temporary security credentials (consisting of an access key ID, a secret access key, and a security token) for a federated user. You must call the GetFederationToken operation using the long-term security credentials of an IAM user. A typical use is in a proxy application that gets temporary security credentials on behalf of distributed applications inside a corporate network.
    - GetSessionToken - Returns a set of temporary credentials for an AWS account or IAM user. The credentials consist of an access key ID, a secret access key, and a security token. You must call the GetSessionToken operation using the long-term security credentials of an IAM user. Typically, you use GetSessionToken if you want to use MFA to protect programmatic calls to specific AWS API operations.
- **IAM Access Analyzer**
    - Provides policy checks that help you proactively validate policies when creating them. These checks analyze your policy and report errors, warnings, and suggestions with actionable recommendations that help you set secure and functional permissions.
    - IAM Access Analyzer continuously monitors for new or updated resource policies and permissions granted for S3 buckets, KMS keys, SQS queues, IAM roles, Lambda functions, and Secrets Manager secrets.
- **Best Practices**
    - Lock Away Your AWS Account Root User Access Keys

- Create Individual IAM Users
- Use Groups to Assign Permissions to IAM Users
- Use AWS Defined Policies to Assign Permissions Whenever Possible
- Grant Least Privilege
- Use Access Levels to Review IAM Permissions
- Configure a Strong Password Policy for Your Users
- Enable MFA for Privileged Users
- Use Roles for Applications That Run on Amazon EC2 Instances
- Use Roles to Delegate Permissions
- Do Not Share Access Keys
- Rotate Credentials Regularly
- Remove Unnecessary Credentials
- Use Policy Conditions for Extra Security
- Monitor Activity in Your AWS Account

References:
https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html
https://aws.amazon.com/iam/faqs/

AWS Organizations

- It offers policy-based management for multiple AWS accounts.

Features

- With Organizations, you can create groups of accounts and then apply policies to those groups.
- Organizations provide you with a policy framework for multiple AWS accounts. You can apply policies to a group of accounts or all the accounts in your organization.
- AWS Organizations enables you to set up a single payment method for all the AWS accounts in your organization through **consolidated billing**. With consolidated billing, you can see a combined view of charges incurred by all your accounts, as well as take advantage of pricing benefits from aggregated usage, such as volume discounts for EC2 and S3.
- AWS Organizations, like many other AWS services, is **eventually consistent**. It achieves high availability by replicating data across multiple servers in AWS data centers within its region.

Administrative Actions in Organizations

- Create an AWS account and add it to your organization, or add an existing AWS account to your organization.
- Organize your AWS accounts into groups called *organizational units* (OUs).
- Organize your OUs into a hierarchy that reflects your company's structure.
- Centrally manage and attach policies to the entire organization, OUs, or individual AWS accounts.

Concepts

- An **organization** is a collection of AWS accounts that you can organize into a hierarchy and manage centrally.
- A **management account** is the AWS account you use to create your organization. You cannot change which account in your organization is the management account.
    - From the management account, you can create other accounts in your organization, invite and manage invitations for other accounts to join your organization, and remove accounts from your organization.
    - You can also attach policies to entities such as administrative roots, organizational units (OUs), or accounts within your organization.
    - The management account has the role of a payer account and is responsible for paying all charges accrued by the accounts in its organization.
- A **member account** is an AWS account, other than the management account, that is part of an organization. A member account can belong to only one organization at a time. The management account has the responsibilities of a payer account and is responsible for paying all charges that are accrued by the member accounts.

- An **administrative root** is the starting point for organizing your AWS accounts. The administrative root is the top-most container in your organization's hierarchy. Under this root, you can create OUs to logically group your accounts and organize these OUs into a hierarchy that best matches your business needs.
- An **organizational unit** (OU) is a group of AWS accounts within an organization. An OU can also contain other OUs enabling you to create a hierarchy.
- A **policy** is a "document" with one or more statements that define the controls that you want to apply to a group of AWS accounts.
    - **Service control policy** (SCP) is a policy that specifies the services and actions that users and roles can use in the accounts that the SCP affects. SCPs are similar to IAM permission policies except that they don't grant any permissions. Instead, SCPs are *filters* that allow only the specified services and actions to be used in affected accounts
- AWS Organizations has two available feature sets:
    - All organizations support **consolidated billing**, which provides basic management tools that you can use to centrally manage the accounts in your organization.
    - If you enable **all features**, you continue to get all the consolidated billing features plus a set of advanced features such as service control policies.
- You can remove an AWS account from an organization and make it into a standalone account.
- Organization Hierarchy
    - Including root and AWS accounts created in the lowest OUs, your hierarchy can be five levels deep.
    - Policies inherited through hierarchical connections in an organization.
    - Policies can be assigned at different points in the hierarchy.

Pricing

- This service is free.

References:
https://docs.aws.amazon.com/organizations/latest/userguide/
https://aws.amazon.com/organizations/features/
https://aws.amazon.com/organizations/faqs/

AWS Resource Access Manager

- A service that enables you to easily and securely share AWS resources with any AWS account or, if you are part of AWS Organizations, with Organizational Units (OUs) or your entire Organization. If you share resources with accounts that are outside of your Organization, then those accounts will receive an invitation to the Resource Share and can start using the shared resources upon accepting the invitation.
    - Only the master account can enable sharing with AWS Organizations.
    - The organization must be enabled for all features.
- RAM eliminates the need to create duplicate resources in multiple accounts. You can create resources centrally in a multi-account environment, and use RAM to share those resources across accounts in three simple steps:
    1. Create a Resource Share
    2. Specify resources
    3. Specify accounts
- You can stop sharing a resource by deleting the share in AWS RAM.
- Services you can share with AWS RAM

Service	Resource
Amazon Aurora	DB Clusters
AWS CodeBuild	Projects, Report Groups
Amazon EC2	Capacity Reservations, Dedicated Hosts, Subnets, Traffic mirror targets, Transit gateways
Amazon EC2 Image Builder	Components, Images (AMI), Image recipes
AWS License Manager	License configurations
AWS Resource Groups	Resource groups
Amazon Route 53	Forwarding rules

- Security
    - Use IAM policies to secure who can access resources that you shared or received from another account.
- Pricing
    - There is no additional charge for using AWS RAM.

References:
https://aws.amazon.com/ram/
https://aws.amazon.com/ram/faqs/

AWS Secrets Manager

- A secret management service that enables you to easily rotate, manage, and retrieve database credentials, API keys, and other secrets throughout their lifecycle.
- Features
  - AWS Secrets Manager encrypts secrets at rest using encryption keys that you own and store in AWS Key Management Service [customer managed keys]. When you retrieve a secret, Secrets Manager decrypts the secret and transmits it securely over TLS to your local environment.
  - You can rotate secrets on a schedule or on demand by using the Secrets Manager console, AWS SDK, or AWS CLI.
  - Secrets Manager natively supports rotating credentials for databases hosted on Amazon RDS and Amazon DocumentDB and clusters hosted on Amazon Redshift.
  - You can extend Secrets Manager to rotate other secrets, such as credentials for Oracle databases hosted on EC2 or OAuth refresh tokens, by using custom AWS Lambda functions.
- A secret consists of a set of credentials (user name and password), and the connection details used to access a secured service.
- A secret also contains **metadata** which include:
  - Basic information includes the name of the secret, a description, and the Amazon Resource Name (ARN) to serve as a unique identifier.
  - The ARN of the AWS KMS key Secrets Manager uses to encrypt and decrypt the protected text in the secret. If you don't provide this information, Secrets Manager uses the default AWS KMS key for the account.
  - Information about how frequently to rotate the key and what Lambda function to use to perform the rotation.
  - A user-provided set of tags. You can attach tags as key-value pairs to AWS resources for organizing, logical grouping, and cost allocation.
- A secret can contain **versions**:
  - Although you typically only have one version of the secret active at a time, multiple versions can exist while you rotate a secret on the database or service. Whenever you change the secret, Secrets Manager creates a new version.
  - Each version holds a copy of the encrypted secret value.
  - Each version can have one or more *staging labels* attached identifying the stage of the secret rotation cycle.
- Supported Secrets
  - Database credentials, on-premises resource credentials, SaaS application credentials, third-party API keys, and SSH keys.
  - You can also store JSON documents.
- To retrieve secrets, you simply replace secrets in plain text in your applications with code to pull in those secrets programmatically using the Secrets Manager APIs.
- Secrets can be cached on the client side, and updated only during a secret rotation.

- During the secret rotation process, Secrets Manager tracks the older credentials, as well as the new credentials you want to start using, until the rotation completes. It tracks these different versions by using *staging labels*.
- How Secret Rotation Works
    - The rotation function contacts the secured service authentication system and creates a new set of credentials to access the database. Secrets Manager stores these new credentials as the secret text in a new version of the secret with the `AWSPENDING` staging label attached.
    - The rotation function then tests the `AWSPENDING` version of the secret to ensure that the credentials work, and grants the required level of access to the secured service.
    - If the tests succeed, the rotation function then moves the label `AWSCURRENT` to the new version to mark it as the default version. Then, all of the clients start using this version of the secret instead of the old version. The function also assigns the label AWSPREVIOUS to the old version. The version that had `AWSPREVIOUS` staging label now has no label, and is therefore deprecated.

  Network Setup for Secret Rotation
    - When rotating secrets on natively supported services, Secrets Manager uses CloudFormation to build the rotation function and configure the network connection between the two.
        - If your protected database service **runs in a VPC and is not publicly accessible**, then the CloudFormation template configures the **Lambda rotation function to run in the same VPC**. The rotation function can communicate with the protected service **directly within the VPC**.
        - If you run your protected service as a **publicly accessible resource**, in a VPC or not, then the CloudFormation template configures the **Lambda rotation function not to run in a VPC**. The Lambda rotation function communicates with the protected service **through the publicly accessible connection point**.
    - By default, the Secrets Manager endpoints run on the public Internet. If you run your Lambda rotation function and protected database or service in a VPC, then you must perform one of the following steps:
        - **Add a NAT gateway to your VPC**. This enables traffic that originates in your VPC to reach the public Secrets Manager endpoint.
        - **Configure Secrets Manager service endpoints directly within your VPC**. This configures your VPC to intercept any request addressed to the public regional endpoint, and redirect the request to the private service endpoint running within your VPC.

  You can create two secrets that have different permissions
    - User Secret - can be used to connect to linked services, but it cannot be rotated. The user will have to wait for the master secret to be rotated and propagated for it to change.
    - Master Secret - has sufficient permissions to rotate secrets of linked services. This scenario is typically used when you have users that are actively using the old secret, and you do not want to break operations after you rotate the secret. You can have your users update their clients first before using the newly rotated credentials

- Secrets Manager lets you easily copy your secrets to multiple AWS Regions, which includes the primary secret and the associated metadata such as tags, resource policies, and secret updates such as rotation.
- Security
    - By default, Secrets Manager does not write or cache the secret to persistent storage.
    - By default, Secrets Manager only accepts requests from hosts that use the open standard Transport Layer Security (TLS) and Perfect Forward Secrecy.
    - You can control access to the secret using AWS Identity and Access Management (IAM) policies.
    - You can tag secrets individually and apply tag-based access controls.
    - You can configure VPC endpoints to keep traffic between your VPC and Secrets Manager within the AWS network.
    - Secrets Manager does not immediately delete secrets. Instead, Secrets Manager immediately makes the secrets inaccessible and scheduled for deletion after a recovery window of a **minimum of seven days**. Until the recovery window ends, you can recover a secret you previously deleted.
    - By using the CLI, you can delete a secret without a recovery window.

  Compliance
    - Secrets Manager is HIPAA, PCI DSS and ISO, SOC, FedRAMP, DoD SRG, IRAP, and OSPAR compliant.

  Pricing
    - You pay based on the number of secrets stored and API calls made per month.

References:
https://aws.amazon.com/secrets-manager/
https://aws.amazon.com/secrets-manager/faqs/
https://docs.aws.amazon.com/secretsmanager/latest/userguide/intro.html

AWS Security Hub

- AWS Security Hub provides a **comprehensive view** of your **security state** within AWS and your **compliance** with security industry standards and best practices.
- Features
    - You now have a single place that **aggregates, organizes, and prioritizes your security alerts**, or findings, across multiple accounts, AWS partner tools, and AWS services such as Amazon GuardDuty, Amazon Inspector, Amazon Macie, AWS IAM Access Analyzer, AWS Firewall Manager, and AWS Audit Manager.
    - AWS Security Hub works with AWS Organizations to simplify security posture management across all of your existing and future AWS accounts in an organization.
    - You can run automated, continuous account-level configuration and compliance checks based on industry standards and best practices, such as the Center for Internet Security (CIS) AWS Foundations Benchmark. These checks provide a compliance score and identify specific accounts and resources that require attention.
    - AWS Security Hub compliance checks also leverage configuration items recorded by AWS Config.
    - Integrated dashboards consolidate your security findings across accounts to show you their current security and compliance status.
    - You can send security findings to ticketing, chat, email, or automated remediation systems through integration with Amazon EventBridge (Amazon CloudWatch Events).
    - All findings are stored for at least 90 days within AWS Security Hub.
- Security Hub receives and processes only those findings from the same Region where you enabled Security Hub in your account.
- Concepts
    - AWS Security Finding Format - A standardized format for the contents of findings that Security Hub aggregates or generates.
    - Control - A safeguard or countermeasure prescribed for an information system or an organization designed to protect the confidentiality, integrity, and availability of its information and to meet a set of defined security requirements. A security standard consists of controls.
    - Custom action - A Security Hub mechanism for sending selected findings to Amazon EventBridge (Amazon CloudWatch Events).
    - Finding - The observable record of a compliance check or security-related detection.
    - Insight - A collection of related findings defined by an aggregation statement and optional filters. An insight identifies a security area that requires attention and intervention.
    - Compliance standards - Sets of controls that are based on regulatory requirements or best practices.
    - You can disable specific compliance controls that are not relevant to your workloads.
- Compliance standard vs. Control vs. Compliance check

    - A compliance standard is a collection of controls based on regulatory frameworks or industry best practices. Security Hub conducts automated compliance checks against controls. Each compliance check consists of an evaluation of a rule against a single resource. A single control may involve multiple resources and a compliance check is performed against each resource.
    - AWS Security Hub uses a **service-linked role** that includes the permissions and trust policy that Security Hub requires to detect and aggregate findings, and to configure the requisite AWS Config infrastructure needed to run compliance checks. In order for Security Hub to run **compliance checks** in an account, you must have **AWS Config enabled** in that account.
- Pricing
    - AWS Security Hub is priced based on the **quantity of compliance checks** and the **quantity of finding ingestion events**.
    - Pricing is on a monthly per account, per region basis.

References:
https://aws.amazon.com/security-hub/
https://aws.amazon.com/security-hub/faqs/

AWS Shield

- A managed Distributed Denial of Service (DDoS) protection service that safeguards applications running on AWS.

Shield Tiers and Features

- **Standard**
  - All AWS customers benefit from the automatic protections of Shield Standard.
  - Shield Standard provides always-on network flow monitoring, which inspects incoming traffic to AWS and detects malicious traffic in real-time.
  - Uses several techniques like deterministic packet filtering and priority based traffic shaping to automatically mitigate attacks without impact to your applications.
  - When you use Shield Standard with CloudFront and Route 53, you receive comprehensive availability protection against all known infrastructure attacks.
  - You can also view all the events detected and mitigated by AWS Shield in your account.
- **Advanced**
  - Shield Advanced provides enhanced detection, inspecting network flows and also monitoring application layer traffic to your Elastic IP address, Elastic Load Balancing, CloudFront, or Route 53 resources.
  - It handles the majority of DDoS protection and mitigation responsibilities for **layer 3**, **layer 4**, and **layer 7** attacks.
  - You have 24x7 access to the AWS DDoS Response Team. To contact the DDoS Response Team, customers will need the Enterprise or Business Support levels of AWS Premium Support.
  - It automatically provides additional mitigation capacity to protect against larger DDoS attacks. The DDoS Response Team also applies manual mitigations for more complex and sophisticated DDoS attacks.
  - It gives you complete visibility into DDoS attacks with near real-time notification via CloudWatch and detailed diagnostics on the "AWS WAF and AWS Shield" Management Console.
  - Shield Advanced comes with "DDoS cost protection", a safeguard from scaling charges as a result of a DDoS attack that causes usage spikes on your AWS services. It does so by providing service credits for charges due to usage spikes.
  - It is available globally on all CloudFront and Route 53 edge locations.
  - With Shield Advanced you will be able to see the history of all incidents in the trailing 13 months.

Pricing

- **Shield Standard** provides protection at no additional charge.

- **Shield Advanced**, however, is a paid service. It requires a 1-year subscription commitment and charges a monthly fee, plus a usage fee based on data transfer out from CloudFront, ELB, EC2, and AWS Global Accelerator.

References:
https://aws.amazon.com/shield/features/
https://aws.amazon.com/shield/pricing/
https://aws.amazon.com/shield/faqs/

AWS WAF

- A web application firewall that helps protect web applications from attacks by allowing you to configure rules that **allow, block, or monitor (count) web requests** based on conditions that you define.
- These conditions include:
    - IP addresses
    - HTTP headers
    - HTTP body
    - URI strings
    - SQL injection
    - cross-site scripting.

Features

- WAF lets you create rules to filter web traffic based on conditions that include IP addresses, HTTP headers, and body, or custom URIs.
- You can also create rules that block common web exploits like SQL injection and cross site scripting.
- For application layer attacks, you can use WAF to respond to incidents. You can set up proactive rules like *Rate Based Blacklisting* to automatically block bad traffic or respond immediately to incidents as they happen.
- WAF provides real-time metrics and captures raw requests that include details about IP addresses, geo locations, URIs, User-Agent and Referers.
- **AWS WAF Security Automations** is a solution that automatically deploys a single web access control list (web ACL) with a set of AWS WAF rules designed to filter common web-based attacks. The solution supports log analysis using Amazon Athena and AWS WAF full logs.

Conditions, Rules, and Web ACLs

- You define your conditions, combine your conditions into rules, and combine the rules into a web ACL.
- **Conditions** define the basic characteristics that you want WAF to watch for in web requests.
- You combine conditions into **rules** to precisely target the requests that you want to allow, block, or count. WAF provides two types of rules:
    - **Regular rules** - use only conditions to target specific requests.
    - **Rate-based rules** - are similar to regular rules, with a rate limit. Rate-based rules count the requests that arrive from a specified IP address every five minutes. The rule can trigger an action if the number of requests exceed the rate limit.
- **WAF Managed Rules** are an easy way to deploy pre-configured rules to protect your applications common threats like application vulnerabilities. All Managed Rules are automatically updated by AWS Marketplace security Sellers.

- After you combine your conditions into rules, you combine the rules into a **web ACL**. This is where you define an action for each rule—allow, block, or count—and a default action, which determines whether to allow or block a request that doesn't match all the conditions in any of the rules in the web ACL.

Pricing

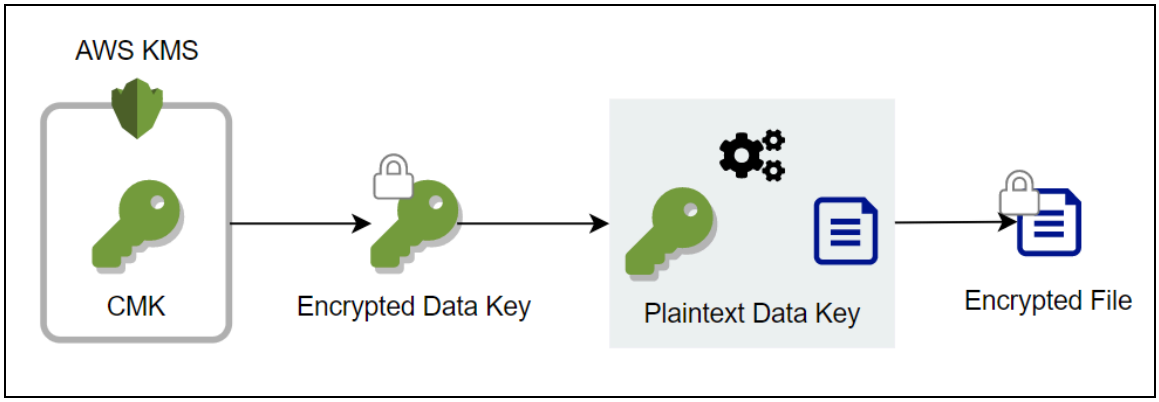
- WAF charges based on the number of web access control lists (web ACLs) that you create, the number of rules that you add per web ACL, and the number of web requests that you receive.

References:
https://aws.amazon.com/waf/features/
https://aws.amazon.com/waf/pricing/
https://aws.amazon.com/waf/faqs/

Comparison of AWS Services

AWS Key Management Service (KMS) vs. AWS CloudHSM

AWS Key Management Service (AWS KMS) lets you create, store, and manage customer master keys (KMS keys) securely. An *AWS KMS key* is a logical representation of a master key. KMS uses KMS keys to generate *data keys* that are used to encrypt data of any size.



With KMS, you can use *AWS Managed Key, Customer Managed Key, and AWS Owned Key*. There is also a key rotation policy available for KMS keys. AWS KMS integrates with several AWS Services like S3, RDS, EBS, and more.

AWS CloudHSM

A *hardware security module (HSM)* is a specialized security device that manages cryptographic keys, and performs encryption and decryption functions for digital signatures, authentication, and other cryptographic functions. AWS brought this solution to the cloud as *AWS CloudHSM*.

AWS CloudHSM is a cloud-based hardware security module (HSM) that enables you to quickly generate and use your encryption keys on the AWS Cloud. Since this is a managed hardware, you don't have to worry about the administrative tasks of maintaining an HSM. With AWS CloudHSM, hardware provisioning, software patching, high availability, and backups are all automated. When you use the AWS CloudHSM service, you create a *CloudHSM Cluster*. These clusters reside inside a VPC and are spread across multiple Availability Zones in a region.

CloudHSM is used to accomplish the following:

- Offload the SSL/TLS Processing for Web Servers
- Protect the Private Keys for an Issuing Certificate Authority (CA)

- Enable Transparent Data Encryption (TDE) for Oracle Databases

KMS and CloudHSM Integration

You can create a Custom Key Store on Cloud HSM, which you can use to store your KMS key on KMS instead of using the standard KMS key store.

When to use AWS KMS?

AWS KMS allows you to have centralized management of your KMS key. You can assign Key Administrators, users, and roles to your KMS key when creating it. You can also track your KMS logs through Amazon EventBridge and AWS CloudTrail.

KMS key is primarily used to generate and encrypt/decrypt your data keys, but it can also encrypt/decrypt small data (up to 4096 bytes). AWS KMS does not store or manage data keys, and you cannot use KMS to encrypt or decrypt with data keys. To do this, you need to use the AWS Encryption SDK.

AWS KMS KMS keys are backed by FIPS-validated hardware service modules (HSMs) that KMS manages.

When to use AWS CloudHSM?

AWS CloudHSM provides you with a FIPS 140-2 Level 3 overall validated single-tenant HSM cluster in your Amazon Virtual Private Cloud (VPC).

It is ideal to use CloudHSM if you want full control of your HSMs that generate and store your encryption keys. This includes creating HSM users and policies. The encryption keys that you generate and use with CloudHSM are accessible only by the HSM users that you specify. You have exclusive control over how your keys are used via an authentication mechanism independent from AWS. You also create the symmetric keys and asymmetric key pairs that the HSM stores.

In case you need to manage and store your AWS KMS keys but do not need to manage the HSM, you can use AWS Key Management Service instead.

References:
https://docs.aws.amazon.com/crypto/latest/userguide/awscryp-choose-kms.html
https://docs.aws.amazon.com/crypto/latest/userguide/awscryp-choose-hsm.html
https://docs.aws.amazon.com/cloudhsm/latest/userguide/use-cases.html

Application Load Balancer vs Network Load Balancer vs Gateway Load Balancer

FEATURE	Application Load Balancer	Network Load Balancer	Gateway Load Balancer
Protocols	HTTP, HTTPS, gRPC	TCP, UDP, TLS	IP
Platforms	VPC	VPC	VPC
Health checks	HTTP, HTTPS, gRPC	TCP, HTTP, HTTPS	TCP, HTTP, HTTPS
Cloudwatch Metrics	✔	✔	✔
Logging	✔	✔	✔
Zonal Failover	✔	✔	✔
Connection Draining (deregistration of delay)	✔	✔	✔
Load Balancing to multiple ports on the same instance	✔	✔	✔
IP addresses as targets	✔	✔ (TCP, TLS)	✔
Load balancer detection protection	✔	✔	✔
Configuration idle connection timeout	✔		
Cross-zone load balancing	✔	✔	✔
Sticky sessions	✔	✔	✔
Static IP		✔	
Elastic IP address		✔	
Preserve Source IP address	✔	✔	✔
Resource-based IAM permissions	✔	✔	✔
Tag-based IAM permissions	✔	✔	✔

FEATURE	Application Load Balancer	Network Load Balancer	Gateway Load Balancer
Slow start	✔		
Web sockets	✔	✔	✔
Private Link Support		✔ (TCP, TLS)	✔(GWLBE)
Source IP address CIDR-based routing	✔		
LAYER 7			
Path-based routing	✔		
Host-based routing	✔		
Native HTTP/2	✔		
Redirects	✔		
Fixed response	✔		
Lambda functions as targets	✔		
HTTP header-based routing	✔		
HTTP method-based routing	✔		
Query string parameter-based routing	✔		
SECURITY			
SSL offloading	✔	✔	

FEATURE	Application Load Balancer	Network Load Balancer	Gateway Load Balancer
Server Name Indication (SNI)	✔	✔	
Back-end server encryption	✔	✔	
User authentication	✔		
Session Resumption	✔	✔	
Terminates Flow/proxy behavior	✔	✔	✔

Common features between the load balancers:

- Has instance health check features
- Has built-in CloudWatch monitoring
- Logging features
- Support zonal failover
- Support cross-zone load balancing (evenly distributes traffic across registered instances in enabled AZs)
- Resource-based IAM permission policies
- Tag-based IAM permissions
- Flow stickiness - all packets are sent to one target and return the traffic that comes from the same target.

Symmetric vs. Asymmetric Encyrption KMS keys

Even before the Internet, the security, privacy, and integrity of information have always been the top concern of institutions like banks, hospitals, and universities. Nobody wants their personal information (name, address, credit card number, etc.) to be exposed in public for anyone to use. Imagine signing up on your favorite social media website, and after a few days, somewhere on the globe has been using your profile and pretending to be you without you knowing! Or maybe you've been using your credit card for shopping online and suddenly, your bank is sending you email reports for fraudulent activities on your account. That would be a creepy and scary world to live in.

The unfortunate truth is that no matter how secure you might think your system is, it will never be one hundred percent secure. There will always be loopholes, and as computers get even more powerful, common attacks like brute force will still be a valid threat. For this reason, tremendous efforts have been made to improve and mitigate scenarios where sensitive data are compromised. Encryption proves to be the most effective solution in battling data breaches.

What is Encryption?

Encryption is the process of converting the information (**plaintext**) into secret code (**ciphertext**) to hide its original meaning. It is used to protect the data so that only authorized users can read it.

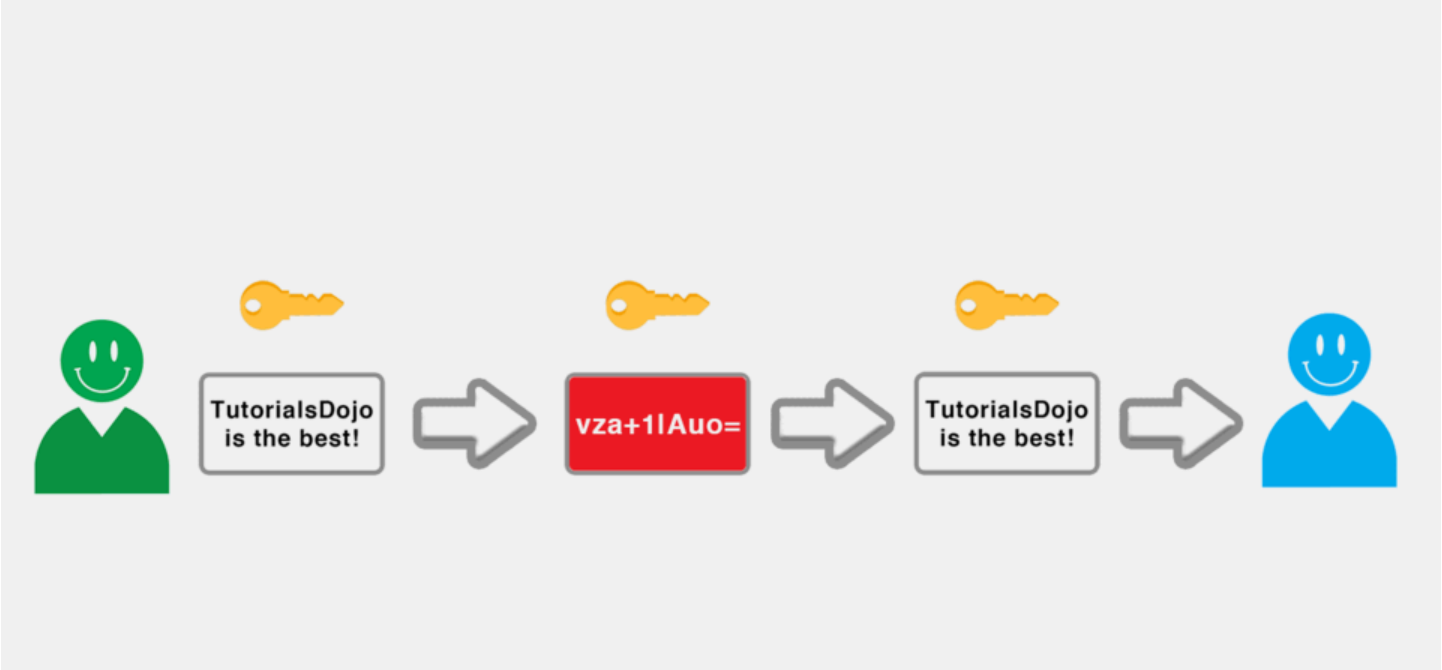
It uses the concept of "keys" which are used to encrypt and decrypt the sensitive information from one end to another. The idea is, without these keys, one cannot simply decrypt and read the hidden information.

There are two types of encryption:

Symmetric encryption - uses a single key for both encryption and decryption. The shared key must be sent together with the encrypted data in order for other parties to read it.

Because of the simplicity of the process, it is usually faster than asymmetric encryption and is efficient in encrypting large amounts of data.

Symmetric Encryption Disadvantage:



The main disadvantage of using a symmetric key is the difficulty of transporting the shared key. It is difficult in the sense that attacks, like man-in-the-middle attacks, could easily obtain both the key and the encrypted data. Since a single key is used for both encryption and decryption, the man behind the attack would be able to decrypt the information sent over the network.

Asymmetric encryption - it uses a mathematically related public and private key for encryption and decryption. The public key is used for encrypting data and can never be used for decryption. The private key is only used for decrypting data. The private key stays on the user while both the public key and the encrypted data is sent to other parties. This kind of method makes the sharing of public keys a lot easier because even if someone has managed to steal the data with the public key, he won't be able to decrypt the information.

Since this type of encryption uses a more complex algorithm than symmetric encryption, asymmetric is used for systems that use small data. It is usually used for establishing secure connections like TLS and SSH. It is also slower than symmetric encryption and is inefficient for encrypting large data.

Symmetric and Asymmetric Keys In AWS Key Management System:

AWS KMS Keys

AWS KMS key is the term used on AWS that refers to the "root key". This is the primary resource that is managed by AWS KMS. You control the lifecycle of the KMS key as well as who can use or manage it.

Three types of KMS keys:

- **Customer Managed key**
  - You can view the KMS key's metadata
  - You can manage the KMS key
  - It is used only for your account
  - Automatic rotation is optional
- **AWS managed key**
  - You can view the KMS key's metadata, but you cannot manage it.
  - It is used only for your account
  - Automatic rotation is required
- **AWS owned key**
  - These are KMS keys that an AWS Service owns and manages for use in multiple AWS accounts.
  - You do not need to create or manage the AWS owned keys.
  - The key rotation strategy for an AWS owned key is determined by the AWS service that creates and manages the KMS key.

KMS key supports both **symmetric** and **asymmetric encryption.** Although integrated in AWS Cloud, the concepts behind the encryption are still the same as the one explained above.

Symmetric KMS key

- Represents a 256-bit encryption key that **never leaves AWS KMS unencrypted.**
- **A Symmetric KMS key type is created by default** when you call the create-key API without specifying value for `--customer-master-key-spec`.
  - The `--customer-master-key-spec` parameter lets you define the KMS key specification. You can either choose symmetric or asymmetric.

- AWS services that are integrated with AWS KMS (Amazon DynamoDB, Amazon S3, Amazon Relational Database Service, etc.) use symmetric KMS key to encrypt and decrypt data and **do not support asymmetric KMS key.**
- You can **import your own key material into a symmetric KMS key** and create symmetric KMS keys in custom key stores.
    - Note that imported key material is supported **only for symmetric KMS keys**.

Asymmetric KMS key

- **Private Key**
    - The private key is created in AWS KMS and never leaves AWS KMS unencrypted.
    - The private **can only be used by calling AWS KMS**.
- **Public Key**
    - The public key can be used **within or outside** of AWS KMS.

Two types of asymmetric KMS key:

- **RSA KMS keys**
    - Can be used for **encryption and decryption or signing and verification**. You can never use RSA KMS key for both purposes at the same time.
- **Elliptic Curve (ECC) KMS keys**
    - Elliptic curve key pair used for **signing and verification**

Use Case

- **Symmetric**
    - Use symmetric if you are encrypting data within the AWS service. Since AWS services integrated with AWS KMS only support Symmetric KMS key, there is no sense to use asymmetric KMS key.
    - Symmetric encryption is commonly used when encrypting data at rest. AWS uses symmetric encryption when you're encrypting objects stored in an S3 bucket or enabling encryption for your EBS volumes.
- **Asymmetric**

- Since you can use the public key outside of AWS KMS in asymmetric, it is a good choice if you are building applications for users who cannot call AWS KMS. The easy process of creating key pairs is one of the main benefits of it.
- Applicable for data signing and verification. You can use asymmetric KMS key to authenticate documents by using a digital signature. Digital signing is used to ensure the integrity of data that passes between networks. Suppose that a contract form is sent to you from your client. And you must ensure that the information within the contract is all true and has not been altered by third-parties. If you have the right key, you can cryptographically verify that the contract is indeed sent from your client.

References:
https://docs.aws.amazon.com/kms/latest/developerguide/symmetric-asymmetric.html
https://docs.aws.amazon.com/kms/latest/developerguide/symm-asymm-compare.html

FINAL REMARKS AND TIPS

That's a wrap! Thank you once again for choosing our Study Guide and Cheat Sheets for the AWS Certified Security Specialty (SCS-C03) exam. The Tutorials Dojo team spent a considerable amount of time and effort to produce this content to help you pass the AWS exam.

We also recommend that before taking the actual SCS-C03 exam, allocate some time to check your readiness by taking our **AWS practice test course** in the Tutorials Dojo Portal. This will help you identify the topics that you need to improve on and help reinforce the concepts that you need to fully understand in order to pass this certification test. It also has different training modes that you can choose from such as Timed mode, Review mode, Section-Based tests, and Final test plus bonus flashcards. In addition, you can read the technical discussions in our forums or post your queries if you have one. If you have any issues, concerns, or constructive feedback on our eBook, feel free to contact us at support@tutorialsdojo.com.

On behalf of the Tutorials Dojo team, we wish you all the best on your upcoming AWS Certified Security Specialty exam. May it help advance your career, as well as increase your earning potential.

With the right strategy, hard work, and unrelenting persistence, you can definitely make your dreams a reality! You can make it!

Sincerely,
Jon Bonso, Nestor Mayagma Jr., and the Tutorials Dojo Team

ABOUT THE AUTHORS

Jon Bonso (10x AWS Certified)

Born and raised in the Philippines, Jon is the Co-Founder of **Tutorials Dojo**. Now based in Sydney, Australia, he has over a decade of diversified experience in Banking, Financial Services, and Telecommunications. He's 10x AWS Certified and has worked with various cloud services such as Google Cloud and Microsoft Azure. Jon is passionate about what he does and dedicates a lot of time creating educational courses. He has given IT seminars to different universities in the Philippines for free and has launched educational websites using his own money and without any external funding.

Nestor Mayagma Jr. (3x AWS Certified)

Nestor is a cloud engineer and a content creator at Tutorials Dojo. He's also been an active AWS Community Builder since 2022, with a growing interest in multi-cloud technologies across AWS, Azure, and Google Cloud.

Nikee Pearl Tomas (2x AWS Certified)

Nikee is a Team Lead and Web Developer at Tutorials Dojo , as well as a two-time AWS Certified professional. Since 2023, she has been a member of the AWS Community Builders program, where she actively engages with the cloud community and shares her expertise. With a strong background in front-end development and website management, she plays a vital role in optimizing the Tutorials Dojo platform. Nikee is passionate about cloud computing, continuous learning, and creating high-quality content that supports learners in achieving their certification goals.