SECURITY SPECIALTY RUNBOOK

INTERACTIVE SUPPLEMENT FOR THE AWS CERTIFIED SECURITY SPECIALIST COURSE

ADRIAN CANTRILL, TRAINING ARCHITECT

INTRODUCTION

INCIDENT RESPONSE

LOGGING & MONITORING

INFRASTRUCTURE SECURITY

IDENTITY & ACCESS MGNT

DATA PROTECTION

Linux Academy

Course Introduction

The AWS Certified Security Specialty is a certification based around securing applications in AWS. It is one of three specialty certifications offered by AWS. The certification focuses on five components or domains when designing and operating security in the cloud. These are:

- Identity and Access Management
- Detective Controls
- Infrastructure Protection
- Data Protection
- Incident Response

This course has been developed to provide you with the requisite knowledge to not only pass the AWS Certified Security Specialty certification exam but also gain the hands-on experience required to become a qualified AWS security specialist working in a real-world environment.

From the Author

Dear Students and Candidates,

It is my pleasure to bring you this course for the newly released AWS Certified Security Specialist! I want you to know I have had a lot of fun putting this course together. Preparing for this course and exam is exciting for me because the content is based on real-world scenarios. We are not just memorizing facts for an exam. The certification covers the main aspects of security in the cloud and should provide you plenty to think about regarding your own organizations and applications.

Security can be a topic that people sometimes overlook, as we have all seen in the news. It can be difficult and requires thinking, approaching events from multiple perspectives, and lots of training. However, it is the most important aspect of running any application, especially when there are compliance concerns and personal information being stored.

I am happy and excited you have decided to take this journey toward the Security Specialty certification with us! As always, feel free to post in the community on Linux Academy if you need any help or have any concerns, and we will be there to help. Also, there are rating features on all the content in this course. I would appreciate it if you would help me by providing feedback.

So if you are ready to go...Let's get started!

Adrian Cantrill

AWS Abuse Notifications

AWS Acceptable Use Policy: https://aws.amazon.com/aup/

Sample Abuse Notification

NOTE: AWS does NOT allow port scanning or pen testing of your resources without permission.

Compromised Resources and Abuse

Abuse activities: Externally observed behavior of AWS customer instances or resources that are malicious, offensive, illegal, or could harm other internet sites.

AWS will shut down malicious abusers, but many of the abuse complaints are about customers conducting legitimate business on AWS.

Example causes of abuse that are not intentional:

- Compromised Resource
- Secondary Abuse
- Application Function
- False Complaints

Abuse Case xxxxxxxxxx

Hello,

We have detected that your instance(s):

xxxxxxxx

have been behaving in the following way that is against our AWS Customer Agreement:

Port Scanning

Please be aware that in terms of the Web Services License Agreement http://aws.amazon.com/agreement/ if your instance(s) continue such abusive behavior, your account may be subject to termination.

EC2 has taken the following administrative action(s) against your instance(s):

THROTTLED OUTBOUND PORT 22.

….

Please confirm that all necessary steps to cease this activity have been taken on your side. Failure to take action to stop abuse may result in suspension of your instance or termination of your account.

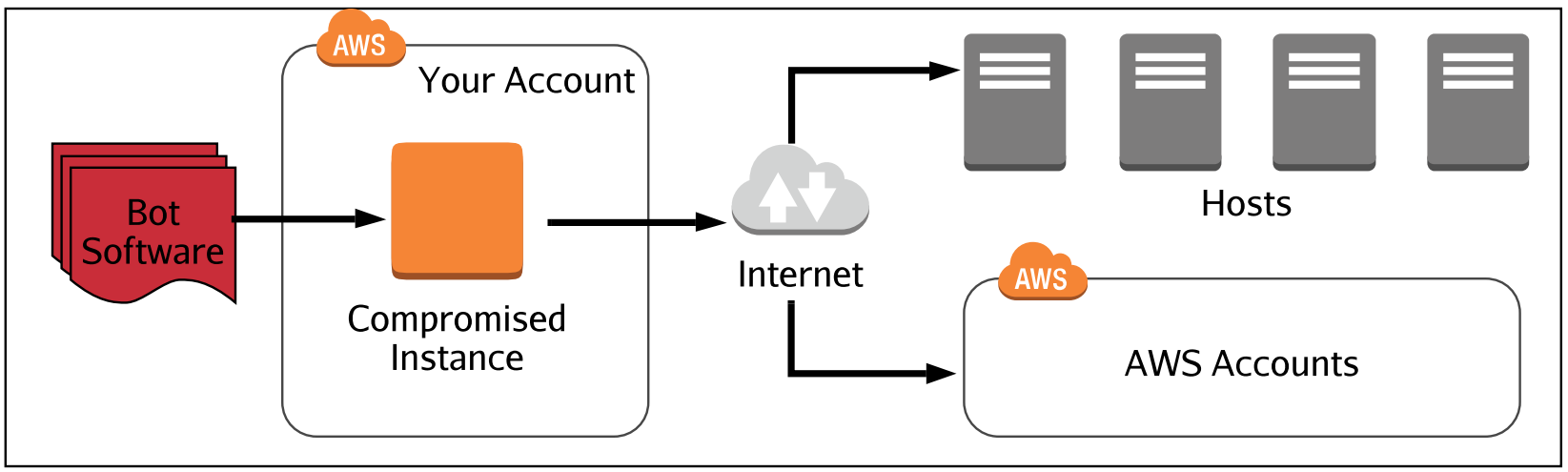
If you feel that our findings are in error, or you have taken necessary steps to address the problem, please contact ec2-abuse@amazon.com to request removing the administrative block to your instance(s). Please make sure your case number is included in your email subject.

SECURITY SPECIALTY RUNBOOK

Compromised Resource

Note: The flow shown is for CloudWatch Logs using filters to trigger Alarms and SNS notifications. The CloudWatch events flow mentioned in the course is not shown.

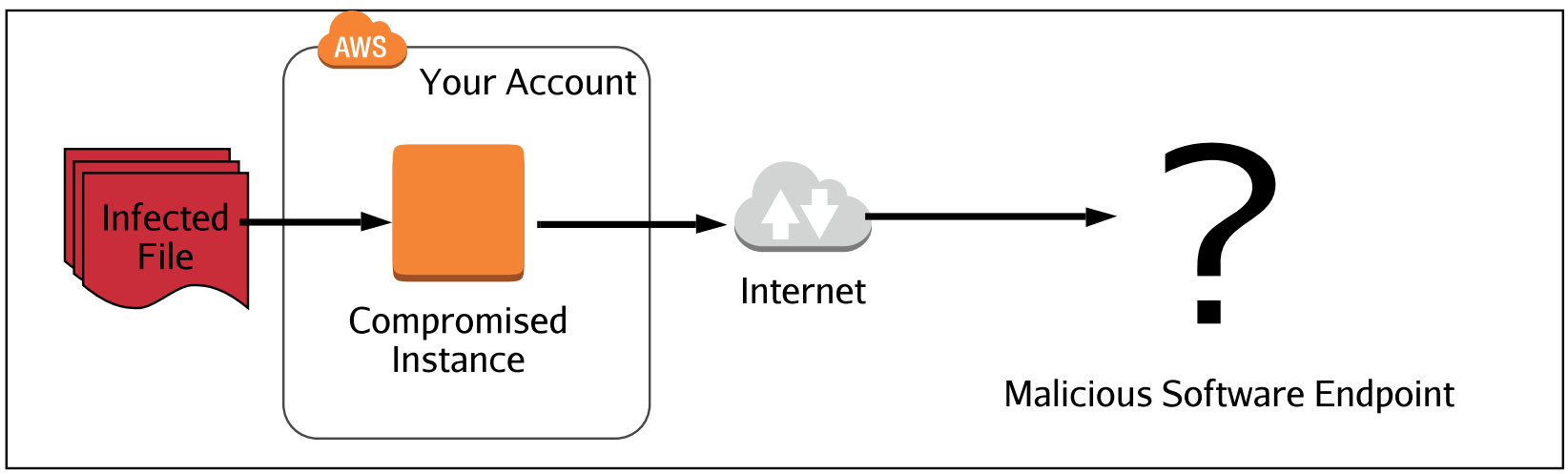
EC2 instance becoming part of a botnet then attacking other hosts on the internet. This traffic could be going to other AWS accounts as well.



Note: The flow shown is for C[illegible]Watch Logs using
Metric Filters to tri[illegible]d SNS notifications.
The Clou[illegible]ow mentioned in the course is

Secondary Abuse

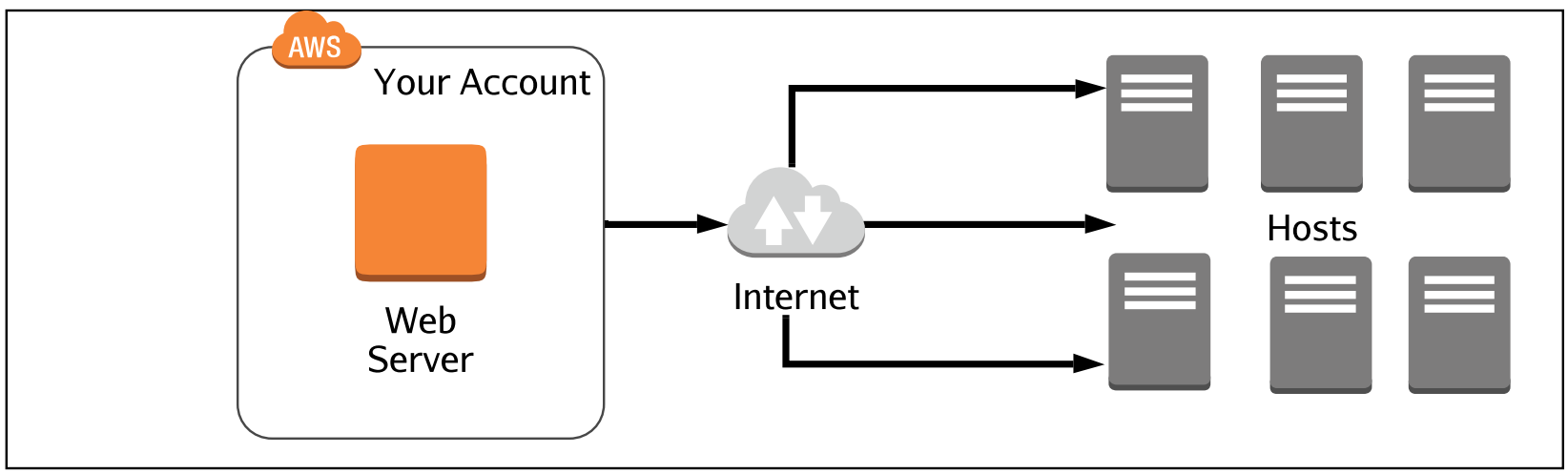
One of your end-users posts an infected file on your resources. When that file calls "home," it is going to appear to be traffic generated in your account.



Note: The flow shown is for CloudWatch Logs using Metric Filters to trigger Alarms and SNS notifications. The CloudWatch events flow mentioned in the course is not shown.

Application Function

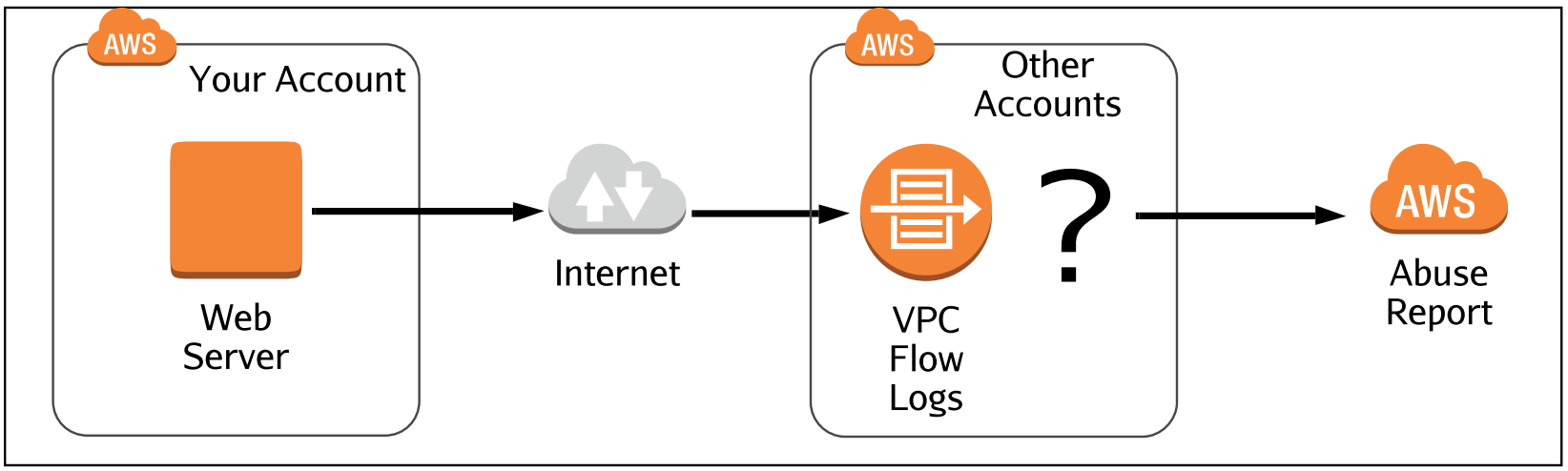
If you are using applications such as web crawlers, it can sometimes appear as a DoS attack and AWS will react accordingly.



Note: The flow shown is for CloudWatch Logs using Metric Filters to trigger Alarms and SNS notifications. The CloudWatch events flow mentioned in the course is not shown.

False Complaints

Other AWS users can report your activity to AWS. The complaint might appear legitimate, and AWS will react accordingly.



Given an AWS abuse notice, evaluate the suspected compromised instance or exposed access keys.

Note: The flow shown is for CloudWatch Logs using Metric Filters to trigger Alarms and SNS notifications. The CloudWatch events flow mentioned in the course is not shown.

Responding to Abuse Notifications

There is a chance that the investigation of abuse will turn out to be a compromised account or resource. If this is the case, the following AWS recommendations can help:

- **Change** the root password and the passwords for all IAM users
- **Add** MFA to all Admin users and anyone who accesses the AWS console
- **Create** a new EC2 key pair and update instances (deleting the compromised key):
  - Create an AMI and relaunch
  - Edit the `.ssh/authorized_keys` file
- **Delete** or **rotate** potentially compromised IAM access keys
- **Delete** unrecognized or unauthorized resources:
  - Instances
  - IAM users
  - Spot bids
- **Contact** AWS Support:
  - Respond to the notification
  - **Important:** Do not ignore AWS abuse communications and make sure they have the most effective email address on file

The CloudWatch events flow mentioned in the course is

Be Proactive: Avoid Being Compromised

- Vault root credentials and remove access keys if they exist
- Require a strong password and MFA on all IAM accounts
- Use roles whenever possible, do not trust humans
- Do NOT copy Ec2 key pairs to instances and protect them on admin machines
- Rotate IAM access keys regularly

There are people scanning repositories like GitHub for access keys, EC2 key pairs, and other sensitive information. AWS has created a tool to help prevent spillage:

- Git-secrets: Prevents committing secrets and sensitive information to git repositories
- https://github.com/awslabs/git-secrets

Verify that the Incident Response plan includes relevant AWS services.

Note: The flow shown is for CloudWatch Logs using Metric Filters to trigger Alarms and SNS notifications. The CloudWatch events flow mentioned in the course is not shown.

What is Incident Response?

Incident- An unplanned interruption or degradation of an IT service

NIST IR Standards (National Institute of Standards and Technology):

- Full Document: https://nvd.nist.gov/800-53/Rev4/family/INCIDENT%20RESPONSE

IR-1 Incident Response Policy and Procedures

IR-2 Incident Response Training

IR-3 Incident Response Testing

IR-4 Incident Handling

IR-5 Incident Monitoring

IR-6 Incident Reporting

IR-7 Incident Response Assistance

IR-8 Incident Response Plan

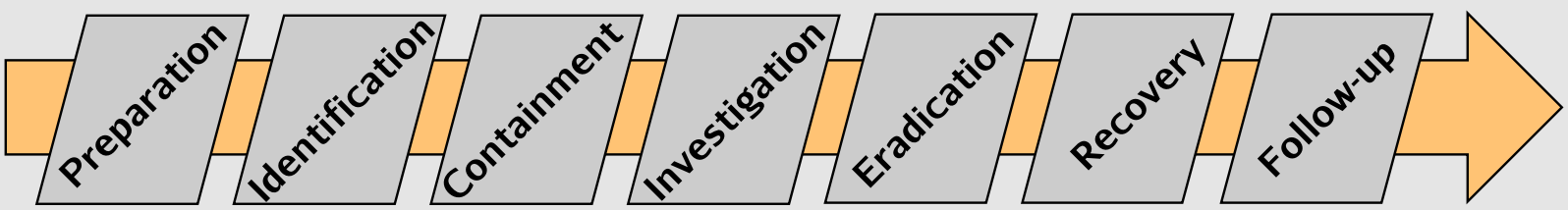
IR-9 Information Spillage Response

IR-10 Integrated Information Security Analysis Team

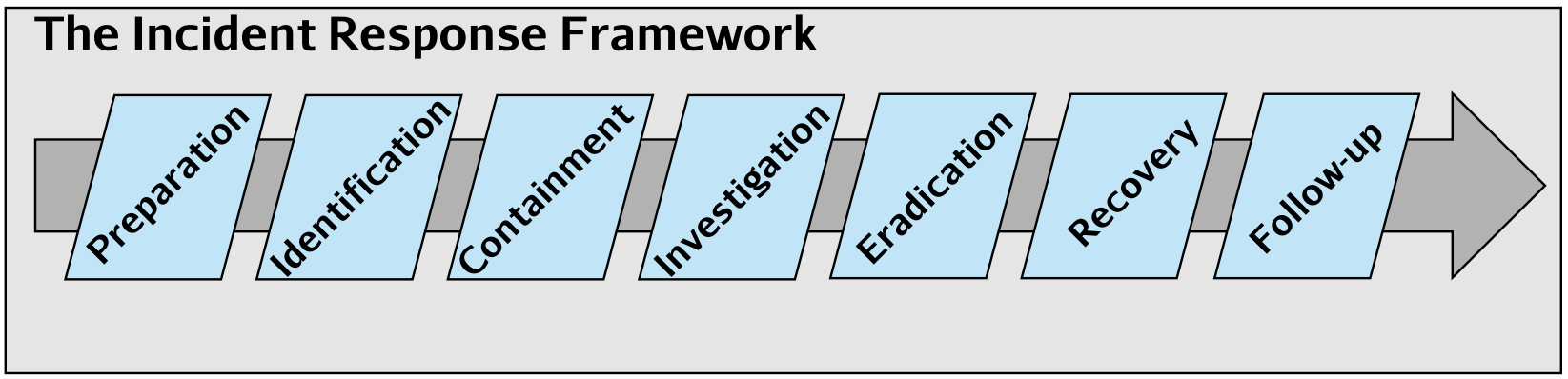
Metric Filters to trigger Alarms and SNS notifications.

The Incident Response Framework

For the purposes of this course and the exam, we can think of incident response in the cloud as a progression of seven steps or phases:



We will be referring back to this framework throughout the course. The next slide will detail the steps.



Note: The flow shown is for CloudWatch Logs using Metric Filters to trigger Alarms and SNS notifications. The CloudWatch events flow mentioned in the course is not shown.

Preparation Phase

This phase is all about doing everything we can to prevent breaches and failures. Eventually, some type of security event will happen, it always does. We are building walls and fortifying barricades here.

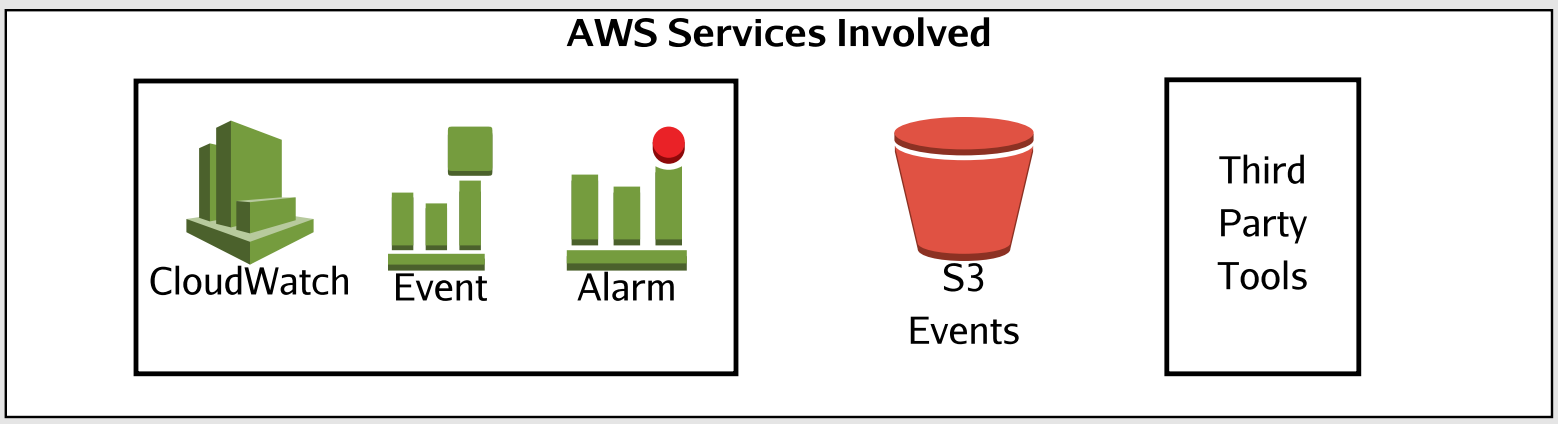
- Be Proactive
- Limit the Blast
- Log Everything
- Encrypt It All

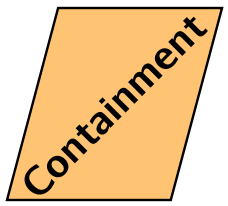
Identification Phase

Also known as the detection phase, this is where we discover an incident is occurring. We can do this through behavior-based rules we configure to help detect breaches. We must then determine the following:

- **Intention**- Knowing this can help us find compromised resources quickly
- **Blast Radius**- What resources were effected? How "deep" did the attack go?
- **Data Loss Protection**- A combination of encryption and access control. What did they get?
- **Resources needing "clean-up"**- What resources do we need to mitigate or isolate?

This phase can be very difficult, and we should be heavily dependent on automation to help us with detection. We can then react accordingly or even automate responses. There are also "stealth" techniques we can use to observe user behavior without being detected if there is questionable behavior.





Containment Phase

The containment phase is about removing the threat. There should be tools and processes ready to make changes to isolate any compromised resources. The ideal situation would be CLI or SDK scripts we can deploy very quickly when needed. For fast isolation, we need to have the following created or have scripts ready:

- A security group that restricts egress traffic and only allows management ports in
- A separate subnet with a restrictive NACL we can move resources to
- An S3 bucket policy that is designed to immediately stop spillage
- An explicit deny policy created in IAM (DENY *), quick removal of privileges
- A key policy that denies all decryption

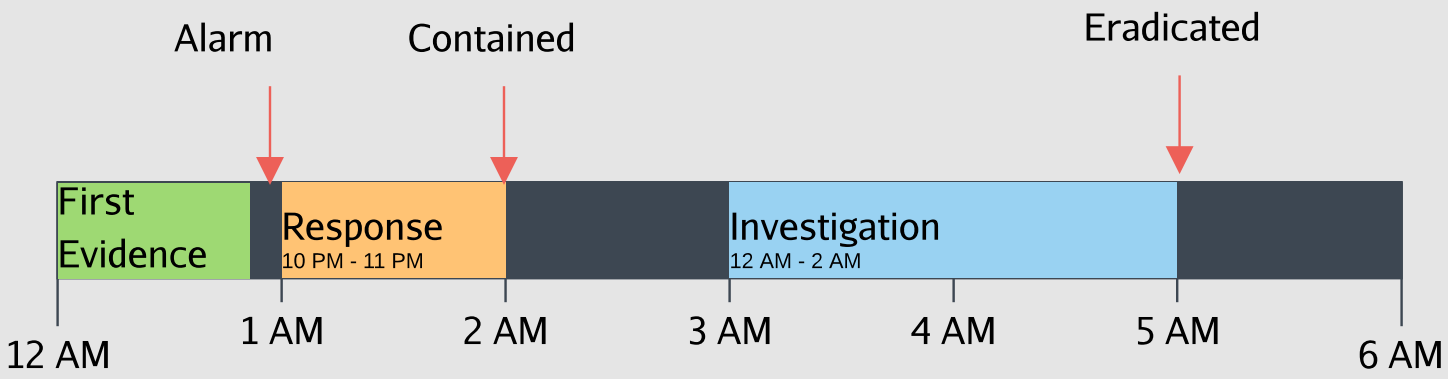
In addition, there may be additional activities we should perform:

- Snapshot volumes of compromised instances
- Stopping instances
- Disabling encryption keys in KMS
- Change Route53 record sets

AWS Services Involved

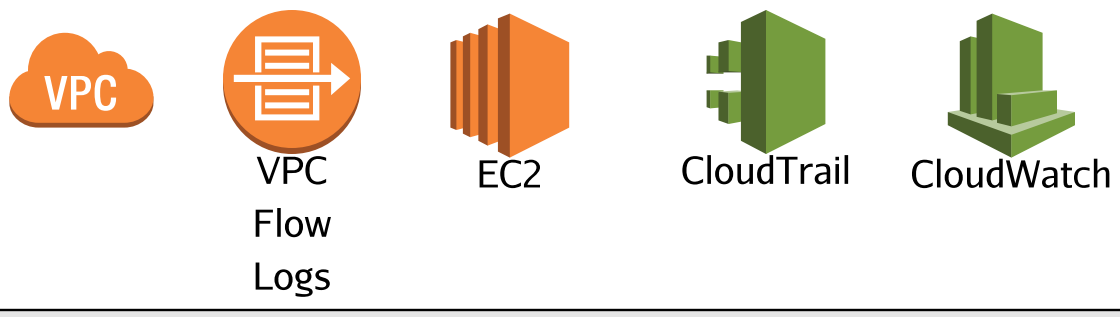
Investigation Phase

Investigation involves event correlation and forensics. We need to determine exactly what happened and when. We also need to determine if the threat is still viable. In most cases, a timeline will be the best representation:



As soon as we start our investigation, forensics can begin. Whether we use live box or dead box forensics here, proceed with caution and make sure it is in a safe, sandboxed environment.

AWS Services Involved



Eradication Phase

In this phase, we try to remove all the infections and compromises in our resources. In most cases, we can just delete the resources. There are some additional concerns when dealing with data.

If encryption was implemented correctly, data that was accessed should not be legible. In this case, we can do the following:

- Delete/disable any KMS keys
- For EBS, delete spilled files, create a new encrypted volume, copy all good files
- For S3 with S3 managed encryption, delete the object
- For S3 with KMS managed keys or customer keys, delete the object and the CMKs
- Secure wipe any affected files

If our data was not encrypted on EBS, we can attempt to sanitize the volume:

- Not recommended
- Create new volumes or instances with clean files or restore them from "last known good" backups

AWS Services Involved

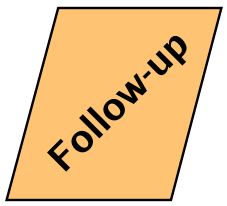
Recovery Phase

We need to put everything back to normal. This normally includes verifying eradicated resources and reversing the steps taken in the containment phase. Consider the following approach:

- Restore resources one at a time (or group)
- Use new encryption keys
- Restore network access
- Monitor, monitor, monitor
- Have the containment phase tools ready

This phase can be potentially dangerous as the forensic process may not have revealed everything.

AWS Services Involved



Follow-Up Phase

The follow-up phase can also be referred to as a post-mortem, lessons learned, or debriefing. It is important to make changes to the incident response plan based on what the team just experienced. With every test or real event, we should be iterating over the plan to mitigate issues or to make it more efficient. Consider the following:

- Testing and simulations are vital
- Must strive for efficiency (tagging, automation)
- Teams need experience

AWS Services Involved

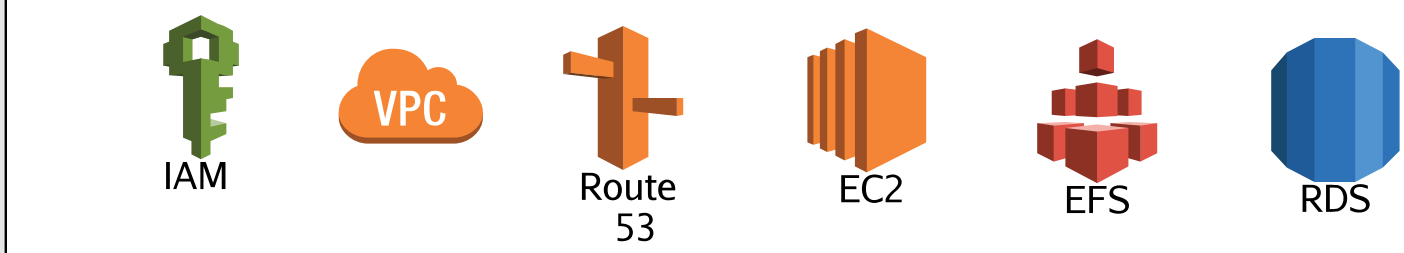
Any Relevant Services

Be Proactive

To be proactive is to work ahead of the incident and anticipating outcomes. Some of the best practices are:

- **Risk Management** - Determine where the different levels of risk are
- **Principle of least privilege** - No unnecessary permissions
- **Architect for failure** - High availability and fault tolerance always
- **Train for the real thing** - Test and simulate; a real incident is a horrible place to learn lessons
- **Clear ownership and governance** - Tag all resources so no time is wasted finding who or what group to contact
- **Data Classification** - Tagging data stores with classification can quickly identify spillage

AWS Services Involved



Limit the Blast

Careful planning can reduce the "blast radius" of any attack. The idea here is to segment/section off resources from each other. Some of the best practices are:

- Organizations - we can add accounts under our main account:
  - Benefits:
    - If there is a breach, it will not affect multiple accounts
    - Service Control Policies can be set so "child" accounts can be limited
- Using multiple Regions and VPCs can have a similar affect

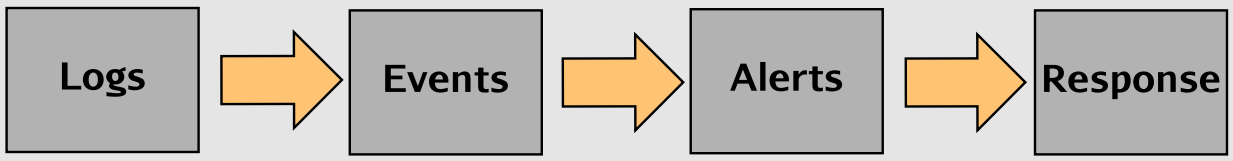
AWS Services Involved

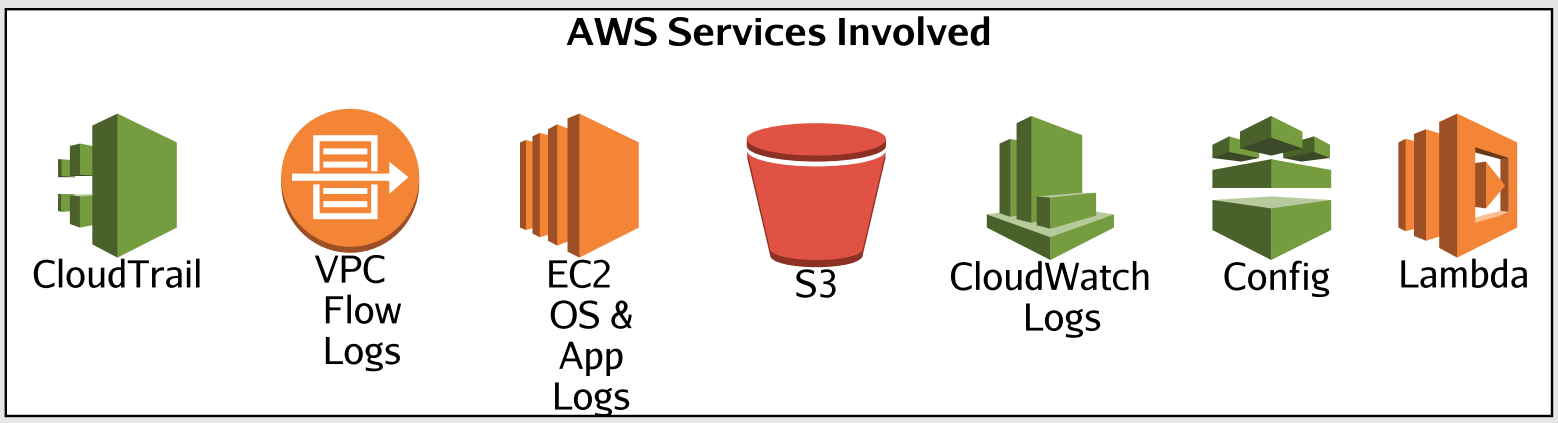
Organizations

Log Everything

Logging is the best way to collect information about our environments. Logs are also the beginning of being able to monitor a lot of resources. We can search our logs for information or automate responses based on patterns and alarms. Some of the best practices are:

- Centralized logging - collect all the logs from the organization in one place
  - Encrypt and protect (logs contain sensitive data that should not be clear text)
- It all starts with logs. The following pattern applies:





Encrypt It All

Encryption is the process of masking data by "scrambling" it using an algorithm and encryption keys. When done properly, the data on a volume or in a database cannot be interpreted if compromised or "spilled." There are two types of encryption:

- Server-side encryption (data-at-rest)
- Client-side encryption (data-in-transit)

Important: Treat your data as if everyone is looking at it all the time because they might be.

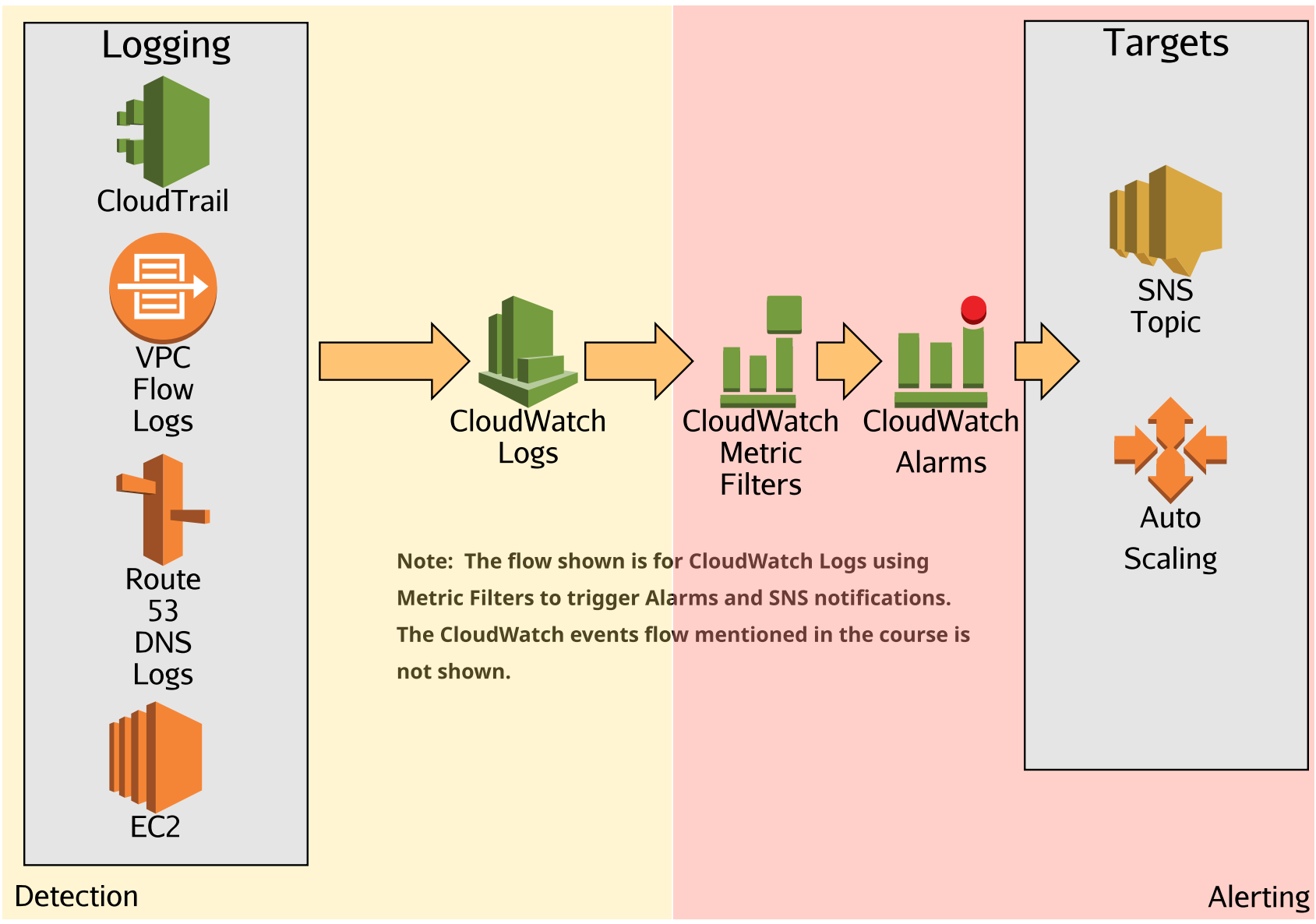
AWS Services Involved

Evaluate the configuration of automated alerting, and execute possible remediation of security-related incidents and emerging issues.

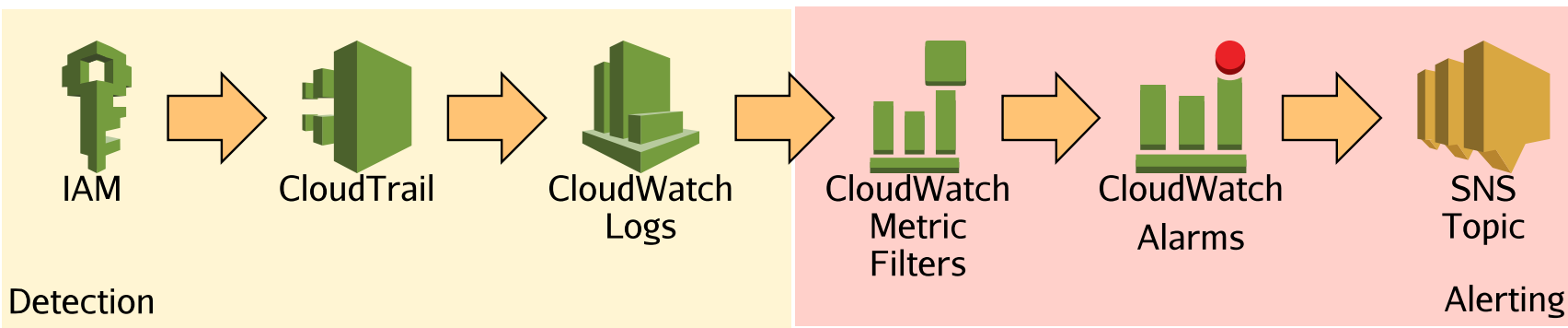
Note: The flow shown is for CloudWatch Logs using Metric Filters to trigger Alarms and SNS notifications. The CloudWatch events flow mentioned in the course is not shown.

Automated Alerting

The services we use in the cloud make scalability and reliability easy. These concepts should apply to our logging, monitoring, and alerting as well. Humans are much less reliable than cloud automation. Here is the base architecture:

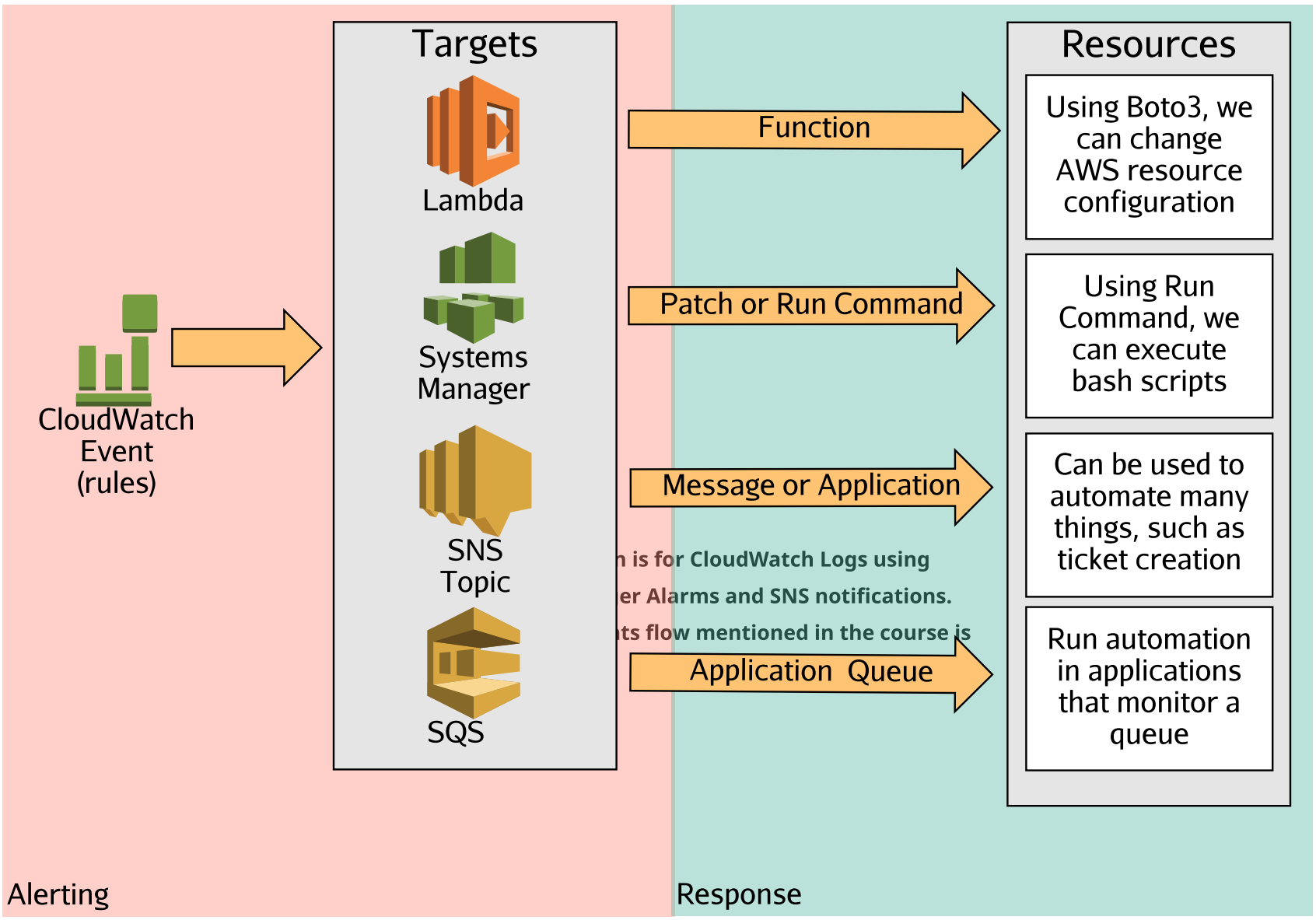


Keep in mind, there are multiple functions and uses for this workflow. We can use metric filters in CloudWatch Logs to trigger on specific API calls. For example, if we need to know when new IAM users are created. Here is an example workflow:



Automated Response

Once we get alerts generated in CloudWatch, there are a lot of target services we can trigger with those alerts. The most powerful and useful ones are Lambda and Systems Manager. We can configure these target services to automatically remediate our resources.



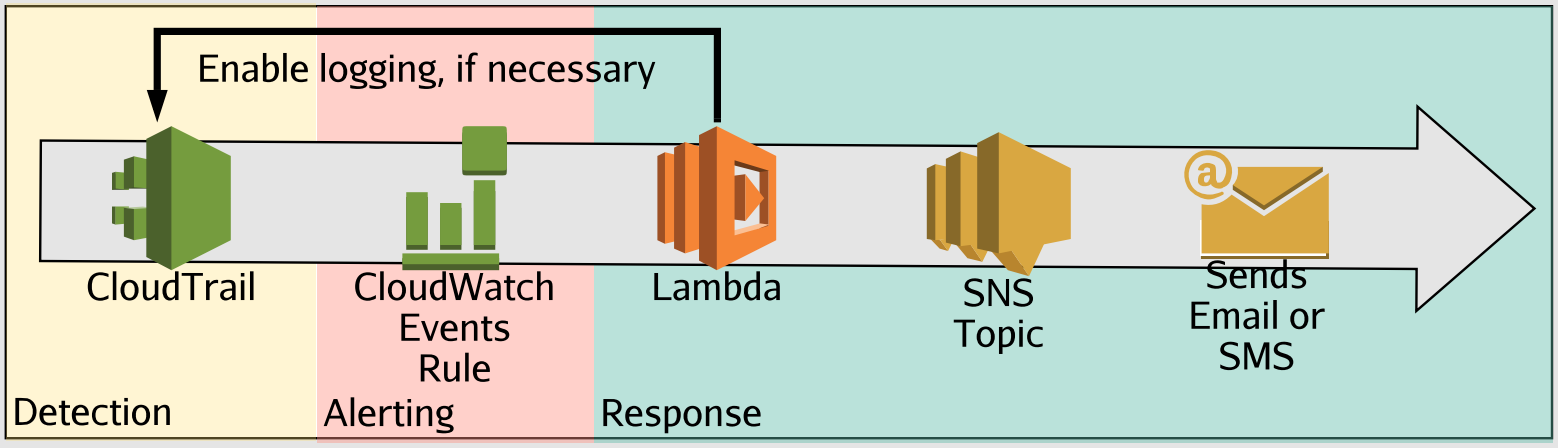
We can automate lots of our possible exposures out of existence. There are some other services that can use automation to help us audit resources and security. More on that later.

Even though the services and triggers may change, just remember the three actions or stages:

Automation Examples

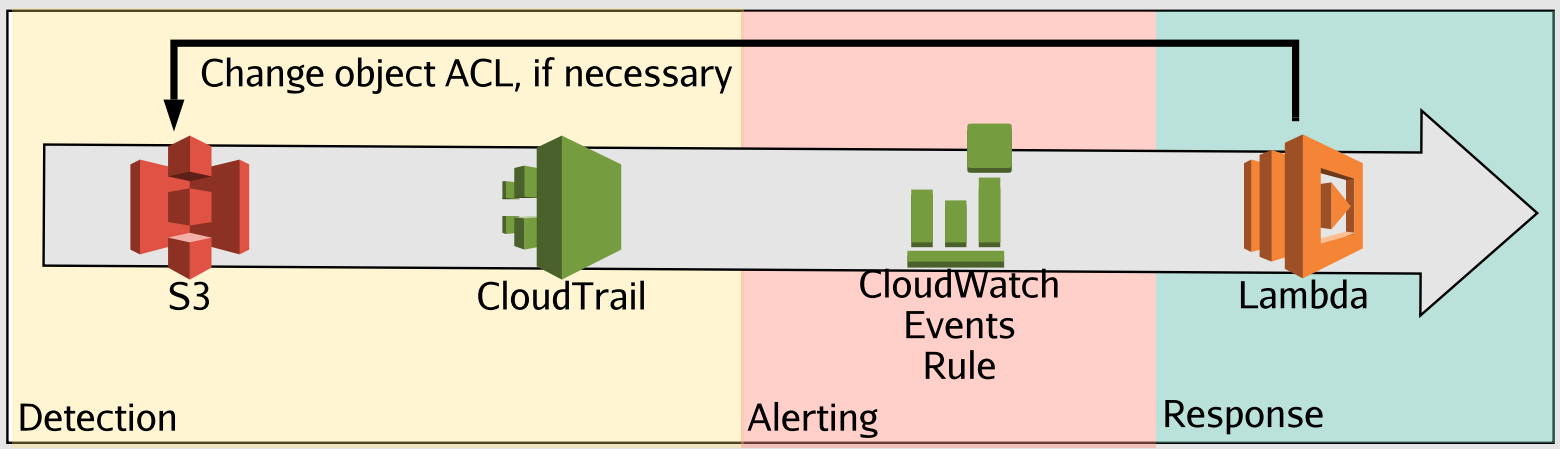
Scenario 1:

We need to ensure CloudTrail logging is never turned off in our account. We can set up automation that can detect a change in CloudTrail. We can then alert on that change with CloudWatch Events and trigger a Lambda function. That Lambda function responds by publishing to an SNS topic, then it will re-enable CloudTrail logging if the "StopLogging" API was called.

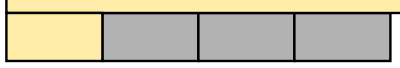


Scenario 2:

We have an S3 bucket that is restricted from public access by an ACL. We need to ensure that objects added to the bucket will not be publicly accessible. We can create automation to enforce these requirements. CloudTrail will log the "PutObject" and "PutObjectACL" API calls. CloudWatch Events can alert on those calls and invoke a Lambda function. The Lambda function responds by checking the object ACL and changing it if necessary.



Design and implement a logging solution.



CloudTrail

CloudTrail is the primary service we use for logging in AWS.

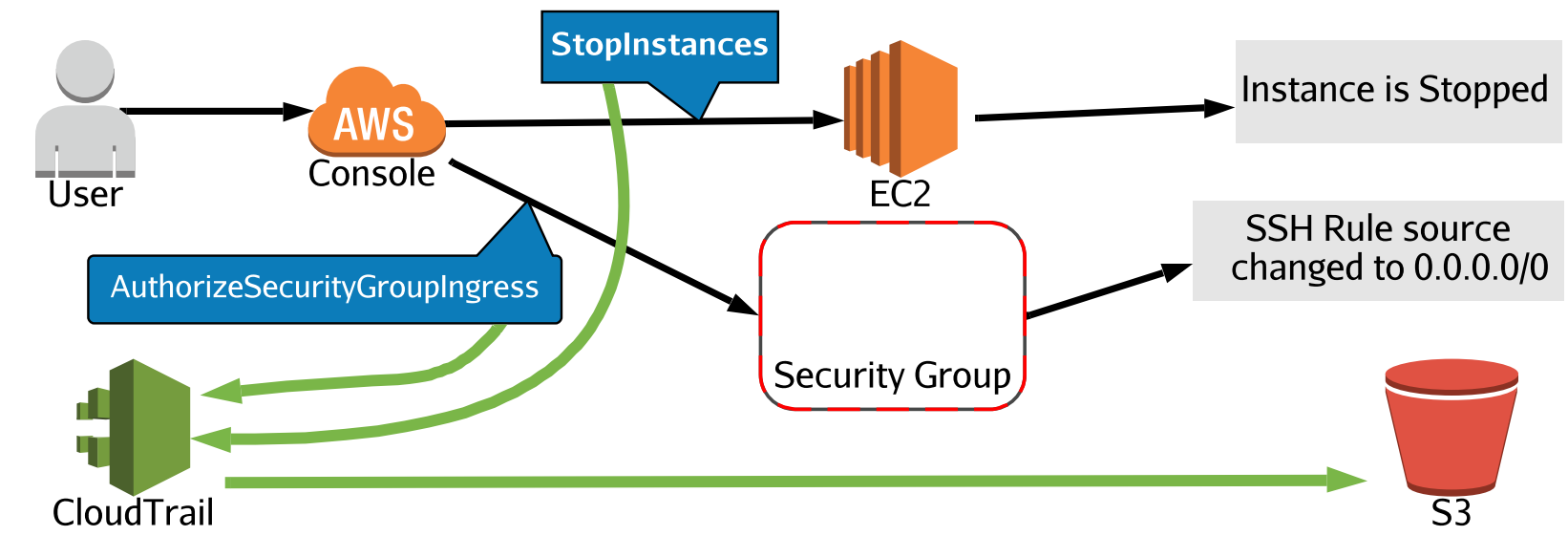
Important Features:

- It logs all the API calls in an AWS account (includes Console, CLI, API/SDK calls)
- Is enabled when your account is created
- Entries can be viewed using the **Event History** (past 90 days)
- **Trail**- a configuration allowing for logs to be sent to an S3 bucket:
  - Single region or multi-region trails can be configured
  - Trails can make multi-account logging possible, more on this later

Trails have several configuration options:

- **Management events**- enabling will log control plane events, such as:
  - User login events
  - Configuring Security
  - Setting up logging
- **Data Events**, which include:
  - Object-level events in S3
  - Function-level events in Lambda
- **Encryption** flexibility:
  - Encrypted in S3 server-side by default, can be changed to KMS
- The logs can be sent to an **S3 bucket** of choice and even prefixed (folders)

Scenario: An IAM user logs into the AWS Management Console. That user then proceeds to stop an EC2 Instance and edit a security group. CloudTrail will log all of these actions. Here is the workflow:



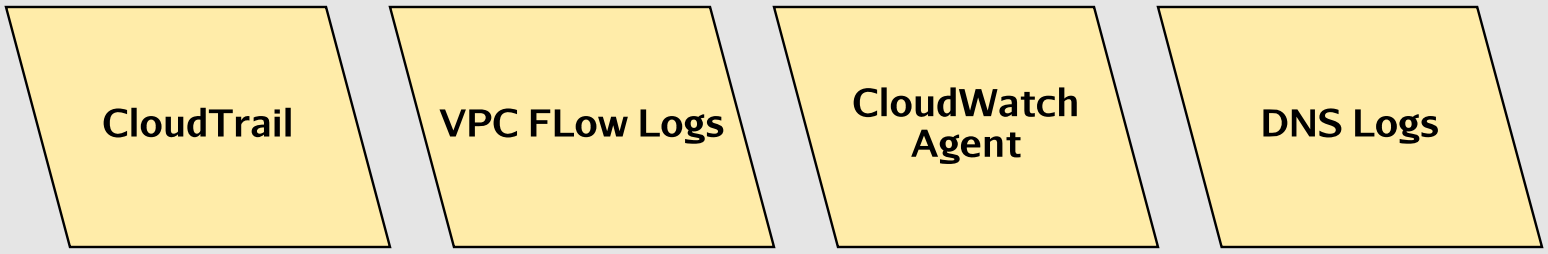
CloudWatch Logs

A central location which aggregates logs from many different services.

There are several components:

- **Log Events**- Record of activity recorded by the monitored resource
- **Log Streams**- Sequence of log events from the same source/application
- **Log Groups**- A collection of log streams with same access control, monitoring, and retention settings
- **Metric Filters**- Assigned to log groups, it extracts data from the group's log streams and converts that data into a Metric data point
- **Retention Settings**- Period of time logs are kept. Assigned to log groups, but applies to all the streams in a group (1 day to never expire)

Use Cloudwatch Logs to monitor, store, and access your log files from:



S3 Access Logs

The role of S3 storage in logging:

- The default storage for CloudTrail is S3
- CloudWatch Logs can be exported to S3
- S3 can help cost savings while still assisting with compliance:
  - Lifecycle policies to reduce storage costs
  - Archive older logs to Glacier

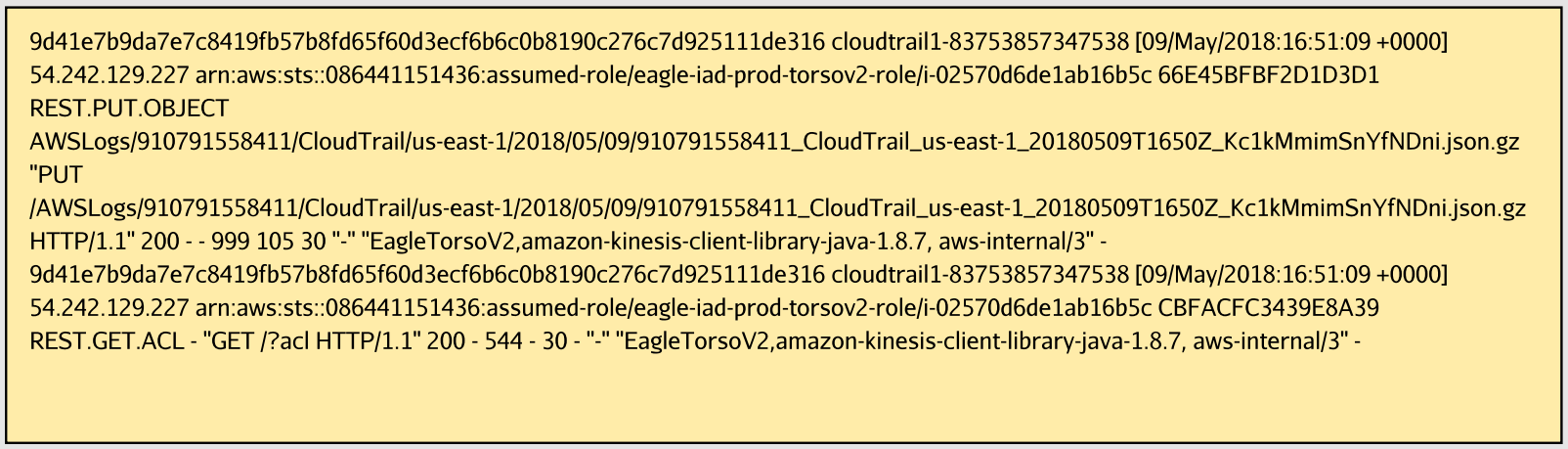
S3 access logs- an access logging mechanism in S3:

- Tracks access requests to buckets
- Each log event contains one access request
- Log events contain:
  - Requester
  - Bucket name
  - Request time
  - Request action
  - Response status
  - Error code

Important features of S3 access logging:

- The Log Delivery group must be granted write permission on the target bucket
- NOT near-real-time logging
- Logs are delivered on a "best effort" basis:
  - Newly enabled access logs might not be displayed in the target bucket for up to an hour
  - Changes to the target bucket might take up to an hour to propagate

Example Log (click to enlarge):



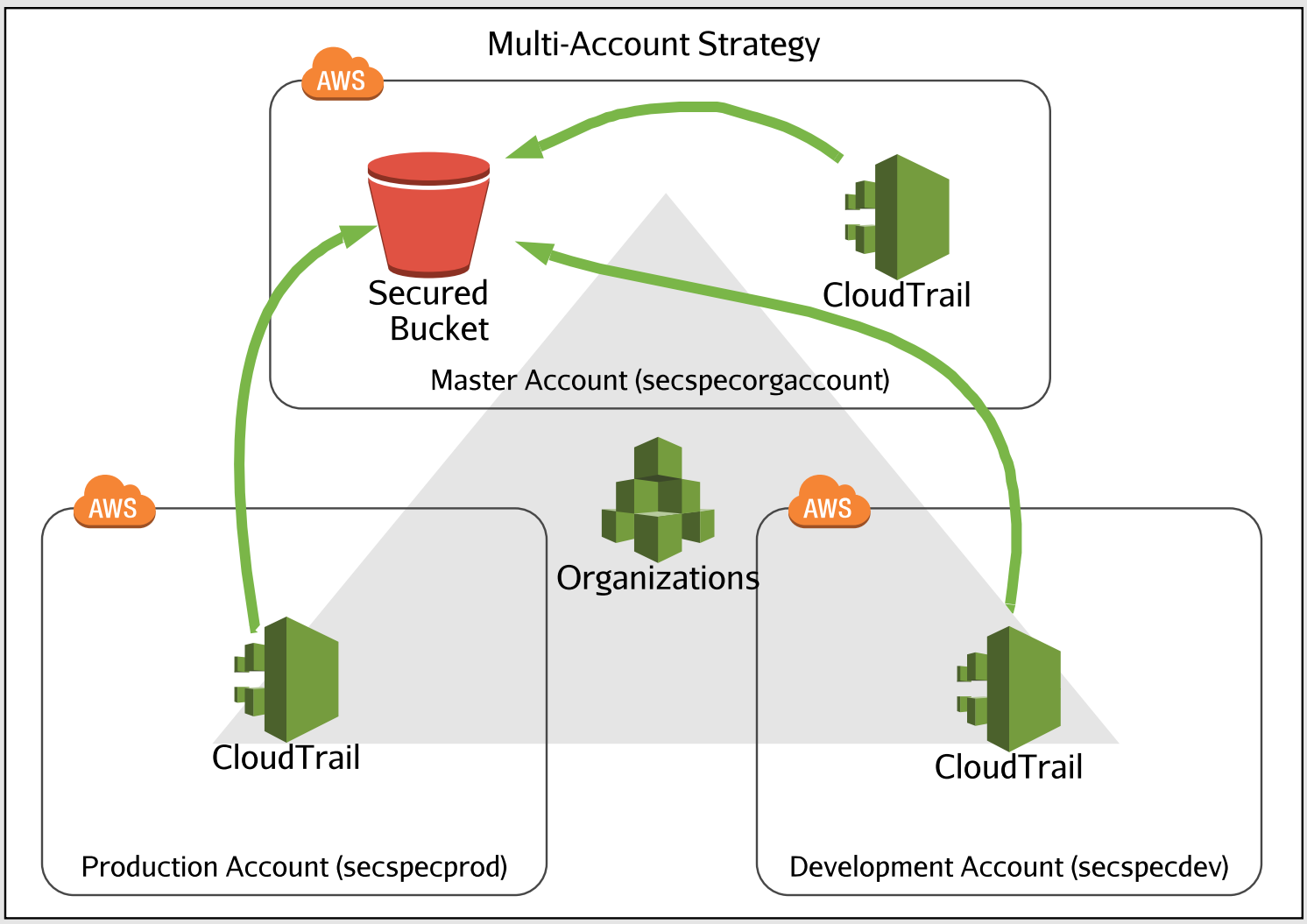
Centralized Logging

The Multi-Account Strategy:

- Use Organizations and set up accounts by environments or function:
  - Production, Development, Staging, etc.
  - Security, Administration, etc.
- Will help reduce the blast radius of any incident
- An additional layer of security:
  - Cross-account roles

Centralized logging:

- Logs should be contained in one location (the complete picture)
- Logs should be read-only for most job functions (including security)
- Logs should be encrypted (KMS preferred)
- Roles can provide cross account access



Troubleshoot logging solutions

Troubleshooting Log Aggregation - Common Issues

CloudTrail

- All regions? (for IAM)
- Data Events? (for S3 and Lamda)
- S3 bucket name

CloudWatch Logs

CloudTrail

- Trail and Logging enabled
- CloudWatch Logs configured
- Role for CloudWatch Logs
- Log group name

VPC Flow Logs

- Flows enabled on VPC or Subnet
- Check filter
- Role for CloudWatch Logs
- Log group name

Route 53 DNS Logs

- Query logging configured
- New or existing log group
- Log group name

EC2

- Agent installation and config
- Agent must be started
- Role for CloudWatch logs

CloudWatch Logs

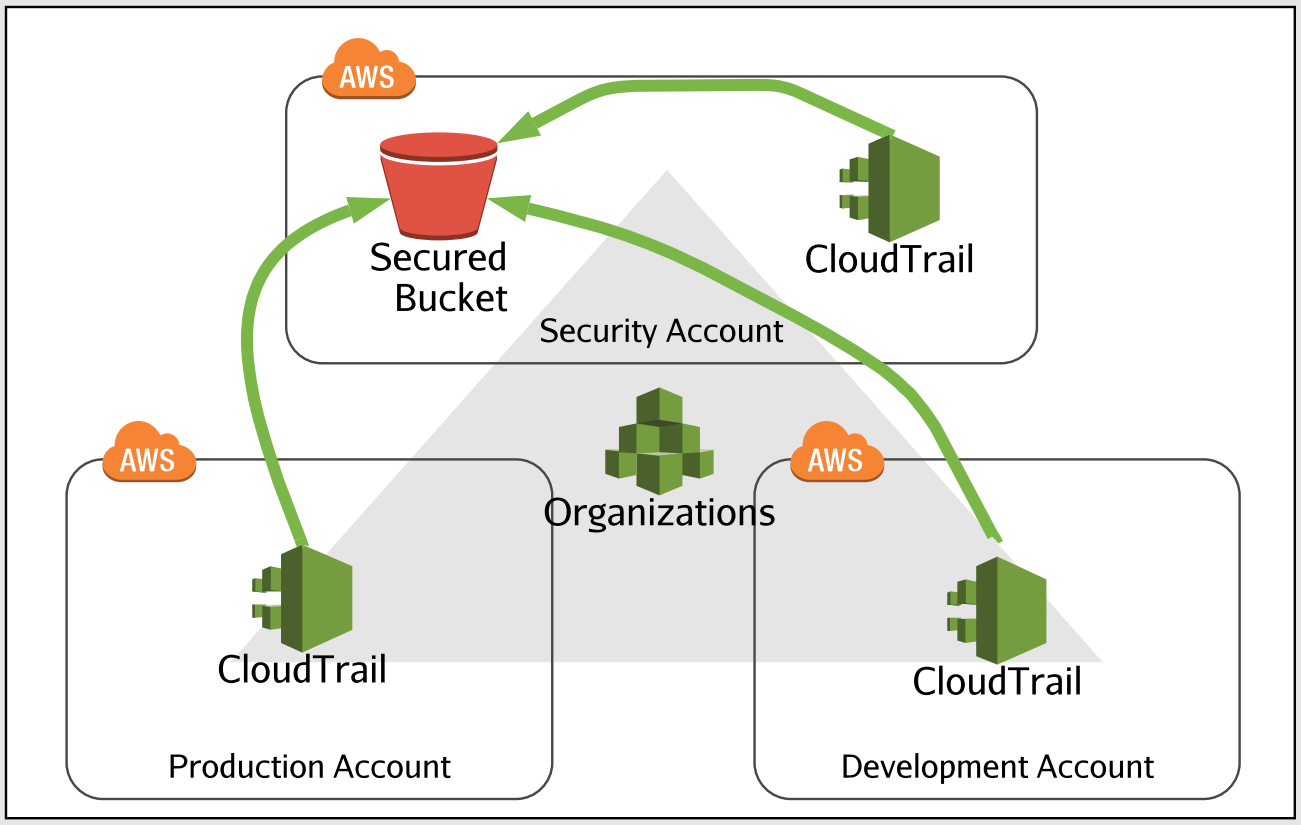
S3 Access Logs

- Confusion over source and destination buckets
- Log Delivery group permissions

Troubleshoot Multi-Account Logging

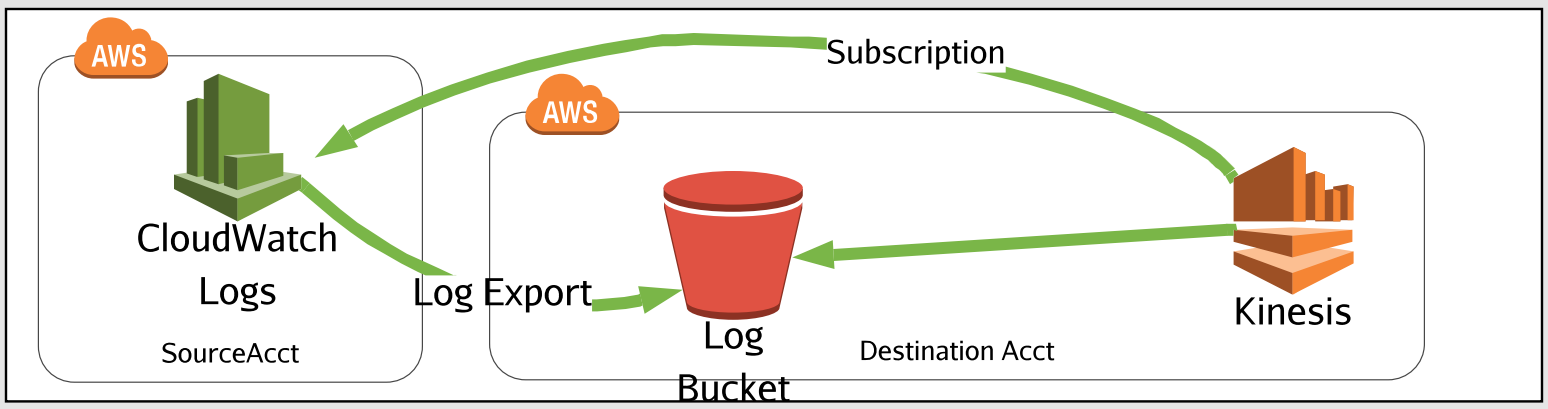
CloudTrail logging across multiple accounts:

- S3 bucket policy for accounts sending the logs
- Bucket names should be double-checked for accuracy



CloudWatch Logs across multiple accounts:

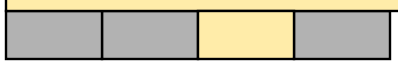
- CloudWatch does not send logs directly to another account
- S3 access issues blocking exports (scheduled or manual)
- Kinesis stream is not setup properly (only target for "real-time" logs)



Common Issues with Multi-Account Logging

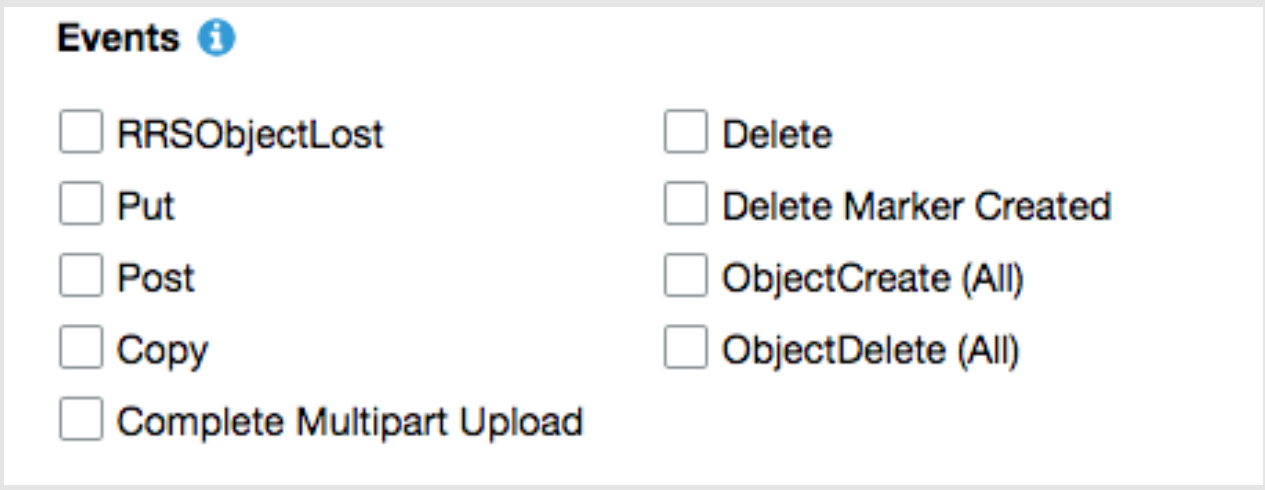
- Issues will mostly be around permissions (roles and resource policies)
- Make sure all permissions only grant read-only access

Design and implement security monitoring and alerting.



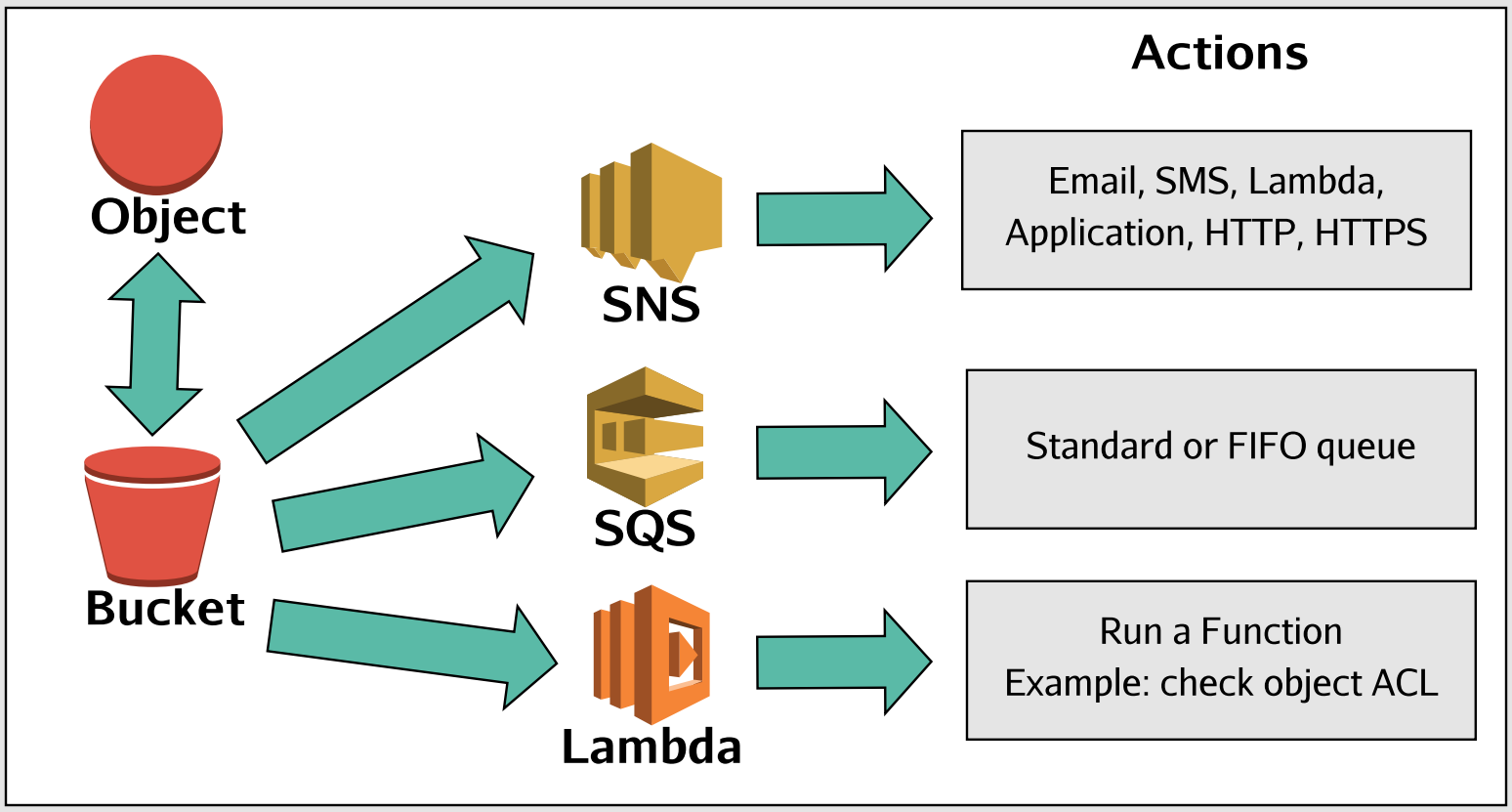
S3 Events

Allows for alerting on object actions in S3:



Then we send notification to three different services:

- SNS Topic
- SQS Queue
- Lambda Function

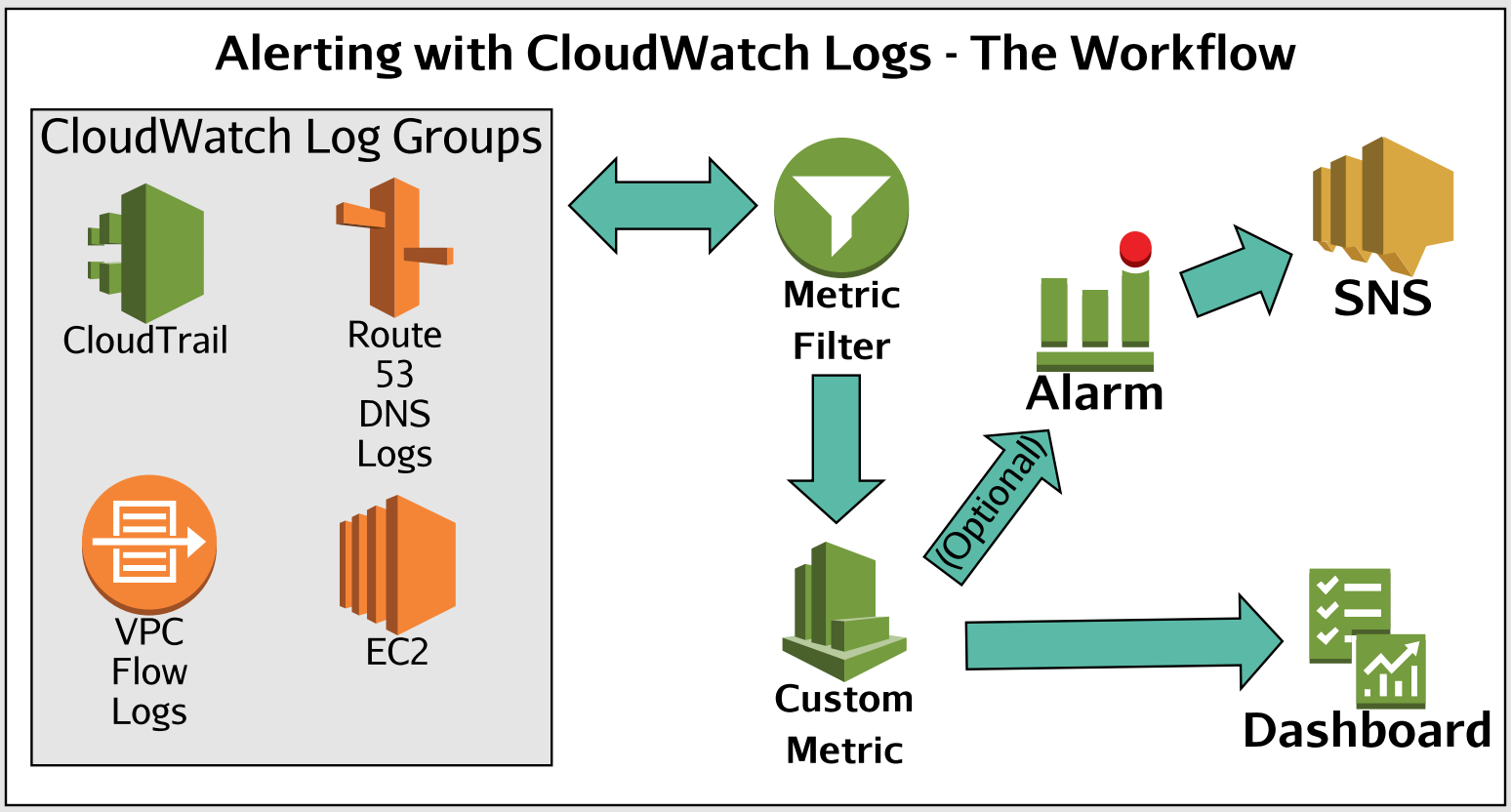


CloudWatch Logs: Monitoring and alerting

We can create custom metrics and alarms from our logs in CloudWatch.

Components:

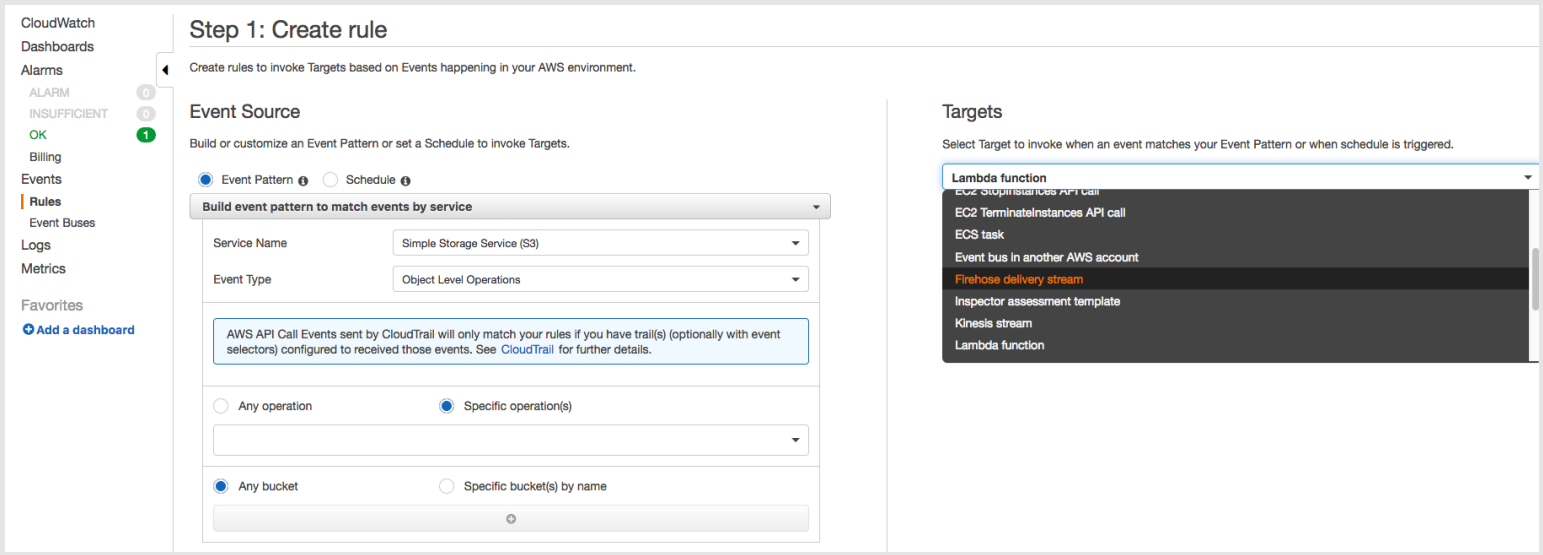
- **Metric Filters**: Used to create a custom metric from log data:
  - Assigned at the Log Group level:
    - Will filter all the streams in that group
  - Uses a filter and pattern syntax:
    - Example: **{ $.eventName = "CreateUser" }**
- **Metric Namespace:** The "folder" or category the custom metric will appear in
- **Metric Name:** The name given to the custom metric
- **Alarms:** Assigned to the filter:
  - Alarms can trigger:
    - SNS topics
    - AutoScaling Actions
    - EC2 actions (if the metric chosen is related)
- Can **export** log data to S3
- Can **stream** log data to Lambda and Elasticsearch Service



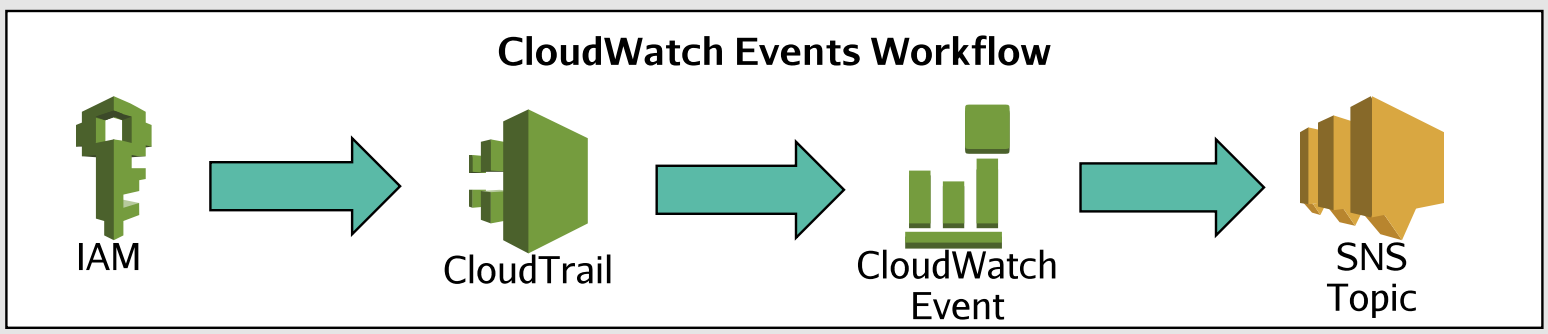
CloudWatch Events

CloudWatch Events are similar to alarms. Instead of configuring thresholds and alarming on metrics, CloudWatch Events are matching event patterns. They use targets to react and:

- Performs in near real-time
- Consists of three parts:
  - **Event Source:** An operational **change** in a service or a **scheduled** event
  - **Rules:** Route matching events to targets
  - **Targets:** The services that will react to the event:
    - There can be more than one
    - Some of the services that can be targets:
      - EC2, Lambda functions, ECS tasks
      - Kinesis Data Streams and Firehose
      - Systems Manager Run Command and Automation
      - Code* projects and pipelines
      - SNS and SQS

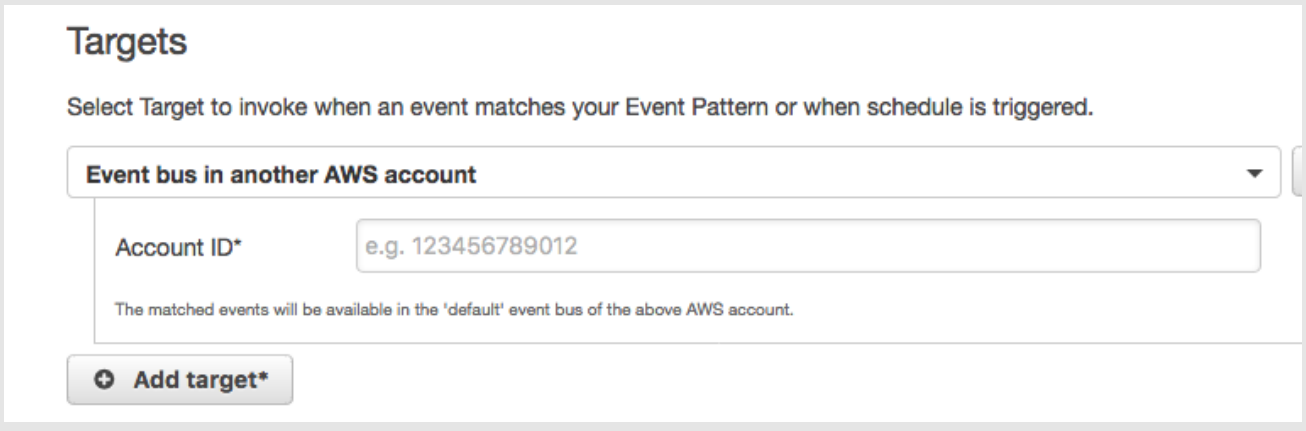


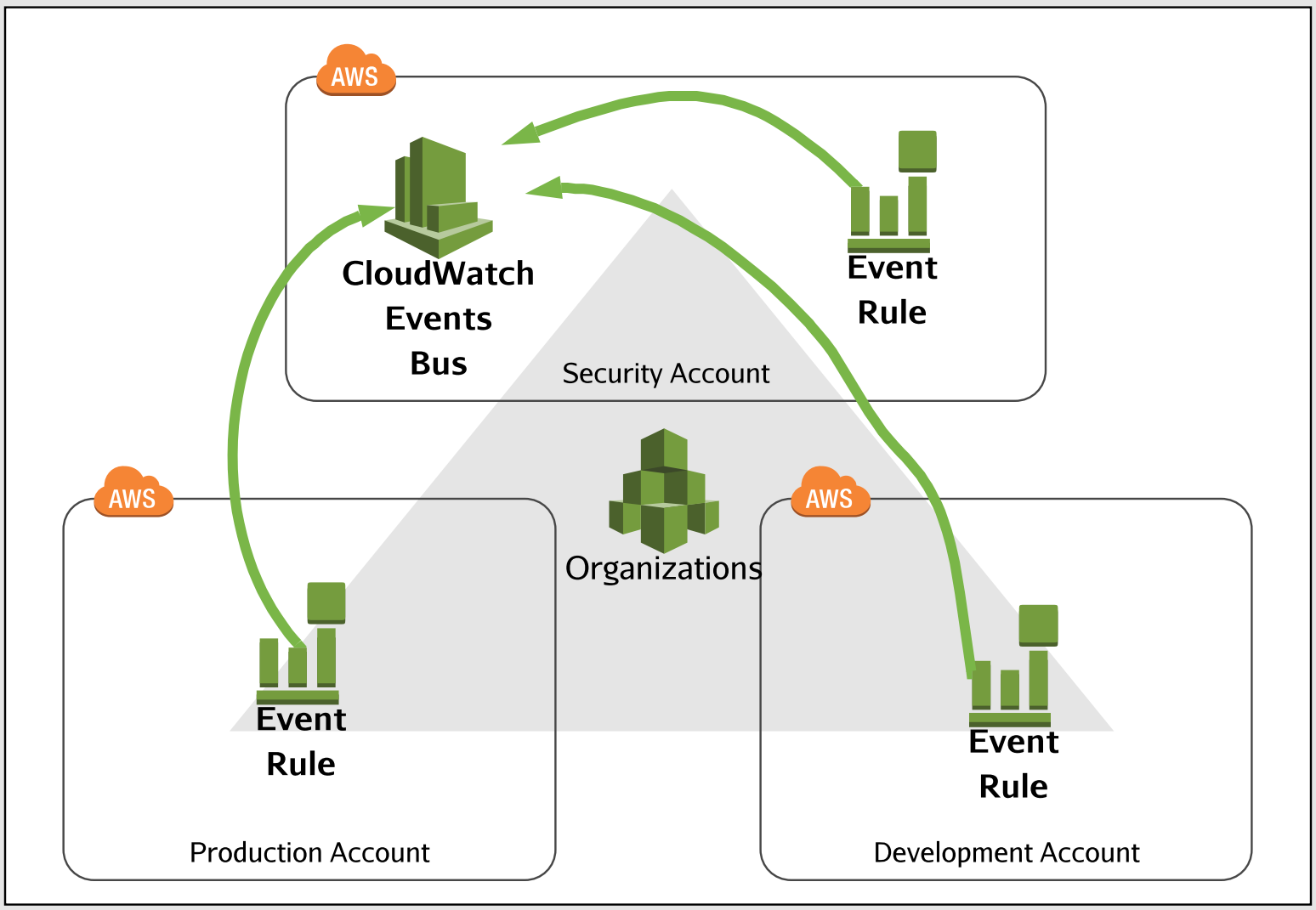
- Examples:
  - Alerting on object uploads in S3 (can trigger automatic ACL remediation)
  - Alerting on EC2 instance state changes (can trigger actions on the instances)
  - Alerting on user creation in IAM (earlier example)



CloudWatch Buses

- Newer feature that released about a year ago
- Allows different AWS accounts to share CloudWatch Events
- Can collect events from all your accounts together in one account
- Must grant an account permission by adding and then sending the account number to the receiving account bus configuration:
    - The sending account sends an event to an Event bus target





AWS Config

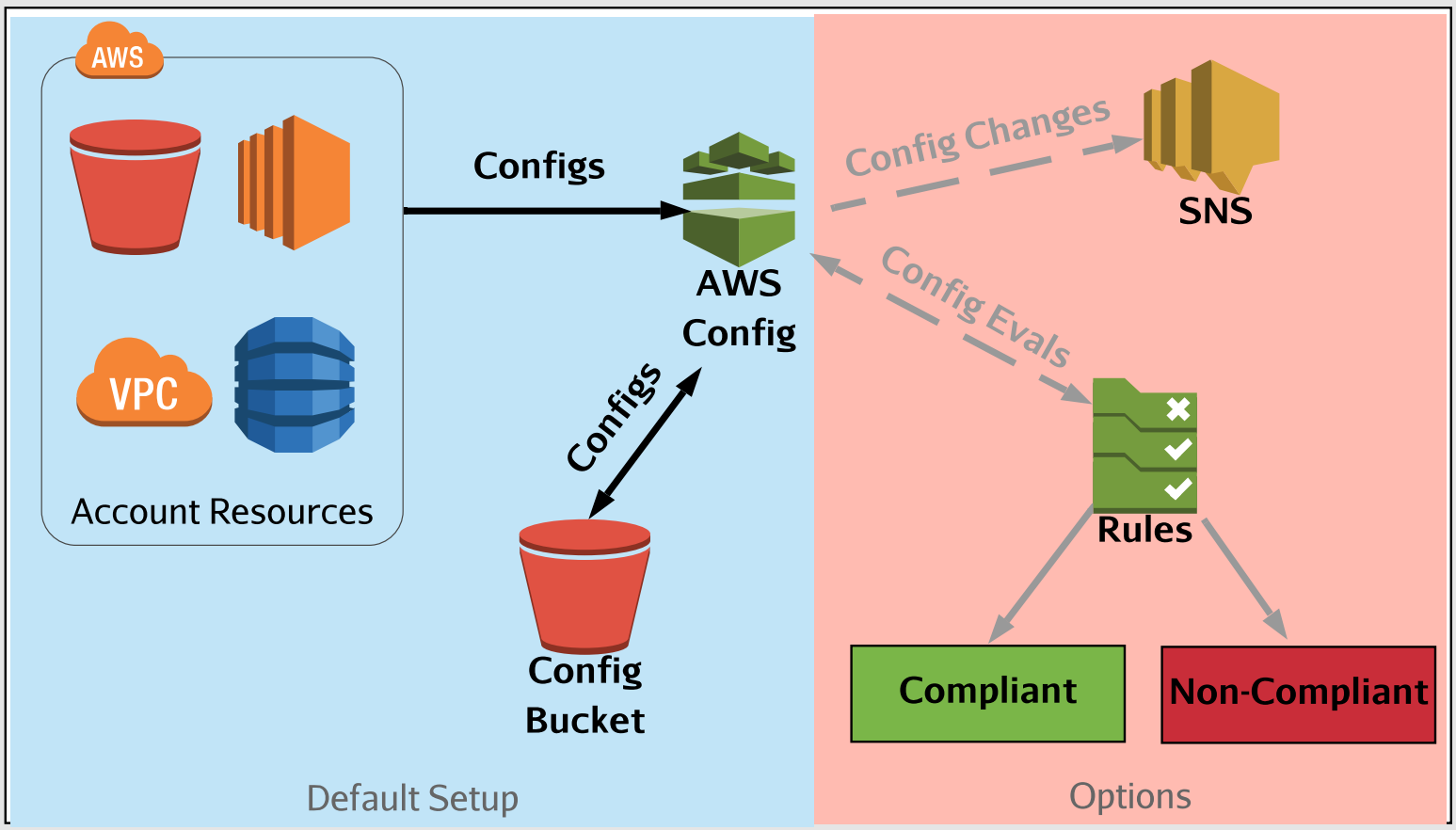
A detailed view of the configuration of AWS resources (EC2, EBS, security group, VPC, etc.). The following is a complete list of supported services available in AWS documentation.

With AWS Config, you can:

- Evaluate resource configurations for desired settings
- Get a snapshot of the current configurations associated with your account
- Retrieve configurations of resources in your account
- Retrieve historical configurations
- Receive a notification for creations, deletions, and modifications
- View relationships between resources (e.g., members of a security group)

Uses of AWS Config:

- Administering resources:
  - Notification when a resource violates configuration rules
- Auditing and compliance:
  - Historical records of configurations are sometimes needed in auditing
- Configuration management and troubleshooting:
  - Configuration changes on one resource might affect others
  - Can help find these issues quickly and can restore last known good configurations
- Security Analysis:
  - Allows for historical records of IAM policies:
    - For example, what permissions a user had at the time of an issue
  - Allows for historical records of security group configurations.



AWS Inspector

Allows for :

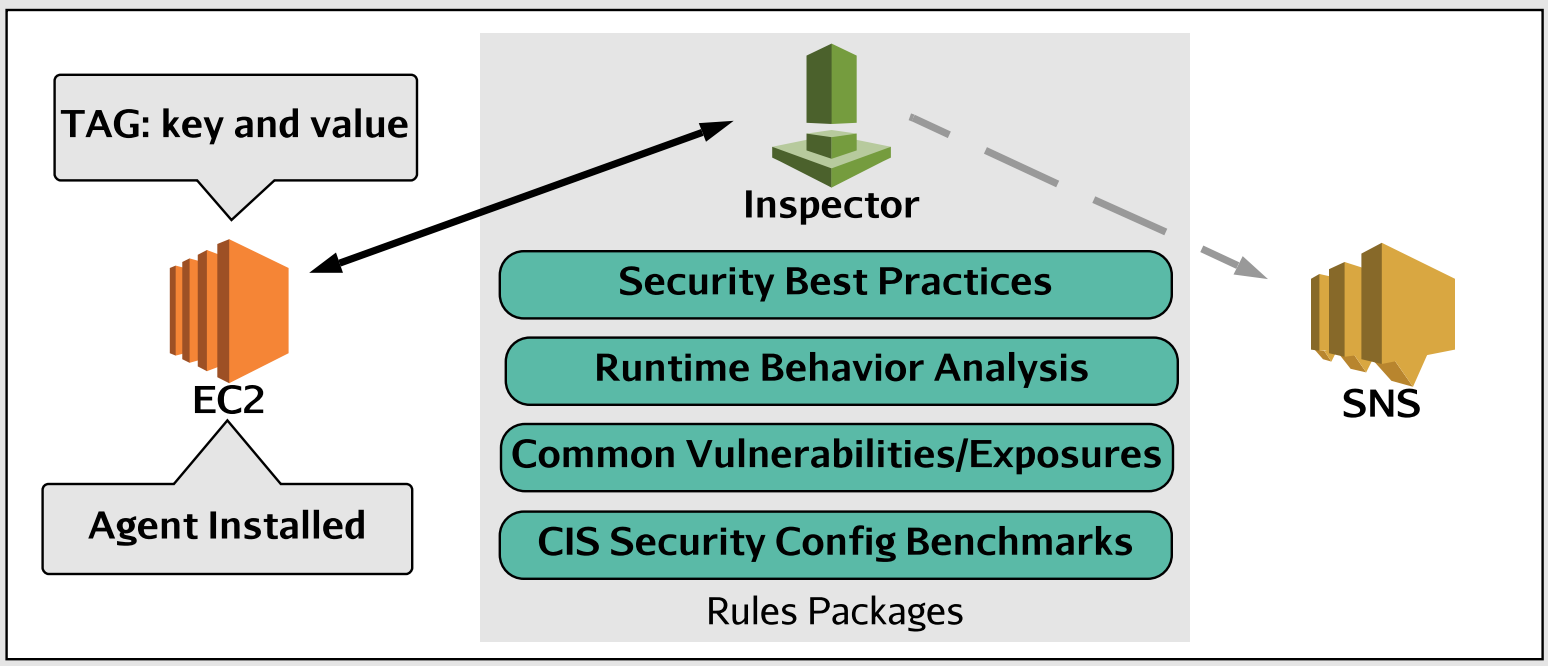
- Analyzing the behavior of your AWS resources
- Identifying potential security issues

Target: A collection of AWS resources
Assesment Template: Made up of security rules and produces a list of findings
Assessment Run: Applying the assessment template to a target

Features:

- Configuration Scanning and Activity Monitoring Engine:
  - Determines what a target looks like, its behavior, and any dependencies it may have
  - Identifies security and compliance issues
- Built-In Content Library:
  - Rules and reports built into Inspector
  - Best practice, common compliance standard, and vulnerability evaluations
- Detailed recommendations for resolving issues:
  - API Automation
  - Allows for security testing to be included in the development and design stages

NOTE: AWS does not guarantee that following the provided recommendations will resolve every potential security issue



Automation Review (Exam Tips)

CloudWatch Events vs. Logs:

- Events are the alerting mechanism of a majority of automation workflows
- **Important:** CloudTrail API calls can trigger an Event (even without sending to Logs)
- More information in the alert than metric filters with Logs:
  - Events are better for Lambda functions and notifications
- More target options:
  - CloudWatch Log (metric filter) alarms can only target SNS

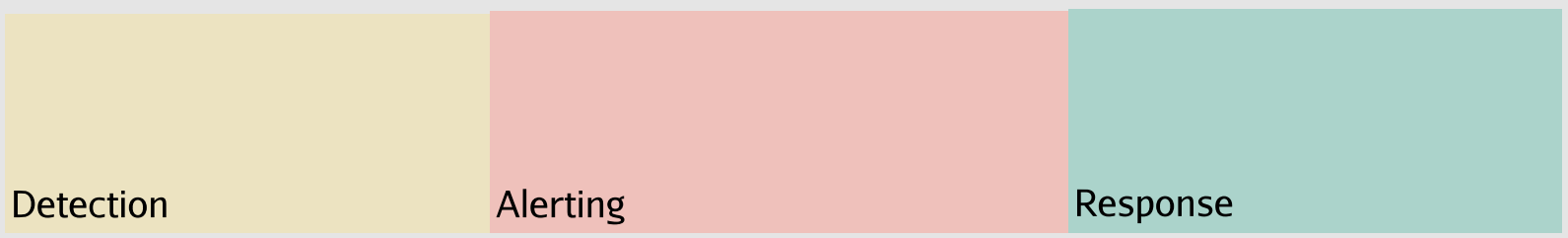
Multiple Paths:

- There are multiple ways to detect and respond in some cases:
  - Some services can only use SNS directly for alerting:
    - We can automate from there using SNS subscriptions (i.e., Lambda or application) along with email and SMS
  - **NOTE:** These services also use API calls logged to CloudTrail:
    - We can use the CloudTrail API call log to trigger CloudWatch Events

Strategy:

Read carefully!!!!

- Draw out (sketch) the automation workflow from the question so you can visualize it:
  - There are a lot of details in the questions; you don't have to picture it in your head
  - Having a visual model can also help eliminate wrong answers quickly
- Remember the three actions of automation workflows:



Troubleshoot security monitoring and alerting.

Troubleshoot Monitoring and Alerting

There are many things that can go wrong with our monitoring and alerting. It is important to troubleshoot these issues in an organized fashion. I suggest starting where the detection occurs and working your way through one "link" at a time.

Common Issues:

- General configuration issues:
  - Wrong resource name when "connecting " resources
- Typos- this should be checked during configuration, but we all make mistakes:
  - Huge with API calls, filter patterns, and Lambda functions
- Not waiting long enough after making changes or new configurations
- Roles do not have sufficient permissions (AWS does not always tell you with an error):
  - This includes targets and subscriptions to encrypted resources- must include KMS policies
  - More on this in the IAM section of the course

Good to know
According to AWS documentation, IAM API calls are only supported in us-east-1.

Using Automation to Monitor Automation

CloudWatch Events

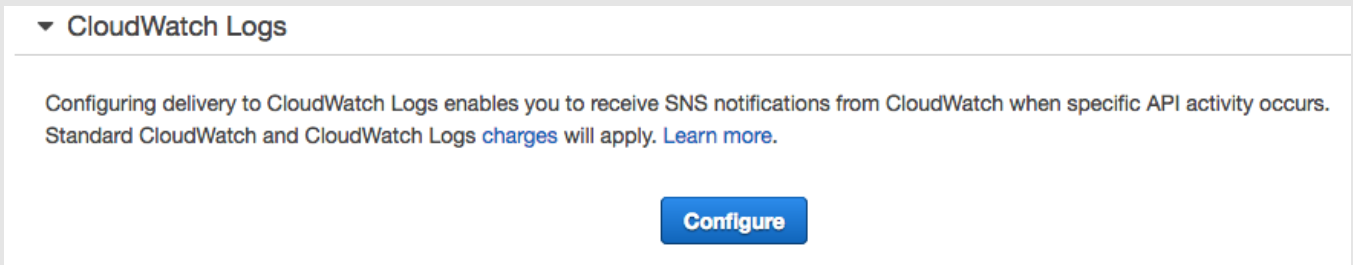
With CloudWatch, we can create an alarm on Events Metrics. We can use FailedInvocations to notify us when our CloudWatch Events rules are broken.

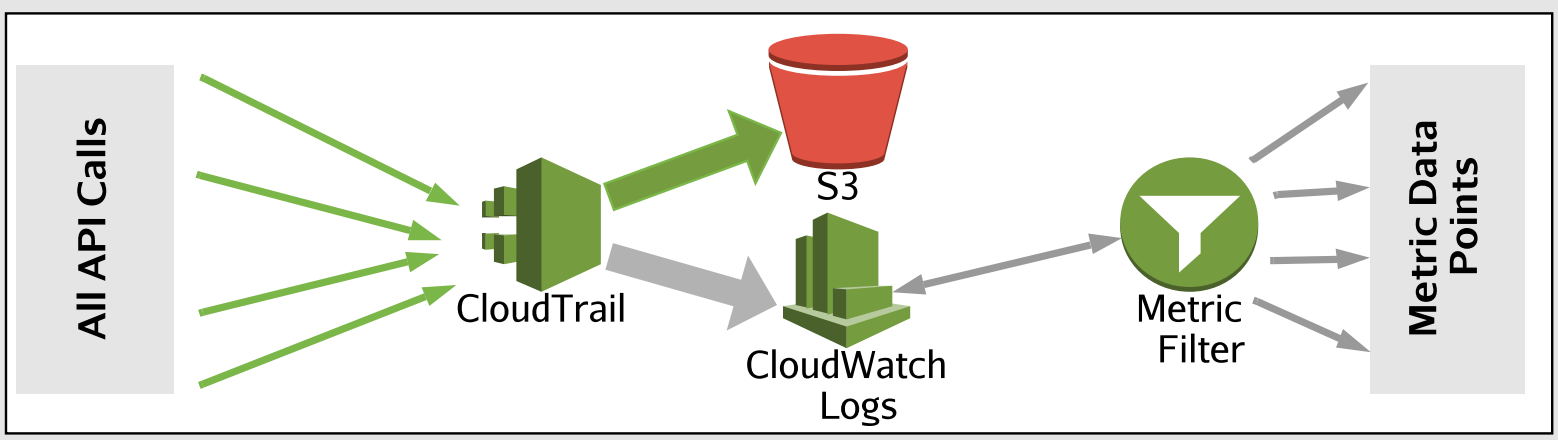
Lambda Functions

Lambda delivers logs to CloudWatch Logs. It will log errors with invocations. We can then alarm on this using a metric filter and notify via SNS.

CloudTrail

A trail in CloudTrail can be configured to send copies of logs to CloudWatch Logs.



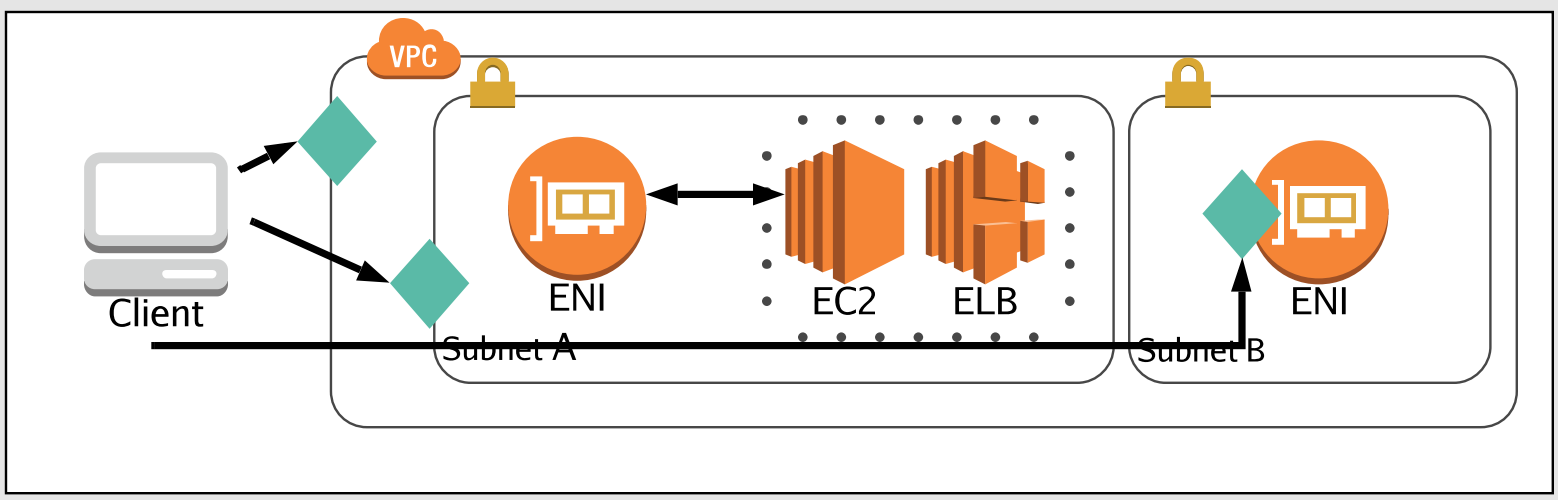


VPC FLow Logs

VPC Flow Logs are comprised of IP traffic information. These logs are sent to CloudWatch Logs by default. Flow logs are useful for troubleshooting network conversations and can be assigned to a VPC, a subnet, or an Elastic Network Interface (ENI).

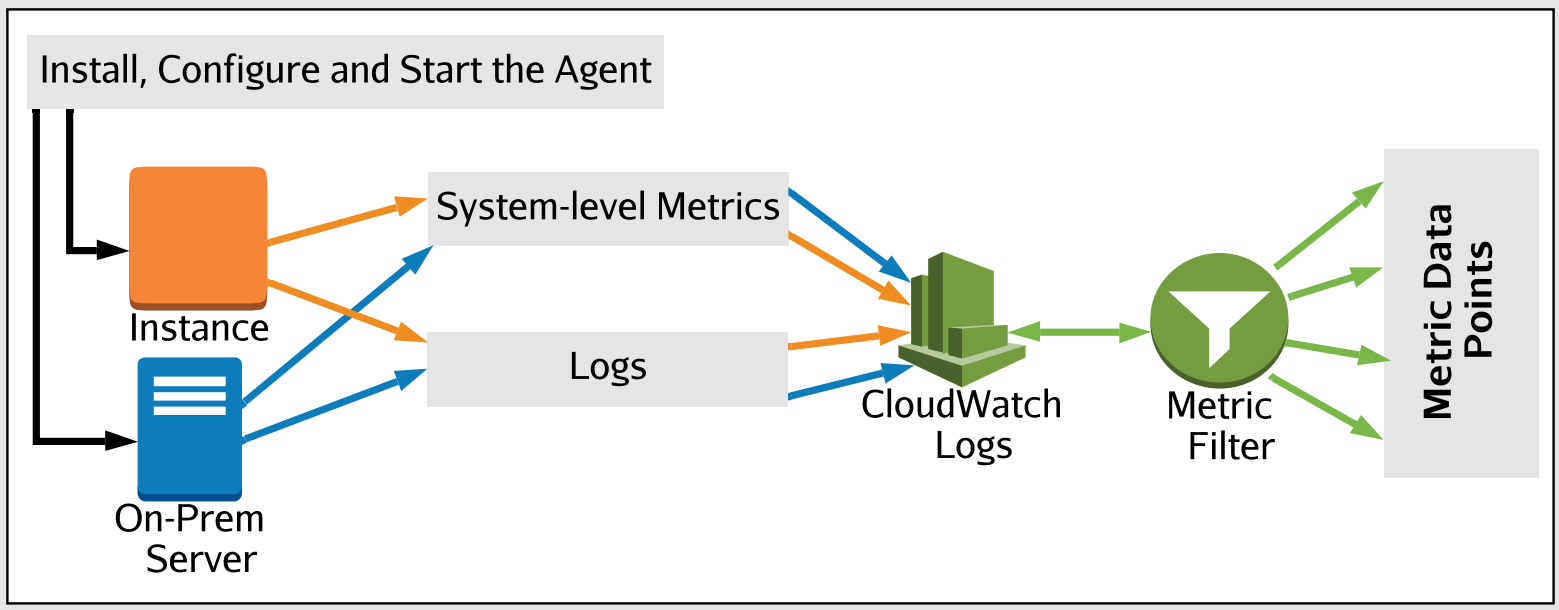
Here is how it works: (= capture points)

Examples



CloudWatch Agent

CloudWatch Agent allows us to collect additional in-guest metrics and logs from EC2 (and on-prem). These include memory, disk-use percentages, and swap file usage. It can also collect logs from the application. These metrics and logs are sent to CloudWatch Logs.



DNS Logs

DNS Query Logs can be enabled on Route53 hosted zones and sent to CloudWatch. Route 53 uses common DNS return codes in the log and includes the edge location (based on airport codes). These logs can be used to determine when there is a DNS problem in an application.

These logs are only available for hosted zones where Route53 is the endpoint (no outside hosting). Also, the logs are not available for private hosted zones.

An Example Log:

```
1.0 2018-05-15T16:20:24Z Z3URAC2AGNUNNQ sysopscodex.com A NOERROR UDP ATL51 68.87.56.203 -
1.0 2018-05-15T16:26:11Z Z3URAC2AGNUNNQ sysopscodex.com A NOERROR UDP ATL51 68.87.56.203 -
1.0 2018-05-15T16:32:12Z Z3URAC2AGNUNNQ sysopscodex.com A NOERROR UDP ATL51 68.87.56.203 -
1.0 2018-05-15T16:40:11Z Z3URAC2AGNUNNQ www.sysopscodex.com A NOERROR UDP ATL51
68.87.56.203 -
1.0 2018-05-15T16:44:00Z Z3URAC2AGNUNNQ sysopscodex.com A NOERROR UDP ATL51 68.87.56.203 -
```

Design edge security on AWS.

Amazon CloudFront - Global Content Delivery Network (CDN)

Architecture

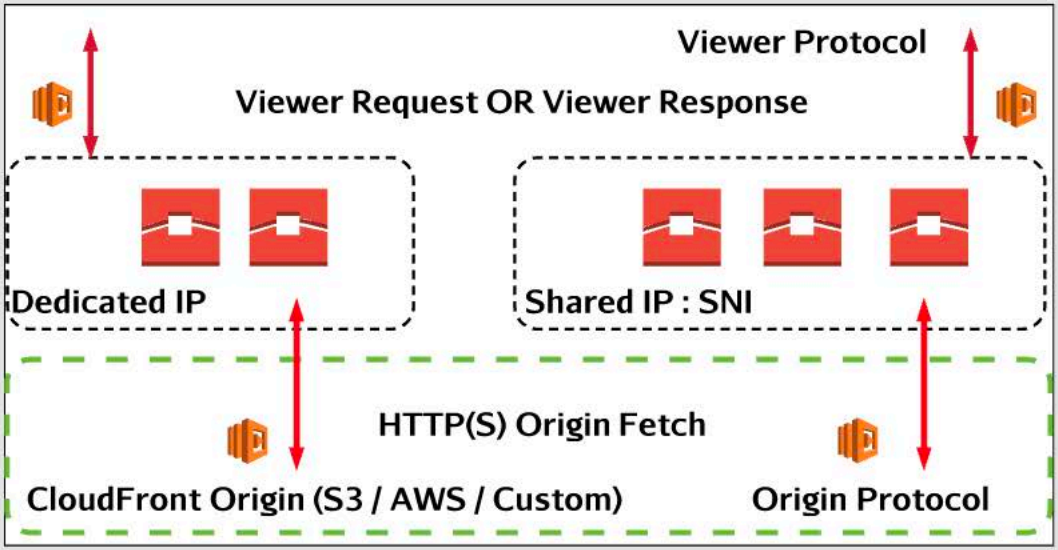
CloudFront is a global CDN operating from AWS Edge Locations. Connections to a CloudFront distribution can utilize HTTP or HTTPS. Connections from CloudFront to your content (origin server) can occur using HTTP or HTTPS. It also removes many invalid HTTP requests at the edge - basic filtering.

CloudFront does support SNI (**S**erver **N**ame **I**dentifier) - Edge Location IP's can be shared. Dedicated IP SSL is supported in ALL browsers, but costs extra. SNI has no extra cost, but browsers need to support it.

Viewer protocol policy (per distribution) allows redirection of HTTP->HTTPS.

Advanced Security Features

- Integrates with AWS WAF
- Supports full access control and signed URL's/cookies
- Provides basic white/blacklist geo-restriction per distribution
- Can integrate with 3rd party solution using signed URL's/Cookies
- Field-Level Encryption
- Supports Lambda at the edge



Design edge security on AWS.

Restricting S3 to CloudFront

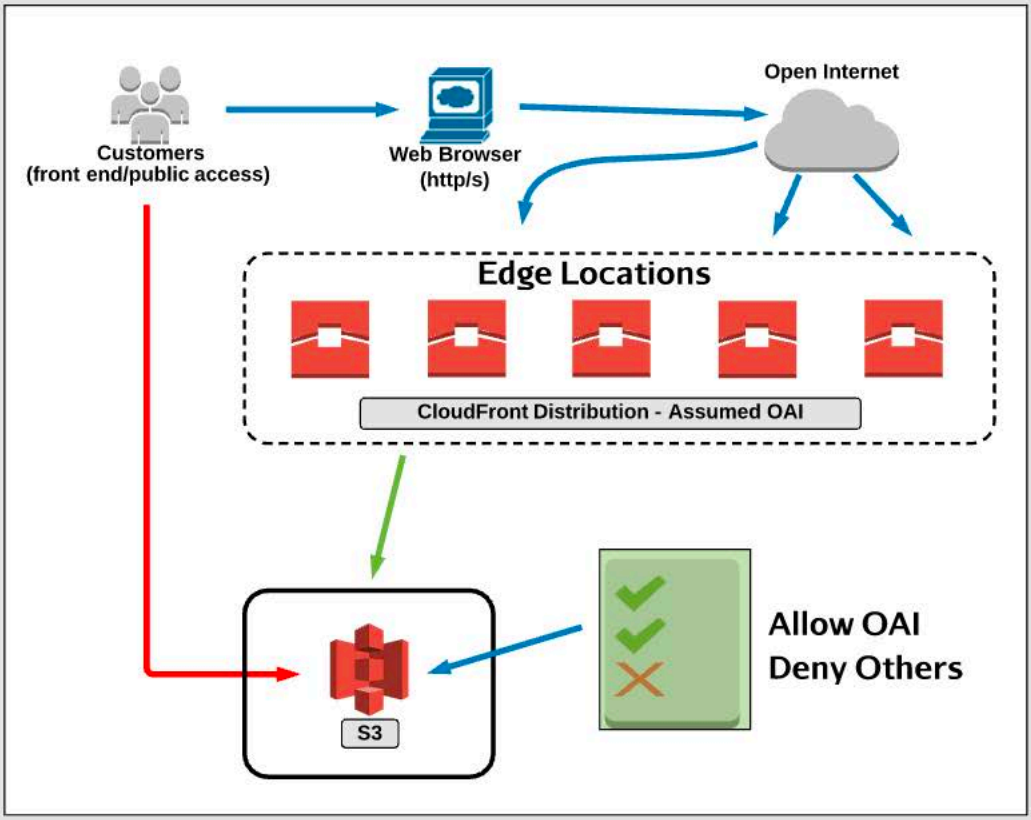
By default, when using CloudFront with S3, CloudFront is optional, and S3 can be accessed directly. This can be changed by creating an Origin Access Identity (**OAI**).

What's an OAI

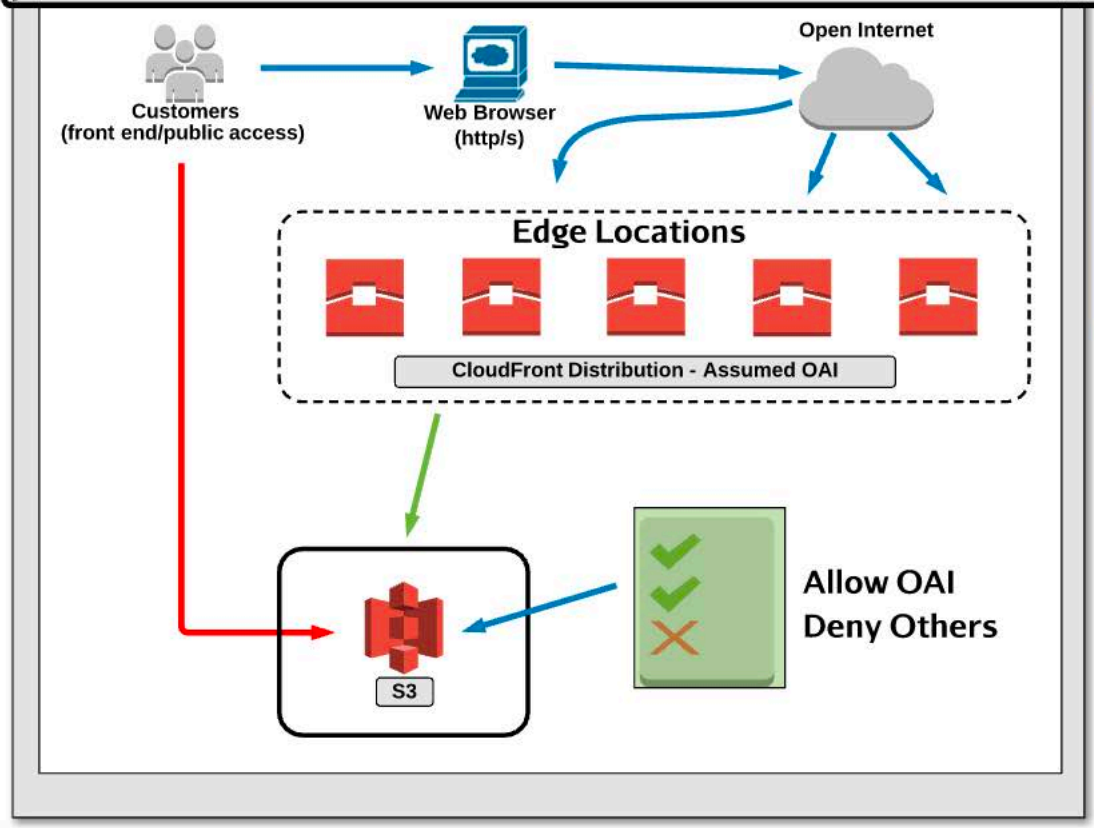
An OAI is a 'virtual' identity. A distribution can be configured to use it, so when accessing S3, CloudFront assumes this identity.

How is an OIA Used, and Why?

To use an OAI, public permissions are removed from your S3 bucket policy and permissions for the OAI are added. Only the CloudFront using that OAI can access your S3 bucket.



```
{
    "Version": "2008-10-17",
    "Id": "PolicyForCloudFrontPrivateContent",
    "Statement": [
        {
            "Sid": "1",
            "Effect": "Allow",
            "Principal": {
                "AWS":
"arn:aws:iam::cloudfront:user/CloudFront Origin Access
Identity XXXXXXXXXXXX"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::mybucket/*"
        }
    ]
}
```



Design edge security on AWS.

Signed URL's and Cookies

Signed URL's allow an entity (generally an application) to create a URL which includes the necessary information to provide the holder of that URL with read/write access to an object, even if they have no permissions on that object.

Cookies extend this, allowing access to an *object type* or *area/folder* and don't need a specifically formatted URL.

Features/Limits

- Signed URL's/Cookies are linked to an existing identify (Role/User), and they have the permissions of that entity.
- They can have their own validity period; the default is 60 minutes.
- They expire either at the end of the period or until the entity on which they are based expires. If you use a role, this is when the roles temp credentials expire.
- Anyone can create a signed URL, even if they don't have permissions on the object.
- With CloudFront you defined the accounts which can sign; the key pair `TrustedSigners` is needed for CloudFront.
- Signed Cookies *don't* work with RTMP distributions.

S3 **CloudFront**

Design edge security on AWS.

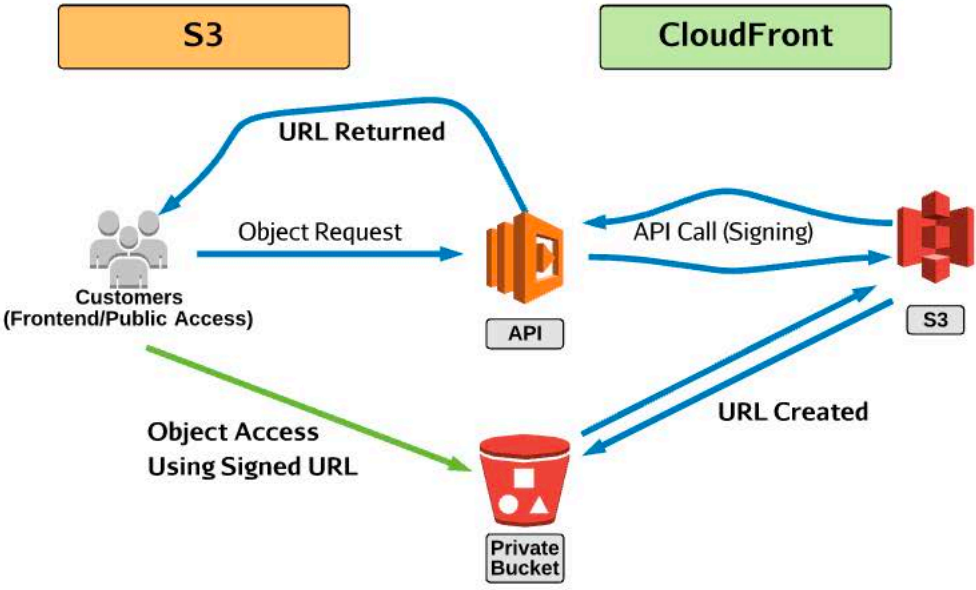
Signed URL's and Cookies

Signed URL's allow an entity (generally an application) to create a URL which includes the necessary information to provide the holder of that URL with read/write access to an object, even if they have no permissions on that object.

Cookies extend this, allowing access to an *object type* or *area/folder* and don't need a specifically formatted URL.

Features/Limits

- Signed URL's/Cookies are linked to an existing identify (Role/User), and they have the permissions of that entity.
- They can have their own validity period; the default is 60 minutes.
- They expire either at the end of the period or until the entity on which they are based expires. If you use a role, this is when the roles temp credentials expire.
- Anyone can create a signed URL, even if they don't have permissions on the object.
- With CloudFront you defined the accounts which can sign; the key pair `TrustedSigners` is needed for CloudFront.
- Signed Cookies *don't* work with RTMP distributions.



Design edge security on AWS.

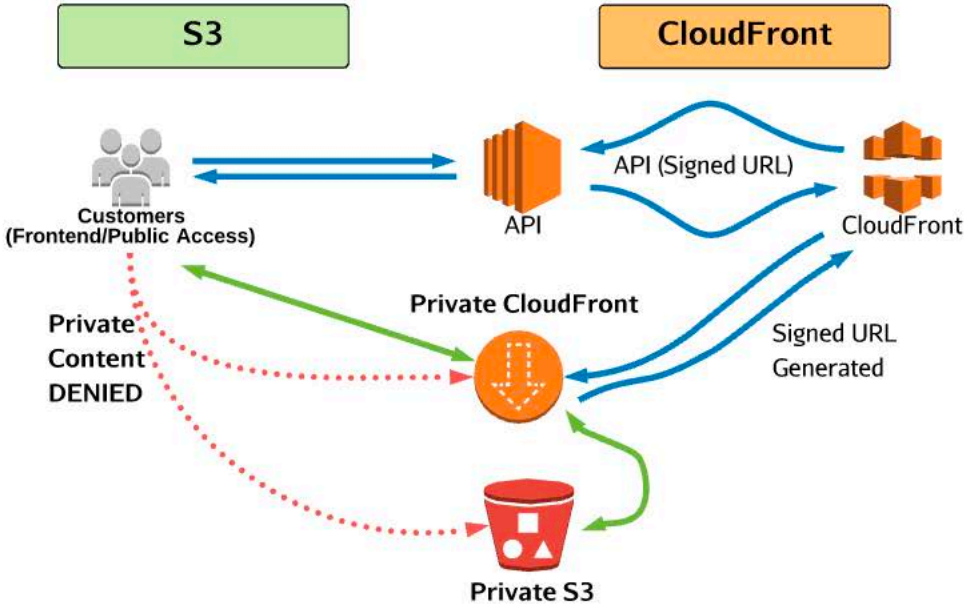
Signed URL's and Cookies

Signed URL's allow an entity (generally an application) to create a URL which includes the necessary information to provide the holder of that URL with read/write access to an object, even if they have no permissions on that object.

Cookies extend this, allowing access to an *object type* or *area/folder* and don't need a specifically formatted URL.

Features/Limits

- Signed URL's/Cookies are linked to an existing identify (Role/User), and they have the permissions of that entity.
- They can have their own validity period; the default is 60 minutes.
- They expire either at the end of the period or until the entity on which they are based expires. If you use a role, this is when the roles temp credentials expire.
- Anyone can create a signed URL, even if they don't have permissions on the object.
- With CloudFront you defined the accounts which can sign; the key pair `TrustedSigners` is needed for CloudFront.
- Signed Cookies *don't* work with RTMP distributions.



Design edge security on AWS.

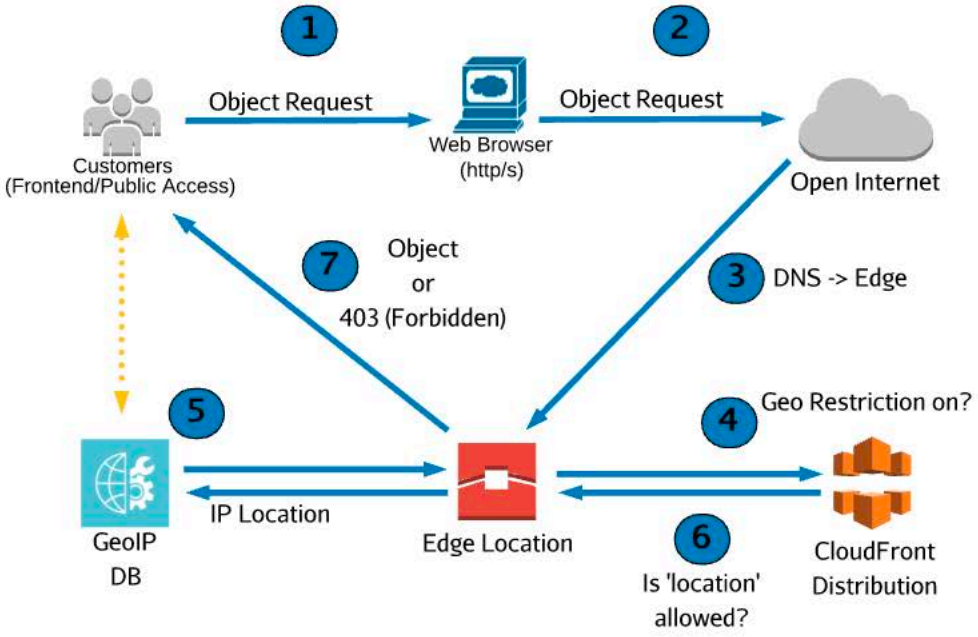
Geo Restriction

CloudFront can restrict content in one of two ways:

- Using CloudFront Geo Restriction
- Using a Third-Party Geolocation Service

CloudFront Geo Restriction | **Third-Party Geolocation**

Cloud Front Geo Restriction is a simple implementation.
Whitelist OR Blacklist and it works on country restrictions ONLY.
Location is based on IP country location - backed by a GeoIP Database (~99.8% accuracy)
No restrictions on ANYTHING ELSE - session/cookie/content/browser etc



Design edge security on AWS.

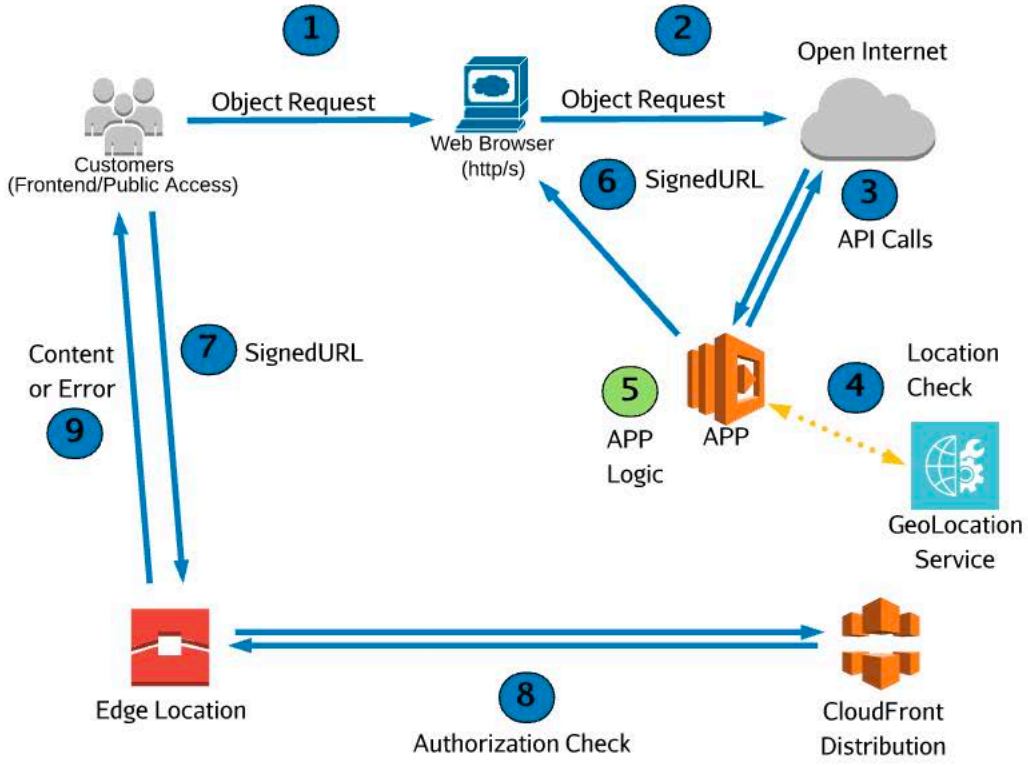
Geo Restriction

CloudFront can restrict content in one of two ways:

- Using CloudFront Geo Restriction
- Using a Third-Party Geolocation Service

CloudFront Geo Restriction | **Third-Party Geolocation**

Third-Party Geo Restriction needs a server/serverless application - SignedURL's are used.
A Third-Party Geolocation service is used ... extra accuracy
Your application can apply additional restriction - session/browser/account level etc...
Location can be MUCH more accurate .. city, locale, Lat/Long in some cases

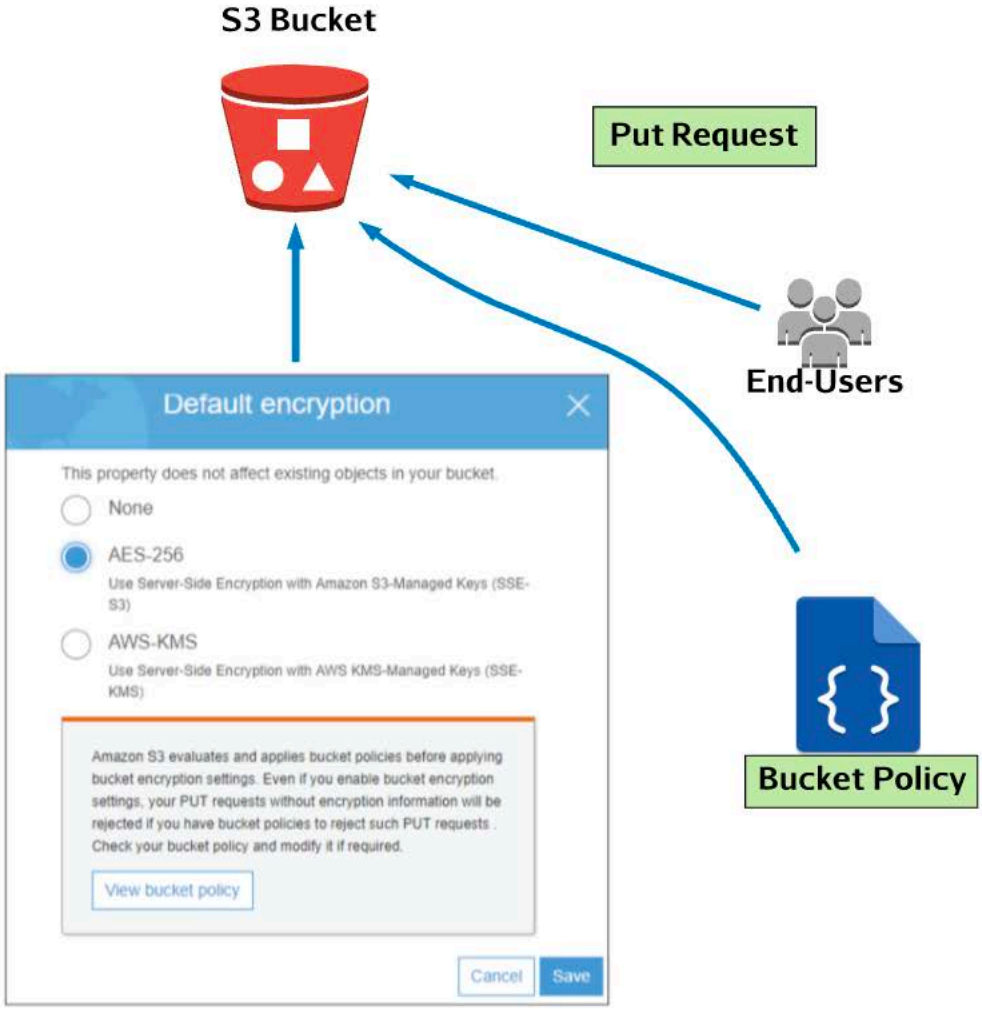


Design edge security on AWS.

Forcing S3 Encryption

S3 doesn't encrypt buckets, objects are encrypted and the settings are defined at an object level. Historically, it wasn't possible to define encryption at a bucket level, but you can now set **S3 Default Encryption** on a bucket level. If set, then any objects put into a bucket without encryption headers are encrypted using the bucket-level default settings.

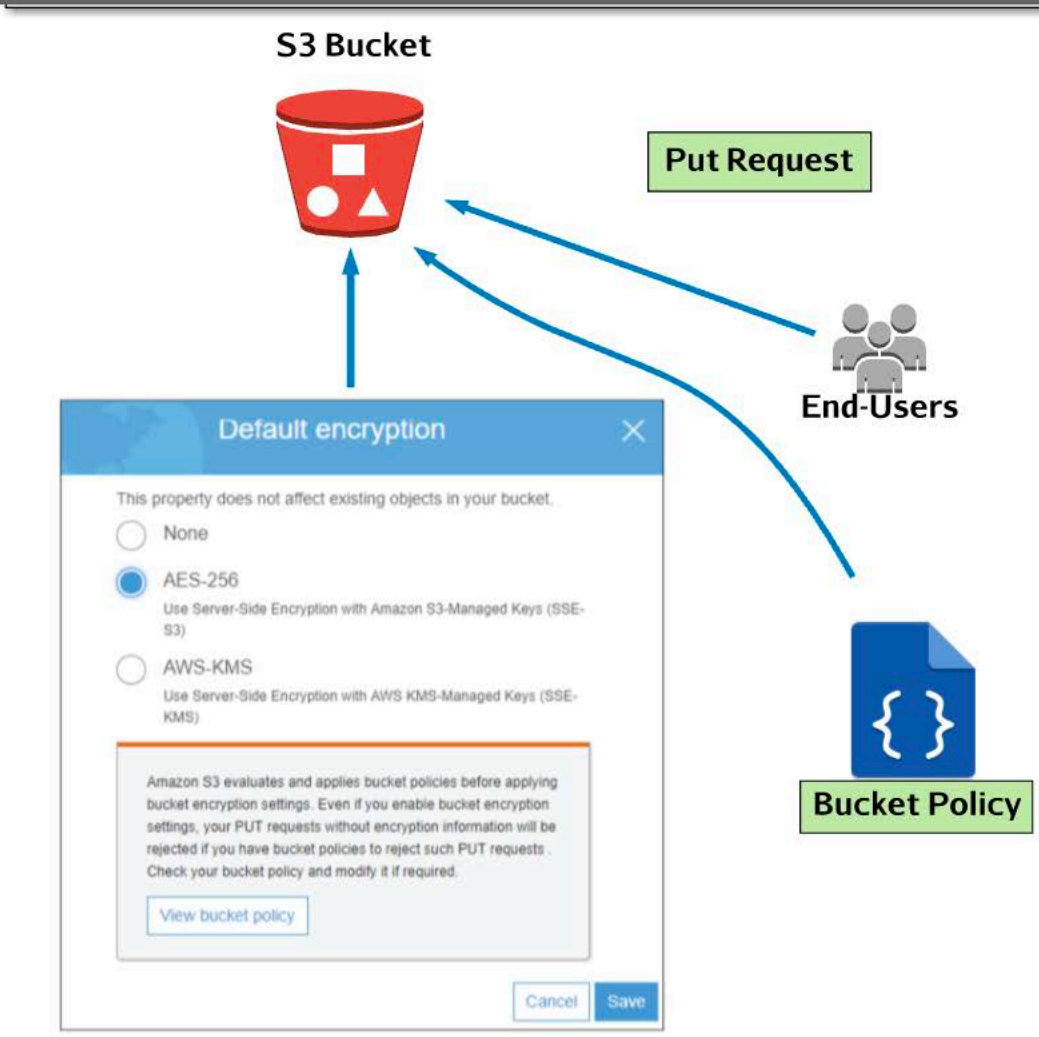
Additionally, bucket policies can be used to deny attempts to put objects into a bucket with individual encryption methods.



SECURITY SPECIALTY RUNBOOK

```
PUT /cat.jpg HTTP/1.1
Host: lacatpics.s3.amazonaws.com
Date: Wed, 15 Aug 2018 11:11:00 GMT
Authorization: authorization string
Content-Type: text/plain
Content-Length: 13337
x-amz-meta-author: Anthony
Expect: 100-continue
[11434 bytes of object data]
```

```
PUT /cat.jpg HTTP/1.1
Host: lacatpics.s3.amazonaws.com
Date: Wed, 15 Aug 2018 11:11:00 GMT
Authorization: authorization string
Content-Type: text/plain
Content-Length: 13337
x-amz-meta-author: Anthony
Expect: 100-continue
x-amz-server-side-encryption: AES256
[11434 bytes of object data]
```



LOGGING & MONITORING

INFRASTRUCTURE SECURITY

IDENTITY & ACCESS MGNT

DATA PROTECTION

Design edge security on AWS.

Forcing S3 Encryption

S3 doesn't encrypt buckets, objects are encrypted and the settings are defined at an object level. Historically, it wasn't possible to define encryption at a bucket level, but you can now set **S3 Default Encryption** on a bucket level. If set, then any objects put into a bucket without encryption headers are encrypted using the bucket-level default settings.

Additionally, bucket policies can be used to deny attempts to put objects into a bucket with individual encryption methods.


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

Design edge security on AWS.

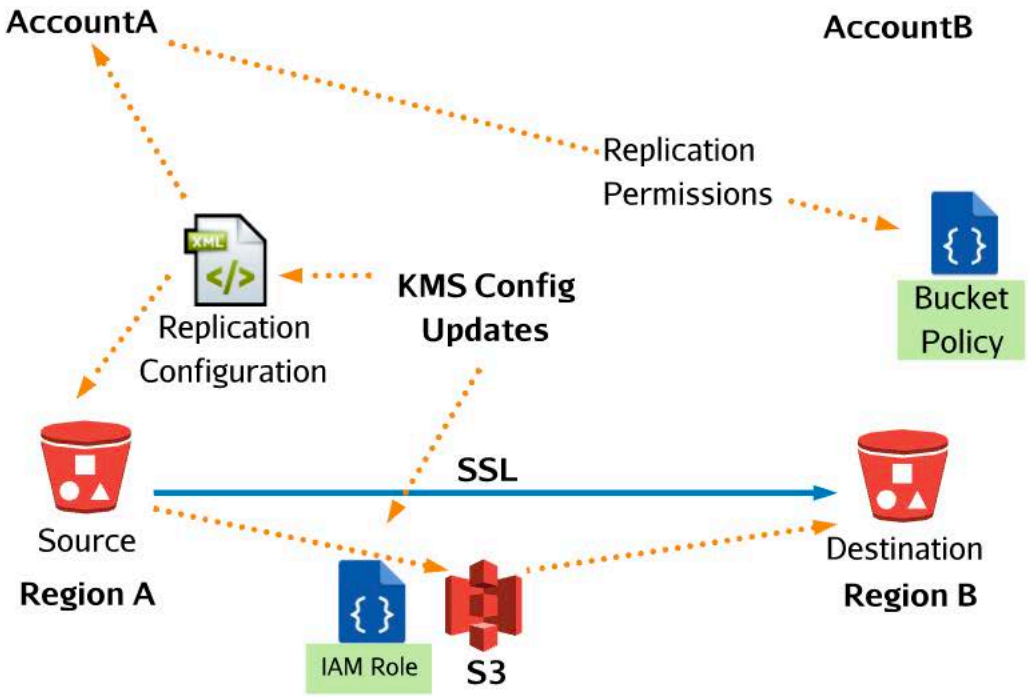
S3 : Cross-Region Replication (CRR) Security

CRR is configured on a bucket level and provides an asynchronous replication of objects from one source bucket to one destination bucket located in different AWS regions.

Key Features and Limits

- No retroactive replication only replicates objects created after enabling CRR.
- CRR replicates un-encrypted and SSE-S3 encrypted objects by default.
- SSE-C are not supported. SSE-KMS is supported but needs extra configuration.
- By default, ownership and ACL's are replicated and maintained, but CRR can adjust these!
- The storage class is maintained by default
- Only Customer actions are replicated (human or app); Lifecycle events are not replicated.
- When the bucket owner has no permissions, objects are not replicated.

Standard | **Other Acc** | **Owner Change** | **KMS**

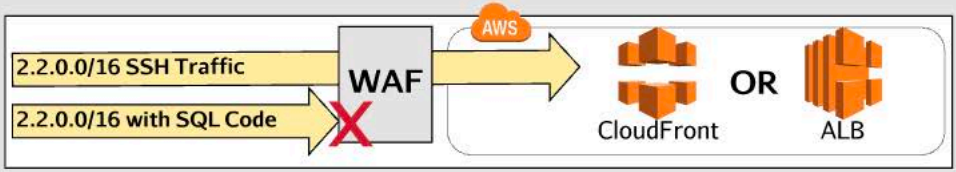


Design edge security on AWS.

Protecting Web Applications

AWS Web Application Firewall (WAF)

- Allows for conditions or rules to be set on CloudFront web traffic or an Application Load Balancer.
- WAF can watch for cross-site scripting, IP addresses, the location of requests, query strings, and SQL injection.
- When multiple conditions exist in a rule, the result must include all conditions:
  - **Example Rule:** Block requests from 2.2.0.0/16 that appear to have an SQL code.
  - Both conditions must match for a block.



Denial of service attacks

- Flooding a system with traffic to overwhelm and prevent legitimate traffic access to resources.
- Distributed DoS (DDoS) is that same attack from multiple sources or systems.
- AWS provides resilience for network and transport layer attacks using **AWS Shield**.

AWS Shield Standard

- The basic level of DDoS protection for your web applications.
- Included with WAF with no additional cost.

AWS Shield Advanced

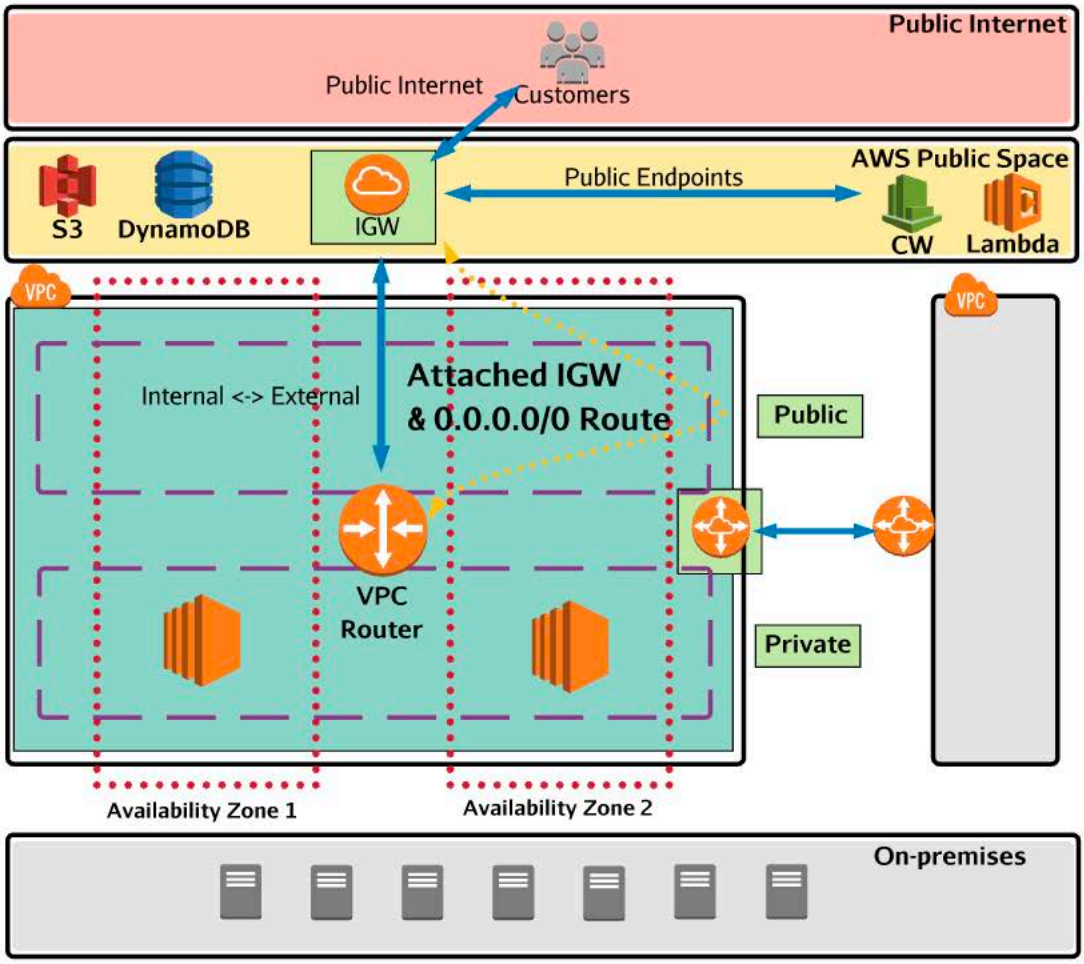
- Expands services protected to include Elastic Load Balancers, Cloudfront Distributions, Route 53 hosted zones, and resources with Elastic IPs.
- Some of the advantages:
  - Contact 24x7 DDoS response Team (DRT) for assistance during an attack.
  - Some cost protection against spikes in a bill from DDoS attacks.
  - Expanded protection against many types of attacks.
- WAF is included in Shield Advanced pricing:
  - $3,000/month per organization.
  - Plus Data Transfer Out usage fees.

Design and implement a secure network infrastructure.

VPC Design and Security

VPC's are the base 'private' security and networking construct in AWS. They provide private networking and security features to create isolated networks and security domains and to connect them. Being able to design, secure, and troubleshoot these private environments is essential for real-world usage and the security exam.

VPC's provide a separate security domain and limit the infrastructure layer blast radius. They do *not* provide any restriction for account level exploits.

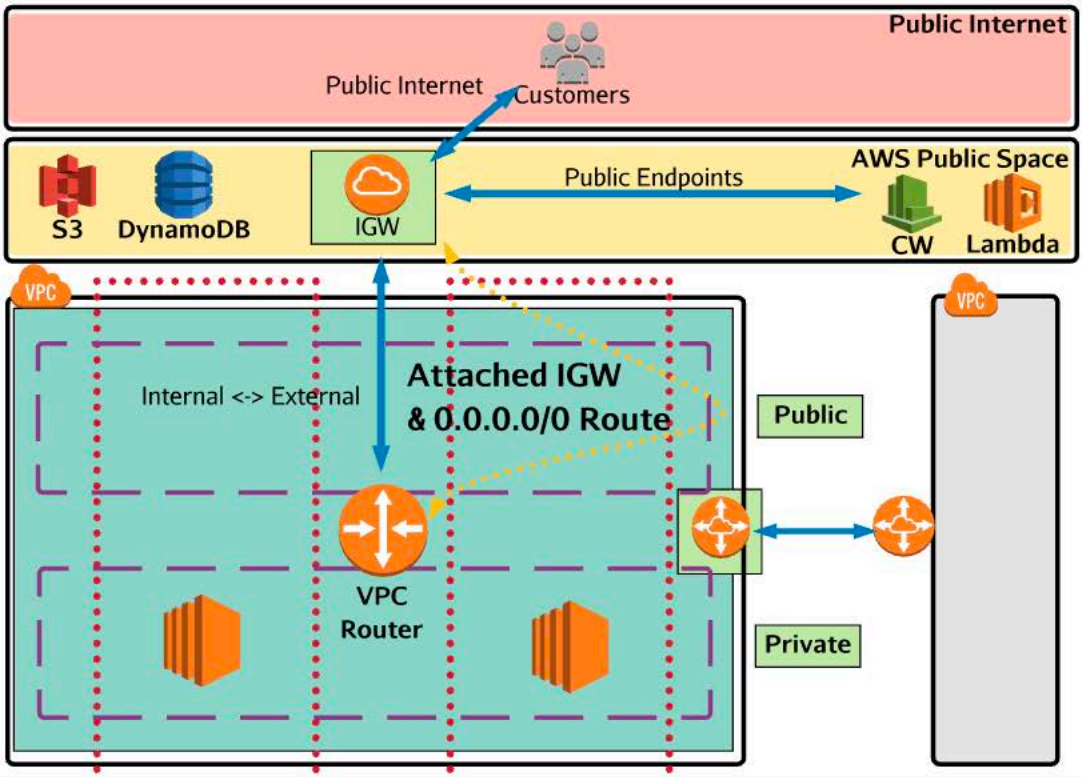


Design and implement a secure network infrastructure.

VPC Design and Security

VPC's are the base 'private' security and networking construct in AWS. They provide private networking and security features to create isolated networks and security domains and to connect them. Being able to design, secure, and troubleshoot these private environments is essential for real-world usage and the security exam.

VPC's provide a separate security domain and limit the infrastructure layer blast radius. They do *not* provide any restriction for account level exploits.

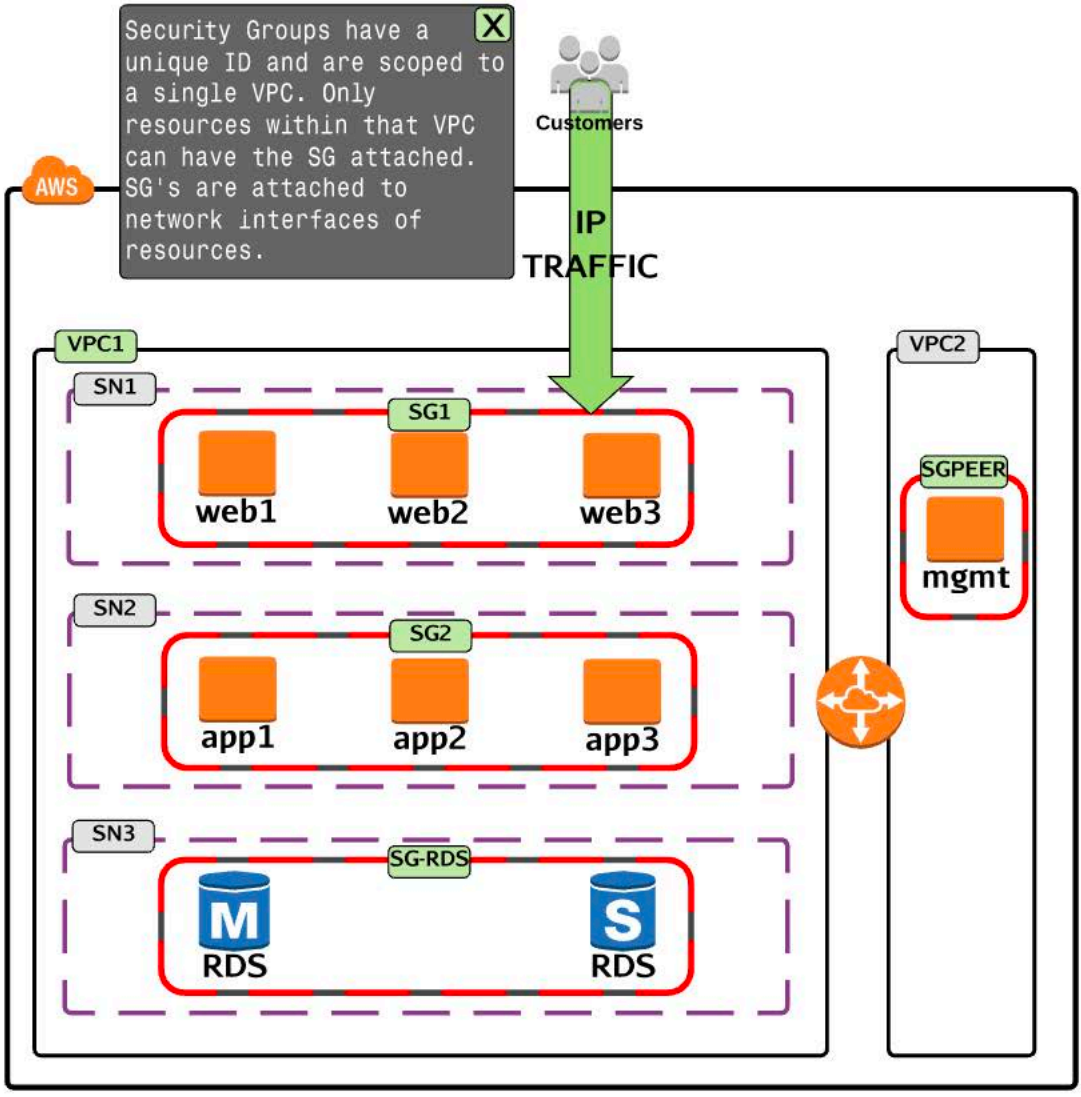


A Private subnet is the default Configuration for a subnet. There is no security exposure because it's entirely private and isolated from any public AWS and public internet connectivity.

Design and implement a secure network infrastructure.

Security Groups

Security groups (SGs) are security filters/security perimeters which operate within a specific VPC. The are applied to network interfaces of EC2 instances or other products which can function within a VPC. Examples include ELB's/RDS instances and Lambda functions.

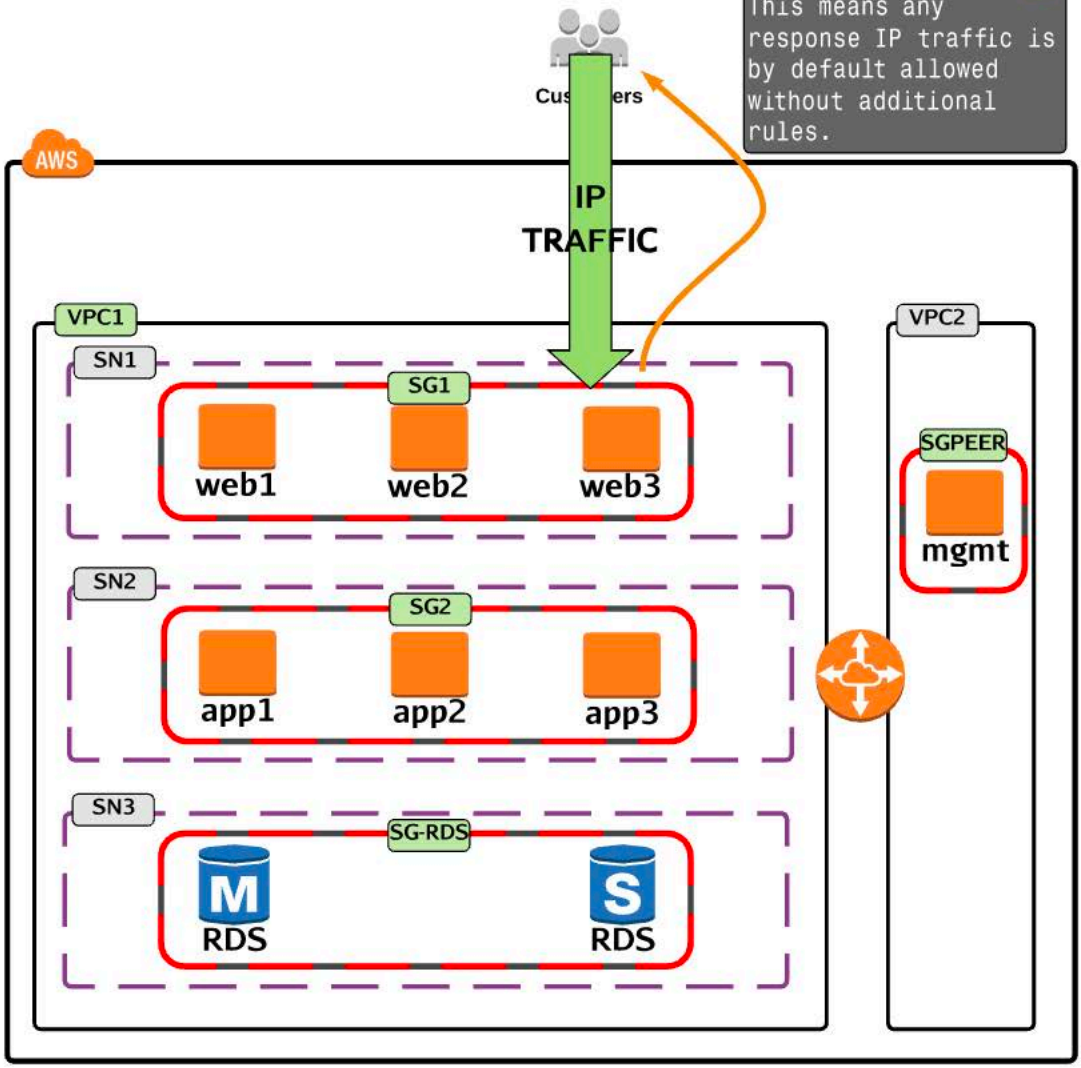


Design and implement a secure network infrastructure.

Security Groups

Security groups (SGs) are security filters/security perimeters which operate within a specific VPC. The are applied to network interfaces of EC2 instances or other products which can function within a VPC. Examples include ELB's/RDS instances and Lambda functions.

SG's are stateful. This means any response IP traffic is by default allowed without additional rules.



Design and implement a secure network infrastructure.

Security Groups

Security groups (SGs) are security filters/security perimeters which operate within a specific VPC. The are applied to network interfaces of EC2 instances or other products which can function within a VPC. Examples include ELB's/RDS instances and Lambda functions.

A Security Group (SG) is applied to one or more network interfaces. Before traffic is allowed to ingress or egress from that interface, all SG rules are evaluated. An SG can be attached to multiple resources, and a resource can have multiple SGs. Remember, it's attached to the network interface, *not* the resource. For some products like EC2, this means you might have some SGs only apply to one ENI.

Type	Protocol	Port Range	Source
SSH (22)	TCP (6)	22	0.0.0.0/0
HTTP (80)	TCP (6)	80	0.0.0.0/0

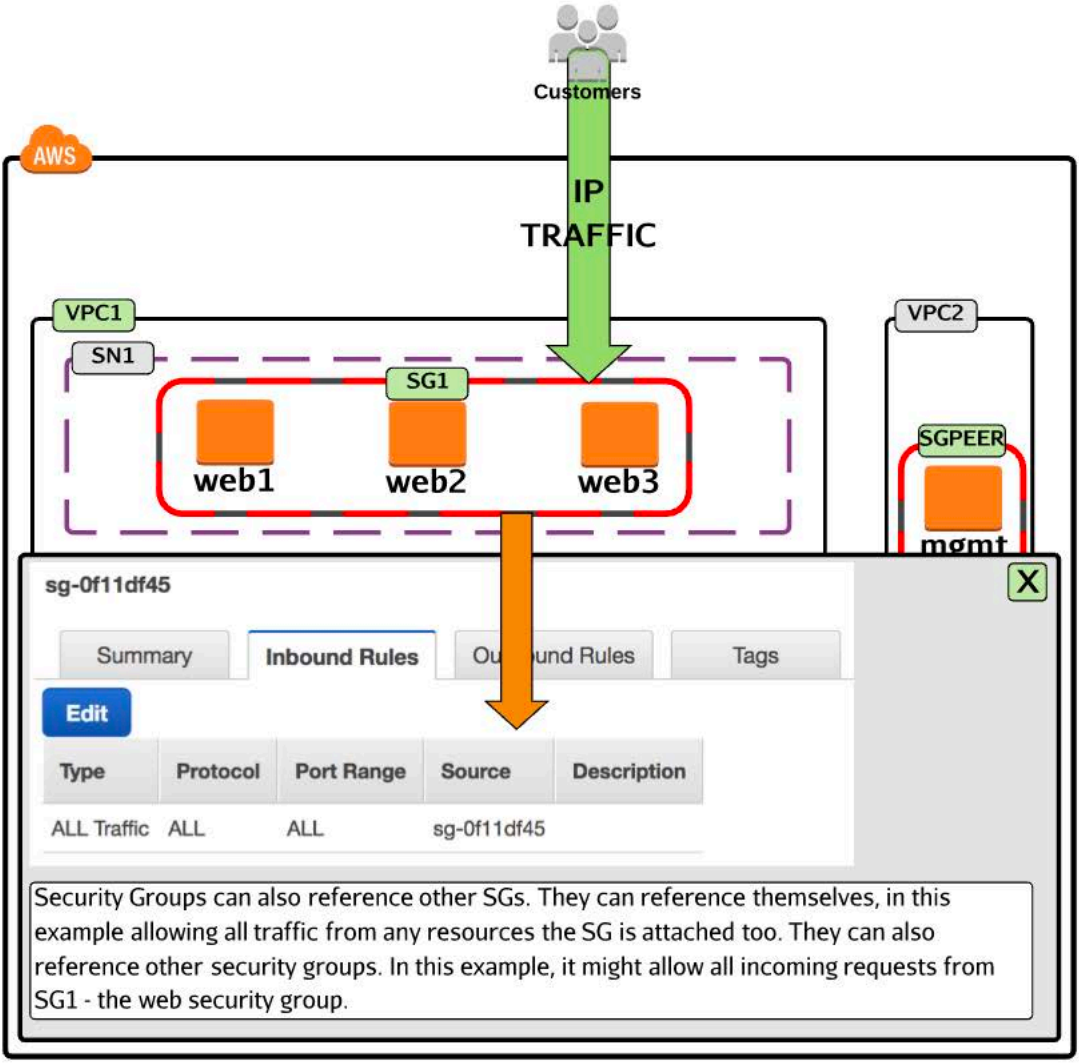
You can select specific traffic types and/or protocols or choose custom and add a custom port range. **Remember, you don't need to explicitly allow response traffic; SGs are stateful.** Source or destination can be an IP, a CIDR or an AWS resource.

There is always an implicit `DENY` rule. If no other rule allows traffic through the security group, it is implicitly denied. Security groups cannot explicitly `DENY` traffic, only `ALLOW`.

Design and implement a secure network infrastructure.

Security Groups

Security groups (SGs) are security filters/security perimeters which operate within a specific VPC. The are applied to network interfaces of EC2 instances or other products which can function within a VPC. Examples include ELB's/RDS instances and Lambda functions.

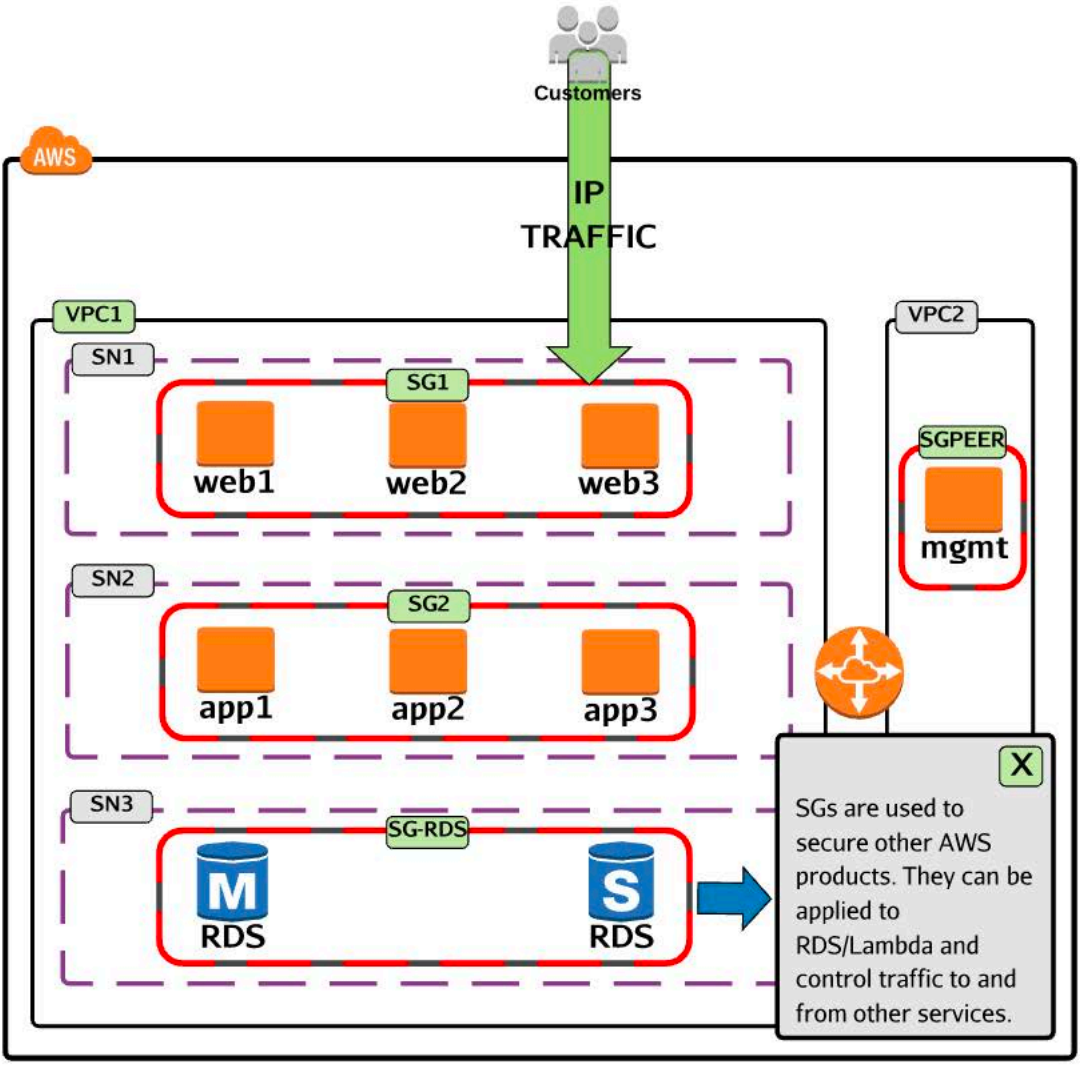


Security Groups can also reference other SGs. They can reference themselves, in this example allowing all traffic from any resources the SG is attached too. They can also reference other security groups. In this example, it might allow all incoming requests from SG1 - the web security group.

Design and implement a secure network infrastructure.

Security Groups

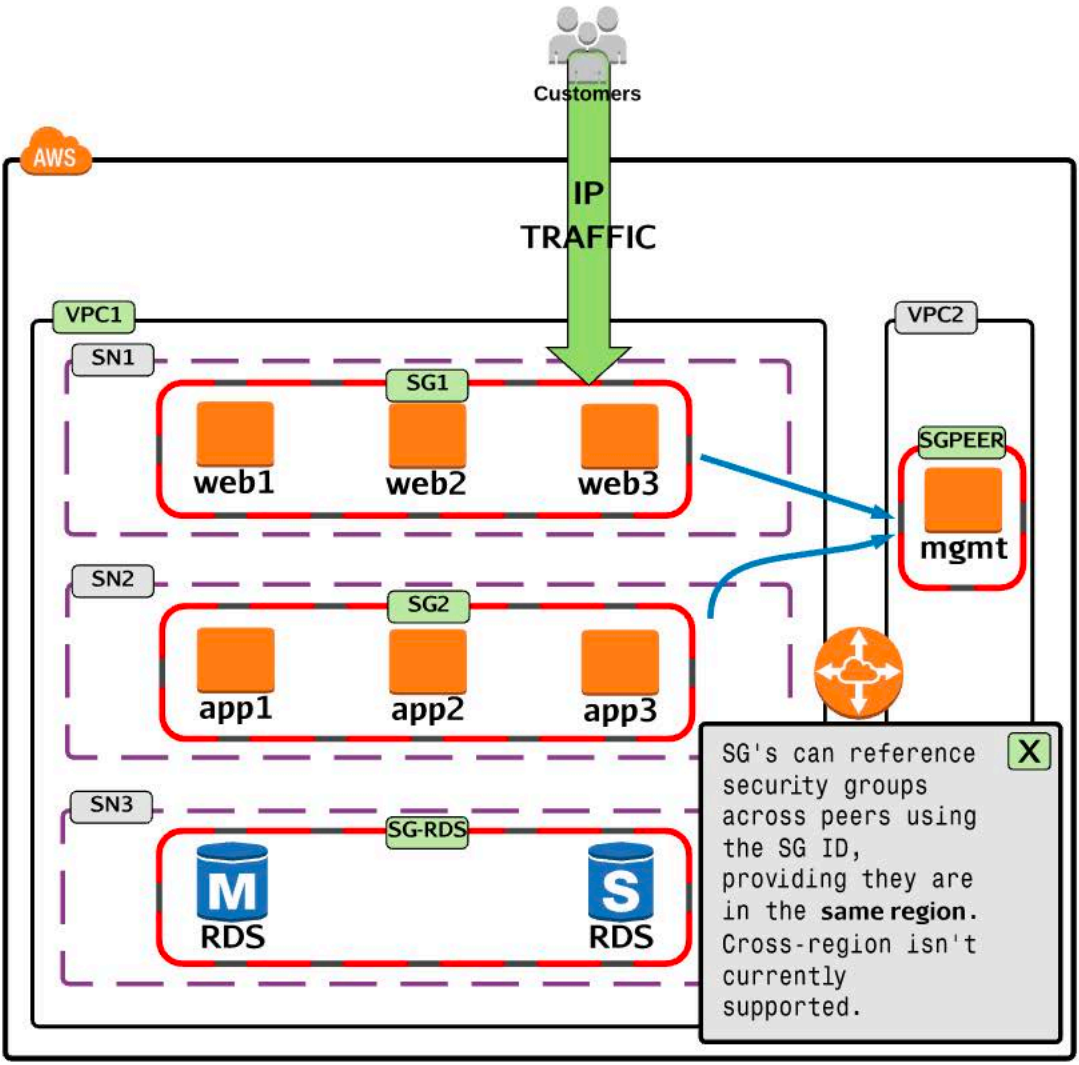
Security groups (SGs) are security filters/security perimeters which operate within a specific VPC. The are applied to network interfaces of EC2 instances or other products which can function within a VPC. Examples include ELB's/RDS instances and Lambda functions.



Design and implement a secure network infrastructure.

Security Groups

Security groups (SGs) are security filters/security perimeters which operate within a specific VPC. The are applied to network interfaces of EC2 instances or other products which can function within a VPC. Examples include ELB's/RDS instances and Lambda functions.



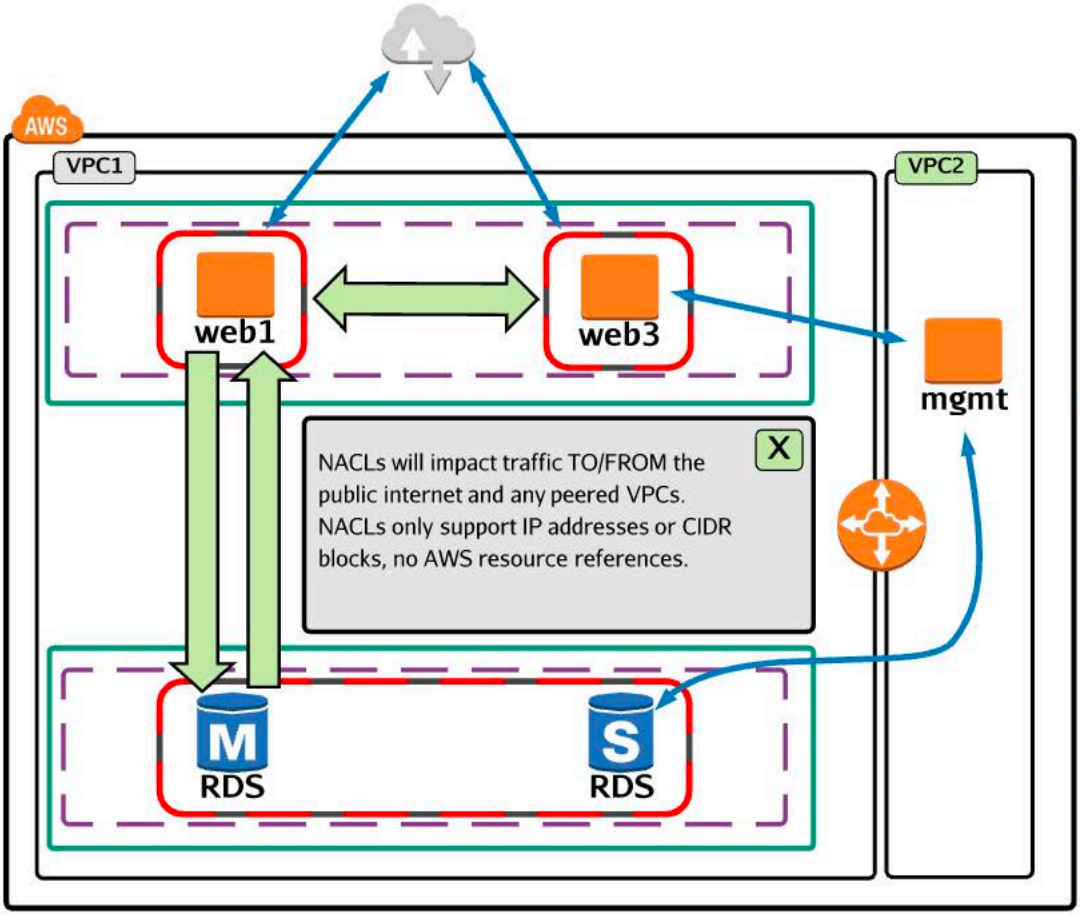
Design and implement a secure network infrastructure.

NACL

Network Access Control Lists (NACLs) can be associated with 0 or more subnets. A VPC has a default NACL which is associated with any subnets that don't have a nexplicit alternate association. A Subnet can only have *one* NACL associated with it.

Key Facts and Limitations

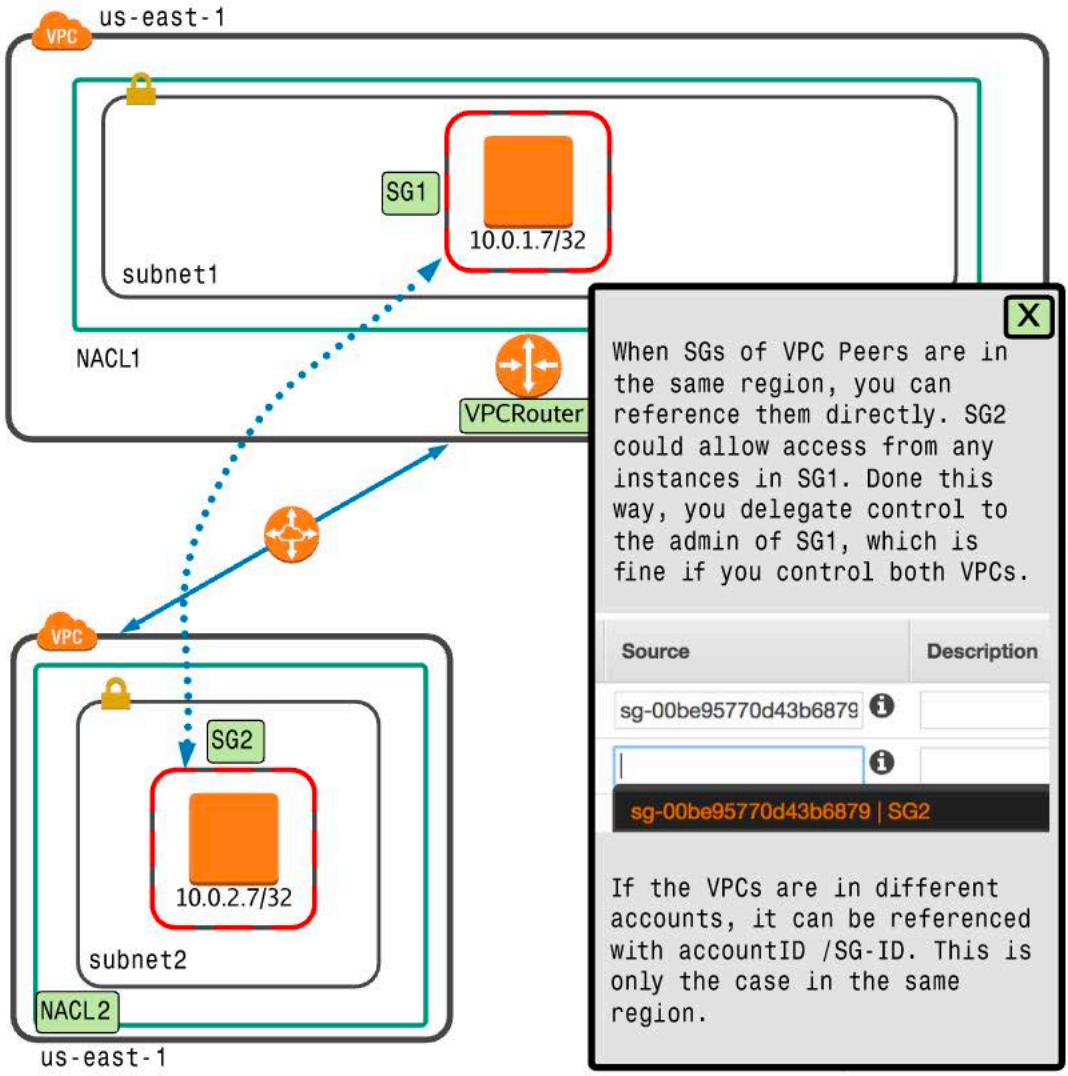
- NACLs are *stateless* - you need rules for *primary* traffic *and* any *response*
- NACLs are processed in order, lowest rule number first and have a default `DENY`.
- You CAN add explicit ALLOWS and DENY's, including an explicit default `ALLOW`
- NACLs are processes only when data enters or leaves a subnet, before Security Groups
- NACLs work on IP and CIDR only. You *can't* reference AWS services (ELB, NAT, SGs, etc.).



Design and implement a secure network infrastructure.

VPC Peering

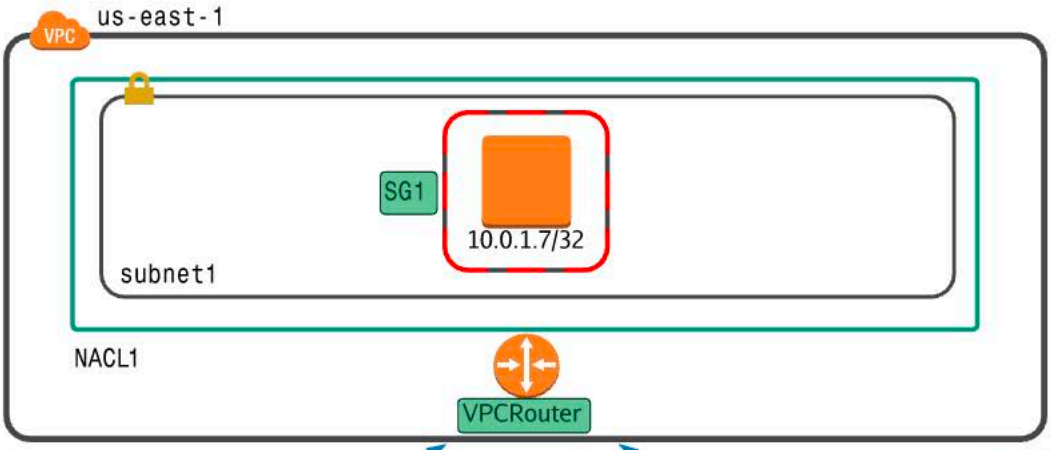
A VPC peer is a secure routed connection between two VPCs; these VPCs can be in the same account or different accounts. Peers can operate within the same AWS region, or across regions, known as an inter-region peer, with reduced functionality. A VPC peer opens your otherwise private and isolated VPC network to external access and potential, exploit so its essential to secure it correctly.



Design and implement a secure network infrastructure.

VPC Peering

A VPC peer is a secure routed connection between two VPCs; these VPCs can be in the same account or different accounts. Peers can operate within the same AWS region, or across regions, known as an inter-region peer, with reduced functionality. A VPC peer opens your otherwise private and isolated VPC network to external access and potential, exploit so its essential to secure it correctly.



Routes can be utilized as a security feature - only CIDRs or IPs (/32) explicitly defined in route tables at both sides will be able to communicate regardless of any Security groups of Network ACLs

If you don't 'own' both accounts then you should aim to be as granular as possible .. /32's

If both accounts are under your control, then full VPC CIDRs are suitable.

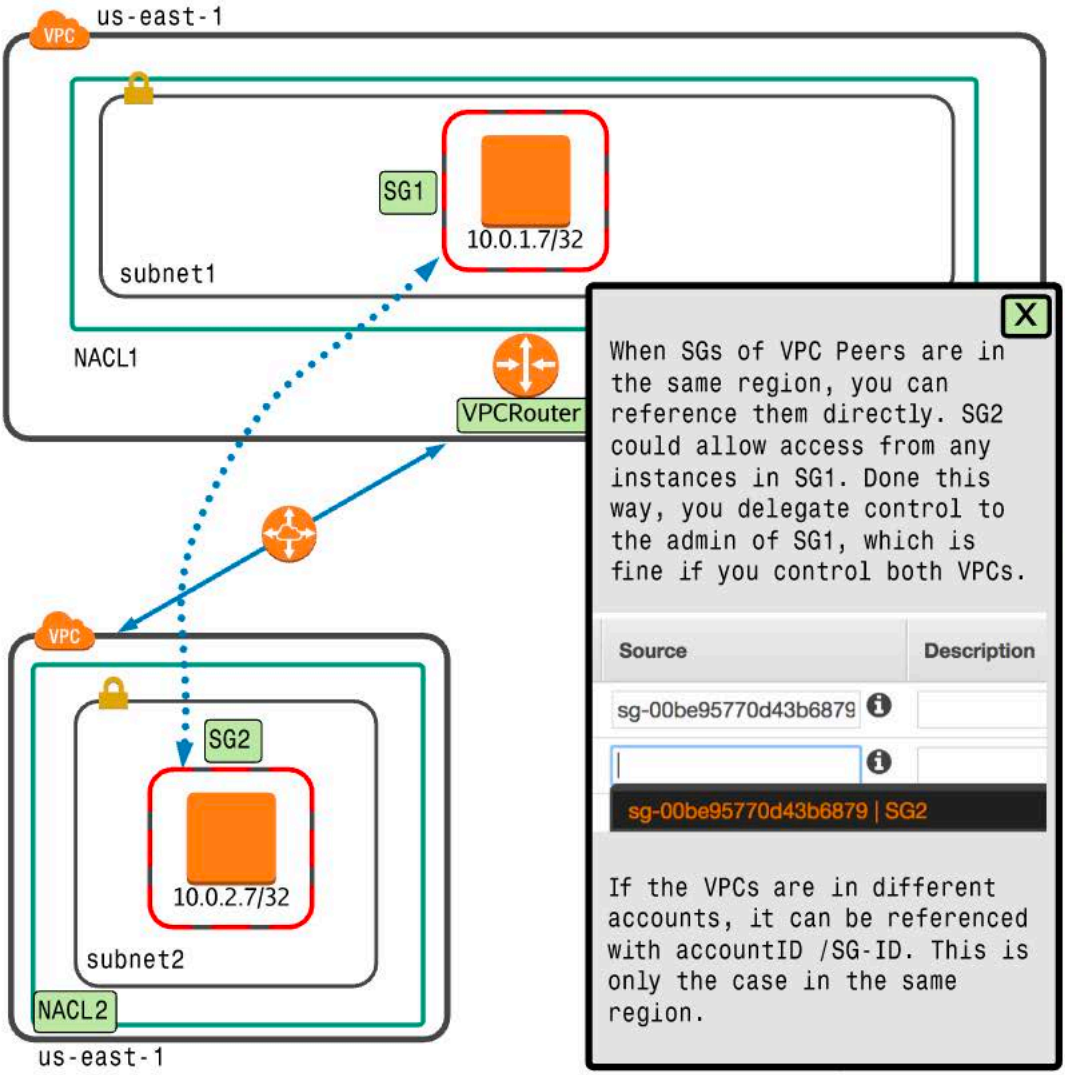
Destination	Target
10.0.0.0/24	local
2600:1f18:24b4:d300::/56	local
10.0.1.0/24	pcx-0728abff2cf0b6647

Destination	Target
10.0.0.0/24	local
2600:1f18:24b4:d300::/56	local
10.0.1.7/32	pcx-0728abff2cf0b6647

Design and implement a secure network infrastructure.

VPC Peering

A VPC peer is a secure routed connection between two VPCs; these VPCs can be in the same account or different accounts. Peers can operate within the same AWS region, or across regions, known as an inter-region peer, with reduced functionality. A VPC peer opens your otherwise private and isolated VPC network to external access and potential, exploit so its essential to secure it correctly.



X

When SGs of VPC Peers are in the same region, you can reference them directly. SG2 could allow access from any instances in SG1. Done this way, you delegate control to the admin of SG1, which is fine if you control both VPCs.

Source	Description
sg-00be95770d43b6879	

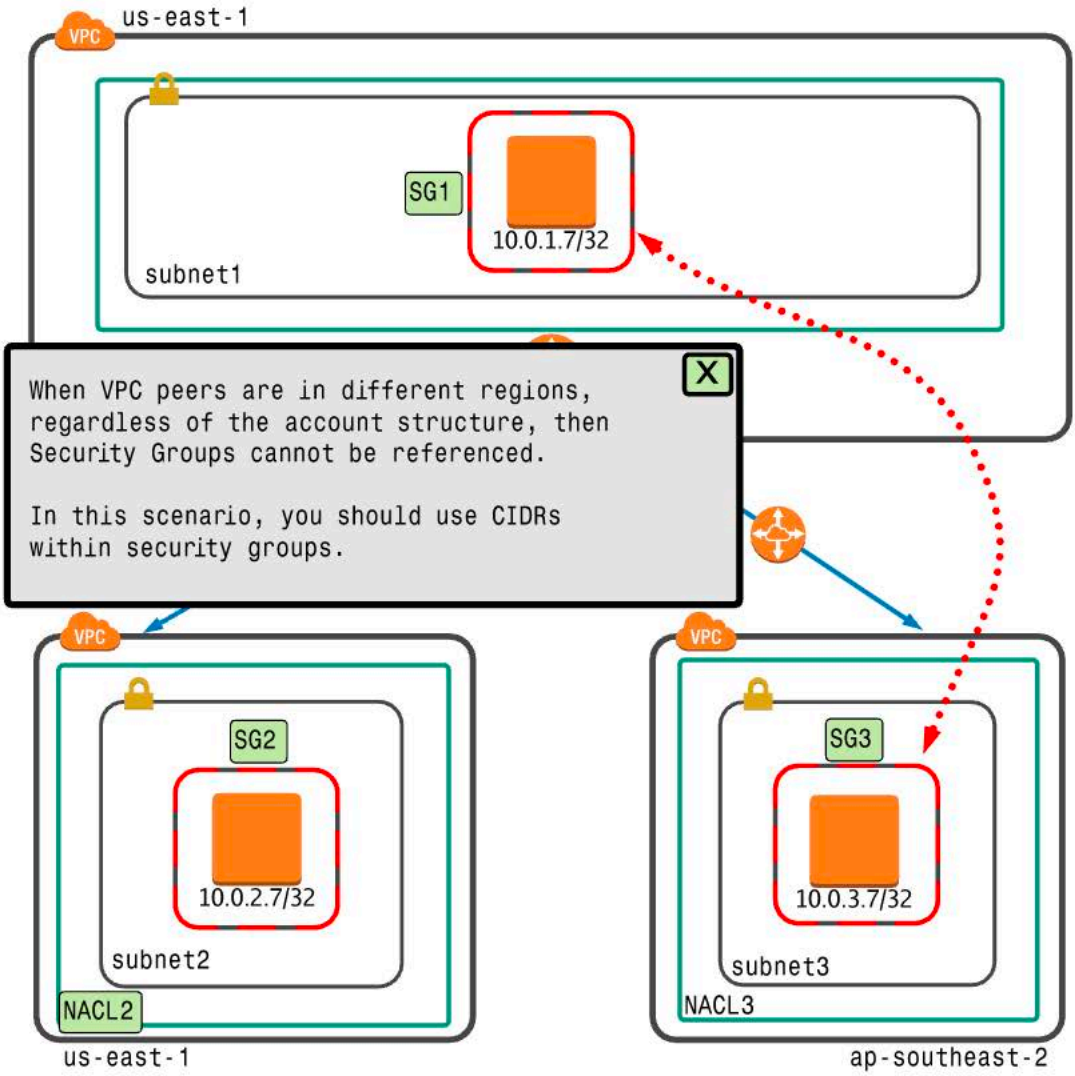
sg-00be95770d43b6879 | SG2

If the VPCs are in different accounts, it can be referenced with accountID /SG-ID. This is only the case in the same region.

Design and implement a secure network infrastructure.

VPC Peering

A VPC peer is a secure routed connection between two VPCs; these VPCs can be in the same account or different accounts. Peers can operate within the same AWS region, or across regions, known as an inter-region peer, with reduced functionality. A VPC peer opens your otherwise private and isolated VPC network to external access and potential, exploit so its essential to secure it correctly.

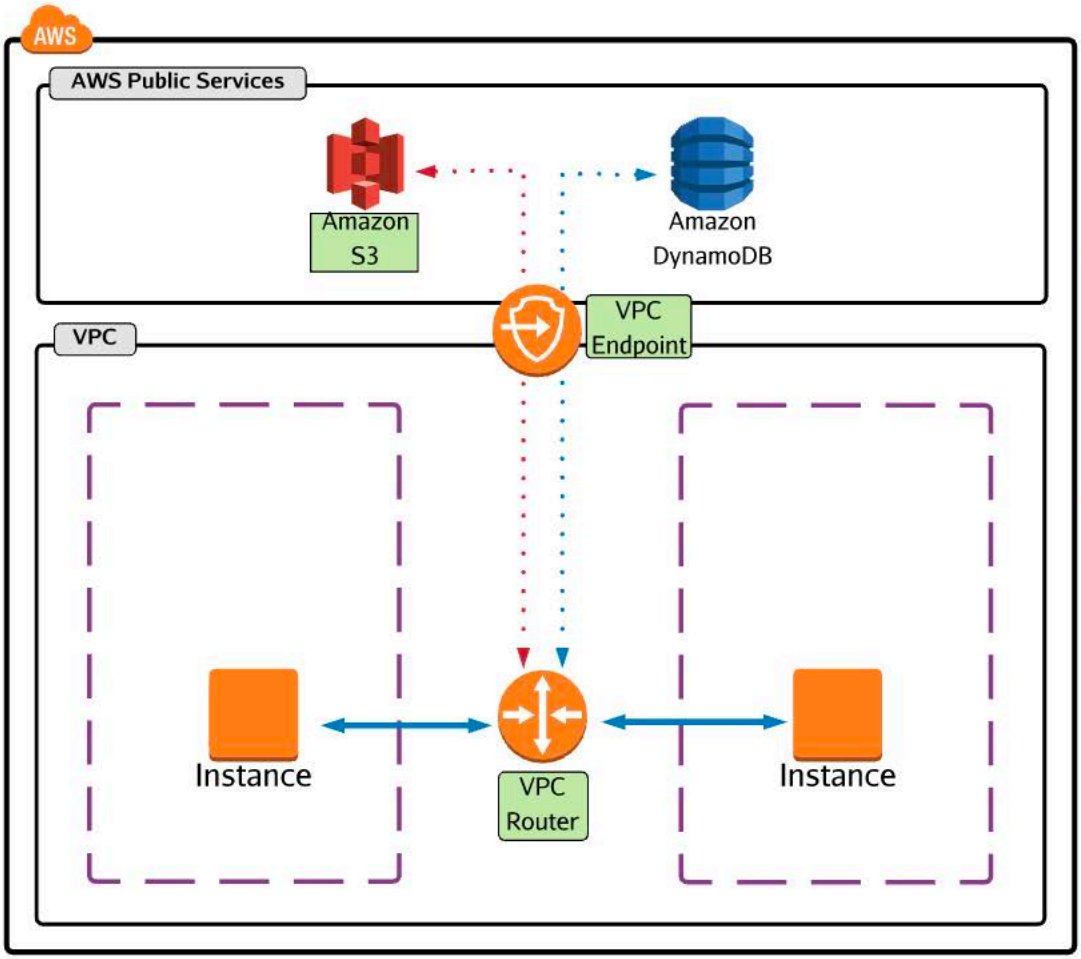


Design and implement a secure network infrastructure.

VPC Endpoints

VPC endpoints allow access to public AWS services, or services provided by 3rd parties, without requiring an Internet Gateway to be attached to the VPC, or any NAT instance/gateway. VPC endpoints are delivered in one of two forms, *interface endpoints* or *gateway endpoints*.

Gateway Endpoints | **Interface Endpoints**

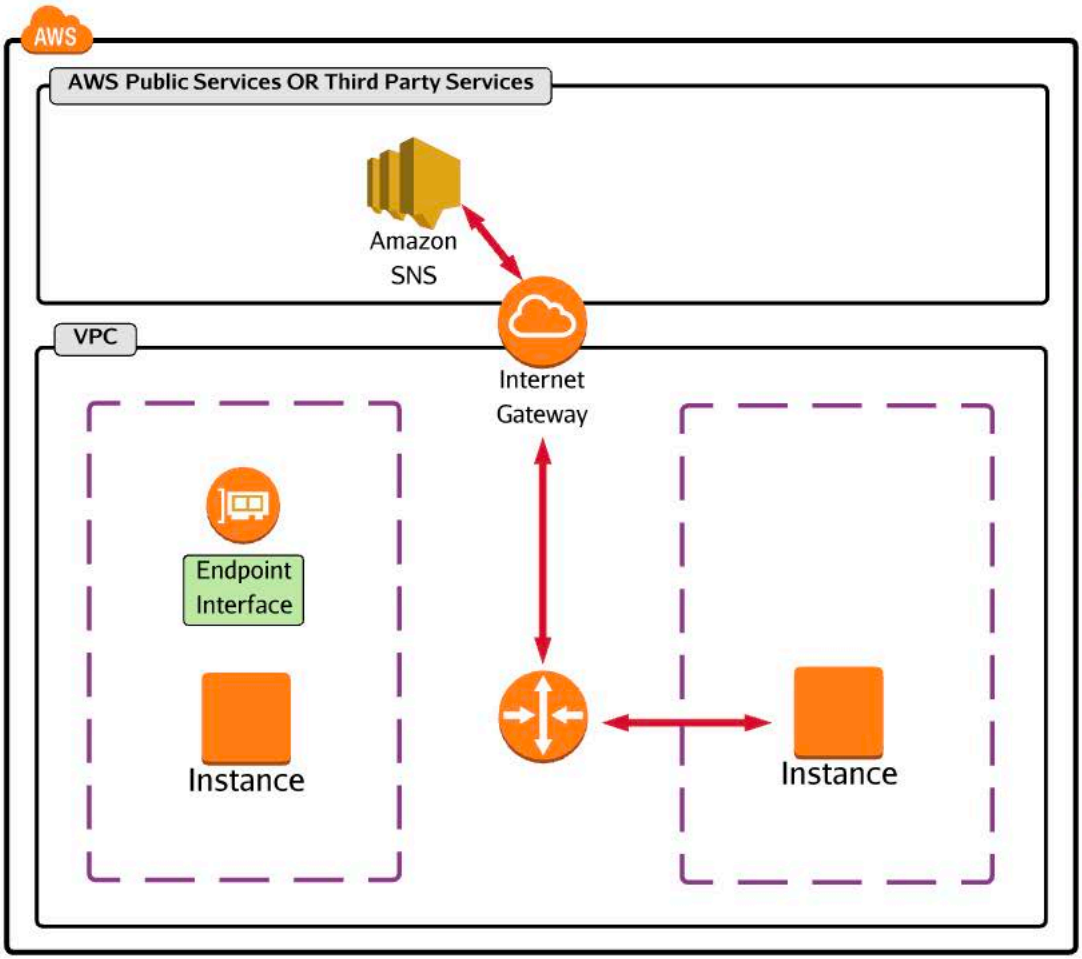


Design and implement a secure network infrastructure.

VPC Endpoints

VPC endpoints allow access to public AWS services, or services provided by 3rd parties, without requiring an Internet Gateway to be attached to the VPC, or any NAT instance/gateway. VPC endpoints are delivered in one of two forms, *interface endpoints* or *gateway endpoints*.

Gateway Endpoints | **Interface Endpoints**



Design and implement a secure network infrastructure.

Serverless Security

Applying appropriate security in a lambda based serverless environment or product set presents its own unique set of challenges. Lambda functions run in a temporary runtime environment, which is used for a limited number of function executions.

Lambda Function Policy

Every lambda function has a policy which controls who or what can invoke it. Any event sources which 'invoke' functions, i.e., S3 or CloudWatchEvents will need permissions to invoke via the function policy:

- For 'poll-based' services (Kinesis, DynamoDB) or SQS - lambda polls on your behalf, and so permissions are granted via its execution role.
- For anything else, or for external entities or accounts, the PUSH model is used.
- Changes to the function policy will be required. Any services that PUSH will need to be allowed to PUSH on a function by function basis.
- This is (under normal circumstances) handled by AWS when you define an event-source.
- PUSH services generally deliver the event to Lambda, so extra permissions aren't always needed.

IAM Execution Role

The IAM execution role is the role assigned to the function when it executes. Much like an EC2 role, it provides the runtime environment and function with the ability to interact with other AWS products via temporary credentials managed by STS:

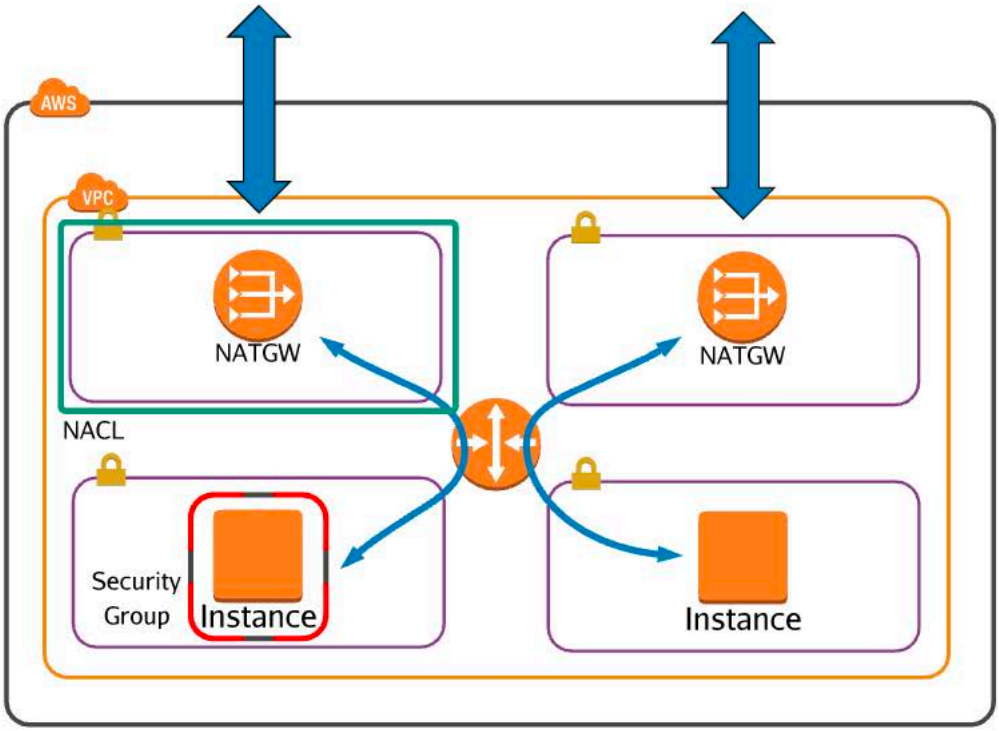
- Ensure it has enough permissions to log to CloudWatch Logs.
- To access any resources it needs to PULL from or PUSH too.
- For *event-driven* invocation, the execution role doesn't need permissions to access it.
- For *poll based* sources such as DynamoDB, SQS ,and Kinesis, it does.

Design and implement a secure network infrastructure.

Nat Gateways

NAT Gateways are a new 'managed' Network Address Translation (NAT) service from AWS. They replace or complement the older NAT Instance, which is a pre-configured EC2 instance performing the same function. There are some security implications of using a NAT gateway that you need to be aware of:

- NAT Gateways are located in a single subnet and utilize elastic IPs.
- They provide internet access for EC2 and other services within private subnets.
- NAT Instances (old) could be secured with NACL (on the private or NAT subnet) and security groups on the NAT instance and source service.
- NAT Gateways *cannot* have security groups associated with them.
- NAT Gateways operate in subnets which can have NACLs, and the source subnets can have NACLs.
- The source service (e.g., EC2) can also in many cases have an attached SG.

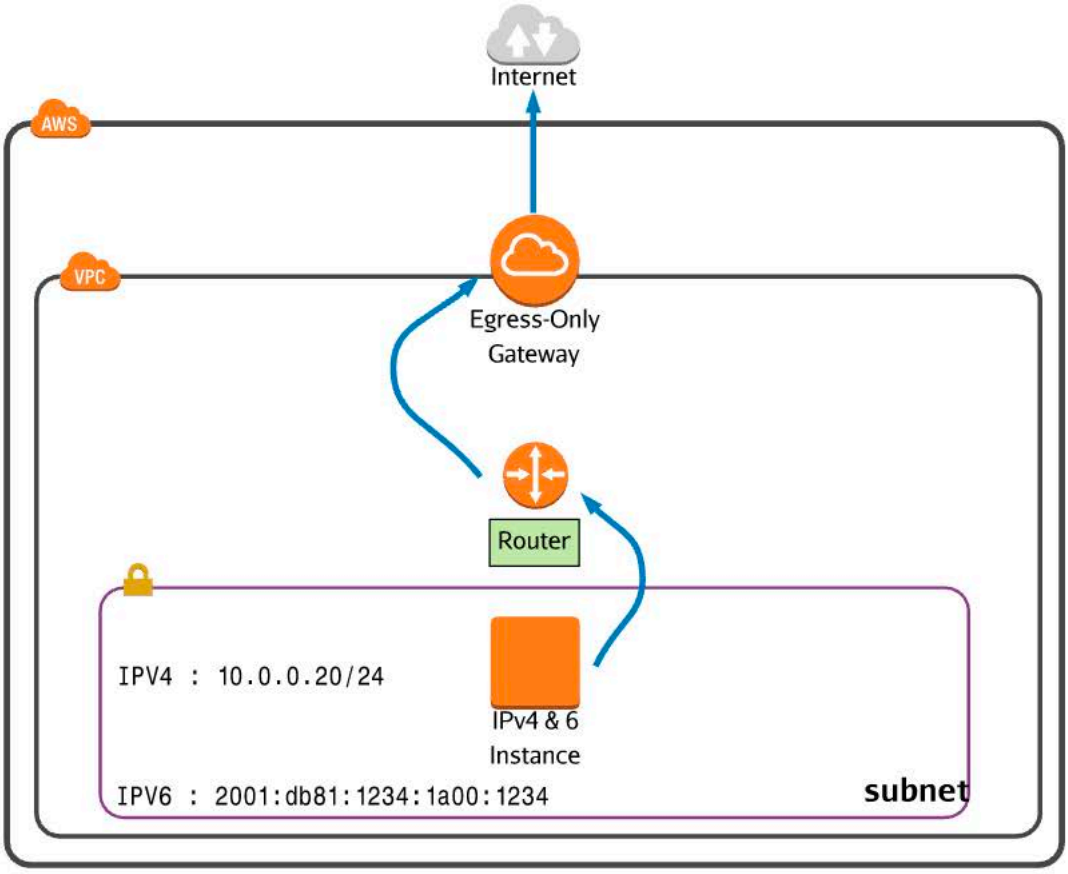


Design and implement a secure network infrastructure.

Egress-Only Internet Gateway

With IPv4, all AWS resources have a private IP. Some can be provided with a public IP and connectivity, using an Internet Gateway. With IPv4 a NAT instance/gateway can be utilized to provide outgoing only access.

IPv6 addressing is globally unique and publically routable. Supported resources in AWS are all publically addressable, so a NAT gateway isn't an option. An Egress-Only internet gateway provides a feature limited internet gateway, specifically for IPv6, and only allowing outbound connections and return traffic. No incoming IPv6 connections can be initiated to VPC resources using an Egress-Only gateway.



Design and implement a secure network infrastructure.

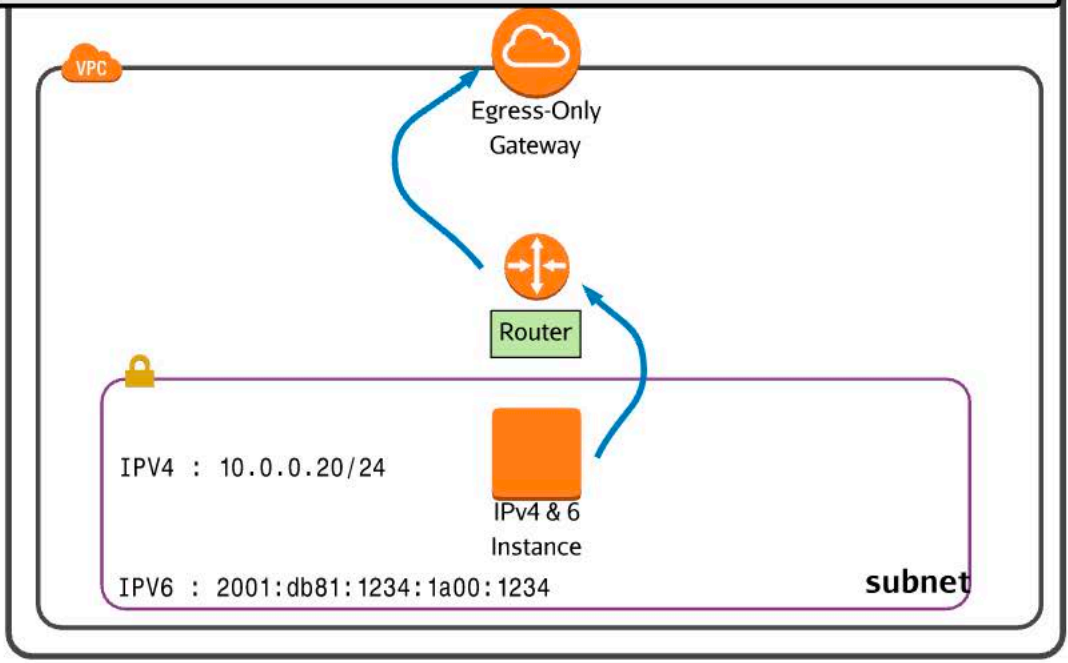
Egress-Only Internet Gateway

With IPv4, all AWS resources have a private IP. Some can be provided with a public IP and connectivity, using an Internet Gateway. With IPv4 a NAT instance/gateway can be utilized to provide outgoing only access.

IPv6 addressing is globally unique and publically routable. Supported resources in AWS are all publically addressable, so a NAT gateway isn't an option. An Egress-Only internet gateway provides a feature limited internet gateway, specifically for IPv6, and only allowing outbound connections and return traffic. No incoming IPv6 connections can be initiated to VPC resources using an Egress-Only gateway.

::/0	eigw-id

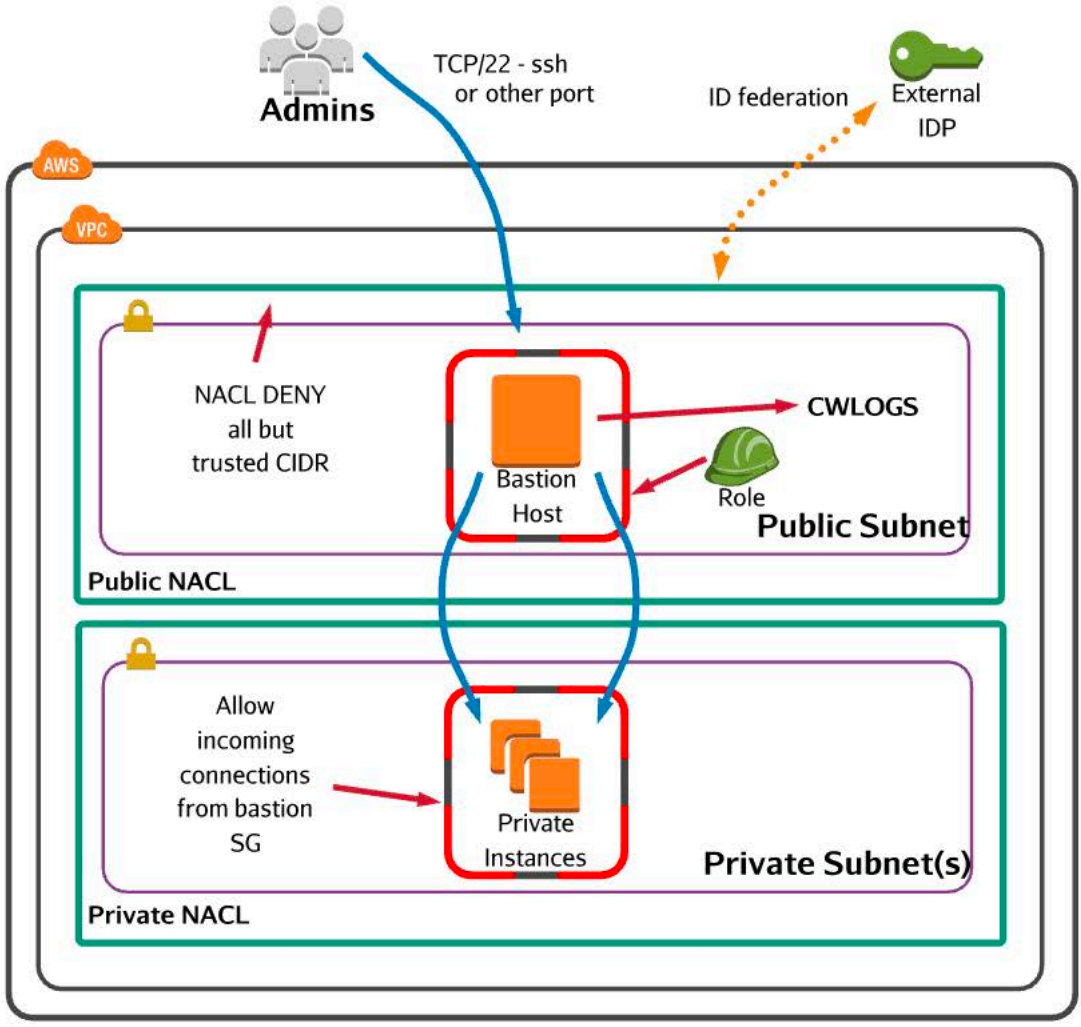
The VPC router via a route table needs to have a IPv6 default route (or a specific one) added. In this case, ::/0 with the Egress-Only Internet Gateway (EIGW) ID as a target. Optionally, you can have a route for IPV4 traffic via a NAT and IGW combination, but an EIGW can be used to provide IPv6 instances with outgoing only access - private IPV6 instances.



Design and implement a secure network infrastructure.

Bastion Host or Jumpbox

A Bastion host, or jumpbox, is a security concept which allows the access of otherwise private resources via a hardened, secure public connection point; generally a virtual server or EC2 instance. Bastion hosts are generally used when accessing an otherwise fully private VPC, or when a single publically accessible management server is required to reduce management overhead.



Troubleshoot a secure network infrastructure.

TroubleShooting VPC

Routing

- Routes are required at both sides
- Overlaps in CIDR are bad - splitting
- Check your CIDRs for typos

Filtering

NACL

- NACLs are great for explicit blocks
- They are dumb and stateless
- Rules are needed in both directions
- Rules are processed in order
- NACL's only apply to traffic crossing a subnet.
- Can ONLY reference IP/CIDR - no logical names or DNS names.
- One subnet, one NACL. NACL -> *Subnets

SG

- SGs apply to network interfaces, not VPCs and not subnets
- SGs can't dent traffic, only allow
- They are stateful - return traffic is allowed
- They can reference themselves and other SGs in a region (VPC Peering)
- They can use logical destinations and CIDR/IP
- * SG -> Interface, SG -> * Interfaces
- No order, all rules evaluated

Logging & Monitoring

- Use VPC Flow logs to check for Allow/Deny
- An allow and deny pair could indicate NACL allowing SG
- Use Cloudwatch logs/metrics

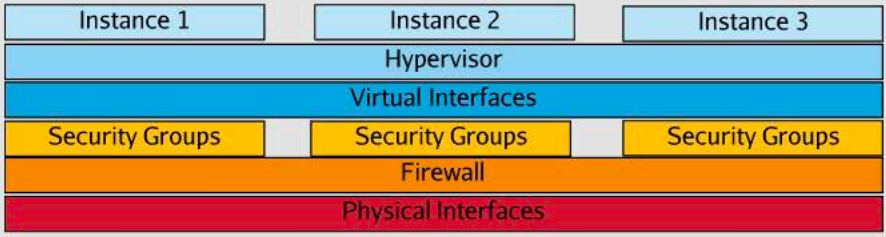
Design and implement host-based security.

AWS Host/Hypervisor Security (disk/memory)

Running any workloads in a shared environment is a risk unless precautions are taken to ensure the isolation of compute, storage, and networking between clients. AWS has implemented a number of steps to prevent data leakage between clients within their shared environment.

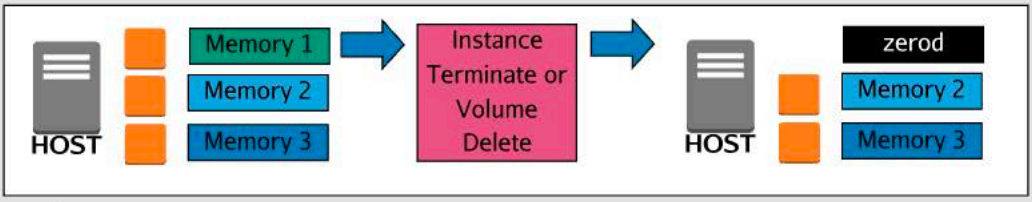
Isolation

Instances are always isolated from other customer instances, and unless you configure otherwise, other instances in your own environment. They have no direct access to hardware and *must* go via the hypervisor and firewall system.



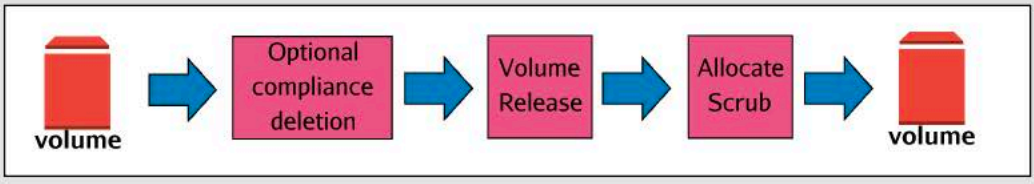
Memory

Host memory is allocated to an instance scrubbed, meaning it's set to zero. When unallocating memory, the hypervisor zeros it out *before* returning it to the pool.



Disk

EBS volumes are provided to instances in a Zero'd state - this zeroing occurs *immediately* before reuse. If you have specific deletion requirements, you need to do this *before* terminating the instance/deleting the volume.

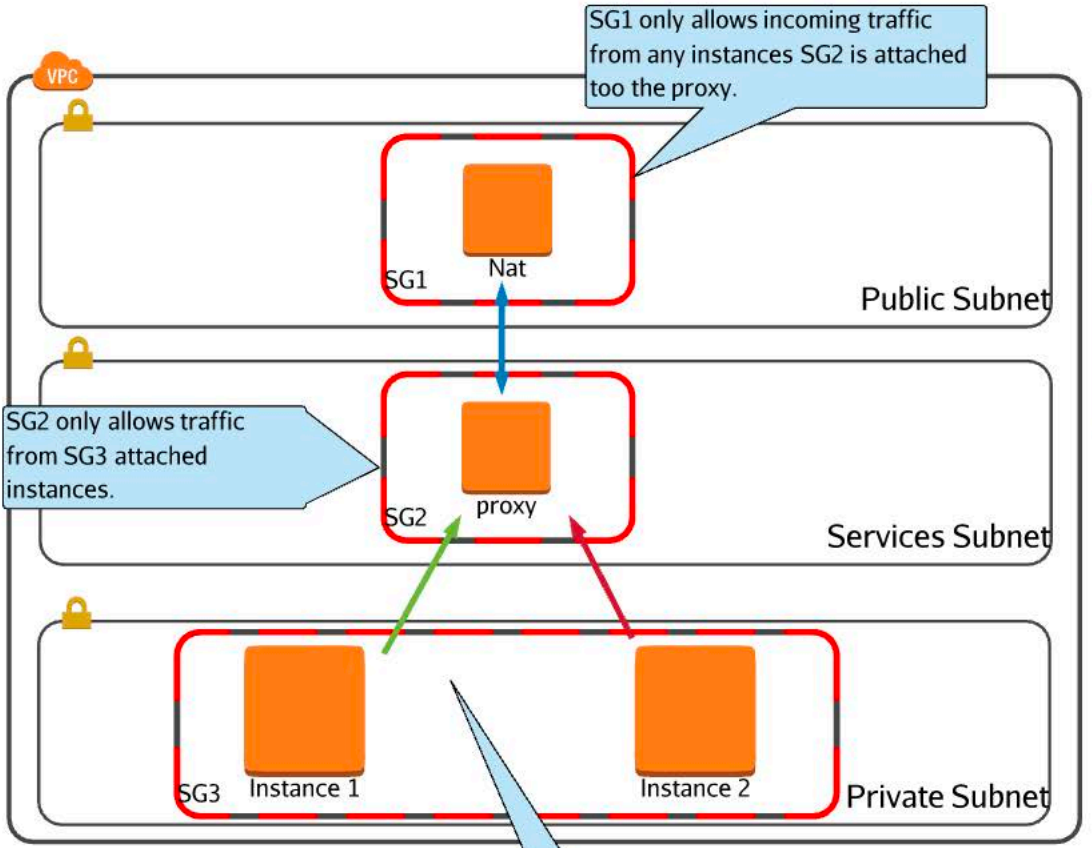


Design and implement host-based security.

Host Proxy Servers

Filtering within AWS is performed at two points: Security groups attached to network interfaces and Network ACLs (NACLS) attached to subnets within VPCs. Security groups and NACLS have viability of protocols, IPs, CIDRs, and ports. They cannot filter on DNS names, nor can they decide between allowing and denying traffic based on any form of authentication.

If authentication or additional intelligence beyond IP/CIDR/PORT/PROTOCOL is needed, a proxy server or an enhanced NAT architecture is required.



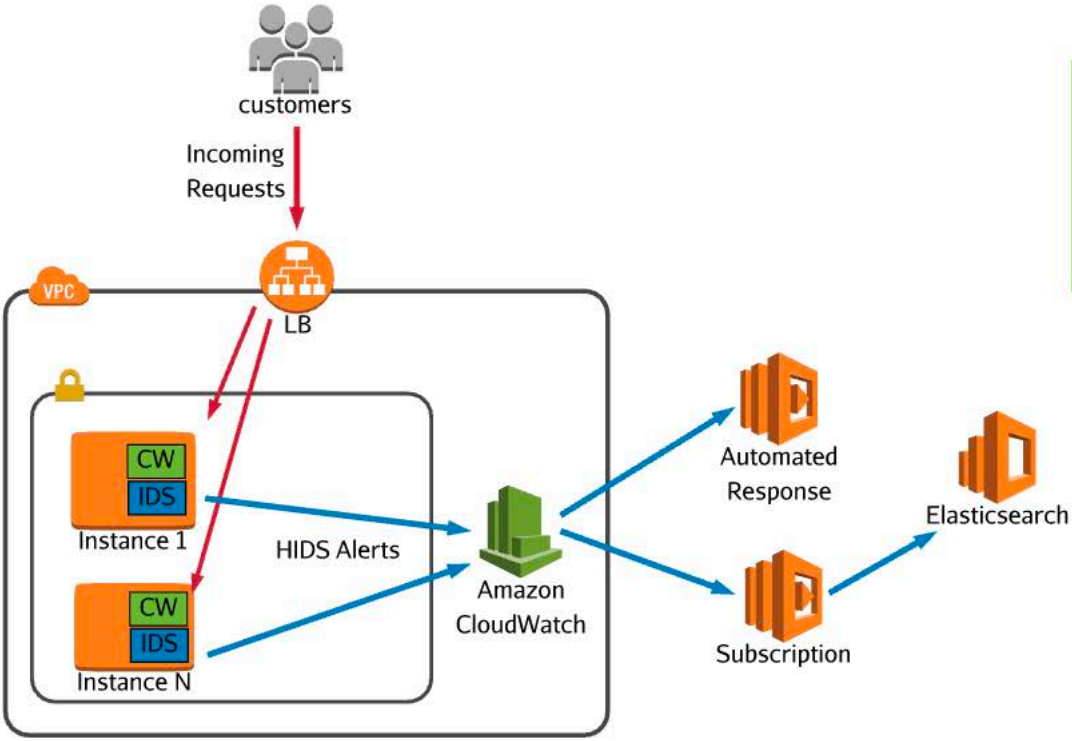
Design and implement host-based security.

Host-Based IDS/IPS

The Intrusion Detection System (IDS), is a category of software designed purely to monitor resources and identify any suspicious activity that could indicate a current, past, or future threat.

A host-based IDS solution compliments the features available within AWS :

- WAF — Provides edge security before a threat arrives at your environment edge.
- IDS Appliance — Monitor and analyses data as it moves into your platform.
- AWS Config — Ensures a stable and compliant configuration of account level aspects.
- SSM — Ensures compute resources are compliant with patch levels.
- Inspector — Reviews resources for known exploits and questionable OS/Software configurations.
- Host Based IDS — Handles everything else...



Design and implement host-based security.

Systems Manager

AWS systems manager is a systems management product. It provides insight, so information gathering and management, or action services to compute resources at scale. Its two core functional areas are insights and actions - insights gather information, actions perform tasks. It provides a wide range of additional functions with which assist with these core abilities.

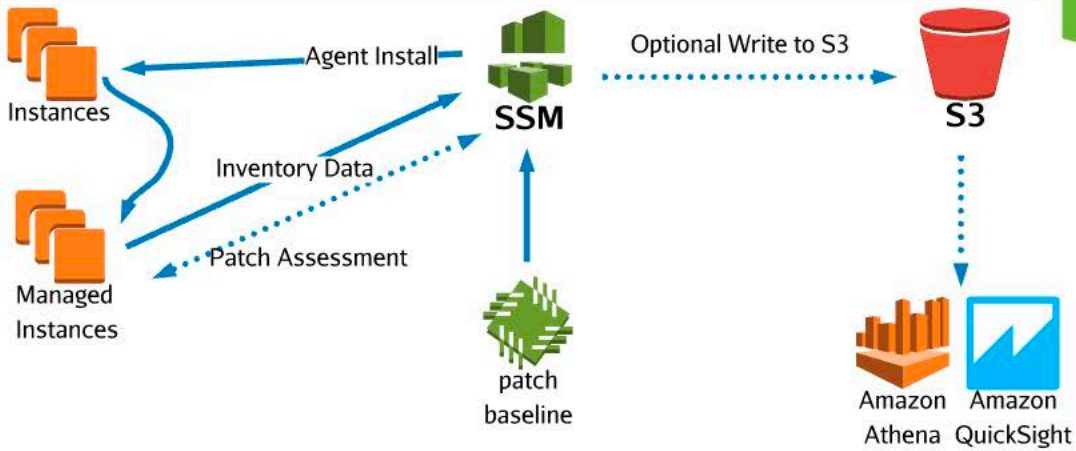
Insights | Actions | Shared Tooling

Insights

Systems manager offers two 'insight' features, Inventory and Compliance. Both supported by SSM State manager.

Inventory periodically scans EC2 instances, or on-premise servers/VMs, retrieving details of installed applications, AWS components, network config, windows updates, detailed information on an instance/VM, details on running services, windows roles, and optional custom data SSM can collect on your behalf.

Compliance allows that data to be compared against a 'baseline', providing a Compliant or Non-Compliant state to a resource. Compliance uses state manager, SSM patching, and custom compliance types.



Design and implement host-based security.

Systems Manager

AWS systems manager is a systems management product. It provides insight, so information gathering and management, or action services to compute resources at scale. Its two core functional areas are insights and actions - insights gather information, actions perform tasks. It provides a wide range of additional functions with which assist with these core abilities.

Insights | Actions | Shared Tooling

Actions

Actions are the operational engine part of systems manager. Actions is the part of systems manager which performs collections, runs commands, controls patching and manages the general state of *managed instances*.

Automation

Run Command

Patch Manager

State manager is a desired state engine. You define the 'desired state' in the form of a systems manager document. A document can be a 'command document' or a 'policy document'. A command document is used by running command and state manager, a policy document defines desired states and is only used by State Managed. A document is 'associated' with one or more managed instances.

Design and implement host-based security.

Systems Manager

AWS systems manager is a systems management product. It provides insight, so information gathering and management, or action services to compute resources at scale. Its two core functional areas are insights and actions - insights gather information, actions perform tasks. It provides a wide range of additional functions with which assist with these core abilities.

Insights | Actions | Shared Tooling

Shared Tooling

Systems Manager (SSM) provides a number of important services which support effective security within an AWS environment. You won't need to understand these in-depth for the exam, but it's worth knowing how they work for real-world usage:

- **Managed Instances** — Any machine (EC2 Instance, external VMs, or physical servers) configured to use systems manager. System manager supports a range of Linux distributions, Windows, and even IOT style devices such as a Raspberry Pi.
- **Activations** — The method used to activate non EC2 instances within Systems manager. Activation generates a code to activate the external machine.
- **Document** -— Think of these are scripts or lists of commands that can be run against a managed instance. We have used this elsewhere in the course. For example, to install the CloudWatchLogs agents.
- **ParameterStore** — An AWS provided services to store configuration data and secrets. It's a managed service, scalable, free to use, and is capable or storing KMS managed secrets. It can operate hierarchically i.e., /environmental/servertype/servername/param-name.

Design and implement host-based security.

Packet Capture on EC2

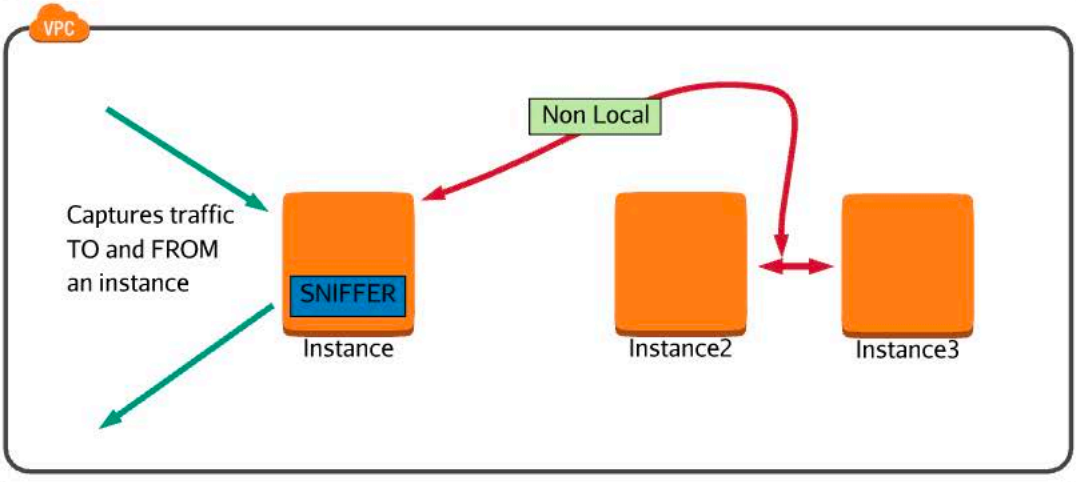
Packet capture or packet sniffing is a process where network traffic can be intercepted, analyzed and logged. Sniffed packets are captured in their entirety and unlike VPC flow logs can be inspected at a data level - providing they are not encrypted.

Common Scenarios:

- Review data flows between components to identify networking problems
- Support IDS/IPS systems - help detect and remediate intrusion attempts
- Debug connections between clients and the edge components of an environment
- Debug communications between tiers of your application
- Verify the functionality of other networking components such as firewalls, NATs, and Proxies

VPC flow logs meet a subset of the above scenarios but don't allow traffic capture - only metadata.

Important: Traditionally packet sniffing was done in a promiscuous way - a network interface listened for all traffic - even that not destined for the interface. This isn't supported in AWS.



By default, a network interface will only process traffic to or from itself and won't see traffic between instance 2 and 3 in this case. Historically, promiscuous mode could be enabled to see all traffic. The AWS hypervisor won't allow this; only sending instances relevant traffic. Sniffing is therefore only valid when installed on each instance.

Design and implement a scalable authorization and authentication system to access AWS resources

IAM Policies

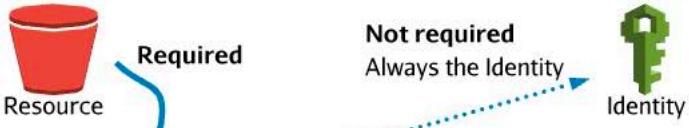
A policy document (JSON) defines if one or more actions, on one or more resources, with one or more conditions, is **allowed** or **denied**. They come in two main types:

- Identity Policies - Attached to identities (users, groups, roles)
- Resource Policies - Attached to resources

Anatomy

Policy Document - A list of statements...

```
{ "Version": "2012-10-17", "Statement": [ {..}, {..}, {..} ] }
```



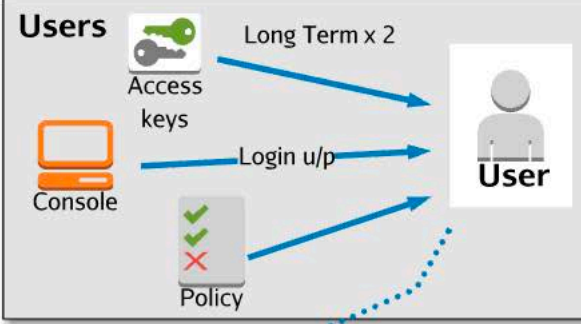

```
{
   "Principal" : "",
   "Effect": "Allow OR Deny",
   "Resource" : "*" OR "arn" OR ["arn", "arn2", ..],
   "Action" : "*" OR "service:operation" OR "service:*" OR [....]
   "Condition" : CONDITION
}
```

Design and implement a scalable authorization and authentication system to access AWS resources

Users, Groups and Roles

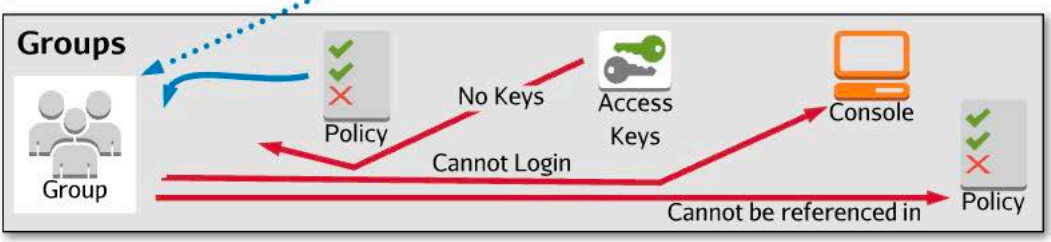
Users, Groups, and Roles are all identities managed by AWS Identity and Access Management (IAM). Each comes with its own set of features and limitations.

Users

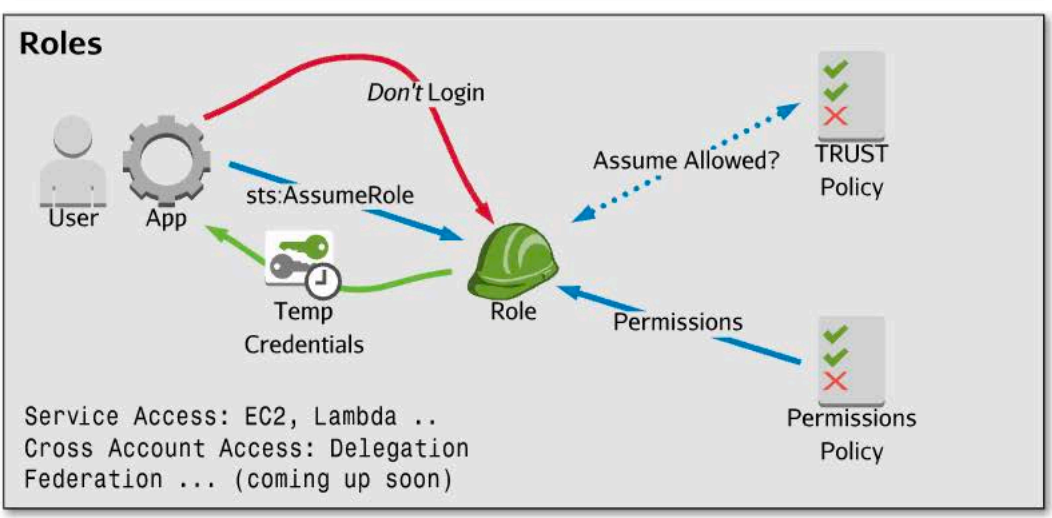


A User is an identity
Can login to console and use CLI/API
Has an ARN that can be referenced
No permissions by default
User as Service Account
Allows MFA

Groups



Roles

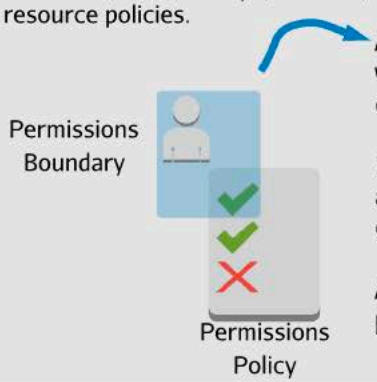


Service Access: EC2, Lambda ..
Cross Account Access: Delegation
Federation ... (coming up soon)

Design and implement a scalable authorization and authentication system to access AWS resources

Permissions Boundaries

A permissions policy **allows** or **denies**, **actions** on **resources**. Policies are applied to identities (Users, Groups, or Roles) in the case of identity policies, or resources in the case of resource policies.

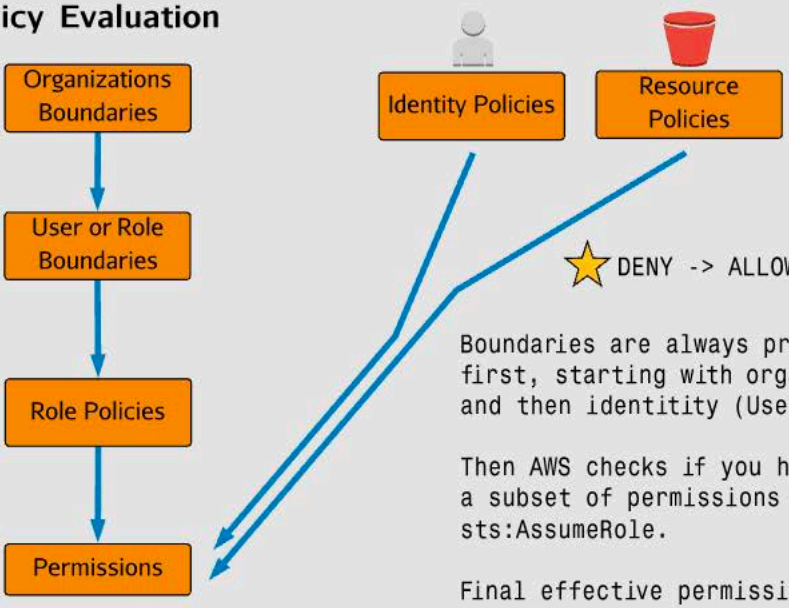


A permissions boundary is a set of access which an entity (user, role, organisation) can NEVER exceed.

It can act as a safety net to ensure adherence to organisational policies, or it can act as a delegation tool.

A Permissions boundary on its own grants no permissions, it only restricts.

Policy Evaluation



Boundaries are always processed first, starting with organizational and then identitity (User or Role).

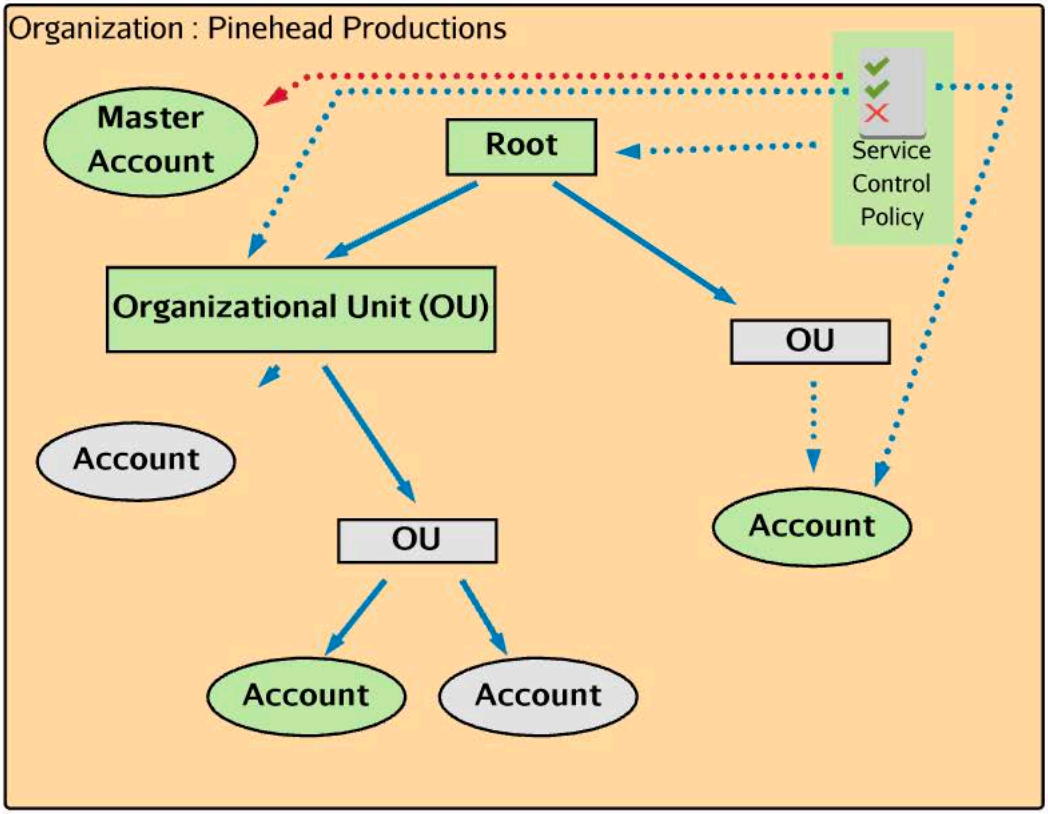
Then AWS checks if you have chosen a subset of permissions for a sts:AssumeRole.

Final effective permissions are a merge of identity, resource, and ACL.

Design and implement a scalable authorization and authentication system to access AWS resources

Organizations and Service Control Policies

AWS Organizations is an multi-account management system. It allows you to manage multiple accounts within an *organization*, which can be structured into *organisational units* (OUs). Accounts can be created directly from AWS organizations, or you can add accounts that already exist. Organizations offers centralized management of accounts, consolidated billing, and hierarchical control of accounts from a security perspective using Service Control Policies.

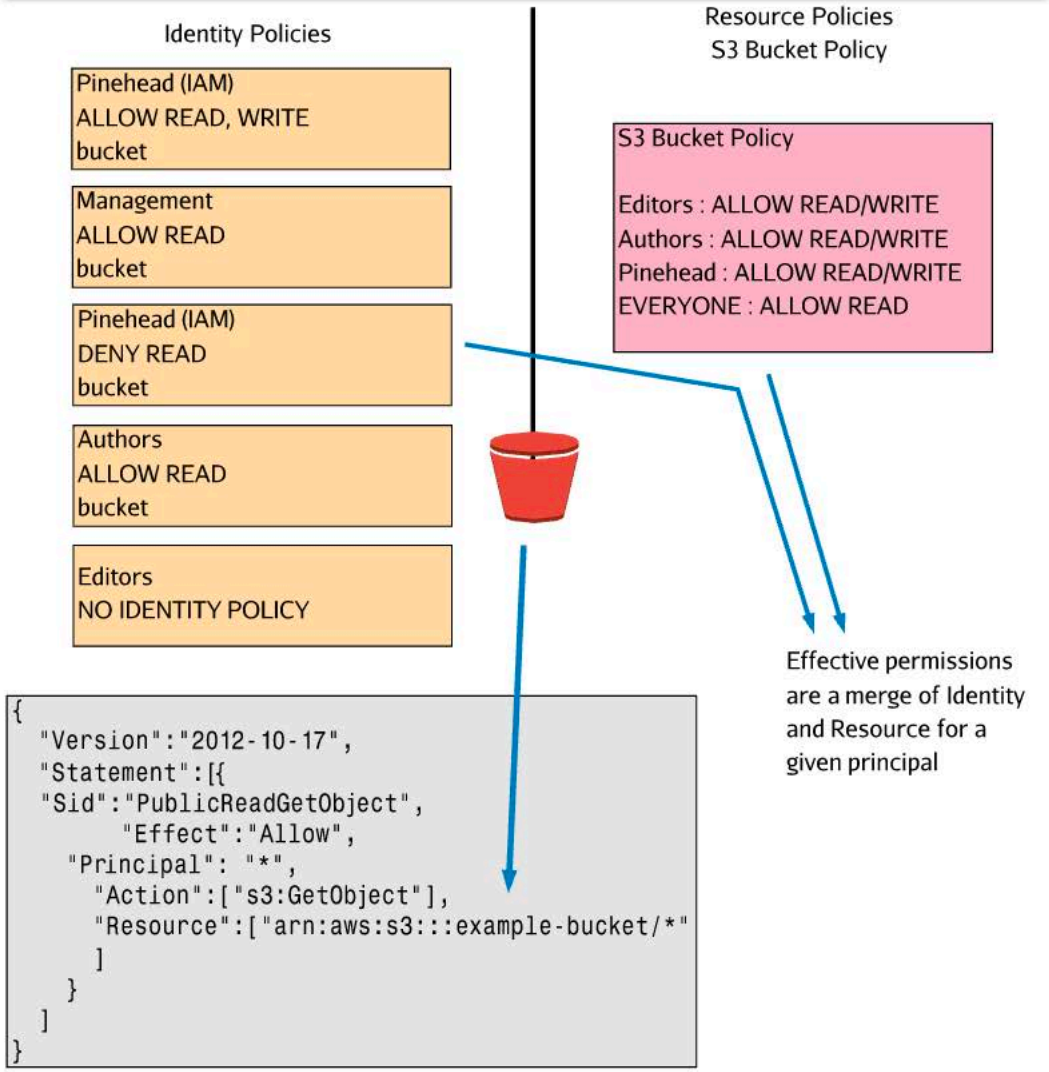


AWS Organisations is enabled from any AWS account. Once it's enabled, you can create an organisation and choose **just** consolidated billing or **All Features**, which includes service control policies and roles to allow account switching.

Design and implement a scalable authorization and authentication system to access AWS resources

S3 Bucket Policies

Identity-based IAM policies are attached to identities, and allow those identities to be ALLOWED or DENIED access to resources. Resource based policies, such as S3 bucket policies, are attached to a resource and control who has access to that specific resource from its perspective.




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

Design and implement a scalable authorization and authentication system to access AWS resources

KMS Key Policies

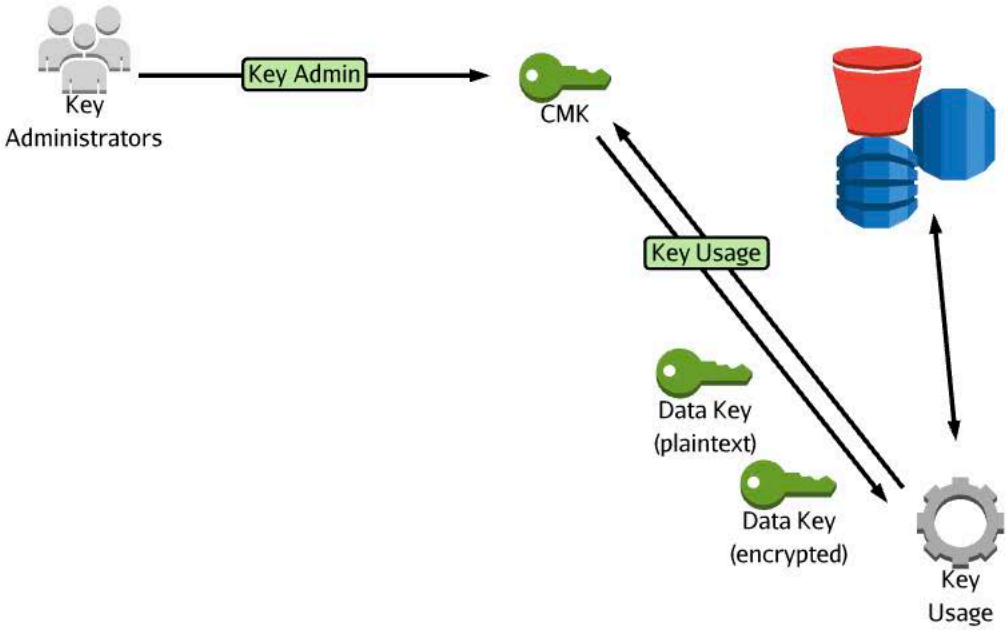
KMS Key policies are resource policies which control access to the Customer Master Keys (CMKs). Unlike most AWS services, without specifically being allowed, the `root` user has **no** access to CMKs. In cases where nobody has access to CMKs, **only** AWS can restore access.

```
{
  "Sid": "Enable IAM User Permissions",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:root"},
  "Action": "kms:*",
  "Resource": "*"
}
```

CMK

If removed, nothing can use the key. AWS support is required.

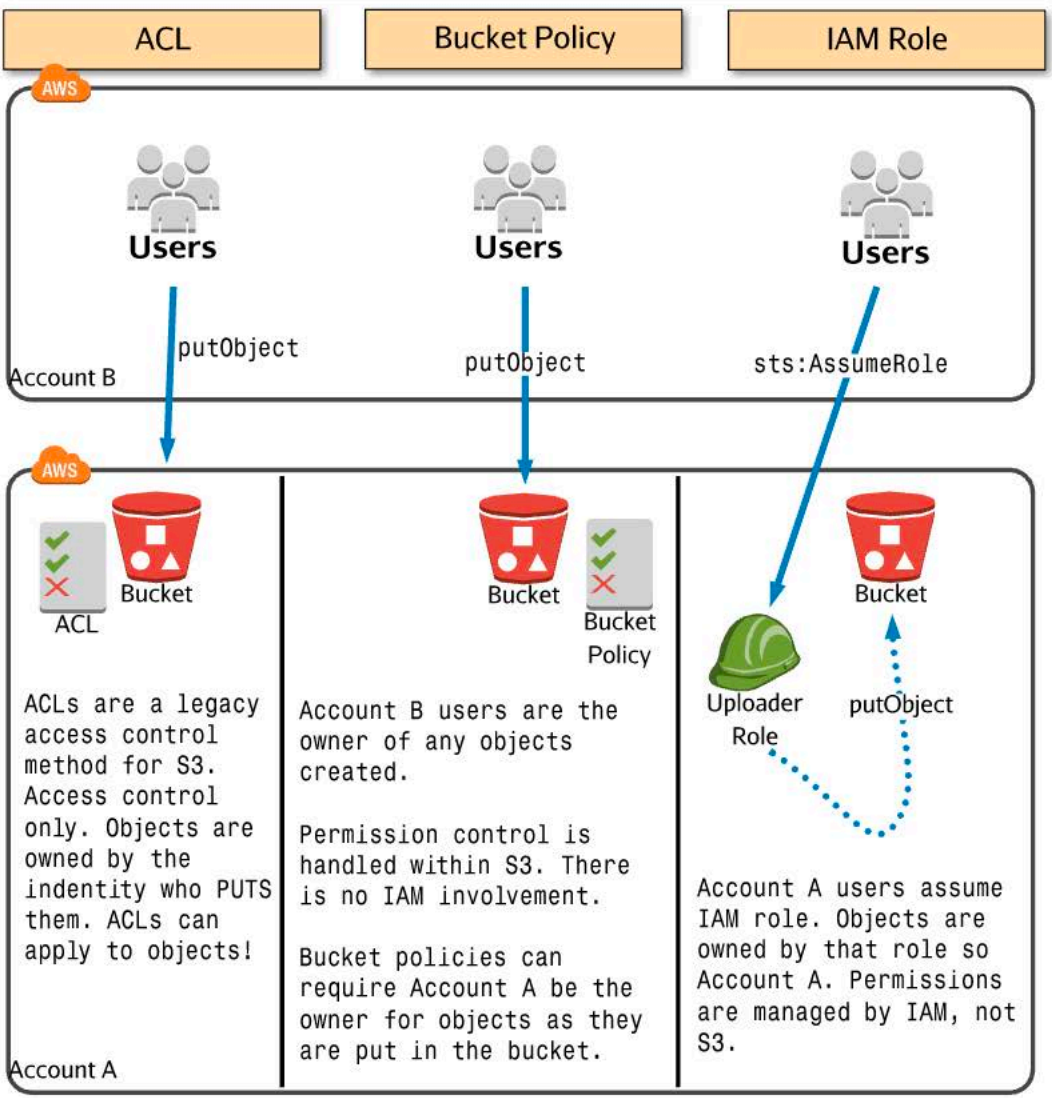
A Key Policy to give the account full control of a key, allowing IAM identity policies to control access.



Design and implement a scalable authorization and authentication system to access AWS resources

Cross-Account Access to S3 Buckets and Objects

There are three main ways to provide access to your S3 buckets from external AWS accounts. IAM Roles, Bucket Policies and Bucket ACL's. Knowing the suitability of each and their limitations is critical for the exam.



Design and implement a scalable authorization and authentication system to access AWS resources

Identity Federation

Identity Federation is where an AWS account is configured to allow external identities from an external identity provider (IdP). Organisations often maintain external IdPs for other business functions. With federation you can use these to login to AWS or access AWS services. Utilising federation means you can reuse accounts and not maintain login and authentication infrastructure.

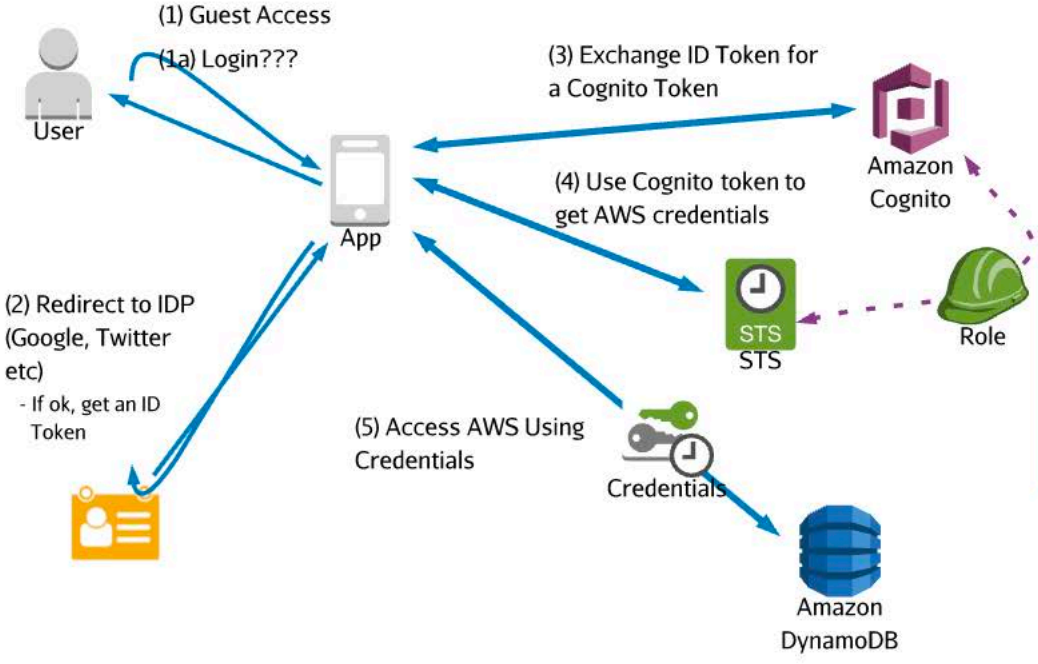
AWS Supports federation with IdPs which are OpenID Connect (**OIDC**) or **SAML 2.0** compatible.

Identity federation is generally grouped into three types:

- Web Identity Federation
- SAML 2.0 Identity Federation
- Custom ID Broker Federation (used when SAML2.0 compatability isn't available)

Web Identity Federation

SAML 2.0 Federation



Design and implement a scalable authorization and authentication system to access AWS resources

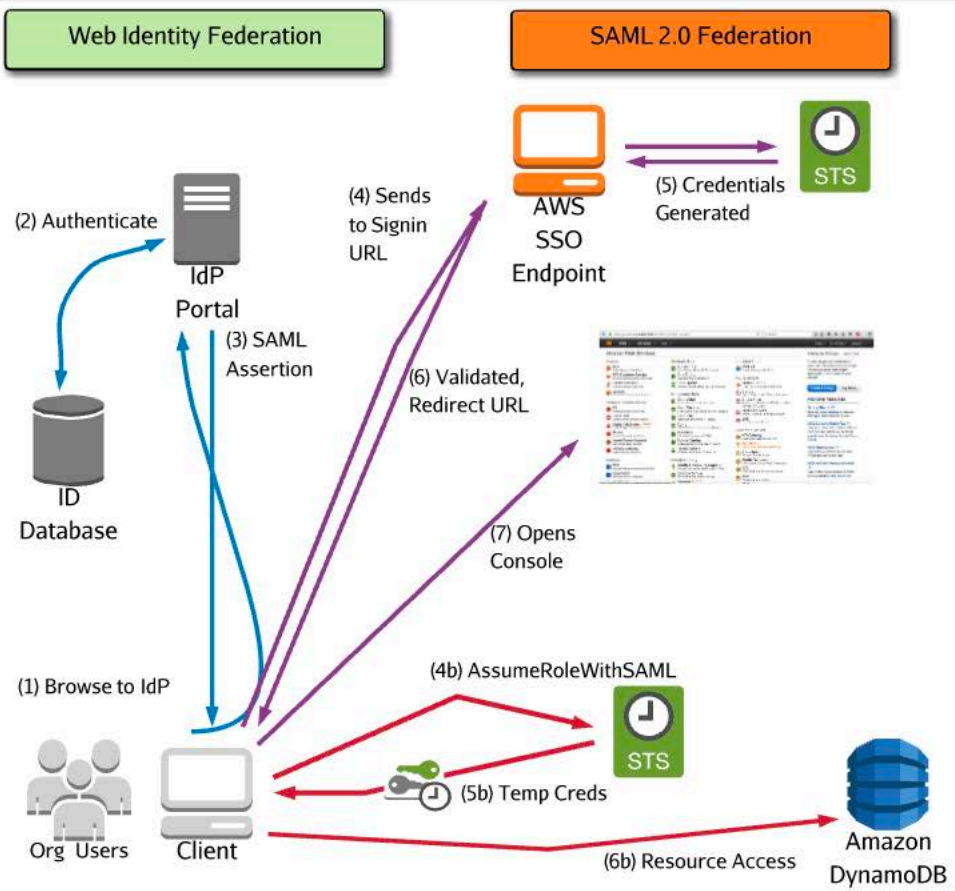
Identity Federation

Identity Federation is where an AWS account is configured to allow external identities from an external identity provider (IdP). Organisations often maintain external IdPs for other business functions. With federation you can use these to login to AWS or access AWS services. Utilising federation means you can reuse accounts and not maintain login and authentication infrastructure.

AWS Supports federation with IdPs which are OpenID Connect (**OIDC**) or **SAML 2.0** compatible.

Identity federation is generally grouped into three types:

- Web Identity Federation
- SAML 2.0 Identity Federation
- Custom ID Broker Federation (used when SAML2.0 compatability isn't available)



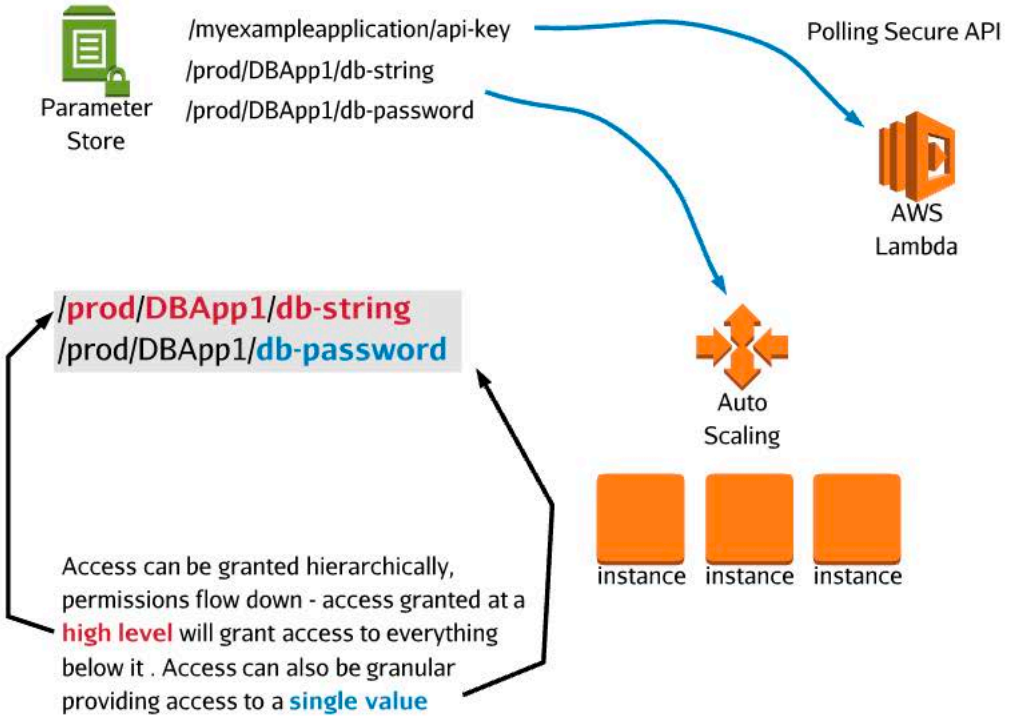
Design and implement a scalable authorization and authentication system to access AWS resources

AWS Systems Manager Parameters Store

AWS Papameter store provides secure storage for configuration data and secrets. Values can be stored as plaintext or as encrypted data using KMS. Data can be referenced using a unique name, providing you have permissions to access the data.

Key Features:

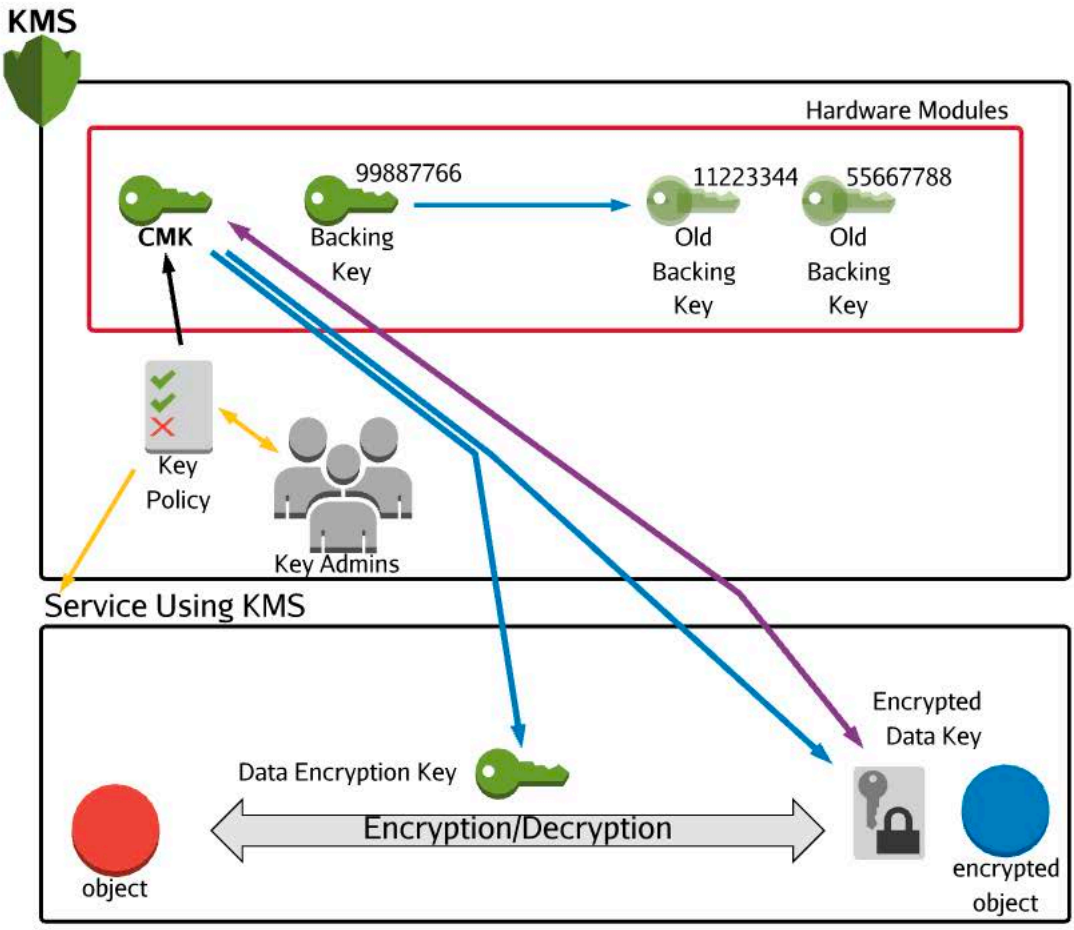
- Configuration and data is separated from code - no chance of leakage via Git
- Data is stored hierarchically - aids management
- Data is versioned, and access can be controlled and audited
- Parameter store integrates with many AWS services - EC2, ECS, Lambda, CodeBuild/Deploy, and many more
- Can also be used for automated deployment using cloudformation
- Serverless, resilient, and scalable



Design and implement key management and use.

Key Management Service (KMS)

KMS is a key management service which uses FIPS 140-2 compliant hardware modules to manage access to key material. It integrates fully with IAM and CloudTrail for permissions management and auditing functions. KMS can be optionally used with most AWS services that support encryption.

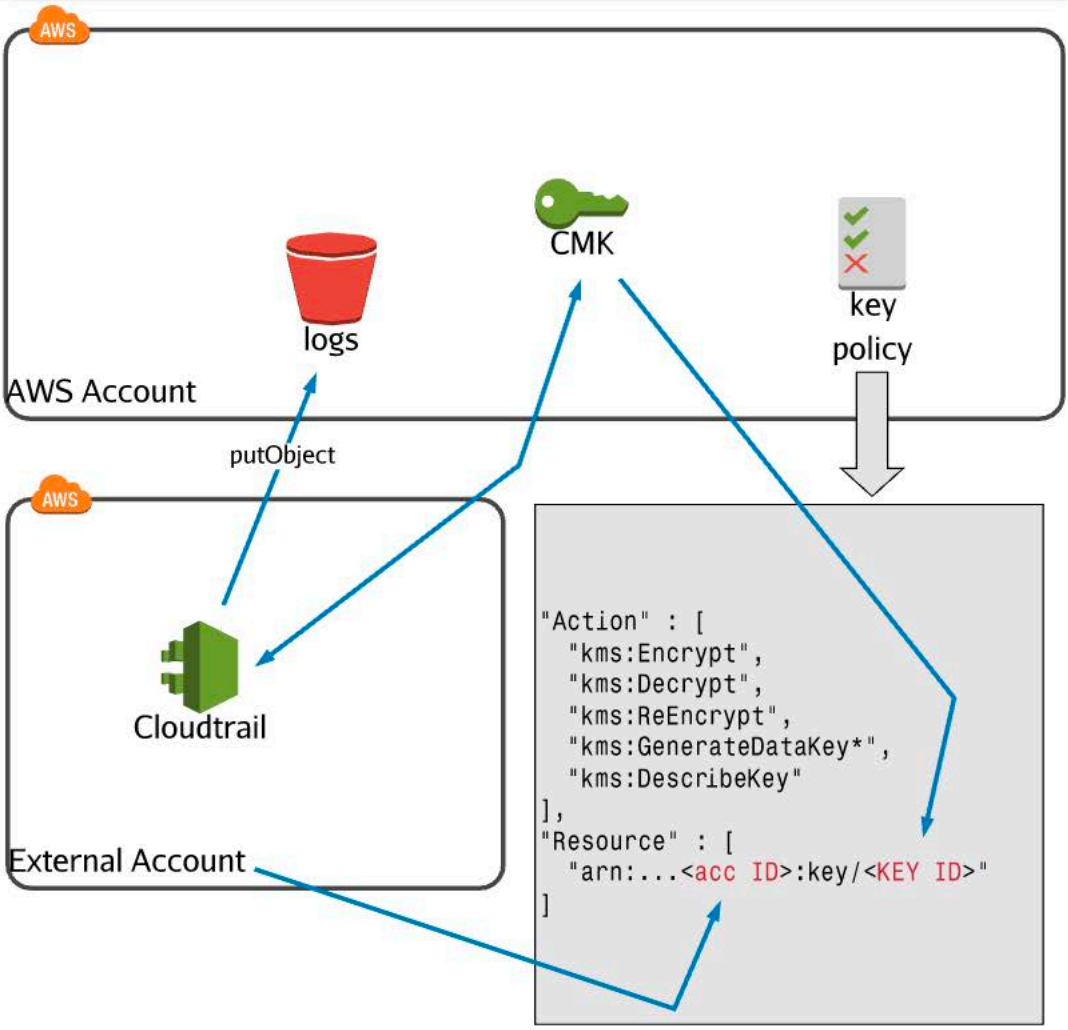


Design and implement key management and use.

KMS in a multi-account configuration

CMK's can be configured to allow other accounts to *use* them. The Key wont *appear* in the external account, but if it is configured using a key policy, that account can interact with the key for cryptographic functions.

Remember: Key *Usage* and Key *Admin* are not the same thing!



Design and implement key management and use.

CloudHSM & On-Premises HSM

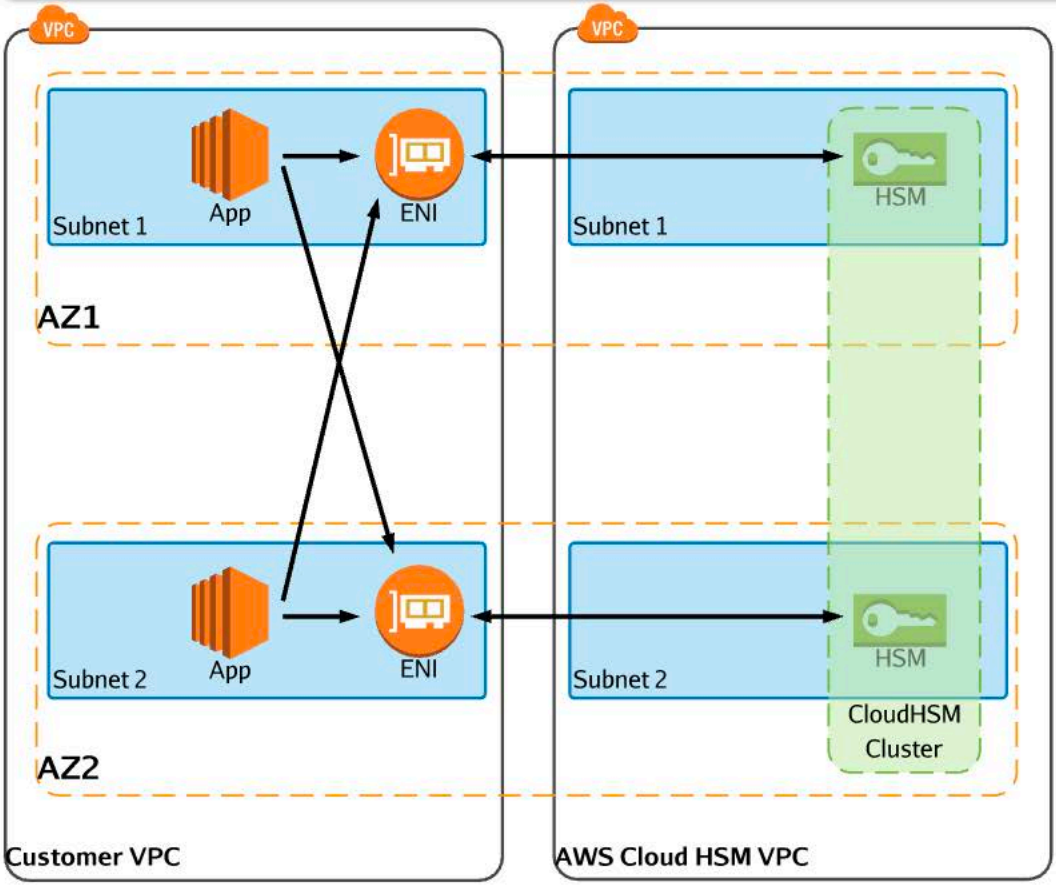
CloudHSM is a dedicated HSM which runs within your VPC, accessible only to you in a single tenant architecture. AWS manages and maintains hardware, but has ***no*** access to the cryptographic component. Interaction with CloudHSM is via industry standard APIs, ***no*** **normal AWS API's**.

- PKCS#11, Java Cryptography Extensions (JCE), Microsoft CryptoNG (CNG)

Keys can be transfered between CloudHSM and other hardware solutions (on premises). Keys are shared between cluster members. NO HA unless multiple HSM's are provisioned.

Applications can be **OUTSIDE** the VPC - Direct Connect, Peered or VPN.

On-Premises HSM - for if you ***really*** need to **control your own physical hardware**



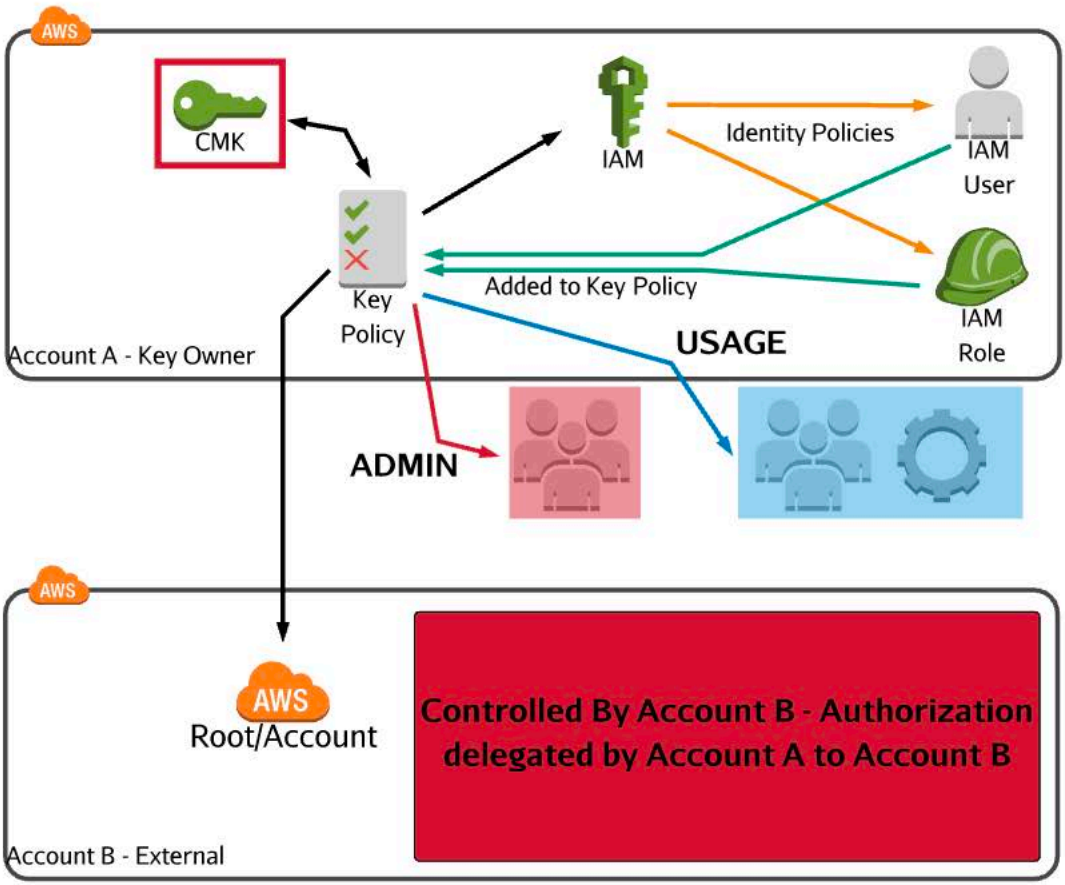
Troubleshoot key management.

Troubleshooting KMS Permissions

Permisions within KMS are centered around Customer Master Keys (CMKs). A default policy applied to a CMK trusts the account the key is created within, and this trust can be provided to IAM users via IDENTITY policies, or, on the KEY policy itself.

Permissions within KMS are either ADMIN permissions, or USAGE permissions.

WARNING: You CAN lock out a CMK making it unusable to everyone.



Troubleshoot key management.

KMS Limits

KMS has some limits that you should be aware of, simple limits, rate limits and cross-account limits.

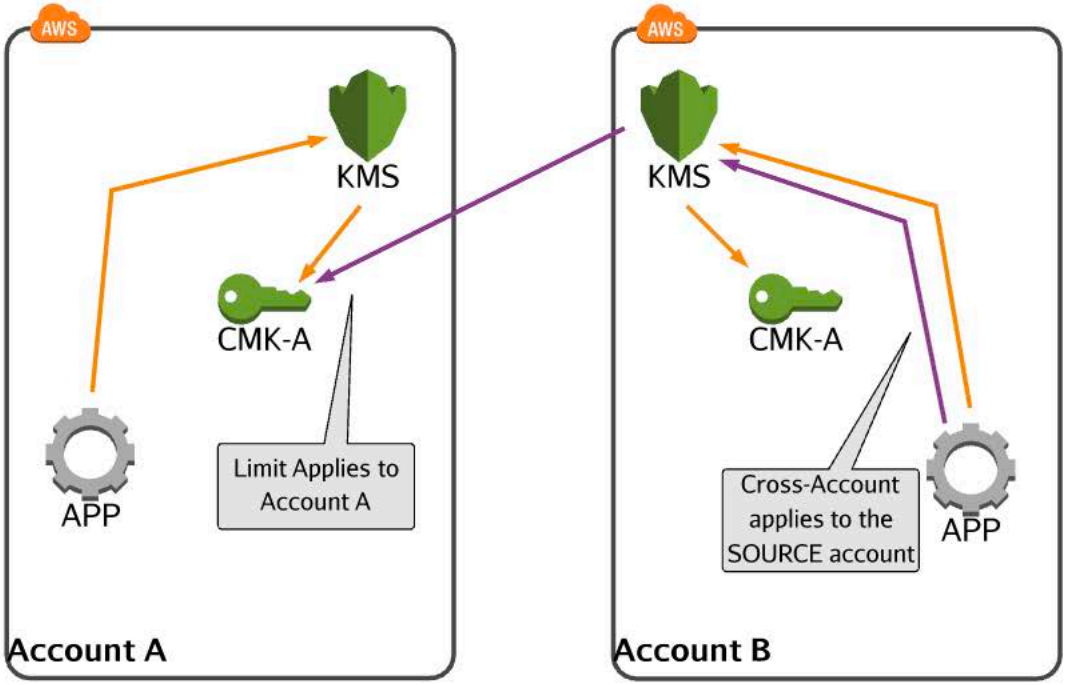
Simple Limits :-

1000 (customer managed) CMK's per region - in ANY state

1100 Aliases per account

2500 Grants per CMK - e.g max of 2500 EBS Volumes using a CMK

Breaching the shared, or per operation limits result in KMS throttling the requests



There is a 5500 Shared API limit shared across a number of operations relating to KMS - the high volume operations. The default is 5500 - us-east-1, us-west-2 and eu-west-1 currently have a 10,000 limit.

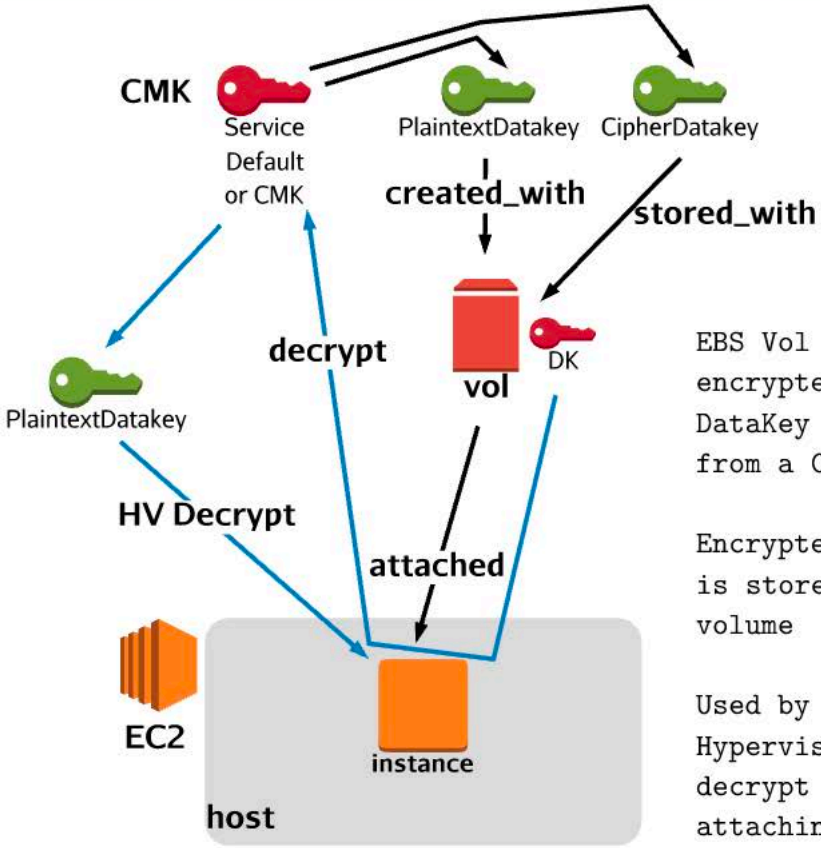
```
Decrypt
Encrypt
GenerateDataKey
GenerateDataKeyWithoutPlaintext
GenerateRandom
ReEncrypt
```

Design and implement a data encryption solution for data at rest and data in transit.

Data at Rest : KMS

KMS integrates with many other AWS Products to provide encryption services and key management. The exact form this integration takes is service specific. KMS responsibility ends at CMKs or DataKeys (handed over to services and not managed by KMS):

EBS | **DynamoDB** | **RDS** | **S3**



EBS Vol is encrypted using a DataKey generated from a CMK.

Encrypted data key is stored with volume

Used by the Hypervisor to decrypt upon attaching

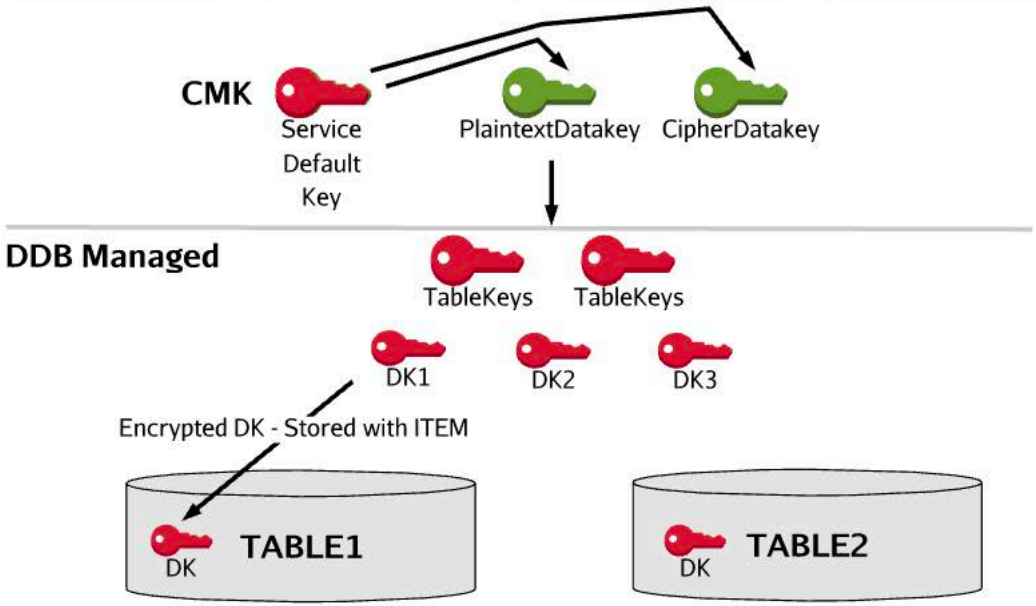
IO, Snapshots and Persisted data is Encrypted

Design and implement a data encryption solution for data at rest and data in transit.

Data at Rest : KMS

KMS integrates with many other AWS Products to provide encryption services and key management. The exact form this integration takes is service specific. KMS responsibility ends at CMKs or DataKeys (handed over to services and not managed by KMS):

EBS | **DynamoDB** | **RDS** | **S3**



For any encrypted table created in a region, DynamoDB uses KMS to create an AWS/DynamoDB service default CMK (in each region).

When a table is created and set to be encrypted, this CMK is used to create a data key unique to that table, called a table key. This key is managed by DynamoDB and stored with the table in an encrypted form.

Every ITEM that DynamoDB encrypts is done with a data encrypted key. That key is encrypted with this table key, and stored with the data.

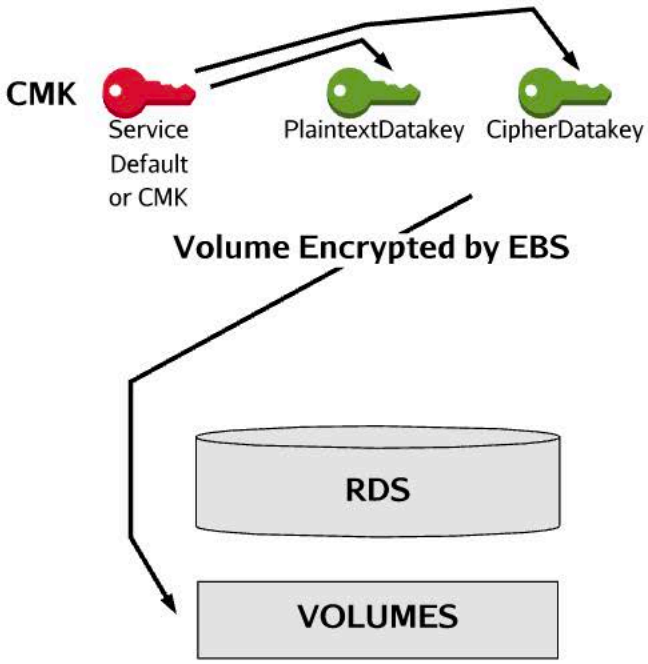
Table keys are cached for up to 12 hours in plaintext by DynamoDB, but a request is sent to KMS after 5 minutes of table key inactivity to check for permissions changes.

Design and implement a data encryption solution for data at rest and data in transit.

Data at Rest : KMS

KMS integrates with many other AWS Products to provide encryption services and key management. The exact form this integration takes is service specific. KMS responsibility ends at CMKs or DataKeys (handed over to services and not managed by KMS):

EBS | **DynamoDB** | **RDS** | **S3**



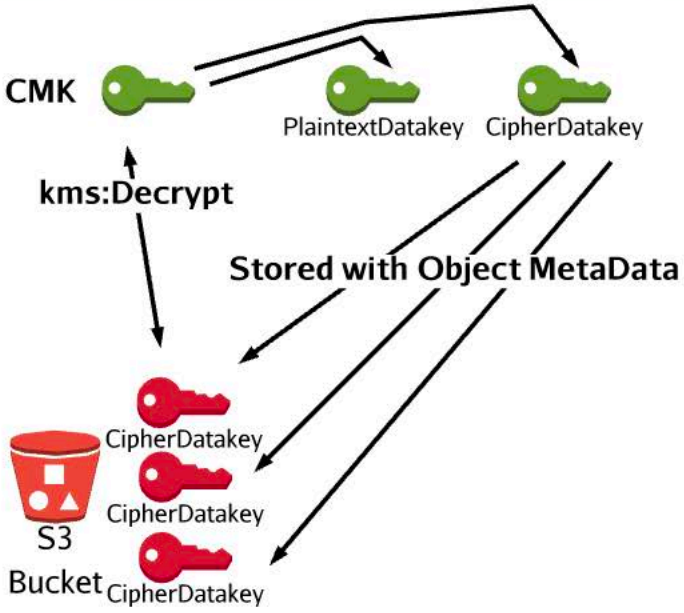
RDS utilizes EBS for its encryption. RDS instances are managed versions of EC2 instances, configured to act as a managed DB cluster. In a similar way to EC2, encrypted volumes attached to RDS are handled by the host, with persistent data, snapshots, and IO encrypted and decrypted using KMS.

Design and implement a data encryption solution for data at rest and data in transit.

Data at Rest : KMS

KMS integrates with many other AWS Products to provide encryption services and key management. The exact form this integration takes is service specific. KMS responsibility ends at CMKs or DataKeys (handed over to services and not managed by KMS):

EBS | **DynamoDB** | **RDS** | **S3**



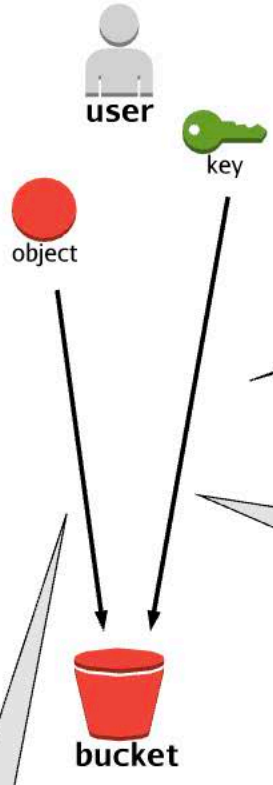
Every object in a bucket is encrypted by S3 using a DataKey provided by KMS. The DataKey is generated from a CMK (Service Default or a custom CMK).

CipherText DataKey is stored with the object as metadata. When decryption is needed, it's passed to KMS, Decrypted, and used by S3 to Decrypt the Object.

Design and implement a data encryption solution for data at rest and data in transit.

Data at Rest : S3 Customer Provided Encryption Keys (SSE-C)

SSE-C is a feature of S3 Server Side Encryption where S3 still handles the cryptographic operations, but does so with keys that you as the customer manage and supply with every object operation.



x-amz-server-side-encryption-customer-key allows the key to be provided. The key is used for encryption, and then discarded. The customer is 100% responsible for key management and rotation. Versions can have alternative keys.

x-amz-server-side-encryption-customer-key-MD5 allows S3 to validate the key (for damage is transit).

x-amz-server-side-encryption-customer-algorithm = AES256 informs S3 that a customer managed key will be supplied as part of the putObject request.

Design and implement a data encryption solution for data at rest and data in transit.

Data in Transit: Certificate Manager (ACM)

ACM is a managed service providing X509 v3 SSL/TLS certificates. The Certificates are Asymmetric. One half is private and stored on resources (Servers, Load balancers etc), and the other half is public. The latter can prove to the client that a remote resource has the private component. ACM integrates with other AWS services and generates certificates that are valid for 13 months.

Key Features:

- Native integration with ELB, CloudFront, AWS Elastic Beanstalk & API Gateway
- No cost associated with Certificates - only the resources they are used with
- Certificates automatically renewed when actively used within supported services
- Integrates with Route53 to perform DNS checks as part of certificate issuing process
- ACM is regional - certificates can be applied to services in that region
- **KMS is used - certificates are *never* stored unencrypted**