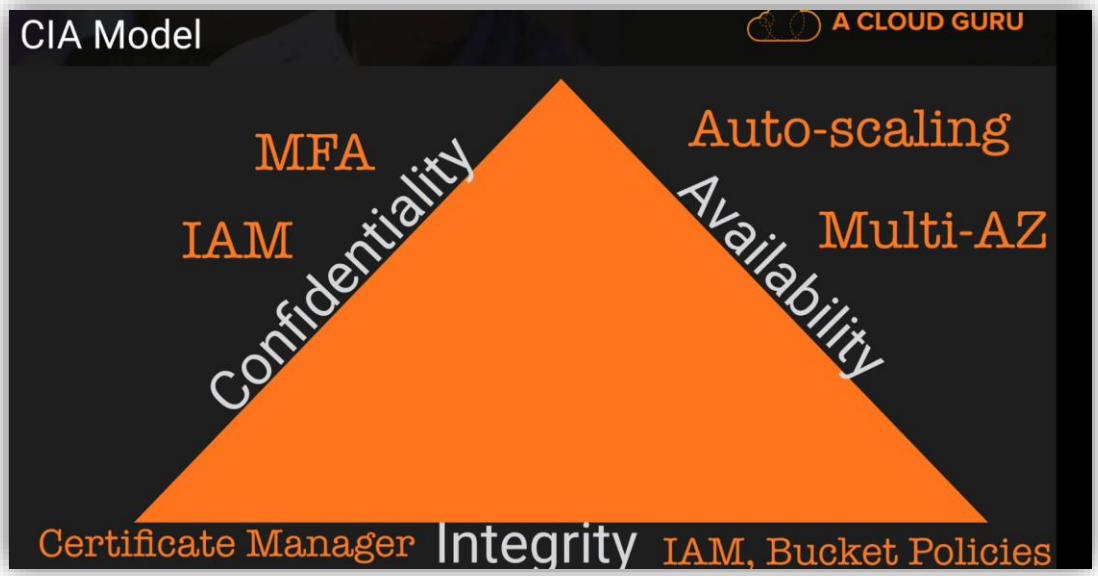
AWS Certified Security - Specialty

Contents

CIA:

- **Confidentiality:** IAM, MFA
- **Availability:** Auto-scaling, multi-AZ
- **Integrity:** Certificate Manager, IAM, Bucket Policies



AAA Model:

- Authentication (IAM), Authorization (Policies), Accounting (CloudTrail)

Security in AWS:

Visibility:

- What assets do customers have? – AWS Config
- AWS Config – provides an inventory of customer's AWS resources, history of configuration changes to these resources, can define rules that evaluate these configurations for compliance
    - A service that enables to assess, audit, and evaluate the configurations of your AWS resources

Auditability:

- Does customer comply with policies and regulations? – AWS CloudTrail
- AWS CloudTrail – records every API call in the customer's environment and provides event history of AWS account activity, including actions taken through the AWS Management Console, AWS SDKs, command line tools, and other AWS services.

Controllability:

- Is customer's data controlled? – AWS KMS, AWS CloudHSM

- AWS KMS – multi-tenant
- CloudHSM – dedicated – FIPS 140-2 Compliance
- AWS Key Management Service (KMS) makes it easy for customers to create and manage keys and control the use of encryption across a wide range of AWS services and in customer's applications.

Agility:

- How quickly can customers adopt changes?
- AWS CloudFormation, AWS Elastic Beanstalk
- AWS Elastic Beanstalk makes it even easier for developers to quickly deploy and manage applications in the AWS Cloud.
    - Developers simply upload their application, and Elastic Beanstalk automatically handles the deployment details of capacity provisioning, load balancing, auto-scaling, and application health monitoring.
- AWS CloudFormation is a service that gives developers and businesses an easy way to create a collection of related AWS and third-party resources, and provision and manage them in an orderly and predictable fashion.

Automation:

- Are customer's processes repeatable?
- AWS OpsWorks, AWS CodeDeploy
- CodeDeploy is a deployment service that automates application deployments to Amazon EC2 instances, on-premises instances, serverless Lambda functions, or Amazon ECS services
- AWS OpsWorks is a configuration management service that provides managed instances of Chef and Puppet. Chef and Puppet are automation platforms that allow users to use code to automate the configurations of customer's servers.
- AWS CodeDeploy is a fully managed deployment service that automates software deployments to a variety of compute services such as Amazon EC2, AWS Fargate, AWS Lambda, and on-premises servers.

Scale:

- Amazon CloudWatch is a monitoring and management service which collects monitoring and operational data in the form of logs, metrics, and events, providing with a unified view of AWS resources, applications and services that run on AWS, and on-premises servers

Logging in AWS:

- Services include:
    - CloudTrail – trace all API calls
    - Config Rules – for config and compliance over time
    - CloudWatch Logs – for full data retention
    - VPC Flow Logs – IP traffic within the VPC
    - ELB Access Logs – metadata of requests made to the user's load balancer

    - CloudFront Logs – web distribution access logs
    - WAF Logs – full logging of all requests analyzed by the service
- Logs can be analyzed using AWS Athena if they're stored in S3
- Logs should be encrypted in S3, access should be controlled using IAM & Bucket policies, MFA
- Prevent unauthorized access to Log Files using:
    - IAM users, groups, roles, and policies
    - Amazon S3 bucket policies
    - Multi factor authentication
- Ensure role-based access:
    - IAM users, groups, roles, and policies
    - Amazon S3 bucket policies
- Alerts when logs are created or fail:
    - CloudTrail notifications
    - AWS Config Rules
- CloudTrail SNS notifications only point to log file location
- Log changes to system components – config changes etc.
    - AWS Config Rules
    - AWS CloudTrail
- Controls exist to prevent modifications to logs:
    - IAM and S3 controls and policies
    - CloudTrail log file validation
    - CloudTrail log file encryption
- Logs are stored for at least one year
- Logs can be moved to Glacier for cost saving

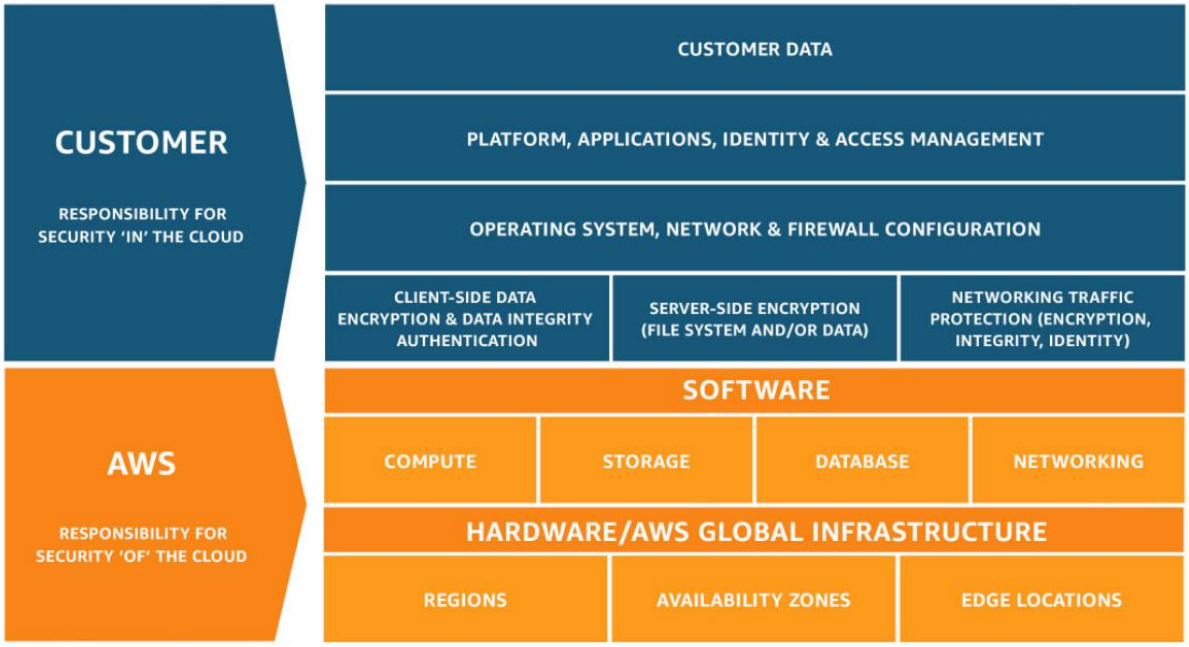
Monitor Account Activity:

- AWS Config Configuration History:
    - Must have AWS Config Configuration Recorder on
- CloudTrail Event History:
    - Search API history for past 90 days
    - Filter by resource name, resource type, event name, etc.
    - Filter by IAM user, assumed IAM role session name, or AWS Access Key
- CloudWatch Logs Insights:
    - Search API history beyond the past 90 days
    - CloudTrail Trail must be configured to send logs to CloudWatch logs
- Athena Queries:
    - Search API history beyond past 90 days in S3

Shared Responsibility Model:

- Security and Compliance are shared responsibilities.

- AWS is responsible for protecting the infrastructure that runs all the services offered in the AWS Cloud. This infrastructure is composed of the hardware, software, networking, and facilities that run AWS Cloud services.
- If a customer deploys an Amazon EC2 instance, they are responsible for management of the guest operating system (including updates and security patches), any application software or utilities installed by the customer on the instances
- Shared Responsibility Model Documentation Link



AWS Responsibility:

- AWS managed the security of the cloud
- Global infrastructure
- Hardware, software, networking, and facilities
- Managed services
- Compute, storage, database, networking
- Regions, AZs, edge locations

Customer Responsibility:

- Security in the cloud is responsibility of the customer
- Retain control of what security they choose to implement to protect their own content
- Infrastructure as a service (IaaS)
- Updates and security patches
- Configuration of AWS-provided firewall
- Customer data
- Platform, applications, IAM
- OS, network, and firewall configurations

- Client-side data encryption and data integrity authentication
- Server-side encryption (filesystem or data)
- Network traffic protection (encryption/integrity/identity)

Infrastructure Services:

- Includes computing services – EC2, EBS, Autoscaling, VPC
    - EC2 – AMIs, OS, Applications, Data in transit, Data at rest, Data stores, Credentials, Policies and Configurations
- Can be used to architect and build a cloud infrastructure using technologies
- Customer controls the operating system, configure, and operate any IAM

Container Services:

- Typically run on separate Amazon EC2 or other infrastructure instances, sometimes customer doesn't manage the OS or platform layer
- AWS provides a managed service for these application "containers"
- Customer is responsible for setting up and managing network controls, firewall rules, managing platform-level identity and access management separate from IAM
- Examples – RDS, EMR, Elastic Beanstalk

Abstract Services:

- Includes high-level storage, database, and messaging services – S3, Glacier, DynamoDB, SQS, SES
- These services abstract the platform or management layer on which customer can build and operate cloud applications
- Customer can access the endpoints of these abstract services using AWS APIs – AWS manages the underlying service components or the operating system on which they reside

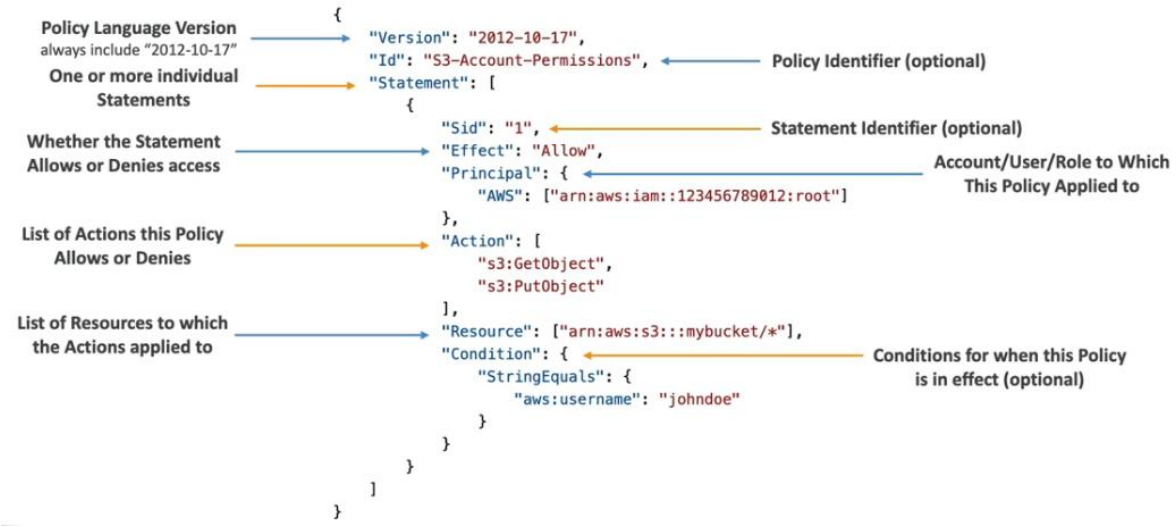
AWS IAM:

- Identity & Access Management
- Allows to manage users and their level of access to the AWS console
- Centralized control of AWS account
- Shared access to AWS account
- Granular permissions
- Identity federation – Active Directory, Facebook/Linkedin etc.
- Multifactor authentication
- Provide temporary access for users/devices and services where necessary
- Allows to setup customized password rotation policy
- Integrates with many AWS services
- Supports PCI DSS compliance
- Power users don't have access to IAM
- Users – end users
- Groups – a collection of users under one set of permissions
- Roles – can assign them to AWS resources

- Embedding user credentials in application code is insecure and not recommended – roles can be used to delegate access to users, applications, or services that don't normally have access to AWS resources
    - If the IAM console is used to create the role, an instance profile is created with the same name. The role that was created is added to the instance profile. An instance profiles contain one IAM role
- Policies – a document that defines one or more permissions
- When using AWS IAM to create user accounts you can specify a password policy in IAM
- When using federation with Active Directory and AWS IAM the user account is created and managed in Active Directory so the password policy should be configured there
- IAM Lecture Video

IAM Policies:

- Specified what principle are allowed to do with any AWS resource
- Global and apply all areas of AWS
- Can be attached to IAM users, groups, and roles – these users, groups and roles are then subject to the permissions defined in the attached policy
- You can attach an IAM Policy to a user, group, or role. You can associate a role with an EC2 instance, but you cannot attach an IAM Policy directly to the EC2 instance. You cannot attach policies to S3 buckets
- Three different types – AWS managed policies, customer managed policies, inline policies
- AWS Managed Policies:
    - Standalone policy that is created and administered by AWS
    - They can be changed
    - They are the same applied across multiple accounts
- Customer Managed Policies:
    - Standalone policies that customers administrator in their own AWS account
    - These policies can be attached to multiple principal identities in the AWS account
    - When the policy is attached to a principal entity, it gives the identity the permissions that are defined in the policy
- Inline Policies:
    - Useful for strict one-to-one relationship between a policy and the principal identity that it's applied to
    - For example, if we want to be sure that the permissions in the policy are not inadvertently assigned to a principal entity other than the one, they're intended for
- IAM Policies Structure explained in below diagram



- NotAction with Allow – provide access to all actions in an AWS service, except for the action specified in NotAction

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

- Use with the Resource element to provide scope for the policy, limiting the allowed actions to the actions that can be performed on the specified resource

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

- Use the NotAction element in statement with 'Deny' denying access to all listed resources except for the actions specified in the NotAction element
- This combination does not allow the listed items, but instead explicitly denies the actions not listed
- The actions that are desired to be allowed still need to be explicitly allowed

- Like below policy - this policy grants unrestricted access to users with MFA enabled, allowing them to perform any action, while users without MFA are restricted from performing any AWS actions except IAM-related actions

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

- Restrict to one region (Not Action) – deny everything outside of specific region
    - Like below policy denies access to all actions in AWS services except for "cloudfront," "iam," "route53," and "support." The denial is enforced for resources in the AWS account if the requested region is not "eu-central-1." However, if the requested region is "eu-central-1," all actions in all AWS services, including "cloudfront," "iam," "route53," and "support," are allowed for the resources within the AWS account

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

        - This IAM policy is designed to deny certain actions in AWS services
        - All actions in the "cloudfront," "iam," "route53," and "support" services are explicitly allowed (NotAction is used, so these actions are not affected by the policy)
        - The policy applies to all resources within the AWS account (Resource: "*")
        - The policy has a condition that checks the requested AWS region
        - If the requested region is NOT "eu-central-1," then all actions in all other AWS services (except "cloudfront," "iam," "route53," and "support") will be denied for those resources in the AWS account

- However, if the requested region is "eu-central-1," then all actions in all AWS services (including "cloudfront," "iam," "route53," and "support") will be allowed for those resources in the AWS account

- Deny everything outside of eu-central-1 except S3 – below policy

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

- Action/Not Action and Allow/Deny are explained below:

	Allow	Deny
Action	Allow IAM `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": ["iam:*"], "Resource": "*" } ] }`	Deny IAM `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Deny", "Action": ["iam:*"], "Resource": "*" } ] }`
NotAction	Allow anything not IAM `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "NotAction": ["iam:*"], "Resource": "*" } ] }`	Deny anything not IAM `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Deny", "NotAction": ["iam:*"], "Resource": "*" } ] }`

- **Principle Options** in IAM Policies – explained below with examples
  - **AWS Account and Root User**

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

  - **IAM Roles**

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

- **IAM Role Sessions**

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

- **IAM Users**

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

- **Federated User Sessions**

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

- **AWS Services**

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

- **All Principals**

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

- IAM Conditions – **Condition Operators**
  - *StringEquals* /*StringNotEquals* – case sensitive, exact matching

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

  - *StringLike* /*StringNotLike* – case sensitive, optional partial matching using .*?

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

    - Policy variables let you specify placeholders in a policy
    - When the policy is evaluated, the policy variables are replaced with values that come from the context of the request itself
    - When this policy is evaluated, IAM replaces the variable *${aws:username}* with the friendly name of the actual current user
    - This means that a single policy applied to a group of users can control access to a bucket – It does this by using the username as part of the resource's name.

- *DateEquals* /*DateLessThan* …

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

- *ArnLike* / *ArnNotLike* – used specifically for ARN
- *Bool* – check for Boolean value

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

- *IpAddress* /*NotIpAddress* -CIDR format
  - Resolves to the IP address that request originates from
  - Public IP only – IpAddress does not apply to request through VPC endpoints

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

- IAM Conditions – **RequestedRegion**
  - The AWS region of the request
  - Used to restrict specific actions in specific AWS regions

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

  - this IAM policy allows all EC2 actions for resources in AWS regions other than "eu-west-1," "eu-west-2," and "eu-west-3."
  - However, it denies "Stop" and "Terminate" actions for resources in the mentioned AWS regions

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

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

  - this policy denies access to most AWS actions, but it allows specific actions related to "cloudfront," "iam," "route53," and "support"
  - However, these allowed actions will only be permitted if the user makes the request from one of the specified AWS regions ("eu-west-1," "eu-west-2," or "eu-west-3")
  - For all other regions, most AWS actions will be denied

- IAM Conditions – **PrincipalArn**
  - Compare the ARN of the principal that made the request with the ARN specified in the policy
  - For IAM roles the request context returns the ARN of the role, not the ARN of the user that assumed the role


```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::my-bucket",
            "Condition": {
                "StringLike": {
                    "aws:PrincipalArn": "arn:aws:
iam::123456789012:user/Alice"
                }
            }
        }
    ]
}
```

  - this IAM policy permits "Alice" (with the specified ARN) to list the contents of the S3 bucket named "my-bucket" and denies this action for all other users

- IAM Conditions – **SourceArn**
  - Compare the ARN of the source making a service-to-service request with the ARN specified in the policy
  - This key is included in the request context only if accessing a resource triggers an AWS service to call another service on behalf of the resource owner
    - For example – an S3 bucket update triggers an SNS topic *sns:Publish* API, the policy set the value of the condition key to the ARN of the S3 bucket


```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "sns:Publish",
            "Resource": "arn:aws:sns:us-east-1:12
3456789012:MyTopic",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:
*:123456789012:my-bucket"
                }
            }
        }
    ]
}
```

- this IAM policy permits any AWS service to publish messages to the SNS topic "MyTopic" located in the "us-east-1" region with the account number "123456789012"
- However, it restricts this permission based on the condition that the calling AWS service's ARN must match the pattern "arn:aws:s3:*:123456789012:my-bucket"

- IAM Conditions – **CalledVia**
  - Look at the AWS service that made the request on behalf of the IAM User or Role
  - Contains the ordered list of each AWS service in the chain that made requests on the principal's behalf
  - Supports
    - Athena, CloudFormation, DynamoDB, KMS


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

- this IAM policy denies access to any "s3:Get" actions for all resources in the S3 bucket "bucket" unless the request is made via Amazon Athena
- Requests from Amazon Athena will be allowed to perform "s3:Get" actions on the specified bucket and its contents, while all other requests will be denied

- IAM Conditions – **IP & VPC Conditions**

Restrict Access from Public IP Addresses

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

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

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

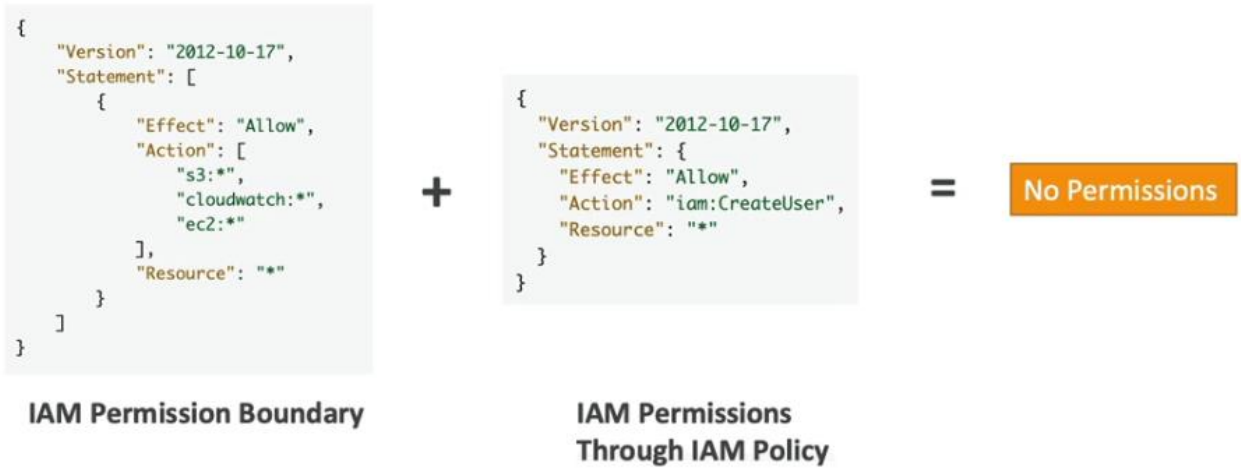
- the second IAM policy denies access to all AWS actions for all resources, but it allows access if the request originates from IP addresses within the CIDR blocks "192.0.2.0/24" or "198.51.100.0/24"
- Requests from any other IP addresses will be denied for all AWS actions

- IAM Conditions – **ResourceTag & PrincipalTag**
  - Controls access to AWS resources using tags
  - *aws:ResourceTag* – tags that exist on AWS resources – can be service-specific tags for example *ec2:ResourceTag*
  - *aws:PrincipalTag* – tags that exist on the IAM user or IAM role making the request

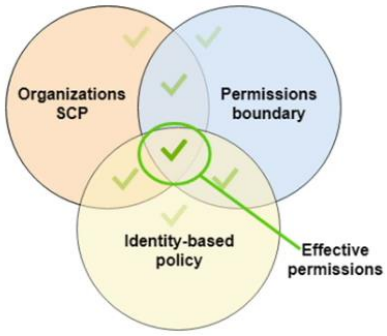
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ec2:startInstances", "ec2:StopInstances"],
            "Resource": "arn:aws:ec2:us-east-1:123456789012:instance/*",
            "Condition": {
                "StringEquals": {
    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

- this IAM policy permits users with the tag "Department: Data" to start and stop EC2 instances that have the tag "Project: DataAnalytics" in the AWS account's "us-east-1" region
- Users without this tag combination or trying to perform these actions on instances without the specified tag will be denied access

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



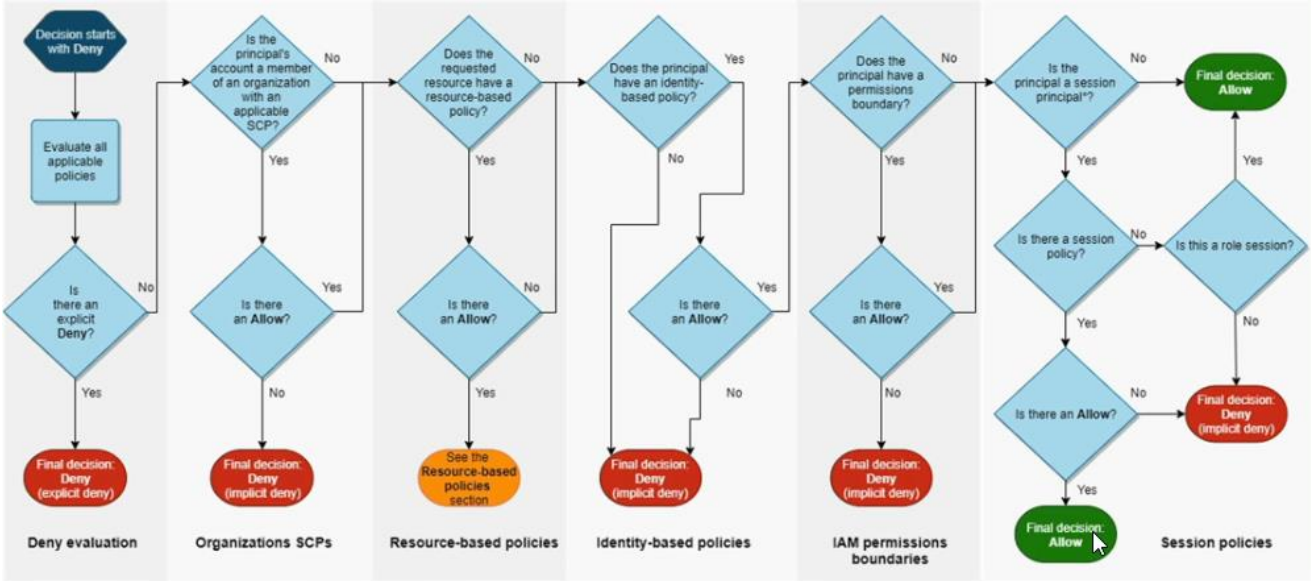
- Can be used in combinations of AWS Organization SCP



- Use cases could be as below
  - delegate responsibilities to non-administrators within their permission boundaries, for example create new IAM users
  - allow developers to self-assign policies and manage their own permissions, while making sure they can't escalate their privileges (make themselves admin)
  - useful to restrict one specific user – instead of a whole account using Organization and SCP

- IAM Policy – Simplified Evaluation Logic (Allow/Deny)
  1. By default, all requests are implicitly denied except for the AWS account root user, which has full access
  2. An explicit allow in an identity-based or resource-based policy overrides the default (1)
  3. If a permissions boundary, Organizations SCP, or session policy is present, an explicit allow is used to limit actions – anything not explicitly allowed is an implicit deny and may override the decision 2
  4. An explicit deny in any policy override all allows
  5. Difference b/w implicit deny/allow and explicit deny/allow:
     - Implicit deny/allow refers to the default behavior of access control, where access is denied unless explicitly allowed or vice versa – explicit deny/allow, on

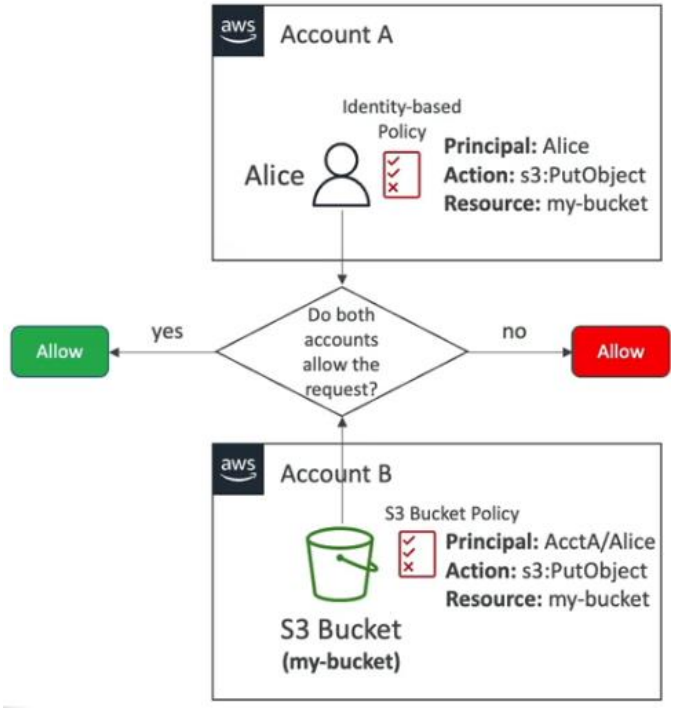
the other hand, involves explicitly configuring rules to either deny or allow access, overriding the default behavior



- Examples – to understand the policy evaluation logic
  - Can user perform *sqs:CreateQueue* in below policy?
    - Answer – NO (not explicitly allowed, all the SQS actions are denied and only SQS Delete allowed in policy explicitly)
  - Can user perform *sqs:DeleteQueue* in below policy?
    - Answer – NO (explicitly allowed but also explicit deny override it)
  - Can user perform *ec2:DescribeInstances*?
    - Answer – NO (not explicitly allowed, hence implicitly denied)

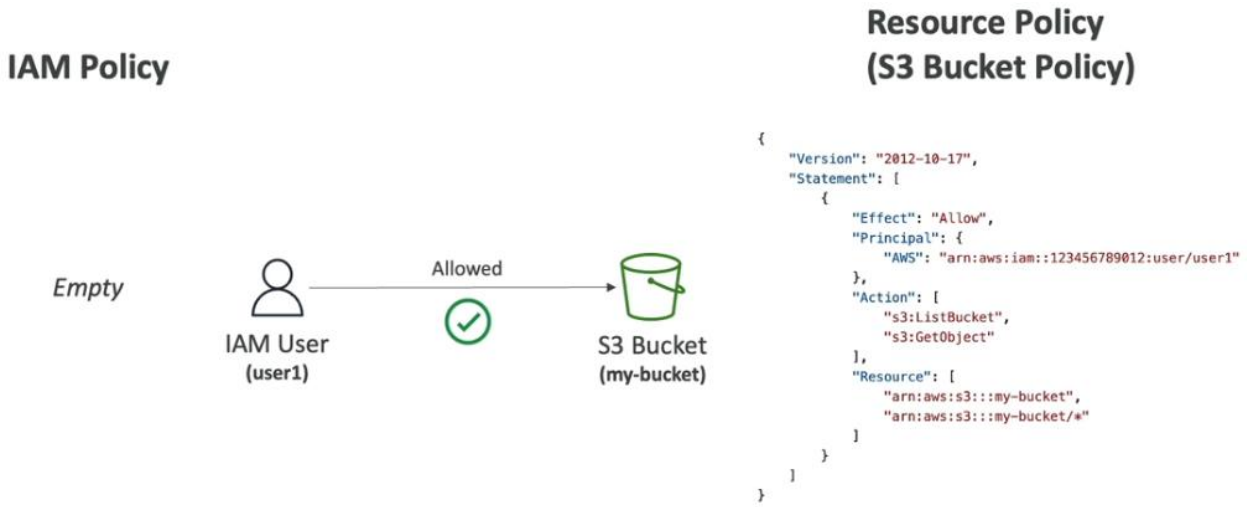
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "sqs:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "sqs:DeleteQueue",
            "Resource": "*"
        }
    ]
}
```

- Cross-Account Policy Evaluation Logic – explained in below example



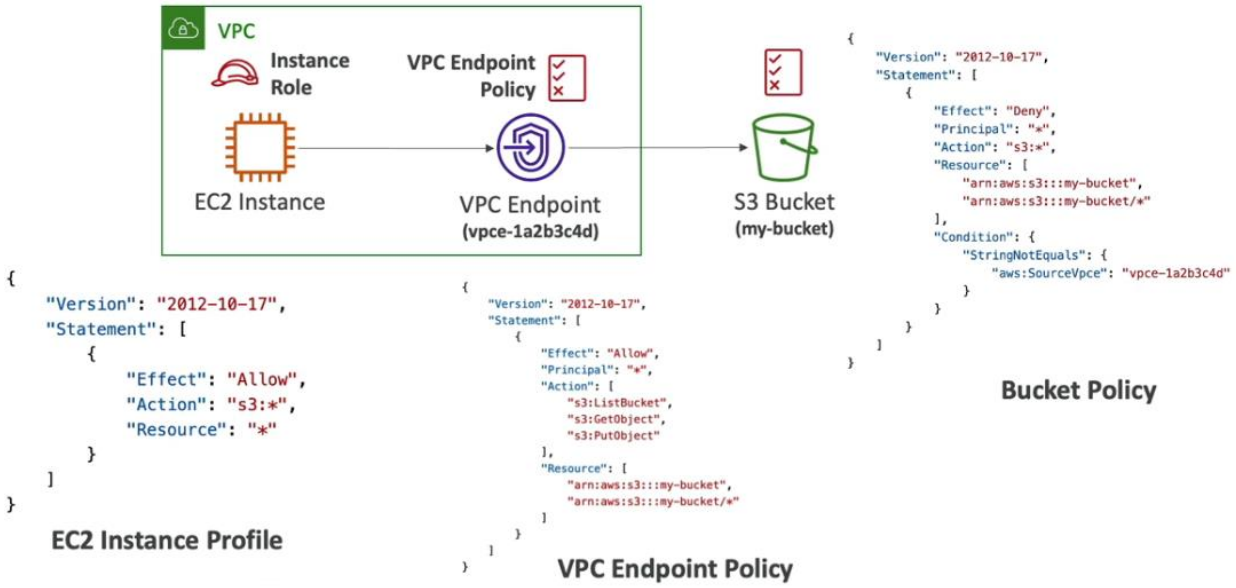
- Request from Account A to Account B
- The requester in Account A must have an identity-based policy
- That policy must allow the requester to make a request to the resource in Account B
- The resource-based policy in Account B must allow the requester in Account A to access the resource

- IAM Policy + Resource Policy – Example explained below

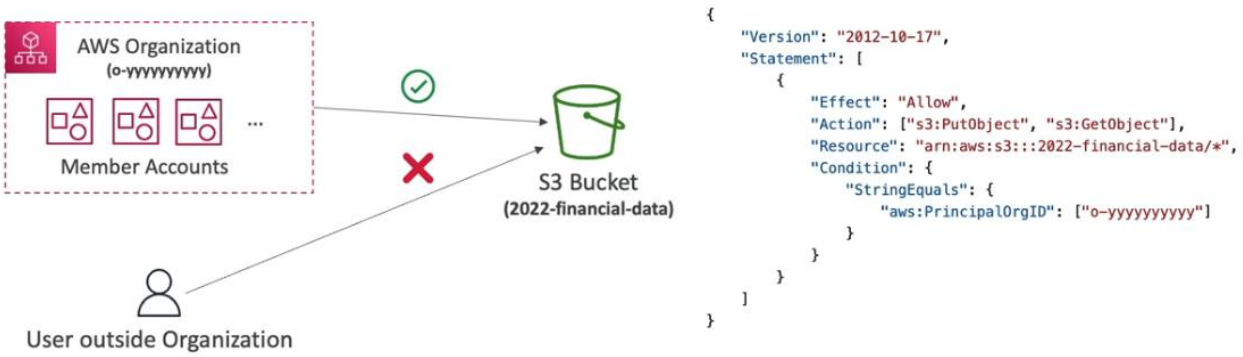


- IAM Policy + VPC Endpoint Policy + Resource Policy – Example explained below
  - IAM Policy allows all the actions on all S3 resources from any Principal
  - VPC Endpoint Policy allows all the principals to perform specific actions on specific S3 buckets (all objects)

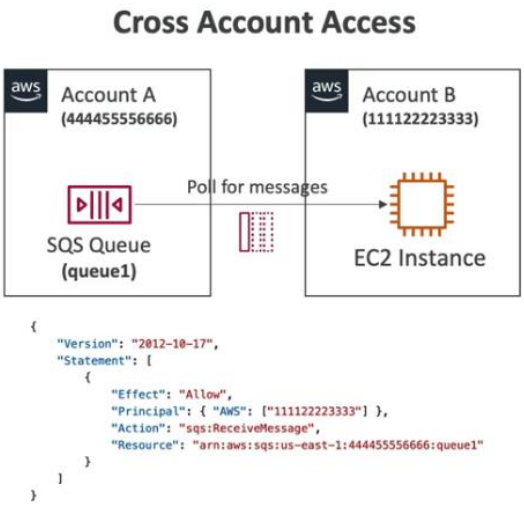
- Bucket Policy explicitly denies all principals to perform all the actions on the specified S3 buckets (all objects) except the ones coming through the defined VPC endpoint
- So, at the end this instance role will have the list, get, and put object permissions on the S3 bucket my-bucket (all objects) if it comes through the VPC endpoint vpce-1a2b3c4d

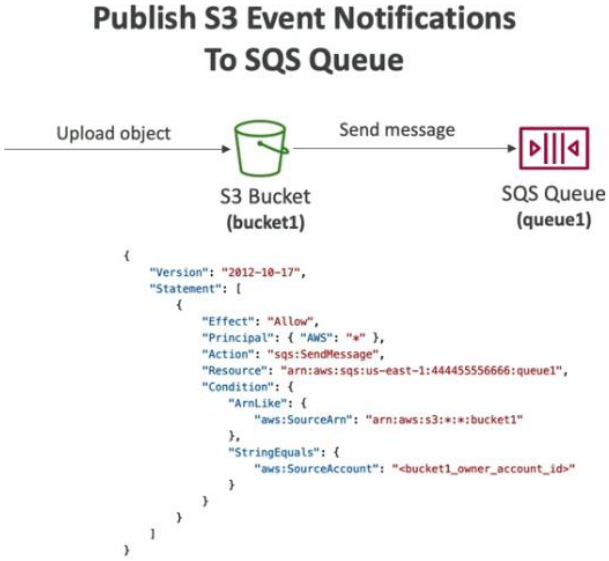


- IAM Role vs Resource Based Policies:
  - **IAM Roles** – helpful to get temporary permissions for a specific task
    - Allow a user/application to perform many actions in a different account
    - Permissions expire over time
  - **Resource-based policies** – used to control access to specific resources (resource-centric view)
    - Allow cross-account access
    - Permanent authorization – as long as it exists in the resource-based policy
  - Resource policies and *aws:PrincipalOrgID*
    - *aws:PrincipalOrgID* can be used in any resource policies to restrict access to accounts that are member of an AWS Organization



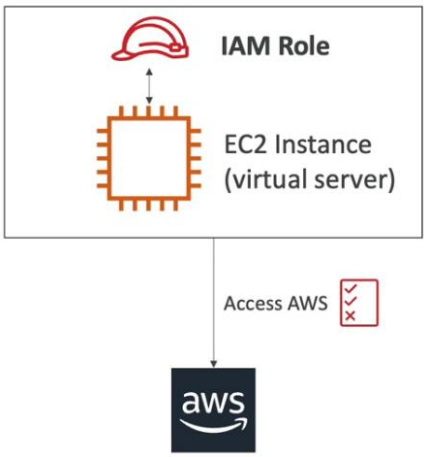
  - Another example – SQS queue access policy





IAM Roles:

- Secure and reusable AWS identity that can be assigned to AWS resources
- Not tied to a specific user or group, but can be assumed by trusted entities such as IAM users, AWS services, or external identities (federated users)
- When the customer assumes the IAM role, the customer temporarily becomes an identity in the production account
- Define the permissions and policies that determine what actions can be performed on AWS resources
- Can be used to grant temporary access to resources, allowing fine-grained control over permissions
    - So provides a way to delegate access to AWS resources without the need for sharing long-term security credentials
- Support cross-account access – user from one AWS account to assume role in another AWS account
- Can have trusted policies that defines which entities are allowed to assume the role
- IAM Roles for Services – some AWS services will need to perform action on our behalf
    - To do so, we will assign permission to AWS services with IAM Roles
    - Common Roles:
        - EC2 instance roles
        - Lambda function roles
        - Roles for CloudFormation



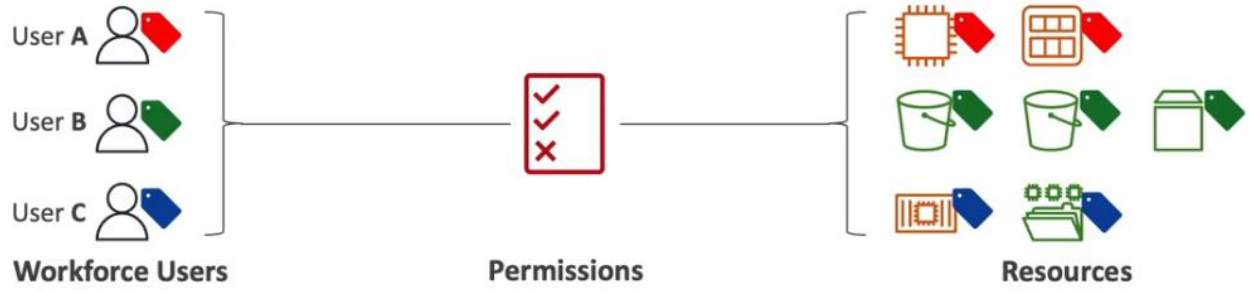
- Delegate passing permissions to AWS services – users can be granted permissions to pass an IAM role to an AWS service
    - Ensure that only approved users can configure an AWS service with an IAM role that grant permissions
    - Grant *iam:PassRole* permission to the user's IAM user, role, or group
    - *PassRole* is not an API call – no CloudTrail logs generated

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

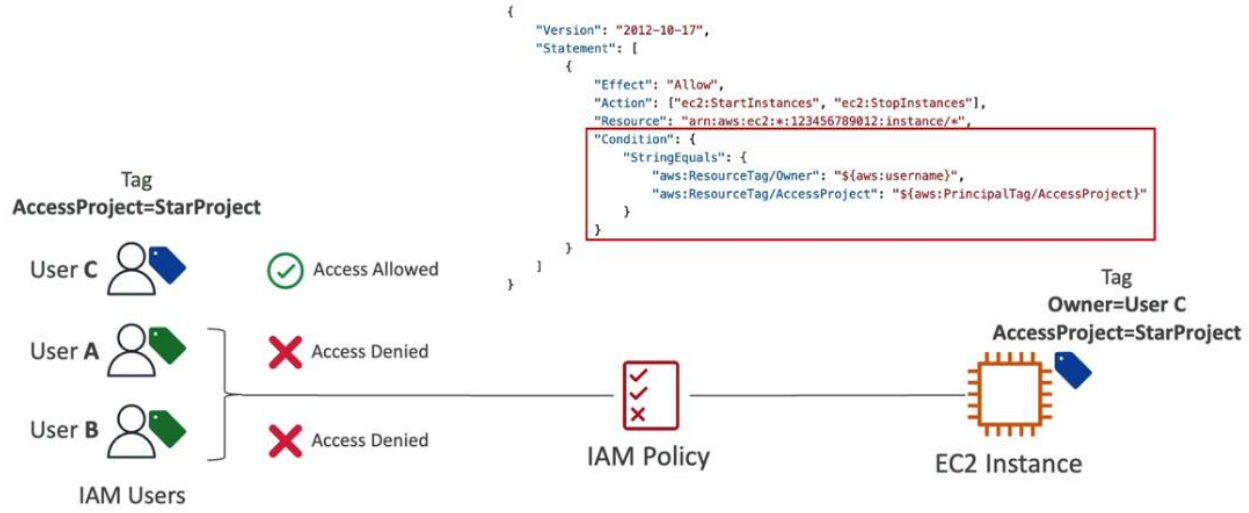
Allow User to Pass Only Approved Roles

Attribute-Based Access Control (ABAC):

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



- Example – User C tag and EC2 Instance tag should be matched for the access to be allowed



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

IAM MFA:

- Can force users to use MFA before doing important operations on S3
  - For example – MFA will be required to
    - Permanently delete an object version

        - Suspend versioning on the bucket
    - MFA won't be required to
        - Enable versioning
        - List deleted versions
    - To use MFA Delete, versioning must be enabled on the bucket
    - Only the bucket owner (root account) can enable/disable MFA delete
- IAM Conditions – MultiFactorAuthPresent
    - Restrict access to AWS services for users not authenticated using MFA
    - Compatible with AWS Console and AWS CLI

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

IAM Credential Report:

- Lists all users in the account and the status of their various credentials, including passwords, access keys and MFA devices. For example:

user	arn	user_creation_time	password_enabled	password_last_used	password_last_changed	password_next_rotation	mfa_active
<root_a	arn:aws:iam::7	2018-02-28T1	not_suppo	2020-02-12	not_supported	not_supported	TRUE
ryan	arn:aws:iam::7	2019-05-03T1	FALSE	N/A	N/A	N/A	TRUE
erika	arn:aws:iam::7	2019-05-13T0	FALSE	N/A	N/A	N/A	TRUE
faye	arn:aws:iam::7	2018-05-15T1	TRUE	2020-04-02	2019-01-29T11:	N/A	TRUE
allan	arn:aws:iam::7	2020-02-13T1	FALSE	N/A	N/A	N/A	TRUE
sam	arn:aws:iam::7	2019-05-30T1	FALSE	N/A	N/A	N/A	TRUE
julian	arn:aws:iam::7	2019-06-28T1	FALSE	N/A	N/A	N/A	TRUE
nicola	arn:aws:iam::7	2020-03-05T1	FALSE	N/A	N/A	N/A	TRUE

- This report can be downloaded from *IAM > Credential Report* option or from CLI via below commands and for that *GenerateCredentialReport* and *GetCredentialReport* permissions will be required

```
#To generate a credential report:
 aws iam generate-credential-report

#To download a credential report:
aws iam get-credential-report
```

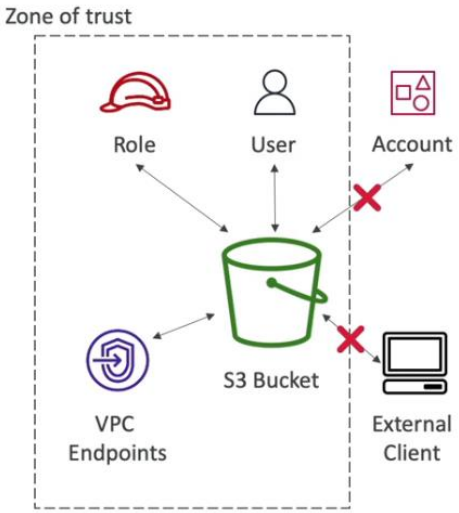
- Can be generated as often as once every 4 hours

IAM Access Advisor:

- Shows the service permissions granted to a user and when those services were last accessed – provides insights into the access permissions of IAM users and roles
- Helps to identify unused or excessive permissions granted to IAM entities
- Analyzes the usage patterns of IAM entities and provides recommendations for removing unnecessary permissions – shows the last accessed timestamp for each service and the associated permissions
- Users can use this information to revise their policies – provides visibility into the least recently used services
- Supports both user-level and role-level access analysis

IAM Access Analyzer:

- Analyze and identify unintended access to resources
- Automatically reviews resource policies and IAM policies to detect potential security risks
- Uses automated reasoning algorithms to identify resource access patterns and generate actionable findings
    - Define Zone of Trust = AWS Account or AWS Organization
    - Access outside zone of trusts >> findings

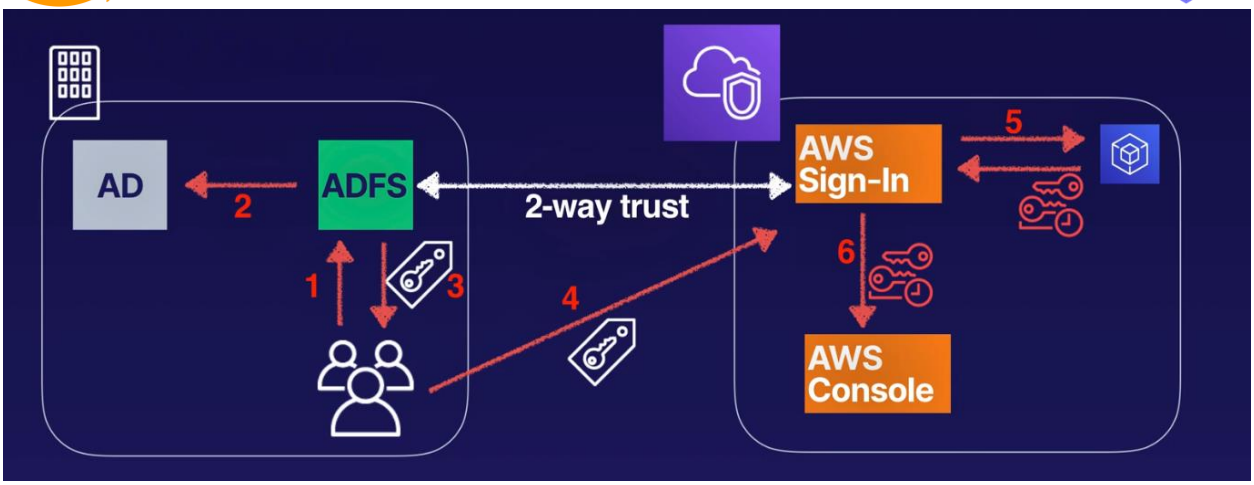


- Helps in identifying resources that can be accessed publicly or unintended principals

    - S3 buckets, IAM Roles, KMS Keys, Lambda Functions and Layers, SQS queues, Secret Manager Secrets
- It does not identify access key IDs that may have been compromised
- Supports both proactive and reactive analysis
- Provides detailed findings that highlight the access paths and potential vulnerabilities

AD Federation:

- Enhances security by centralizing authentication and access control through active directory
- Reduces the risk of password-related vulnerabilities by eliminating the need for users to remember and manage multiple passwords
- Provides a secure and centralized mechanism for exchanging authentication and authorization information between systems
- The use of security standards like SAML or OpenID Connect ensures the integrity and confidentiality of the exchanged tokens
- Trust relationships are established and validated, ensuring that only trusted entities can participate in the federation process
- Allows for fine-grained access control, ensuring that users only have access to authorized resources and applications
- Enables organizations to enforce consistent security policies and access controls across federated systems
- Supports MFA, adding an additional layer of security for user authentication
- Auditing and monitoring capabilities can be implemented to track authentication events and detect suspicious activities
- By centralizing authentication, AD Federation provides better visibility and control over user access, enabling efficient security management and compliance enforcement
- Provides SSO for users
- ADFS – Active Directory Federation Service
    - Provides SSO and identity federation capabilities, allowing users to access multiple applications and systems using their AD credentials
- SAML – Security Assertion Markup Language
    - Standard for exchanging authentication and authorization data between identity providers and service providers in a secure and interoperable manner
    - Commonly used with SSO – where users can authenticate once with an identity provider and gain access to multiple service providers without needing separate logins



1. Corporate users access the corporate ADFS portal sign-in and provides their AD username and password
2. ADFS authenticates the user against AD
3. AD returns user's information including group membership
4. ADFS sends a SAML token to the user's browser which sends the token to AWS sign-in endpoint
5. The AWS sign-in endpoint makes an STS AssumeRoleWithSAML request and STS returns temporary credentials
6. User is authenticated and allowed to access the AWS management console

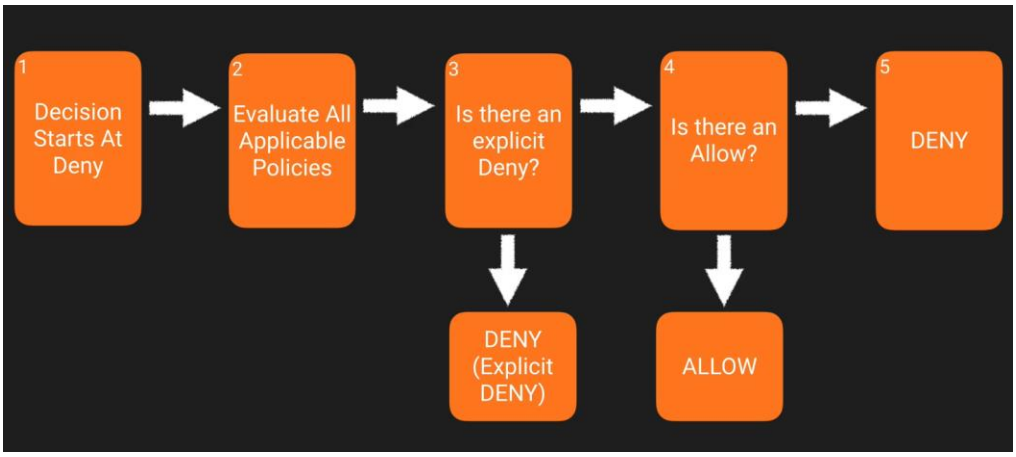
- ADFS basically acts as identity broker between AWS and user's AD
- AD users can assume roles in AWS based on group membership in AD

AWS S3:

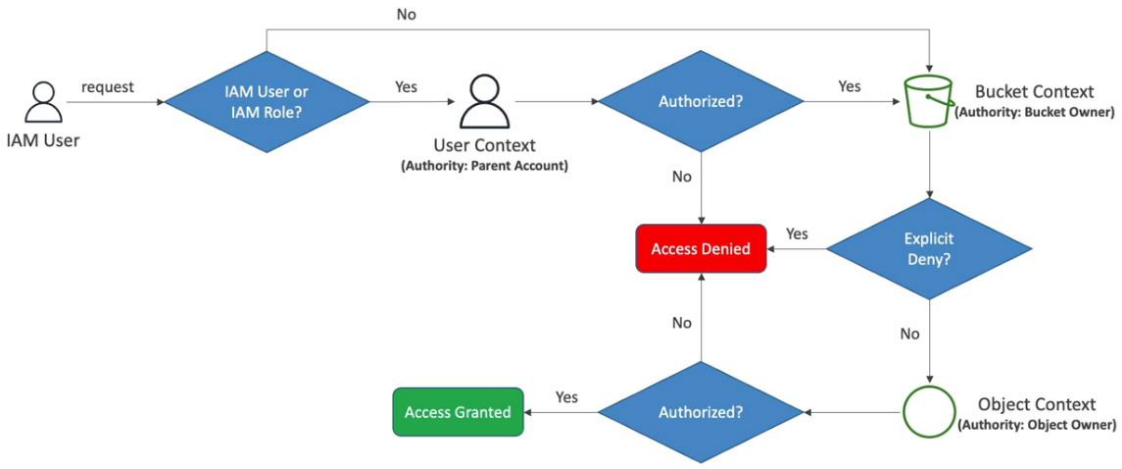
- Whenever an AWS principal (user, group, or role) issues a request to S3, the authorization decision depends on the union of all the IAM policies, S3 bucket policies, and S3 ACLs that apply
- With least privilege, decisions always default to DENY
- Also, an explicit DENY trumps an ALLOW
- If we DENY access to something somewhere and then something else allows access, the DENY will override the ALLOW

How Authorization Works in Amazon S3?

- If an IAM policy grants access to an object, the S3 bucket polices denies access to that object, and there is no S3 ACL, then access will be denied
- Similarly, if no method specifies an ALLOW, then request will be denied by default
- Only if no method specifies a DENY and one or more methods specify an ALLOW will the request be allowed



- User Context
  - Is the IAM principal authorized by the parent AWS account? – IAM Policy
  - If the parent owns the bucket or object, then bucket policy/ACL or object ACL is evaluated
  - If the parent owns the bucket/object, it can grant permissions to its IAM principal using identity-based policy or resource-based policy
- Bucket Context
  - Evaluates the policies of the AWS account that owns the bucket – check for Explicit Deny
- Object Context
  - Requester must have permissions from the object owner – using Object ACL
  - If bucket owner = object owner, then access granted using Bucket Policy
  - If bucket owner != object owner, then access granted using object owner ACL
  - If users want to own all object in their bucket and only use Bucket Policy and IAM-Based Policy to grant access, enable Bucket Owner Enforced Setting for Object Ownership
    - Then bucket and objects ACLs can't be edited and are no longer considered for access



- Bucket Operations Vs Object Operations
  - Bucket level permissions

- *s3:ListBucket*
- Object level permissions
  - *s3:GetObject*
  - *s3:PutObject*
  - *s3:DeleteObject*

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

S3 Bucket Policies:

- Are attached only to S3 buckets
- S3 buckets are private by default unless allow policies are defined
- S3 bucket policies specify what actions are allowed or denied on the bucket
- Resource-based policies that can be used to grant access permissions to the bucket and objects in it
- To enable an AWS service to invoke function, user can grant permission in a statement on a resource-based policy - user can apply the statement to the entire function or limit the statement to a single version or alias - the execution role defines which resources user's Lambda function has access to
- Only the bucket owner can associate a policy with a bucket
- The permissions attached to a bucket apply to all the objects in the bucket that are owned by the bucket owner – these permissions do not apply to objects owned by other AWS accounts
- By default, when another AWS account uploads an object to your S3 bucket, that account (the object writer) owns the object, has access to it, and can grant other users access to it through ACLs
- They can be broken down to user level
- IAM Policies and Bucket Policies work together in combination to determine who or what can access an S3 bucket and what actions they are allowed to take

S3 Bucket Policies Use Cases:

- Simple way to grant cross-account access to S3 environment, without using IAM roles

- IAM policies bump up against the size limit (up to 2kb for users, 5kb for groups, and 10kb for roles). S3 supports bucket policies up to 20kb.
- If preference is to keep access control policies in S3 environment

- Example1 – granting permissions to multiple accounts with added conditions

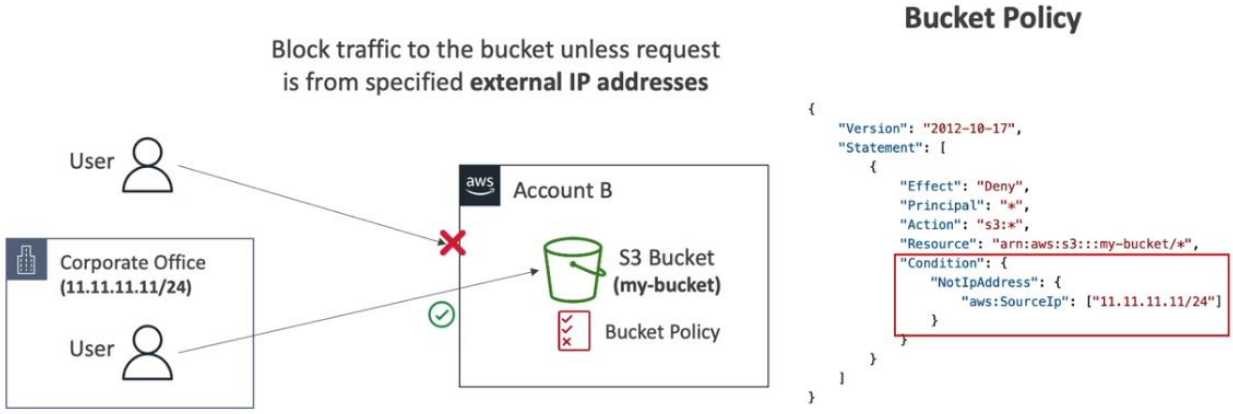
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AddCannedAcl",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:root",
                    "arn:aws:iam::444455556666:root"
                ]
            },
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": [
                        "public-read"
                    ]
                }
            }
        }
    ]
}
```

- this IAM policy permits the two specified AWS root users to upload objects and modify object ACLs within the S3 bucket named "DOC-Example-BUCKET," but only if the objects have their ACLs set to "public-read

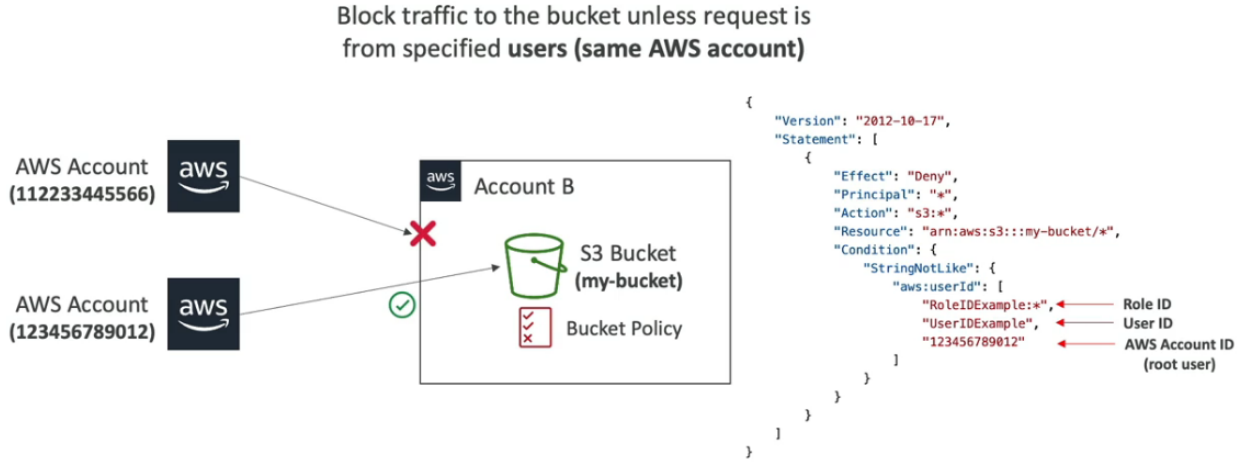
- Example2 – granting read only permissions to anonymous user

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicRead",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
            ]
        }
    ]
}
```

- Example3 – limiting access to specific public IP address



- Example4 – restricting by user ID



- Example5 – restricting access to a specific HTTP referrer

```
{
  "Version":"2012-10-17",
  "Id":"HTTP referer policy example",
  "Statement":[
    {
      "Sid":"Allow GET requests originating from www.example.com and example.com.",
      "Effect":"Allow",
      "Principal":"*",
      "Action":["s3:GetObject","s3:GetObjectVersion"],
      "Resource":"arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
      "Condition":{
        "StringLike":{"aws:Referer":["http://www.example.com/*","http://example.com/*"]}
      }
    }
  ]
}
```

  - this IAM policy allows public access to retrieve objects and their versions from the S3 bucket named "DOC-Example-Bucket"
  - However, access is only permitted if the request includes an HTTP Referrer header that matches either "http://www.example.com/" or "http://example.com/"

- This policy effectively enables the bucket to be accessed by specific web pages hosted on the mentioned domains while blocking direct access or access from other origins

- Example6 – adding a bucket policy to require MFA

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

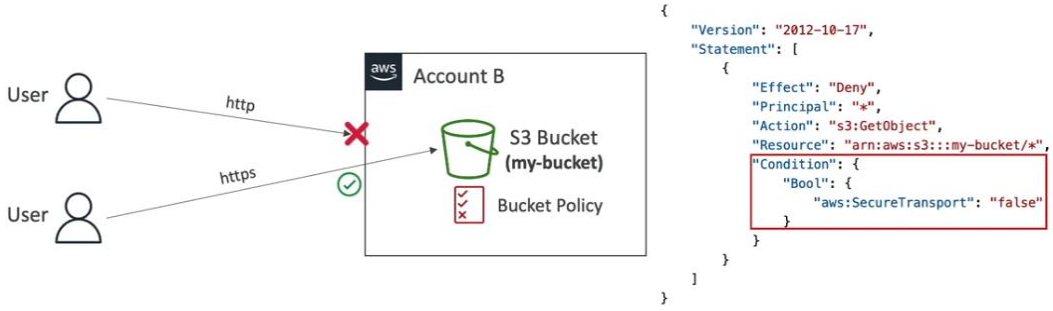
  - this IAM policy denies access to perform any S3 action on any object within the "taxdocuments" folder in the "DOC-EXAMPLE-BUCKET" S3 bucket, but it allows access if the user has authenticated with MFA (Multi-Factor Authentication) for the request
  - If MFA is not used or available, the request will be denied for any S3 action on objects within the specified folder

S3 ACLs:

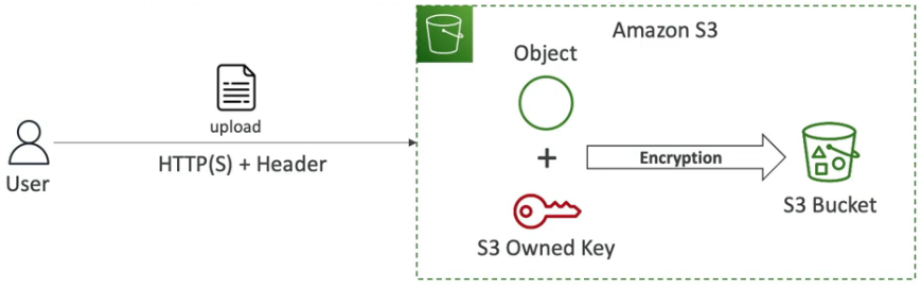
- S3 ACLs are a legacy access control mechanism that predates IAM
- AWS recommends to S3 bucket policies and IAM policies
- Bucket policies can only be applied at the bucket level where as S3 ACLs can be applied to individual files (objects)
- S3 ACLs Use Cases:
  - For fine grained permissions on individual files/objects within S3
  - Bucket policies are limited to 20kb in size, so in case bucket policy grows too large, S3 ACLs can be used

Encryption on S3:

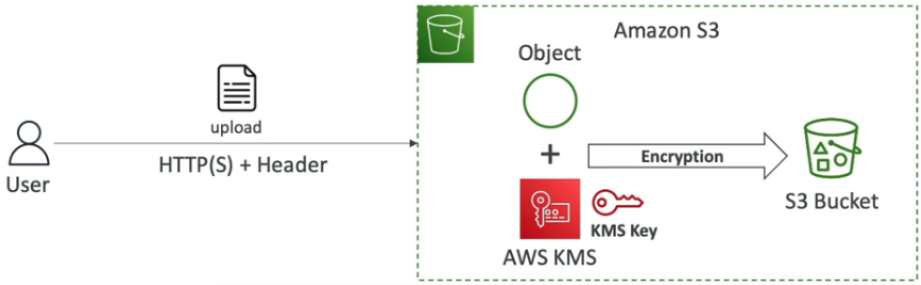
- Encryption in transit – encryption in flight (SSL/TLS)
  - S3 exposes two endpoints
    - HTTP Endpoint – non encrypted
    - HTTPS Endpoint – encryption in flight
  - So, HTTPS is recommended when uploading to S3
  - HTTPS is mandatory for SSE-C
- Encryption can be forced via bucket polices
- Like in below policy, get object for the specific bucket is allowed to all AWS resources while after the allow section, it's denied with a condition if it's not using secure transport using Bool statement
  - So, it will only allow if secure transport protocol is being used, otherwise it will deny access to it



- Object Encryption – can encrypt objects in S3 buckets using one of 4 methods
    - **Server-Side Encryption** – **SSE**
        - With Amazon S3 Managed Keys – **SSE-S3** – enabled by default
            - Encrypt S3 objects using keys handled, managed, and owned by AWS
            - Object is encrypted at server side
            - Encryption type is AES-256
            - Must set header *"x-amz-server-side-encryption": "AES256"*

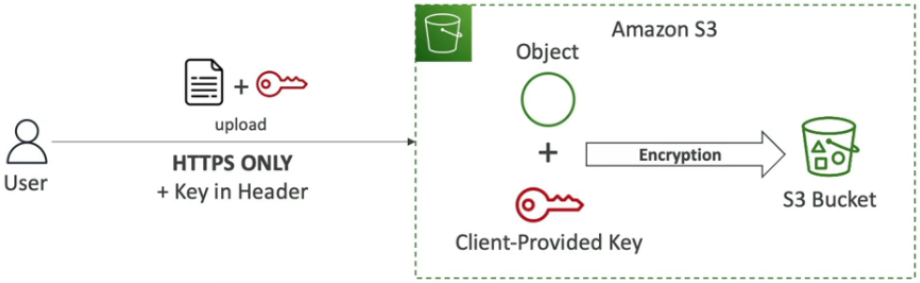


            - Objects encrypted with SSE-S3 (default master key) is automatically decrypted when the requester has access to the object
            - The default master key is AWS managed, and you cannot share the default key with other accounts
        - With KMS Keys stored in AWS KMS – **SSE-KMS**
            - Encryption using keys handled and managed by AWS KMS
            - User control + audit key usage CloudTrail
            - Object is encrypted at server side
            - May be impacted by the KMS limits
            - Objects made public can never be read
            - When uploading, it calls the *GenerateDataKey* KMS API
            - When downloading, it calls the *Decrypt* API call
            - On *s3:PutObject*, make the permissions *kms:GenerateDatakey* is allowed
            - Must set the header *"x-amz-server-side-encryption": "aws:kms"*

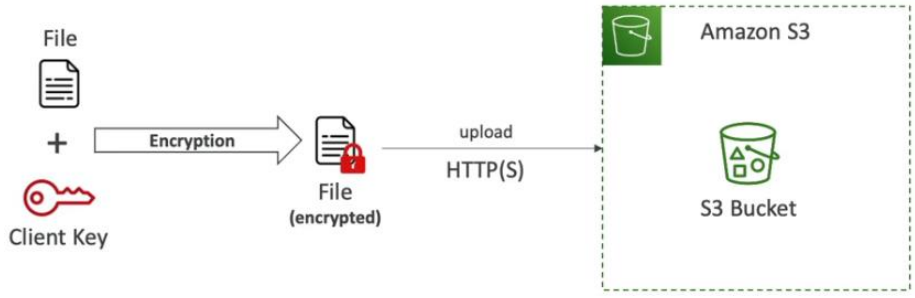


- Objects encrypted with SSE-KMS can be accessed from another account provided the requester account has permission to read the object and permission to use the master key for decrypt operation

- With Custom Provided Keys – **SSE-C**
  - Server-side encryption using keys fully managed by the customer outside of AWS
  - S3 does not store the encryption keys being provided by customer
  - HTTPS must be used
  - Encryption key must be provided in HTTP header, for every HTTP request made



- **Client-Side Encryption** – **CSE**
  - Use client libraries such as S3 Client-Side encryption library
  - Clients must encrypt data themselves before sending to S3
  - Clients must decrypt data themselves when retrieving from S3
  - Customer fully manages the keys and encryption cycle



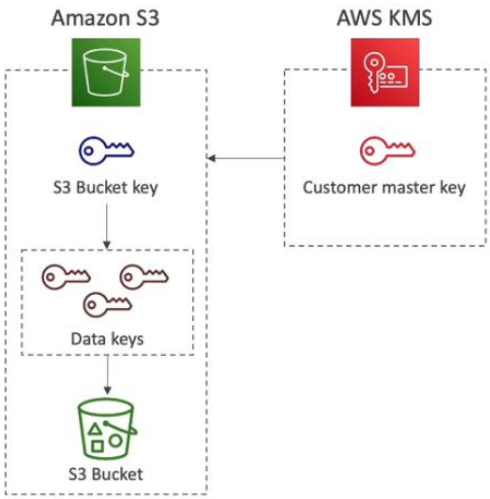
- To decrypt an S3 object, the encrypted data key must be decrypted first using the master key
  - If the master key is not available, the data key cannot be decrypted

- So, the object is no longer accessible
- SSE-S3 encryption is automatically applied to new objects stored in S3 bucket
- Optionally, we can force encryption using bucket policy and refuse any API call to PUT an S3 object without encryption header – SSE-KMS or SSE-C

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

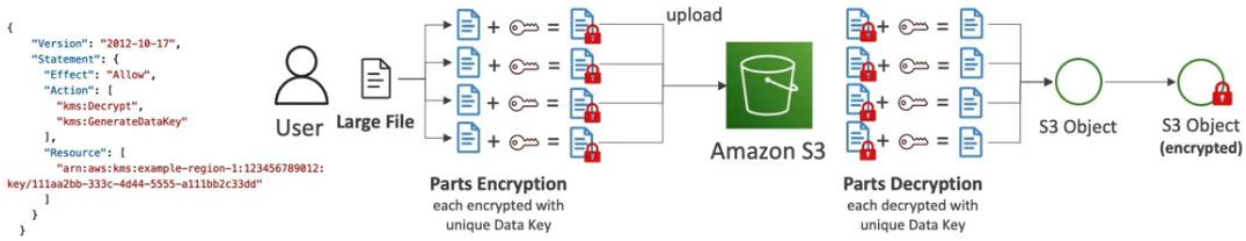
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "s3:PutObject",
            "Principal": "*",
            "Resource": "arn:aws:s3:::my-bucket/*",
            "Condition": {
                "Null": {
                    "s3:x-amz-server-side-encryption-customer-algorithm":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

  - This second IAM policy denies access to upload objects to the "my-bucket" S3 bucket for all AWS principals
  - However, there is an exception: If the user specifies customer-provided server-side encryption (indicated by "s3:x-amz-server-side-encryption-customer-algorithm" being "true"), the denial is not applied, and the user can proceed with the upload
- Bucket policies are always evaluated before Default Encryption
- S3 Bucket Key for SSE-KMS encryption – new settings
  - To decrease no. of API calls made to KMS from S3 by 99%
  - To decrease cost of overall KMS encryption with S3 by 99%
  - Less KMS CloudTrail events
  - This leverage Data Keys
    - A S3 bucket key is generated
    - That key is used to encrypt KMS objects with new data keys

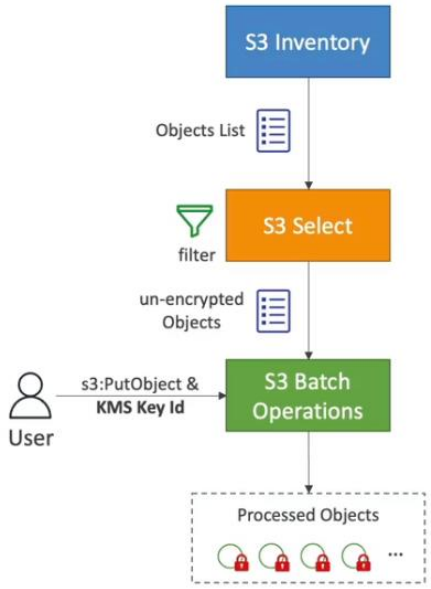


- Large File Upload to S3 with KMS Key – user will have to use S3 multi-part upload
  - Must have the following permissions to the KMS Key

- *kms:GenerateDataKey* – allows to encrypt each object part with a unique data key
- *kms:Decrypt* – decrypt object parts before they can be assembled, then re-encrypt them with the KMS key

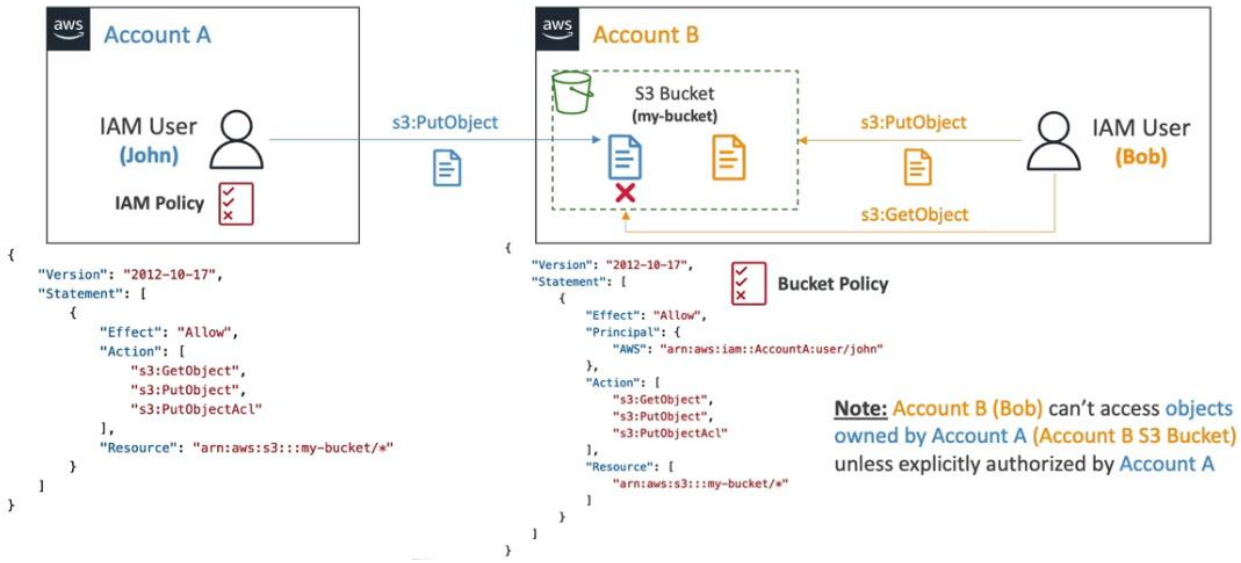


- To encrypt an object at the time of upload, you need to add a header called *x-amz-server-side-encryption* to the request to tell S3 to encrypt the object using SSE-C, SSE-S3, or SSE-KMS.
  - To enforce object encryption, create an S3 bucket policy that denies any S3 Put request that does not include the *x-amz-server-side-encryption* header.
  - This enforces encryption at rest.
- When you use server-side encryption with AWS KMS (SSE-KMS), you can use the default AWS managed key, or you can specify a customer managed key that you have already created
  - The data keys used to encrypt your data are also encrypted and stored alongside the data that they protect
  - If you don't specify a customer managed key, Amazon S3 automatically creates an AWS KMS key in your AWS account the first time that you add an object encrypted with SSE-KMS to a bucket
  - By default, Amazon S3 uses this KMS key for SSE-KMS
  - If you want to use a customer managed key for SSE-KMS, create the customer managed key before you configure SSE-KMS
    - Then, when you configure SSE-KMS for your bucket, specify the existing customer managed key
- With SSE-KMS encryption, you have fine-grained access control as the requester also needs permission to use the key in addition to the object access
  - Using SSE-KMS will protect your objects from unauthorized access if the bucket is accidentally made public
  - You can also use Client-side encryption where the object is encrypted in your application, and the encrypted object is stored as-is in S3
  - The object is decrypted by your application when needed
- S3 Batch – perform bulk operations on existing S3 objects with a single request (encrypt un-encrypt objects etc.)
  - S3 Inventory – to get the list of all objects and its associated metadata (select Encryption status from optional fields)
  - S3 Select or Athena – to filter and list only un-encrypted objects
  - **Note:** S3 Batch Operations job must have access to the S3 bucket and KMS key



Cross-Account Access to Objects in S3 Buckets:

- One of the following ways can be used to grant cross-account access to S3 object
  - IAM Policies and S3 Bucket Policy



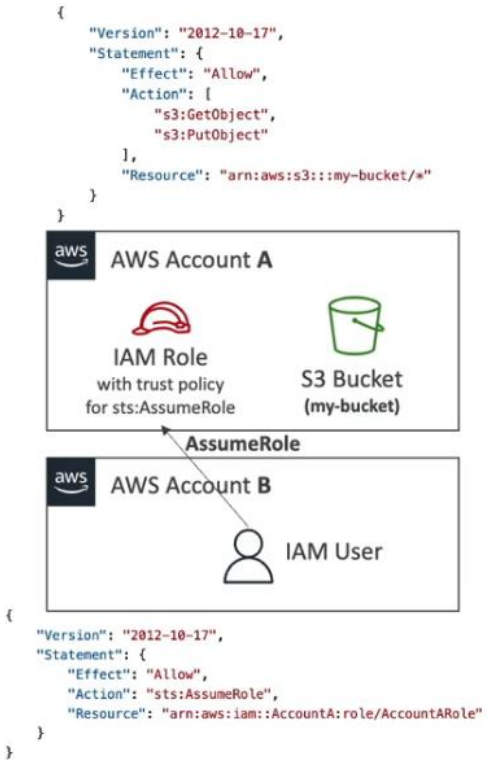
  - IAM Policies and ACLs
    - ACLs only work if *Buckets Owner Enforced setting = Disabled*
    - By default, all newly created buckets have *Buckets Owner Enforced setting = Enabled*
    - Account A user can make sure it gives up the object ownership by granting the object ownership to the Account B administrator

- With adding condition that requests include ACL-specific headers
  - Grant full permissions explicitly – *s3:x-amz-grant-full-control*
  - Use Canned ACL

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

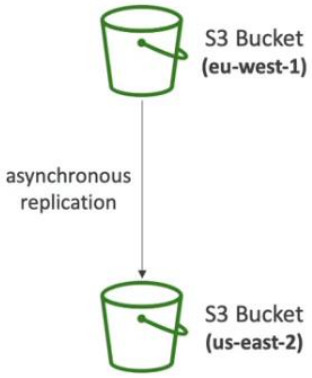
- Using S3 Object Ownership – bucket level setting to disable ACL

- Cross-Account IAM Roles
  - Cross-account IAM roles can be used to centralize permission management when providing cross-account access to multiple services
  - Bucket Policy is not required as the API calls to S3 come from within the account – through the assumed IAM role



Cross Region Replication:

- Replicates objects from one region to another
- By default, this is done using SSL. Creating bucket policies or IAM policies are not required to turn on this feature



- Objects can be replicated from a source bucket to only one destination bucket
  - **Note:** now Amazon S3 replication support for multiple destination buckets – with S3 Replication (multi-destination) data can be replicated in the same AWS Regions using S3 SRR or across different AWS Regions by using S3 CRR, or a combination of both
- After Amazon S3 replicates an object, the object cannot be replicated again
- Cross-Region Replication Requirements:
  - The source and destination buckets must have versioning enabled

- The source and destination buckets must be in different AWS regions
- AWS S3 must have permissions to replicate objects from that source bucket to the destination bucket on your behalf
- If the source bucket owner also owns the object, the bucket owner has full permission to replicate the object. If not, the object owner must grant the bucket owner the READ and READ_ACP permissions via the object ACL

- Cross-Region Replication Cross Accounts:
  - The IAM role must have permissions to replicate objects in the destination bucket
  - The owner of the destination bucket must grant the owner of the source bucket permissions to replicate objects with a bucket policy
  - In the replication configuration, you can optionally direct Amazon S3 to change the ownership of object replica to the AWS account that owns the destination bucket
  - Use case could be CloudTrail auditing, log everything to S3 bucket inside the AWS account, then turn on cross-region replication, then duplicate those audit logs to another AWS account – users from the first AWS account can't go and write or and delete those logs – AWS Standard Architecture for CloudTrail auditing
- What is replicated?
  - Any new objects created after replication configurations are added
  - In addition to unencrypted objects, Amazon S3 replicates objects encrypted using Amazon S3 managed keys (SSE-S3) or AWS KMS managed keys (SSE-KMS)
  - Objects metadata
  - Any object ACL updates
  - Any object tags
  - Amazon S3 replicates only objects in the source bucket for which the bucket owner has permissions to read objects and read ACLs
  - if delete marker is used, then the delete marker is replicated – old versions will be there both in source and destination buckets
- What is not replicated?
  - Anything created before cross-region replication is turned on
    - Existing objects can be replicated using S3 Batch Replication
  - Objects created with server-side encryption using customer-provided encryption keys (SSE-C)
  - Objects created with server-side encryption using AWS-KMS-managed encryption keys (SSE-KMS), unless this option is explicitly enabled
  - Objects in the source bucket for which the bucket owner does not have permissions – this can happen when the object owner is different from the bucket owner
  - Deletes to a particular VERSION (with version ID) of an object (delete markers will be replicated, deleted versions of files are not replicated) – this is a security mechanism

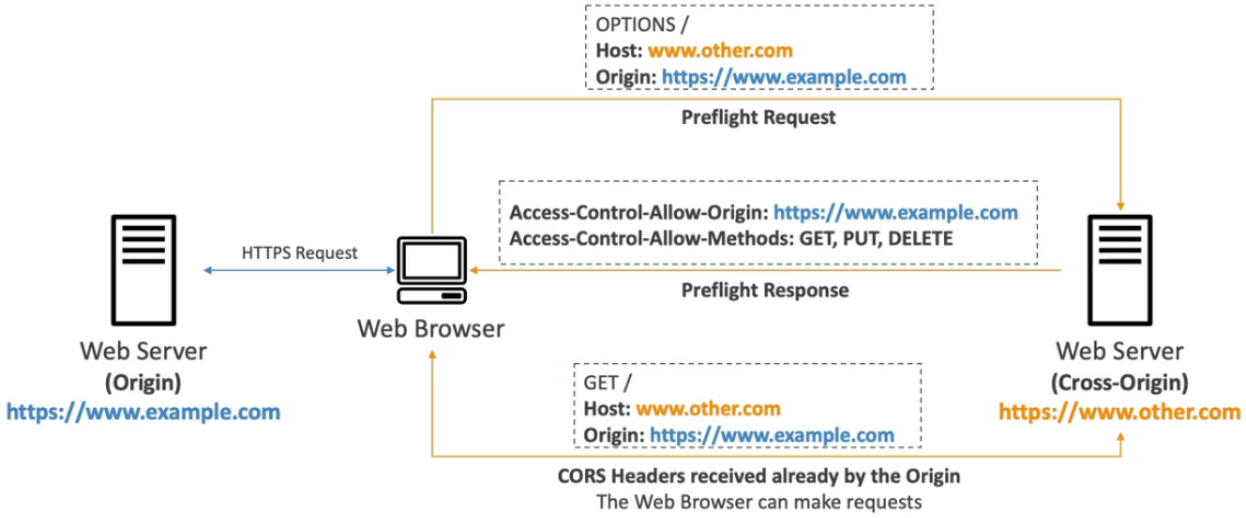
Same-Region Replication:

- Amazon S3 introduces Same-Region Replication
- Amazon S3 now supports automatic and asynchronous replication of newly uploaded S3 objects to the destination bucket in the same region

- With this, new objects uploaded to an Amazon S3 bucket are configured for replication at the bucket, prefix, or object tag levels
- Replicated objects can be owned by the same AWS accounts as the original copy or by the different accounts, to protect from accidental deletion
- When an S3 object is replicated using SSR, the metadata, ACLs, and object tags associated with the object are also part of the replication

Cross-Origin Resource Sharing:

- A security mechanism that allows web browsers to make cross-origin requests, meaning requests to a different domain, port, or protocol
    - Means it's like a permission system that allows one website to request and use things from another website
- It ensures that resources (scripts, fonts, APIs, etc.) on one domain are accessible from another domain
- It uses HTTP headers to specify which domains are allowed to access
- By default, browsers block cross-origin requests, but CORS allows authorized cross-origin requests bases on the specified rules
    - The requests won't be fulfilled unless the other origin allows for the requests, using CORS headers (like Access-Control-Allow-Origin)
- Provides flexibility and security by allowing server-side configuration to define which domains are allowed to access resources
- Helps prevent malicious scripts on one domain from accessing sensitive data on another domain



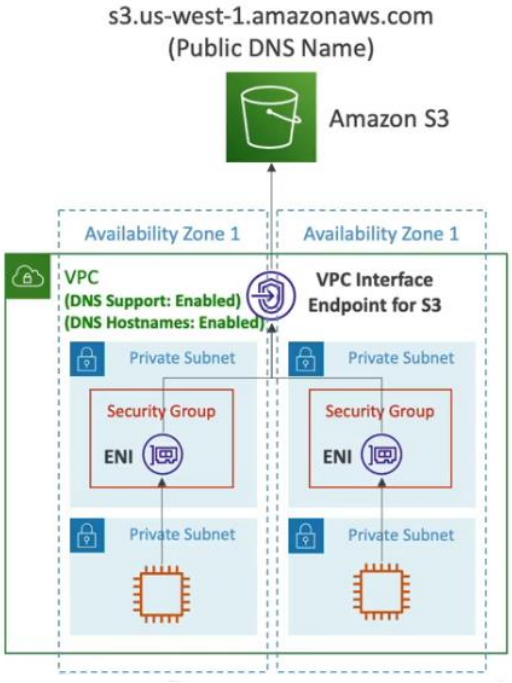
- By default, S3 buckets do not allow cross-origin requests, and the browser blocks such requests due to the Same-Origin policy
- To enable cross-origin access, CORS needs to be configured on the S3 bucket by specifying the allowed origins, method, headers, and other parameters
- The CORS configuration is set at the bucket level and applies to all objects within the bucket
- We can allow for specific origin or for all origins (using *)

- When a browser makes a cross-origin request to an S3 bucket, it sends an HTTP OPTIONS preflight request to the bucket to check the CORS headers
- The S3 service responds with CORS headers, specifying if the request is allowed or denied based on the configured CORS rules
- If the CORS headers allow the request, the browser can proceed with the actual cross-origin request to access the S3 resources

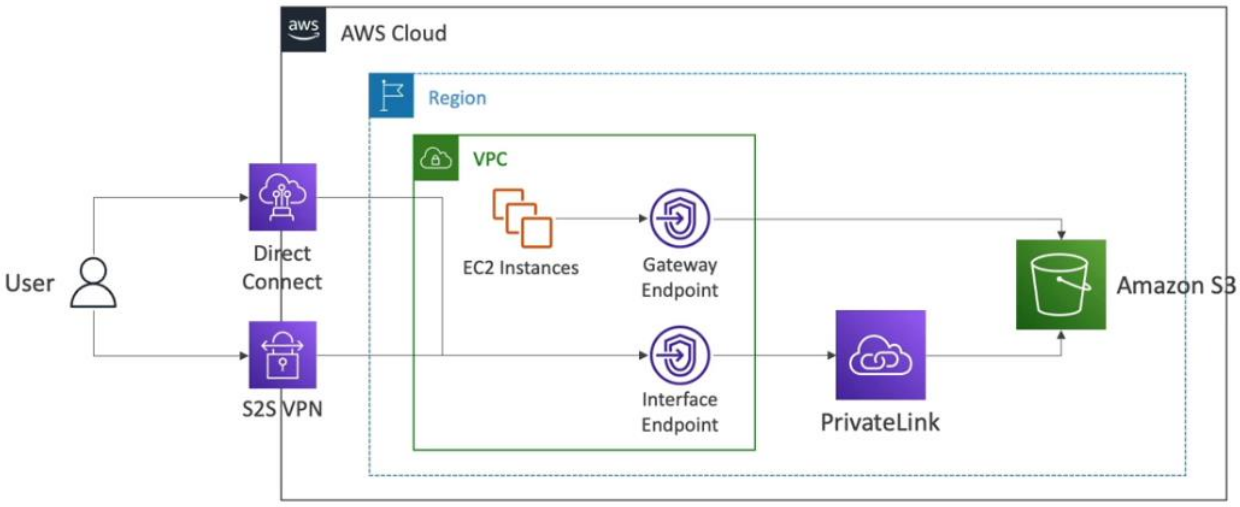
VPC Endpoints Strategy – S3:

- Using VPC Gateway Endpoint – for S3
  - No cost
  - Only accessed by resources in the VPC where it's created
  - Configure the Route Table accordingly
  - Make sure DNS Support is Enabled
  - Keep on using the public DNS of S3
  - Make sure Outbound rules of Security Group of EC2 instance allows traffic to S3

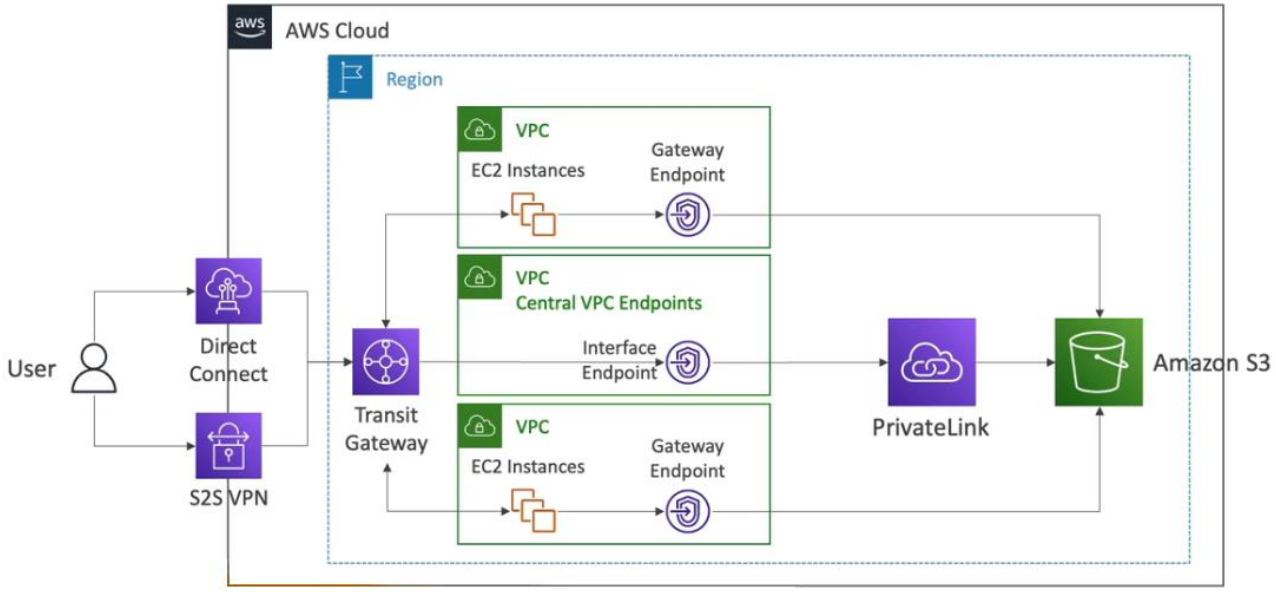
- Using VPC Interface Endpoint – for S3
  - Costs $0.01 per hour per AZ
  - Can access from on-premises – VPN or Direct Connect
  - The public hostname of a service will resolve to the private Endpoint Interface hostname
  - Both VPC setting Enable DNS hostname and Enable DNS Support must be true
  - No Private DNS name option for VPC Interface Endpoint for S3



- Single VPC Architecture – Example



- Multiple VPCs Architecture – Example



- VPC Endpoint Restrictions – can be done using *aws:SourceVpc* and *aws:SourceVpce*

Restrict Access from **specific VPCs**

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

Restrict Access from **Specific VPC Endpoints**

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

- IP Address Restrictions – can be done using *aws:SourceIp* and *aws:VpcSourceIp*

Restrict Access from **Public IP** Addresses

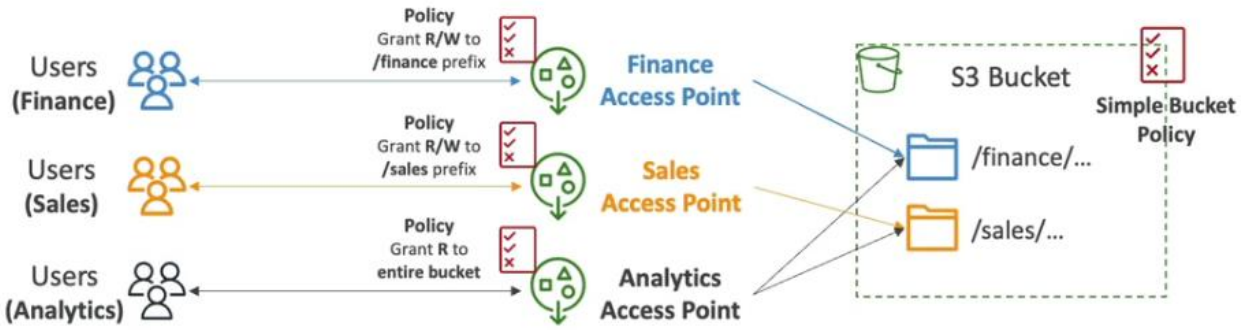
```
{
    "Effect": "Deny",
    "Action": "s3:*",
    "Principal": "*",
    "Resource": [
        "arn:aws:s3:::my-bucket",
        "arn:aws:s3:::my-bucket/*"
    ],
    "Condition": {
        "NotIpAddress": {
            "aws:SourceIp": [
                "192.0.2.0/24",
                "203.0.113.0/24"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

Restrict Access from **Private IP** Addresses

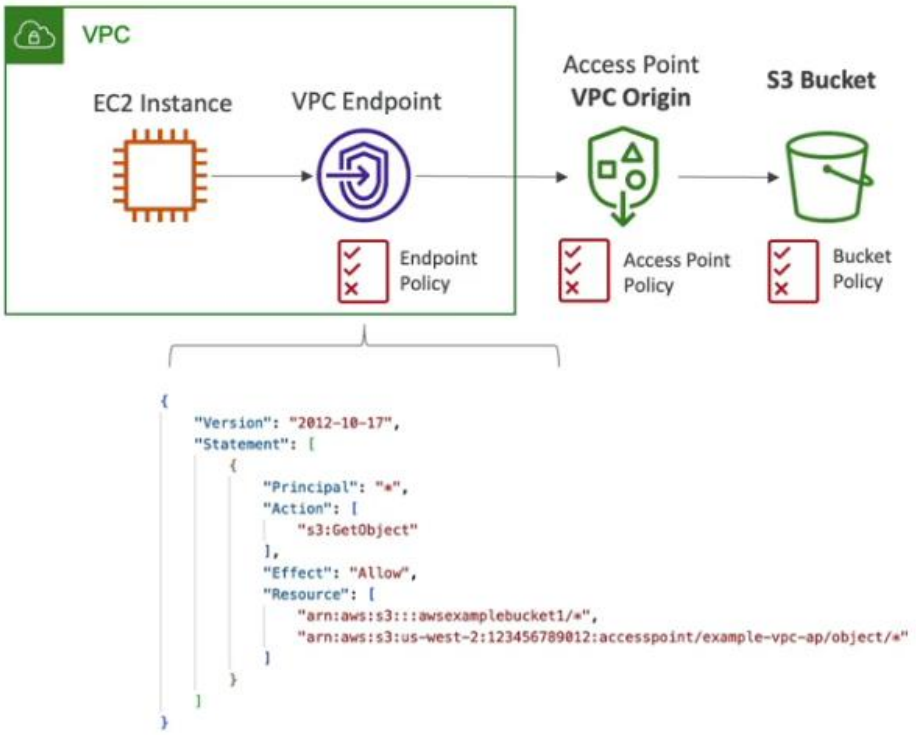
```
{
    "Effect": "Deny",
    "Action": "s3:*",
    "Principal": "*",
    "Resource": [
        "arn:aws:s3:::my-bucket",
        "arn:aws:s3:::my-bucket/*"
    ],
    "Condition": {
        "NotIpAddress": {
            "aws:VpcSourceIp": [
                "10.1.1.1/32",
                "172.1.1.1/32"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

S3 Access Points:

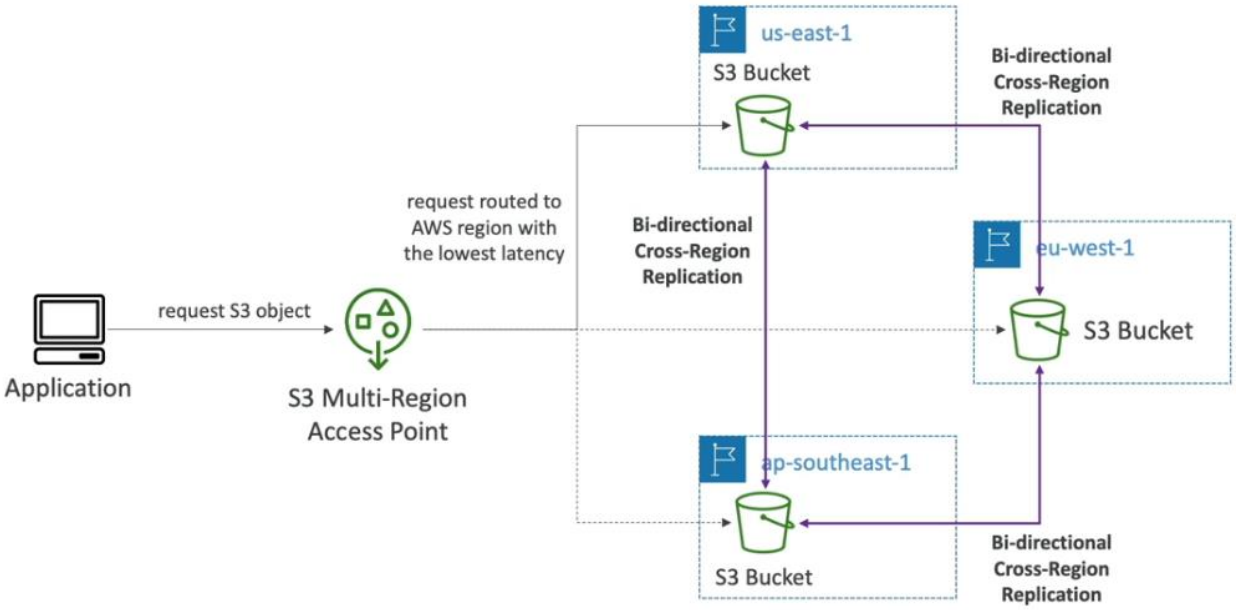
- Access Points simplify security management for S3 buckets
- Each Access Point has:
    - Its own DNS name – Internet Origin or VPC Origin
    - An access point policy like S3 bucket policy – manage security at scale



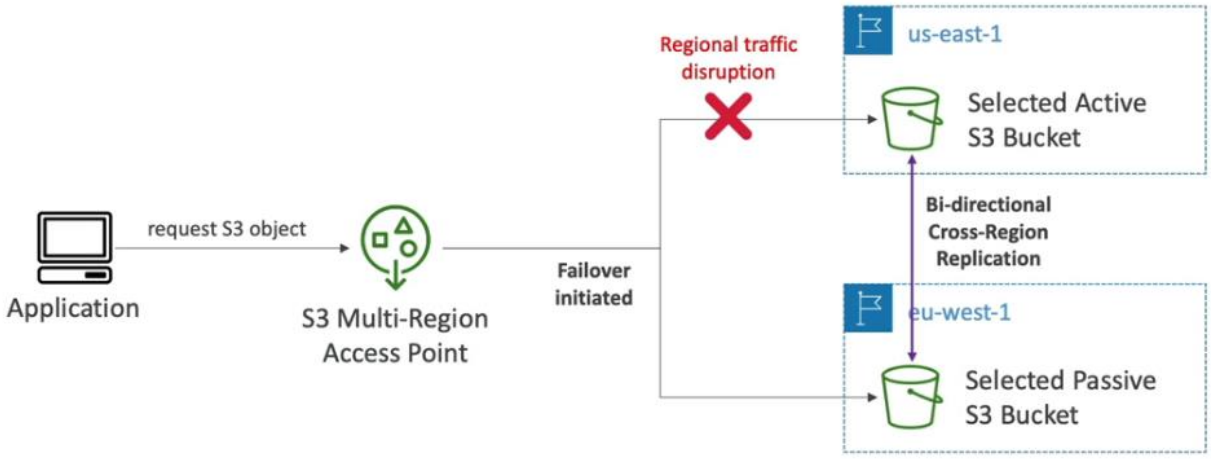
- VPC Origin – S3 Access Points
    - We can define the access point to be accessible from within the VPC
    - User must create a VPC Endpoint to access the Access Point (Gateway or Interface Endpoint)
    - The VPC Endpoint policy must allow access to the target bucket and Access Point



- Multi-region access points – provide a global endpoint that span S3 buckets in multiple AWS regions
    - Dynamically route requests to the nearest S3 buckets – lowest latency
    - Bi-directional S3 bucket replication rules are created to keep data in sync across regions



    - Failover controls – allows to shift requests across S3 buckets in different AWS region within minutes (Active-Active, Active-Passive)

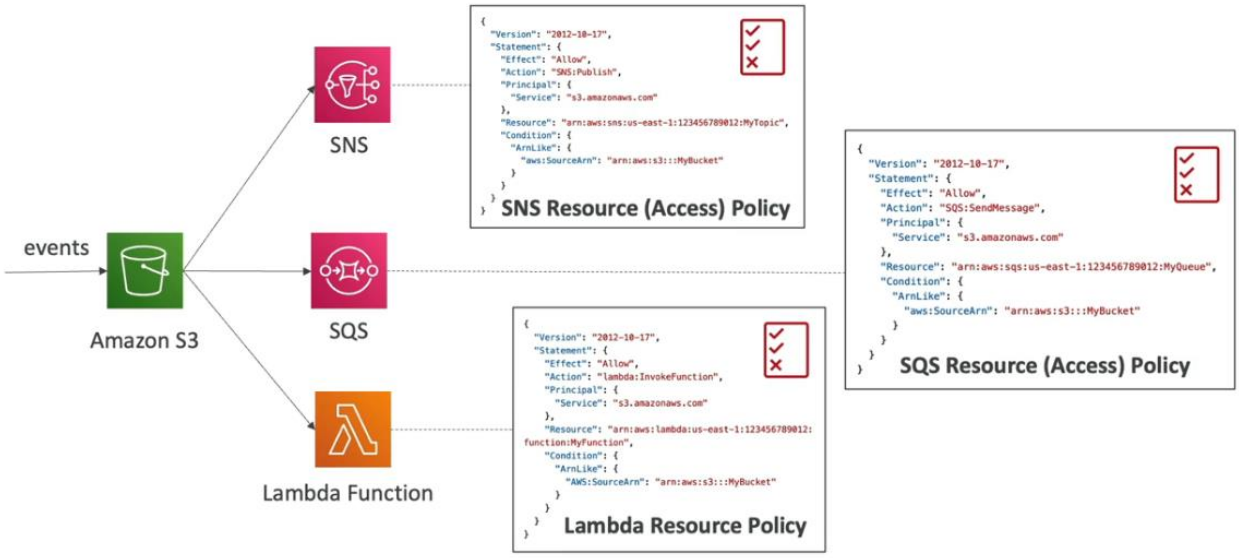


Securing S3:

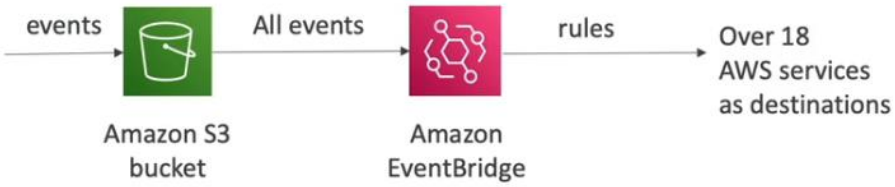
- Using CloudFront:
  - Separate certificates for load balancers – separate certificates for CloudFront distribution
  - Edit origin > Restrict bucket access (yes) > origin access identity > grant read permissions on bucket (yes, update bucket policy) > yes edit
- Using Pre-Signed URLs:
  - Objects can be accessed using pre-signed URLs
  - Typically, these are done via SDK but can be done via CLI
  - They exist for certain length of time in seconds (default is 1 hour) – (time length can be changed using -- expires-in parameter)
  - All objects by default are private. Only the object owner has permission to access these objects. However, the object owner can optionally share objects with others by creating a pre-signed URL, using their own security credentials, to grant time-limited permission to download the objects

S3 Event Notifications:

- S3:ObjectCreated, S3:ObjectRemoved, S3ObjectRestored, S3:Replications
- Object name filtering possible – *.jpg



- S3 event notifications IAM permissions
    - SNS resource access policy
    - SQS resources access policy
    - Lambda resource policy
- S3 event notifications with EventBridge
    - Advanced filtering – options with JSON rules (metadata, object size, name…)
    - Multiple destinations – step functions, kinesis streams/firehose
    - EventBridge capabilities – archive, replay events, reliable delivery

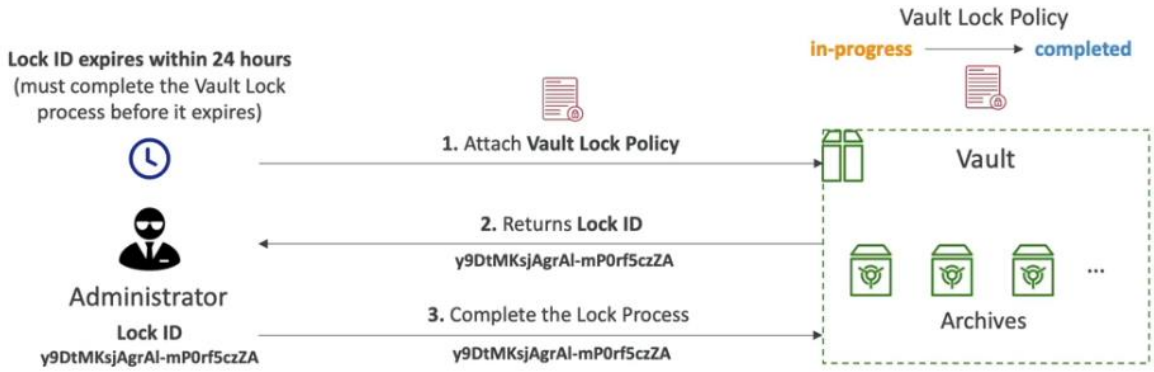


Glacier Vault Lock:

- Extremely low-cost cloud storage service for data archiving and long-term backup
- Data is stored in archived – a single or multiple files stored in a .tar/.zip file
- Vaults are containers which store one or more archives – and an archive is any object, such as a photo, video, or document that can be stored in a vault. An archive is the base unit of storage in Glacier
- Unlimited number of archives can be stored in a vault
- Vault Lock Policy – enforce compliance controls
    - Similar to an IAM policy
    - Configure WORM archives (write once read many)
    - Create data retention policies
    - Once locked, the policy can no longer be changed or deleted
    - Example of a vault lock policy: - preventing deletion of archive in 365 days (retention)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "deny-based-on-archive-age",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "glacier:DeleteArchive",
      "Resource": ["arn:aws:glacier:us-west-2:123456789012:vaults/examplevault"],
      "Condition": {
        "NumericLessThan": {
          "glacier:ArchiveAge": "365"
        }
      }
    }
  ]
}
```

- Configuring a Vault Lock – 2 steps to configuring a Vault Lock
  - Initiate a lock by attaching a vault lock policy to the vault – this sets the lock to an in-progress state
  - Then there will be 24 hours to validate the lock policy
  - If the policy doesn't work as expected, it can be aborted
  - Once validate, lock policies are immutable

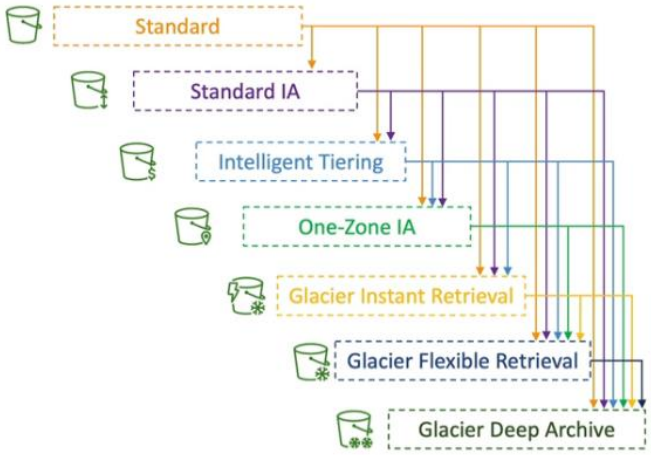


S3 Object Lock:

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

S3 Lifecycle Rules:

- Moving between storage classes - users can transition objects between storage classes
  - Standard IA – for infrequently accessed object
  - Glacier or Glacier Deep Archive – for archive objects to whom fast access is not required
  - Moving objects can be automated via Lifecycle Rules



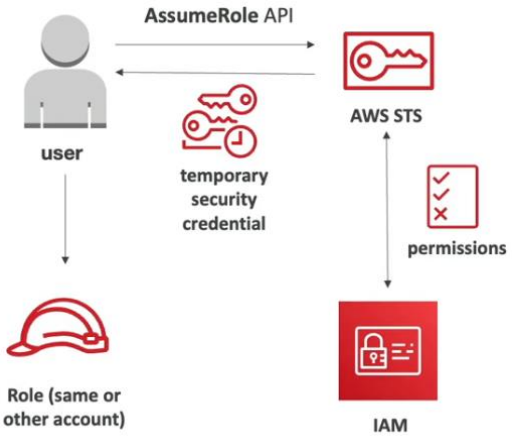
- Transition actions – configure objects to transition to another storage class, Examples:
  - Move objects to Standard IA class 60 days after creation
  - Move to Glacier for archiving after 6 months
- Expiration actions – configure objects to expire/delete after some time
  - Access log files can set to delete after 365 days, Examples:
  - Can be used to delete old versions of files – if versioning is enabled
  - Can be used to delete incomplete multi-part uploads
- Rules can be created for a certain prefix – s3://mybucket/leads/*
- Rules can be created for certain object tags – Department: SOC

S3 Analytics:

- Provides detailed metrics and insights about the access patterns and usage of user's S3 usage
  - Storage metrics
  - Request metrics
  - Data transfer metrics
- Helps to understand and analyze storage access patterns, identify trends, and optimize S3 bucket configurations
- Allows to visualize the data through S3 lens – intuitive dashboards and recommendations for storage optimization
  - Helps in making informed decisions about storage optimization, cost management, and performance enhancements – when to transition objects to the right storage class
  - Recommendations for Standard and Standard IA – does not work for One-Zone IA or Glacier
- Report is updated daily

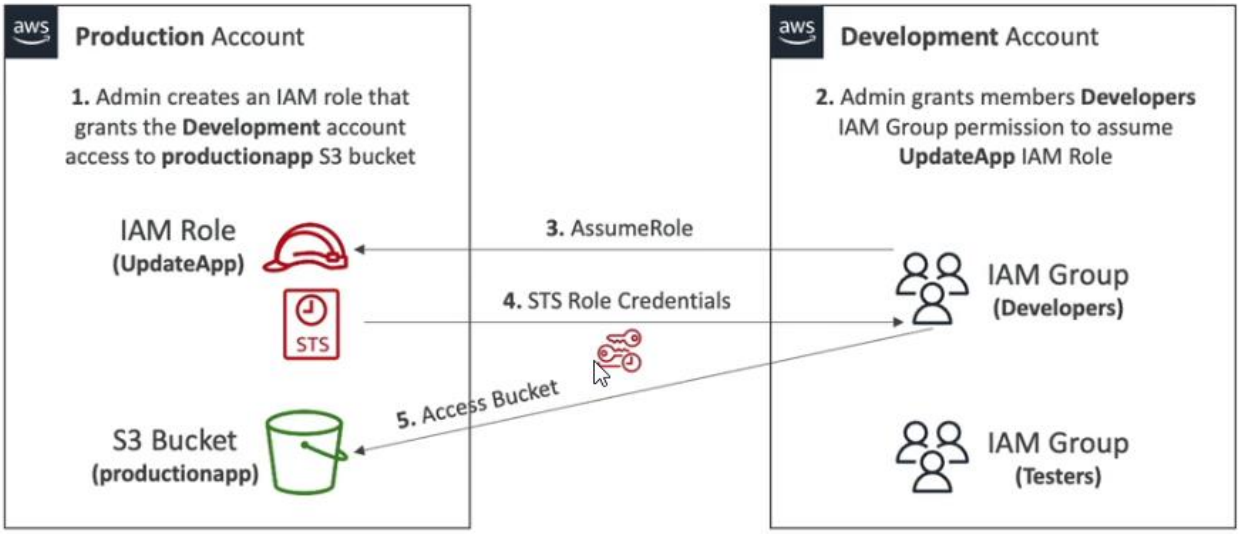
AWS STS:

- Security Token Service – grants user limited and temporary access to AWS resources
- A web service that enables to request temporary, limited-privilege credentials for AWS Identity and Access Management (IAM) users or for users that authenticate (federated users)
- Provides an API for requesting credentials, including IAM user credentials, federated user credentials, or AWS account root user credentials
- Provides 4 main operations:
    - The AWS STS API *AssumeRole* supports Cross Account Access
        - With user's own account for enhanced security
        - Cross account access – assume role in target account to perform actions there
    - The AWS STS API *AssumeRoleWithWebIdentity* supports Web ID Federation
        - Returns creds for users logged with IdP (Facebook/Google login etc.)
        - AWS recommends using Cognito instead of this
    - The AWS STS API *AssumeRoleWithSAML* supports Active Directory/ Azure AD Federation
        - Returns credentials for users logged with SAML
    - The AWS STS API *GetSessionToken* supports retrieving temporary security credentials for IAM users
        - For MFA, from a user or AWS account root user
- Using STS to assume a role – steps required
    - Define an IAM role within the account or cross-account
    - Define which principal can access this IAM role
    - Use AWS STS to retrieve credentials and impersonate the IAM Role – *AssumeRole* API
    - Temporary credentials can be valid between 15 minutes to 1 hour



- In case of Cross-account access with STS:
    - Admin in target account creates an IAM role that grants the source account access to the resource that is going to be accessed
    - Admin in source account grants members of the specific IAM group permissions to assume the specific role

- The group's members will be able to assume role
- The role will get the temporary credentials via STS
- The users assuming the role will be able to access the specific resource that was intended to be accessed
- Explained with example in below diagram

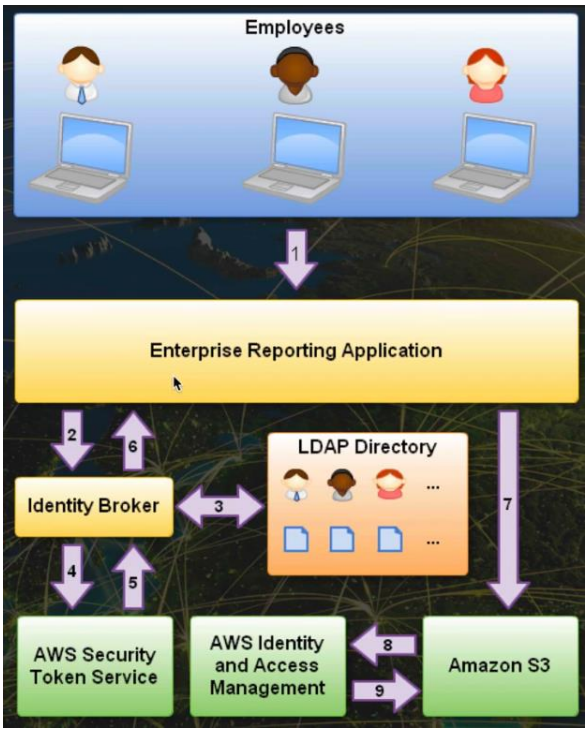


Federation:

- Federation with AD:
  - Uses SAML
  - Grants temporary access based off the users Active Directory credentials – does not need to be a user in IAM
  - Single sign on allows users to log in to AWS console without assigning IAM credentials
- Federation with Mobile Apps – uses Facebook/Amazon/Google or other OpenID provides to log in
- Cross-Account Access let's user from one AWS account access resources in another
- Federation – Combining or joining a list of users in one domain (such as IAM) with a list of users in another domain (such as Active Directory, Facebook etc)
- Identity Broker – a service that allows to take an identity from point A and join it (federate it) with point B
- Identity Store – services like Active Directory, Facebook, Google etc.
- Identities – users of a service like Facebook etc.
- Web Identity Federation:
  - It lets users access to AWS resources after they have successfully authenticated with a web-based identity provider like Amazon, Facebook, or Google
  - Following successful authentication, the user receives an authentication code from the Web ID provider, which they can trade for temporary AWS security credentials
  - The user authenticates first with the Web ID provider and receives an authentication token, which is exchanged for temporary AWS credentials allowing them to assume an IAM role

How STS works? An Example

- A company is hosting its website on some EC2 web servers in its VPC
- Users of the website must log in to the site which then authenticates against the companies AD servers which are based on-site at the companies HQ
- Company's VPC is connected to Company's HQ via a secure IPSEC VPN
- Once logged in the user can only have access to their own S3 bucket
- How it will be set up? Consider the following architecture diagram
    1. Employees enter their username and password
    2. The application calls an Identity Broker – the broker captures the username and password
    3. The Identity Broker uses the organization's LDAP directory to validate the employee's identity
    4. The Identity Broker calls the new *GetFederationToken* function using IAM credentials – the call must include an IAM policy and a duration (1 to 36 hours), along with a policy that specifies the permissions to be granted to the temporary security credentials
    5. The STS confirms that the policy of the IAM user making the call to *GetFederationToken* gives permissions to create new tokens and then returns four values to the application:
        a. An Access Key
        b. A Secret Access Key
        c. A Token
        d. A duration (the token's lifetime)
    6. The Identity Broker returns the temporary security credentials to the reporting application
    7. The data storage application uses the temporary security credentials (including token) to make requests to Amazon S3
    8. Amazon S3 uses IAM to verify that the credentials allow the requested operation on the given S3 bucket and key
    9. IAM provides S3 with the go-ahead to perform the requested operation



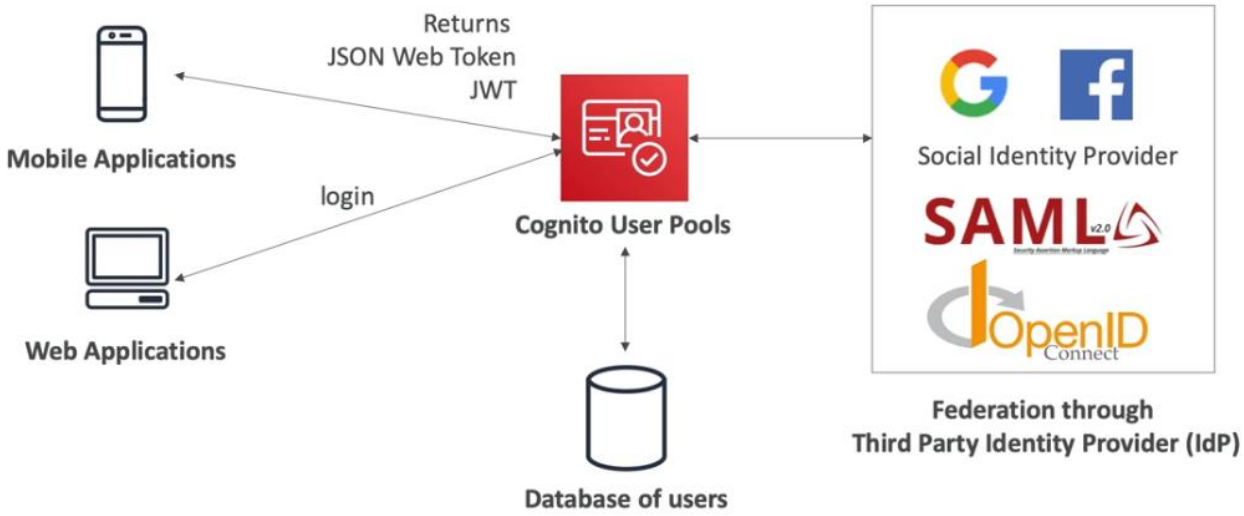
AWS Cognito:

- Cognito User Pools provide user management and authentication within the application
- Cognito Identity Pools extend authentication to access AWS services and resources using temporary credentials
- Identity federation in AWS enables integration with external identity providers for user authentication
    - Web identity federation is a specific type of identity federation that enables users to authenticate through popular web identity providers – Amazon, Facebook, Google, or Apple
    - It grants temporary AWS credentials for accessing AWS services

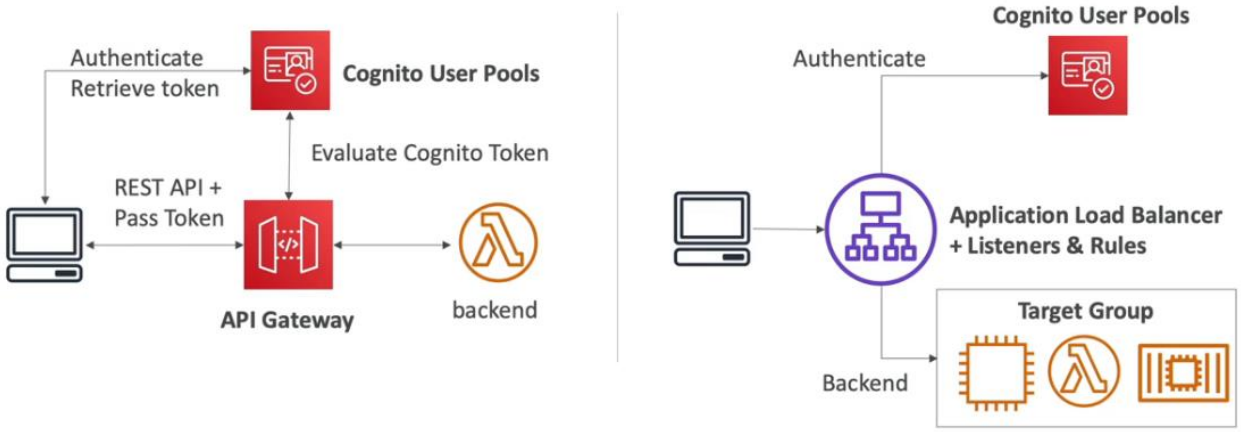
Cognito User Pools:

- Fully managed user directory service provided by AWS – allows to create and maintain a user pool, which is a collection of user identities, in the application
- Cognito uses User Pools to manage user sign-up and sign-in directly or via Web Identity Providers – for mobile and web applications
- Cognito User Pools provide built-in functionality, including MFA and email or phone verification
- User Pools support various identity providers – Google, Facebook, SAML, and Amazon for social sign-in
    - Users can sign-in directly to the User Pool, or indirectly via an identity provider
- Cognito simplifies the process of adding authentication and access control to applications, allowing developers to focus on building their core features and functionality

- Cognito acts as an Identity Broker between the ID provider and AWS
- Provides secure access to user data through JWT, which are issued upon successful authentication
- Feature – block users if their credentials are compromised elsewhere



- Cognito User Pools integrations
  - It can be integrated with API Gateway and Application Load Balancer

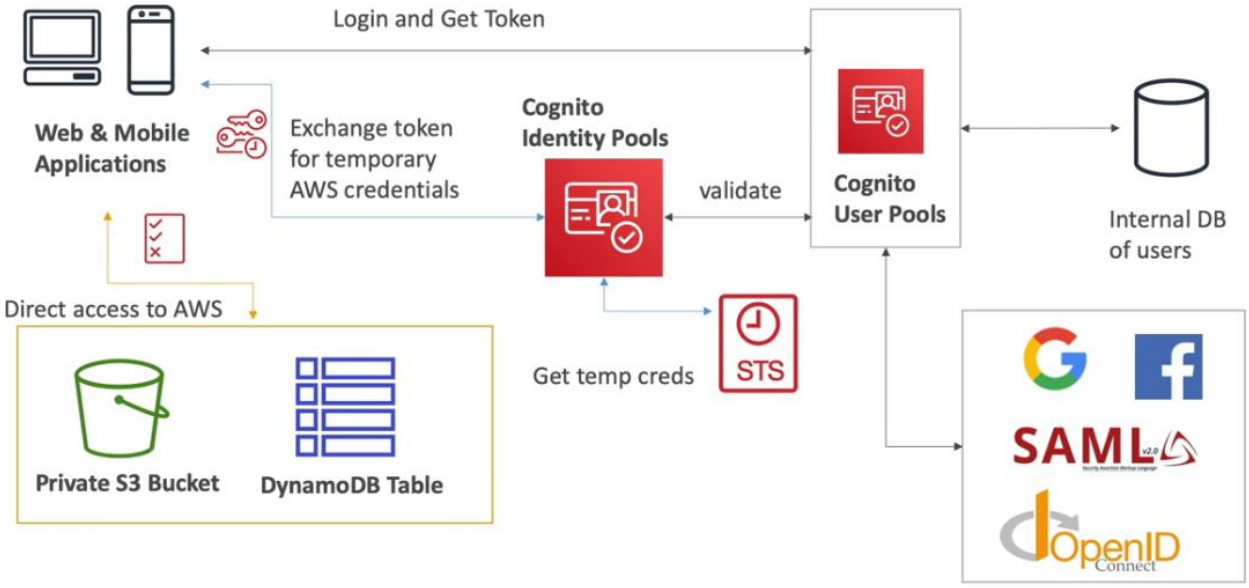


Cognito Identity Pools:

- Also knows as federated identities – service provided by AWS for identity federation
- Allows to create unique identities for users across multiple identity providers and authenticated backend services – identity providers like:
  - Public Providers – Amazon, Facebook, Google, Apple etc.
  - Users in Amazon Cognito user pool
  - OpenID Connect Providers & SAML Identity Providers
  - Developer Authenticated Identities – custom login server
- Identity pools provide temporary limited-privilege AWS credentials to end users, allowing them to access services securely – through API Gateway or directly

- The IAM policies applied to the credentials are defined in Cognito
- They can be customized based on the *user_id* for fine grained control

- With identity pools, different roles and permissions can be defined for authenticated and unauthenticated users
- Supports fine-grained access control through IAM roles
- Integrates with Cognito User Pools to provide a comprehensive solution for user authentication and authorization
- Supports cross-account access
- Offers built-in support for handling guest users and unauthenticated access to resources
- Leverages AWS STS to generate temporary credentials for users



- IAM Roles – Cognito Identity Pools
  - Default IAM roles for authenticated and guest users – example policy is below

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

- Define rules to choose the role for each user based on the user's ID
- User's access can be partitioned using policy variables – example policy is below

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

- IAM credentials are obtained by Cognito Identity Pool through STS
- The roles must have a trust policy for Cognito Identity Pools
- An example policy for DynamoDB could be as below

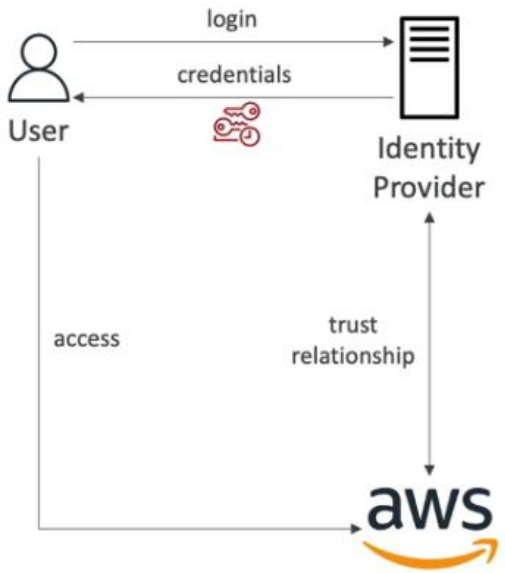
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetItem", "dynamodb:BatchGetItem", "dynamodb:Query",
                "dynamodb:PutItem", "dynamodb:UpdateItem", "dynamodb:DeleteItem",
                "dynamodb:BatchWriteItem"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

Identity Federation in AWS:

- Identity federation in AWS enables integration with external identity providers for authentication purposes
- It allows users to authenticate using their external credentials from external sources like
  - AD or SAML-based providers

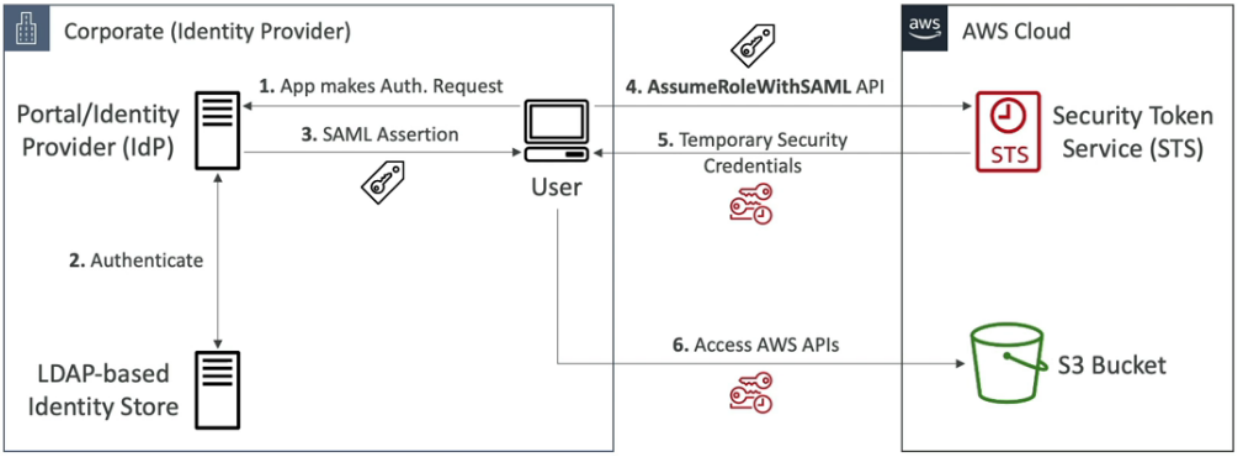
- Custom Identity Broker
- Web Identity Federation with/without Amazon Cognito
- SSO – single sign-on

- IAM users are not required to be created – users management is outside of AWS

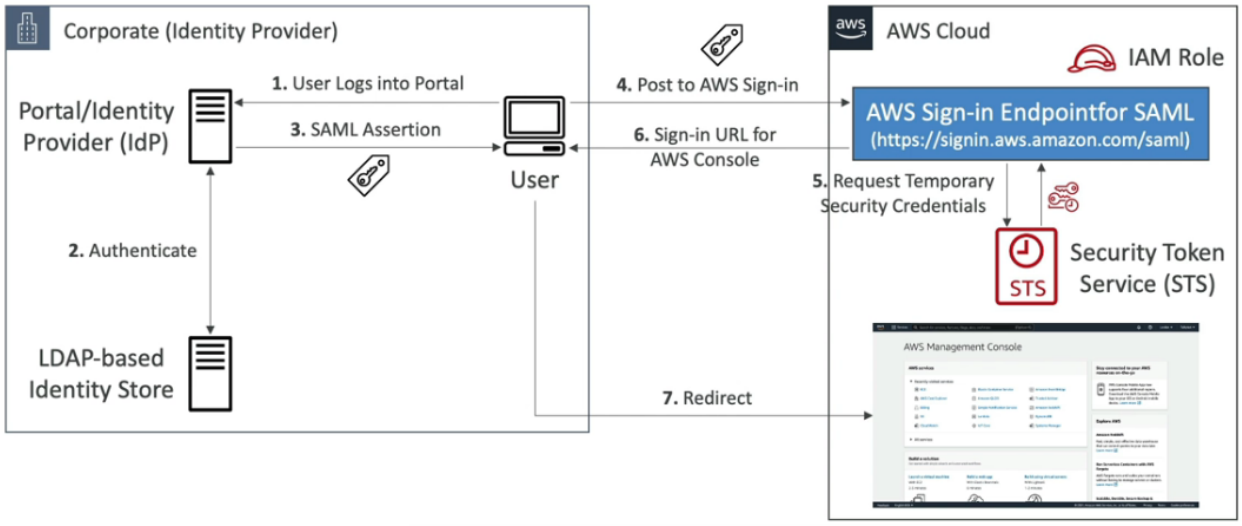


SAML 2.0 Federation:

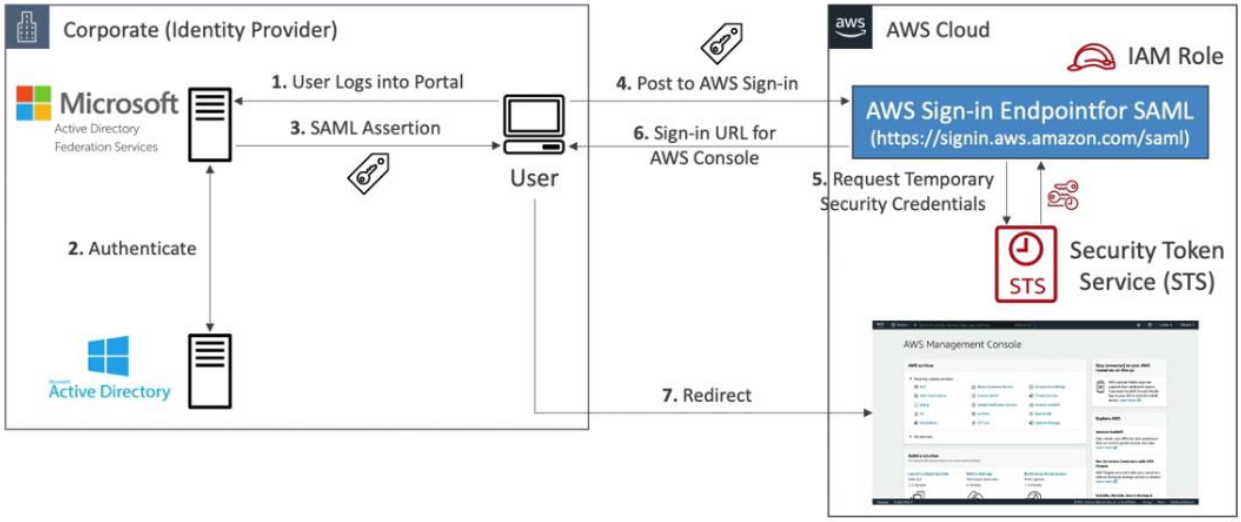
- *AssumeRoleWithSAML* is an API call – an example is below



- An example of AWS console access via SAML 2.0 federation

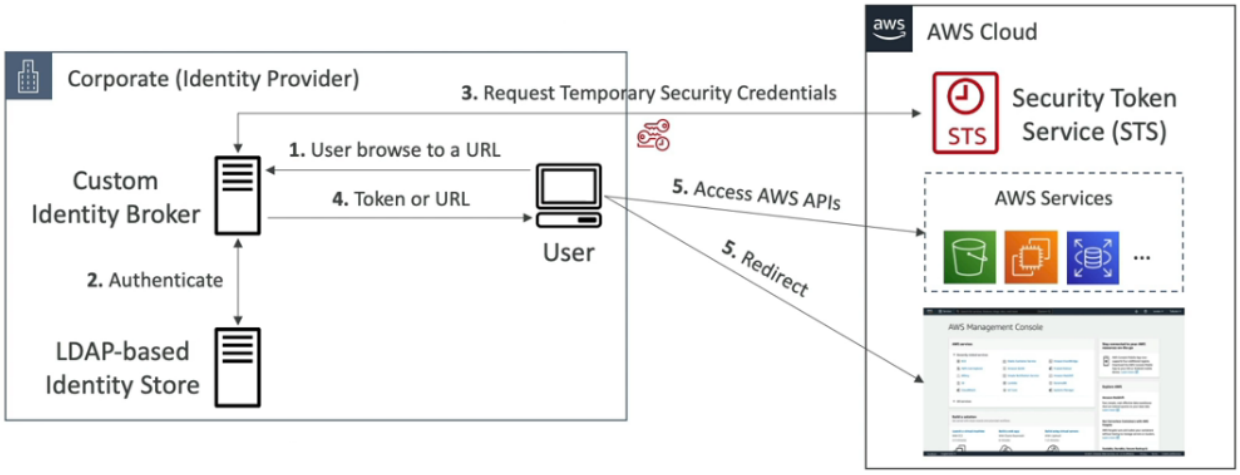


- Another similar example where AWS console access is provided via ADFS (Active Directory Federation Service)



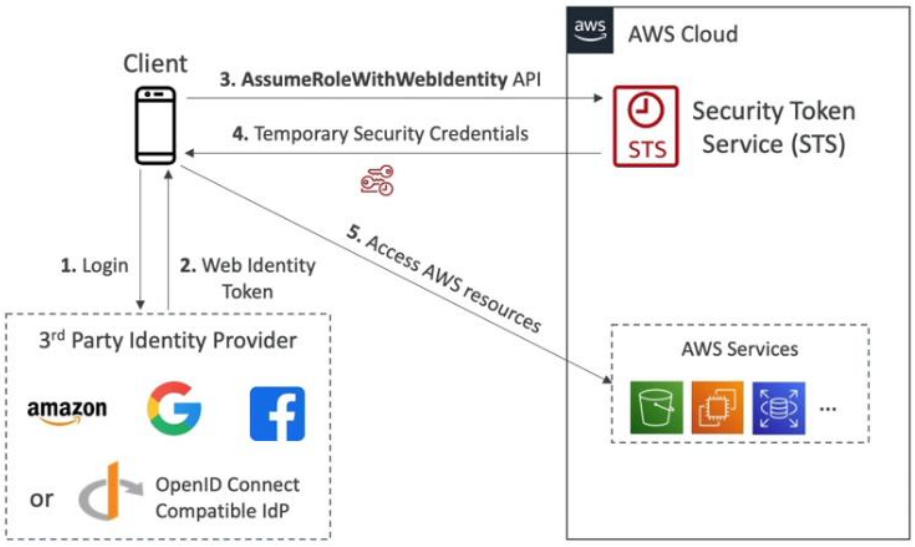
Custom Identity Brokers:

- Use only if provider is NOT compatible with SAML 2.0
- The identity broker authenticates users and requests temporary credentials from AWS
- The identity broker must determine the appropriate IAM role
- Uses the STS API *AssumeRole* or *GetFederationToken*



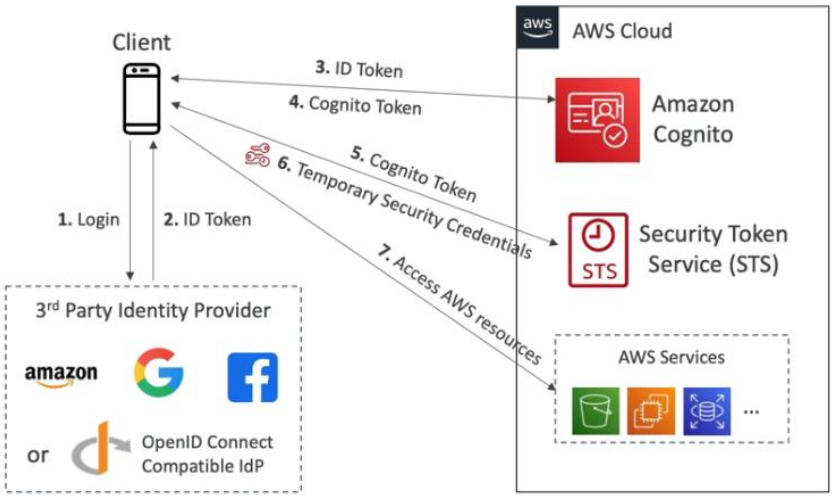
Web Identity Federation:

- A mechanism that enables users to authenticate with AWS services using identity providers like Amazon, Facebook, Google, or other OpenID Connect-compatible providers
- After being authenticated with Web Identity Federation, user can be identified with IAM policy variable
    - Cognito-identity.amazonaws.com:sub
    - www.amazon.com:user_id
    - Graph.facebook.com:id
    - Accounts.google.com:sub
- Two approaches – with/without Cognito
- Without Cognito – not recommended by AWS
    - Uses *AssumeRuleWithWebIdentity* API call



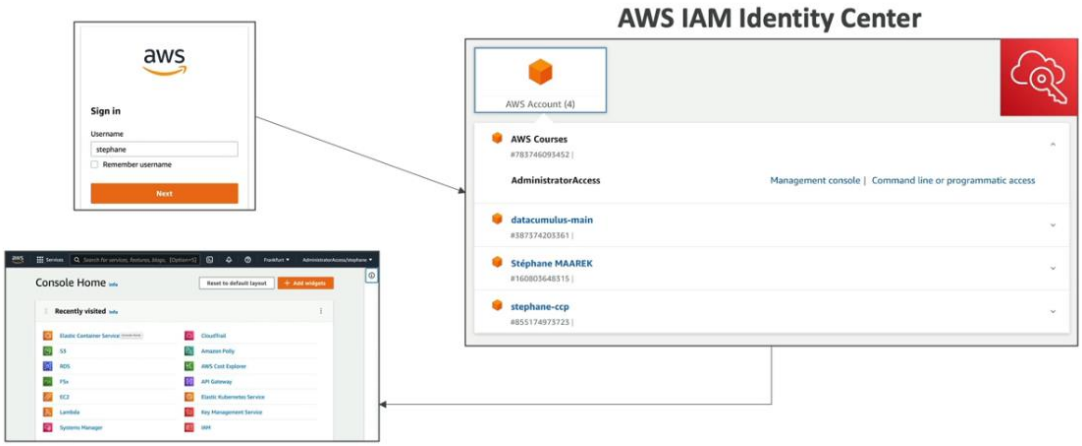
- With Cognito – preferred for web identity federation
    - Sign-up and sign-in to the customer's apps
    - Create IAM Roles using Cognito with the least privilege needed

- Build trust between the OIDC IdP and AWS
- Acts as an Identity Broker between customer's application and Web ID providers, so no additional code is required
- Supports anonymous users
- Supports MFA
- Data synchronization



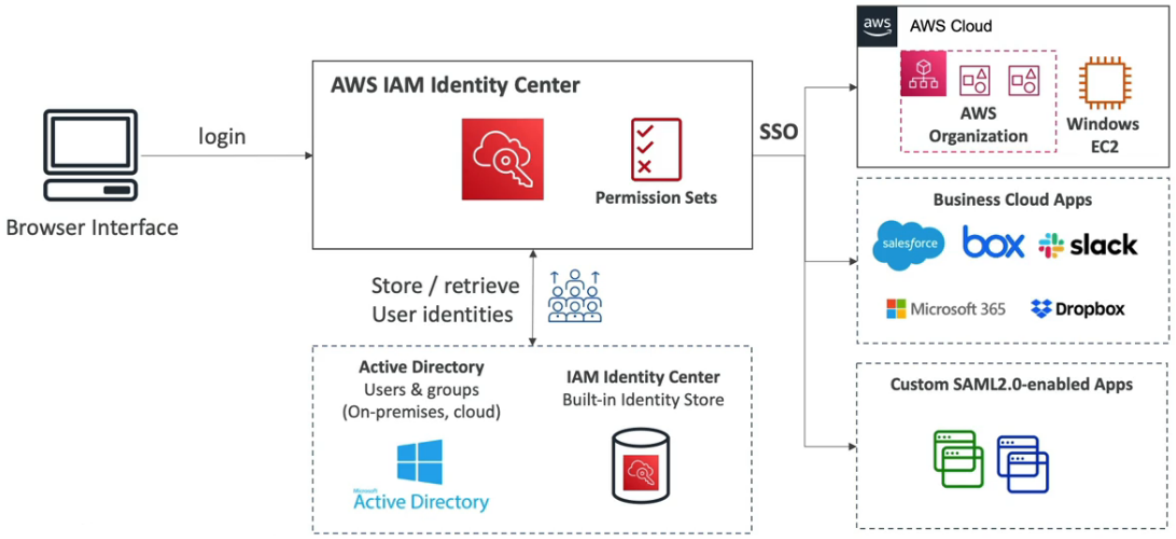
AWS IAM Identity Center or SSO:

- With one login (single sign-on), it simplifies user access to
  - AWS accounts in AWS Organizations
  - Business cloud applications – salesforce, box, Microsoft 365 etc.
  - SAML 2.0-enabled applications
  - EC2 Windows Instances

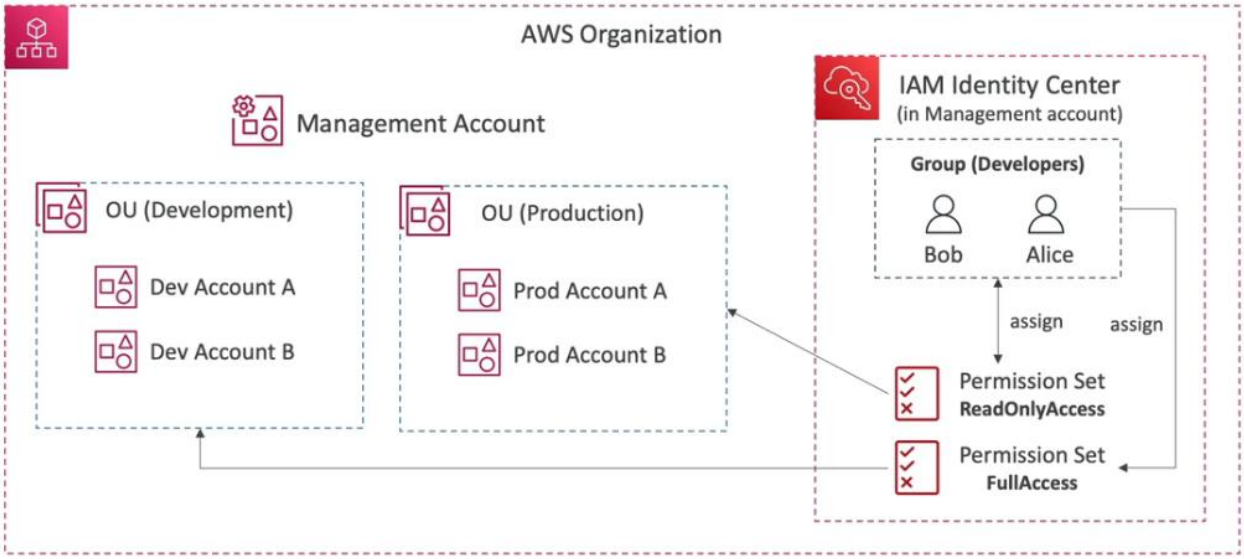


- Identity providers could be
  - Built-in identity store in IAM Identity Center

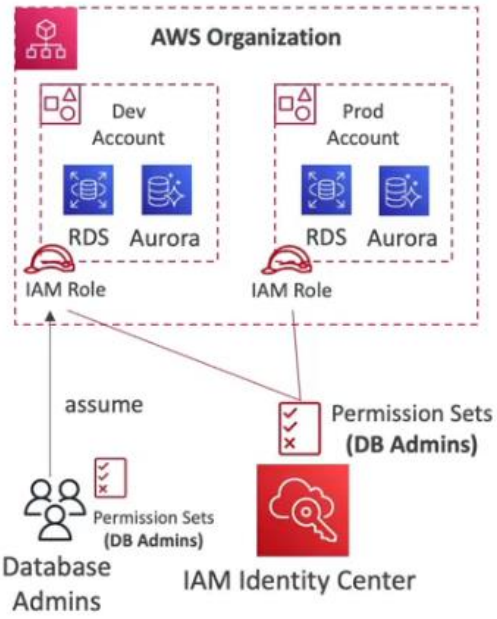
- 3rd party – AD, OneLogin, Okta etc.



- How does everything relate in IAM Identity Center for permissions, users, and groups? Explained in below diagram



- Another example explaining the fine-grained permissions and assignments in managing access across AWS accounts in AWS Organization

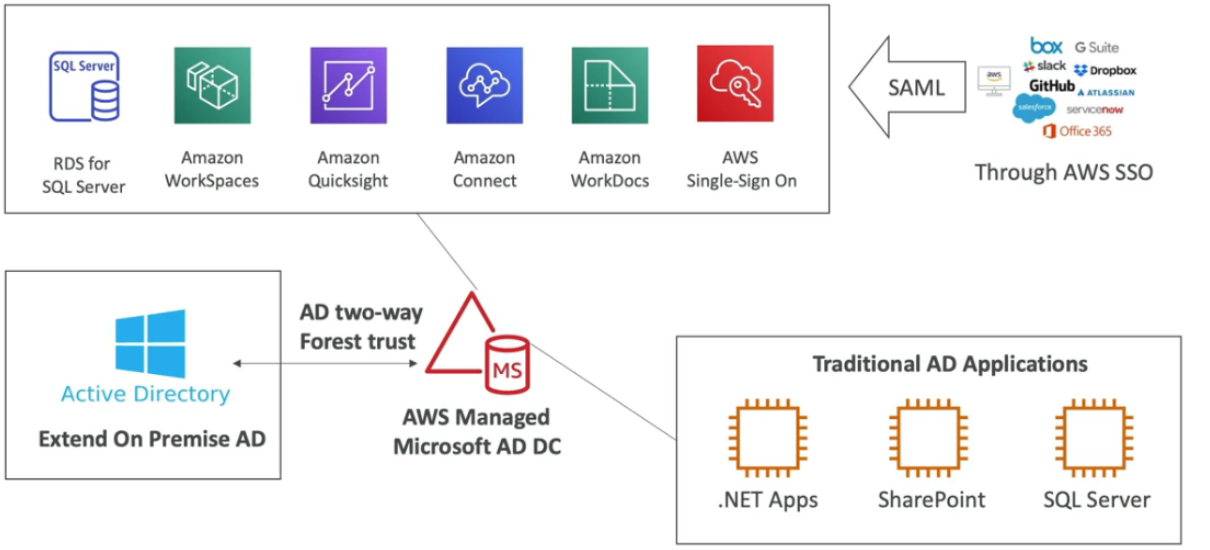


AWS Directory Service:

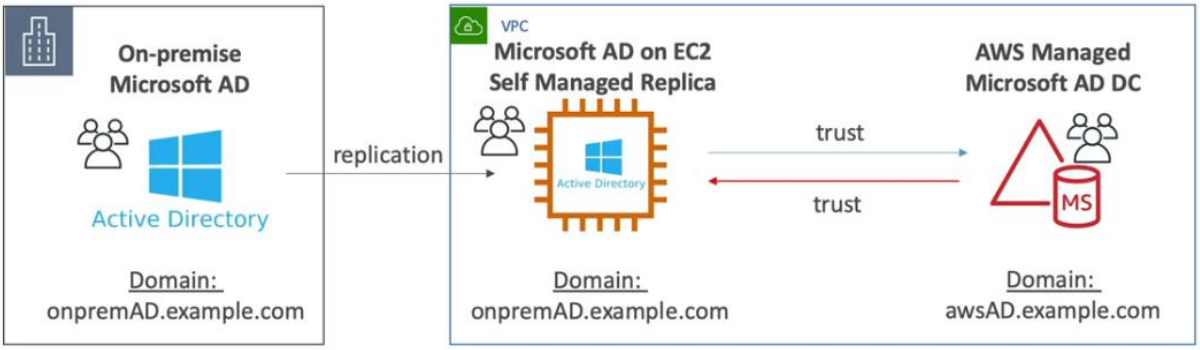
- Managed service provided by AWS that simplifies the setup and management of directory solutions in AWS
    - AWS Managed Microsoft AD
    - AD Connector
    - Simple AD

AWS Managed Microsoft AD:

- Users can create their own AD in AWS, manage users locally, supports MFA
- Establish "trust" connections with their on-premises AD
- Microsoft AD in your AWS VPC
- EC2 Windows Instances:
    - EC2 Windows instances can join the domain and run traditional AD applications – SharePoint etc.
    - Seamlessly Domain Join Amazon EC2 Instances from Multiple Accounts & VPCs
- Integrations
    - RDS for SQL Server, AWS Workspaces, QuickSight etc.
    - AWS SSO to provide access to 3rd party applications

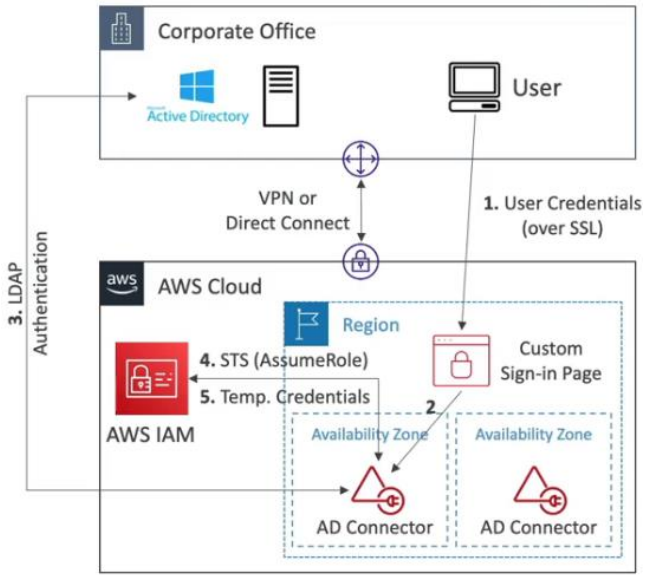


- Standalone repository in AWS or joined to on-premises AD
- Multi-AZ deployment of AD in 2 AZs – no of domain controllers can be increased for scaling
- Automated backups
- Automated multi-region replication of the directory
- An example where users can create replica of on-premises MS AD into EC2 in cloud – minimize latency, establish trust between AWS Managed MS AD and EC2



AD Connector:

- Directory Gateway (proxy) to redirect to on-premises AD, supports MFA
- Users are managed on the on-premises AD
- No caching capability
- Manage users solely on-premises, no possibility of setting up a trust
- VPN or Direct Connect
- Doesn't work with SQL server, doesn't do seamless joining, can't share directory

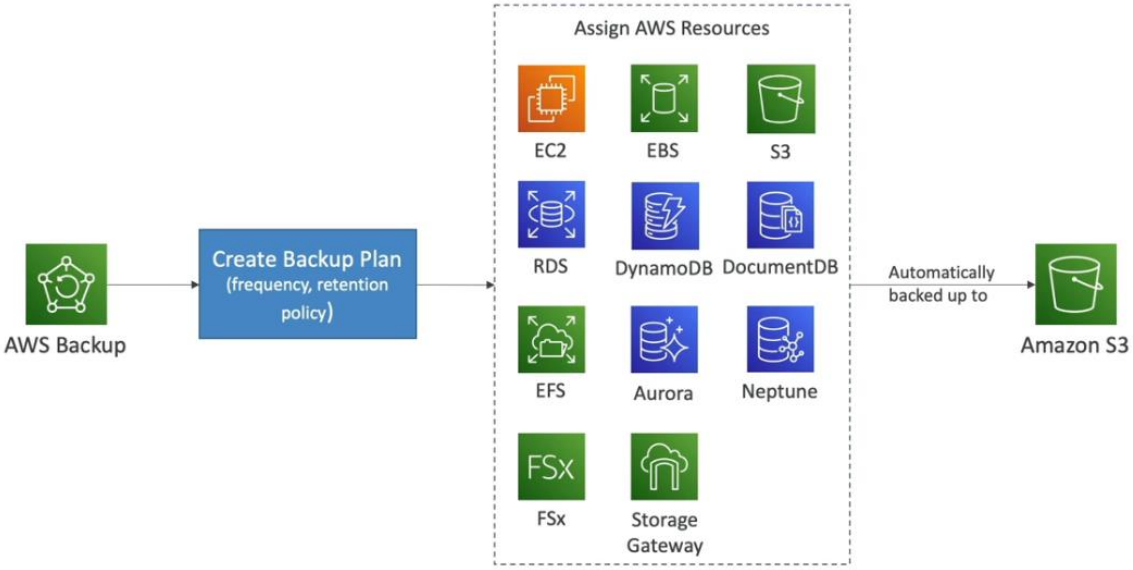


Simple AD:

- Inexpensive basic AD-compatible or LDAP-compatible managed directory on AWS with the common directory features
- Cannot be joined on-premises AD
- Supports joining EC2 instances, manage users and groups
- Does not support MFA, RDS SQL server, AWS SSO
- No trust relationship

AWS Backup:

- Fully managed service provided by AWS
- Centrally manage and automate backups across AWS services – no need to create custom scripts and manual processes
- Supports backup and restore operations of various AWS services
    - EC2, EBS, S3, RDS, Aurora, DynamoDB, DocumentDB, Neptune, EFS, FSx, Storage Gateway etc.
- Provides a unified console and API to manage backups across different services
- Offers features like backup scheduling, retention policies, cross-region backups, cross-account backups, tag-based backups, and incremental backups
- Supports PITR (Point-in-Time-Recovery) for supported services
- Customers can create backup policies known as Backup Plans
    - Backup frequency
    - Backup window
    - Transition to cold storage
    - Retention period
- Customers can monitor and audit backup activity – CloudWatch, CloudTrail



AWS Backup Vault Lock:

- Enforce a WORM (write once read many) state for all the backups that are stored in customer's AWS backup vault
- Additional layer of defense to protect backups against
    - Malicious delete operations
    - Updates that shorten or alter retention periods
- Even the root user cannot delete backups when enabled

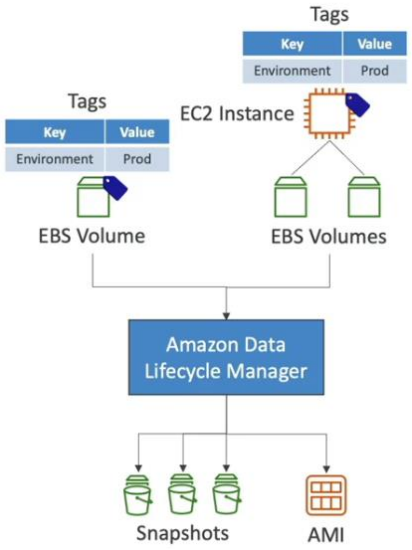
Amazon EBS:

- A block-level storage service provided by AWS – allows to create and attach persistent block storage volumes to EC2 instances
- EBS volumes provide durable, low-latency storage for application and data
- Types – General Purpose SSD, Provisioned IOPS SSD, Throughput Optimized HDD, Cold HDD etc.
- EBS volumes are automatically replicated within AZs – high availability and durability
- Can take EBS snapshots – point-in-time backups stored in S3
    - Restore or create new volumes from snapshots
- EBS volumes can be resized – increase or decrease the storage capacity of volumes as needed
- EBS volumes can be encrypted using KMS for enhanced data protection
- EBS volumes are highly scalable – can be attached or detached from EC2 instances as necessary
- To encrypt an unencrypted volume, you need to take a snapshot.
    - You can create a new volume from the snapshot and enable encryption during the restore step.
    - Another option is to make a snapshot copy and specify the encryption configuration during the copy process.

- The third option is to use EC2 instance to migrate data by mounting a new encrypted volume and existing unencrypted volume. You can use data migration utilities to migrate the data.

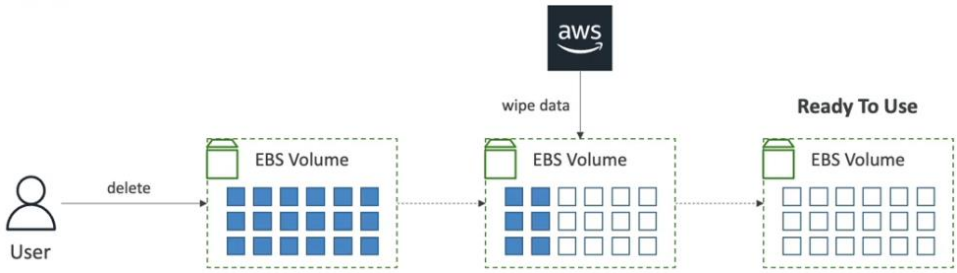
Amazon Data Lifecycle Manager:

- Automate the creation, retention, and deletion of EBS snapshots and EBS-backed AMIs
- Schedule backups, cross-account snapshot copies, delete outdated backups etc.
- Use resource tags to identify the resources – EC2 instances, EBS volumes etc.
- Can't be used to manage snapshots/AMIs create outside DLM
- Can't be used to manage instance-store backed AMIs



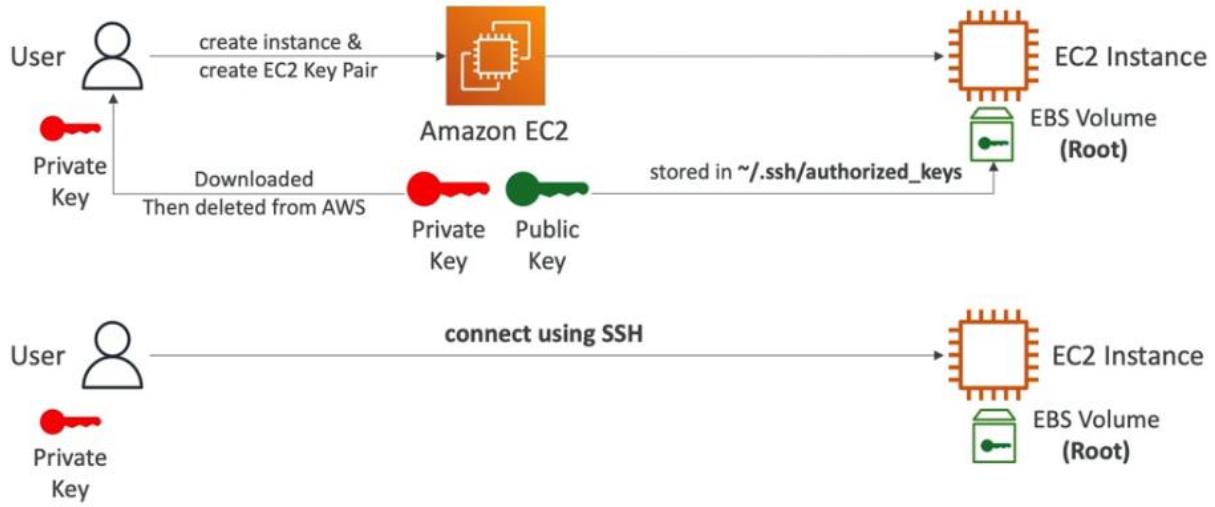
Data Volume Wiping:

- Overwriting the physical block storage used by deleted EBS volume with zeros or random patterns multiple times to make it difficult or impossible to recover original data – before its allocated to a new volume
- **Note:** no need to manually wipe data – although can be triggered manually when deleting a volume or releasing EC2 instance that is using the volume
- Especially important when handling volumes that store sensitive/confidential data



AWS EC2:

SSH Key Pair:



- Private key needs to be kept secure – no way to recover it
- Key pair keys can also be created outside of AWS account and can be uploaded then

Dedicated Instances:

- Dedicated Instances are Amazon EC2 instances that run in a VPC on hardware that's dedicated to a single customer
- These dedicated instances are physically isolated at the host hardware level from instances that belong to other AWS accounts
- Dedicated Instances may share the hardware with other instances from the same AWS account that are not Dedicated Instances
- Pay for Dedicated Instances on demand – save 70% by purchasing Reserved Instances – save 90% by purchasing Spot Instances
- Dedicated Instances and Dedicated Hosts have dedicated hardware
- Dedicated Instances are charged by the instances

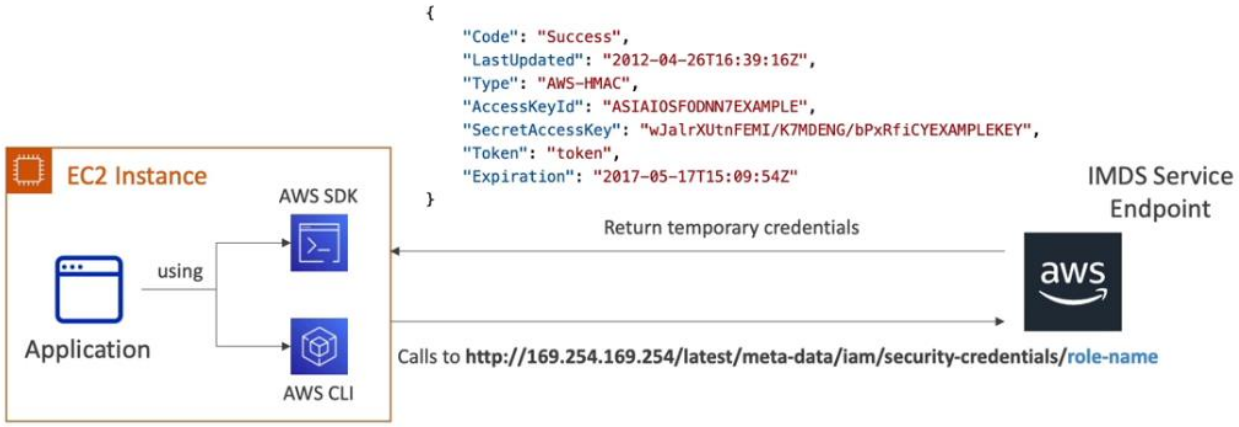
Dedicated Hosts:

- Dedicated Hosts and Dedicated Instances can be used to launch Amazon EC2 instances on physical servers that are dedicated for customer's use
- An important difference between a Dedicated Hosts and Dedicated Instances is that a Dedicated Host gives customer additional visibility (in things like sockets, cores, and host id) and control over how Instances are placed on a physical server and customers can consistently deploy their instances on the same physical server over time
- As a result, Dedicated Hosts enable customers to use their existing server-bound software licenses and address corporate compliance and regulatory requirements
- Dedicated Hosts are charged by the host

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

EC2 Instance Metadata Service (IMDS):

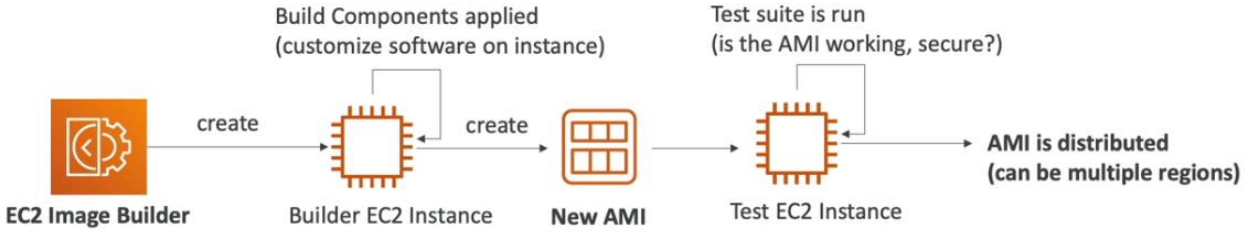
- Information about an EC2 instance – hostname, instance type, network settings, etc.
    - ami-id, block-device-mapping/, instance-id, instance-type, network/
    - local-hostname, local-ipv4, public-hostname, public-ipv4
    - InstaceProfileArn
    - iam/security-credentials/role-name – temporary credentials for the role attached to the instance
    - placement/
    - security-groups
    - tags/instance – tags attached to the instance
- Can be accessed from within the EC2 instance itself by making a request to the EC2 metadata service endpoint – http://169.254.169.254/latest/meta-data
- Can be accessed using EC2 API or CLI tools – curl or wget
- Metadata stored in key:value pairs
- Useful for automating tasks such as setting up an instance's hostname, configuring networking, or installing software
- Instance metadata service is accessible to any application running in the instance.
    - The new IMDS V2 (instance metadata service version 2) has the option to completely turn-off IMDS access in the instance
- EC2 instance role – how does it work? Below diagram will explain



- Access to instance' metadata can be restricted by:
    - Local firewall rules to disable access for some or all processes
        - Iptables for Linux
        - PF or IPFW for FreeBSD
    - Turn off using AWS console or AWS CLI – *HttpEndpoint=disabled*

EC2 Image Builder:

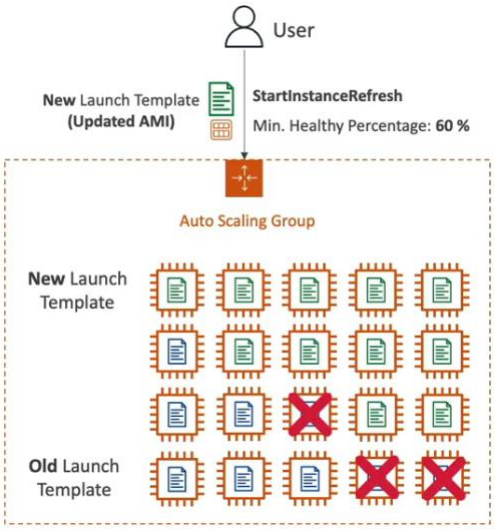
- Used to automate the creation of Virtual machines or container images
    - Automate the creation, maintain, validate, and test EC2 AMIs
- Can be run on a schedule – weekly, whenever packages are updated, etc.
- Free service – only pay for underlying resources



Autoscaling:

- Automatically adjusting the number of EC2 instances in an application fleet based on demand
- Helps ensuring that number of instances are available to handle varying levels of application traffic
- Can scale instances both – vertically and horizontally
    - Vertical – by changing instance type
    - Horizontal – by adding or removing instances
- Uses predefined scaling policies and CloudWatch metrics to determine when and how to scale the instances
- Instance Refresh – allows for the replacement of instances in an Autoscaling group with new ones
    - During Instance Refresh, new instances are launched based on the latest launch configuration or launch template of the auto scaling group

- Setting of minimum healthy percentage
- Specify warm-up time – how long until the instance is ready to use



EC2 as Proxy:

- Each EC2 instance performs source/destination checks by default
- This means that the instance must be the source or destination of any traffic it sends or receives
- However, instances that acts as proxy, NAT must have it disabled.
  - An inline security appliance must be able to send and receive traffic when the source or destination is not itself
  - Therefore, you must disable source/destination checks on these instances

Amazon Relational Database Services:

RDS:

- Relational database service – managed service provided by AWS
- A web service that makes it easier to set up, operate and scale a relational database in the AWS cloud
- Can use the database products – MariaDB, MS SQL Server, Oracle, PostgreSQL
- Manages backups, software patching, automatic failure detection, and recovery
- Can control who can access RDS database – IAM and VPC
- Offers different deployment options – single-AZ, multi-AZ
- Read replicas – for heavy workloads

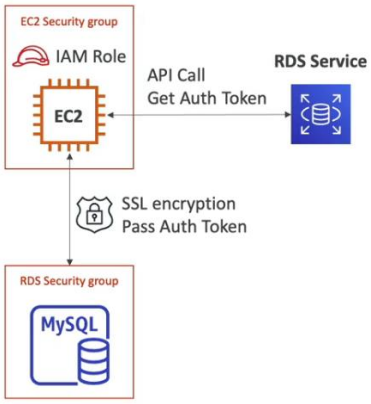
Aurora:

- Fully managed relational database service
- MySQL and PostgreSQL compatible RDS
- Uses a distributed storage architecture that replicates data across multiple AZs – fault tolerance
- An Aurora cluster volume can grow to a max size of 128 TiB

- Supports read replicas – can automatically create up to 15 replicas
- Aurora takes advantage of AWS RDS features for management and administration

RDS and Aurora Security:

- Encryption at rest – KMS
    - Database master and replicas encryption using AWS KMS – must be defined at launch time
    - If the master database is not encrypted, the read replicas cannot be encrypted
    - To encrypt an un-encrypted database – go through a DB snapshot and restore as encrypted
- In-flight encryption – supports SSL/TLS encryption for data in transit
- IAM Authentication – supports using IAM database authentication (roles instead of username password)
    - Works with MariaDB, MySQL and PostgreSQL
    - Centrally manage users in IAM instead of DB
    - Don't need password – just an authentication token obtained through IAM and RDS API call
    - Network in/out must be encrypted using SSL
    - Auth token has a lifetime of 15 min
    - Can leverage IAM roles and EC2 instance profiles for easy integration

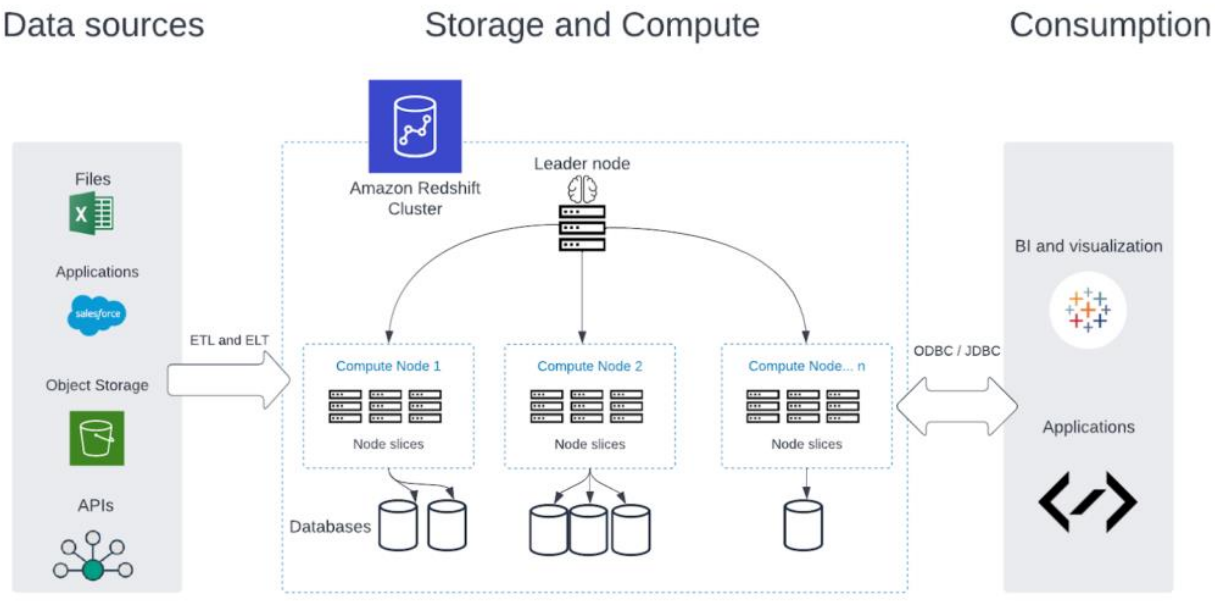


- Security Groups – control access to DB instances' inbound and outbound
- NACLs – control inbound and outbound traffic to the subnet where DB instances are launched
- No SSH available – except on RDS custom
- Network isolation -instances can be deployed within a VPC
- Audit Logs can be enabled – and sent to CloudWatch logs for longer retention
- Encrypt un-encrypted Aurora snapshots – when aurora cluster is encrypted, then all DB instances, snapshots, logs, and backup are all are encrypted

- Can't create an encrypted snapshot of an un-encrypted DB cluster
- Can't encrypt an un-encrypted snapshot while snapshot is copied
- Only when restoring – KMS key can be specified to encrypt

Amazon RedShift:

- Fully managed data warehousing service – designed to handle large-scale data sets
- An enterprise-class relational database query and management system
- Supports high-performance analytics and reporting
- Fast and optimum query performance on large volume of data through:
  - A columnar storage formats
  - Massively parallel processing
  - Efficient, targeted data compression encoding schemes
- Supports standard SQL queries and tools – BI and analytics applications
- Integrates with S3, Data Pipeline, AWS Glue for data ingestion and transformation
- When analytics queries are run, large amount of data will be retrieved, compared, and evaluated in multiple-stage operations to produce a result

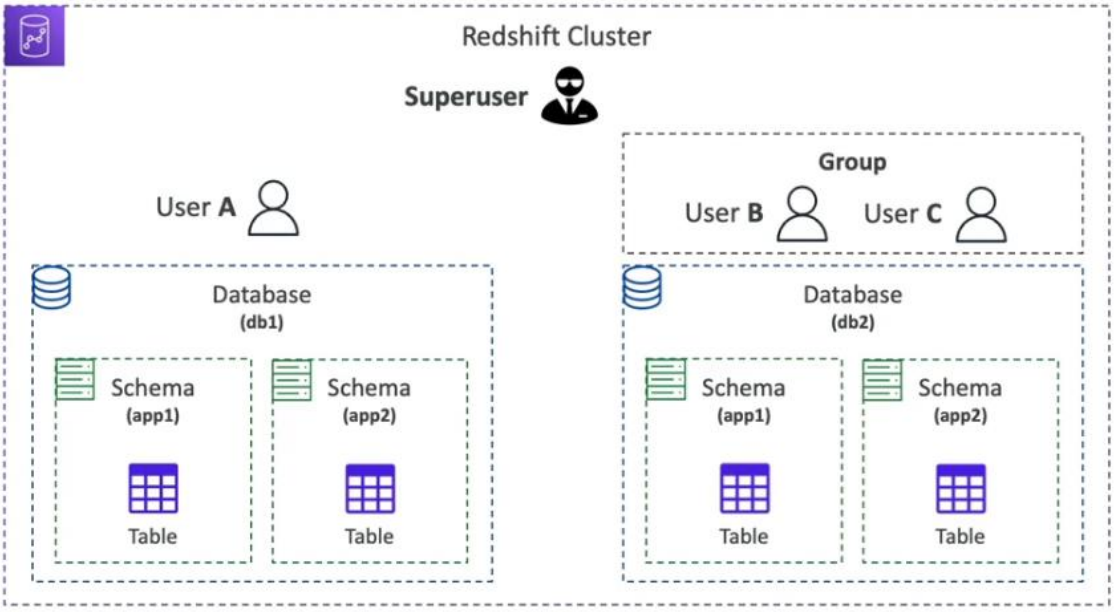


Database Hierarchy:

- Superusers – admin user which is created when RedShift cluster is launched

- Have the same permissions as DB owners for all databases
- A superuser can only be created by superuser

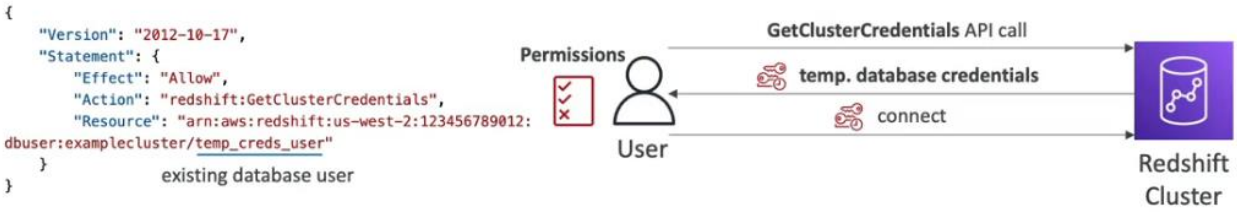
- Users – can only be created and dropped by superuser
  - Can own database and database objects
  - Can grant permissions on those objects to other DB users/groups/schemas
  - User is granted permissions in two ways
    - Explicitly – by having those permissions assigned directly to the account
    - Implicitly – by being a member of a group that is granted permissions
- Groups – collections of users that can be collectively assigned permissions
  - Good for streamlined security maintenance
- Database – collection of one or more Schema
  - When a user creates a database, the user becomes its owner
  - Superusers have the same permissions as database owners
- Schema – collections of database tables and other database objects
  - Used to group database objects under a common name
  - Users can be granted access to a single schema or to multiple schemas



RedShift Security:

- Cluster Security Groups – can be associated with RedShift cluster to control inbound and outbound
- VPC – can launch RedShift cluster by launching in VPC – isolation via subnets
- Cluster encryption – turn on this feature to encrypt data in user-created tables
- SSL connections – encrypt connection between SQL client and RedShift cluster
- Load data encryption – server-side or client-side to encrypt table load data files when uploading
- Data in transit – uses hardware accelerated SSL to communicate with S3 or DynamoDB
- Column-level access control – column-level grant and revoke statements

- Row-level access control – create and attach policies to roles/users that restrict access to rows defined in the policy
- Sign-in credentials – access to RedShift controlled by AWS account permissions
- Access management – access to specific resource of RedShift is controlled by IAM
    - RedShift provides the *GetCredentials* API operations to generate temporary database user credentials
    - When temporary user credentials are created for an existing DB, user's password can be disabled to force users to log on with temporary password
    - Alternatively, *GetClusterCredentails Autocreate* option can be used to automatically create a new database user each time while connecting



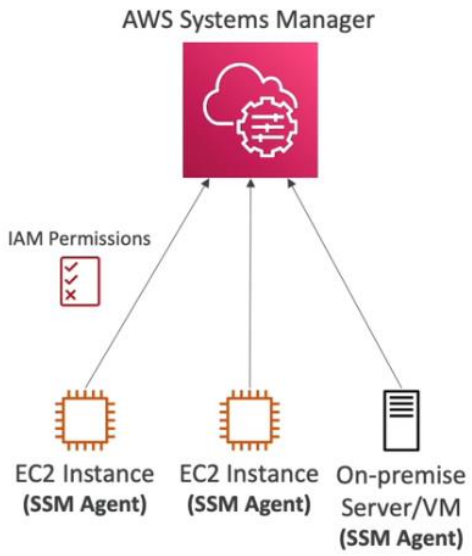
    - To create a read-only user, add a user to a group that only has read-only privileges to the specified Schemas for a database

Systems Manager:

- It provides a unified interface for managing and controlling AWS resources from a single console
- It offers detailed inventory tracking of instances, virtual machines, containers, and more in AWS environment – operational insight about the state of user's infrastructure
- Enables consistent and compliant configuration management across the infrastructure
- Simplifies software patch management for instances and VMs to keep systems secure and up to date
- Allows to automate operational tasks and run scripts across multiple instances
- Integrates with AWS CloudWatch to provide performance monitoring and insights into resources
- Can be integrated with AWS Config
- Free service
- Features include:
    - Resource Group
    - Operations Management
        - Explorer
        - OpsCenter
        - CloudWatch Dashboard
        - PHD
        - Incident Manager
    - Shared Resources
        - Documents

- Change Management
  - Change Manager
  - Automation
  - Change Calendar
  - Maintenance Windows
- Application Management
  - Application Manager
  - AppConfig
  - Parameter Store
- Node Management
  - Fleet Manager
  - Compliance
  - Inventory
  - Hybrid Activations
  - Session Manager
  - Run Command
  - State Manager
  - Patch Manager
  - Distributer

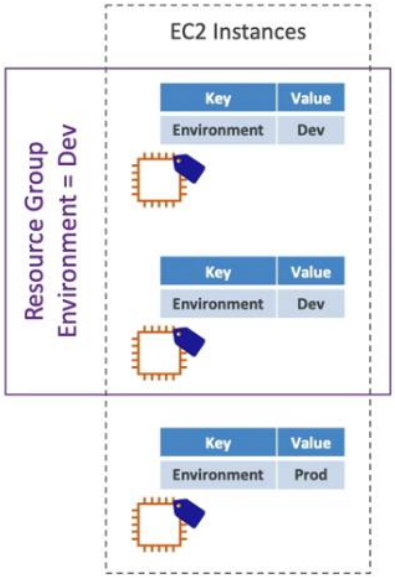
- How Systems Manager works
  - Install the SSM agent onto the systems that need to be controlled
  - Installed by default on Amazon Linux 2 AMI & some Ubuntu AMIs



Resource Group:

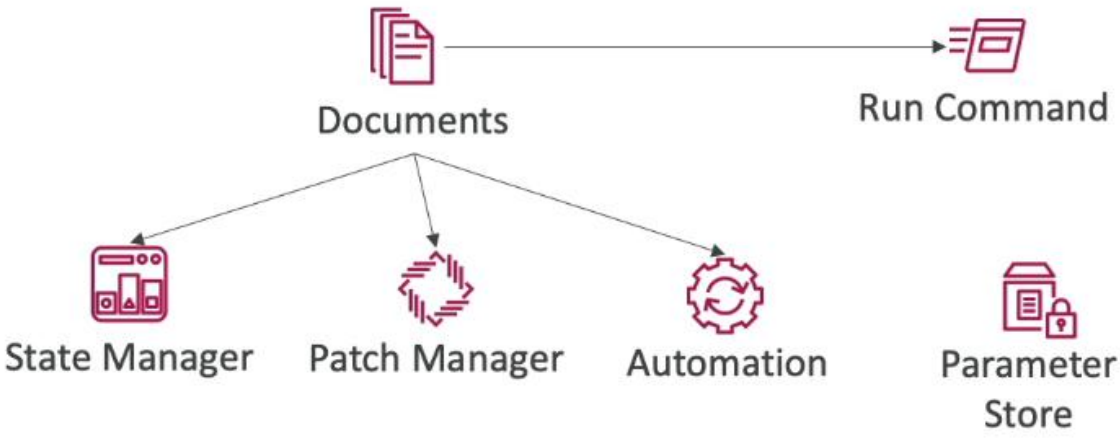
- Create, view, or manage logical group of resources – tags
- Allows creation of logical groups of resources, such as
  - Applications
  - Different layers of application stack

- Prod vs Dev
- Regional service
- Works with EC2, S3, DynamoDB, Lambda etc.



Documents:

- Predefined or custom scripts/commands for managing instances
- Written in JSON or YAML format
- Documents define the actions and parameters of Systems Manager operations

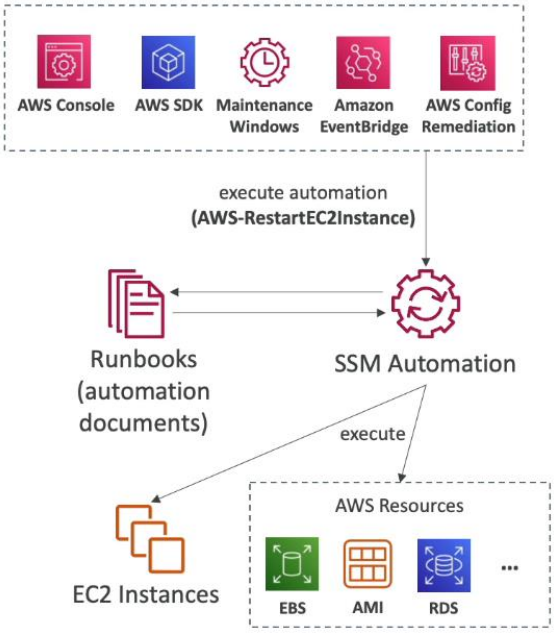


- Can be stored in Systems Manager service for reuse
- It enables automation, configuration changes, and diagnostic commands on managed instances
- Provides a standardized way to execute actions across multiple instances

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

Automation:

- Allows to create, schedule, and track the execution of workflows for managing instances and AWS resources
- Enables to automate common operational tasks – patching, installation, configuration updates
- Workflows are defined using YAML and JSON documents
- Automation documents specify the steps, parameters, and conditions for executing the workflow
- Provides built-in actions for interacting with AWS resources – executing scripts, running commands etc.
- Supports the use of approval steps, error handling, and parallel or sequential execution of steps
- Provides detailed logs and reports to track the execution and status of workflows
- Simplifies common maintenance and deployment tasks – restart instance, create an AMI, EBS snapshot etc.
- Automation Runbook – SSM Documents of type Automation
    - Defines action performed on AWS resources
    - Pre-defined runbooks or custom runbooks can be created
- Can be triggered as:
    - Manually using AWS console, AWS CLI, or SDK
    - By Amazon EventBridge
    - On a schedule using Maintenance Windows
    - By AWS Config for rules remediation



- Document Categories – remediation, patching, security, instance management, data backup, AMI management, resource management, cost management, self-support workflows etc.

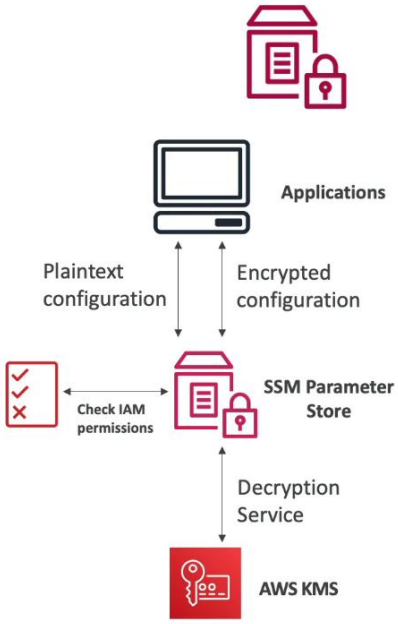
EC2 Run Commands:

- Allows to execute commands remotely on instances for tasks like software installations and troubleshooting
- Allow to execute commands remotely on EC2 instances and VMs without the need for SSH or RDP access
- Enables to automate common operational tasks, such as software installation, patch management, configuration updates, across multiple instances
- Provides a centralized management interface for executing commands across the fleet of instances
- SSM agent needs to be installed on all managed instances
- Commands can be issued using AWS Console, AWS CLI, AWS Tools for Windows PowerShell, Systems Manager API, or Amazon SDKs
- Commands are executed securely with IAM roles and permissions
- User can choose from set of predefined commands or create custom commands
- Output and error logs of the executed commands are captured and stored
- Also supports scheduling commands
- Commands can be targeted to specific instances based on tags
- Can be integrated with AWS Lambda, Steps Functions

Parameter Store:

- Allows to securely store and manage configuration data and secrets in a central location
- Serverless, scalable and durable

- Confidential information such as passwords, database connection strings, and license codes can be stored in SSM parameter store
- Can store values as plain text or can encrypt the data
- These values can be then referenced by using their names
- This service can be used with EC2, CloudFormation, Lambda, EC2 Run Command etc.

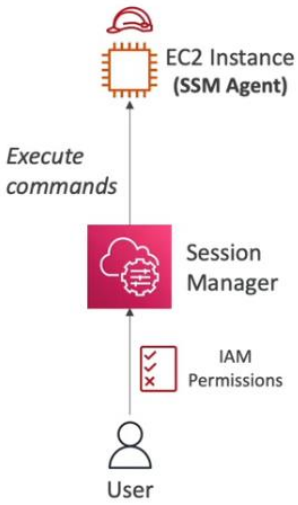


- Parameters can be organized hierarchically, allowing for logical grouping and easier management
- Keeps the track of parameter versions, enabling to easily revert to previous configuration if needed
- Also provides the audit trails for parameters changes
- Access to parameters can be controlled using IAM
- Supports automatic encryption of sensitive data using KMS
- Can be integrated with Systems Manager Automation
- Can be setup with Amazon Event Bridge for notifications
- Parameter Policies – allow to assign a TTL to a parameter for rotation or other purpose
    - Can assign multiple policies at a time
- Two tiers – standard and advanced

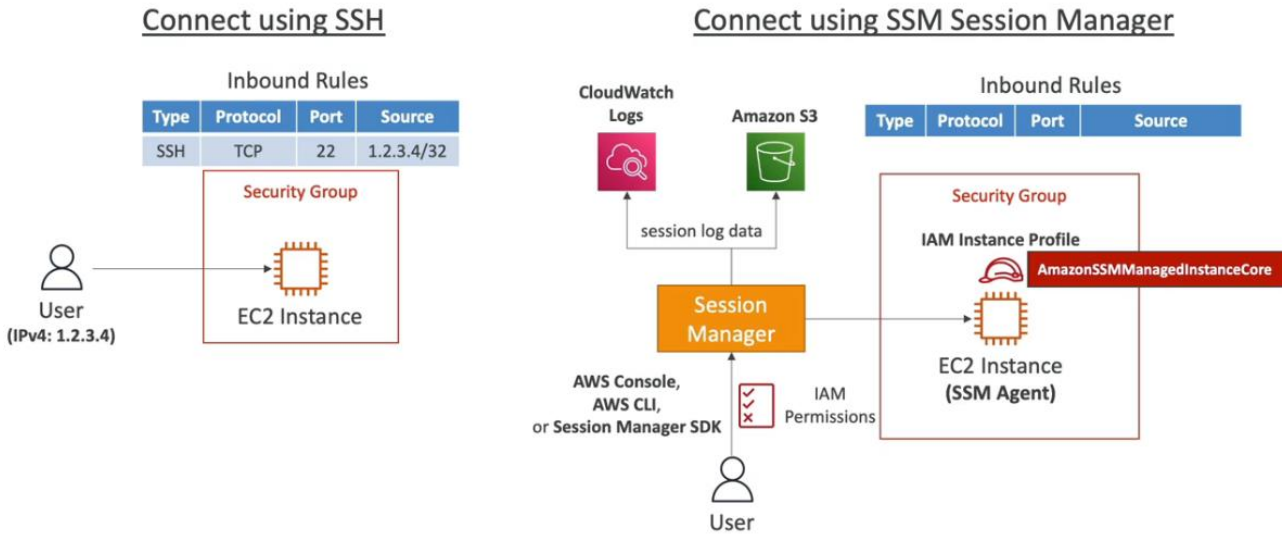
	Standard	Advanced
Total number of parameters allowed (per AWS account and Region)	10,000	100,000
Maximum size of a parameter value	4 KB	8 KB
Parameter policies available	No	Yes
Cost	No additional charge	Charges apply
Storage Pricing	Free	$0.05 per advanced parameter per month

Session Manager:

- A fully managed AWS service that provides secure and controlled shell access to EC2 instances
- It eliminates the need for SSH/RDP connections to EC2 instances and reduces the attack surface
- It uses the Systems Manager service to establish a secure connection to instances using an interactive shell session – using PowerShell or Bash. Console, CLI or SDK

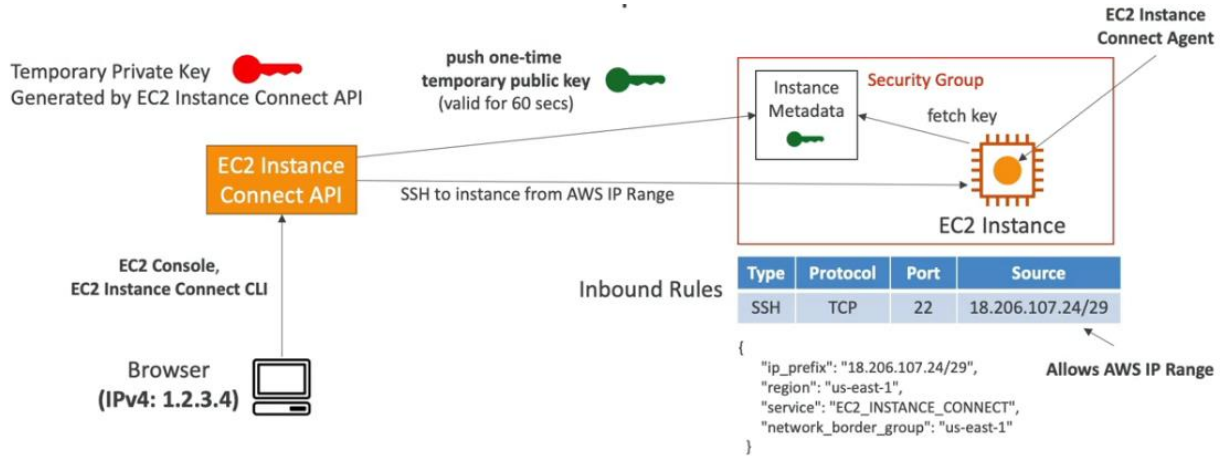


- It supports both Windows and Linux instances and provides a browser-based shell interface through the AWS Management Console
- It enables granular access control with fine-grained permissions using IAM policies
- It logs all session activity, making it easy to audit and monitor user access to instances
- It integrated with AWS CloudTrail for centralized logging and compliance
- It supports port forwarding, allowing users to securely access services running on instances without exposing them to the public internet
- It does not require inbound connections or open security group rules, providing an additional layer of security
- It works across multiple AWS accounts and regions, providing centralized access management for instances
- No need to manage SSH Keys, no Bastion Host to Manage



EC2 Connect:

- Example of browser based SSH connection to EC2 instance via EC2 connect agent



Inventory:

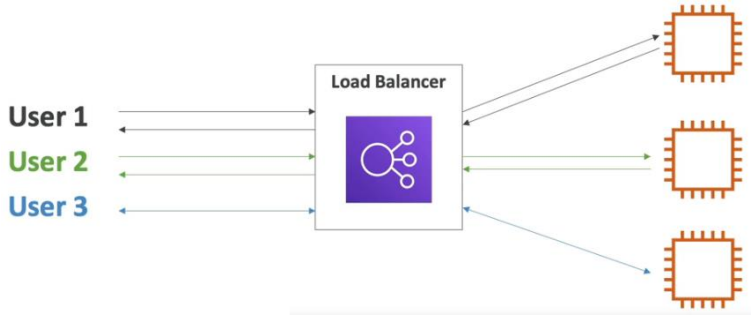
- Collects metadata about resources such as EC2 instances, S3 buckets, RDS instances, and more
- Provides an automated and centralized way to track and manage resource configurations and attributes
- Collected metadata includes information like resource ID, tags, attributes, OS, installed software, installed updates, running services and custom inventory items
- Inventory collections can be scheduled at regular intervals or on-demand
- Collected data is stored in S3 buckets or can be sent to AWS config
    - Can be queried via AWS Athena or QuickSight
    - Can be queried from multiple AWS accounts and regions
- Inventory features can be accessed via AWS CLI, SDKs, or Systems Manager APIs
- Helps in auditing, change tracking, resource configuration reporting
- Supports both – EC2 or on-premises servers

State Manager:

- Defines and maintains the desired configurations for AWS resources
- Uses JSON or YAML documents called State Manger documents – to create an association
- Automatically applies configurations to keep resources in the desired state
- Supports different modes for managing configurations
- Works with managed instances and Systems Manager agents
- Integrates with other AWS services – CloudFomration, OpsWorks, Elastic Beanstalk
- Track execution status of each configuration run and provides logs and reports
- Supports parameterization and dynamic values – reusable and customizable configurations
- Enable defining dependencies between resources – ordering and sequencing
- Can be managed using AWS APIs, CLI or SDKs

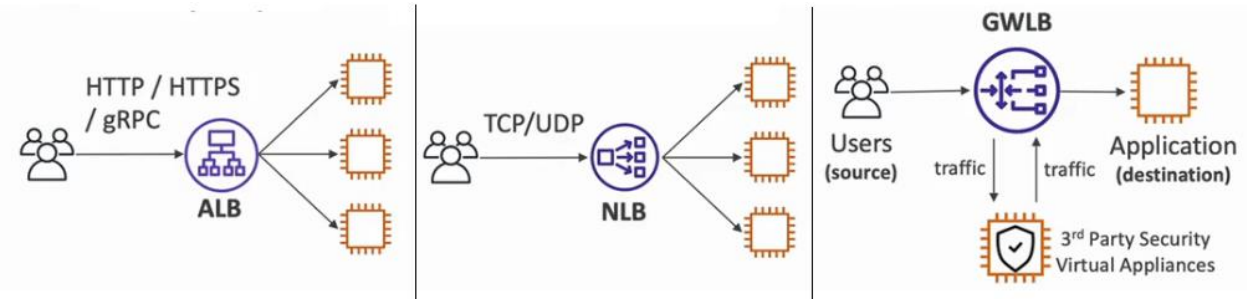
Elastic Load Balancers:

- Distribute incoming application or network traffic across multiple targets (such as EC2 instances, containers, or IP addresses) to ensure high availability and scalability
    - High availability across zones



- Expose a single point of access (DNS) to user's application
- They support features like automatic scaling, SSL/TLS termination, health checks, and session persistence
- Distribute traffic based on configuration rules, such as round-robin, least connections, or based on specific attributes like URL path or host header (ALB)
- They can handle millions of requests per second and automatically scale to accommodate increased traffic
- Help improve fault tolerance by distributing traffic across healthy instances and automatically detecting and routing around failed targets
- They provide advanced health checks to monitor the health of target instances and route traffic only to healthy targets
- Can be configured to support sticky sessions, allowing a user's requests to be consistently directed to the same target for session persistence
- They integrate with other AWS services like Auto Scaling, enabling dynamic scaling based on demand

- Load Balancers support IPv4 and IPv6 traffic can be associated with Amazon Route53 to provide DNS-based load balancing
- AWS takes care of upgrades, maintenance, high availability – provides only few configuration's knobs
- Four types of Elastic Load Balancers – CLB, ALB, NLB, GWLB



Classic Load Balancers:

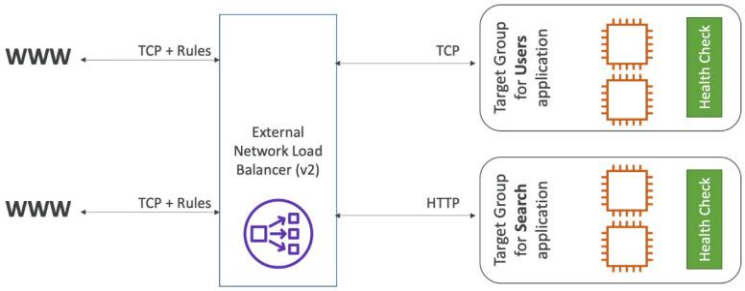
- Operates at Transport Layer – Layer 4 & 7 (retired in 2023)
- Distributes traffic across EC2 instances based on availability zones
- Supports TCP and SSL\TLS protocols

Application Load Balancers:

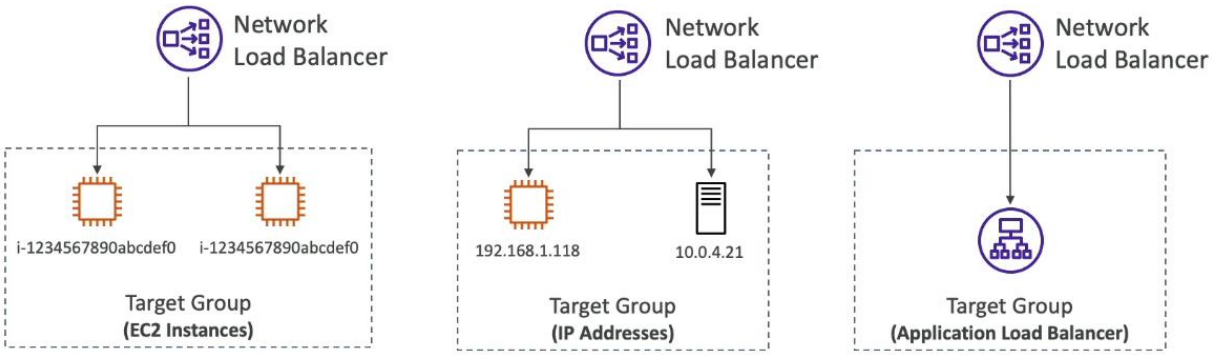
- Operates at Application Layer – Layer 7
- Provides advanced routing capabilities
- Supports HTTP, HTTPS, and WebSocket protocols
- Can route traffic based on URL path, host header, or query string parameters – http routing features

Network Load Balancers:

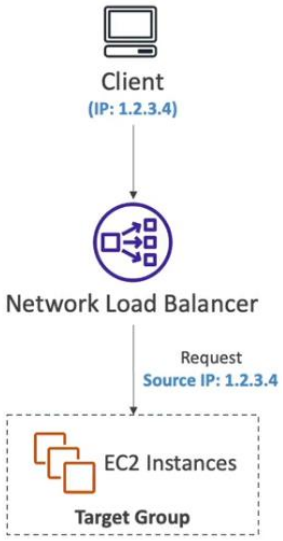
- Operates at Transport Layer – Layer 4
- Designed to handle high volumes of traffic with ultra-low latency – millions of requests per second
- Supports TCP, UDP, and TLS protocols
- Ideal for TCP-based applications that require extreme performance and scalability
- Static IP through Elastic IP
    - One static IP per AZ – helpful for whitelisting specific IP
- Health Checks support the TCP/HTTP/HTTPS protocols



- Not included in AWS Free Tier
- Target groups of NLB can be
    - EC2 instances
    - IP addresses – must be private IPs
    - Application Load Balancer



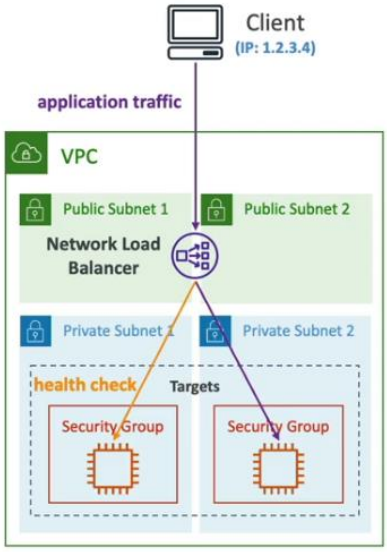
- Client IP Preservation – client IP addresses are forwarded to NLB targets
    - When disabled, the private IP address of the NLB becomes the client IP address for all incoming traffic



    - NLBs do not have associated security groups
        - Client's security groups can't be used as a source in the target's security groups
        - Therefore, target's security groups must use the IP address of the clients to allow the traffic
    - Recommended Security group rules with Client IP Preservation *Enabled*

Inbound Rules

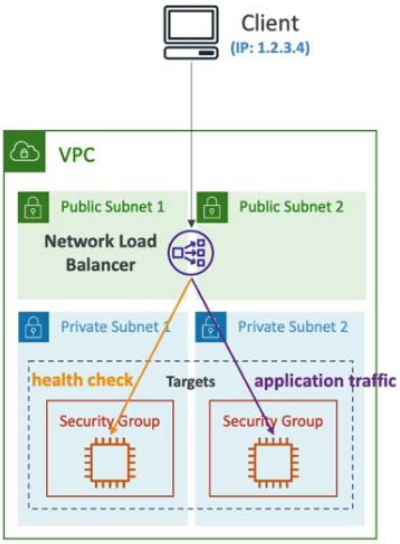
Source	Protocol	Port Range	Comment
Client IP / VPC CIDR	Listener	Listener	Allow traffic from your application, your network, Internet
VPC CIDR	Health Check	Health Check	Allow health check traffic from the Network Load Balancer



- Recommended Security group rules with Client IP Preservation *Disabled*

Inbound Rules

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



- Recommended NACLs inbound and outbound rules configurations for NLB

NLB Subnet NACLs

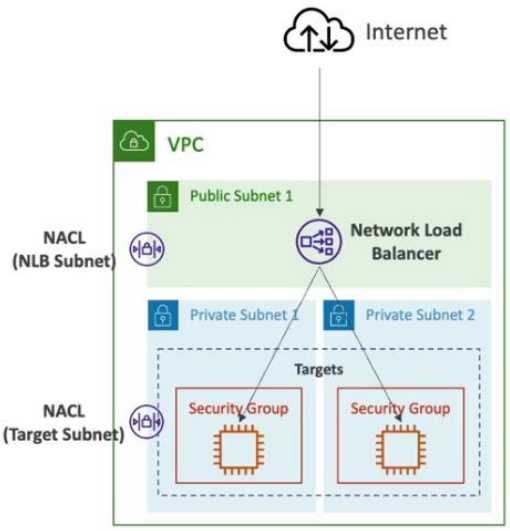
	Source	Protocol	Port Range	Comment
Inbound	Client IP / VPC CIDR	Listener	Listener	Allow traffic from Client and Internet
Inbound	VPC CIDR	Health Check	1024 – 65535	Allow Health Check traffic

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

Targets Subnet NACLs

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

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

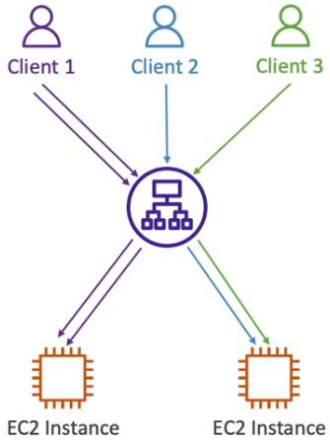


Gateway Load Balancers:

- Operates at Network Layer – Layer 3
  - GENEVE Protocol on IP Packets
- Provides load balancing benefits to third-party virtual appliances
- Supports both NAT and Transparent modes
- Take care of routing traffic to the appropriate virtual appliance in the user's network, instead of traffic directly going to the virtual appliances
  - Virtual appliances such as firewalls, IDS, and deep packet inspection systems
  - Traffic is routed to healthy virtual appliances and routed away from failing ones

Sticky Sessions:

- Also known as session affinity - enables ELB to bind a user's session to specific instance behind the ELB
- Subsequent requests from the same user are sent to the same instance – ensuring session consistency
  - User doesn't lose his session data
- Works for CLB, ALB and NLB
- Can be configured at the load balancer level using cookies or through the load-balancer-generated *x-forwarded-for* http header
  - Application based cookies – generated by target and generated by LB
  - Load balancer-based cookies – generated by LB, duration-based cookies
    - The cookie used for stickiness has an expiration date user control
  - NLB works without cookies

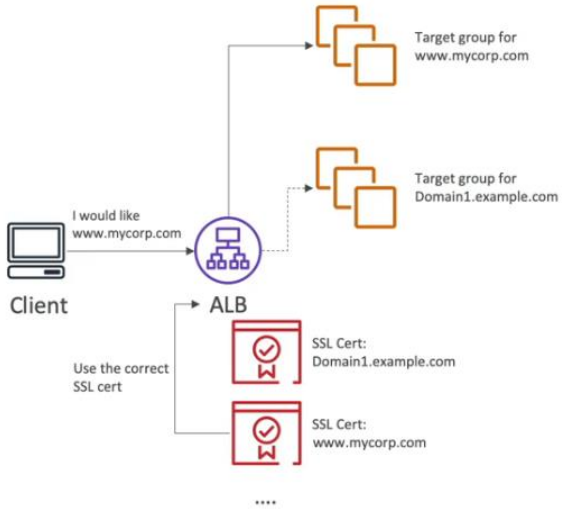


- Enabling stickiness may bring imbalance to the load over the backend EC2 instances

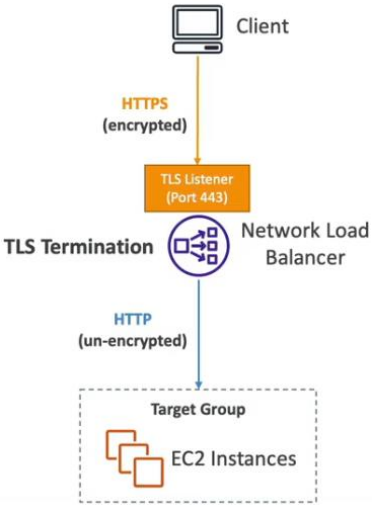
ELB SSL Certificates:

- The load balancer uses an X.509 certificate – SSL/TLS certificate
- Can be managed via AWS Certificate Manager
- Users can create upload their own certificates immediately

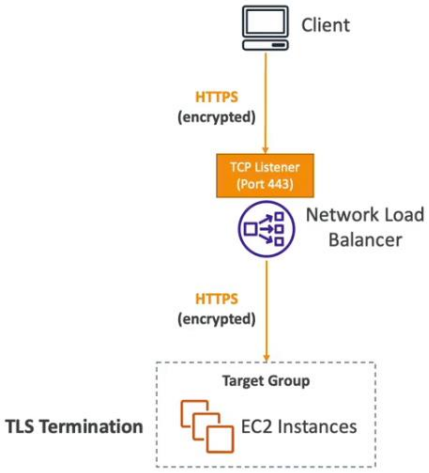
- Server Name Indication (SNI) – solves the problem of loading multiple SSL certificates onto web server – to server multiple website
    - New protocol and requires client to indicate the hostname of the target server in the initial SSL handshake
    - The server will then find the correct certificate or return the default one
    - Only works for ALB and NLB, CloudFront – does not work with CLB



- Classic Load Balancer – supports only one SSL certificate
    - Must use multiple CLBs for multiple hostnames with multiple SSL certificates
- Application Load Balancer – supports multiple listeners with multiple SSL certificates
    - Use SNI to make it work
- Network Load Balancer – supports multiple listeners with multiple SSL certificates
    - Use SNI to make it work
    - NLB uses a TLS Certificate to terminate and decrypt the frontend connections before sending them to targets



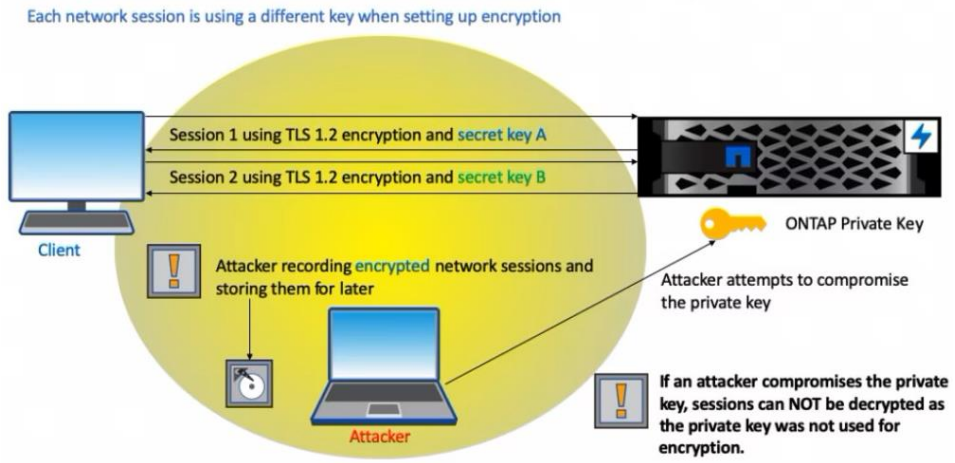
    - NLBs do not support TLS re-negotiation or mutual TLS authentication – mTLS
        - Use TCP listener on port 443 to pass encrypted traffic to the targets without the NLB decrypting it – do not use TLS listener
        - The targets must be able to decrypt the traffic
        - TCP listeners also support mTLS, because the NLB passes the request as is, and mTLS encryption can be implemented on targets



- HTTPS/SSL Listener – Security Policy
  - A combination of SSL protocols, SSL ciphers, and Server Order Preference option supported during SSL negotiations
  - Predefined Security Policies – ELBSecurityPolicy-2016-08 etc.
  - For ALB and NLB
    - Frontend connections – predefined security policies can be used
    - Backend connections – ELBSecurityPolicy-2016-08 security policy is always used

Perfect Forward Secrecy:

- Enhances encryption security in SSL/TLS communications
- Uses unique session keys for each session negotiated between the client and ALB
- Derives session keys using the Diffie-Hellman key exchange algorithm
- Prevents past communications from being decrypted if the private key is compromised in future
- Provided enhanced forward secrecy by using ephemeral session keys not linked to the ALB's long-term private key
- Mitigates the impact of potential future private key compromises

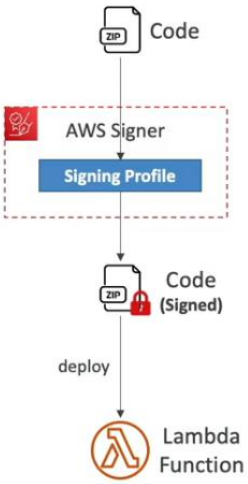


AWS Signer:

- Enables the signing of code using various signing algorithms – RSA, ECDSA, HMAC etc.
  - To ensure trust and integrity of code
- Utilizes certificates managed by ACM for code signing
  - Manages code-signing certificate public and private keys
- Supports the creation of signature profiles that define the signing configurations and policies
- Code is validated against a digital signature to confirm that the code is unaltered and from a trusted publisher
- The signing process is auditable and tamper-evident

Integrations:

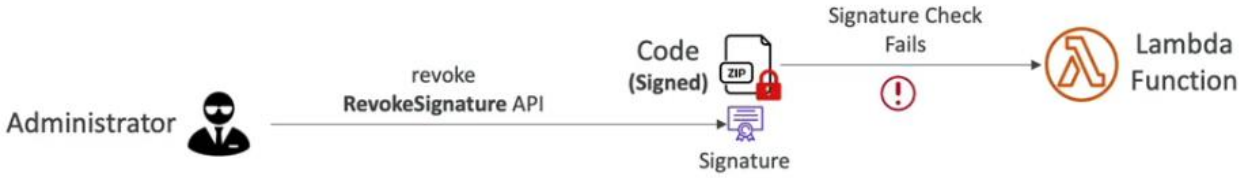
- Integrates with other AWS services – Lambda, CodePipeline etc. for automated code signing workflows
- Code Signing for AWS Lambda – digitally sign code packages for Lambda
  - Enforce only trusted code runs in Lambda functions
  - Not supported for Container Lambda functions



- Code Signing for AWS IoT – sign code that is created for AWS IoT and Amazon FreeRTOS
  - Integrated with ACM to generate/import certificates

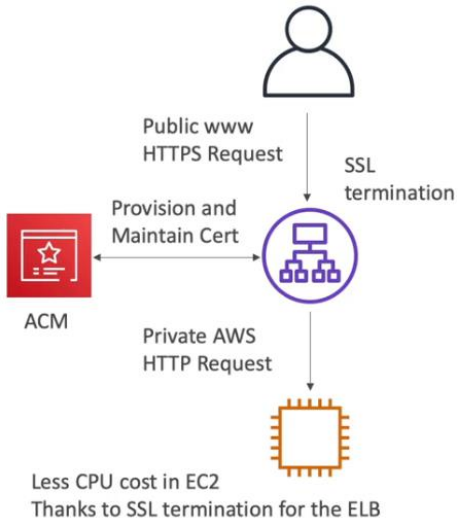
Revoking Signing Profile:

- Either revoke individual signatures using *RevokeSignature* API call
- Or revoke a Signing Profile using *RevokeSigningProfile* API call
- For example, for lambda functions, user can revoke the signature of Lambda deployment package – invalidate it
  - Will cause it to fail Lambda signature checks



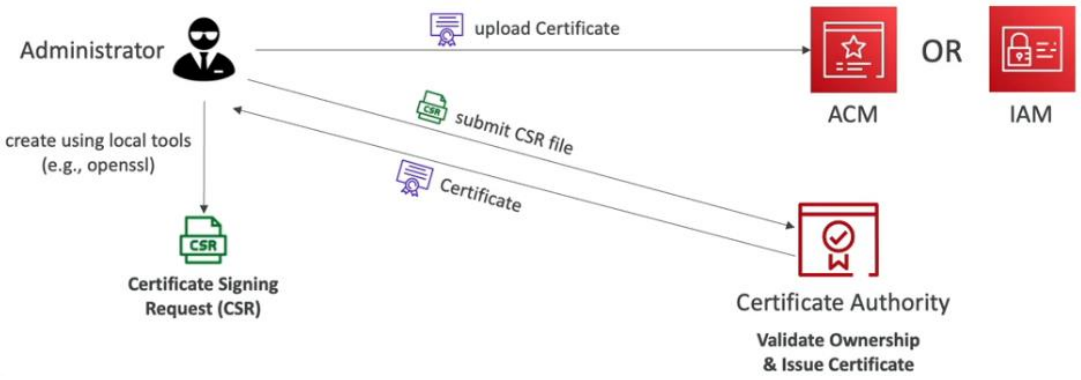
AWS Certificate Manager:

- A managed service for SSL/TLS certificate management
- Automates the provisioning and renewal process, ensuring certificates are always up to date – monitors the certificate expiration
- It integrates seamlessly with AWS services like ELB, CloudFront, Elastic Beanstalk, API Gateway, and more
- ACM offers free SSL/TLS certificates for use with AWS services, reducing costs
- It provides a managed private certificate authority (CA) for issuing internal certificates
- ACM simplifies SSL/TLS certificate management, freeing user from manual workflows
- ACM does not provide a direct export feature for certificates managed for use within AWS service



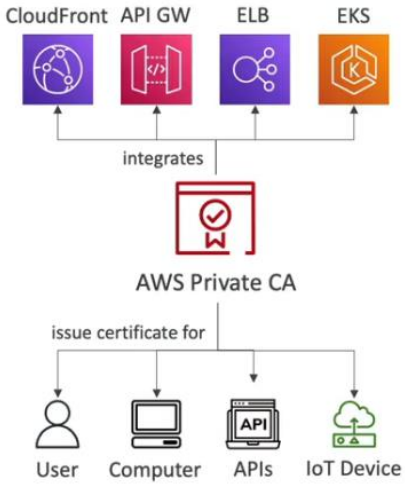
- Certificates provisioned and managed by ACM are intended for use within AWS services
- ACM stores the private keys for certificates in a highly secure and scalable infrastructure
- The private keys are encrypted using HSMs, which provide strong cryptographic protection
- ACM securely stores the certificates in the AWS region where they were issued – it's a regional service
- The certificates are automatically replicated across multiple AZs to ensure durability and availability – certs cannot be copied across regions
- ACM certificates can be used only with supported AWS services such as ELB, CloudFront, API Gateway, and so forth

- Use ACM certificates to enable HTTPS at the load balancer
    - For the webserver, AWS recommends that you use a self-signed certificate
    - As the communication between ELB and the webserver is in a private network, there is very little risk of man-in-the-middle type attacks, and ELB also does not check the validity of certificates
- It integrates with IAM to provide fine-grained access control to certificates
- You can import private certificates into AWS Certificate Manager (ACM) and assign them to all the same resources you can with generated certificates, including an Application Load Balancer
- Users can create certificate manually – the upload the certificate to either ACM or IAM



ACM Private Certificate Authority:

- ACM Private CA allows to create and manage a private CA hierarchy (including root and subordinates CAs) within AWS providing enhanced security for internal applications by allowing user to issue private certificates, reducing the reliance on external CAs
- Certificates are trusted only by customer's organization – not public internet
- Integrates with EKS and with any AWS service that is integrated with ACM

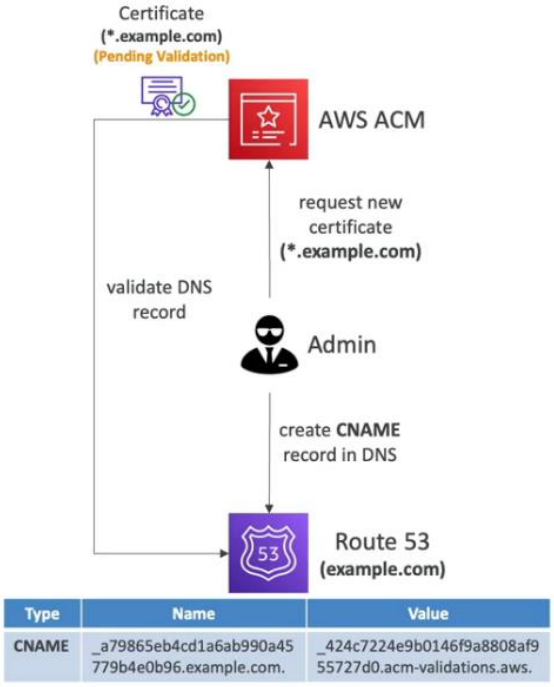


- Use cases of using AWS Private CA
    - Encrypted TLS communication, cryptographically signing code
    - Authenticate users, computers, API endpoints, and IoT devices

- Enterprise customers building a Public Key Infrastructure – PKI

ACM – Validation Techniques:

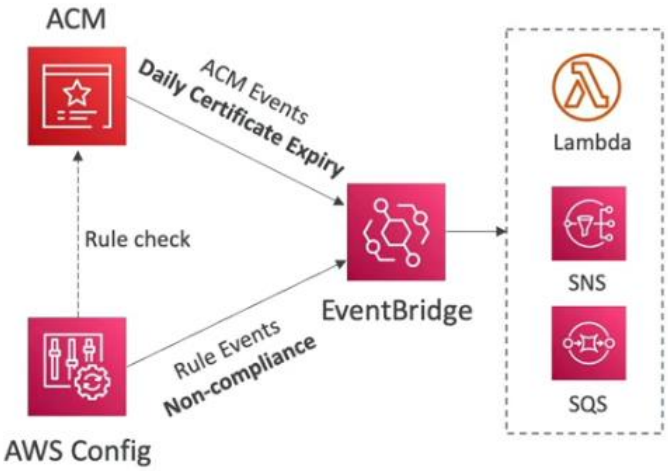
- Before ACM issue a public certificate, users must prove that they own/control the domain
- DNS Validation – recommended
  - Leverage a CNAME record created in DNS config – route53 etc.
  - Preferred for automatic renewal purposes
  - Takes a few minutes to verify



- Email Validation – a validation email is sent to contact addresses in the WHOIS database
  - Takes a few minutes to verify
- ACM validation is not required for imported certificates or certificates signed by a private CA

Monitor Expired Imported Certificates:

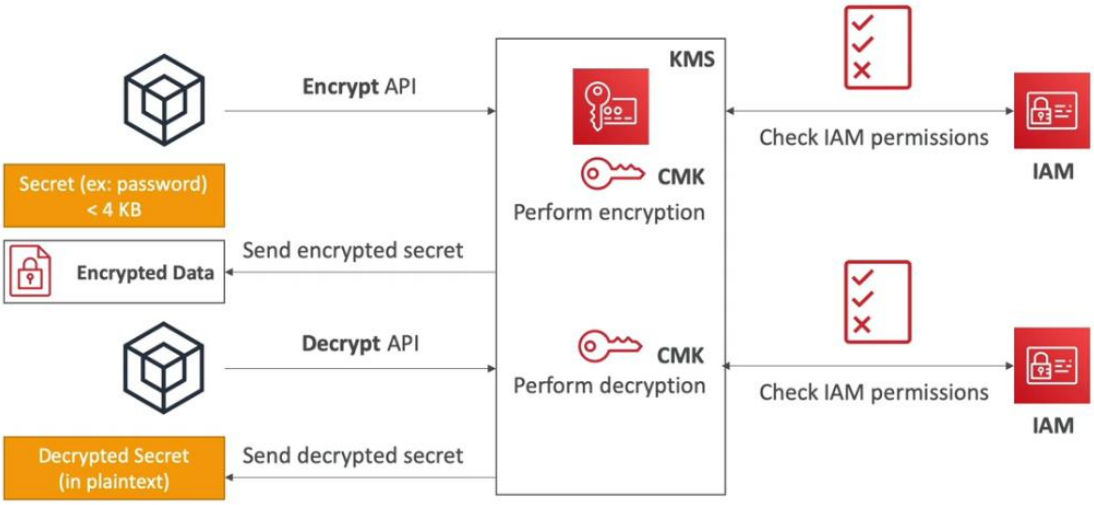
- ACM sends daily expiration events – starting 45 days prior to expiration
  - The no. of days can be configured
  - Events are appearing in EventBridge
- AWS Config has a managed rule named *acm-certificate-expiration-check* to check for expiring certificates – configurable no. of days



Amazon KMS:

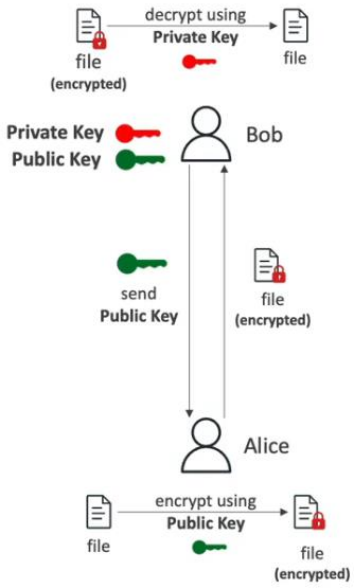
- A managed service that makes it easy and secure to create and control the encryption keys used to encrypt data
- Fully integrated with IAM for authorization
- AWS KMS is integrated with other AWS services, including EBS, S3, RedShift, Elastic Transcoder, WorkMail, RDS, SSM etc.
- Can also be used via CLI or SDK
- Customer is responsible for the key material's overall availability and durability
- KMS can be used to encrypt EBS volumes, but KMS cannot be used to generate a public/private key to log in to EC2
- Public keys can be imported into EC2 Key Pairs, but EC2 key pairs cannot be used to encrypt EBS volumes – Must use KMS or third-party tools/applications
- Encryption keys cannot be copied to another region – can copy AMIs from one region to another region encrypted but key in destination region will be used for encryption
- KMS can be used to encrypt EBS volumes, and it is possible to encrypt root device volumes
- To encrypt a root device volume, create an AMI. The initial AMI will be unencrypted, but then it can be encrypting copying this volume
- Encryption keys can be changed from amazon managed to customer master keys
- Public keys can be viewed using ec2 instance metadata (*curl 169.254.169.254/latest/metadata/*) as well as in the *ec2-user/.ssh/authorized_keys*
- Multiple public keys can be attached to an EC2 instance
- Roles can be added to existing EC2 instances
- Deleting a key pair in the console will not delete it from the instance or the instances metadata
- Keys cannot be exported from KMS so can't SSH to EC2, but it can be exported via CloudHSM

How does KMS work?

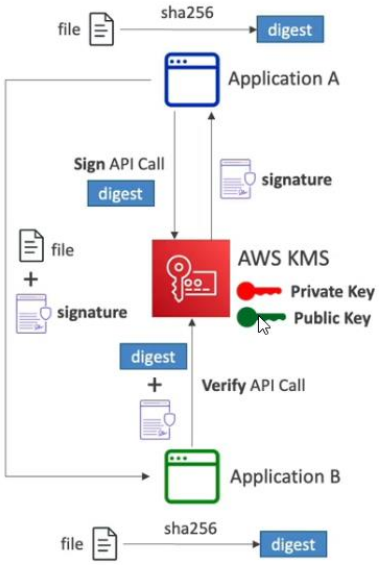


KMS Key Types:

- KMS Key types – Symmetric and Asymmetric
  - Symmetric – AES-256 keys
    - First offering of KMS, single encryption key that is used to Encrypt and Decrypt
    - AWS services that are integrated with KMS use Symmetric KMS keys
    - Necessary for envelope encryption
    - User never get access to the KMS key unencrypted – must call KMS API to use
  - Asymmetric – RSA & ECC key pairs
    - Public (encrypt) and Private (decrypt) key pairs
    - Used for encrypt/decrypt, or sign/verify operations
    - The public key is downloadable, but user can't access the Private key unencrypted – private key never leaves AWS KMS unencrypted



- Use case – encryption outside of AWS by users who can't call the KMS API
- KMS supports 3 types of asymmetric KMS keys:
  - RSA KMS Keys – encryption/decryption or signing/verification
  - Elliptic Curve (ECC) KMS Keys – signing/verification
  - SM2 KMS keys – China regions only
- Digital Signing with KMS Asymmetric – helps to verify the integrity of message or files sent across different systems
  - Verify that file has not been tampered with in transit
  - Public Key – used to verify the signature value
  - Private Key – used in the signing process
  - Use cases could be like:
    - Document e-signing
    - Secure messaging
    - Authentication/authorization tokens
    - Trusted source code
    - E-commerce transactions



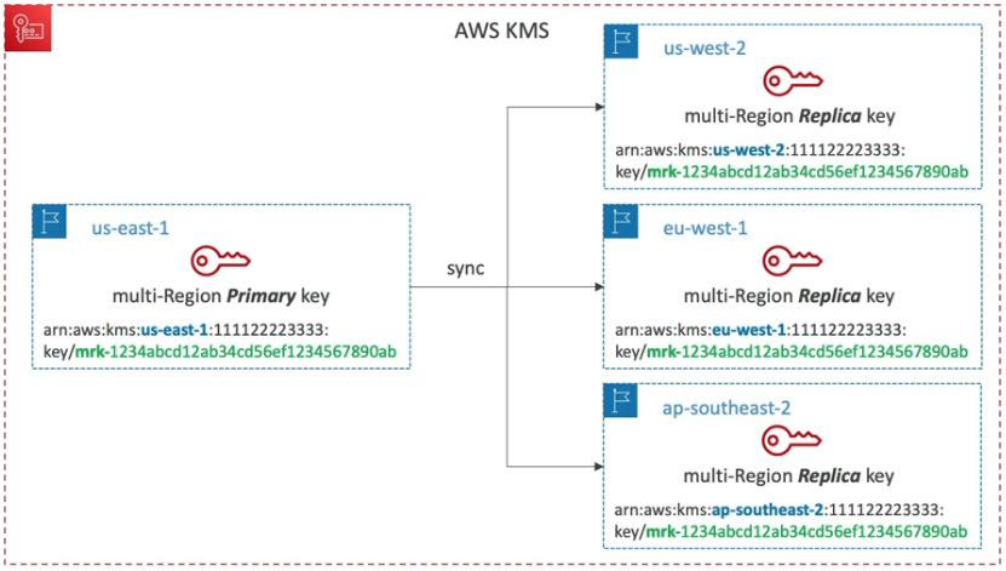
- Types of KMS keys
  - Customer Managed Keys – create, manage, and use, can enable, or disable
    - Possibility of rotation policy – new key generated every year, old key preserved
    - User can add a Key Policy (resource policy) and audit in CloudTrail
    - Leverage for envelope encryption
  - AWS Managed Keys – used by AWS service (S3, EBS, Redshift etc.)
    - Managed by AWS – automatically rotated every 1 year
    - User can view Key Policy and audit in CloudTrail
  - AWS Owned Keys – created and managed by AWS
    - Use by some AWS services to protect user's resources
    - Used in multiple AWS accounts, but they are not in user's AWS account

- User can't view, use, track, or audit

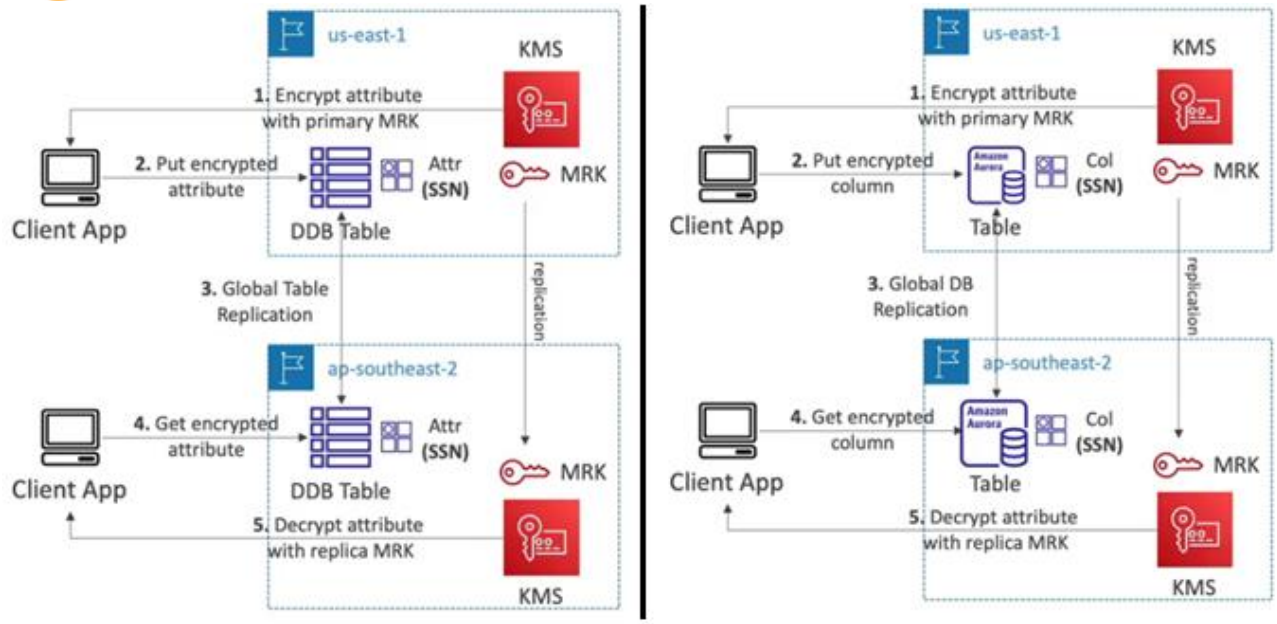
KMS Key	Customer Managed Key	AWS Managed Key	AWS Owned Key
Can view metadata?	✔	✔	✘
Can manage?	✔	✘	✘
Used only for my AWS account?	✔	✔	✘
Automatic Rotation	Optional (every 1 year)	Required (every 1 year)	Varies

KMS Multi-Region Keys:

- A set of identical KMS keys in different AWS regions that can be used interchangeably – same KMS keys in multiple regions
- Encrypt in one region and decrypt in other regions – no need to re-encrypt or making cross-region API calls
- Multi-region keys have the same key ID, key material, automatic rotation
- KMS multi-region are not global – primary + replicas
- Each multi-region key is managed independently
- Only one primary key at a time, can promote replicas into their own primary

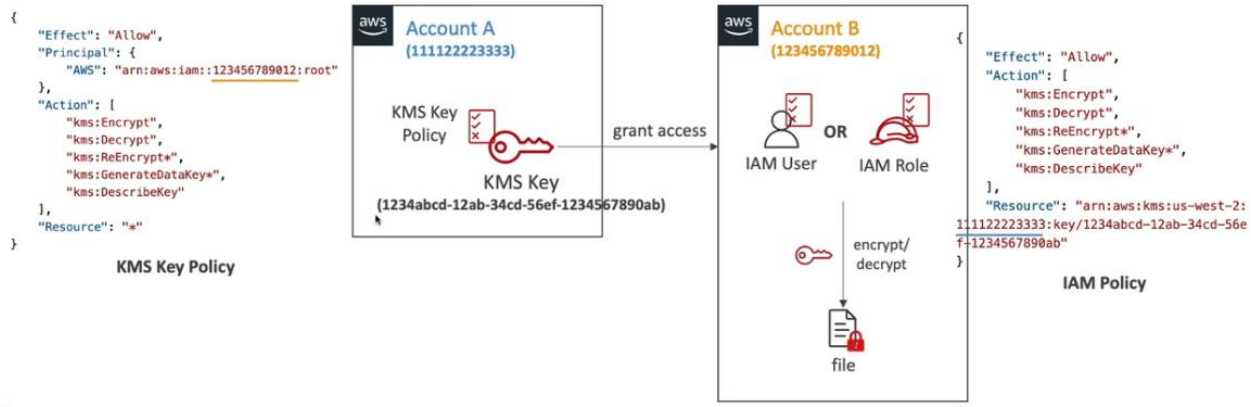


- Use cases – disaster recovery, global data management (DynamoDB global tables), global client-side encryption, global aurora, active-active applications that span multiple regions, distributed signing applications etc.

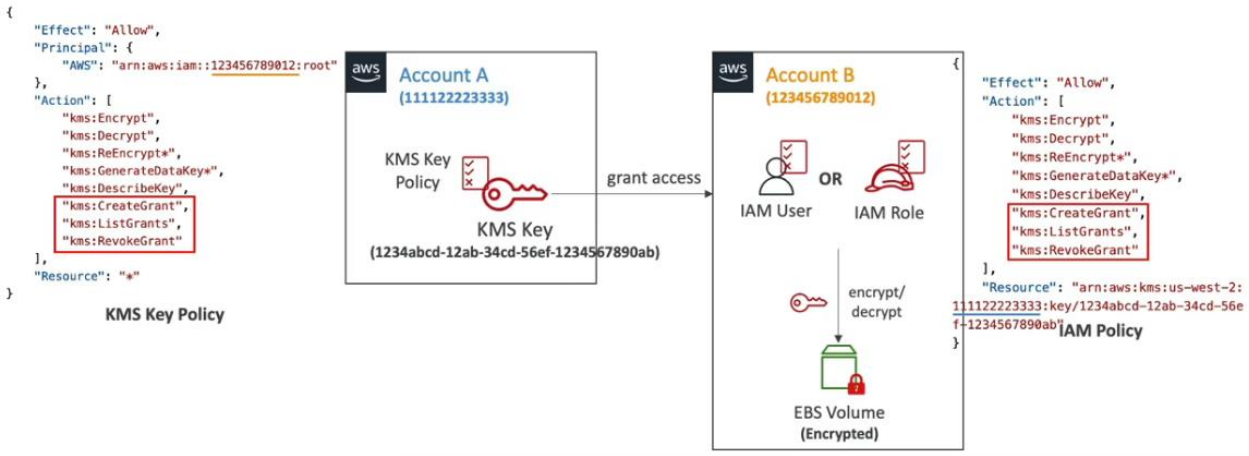


KMS Cross Account Access:

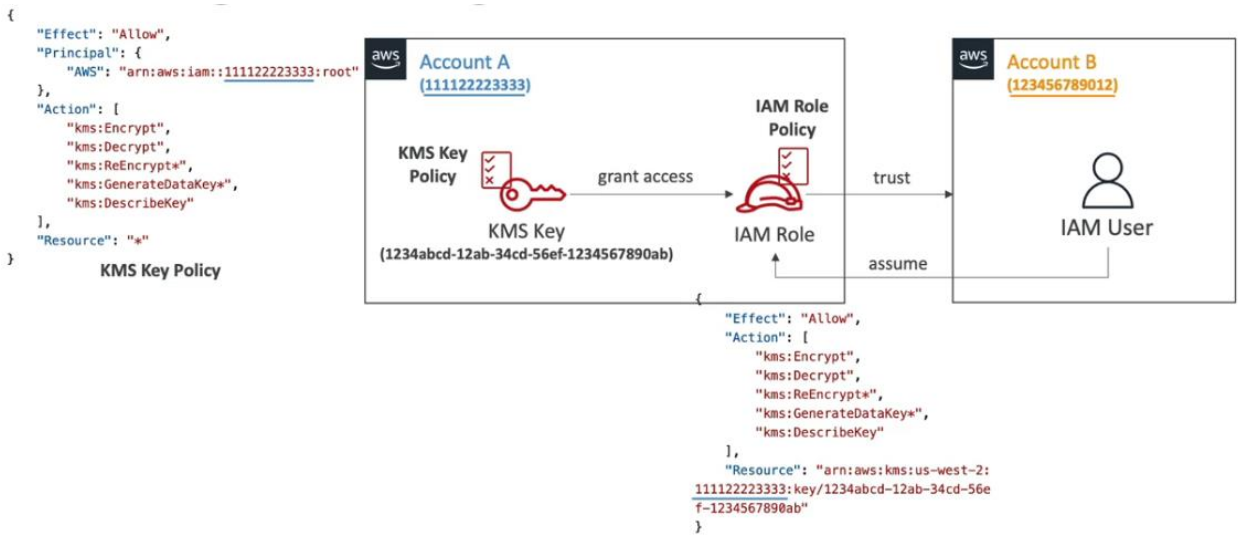
- Access to KMS is controlled by:
  - They Key Policy – add the root user, not individual IAM users/roles
  - IAM Policies – define the allowed actions and the CMK ARN



- To enable another external account to encrypt or decrypt using your CMK, cross account access needs to be enabled first
  - Enable access in the Key Policy for the account which owns the CMK
  - Enable access in the IAM Policy for the external account

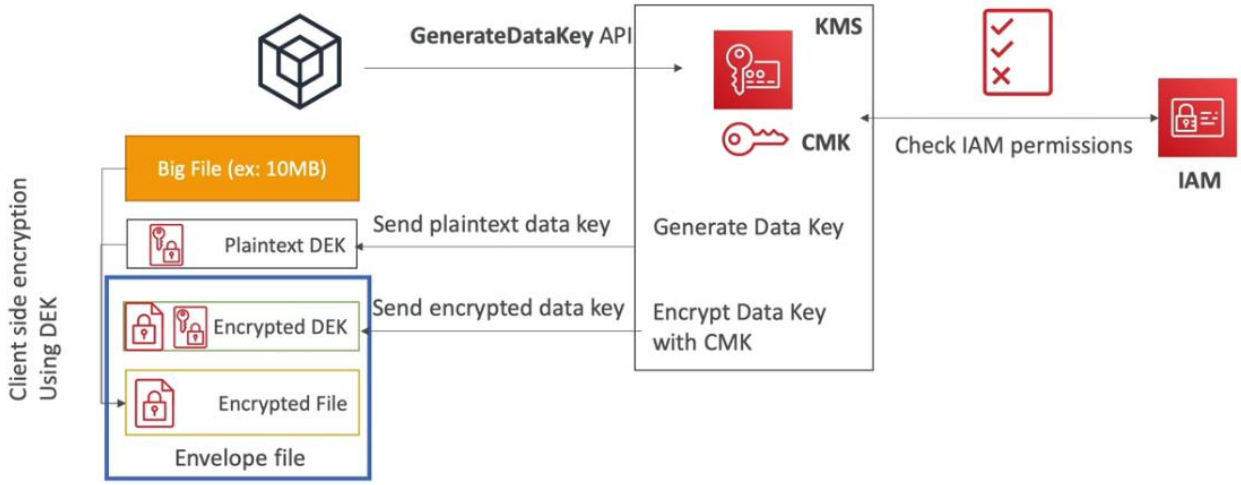


- Using KMS Key for Cross account access through assuming an IAM role

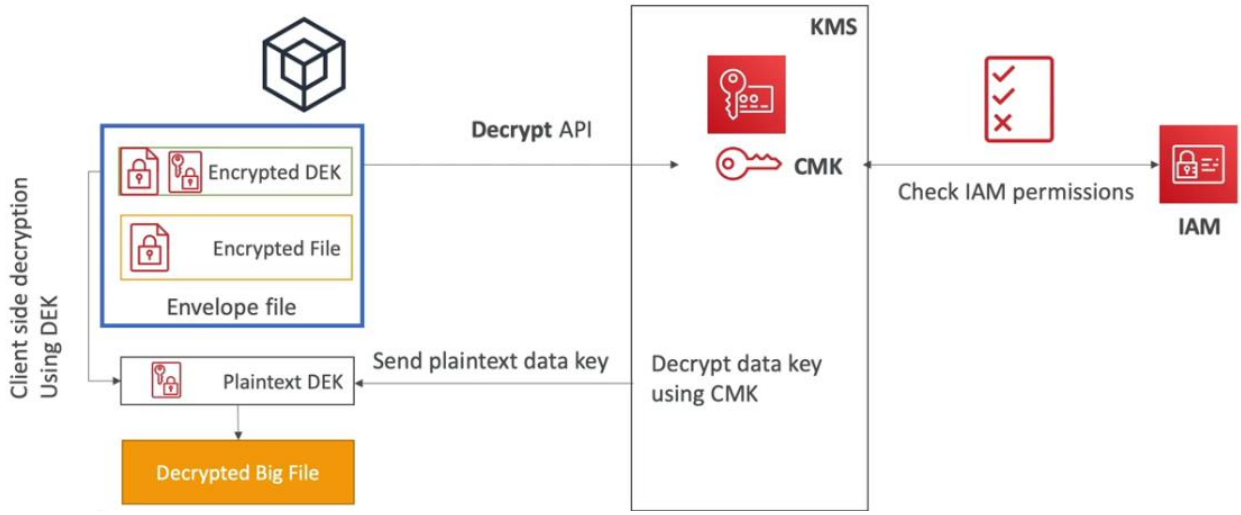


Envelope Encryption:

- KMS *Encrypt* API call has limit of 4KB – to encrypt more than 4KB, we need to use Envelope Encryption
- The main API call will be *GenerateDataKey* API for encryption



- For decryption the API call will be simply *Decrypt* API



Customer Master Key (CMK):

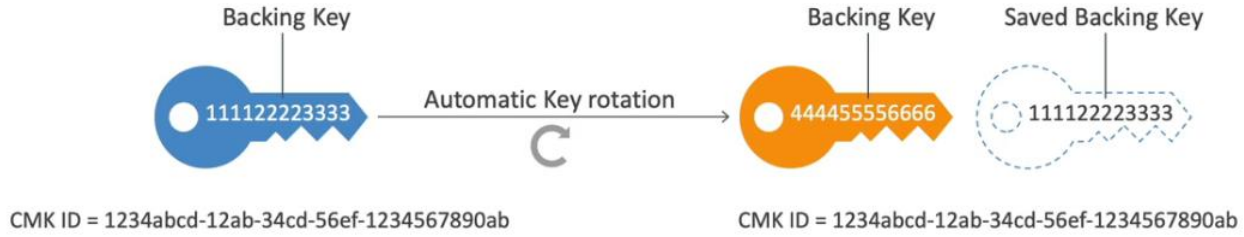
- Consists of the followings:
  - Alias
  - Creation date
  - Description
  - Key state
  - Key material (customer provided or AWS provided)
- Can never be exported
- Two types:
  - AWS-managed CMK for each service that is integrated with AWS KMS
  - Customer-managed CMK that is generated by providing AWS with key material
- When a key material is imported into a CMK, the CMK is permanently associated with that key material – can reimport same key material but can't import different key material into that CMK
- Cannot enable automatic key rotation for a CMK with imported key material

- While each CMK is individually region bound, the setting the key as a multi-region CMK enables the key object to be replicated to the other region (in the same AWS partition) with the same content
    - A partition is a group of AWS Regions
- When data is encrypted under KMS CMK, the ciphertext cannot be decrypted with any other CMK – not even if same key material is imported into a different CMK
- After a CMK is deleted, you can no longer decrypt the data that was encrypted under that CMK, which means that data becomes unrecoverable
- Setup a Customer Master Key:
    - Create Alias and Description
    - Choose material option
    - Define Key Administrative Permissions
        - IAM users/roles that can administer (but not use) the key through the KMS API
    - Define Key Usage Permissions
        - IAM users/roles that can use the key to encrypt and decrypt the key
- Key material options:
    - Use KMS generated key material
    - Customer's own key material
- How to import customer's own key material:
    - Create a CMK with no key material
    - Download a public key (wrapping key) and import token
    - Encrypt the key material
    - Import the key material
    - Consideration for Imported Key Material:
        - Availability and durability are different
        - Secure key generation is up to customer
        - No automatic rotation
        - Ciphertext are not portable between CMKS

KMS Key Rotation:

- Extensive re-use of encryption keys is not recommended
- It is best practice to rotate keys on a regular basis
- AWS Managed Keys:

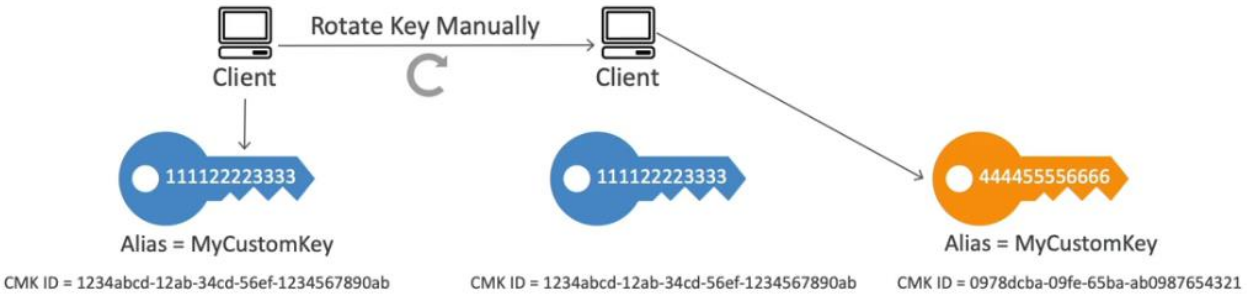
    Automatic:
    - AWS KMS automatically rotates AWS managed keys every year
    - When the CMK is due for rotation, a new backing key is created and marked as the active key for all new requests
    - The old backing key remains available to decrypt any existing cyphertext files that were encrypted using the old key
    - New key has the same CMK ID – only the backing key is changed



- AWS handles everything for customer
- Customer cannot manage rotation itself
- AWS Managed keys cannot be deleted

Manual:

- When user wants to rotate key every 90 days, 180 days etc.
- New key has a different CMK ID
- Previous key should be kept so the old data can be decrypted
- Better to use aliases in this case – to hide the change of key for the application
- Good solution to rotate CMK that are not eligible for automatic rotation – like asymmetric CMK

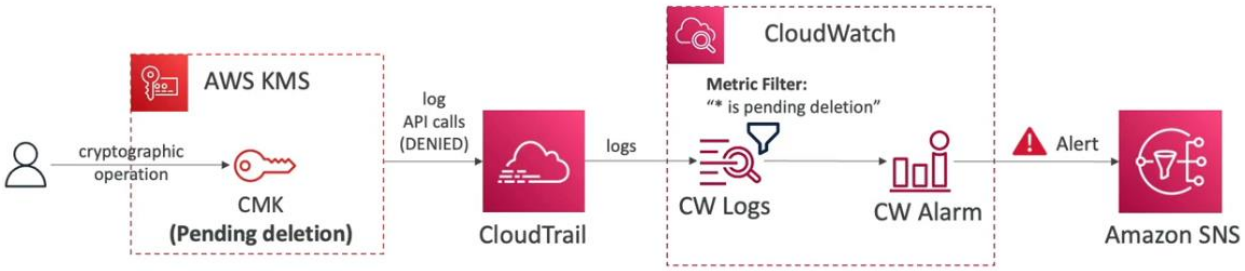


- Customer Managed Keys:
  - Once a year automatically – disabled by default
  - AWS KMS generates new cryptographic material for the CMK every year
  - The CMK's old backing key is saved so it can be used to decrypt data that is previously encrypted
  - Can be rotated On-demand manually
    - Create a new key, then change applications or aliases to use new CMK
  - Customer controls the rotation frequency
  - Keys can be deleted
- Customer Managed Keys with Imported Key Material:
  - Automatic key rotation is not available for CMKs with imported key material – since the CMK was not generated in AWS
  - They only option is to rotate the key manually
    - Create a new key, change the applications or aliases to use new CMK
  - Customer controls the rotation frequency
  - Keys can be deleted

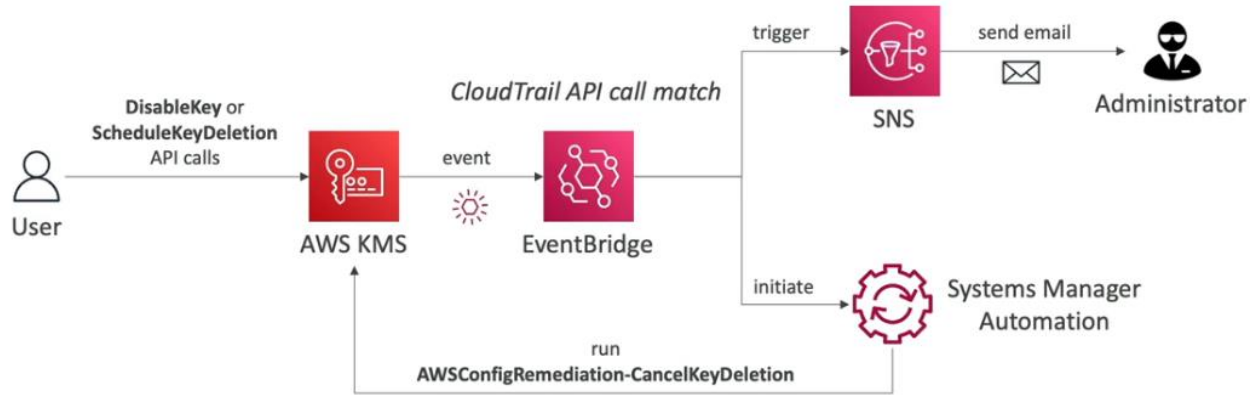
AWS Managed	Customer Managed	Customer Managed, imported key material
Automatically rotates every year	Automatic rotation every 365 days (disabled by default)	No Automatic rotation
You cannot rotate manually	You can rotate manually	You must handle the rotation manually
AWS manages it all for you and saves the old backing key	Create a new CMK and update your applications or Key Alias to use the new CMK	Create a new CMK and update your applications or Key Alias to use the new CMK

KMS Key Deletion:

- Generated Keys – from within KMS
  - No expiration dates
  - Cannot be deleted immediately, mandatory 7 to 30 days waiting period
    - Key deletion can be cancelled during the waiting period by user
    - During the waiting period, the KMS key cannot be used for encrypt/decrypt
    - Everything will be deleted at the end of waiting period
  - They can be manually disabled immediately instead – can be re-enabled later
- Imported Keys
  - Expiration period can be setup on the key
    - KMS will delete the key material
    - Key material can also be deleted on demand by user
    - The metadata is kept so it can be reimported in future
  - User can manually disable it and schedule for deletion
- AWS Managed keys – cannot be deleted since its owned by AWS
- We can use CloudTrail, CloudWatch Logs, CloudWatch Alarms, and SNS to be notified when someone tries to use a CMK that's "Pending Deletion" in a cryptographic operation – Encrypt/Decrypt



- Or we can use EventBridge with CloudTrail to be notified of keys being deleted or disabled – we can even prevent this happening



KMS Key Policies:

- Access control policies that determine who can perform certain operations on KMS keys
- Key policies are attached to individual KMS keys and define permissions for various actions such as encryption, decryption, key rotation, and key deletion
- Default KMS Key Policy – created if specific KMS Key Policy isn't provided
    - Complete access to the key to the root user – entire AWS account that owns the KMS key
    - KMS Key Policy does not automatically give permission to the account or any of its users
    - Allows the account to use IAM policies to allow access to the KMS key, in addition to the key policy

Default KMS Key Policy

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

- Custom KMS Key Policy
    - Define users, roles that can access the KMS key
    - Define who can administer the key
    - Useful for cross-account access of user's KMS key
    - KMS key administrators have permissions to manage the KMS key
        - KMS key administrators cannot use the KMS Key Cryptographic Operations – Encrypt/Decrypt
        - Can add IAM users/roles
    - Allows IAM users/roles to use the KMS Key directly

- IAM users/roles don't need IAM policies if the KMS Key is in the same account
- The KMS key explicitly authorize IAM principal

```
{
    "Sid": "Allow use of the key",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::111122223333:user/ExampleUser",
            "arn:aws:iam::111122223333:role/ExampleRole",
            "arn:aws:iam::444455556666:root"
        ]
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource": "*"
}
```

- Key Policy comparison is explained below:

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

KMS Policy Condition:

- Policy conditions can be used to specify a condition within a Key Policy or IAM Policy
  - Can be used in Key Policies and IAM policies which control access to KMS resources
- The condition must be true for the policy statement to take effect
- You might want a policy statement to take effect only after a specific date has passed
- Allow or deny an action based the requesting service
- Predefined Condition Keys
- The *kms:ViaService* Condition Key:
  - Is a condition key which can allow or deny access to customer's CMK depending on which service originated the request
  - Limits the use of a KMS key to requests from specified AWS services
    - Can be used to limit the use of an AWS KMS customer master key (CMK) to requests from specified AWS services
    - IAM user must be authorized to use the KMS key and Grant it to the AWS service
    - Can be used with AWS Managed Keys – aws/ebs etc.
    - You can specify one or more services in each *kms:ViaService* condition key.

- **Example** – allow access to the CMK only for requests which come from EC2 or RDS from us-west-2 region only

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

  - this IAM policy grants the "ExampleUser" IAM user permission to perform various KMS actions on all KMS resources in the AWS account, but only if the request originates from either an EC2 instance in the "us-west-2" region or an RDS instance in the same region

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

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

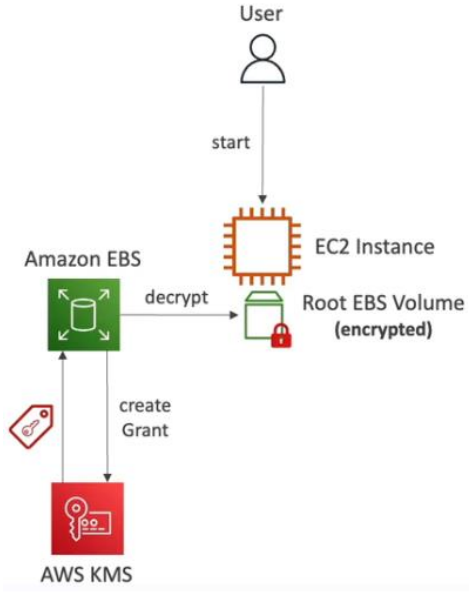
  - this IAM policy grants permission to any AWS principal from any AWS account to perform various KMS actions on all KMS resources, but only if the request

originates from an EC2 instance in the "us-west-2" region of the AWS account with the ID "123456789012"

- The services that user specifies must be integrated with KMS like S3, EBS, RDS, System Manager, SQS, Lambda
- We can also control access to our KMS keys based on tags and aliases

KMS Grants:

- Grants allow users to grant access to specific AWS KMS keys to other AWS accounts and IAM users/roles within user's AWS account
- Grants are alternative access control mechanism to a Key Policy – user don't need to change KMS Key Policy or IAM Policy
- Programmatically delegate the use of KMS CMKs to other AWS principles
- Often used for temporary permissions – encrypt, decrypt, re-encrypt, describekey, sign, verify etc. as well as creating more grants
- Grant for one KMS Key only, and one or more IAM principal
    - Once granted, a principal can perform any operation as specified in the grant
- Grants allow access, not deny
    - Use Key Policies for relatively static permissions and for explicit deny



- Grants are configured programmatically using AWS CLI
- Grants do not expire automatically, they must be deleted manually
- With grants, user can specify the grantee (AWS account or service), the operations they are allowed to perform, and the specific KMS keys they can use

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

- *create-grant* – adds a grant to the CMK, specifies who can use it and a list of operations that grantee can perform
- *list-grant* – to list a grant
- *revoke-grant* – to remove a grant
- A *grant token* is generated and can be passed as an argument to KMS API
- To create a KMS grant using the AWS CLI, you can use the *aws kms create-grant* command. Here's an example command:

aws kms create-grant --key-id <key-id> --grantee-principal <grantee-principal> --operations <operations> --name <name>

  - --key-id: The unique identifier of the KMS key for which the grant is being created.
  - --grantee-principal: The AWS account ID or IAM user/role ARN of the grantee principal who will be given access to the KMS key.
  - --operations: The specific KMS operations that the grantee principal is allowed to perform on the KMS key. For example, Encrypt or Decrypt.

- --name: A friendly name for the grant.

- To encrypt plain text using AWS KMS with a KMS grant, you can use the **aws kms encrypt** command in the AWS CLI. Here's an example command:

```
aws kms encrypt --key-id <key-id> --plaintext <plain-text> --grant-tokens <grant-token> --region <region>
```

  - --key-id: The unique identifier of the KMS key to be used for encryption.
  - --plaintext: The plaintext to be encrypted. This should be enclosed in quotes if it contains spaces or special characters.
  - --grant-tokens: One or more grant tokens that authorize the use of a KMS grant to encrypt the data. You can obtain grant tokens by using the aws kms create-grant command with the --retiring option.
  - --region: The AWS region where the KMS key is located.

- To decrypt data using AWS KMS with a KMS grant, you can use the **aws kms decrypt** command in the AWS CLI. Here's an example command:

```
aws kms decrypt --ciphertext-blob <ciphertext-blob> --grant-tokens <grant-token> --region <region>
```

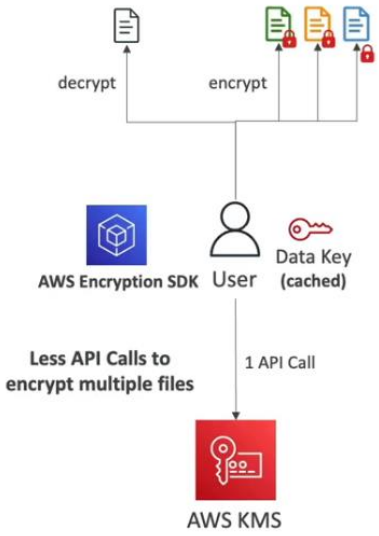
  - --ciphertext-blob: The base64-encoded ciphertext to be decrypted. This should be the output of a previous encryption operation.
  - --grant-tokens: One or more grant tokens that authorize the use of a KMS grant to decrypt the data. You can obtain grant tokens by using the aws kms create-grant command with the --retiring option.
  - --region: The AWS region where the KMS key is located.

```
#Create a new key and make a note of the region you are working in
aws kms create-key

#Test encrypting plain text using my new key:
aws kms encrypt --plaintext "hello" --key-id <key_arn>

#Create a new user called Dave and generate access key / secret access key
aws iam create-user --user-name dave
aws iam create-access-key --user-name dave

#Run aws configure using Dave's credentials creating a CLI profile for him
aws configure --profile dave
aws kms encrypt --plaintext "hello" --key-id <key_arn> --profile dave

#Create a grant for user called Dave
aws iam get-user --user-name dave
aws kms create-grant --key-id <key_arn> --grantee-principal <Dave's_arn> --operations "Encrypt"

#Encrypt plain text as user Dave:
aws kms encrypt --plaintext "hello" --key-id <key_arn> --grant-tokens <grant_token_from_previous_command> --profile dave

#Revoke the grant:
aws kms list-grants --key-id <key_arn>
aws kms revoke-grant --key-id <key_arn> --grant-id <grant_id>

#Check that the revoke was successful:
aws kms encrypt --plaintext "hello" --key-id <key_arn> --profile dave
```

KMS Data Key Caching:

- When multiple API calls to KMS are made and the service limit of requests per second is reached
- Data Key Caching enables the reuse of data keys that protect the data
- Rather than generating a new data key for each encryption operation

- Latency is reduced, throughput is improved, cost is reduced, and service limits are adhered to
- Implementation is done using the AWS encryption SDK

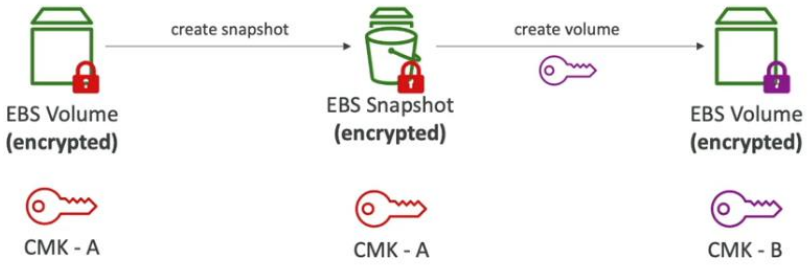


- **Note:** Encryption best practices discourage reuse of data keys

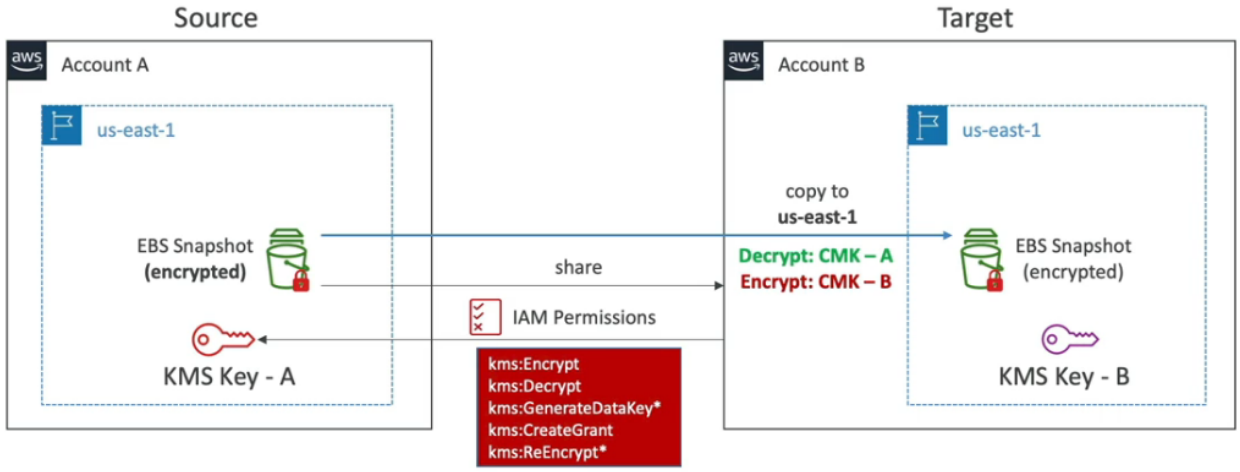
Use KMS with Different AWS Services:

KMS with EBS:

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



- New EBS volumes aren't encrypted by default
- There's an account-level setting to encrypt automatically new EBS volumes and Snapshots
- This setting needs to be enabled on pre-region basis
- We can automate cross-account EBS KMS-encrypted snapshot copies



- The target account policy would be

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

```
        {
            "Effect": "Allow",
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
            ],
            "Resource": [
                "arn:aws:kms:us-east-1:111111111111:key/
1234abcd-12ab-34cd-56ef-1234567890ab",
                "arn:aws:kms:us-east-1:222222222222:key/
4567dcba-23ab-34cd-56ef-0987654321yz"
            ]
        }
    ]
}
```

Encrypt Unencrypted EFS File System:

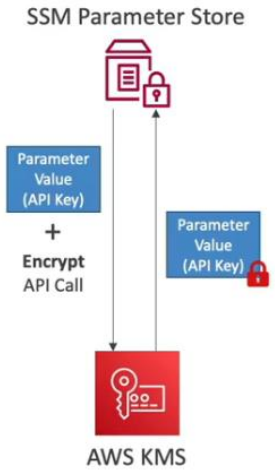
- Existing un-encrypted EFS file system can't be encrypted
- For that we must create a new encrypted EFS File System and migrate all the files using AWS DataSync

KMS with SSM Parameter Store:

- SSM Parameter Store uses KMS to encrypt/decrypt parameter values of type Secure String
- Two types of Secure String parameters

- Standard – all parameters encrypted using the same KMS key
- Advanced – each parameter encrypted with a unique data key (Enveloped Encryption)

- Specify the KMS key or use AWS Managed key – aws/ssm
- Works only with Symmetric KMS Keys
- Encryption process takes place in AWS KMS



- **Note:** you must have access to both the KMS key and the parameter in SSM Parameter Store

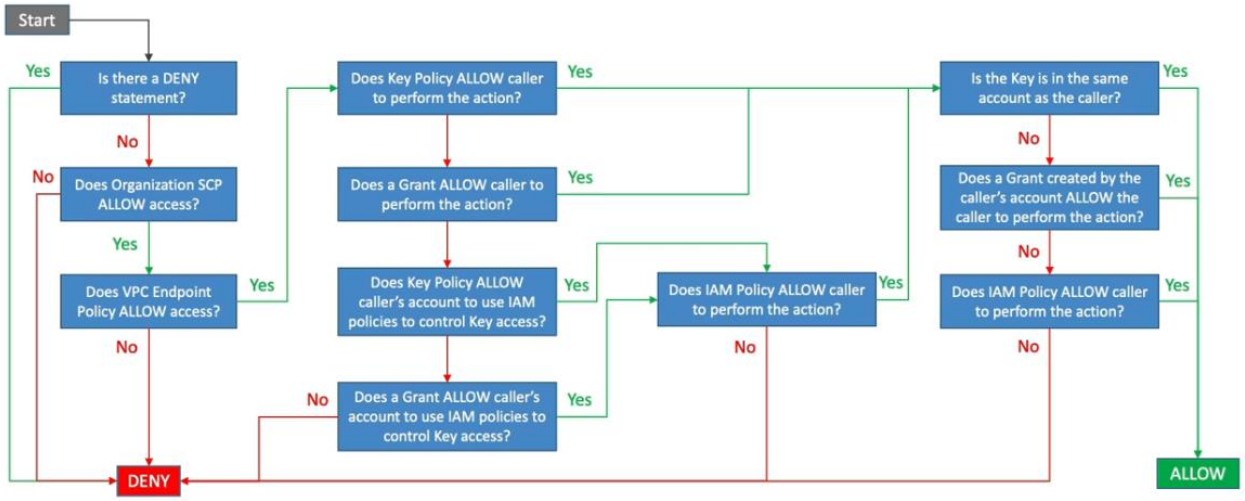
EncryptionContext:

- A key-value map provided to AWS KMS with each encryption and decryption request in AWS
- A collection of metadata or additional information about the data that you are encrypting
  - This context is not encrypted but is included in the encrypted data's cryptographic operations
  - It allows you to define attributes or conditions that must be satisfied during decryption
- EncryptionContext is used to improve the security of applications by protecting the integrity and authenticity of encrypted data
- It provides additional benefits such as additional authenticated data (AAD), audit trail, and authorization context
  - AAD is a part of authenticated encryption that prevents tampering with the ciphertext itself
- EncryptionContext is KMS's implementation of AAD and should contain non-sensitive information related to the ciphertext
- Including EncryptionContext ensures that unencrypted data associated with the ciphertext is protected against tampering
- For example – let's say you have an application that stores sensitive documents
  - Each document has an associated "document type" attribute (e.g., "confidential," "public," "internal")
  - When encrypting the documents, you can include the "document type" as part of the EncryptionContext

- Later, during decryption, you can set up KMS policies to allow different IAM roles or users to decrypt data based on the "document type" value

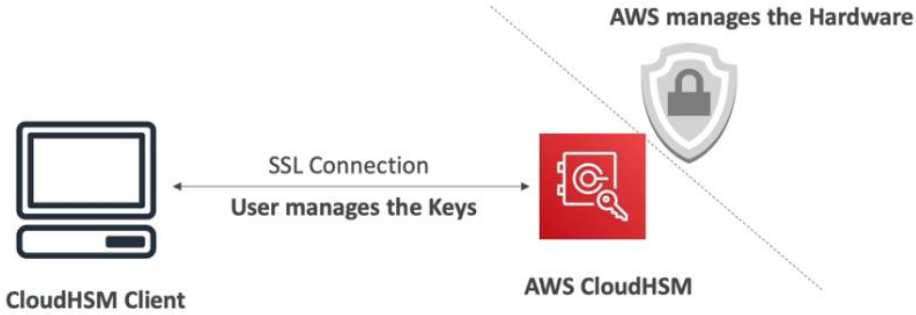
KMS Key Authorization Process:

- The KMS Key authorization process has been explained in the below diagram



AWS CloudHSM:

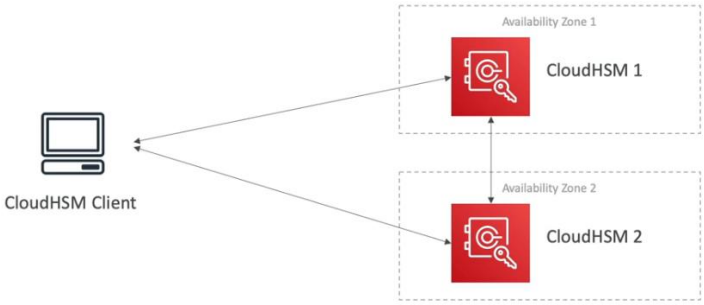
- A cloud-based hardware security module service provided by AWS
- This service helps to meet corporate, contractual, and regulatory compliance requirements for data security by using dedicated Hardware Security Module (HSM) appliance within the AWS Cloud
- If offers dedicated and FIPS 140-2 Level 3 validated hardware security modules to help protect sensitive data and cryptographic keys
- It provides secure key storage and cryptographic operations of applications running in AWS, tamper-resistant hardware security module
- It allows to generate, store, and manage encryption keys using industry-standard APIs and cryptographic algorithms
    - Users will manage their own encryption keys not AWS
- It supports both symmetric and asymmetric encryption (SSL/TLS)
- CloudHSM Software – Manage the keys, Manage the users



- CloudHSM provides strict access controls, including dedicated VPC connectivity, IAM-based authentication, and fine-grained authorization policies
- It enables control of data, evidence of control, meet through compliance controls
- If the CloudHSM detects physical tampering, the keys will be destroyed
- If the CloudHSM detects five unsuccessful attempts to access an HSM partition as Crypto Officer, the HSM appliance erases itself
- If the CloudHSM detects five unsuccessful attempts to access an HSM with Crypto User credentials, the user will be locked and must be unlocked by a Crypto Officer
- The VPC and subnets are defined during the initial cluster creation and cannot be modified afterwards

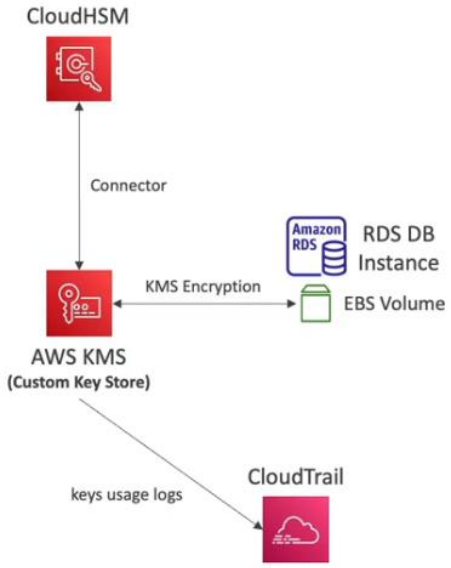
CloudHSM High Availability:

- CloudHSM allows to scale the number of HSM instances based on performance and capacity needs
- The service provides automatic backups, high availability, and automatic replication of HSM instances across multiple AZs
- CloudHSM clusters are spread across the multiple AZs – great for availability and durability



CloudHSM Integrations:

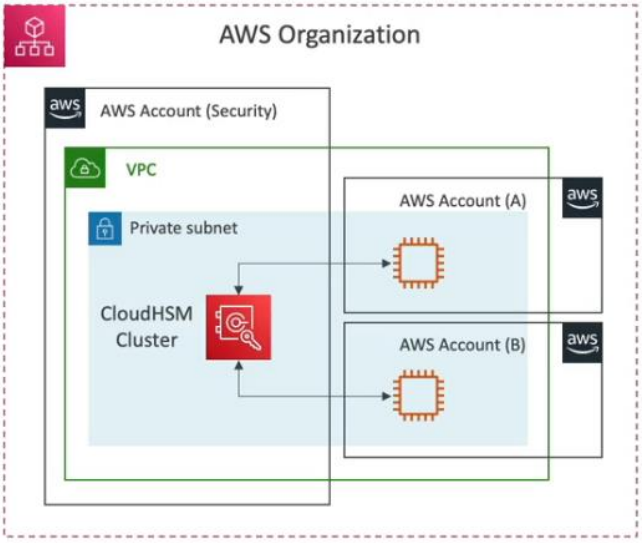
- Integration with AWS Services:
  - Through integration with AWS KMS
  - Configure KMS Custom Key Store with CloudHSM
  - Example – EBS, S3, RDS, etc.
  - Supports RDS Oracle TDE (through KMS)



- Integration with 3rd Party Services:
  - Allows creating and storing keys in CloudHSM
  - Use cases – SSL/TLS Offload, Windows Server Certificate Authority (CA), Oracle TDE, Microsoft SignTool, Java Keytool etc.

Sharing Cluster Across-Account:

- The private subnets in which a CloudHSM cluster resides can be shared using AWS RAM
- The CloudHSM cluster itself cannot be shared
- The VPC subnets can be shared with the entire Organization, specific OUs, or AWS accounts
- The CloudHSM security group should be configured to allow traffic from the clients



CloudHSM User Types:

- 4 users type in CloudHSM users:
  - Precrypto Officer:

- Has the highest level of authority in CloudHSM environment
- Can perform all administrative tasks, including creating and managing users, keys, and policies
- Responsible for managing the overall security and configuration of the CloudHSM cluster
- Can grant or revoke privileges for other users, including Crypto Officers, and Crypto Users

- Crypto Officer:
  - Has administrative privileges within the CloudHSM cluster
  - Can perform various management tasks, such as creating, deleting, changing user passwords – and managing users, keys, and policies
  - Responsible for maintaining the security and configuration of CloudHSM cluster
  - Cannot perform administrative tasks reserved for Precrypto Officer
- Crypto User:
  - Has access to cryptographic keys stored in the CloudHSM cluster
  - Can perform cryptographic operations using the keys, such as encryption, decryption, signing, and verification
  - Managed within the CloudHSM cluster and has user-specific credentials for authentication
  - Performs cryptographic operations on behalf of Application users
  - Can perform key management operations – create, delete, share, import, and export cryptographic keys
- Appliance User:
  - Can perform cloning and synchronization operations
  - AWS CloudHSM uses the Appliance User to synchronize the HSMs in an AWS CloudHSM cluster
  - The Appliance User exists on all HSMs provided by AWS CloudHSM, and has limited permissions

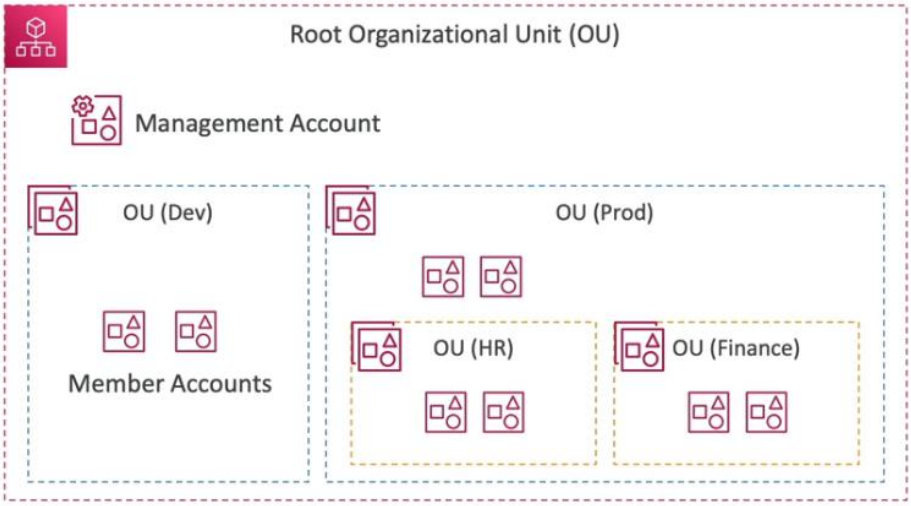
	Crypto Officer (CO)	Crypto User (CU)	Appliance User (AU)	Unauthenticated User
Get basic cluster info[1]	Yes	Yes	Yes	Yes
Zeroize an HSM[2]	Yes	Yes	Yes	Yes
Change own password	Yes	Yes	Yes	Not applicable
Change any user's password	Yes	No	No	No
Add, remove users	Yes	No	No	No
Get sync status[3]	Yes	Yes	Yes	No
Extract, insert masked objects[4]	Yes	Yes	Yes	No
Key management functions[5]	No	Yes	No	No
Encrypt, decrypt	No	Yes	No	No
Sign, verify	No	Yes	No	No
Generate digests and HMACs	No	Yes	No	No

CloudHSM vs KMS:

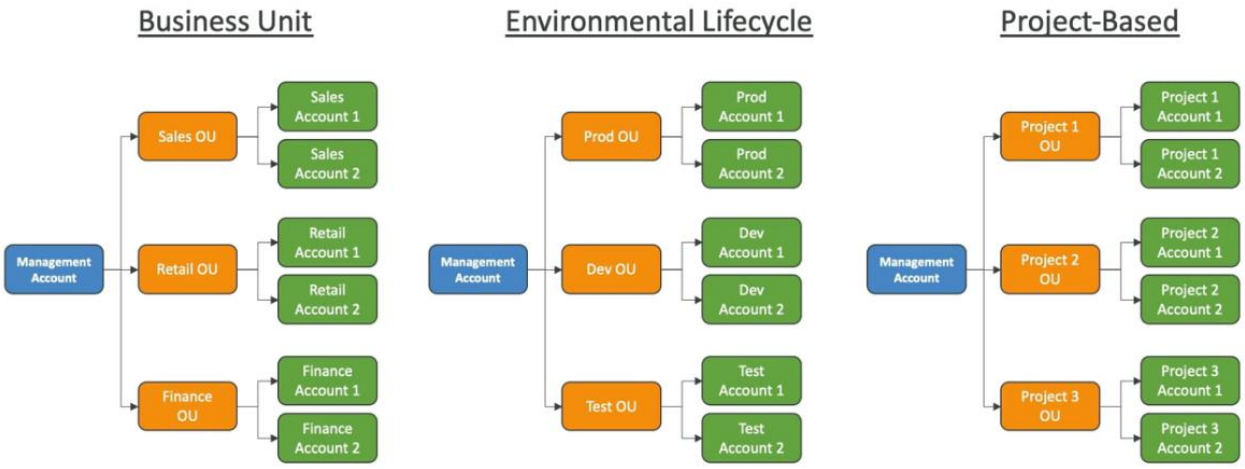
Feature	AWS KMS	AWS CloudHSM
Tenancy	Multi-Tenant	Single-Tenant
Standard	FIPS 140-2 Level 2	FIPS 140-2 Level 3
Master Keys	• AWS Owned Keys • AWS Managed Keys • Customer Managed KMS Keys	Customer Managed CMK
Key Types	• Symmetric • Asymmetric • Digital Signing	• Symmetric • Asymmetric • Digital Signing & Hashing
Key Accessibility	Accessible in multiple AWS regions KMS Key Replication	• Deployed and managed in a VPC • Can be shared across VPCs (VPC Peering)
Cryptographic Acceleration	None	• SSL/TLS Acceleration • Oracle TDE Acceleration
Access & Authentication	AWS IAM	You create users and manage their permissions
High Availability	AWS Managed Service	Add multiple HSMs over different AZs
Audit Capability	• CloudTrail • CloudWatch	• CloudTrail • CloudWatch • MFA support
Free Tier	Yes	No

AWS Organizations:

- An account management service which allows to consolidate multiple AWS accounts into an organization that can be managed and governed centrally
- It's a global service
- Provides a hierarchical structure called an organization – group and manage AWS accounts
    - Organize them into OUs/groups and apply policies to enforce compliance
    - Main account is called the management account
    - Other accounts are member accounts
    - Member accounts can only be part of one Organization



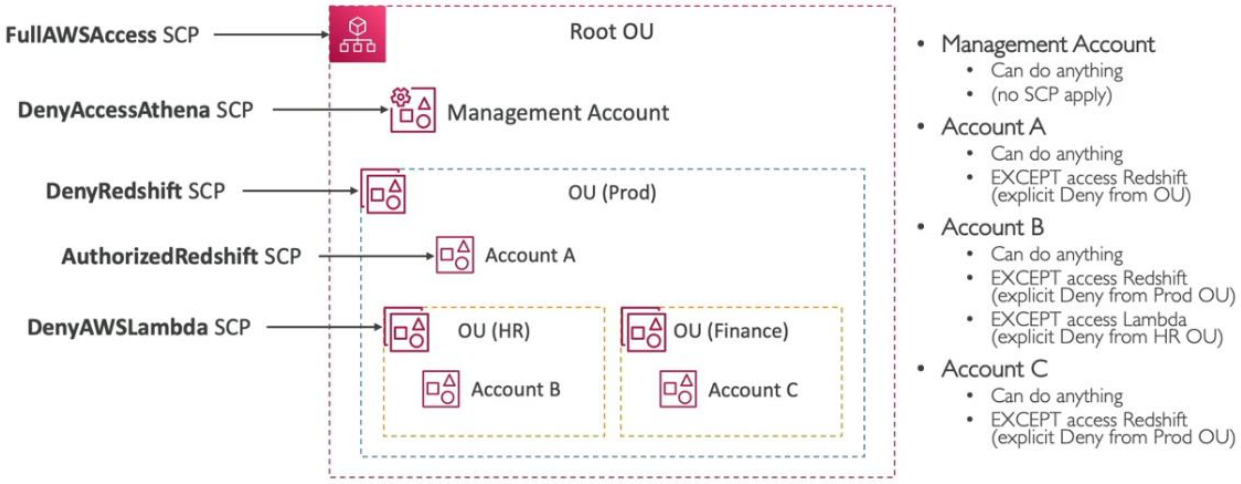
- Centralized consolidate billing – receive a single bill for all the accounts in the Organization
    - Pricing benefits from aggregated usage – volume discount for EC2, S3 etc.
    - Share reserved instances and saving plans discount across accounts



- API is available to automate AWS account creation
- Attach policy base controls Service Control Policies (SCP) – define the permissions and access controls

Service Control Policies:

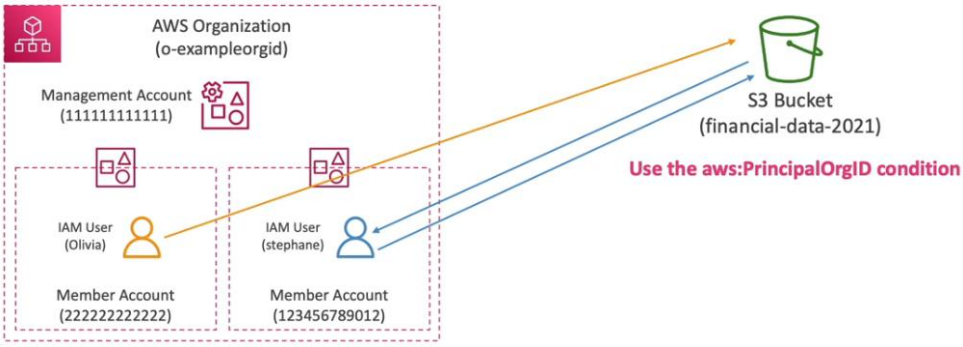
- A service control policy is used to centrally control the use of AWS services across multiple accounts
    - Like a filter which restricts access to AWS services
- The SCP applies to all OUs and accounts below the OU to which it is attached
    - They do not apply to the management account – full admin power
    - Can be used to create a Permissions Boundary
- Restricting the actions, the users, the groups, roles in those accounts can do – including root
- SCPs can deny access only, they cannot allow
    - Must have explicit allow – does not allow anything by default
- The effective permission on a principal in an account that has an SCP attached is the intersection of what is allowed explicitly in the SCP and what is allowed explicitly in the permissions attached to the principal – example is below



- SCPs alone are not sufficient to grant permissions to the accounts in the organization. No permissions are granted by an SCP
- SCP defines a guardrail on the actions that an account's administrator can delegate to the IAM users and roles in the affected accounts
- The administrator must still attach identity-based or resource-based policies to IAM users or roles, or to the resources in AWS accounts to actually grant permissions
- The effective permissions are the logical intersection between what is allowed by the SCP and what is allowed by the IAM and resource-based policies
- Two strategies – Blocklist and Allowlist

IAM Policies – AWS Organizations:

- We can use *aws:PrincipalOrgID* condition key in our resource-based policies to restrict access to IAM principals from accounts in an AWS Organization
- An ID given to an AWS Organization to identify which organization a user or role belongs to



Tag Policies – AWS Organizations:

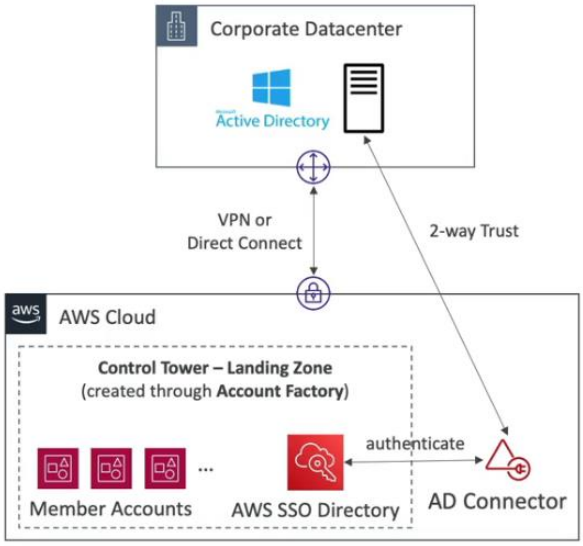
- Helps standardize tags across resources in an AWS Organization
- Helps with AWS Cost Allocation Tags and Attribute-based Access Control
- Can generate a report that lists all tagged/non-compliant resources
- CloudWatch Events – to monitor non-compliant tags

AWS Control Tower:

- Provides an easy way to set up and govern a secure, multi-account AWS environment
- Automates the process of setting up a well-architecture AWS environment with best practices
- Helps enforce policies for security, compliance, and operations across multiple AWS accounts
- Provides guardrails to prevent unintended resource configuration changes
- Offers a centralized dashboard for monitoring and managing AWS accounts and resources

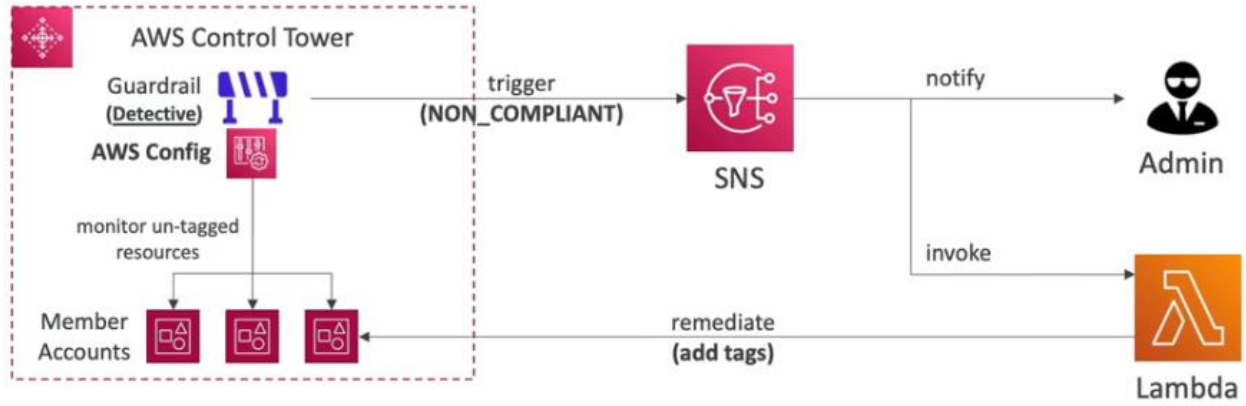
Account Factory:

- Simplifies the on-boarding process for new AWS accounts
    - Automate account provisioning and deployments
- Enables to create pre-approved baselines and configuration options for AWS accounts in Organization – VPC default configuration, subnets, regions etc.
- Uses AWS Service Catalog – to provision new accounts



Guardrail:

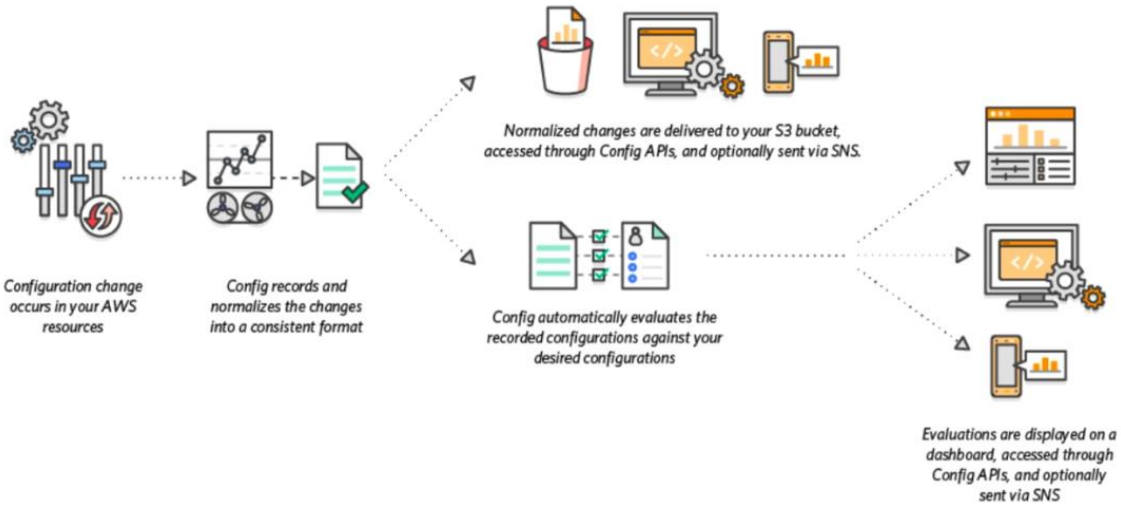
- Detect and remediate policy violations
- Provided on going governance for Control Tower Environment – AWS accounts
- Preventive – using SCPs (e.g., Disallow creation of Access Keys for the root user)
- Detective – using AWS Config (e.g., detect whether MFA for the root user is enabled)



- Guardrail levels can be:
  - Mandatory – automatically enable and enforce by AWS Control Tower
    - Example – disallow public read access to the Log Archive account
  - Strongly recommended – based on AWS best practices
    - Example – enable encryption for EBS volumes attached to EC2 instances
  - Elective – commonly used by enterprise (optional)
    - Example – disallow delete actions without MFA in S3 buckets

AWS Config:

- A service that enables to assess, audit, and evaluate the configurations of AWS resources
- Config continuously monitors and records the AWS resource configurations and allows to automate the evaluation of recorded configurations against desired configurations
  - If AWS Config finds a policy violation, it can trigger an Amazon CloudWatch Event rule to trigger an AWS Lambda function, which corrects the violation
  - While AWS Config can be set up to track changes in security groups, AWS Lambda doesn't inherently support direct remediation or sending notifications upon detection of changes in security groups
- Questions that can be solved by AWS Config:
  - Is there unrestricted SSH access to my security groups?
  - Do my buckets have any public access?
  - How has my ALB configuration changed over time?
- It is a per-region service – can be aggregated across regions and accounts
- Possibility of storing configuration data into S3 (analyzed by athena)

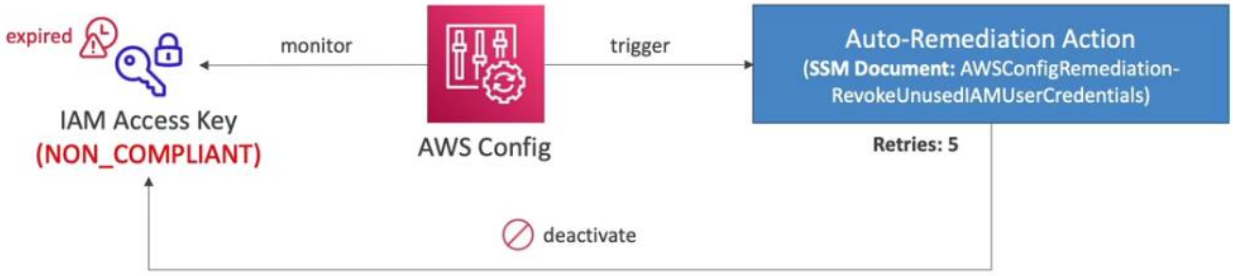


- It enables compliance auditing, security analysis, resource tracking
- It provides configuration snapshots and logs config changes of AWS resources
- It also provides automated compliance checking
- Permissions needed for Config – an IAM role with:
    - Read-only permissions to the recorded resources
    - Write access to S3 logging bucket
    - Publish access to SNS

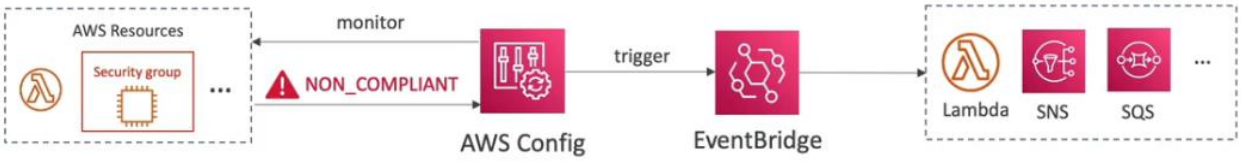
AWS Config Rules:

- A rule represents the desired value for resources.
- AWS Managed Rules:
    - Pre-built and managed by AWS
    - User will simply choose the rule and enable by supplying a few configuration parameters to get started
- Customer Managed Rules: (75)
    - Custom rules, defined and built by the customer
    - Customer can create a function in AWS lambda that can be invoked as part of a custom rule and these functions execute in customer's AWS account. Examples:
        - Evaluate if each EC2 instance is t2.micro
        - Evaluate if each EBS disk is of type gp2
- Rules can be evaluate/triggered:
    - For each config change
    - And/or at regular intervals
- Rules are just for compliance; it does not prevent actions from happening – no deny
- No free tier – 0.003$ per configuration item recorded per region and 0.001$ per config rule evaluation per region
- Remediations – can be automated of non-compliant resource using SSM Automation Documents

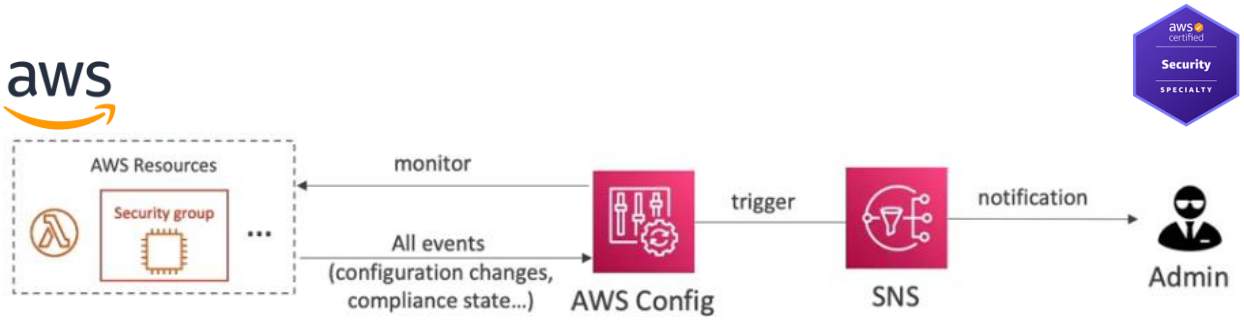
- AWS-Managed Automation Documents
- Custom Automation Documents
  - For example, a custom automation document that invokes Lambda function
- Remediation Retries can be setup if resource is still non-compliant after auto-remediation
- The AWS Config Auto Remediation feature automatically remediates non-compliant resources evaluated by AWS Config rules
  - You can associate remediation actions with AWS Config rules and choose to execute them automatically to address non-compliant resources without manual intervention
- The AWS Config managed rule "*restricted-ssh*" checks if the incoming SSH traffic for the security groups is accessible
  - The rule is COMPLIANT when IP addresses of the incoming SSH traffic in the security groups are restricted (CIDR other than 0.0.0.0/0)
- AWS Config rule can check that running EC2 instances are using approved Amazon Machine Images, or AMIs. You can specify a list of approved AMIs by ID or provide a tag to specify the list of AMI IDs.
  - The AWS Config rule "*approved_ami_by_id*" checks if running instances are using specified AMIs. You must specify a list of approved AMI IDs. Running instances with AMIs that are not on this list are NON_COMPLIANT.



- Notifications – can be done via EventBridge or directly via SNS
  - Use EventBridge to trigger notifications when AWS resources are non-compliant



  - Ability to send configuration changes and compliance state notifications to SNS
    - All events
    - SNS filtering can be used
    - Or filtering at client side



AWS Config Resource:

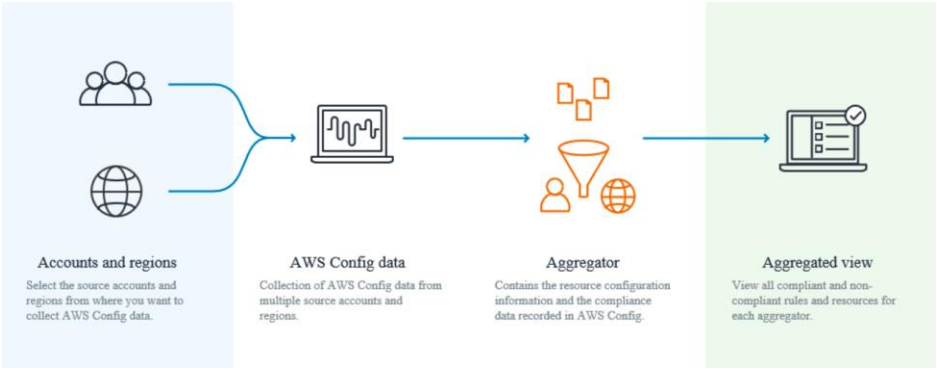
- View compliance of a resource over time
- View configuration of a resource over time
- View CloudTrail API calls of a resource over time

Conformance Packs:

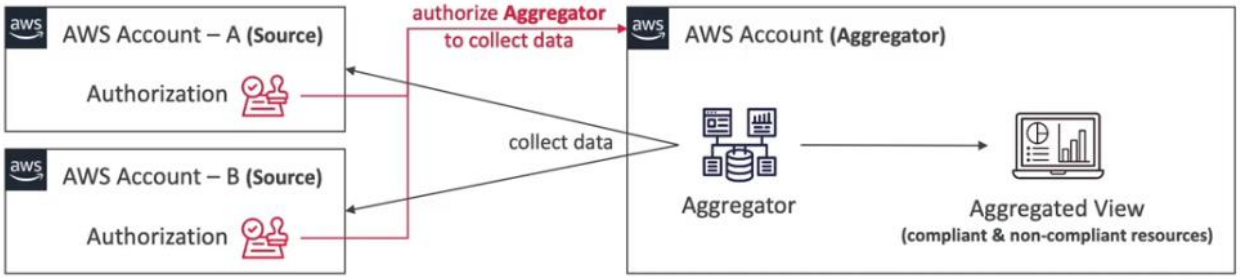
- A conformance pack is a collection of AWS config rules and remediation actions that can be easily deployed as a single entity in an account and a Region or across an organization in AWS Organizations
- Conformance Packs are created by authoring a YAML template that contains the list of AWS Config managed or custom rules and remediation actions
- Non-compliant resources are identified through AWS Config's evaluation results
- Can be applied to multiple accounts and regions simultaneously, ensuring consistent configurations and compliance

Aggregator:

- An AWS Config resource type that collects AWS Config configurations and compliance data from the following:
    - Multiple accounts and multiple regions
    - Single accounts and multiple regions
    - An organization in AWS Organizations and all the accounts in that organization



- The aggregator is created in one central aggregator account
- Aggregates rules, resources, etc. across multiple accounts and regions
- If using AWS Organizations, no need for individual Authorization
- Rules are created in each individual source AWS account
- Can deploy rules to multiple target accounts using CloudFormation StackSets

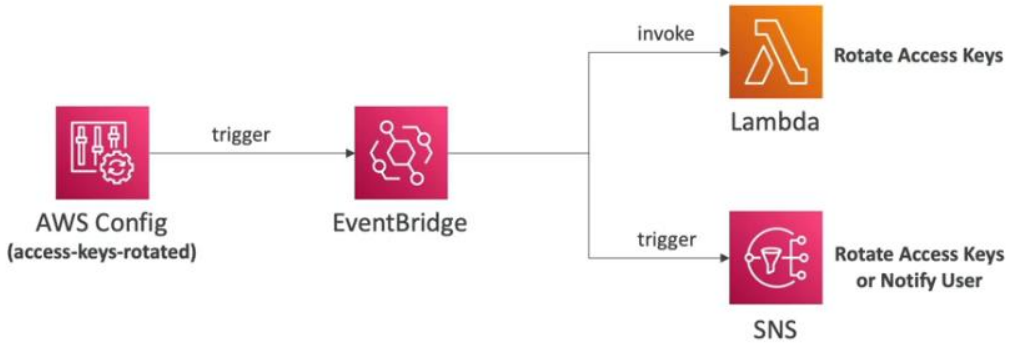


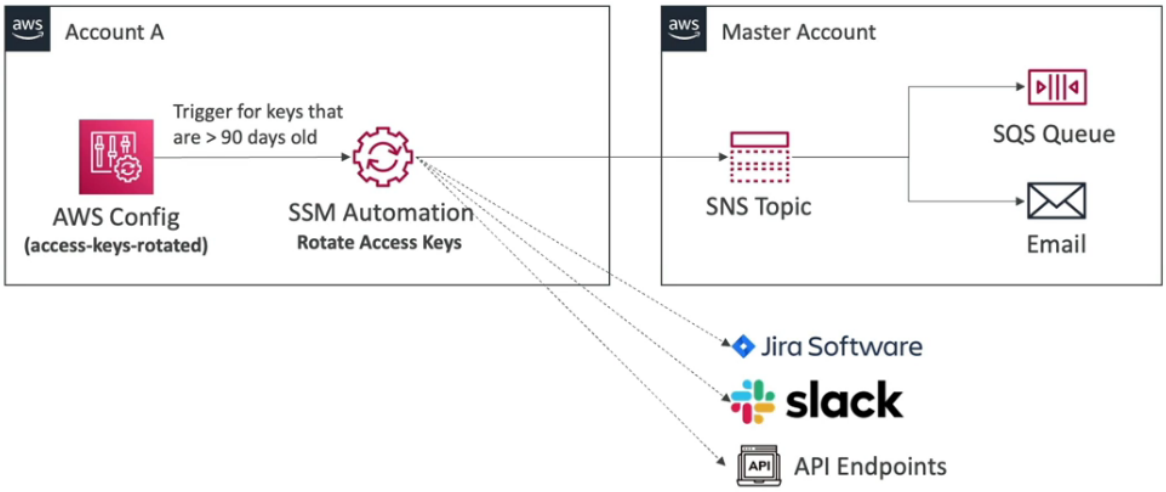
Security & Monitoring Config:

- Restrict access – users need to be authenticated with AWS and have the appropriate permissions set via IAM policies to gain access
- Only Admins needing to set up and manage Config requires full access
- Provide read only permissions for Config day-to-day use
- Use CloudTrail with Config to provide deeper insight into resources
- Use CloudTrail to monitor access to Config, such as someone stopping the Config Recorder

AWS Config – Use Cases:

- Audit IAM policies
    - For example, Expired IAM keys – rotate via EventBridge and Lambda or SSM Automation

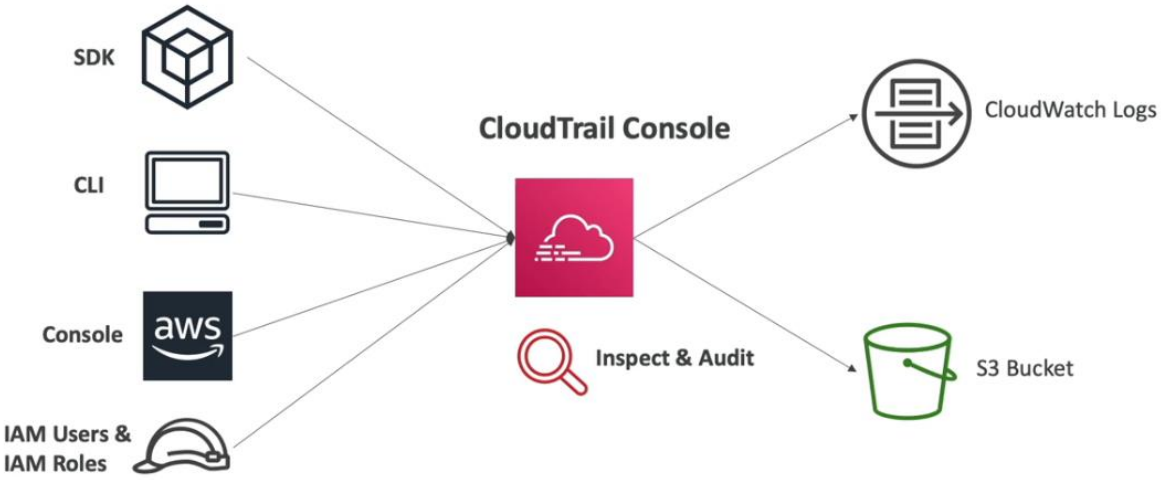




- Detect if CloudTrail has been disabled
- Detect if EC2 instances are created with unapproved AMIs
- Detect if Security Groups are open to public
- Detect if Internet Gateway is added to unauthorized VPC
- Detect if EBS volumes are encrypted
- Detect if RDS databases are public

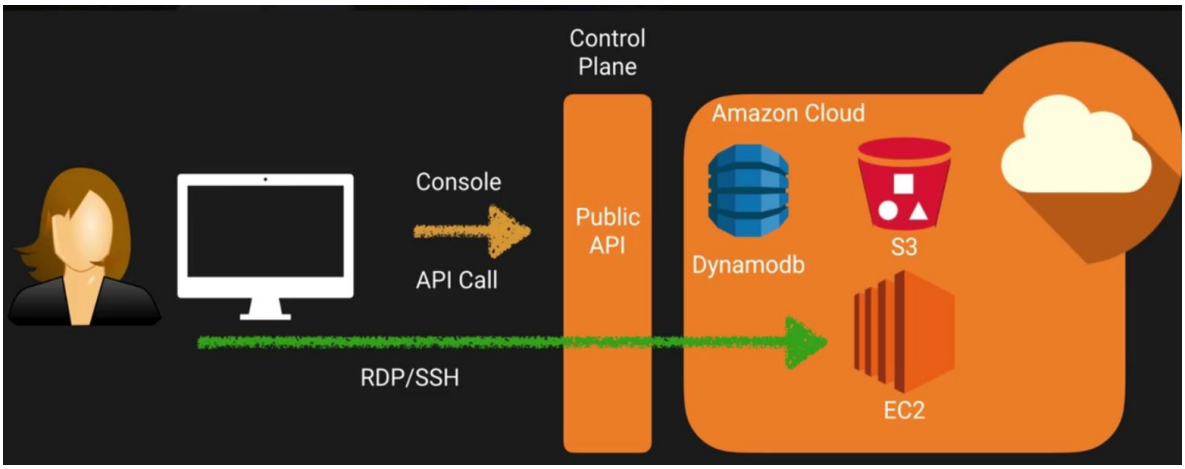
AWS CloudTrail:

- A web service that records AWS API calls for the account and delivers log files – valuable insights into the account activity and maintain visibility over the AWS resources
- Not real-time – delivers an event within 15 minutes of API call
    - Delivers log files to an S3 bucket every 5 minutes
- It logs API calls of supported services, but it won't log the RDP/SSH sessions
    - Console
    - SDK
    - CLI
    - AWS Services
- Can be used to track changes to AWS resources, detect unauthorized activity, changes to security groups, modification to IAM roles, validate compliance, and aid in incident response
- It enables after-the-fact incident investigation, near-real time intrusion detection, and industry & regulatory compliance



- What is logged?
    - Metadata around API calls
    - The identity of the API caller
    - The time of the API call
    - The source IP address of the API caller
    - The request parameters
    - The response elements returned by the service

- CloudTrail event logs are sent to S3 bucket and user can manage the retention in S3
- Delivered every 5 minutes (active) up to 15 minutes delay
- It supports logging of events from most AWS services, including EC2, S3, RDS, IAM, Lambda, and more
- Notifications available (SNS etc.)
- Can be integrated with AWS CloudWatch allowing to setup alarms and receive notifications based on specific API activity or events
- Can be aggregated across regions
- Can be aggregated across accounts



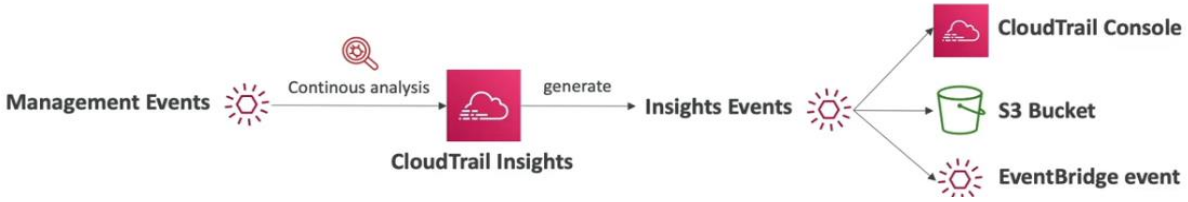
- CloudTrail is enabled by default (for 7 days)
- Operational and security incidents can be troubleshooted over the past 90 days in CloudTrail console by viewing 'Event History'
- Log in JSON format – *key:value* pairs
- CloudTrail for user's all AWS accounts can be enabled within user's organization using AWS Organizations to centralize and standardize logging and compliance
- Can be encrypted using AWS KMS for enhanced security

CloudTrail Events:

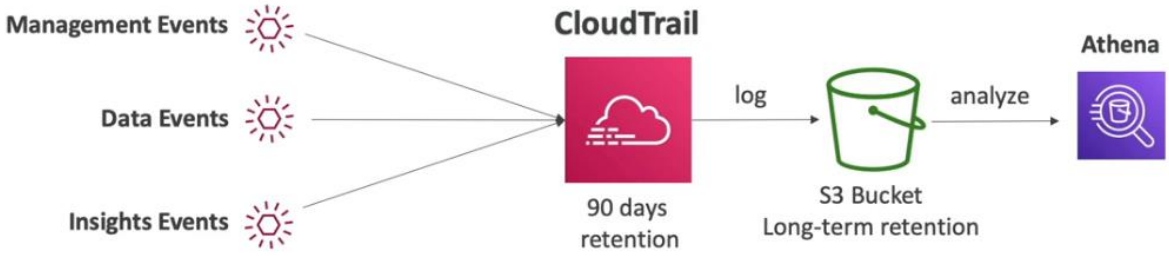
- Management Events:
  - Operations that are performed on resources in AWS account
    - Configuring security – IAM *AttachRolePolicy*
    - Configuring rules for routing data – EC2 *CreateSubnet*
    - Setting up logging – CloudTrail *CreateTrail*
  - By default, trails are configured to log management events
  - Can separate Read Events (that don't modify resources) from Write Events (that may modify resources)
- Data Events:
  - By default, data events are not logged – high volume operations
  - To record CloudTrail data events, you must explicitly add the supported resources or resource types for which you want to collect activity to a trail.

- S3 object level activity – *GetObject*, *DeleteObject*, *PutObject*
- AWS Lambda function execution activity – the *Invoke* API

- CloudTrail Insights Events:
  - To detect unusual activity in the AWS account
    - Inaccurate resource provisioning
    - Hitting service limits
    - Bursts of IAM actions
    - Gaps in periodic maintenance activity
  - Analyzes normal management events to create a baseline
  - Then continuously analyzes write events to detect unusual patterns

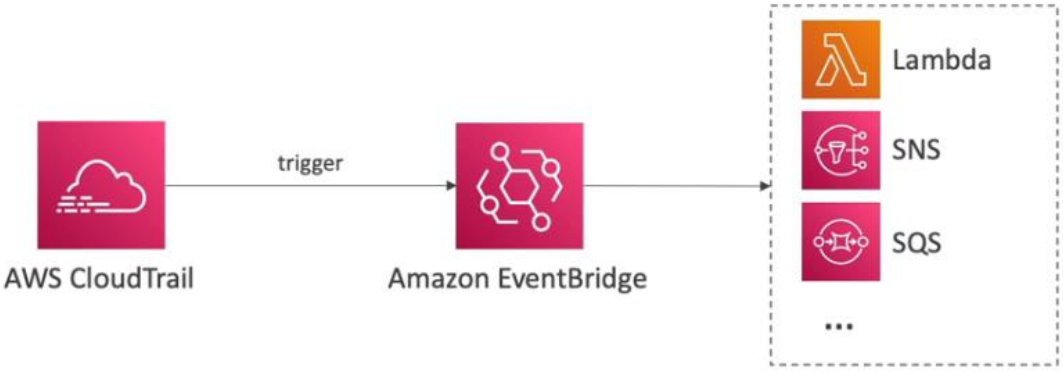


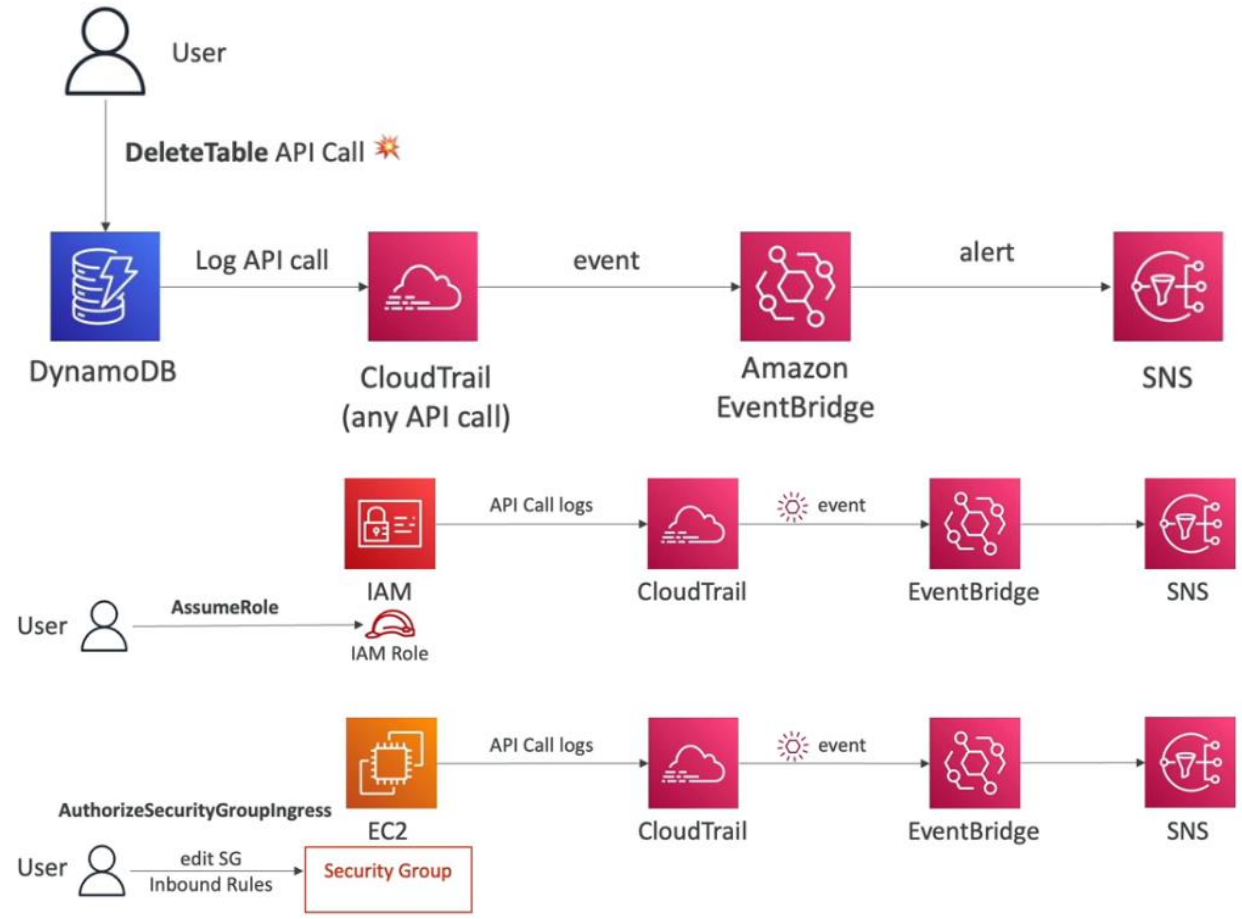
- CloudTrail Events Retention:
  - Default – 90 days events stored in CloudTrail
  - To keep them beyond this period, need to be logged in S3 and can be queried by Athena



CloudTrail with EventBridge:

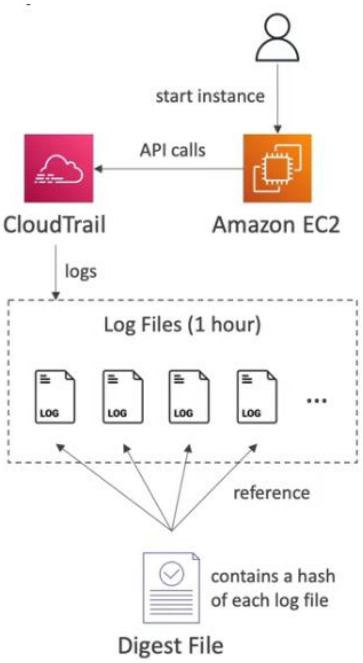
- EventBridge rule can be configured to monitor API calls being recorded in AWS CloudTrail that relate to policies being applied to an Amazon S3 bucket. EventBridge can then generate a notification using Amazon SNS.





CloudTrail-Digest Files:

- Validating the CloudTrail Log File Integrity:
  - Was the log file modified, or deleted after CloudTrail delivered it?
  - Integrity-validation – SHA-256 hashing, SHA-256 with RSA for digital signing
- Log files are delivered with a 'digest' file
- Digest file can be used to validate the integrity of the log file
- References the log files for the last hour and contains a hash of each
- Stored in S3 bucket as log files (different folders)
- Still protect S3 bucket using bucket policy, versioning, MFA delete protection, encryption, object lock
- Protect CloudTrail using IAM



Securing CloudTrail Logs:

- Why consider securing CloudTrail Logs?
  - CloudTrail logs may contain personal identifiable data such as usernames and even team membership. Also detailed configuration information such as DynamoDB table and key names may be stored
  - This information may prove valuable to an attacker, and it is considered best practice to secure CloudTrail logs
- How unauthorized access to log files can be stopped?
  - IAM policies and S3 bucket policies can be used to restrict access to S3 bucket containing the log files
  - SSE-S3 or SSE-KMS can be used to encrypt the logs
- How be log access restricted to only employees with security responsibility?
  - Place employees who have a security role, into an IAM group with attached policies that enable access to the logs
- How to get notified that a log file has been created, and then validate that it's not been modified?
  - Configure SNS notifications and log file validation on the 'Trail'
  - Develop a solution that when triggered by SNS will validate the logs using the provided digest file
  - The CLI command fragment, *'aws cloudtrail validate-logs'* will identify if any of the CloudTrail logs have been modified or deleted
- How can log files be prevented being deleted?
  - Restrict delete access with IAM and bucket policies
  - Configure S3 MFA delete
  - Validate that logs have not been deleted using log file validation

- To ensure that CloudTrail remains enabled in your account, AWS Config provides the cloudtrail-enabled managed rule. If CloudTrail is turned off, the cloudtrail-enabled rule automatically re-enables it by using automatic remediation.

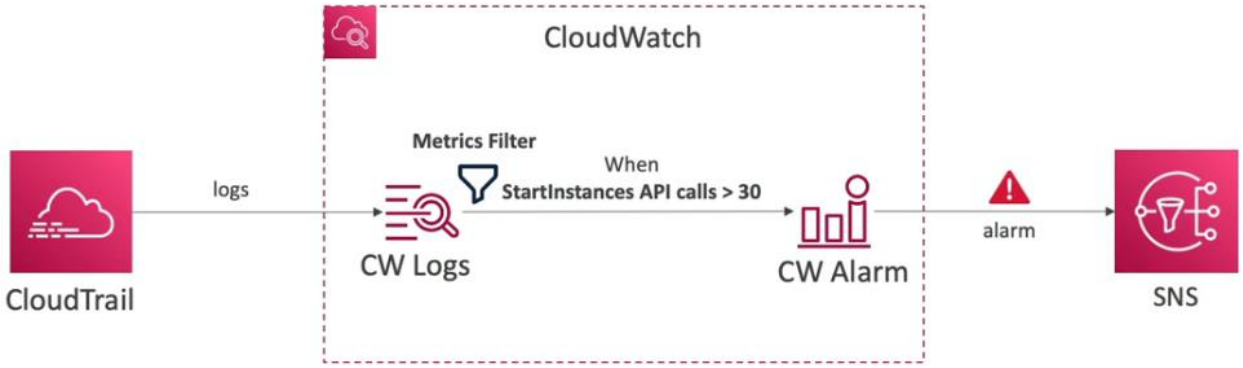
- How can it be ensured that logs are retained for X years in accordance with the compliance standards?
  - By default, the logs will be kept indefinitely
  - S3 Object Lifecycle Management can be used to remove the files after the required period, or move the files to AWS Glacier for more cost-effective long-term storage

Organizations Trails:

- A trail that will log all events for all AWS accounts in an AWS organization
- Log events for management and member accounts
- Trail with the same name will be created in every AWS account – IAM permissions
- Member accounts can't remove or modify the organization trail – view only
- You can have CloudTrail deliver log files from multiple AWS accounts into a single Amazon S3 bucket.
  - To accomplish this, turn on CloudTrail in the account where the destination bucket will belong, configure the bucket policy to allow cross-account permission.
  - Turn on CloudTrail in the other accounts, configure all accounts to log to the destination bucket
- CloudTrail uses a service principal to collect the trail data across accounts and to store in a central bucket and log group.
  - In the master account, when the organization trail is enabled through the management console, the S3 bucket resource policy is updated to grant *PutObject* permission to CloudTrail service principal
  - A service-linked role is created in the master account with CloudTrail as the trusted entity that has permission to log to the log group
- Enable multi-region CloudTrail. You can configure CloudTrail to deliver log files from multiple regions to a single S3 bucket for a single account
  - When a new region launches in the aws partition, CloudTrail automatically creates a trail for you in the new region with the same settings as your original trail.

CloudTrail to CloudWatch Metric Filter:

- Ability to be able to set threshold for detection



AWS CloudWatch:

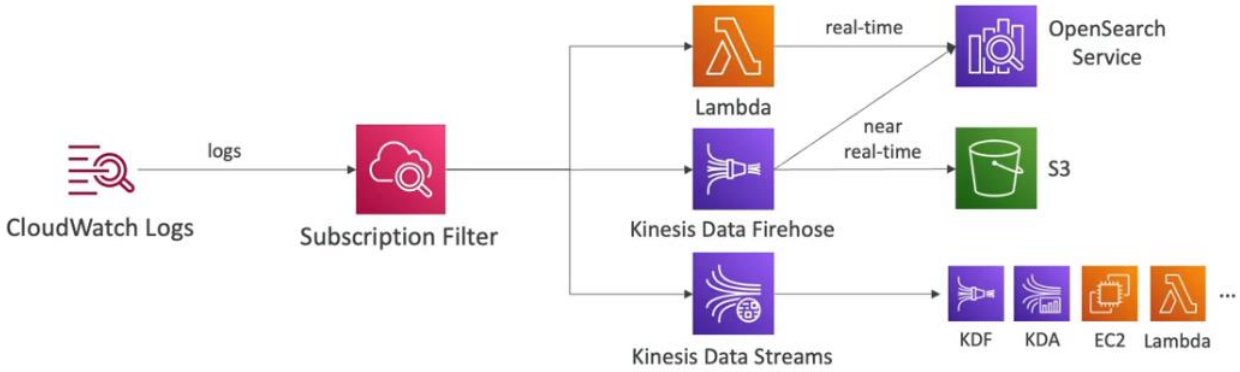
- A monitoring service for AWS cloud resources and the applications being run on AWS
- It enables resource utilization, operational performance monitoring, log aggregation and basic analysis
- It provides real-time monitoring within AWS for resources and applications
- It hooks to event triggers
- Key components – CloudWatch, CloudWatch Logs, CloudWatch Events

CloudWatch:

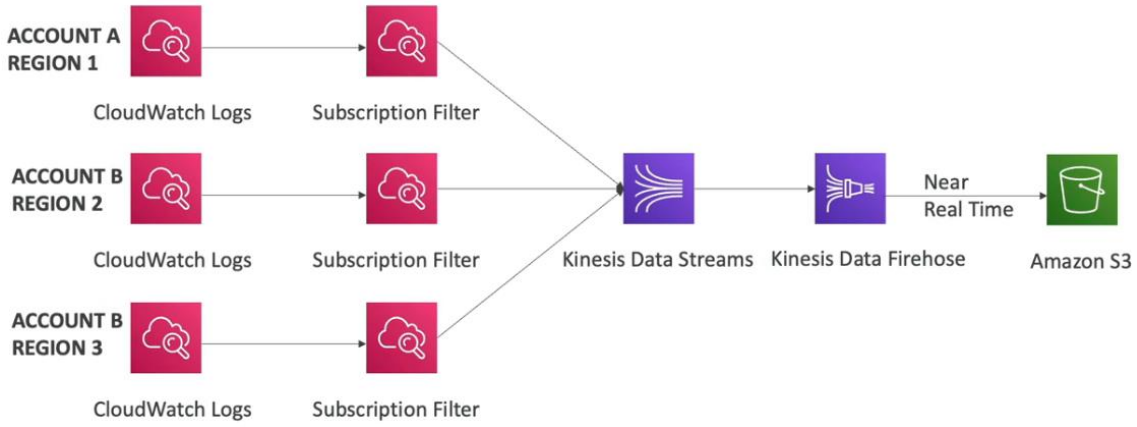
- Real-time (standard – every 5 minutes, detailed – every 1 minute)
- Metrics (CPU utilization, network utilization etc.)
- Alarms (can be set on threshold conditions etc.)
- Notifications (SNS etc.)
- Custom metrics (disk space utilization, RAM utilization etc.)

CloudWatch Logs:

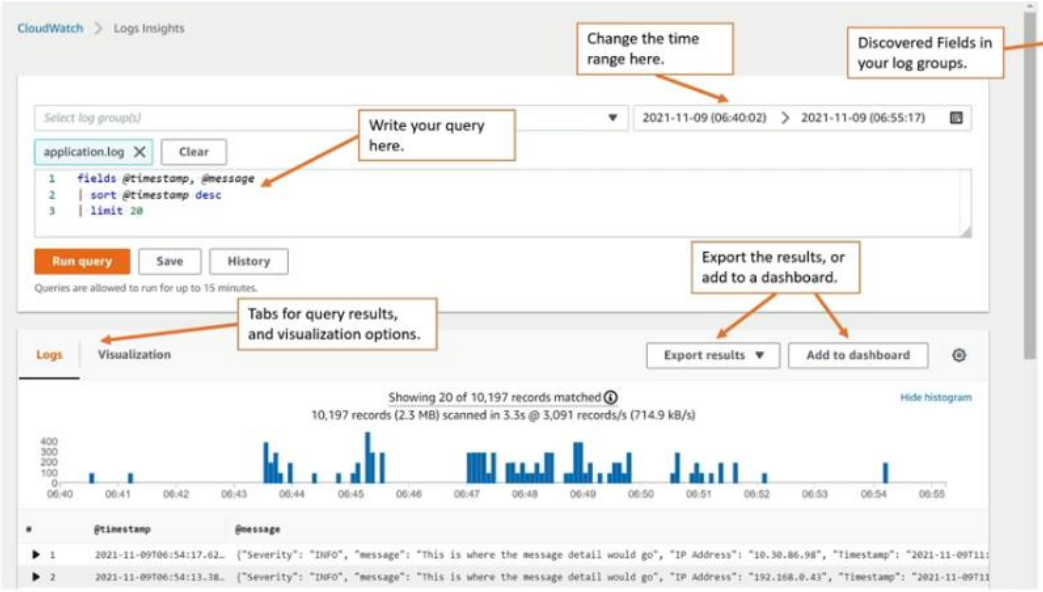
- Collect logs into central repository
    - Pushed from some AWS services (including CloudTrail)
    - Pushed from custom applications or systems
- Metrics from log entry matches
- Provides real-time visibility into log data with configurable metrics, filters, and alarms
- Retains log data for the desired duration – configurable from days to indefinitely
- Stores log data securely and durably in highly available storage
- Supports advanced querying and analysis using CloudWatch Logs Insights
- Can trigger actions or alarms based on log data through CloudWatch Events
- Integrated with AWS services like EC2, Lambda, ECS, and more



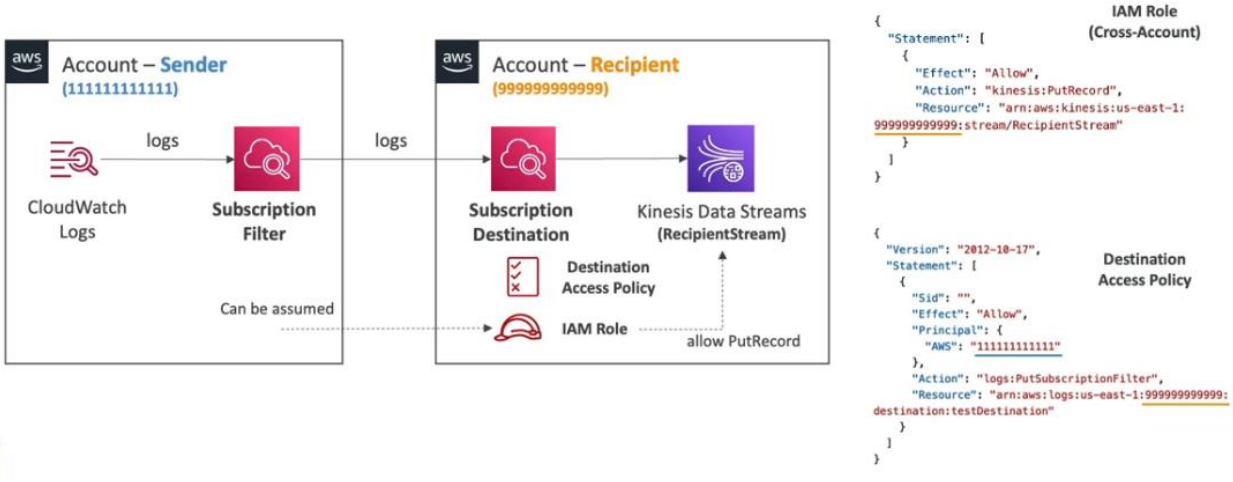
- Supports encryption of log data in transit and at rest
  - Encrypted by default
  - Can setup KMS-based encryption with user's own keys
- Provides fine-grained access control and IAM policies for log data access
- Log groups – arbitrary name, usually representing an application
- Log stream – instances with application /log files /containers
- Can send logs to – S3 (exports), Kinesis Data Streams, Kinesis Data Firehose, Lambda, OpenSearch
- Supports aggregation multi-account and multi-region



- Log sources that feed logs to CloudWatch could be:
  - SDK, CloudWatch Agent Logs, CloudWatch Unified Agent
  - Elastic Beanstalk – collection of logs from applications
  - ECS – collection from containers
  - AWS Lambda – collection from function logs
  - VPC Flow Logs – VPC specific logs
  - API Gateway
  - CloudTrail – based on filter
  - Route53 – Log DNS queries
- To query CloudWatch Logs – CloudWatch Logs Insights can be used



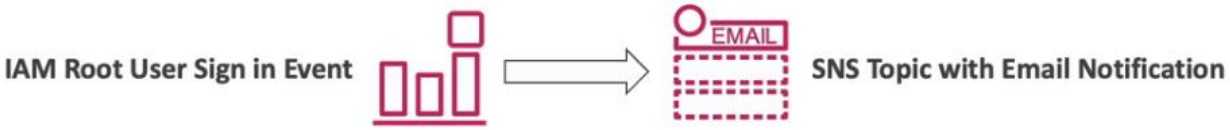
- Also supports cross-account subscription – sends log events to resource in a different AWS account (Kinesis Data Streams, Kinesis Data Fire Hose)



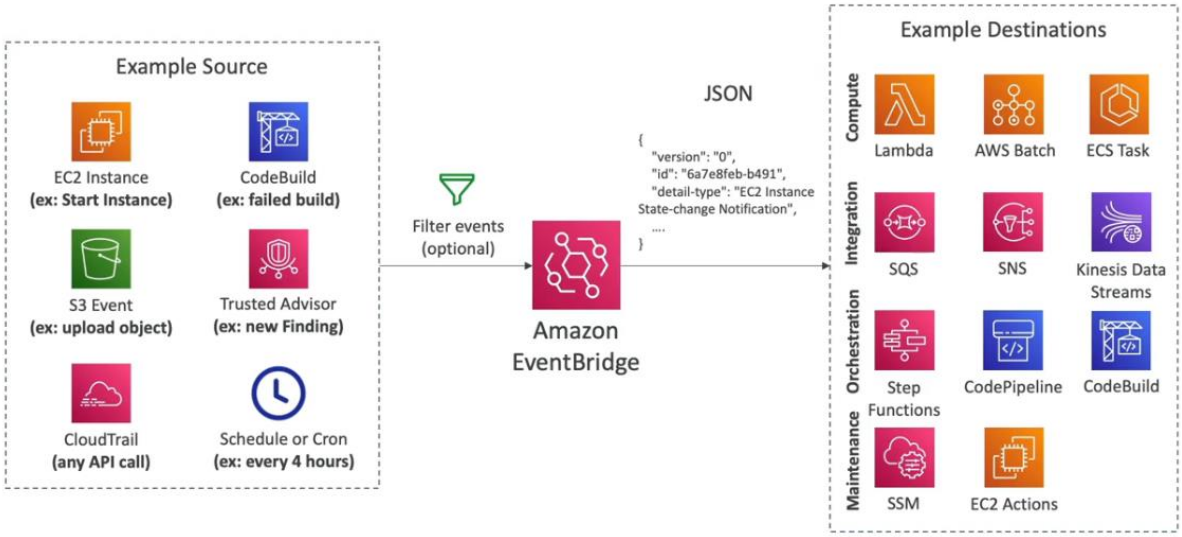
Amazon EventBridge:

- Formerly known as CloudWatch Events
- Near real-time stream of system events
- Helps to respond to state change in AWS resources – when resources change state, they automatically send events into an event stream
- A rule can be created that match selected events in the stream and route them to target to act
- Events could be as follows:
    - AWS resources state change
    - AWS CloudTrail (API calls)
    - Custom Events (Code)
    - Scheduled – cronjobs

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



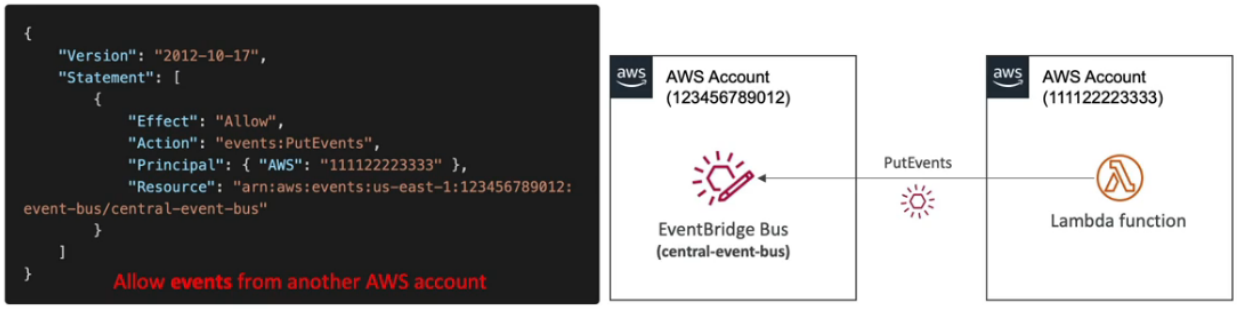
- Rules – match incoming events and route them to one or more targets
- Targets – AWS lambda functions, SNS topics, SQS queues, Kinesis Streams



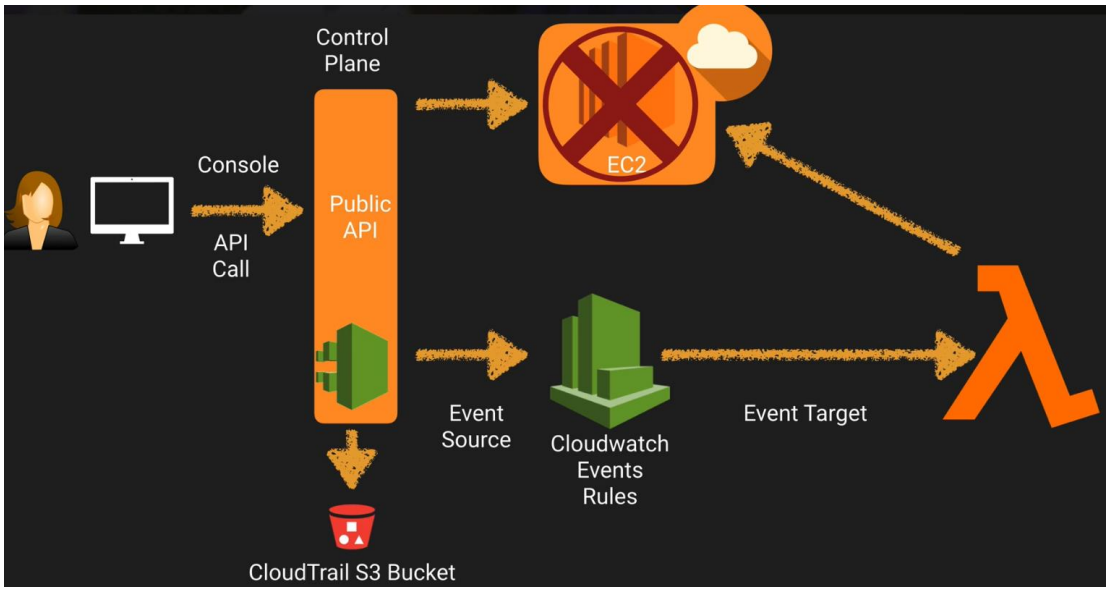
- Event buses – can be accessed by other AWS accounts using Resource-based policies
  - Archive event (all/filter) sent to event bus (set period or indefinitely)
  - Ability to replay archived events

- Schema Registry – allows to generate code for user's application, that will know in advance how data is structured in event bus
  - Event bridge can analyze the event in user's event bus and infer the schema
  - Schema can be versioned

- Resource-based policy – manage permissions for specific event bus
    - Example – allow/deny events from another AWS account or AWS region
    - Use case – aggregate all events from user's AWS Organization in a single AWS account or AWS region



- CloudWatch Events use case example could be to trigger a lambda function in response of CloudWatch Event rules where when EC2 instance will be created (with some conditional criteria), the CloudWatch Event Rules should trigger a lambda function that should delete the EC2 instance – explained in below diagram

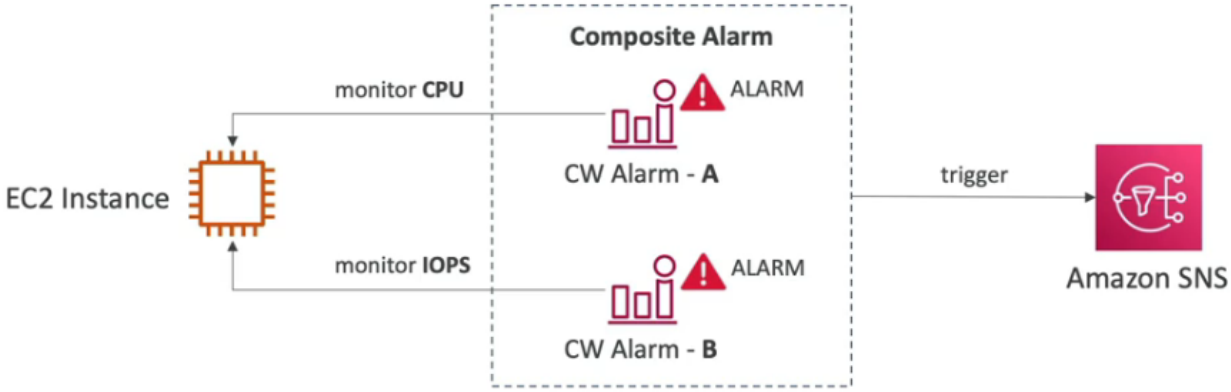


- Other examples could be like:
    - Direct specific API records from CloudTrail to a kinesis stream for detailed analysis of potential security or availability risk
    - Take a snapshot of an Amazon EBS volume on a schedule

CloudWatch Alarms:

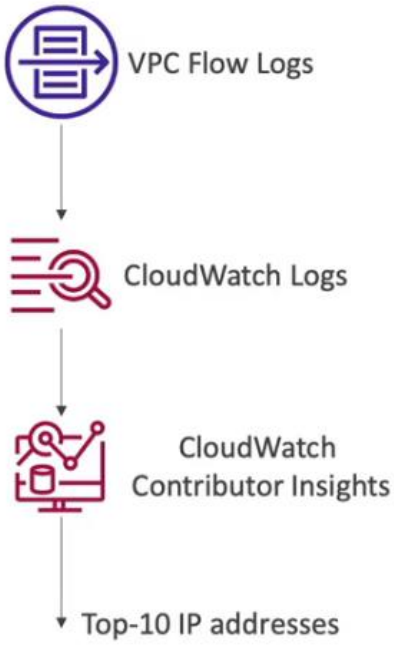
- Metric-based monitoring
- Threshold-based triggers
- Action triggering – trigger actions when alarm states change
    - Sending notifications

- Executing automated actions
- Configurable alarm states based on metric conditions
  - OK
  - INSUFFICIENT_DATA
  - ALARM
- Multiple notification options – email, SNS, SMS, Lambda
- Auto-recovery actions
- Period – length of time in seconds to evaluate the metric
- Alarm targets
  - Stop, Terminate, Reboot or Recover an EC2 instance
  - Trigger Auto Scaling Action
  - Send notification to SNS
- Composite Alarms – monitoring the states of multiple other alarms
  - AND / OR conditions
  - Helpful to reduce alarm noise by creating complex composite alarms



Contributor Insight:

- Analyze log data and create time series that display contributor data
- Help to find top talkers and understand who/what is impacting system performance
  - Find bad hosts
  - Identify heaviest network users
  - Find the URLs that generates the most errors
- Works for any AWS-generated logs – VPC, DNS etc.



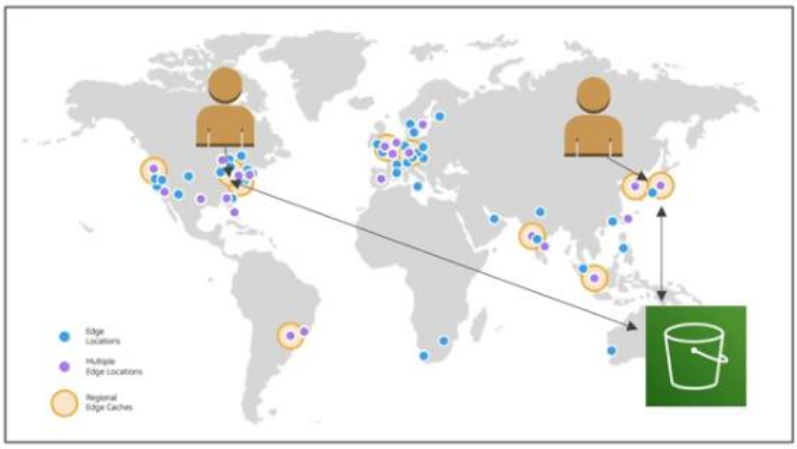
Securing CloudWatch:

- How do it be controlled that who can access CloudWatch and what they can do?
  - Use IAM policies to restrict access to CloudWatch and the actions they can perform
  - However, data is decoupled from its source, therefore restricting access by the originating resource is not possible
- How are unauthorized users prevented from accessing CloudWatch?
  - Users need to be authenticated with AWS and have the appropriate permissions set via IAM policies to gain access
  - All AWS APIs use signed requests via HTTPS/SSL to help mitigate man-in-the-middle attacks

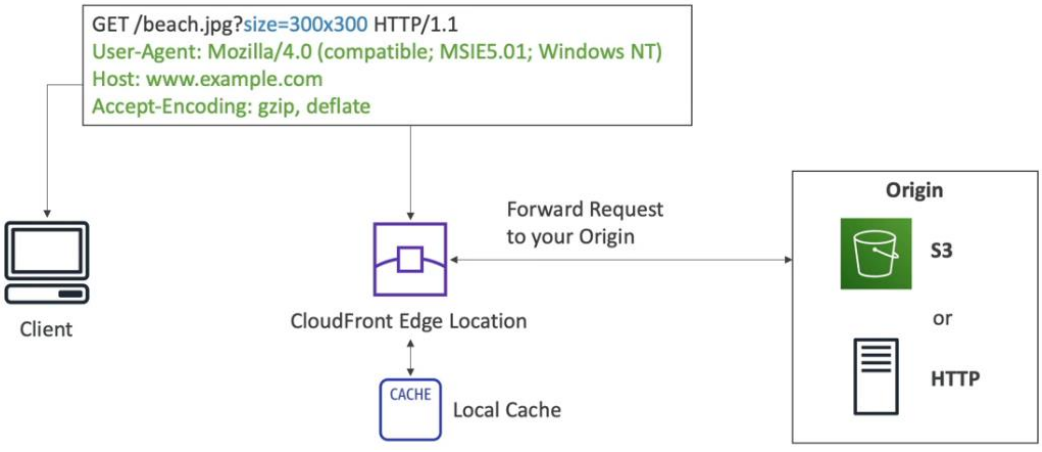
AWS CloudFront:

- Global content delivery network (CDN) service offered by AWS
- Accelerates the delivery of static and dynamic web content to end users by caching content at edge locations – 216 edge locations currently
- CloudFront improves performance and reduces latency by serving content from the nearest edge location to the user's location
- It supports the various types of content, including web pages, images, videos, APIs, and streaming media
- Seamlessly integrates with other AWS services like S3, EC2, and Lambda – for easy content distribution and dynamic content generation
- Provides advanced caching options, such as edge caching, origin caching, and object-level caching – to optimize content delivery

- Supports secure content delivery using SSL/TLS encryption and integration with AWS Certificate Manager
- Offers real-time monitoring and analytics through CloudFront access logs, CloudWatch metrics, and integration with AWS WAF
- DDoS protection (because worldwide) – integration with AWS Shield and AWS WAF
- Provides features like geo-restriction, which allows content to be restricted based on the geographic location of the user

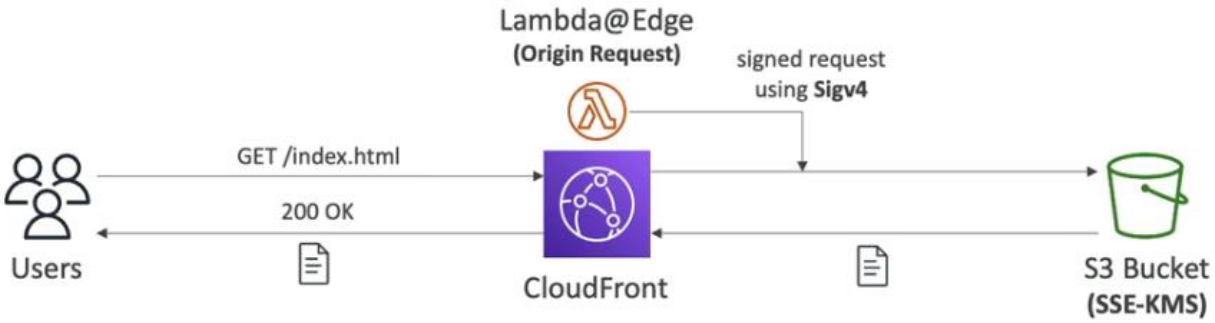


- Supports streaming of live and on-demand media content through integration with AWS Element Media services
- CloudFront is a global service; however, the configuration is maintained in N. Virginia. So, you need to generate one certificate in N. Virginia for CloudFront distribution
- If you want to require HTTPS between viewers and CloudFront, you must change the AWS region to US East 1 (N. Virginia) in the AWS Certificate Manager console before you request or import a certificate
- Offers scalability and high availability, with automatic scaling and distribution of content across multiple edge locations worldwide



- CloudFront Origins:

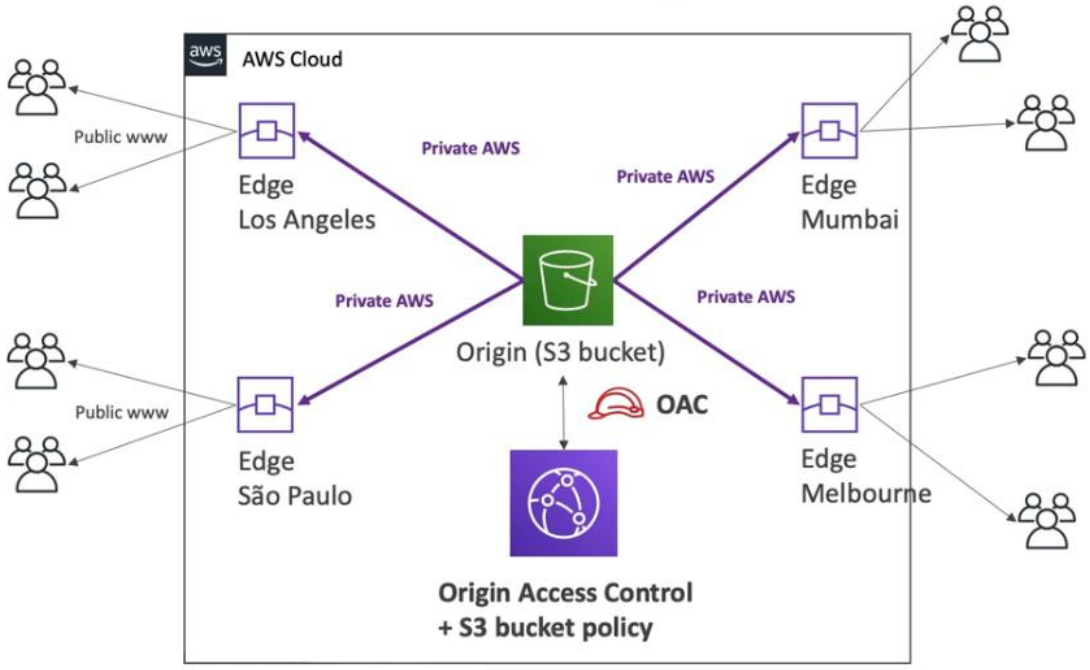
- S3 buckets – for distrusting files and caching them at the edge
  - Enhanced security with CloudFront Origin Access Control (OAC)
    - OAC is replacing Origin Access Identity (OAI)



- Lambda@Edge allows you to run AWS Lambda functions in response to CloudFront events, which enables dynamic, personalized content generation close to your users
- You can write Lambda@Edge functions to modify headers, which would increase security without changing application code
- OAC supports SSE-KMS natively (as requests are signed with Sigv4)
- Add a statement to the KMS Key Policy to authorize the OAC

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

  - CloudFront can be used as an ingress (to upload files to S3)



- Custom Origin – HTTP
  - ALB, EC2 instance, S3 website (must first enable the bucket at S3 as a statis S3 website)
  - Any HTTP backend

- Users must use a custom certificate if they want to use their own domain name. The certificate may be stored in IAM or in ACM
- To use an ACM Certificate with Amazon CloudFront, user must request or import the certificate in the US East (N. Virginia) region
  - ACM Certificates in this region that are associated with a CloudFront distribution are distributed to all the geographic locations configured for that distribution
- Which steps are required to be taken to force users to access the site using CloudFront and not directly using the S3 URL?
  - Create an origin access identity, which is a special CloudFront user
  - associate the origin access identity with your distribution
  - Change the permissions either on your Amazon S3 bucket or on the files in your bucket so that only the origin access identity has read permission (or read and download permission)
  - When your users access your Amazon S3 files through CloudFront, the CloudFront origin access identity gets the files on behalf of your users
  - If your users request files directly by using Amazon S3 URLs, they're denied access

Geo Restriction:

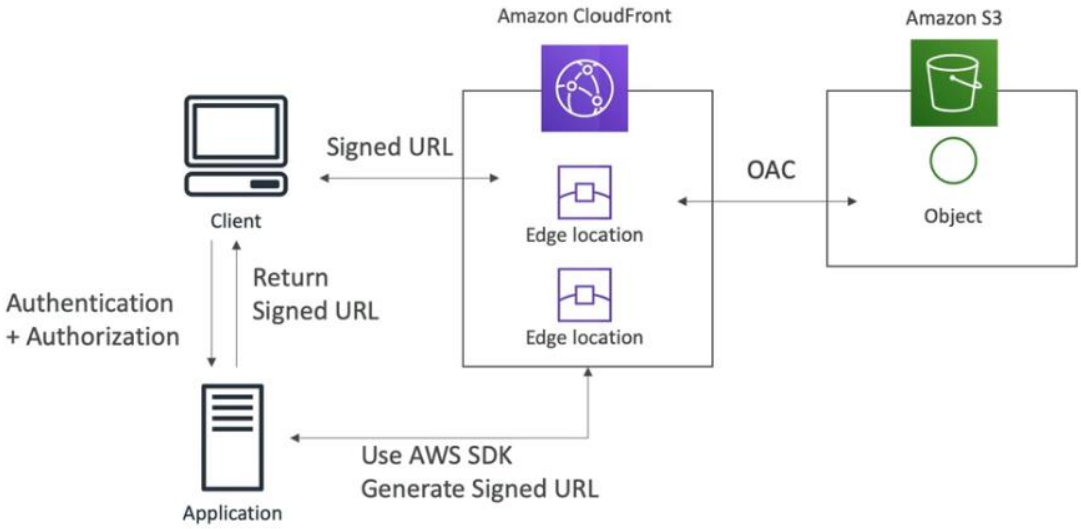
- Allows access to content based on the geographic location of the viewer
- It helps comply with content distribution
- Two methods for implementing geo restriction – whitelisting and blacklisting

- Whitelisting – allows access from specified countries or geographic locations
- Blacklist – restricts access from specified countries or geographic locations

- The **country** is determined using 3rd party Geo-IP database
- Configured through CloudFront's web distribution settings or API
- Useful for controlling access to sensitive or restricted content based on location
- Con be combined with other CloudFront features like signed URLs or cookies

Signed URLs /Signed Cookies:

- Provide temporary access permissions to specific resources – useful for granting time-limited access to private or restricted content
  - Signed URLs allow access to individual files or objects in a distribution
  - Signed cookies enable access to multiple files within a specific duration

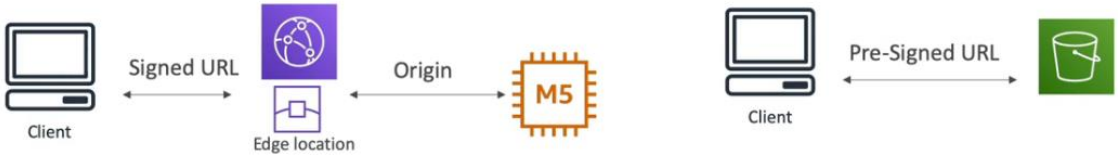


- Both mechanisms use cryptographic signatures to authenticate and authorize access
- They contain expiration time, access permissions, and other configurable parameters – policy can be attached with
  - Includes URL expiration
  - Includes IP ranges to access the data from
  - Trusted signers – which AWS accounts can create signed URLs
- They can be generated via CloudFront APIs or SDKs
- Two types of CloudFront Signed URL Process signers:
  - Either a trusted key group – recommended
    - Can leverage API to create and rotate keys and IAM for API security
    - In CloudFront distribution, create one or more trusted key groups
    - Generate your own private/public key
      - Private key is used by your application to sign URLs
      - Public key (uploaded) is used by CloudFront to verify URLs
  - An AWS account that contains a CloudFront Key Pair
    - Need to managed keys using the root account and the AWS Console

- Not recommended

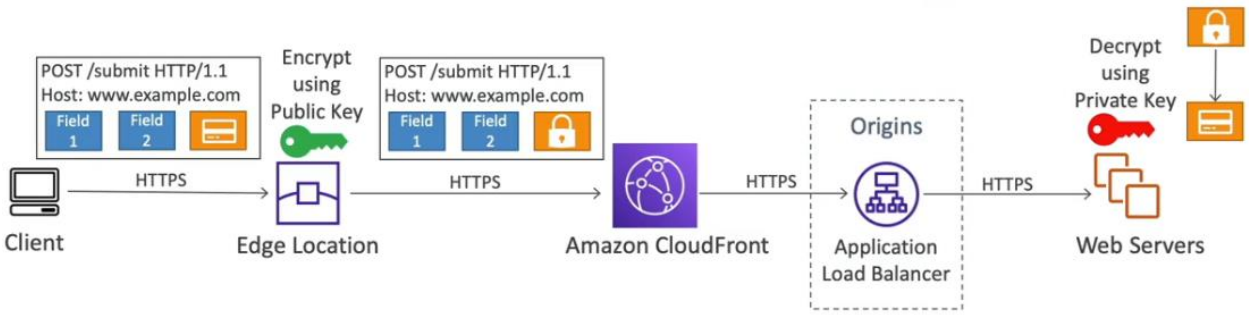
CloudFront Signed URL vs S3 Pre-signed URL:

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



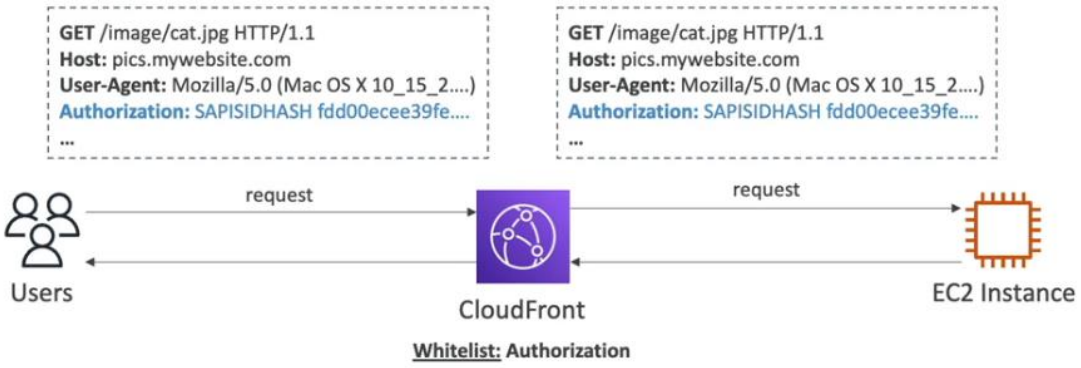
Field Level Encryption:

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



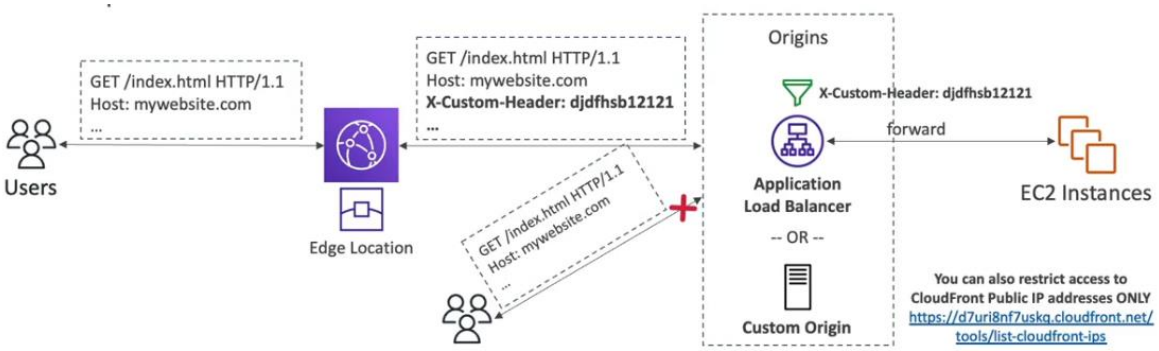
Authorization Header:

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

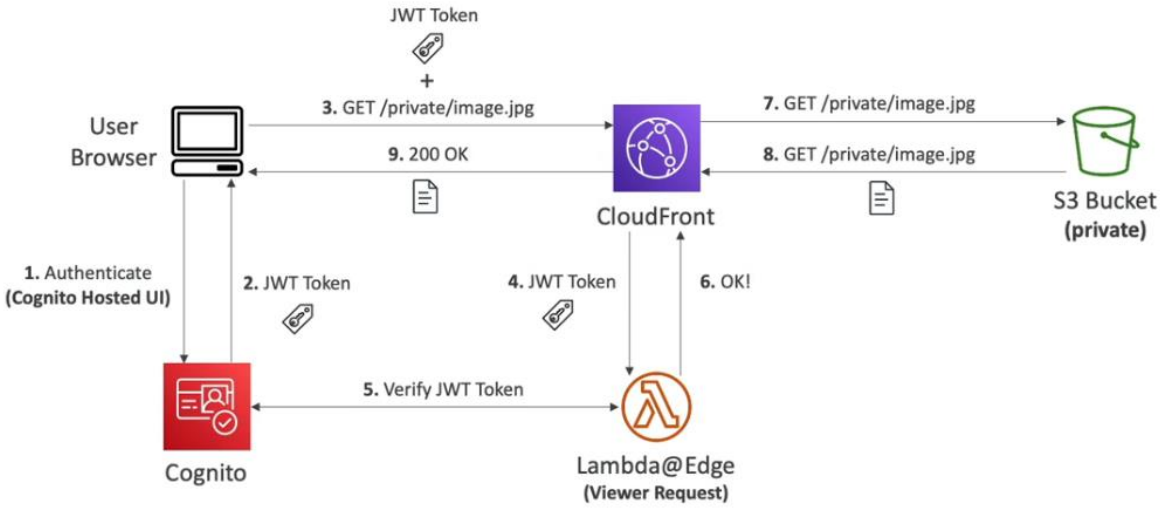


Restrict access to ALB:

- Prevent direct access to ALB or Custom Origins – only access through CloudFront
- First, configure CloudFront to add Custom HTTP header to requests it sends to ALB
- Second, configure the ALB to only forward requests that contain that custom HTTP Header



Integration with Cognito:

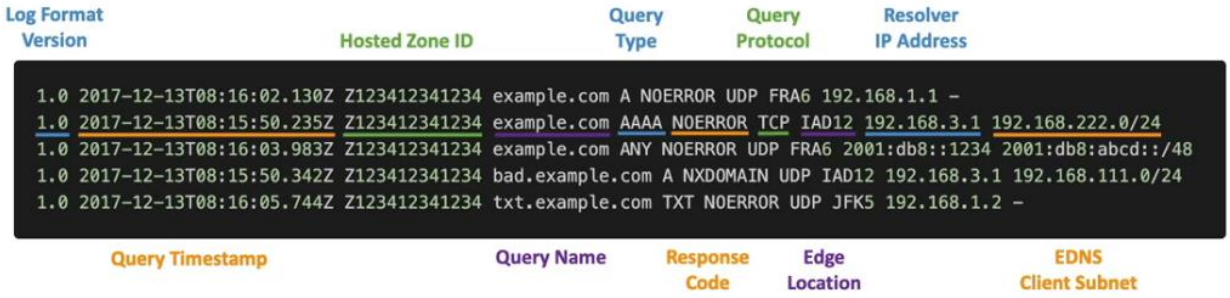


Amazon Route53:

- Provides domain registration, DNS routing, and health checking for the applications
- Supports both public and private DNS configurations
- Offers global coverage with low latency DNS resolution
- Allows to manage DNS records, such as A, CNAME, MX, and TXT records
- Supports advanced features like weighted routing, geolocation-based routing, and latency-based routing
- Provides DNS failover and health checks to monitor availability of resources
- Integrates with other AWS services – ELB, CloudFront, S3 etc.
- Enables easy management of DNS setting through the AWS Management Console, CLI, or SDKs
- Supports domain registration and DNS management for both AWS and non-AWS resources
- Offers DNS query logging for analysis and troubleshooting
- Supports routing policies to control traffic flow and implement sophisticated routing strategies
- Offers DNSSEC – for enhanced security and integrity

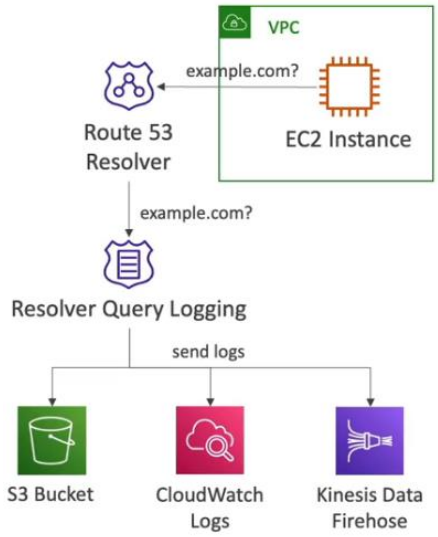
DNS Query Logging:

- Log information about public DNS queries Route53 Resolver receives
- Only for Public Hosted Zones
- Logs are sent to CloudWatch Logs only

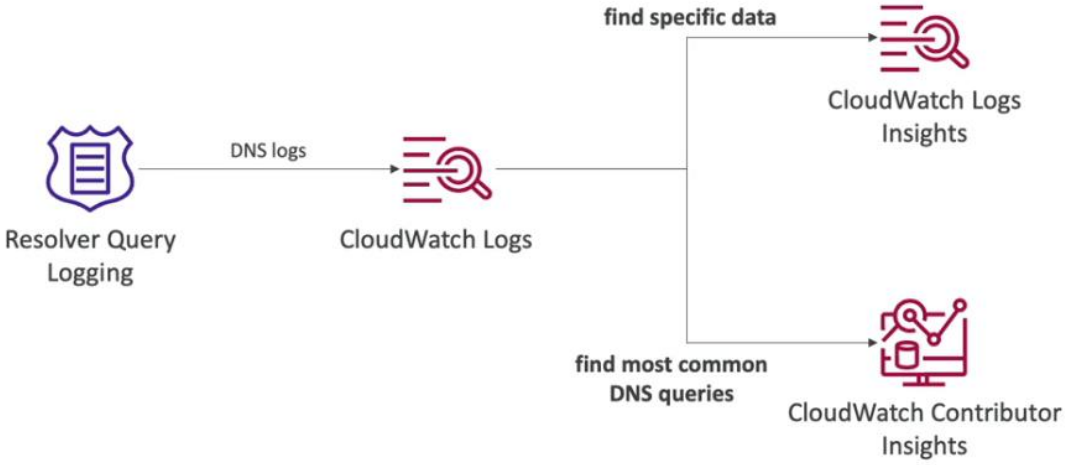


Resolver Query Logging:

- Logs all DNS queries of the following:
  - Made by resources within a VPC – EC2, Lambda etc.
  - From on-premises resources that are using Resolver Inbound Endpoints
  - Leveraging Resolvers Outbound Endpoints
  - Using Resolver DNS firewall
- Can send logs to – CloudWatch Logs, S3 bucket, Kinesis Data Firehose



- Configuration can be shared with other AWS accounts using AWS RAM
- Use case could be like as below diagram:

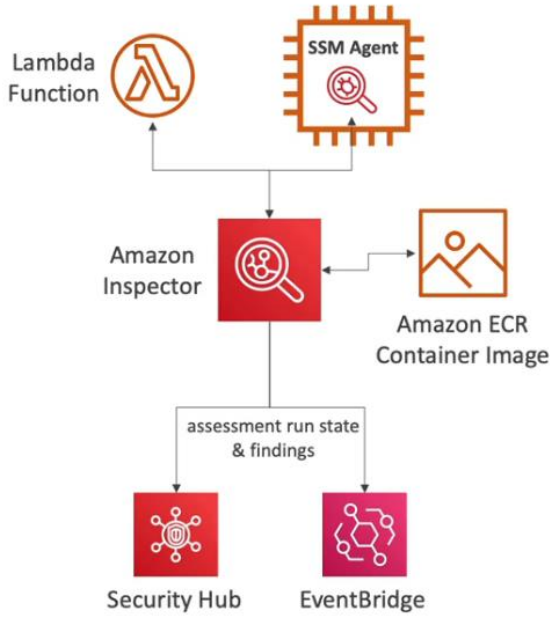


AWS Inspector:

- An automated security assessment service that helps improve the security and compliance of applications deployed on AWS
- Automatically assesses applications for vulnerabilities or deviations from best practices
- After performing an assessment, Amazon Inspector produces a detailed list of security findings prioritized by a level of security
- These findings can be reviewed directly or as part of detailed assessment reports which are available via the Amazon Inspector console or API
- Two types of assessments:

- Network assessment – no inspector agent required
- Host assessment – inspector agent is required
  - EC2 instances – leveraging SSM Agent
    - Unintended network accessibility
    - Running OS against known vulnerabilities
  - Container images push to ECR
    - Assessment of container images as they are pushed
  - Lambda functions – software vulnerabilities in function code and package dependencies

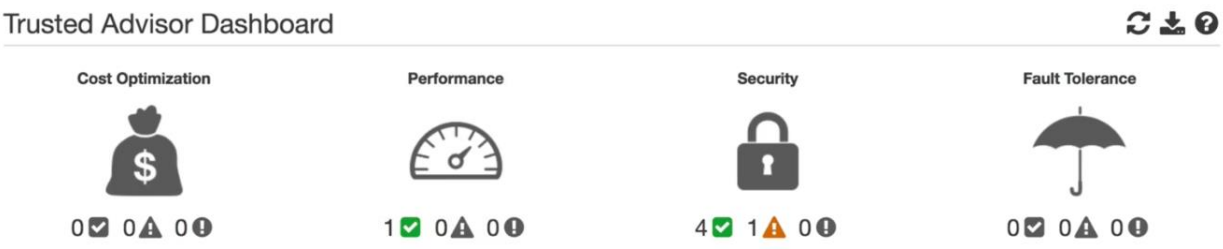
- Rule Packages:
  - Common Vulnerabilities and Exposures
  - CIS Operating System Security Configuration Benchmarks
  - Security Best Practices
  - Runtime Behavior Analysis
- Severity Level – High, Medium, Low, Informational
- It will:
  - monitor the network, file system, and process activity within the specified target,
  - compare what it sees to security rules,
  - report on security issues observed within the target during run,
  - report findings and advise remediation
- How does it work?
  - Create “Assessment Target”
  - Install agent on EC2 instances
  - Create “Assessment Template”
  - Perform “Assessment Run”
  - Review “Findings” against “Rules”
- The rules in the Common Vulnerabilities and Exposures package does not check for instances which enable root login over SSH
  - Security Best Practices will report on instances which allow root login over SSH
  - Amazon Inspector provides the Center for Internet Security Benchmarks rules packages to help establish secure configuration posture reports which include root login over SSH information
- Reporting and integration can be done with AWS Security Hub
- It can send findings to Amazon Event Bridge



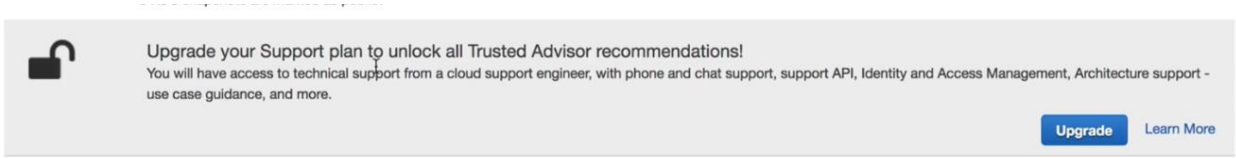
- 15 days free trail

AWS Trust Advisor:

- AWS Trusted Advisor is an online tool that provides real time guidance to help provision resources following AWS best practices
- It analyzes the AWS account and provides recommendations across multiple categories like:
  - Cost Optimization
  - Performance
  - Security
  - Fault Tolerance
  - Service limits



- Full trusted advisor or unlock more features – upgrade to enterprise



- It provides actionable insights such as idle resources, underutilized instances, open security groups, unrestricted access, and more
- Performs checks automatically and provides detailed information on each recommendation, including impact analysis and steps to resolve the issue
- Provides a dashboard with an overview of the user's account status with a list of recommendations for each category
- It offers API integration for programmatically accessing Trust Advisor data and automating actions based on recommendations
- All AWS customers get access to the seven core Trusted Advisor checks to help increase the security and performance of the AWS environment, including: S3 Bucket Permissions, Security Groups - Specific Ports Unrestricted, IAM Use, MFA on Root Account, EBS Public Snapshots, RDS Public Snapshots.
- Support plans:
    - Basic & Developer Support plan – 7 core checks
        - S3 bucket permissions
        - Security groups – specific ports unrestricted
        - IAM use (one IAM user minimum)
        - MFA on root account
        - EBS public snapshots
        - RDS public snapshots
        - Service limits
    - Business & Enterprise plan – full checks
        - Full checks available on 5 categories
        - Ability to set CloudWatch alarms when reaching limits
        - Programmatic access using AWS Support API

AWS Hypervisor:

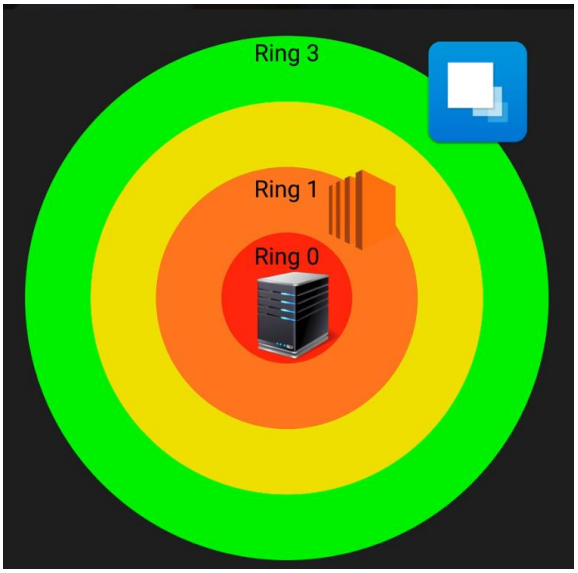
- A hypervisor or virtual machine monitor is a computer software, firmware or hardware that creates and runs virtual machines
- A computer or which a hypervisor runs one or more virtual machines is called a host machine, and each virtual machine is called a guest machine
- EC2 runs on a mixture of Nitro and Xen Hypervisors – eventually all EC2 will be based off Nitro Hypervisors
- Both hypervisors can have guest operating systems running either as Paravirtualization (PV) or using Hardware Virtual Machine (HVM)
- HVM guests are fully virtualized – the VMs on top of the hypervisors are not aware that they are sharing processing time with other VMs
- While PV are lighter and quicker
- However, this performance gap has now closed, and Amazon now recommend using HVM over PV where possible – Windows EC2 instances can only be HVM where Linux can be both HVM and PV

Access Mechanism:

- Hypervisor Access:
  - Administrators with a business need to access the management plane are required to use multifactor authentication to gain access to purpose-built administration hosts
  - These administrative hosts are systems that are specifically designed, built, configured, and hardened to protect the management plane of the cloud
  - All such access is logged and audited
  - When an employee no longer has a business need to access the management plane, the privileges and access to these hosts and relevant systems can be revoked
- Guest OS Access:
  - Virtual instances are completely controlled by the customer
  - Customer will have full root access or administrative control over accounts, services, and applications
  - AWS does not have any access rights to the customer's instances or guest OS

Paravirtualization:

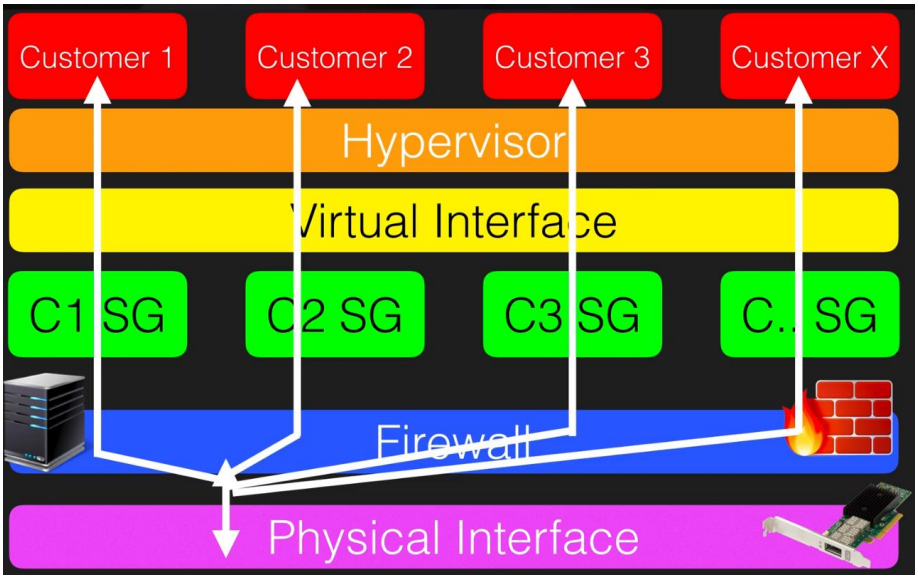
- Lighter from of virtualization – quicker
- Paravirtualized guests rely on the hypervisor to provide support for operations that normally require privileges access – the guest OS has no elevated access to the CPU
- The CPU provides 4 separate privilege modes – 0 to 3 called rings:
  - Ring 0 is the most privileged
  - Ring 3 is the least privileged
  - The host OS executes in Ring 0
  - The guest OS executes in Ring 1
  - Applications in Ring 3



Isolation:

- Hypervisor provides isolation by virtualizing the underlying hardware resources of the physical host machines and present them on each instance as if it were running on dedicated hardware

- AWS uses a customized version of the Xen hypervisor, which is designed to provide strong isolation between instances
- The AWS hypervisor ensures that each instance is isolated from the others, preventing one instance from accessing or interfering with another instance's data resources – this is achieved through hardware-enforced memory and device isolation, as well as strict access controls and separation of network traffic
- AWS provides a range of security services and features, such as security groups and NACLs, that can be used to restrict network access between instances



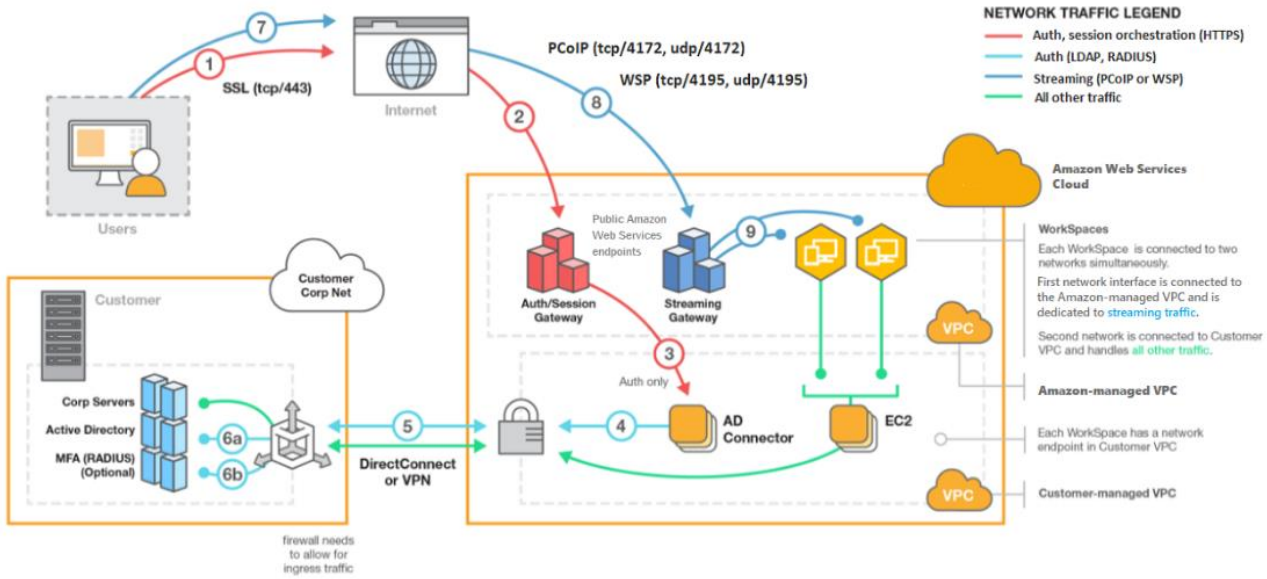
Memory Scrubbing:

- EBS automatically resets every block of storage used by the customer, so that one customer's data is never unintentionally exposed to another
- Memory allocated to guests is scrubbed (set to zero) by the hypervisor when it is unallocated to a guest
- The memory is not returned to the pool of free memory available for new allocations until the memory scrubbing is complete

Amazon WorkSpaces:

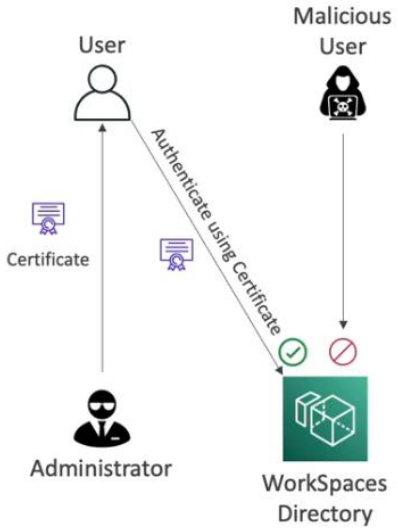
- Fully managed desktop computing service provided by AWS
- Allows users to access virtual Windows or Linux desktops in the cloud from any supported device
- Provides a secure and persistent desktop experience with customizable hardware resources
- Supports integration with AD – centralized user management and authentication
- Users can use the same tools to manage WorkSpaces that they use to manage their on-premises desktops
- Protocols – PCoIP or WSP (WorkSpaces Streaming Protocol)

- Use KMS to encrypt data at rest
- Control the IP addresses from which users are allowed to access their WorkSpaces



WorkSpaces Security:

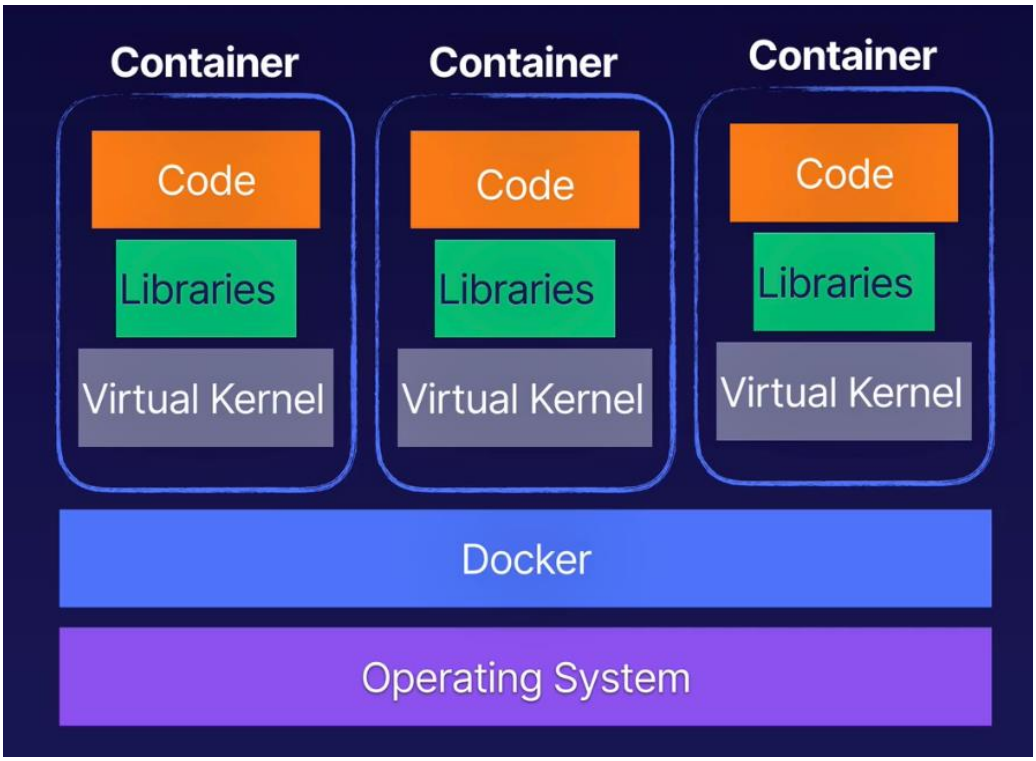
- IP Access Control Group - similar to security groups but for WorkSpaces
  - List of IP addresses / CIDR address ranges that users are authorized to connect from
  - If users access WorkSpaces through VPN or NAT, the IP Access Control Group must authorize the public IP of these
- Access Control Options and Trusted Devices – manage which client devices can access WorkSpaces
  - WorkSpaces Certificate-based authentication – limit access only to trusted devices using Digital Certificates
  - Only Windows, MacOS, and Android Clients



AWS Microservices:

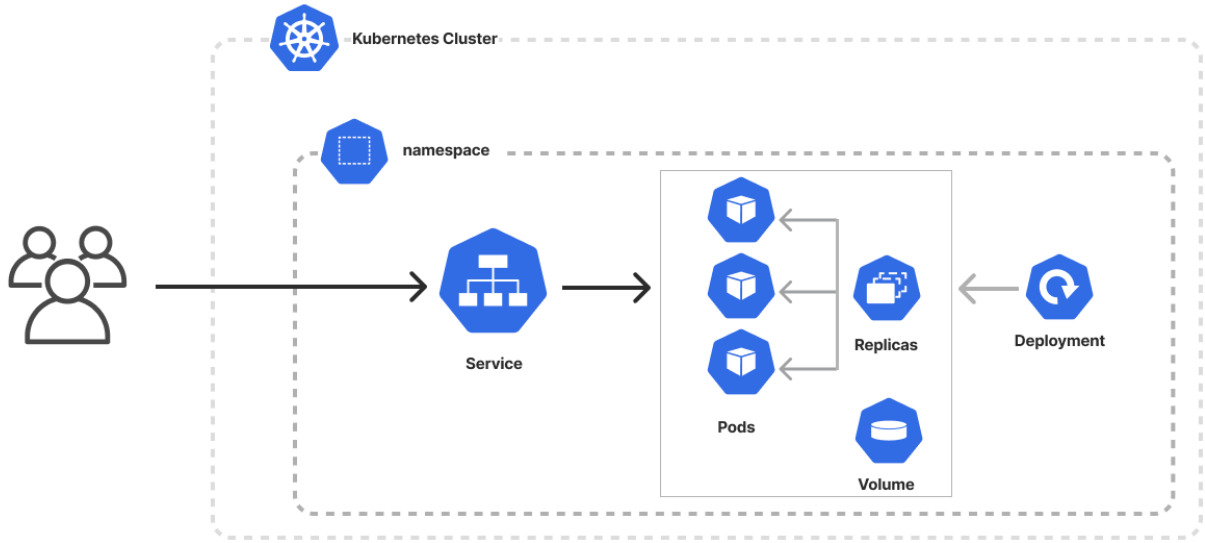
- Architectural approach of building applications as a collection of small, loosely coupled, and independently deployable services
- Each service focuses on a specific business capability and can be developed, deployed, and scaled independently
- These services communicate over well-defined APIs
- Some of the key AWS services commonly used in building microservices include:
    - AWS Lambda
    - Amazon API Gateway
    - Amazon Elastic Container – ECS
    - Amazon Elastic Kubernetes – EKS
    - AWS Step Functions
    - AWS App Mesh
    - Amazon DynamoDB
    - Amazon Aurora
    - AWS CloudFormation
- Many modern applications are made up of a bunch of microservices running in the containers
- Advantages of this architectural approach:
    - Serviceability – easy to fix problem
    - Flexibility – easy to make changes
    - Scalability – Easy to scale, independent scaling
    - Fault isolation – less risk of cascading failures
    - Team autonomy – decentralized decision-making
    - Enhance security and fault tolerance
- Containers:
    - Virtual operating environment
    - Lightweight, standalone, and executable units that encapsulate software components (code, libraries, their dependencies (system tools), and configurations
    - Provide an isolated and portable environment for running microservices, ensuring consistency across different platforms and simplifying deployment and management processes
    - Containers use containerization technologies like Docker – each microservice is packaged into its own container, allowing it to be deployed and executed independently of other microservices in the system
- Dockers:
    - An open-source tool that allows to create, manage, and run containers
    - Provides a user-friendly interface and a CLI that simplifies the process of creating and working with containers
    - It uses the containerization technique to package applications and their dependencies into lightweight and portable containers

- With docker, containers can be built using Dockerfiles, which are text files that define the steps to create a container with specific configurations and dependencies
- Docker images can be shared and distributed through container registries, such as Docker Hub
- It also provides tools for managing container lifecycles, including starting, stopping, restarting containers
- Also offers networking capabilities for connecting containers and exposing ports to enable communication between them and the external world
- It also allows to scale applications by running multiple instances of the same container across different hosts



- Kubernetes:
  - On open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications
  - Developed by Google
  - Allows to efficiently manage and coordinate containers across a cluster of machines
  - Abstracts away the complexities of infrastructure management and provides a consistent, declarative API to define the desired state of applications and their dependencies
  - **Pod –** the basic unit of deployment in Kubernetes
    - A group of one or more containers that share resources and are scheduled together on the same host
    - Represents the small deployable units and encapsulate application logic

- **Service –** provides a stable network endpoint to access a group of pods
  - It enables load balancing and dynamic discovery of pods so that the other services or external clients can communicate with the application
  - It abstracts the underlying IP addresses and provides a consistent way to access and scale the applications
- **Deployment –** a higher-level resource that defines the desired state of a number of pods
  - It manages the rollout and scaling of replicas, ensuring that the desired number of pods are running, and updates are performed seamlessly with minimal disruption
- **ReplicaSet –** is responsible for maintaining a specific number of replicas (pods) in the cluster
  - It ensures high availability and scalability by creating or terminating pods as needed to match the desire state defined by a deployment
- **Namespace** – virtual clusters within a physical cluster
  - Provides isolation and help organize and manage resources by dividing the cluster into logical segments
- **Persistent Volumes** – used to provide persistent storage for applications running on pods
  - Associated with a storage backend
  - Can be dynamically provisioned or statically assigned to pods



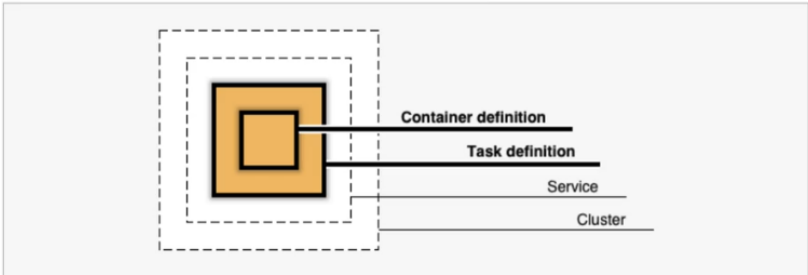
Where do run containers in AWS?

Amazon ECS:

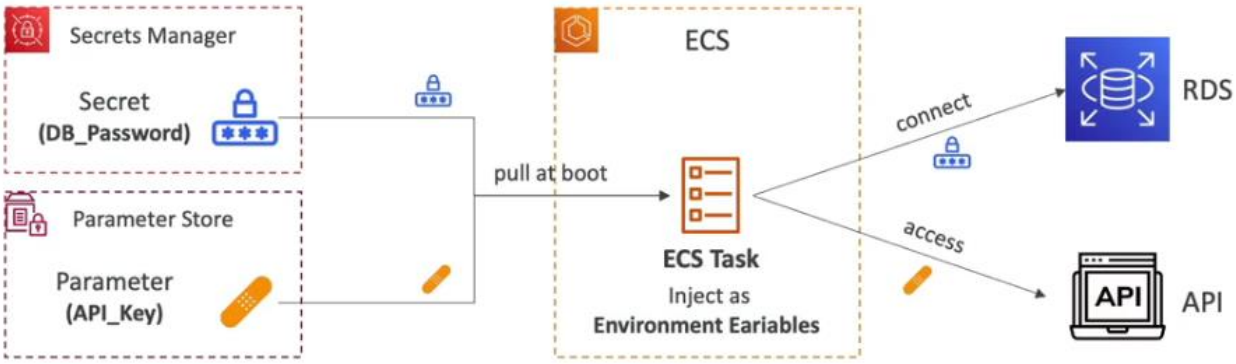
- Elastic Container Service
- Fully managed container orchestration service provided by AWS

- Allows to easily run, manage, and scale Docker containers on AWS infra

Diagram of ECS objects and how they relate



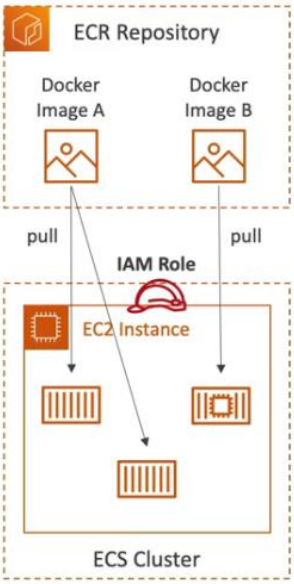
- With ECS – containers can be deployed and managed without the need to manually provision or manage the underlying infrastructure
- It abstracts away the complexity of managing servers
- Allows to create and manage clusters, which are logical groups of EC2 instances or AWS Fargate tasks that run the containers
- AWS Fargate – a serverless compute engine for containers – when ECS run with Fargate; containers can be run without having to provision or manage the underlying servers or clusters
- With Fargate, simply define containers' resource requirements and launch them as tasks – Fargate takes care of provisioning and managing the necessary compute resources to run the containers, automatically scaling them based on demand
- ECS can be integrated with SSM Parameter Store & Secrets Manager
    - ECS can inject sensitive data into containers as environment variables
    - SSM Parameter Store or Secrets Manager can be used for storing sensitive data
    - Then this sensitive data can be referenced in the container definition



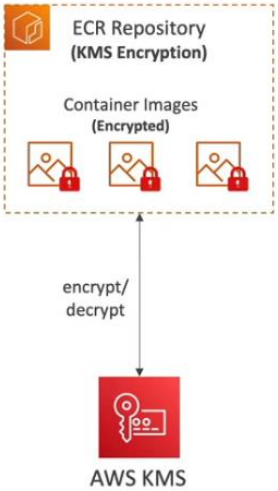
Amazon ECR:

- Elastic Container Registry
- A fully managed Docker container registry that makes it easy to store, manage, and deploy container images – secure and scalable
- Fully integrated with ECS, backed by S3
    - Can be integrated with EKS
- Access and authorization are controlled through IAM

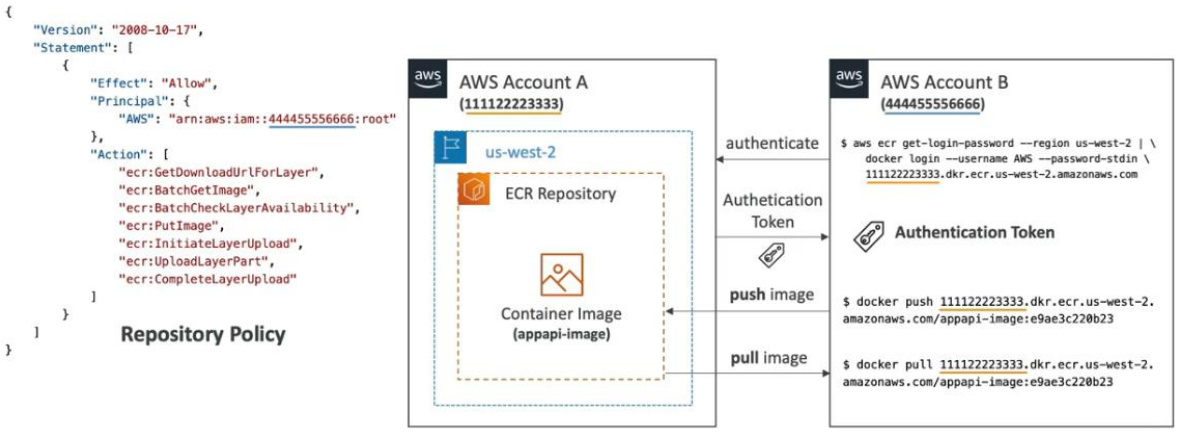
- Provides a registry API for easy push and pull operations using Docker CLI or SDKs
- Supports private and public repositories



- Lifecycle policies can be configured to automatically clean up unused or expired images
- Includes built-in image vulnerability scanning to identify and address security issues
- Supports image tags, versioning etc.
- Supports image replication across different regions
- Supports encryption of images at rest using KMS
    - Container images are encrypted using a unique Data Encryption Key – envelope encryption
    - ECR creates KMS Grant on user's behalf to interact with KMS with following permissions
        - *kms:DescribeKey*, *kms:Decrypt*, *kms:GenerateDataKey*, *kms:RetireGrant*

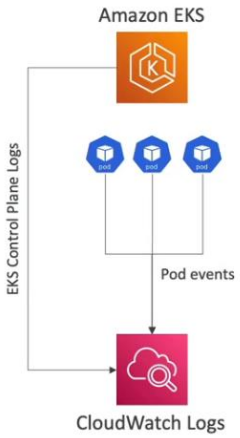


- ECR – Cross-Account Access – can be done via Repository Policy



Amazon EKS:

- Elastic Kubernetes Service - simplifies the process of running Kubernetes clusters on AWS infrastructure
- Automates the provisioning, scaling, and management of Kubernetes control plane
- Integrates with other AWS services – EC2, ELB, EBS, and many more
- Supports both Windows and Linux containers and compatible with standard Kubernetes tooling and APIs
- Supports automatic scaling of worker nodes based on workload demands
- Offers native integration with IAM for authentication and authorization
- Provides built-in security features – encryption, VPC networking, IAM roles
- Supports integration with AWS Fargate – run Kubernetes pods without managing the underlying infrastructure
- EKS Logging:
    - Pod events, node events etc.
    - EKS default event TTL is 60 minutes – can't be changed
    - To keep history of Kubernetes events beyond 60 minutes – send to CloudWatch Logs
    - EKS control plane logging – ability to select the exact log types to be sent to CloudWatch Logs
        - Audit logs, diagnostic logs, controller logs



Container Security:

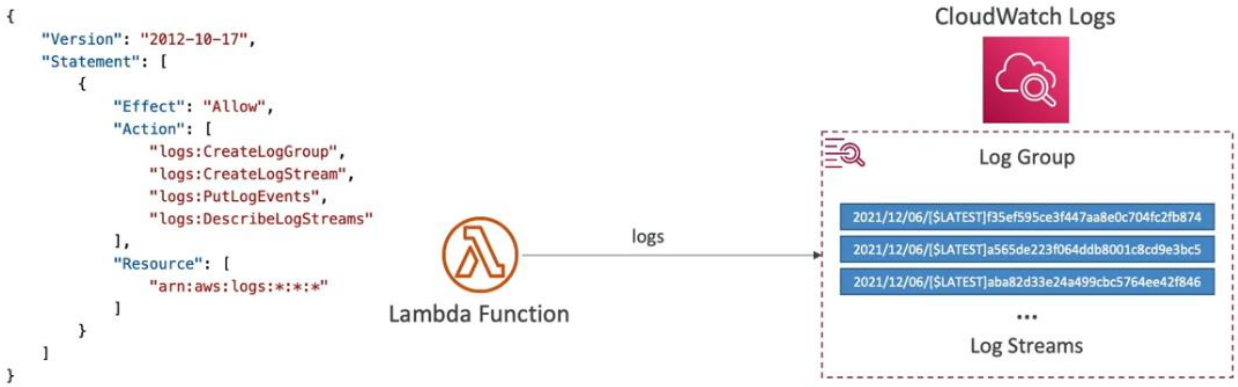
- Don't store secrets and passwords in Containers
  - Instead use Secret Manager
  - Use IAM roles
- Don't run Containers as root – instead use service account
- Use one service per Container
  - Minimize the attack surface
  - Less is More
  - Avoid unnecessary libraries
- Use trusted Images only – avoid using images from public repositories like DockerHub
- Image Scanning – scan for common vulnerabilities and exposures
  - Keep the container images up-to-dated
- Protect infrastructure – use ECS Interface Endpoints to avoid scaling VPC traffic over the internet
- Use encryption in transit using TLS - Amazon Certificate Manger (ACM)
- Isolate Containers – running each service within a separate container
  - Use Kubernetes
- Implement Role-Based Access Control (RABC) – ensures that only authorized individuals or services can manage and interact with containers
- Employ network segmentation

AWS Lambda:

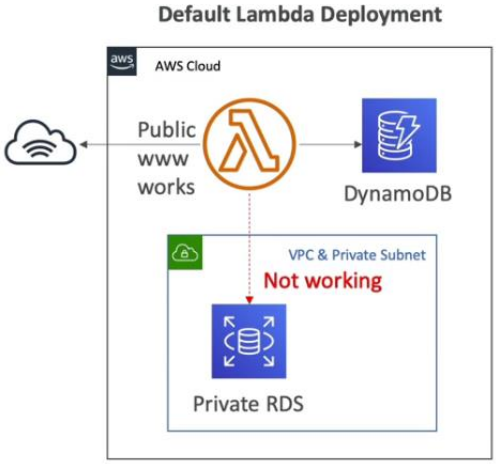
- A serverless compute service – allows to run code without provisioning or managing servers
  - Lambda automatically handles the underlying infrastructure
- Users can upload their code in the form of functions and execute them in response to events or triggers
- Can be written in various programming languages – Python, Java, C#, Ruby, Node.js and Go
- User will only be charged for the compute time consumed by the functions – no charge for idle time
- Scales automatically to handle the incoming request volume
- Integrates with other AWS services – build serverless applications and event-driven architecture
- Lambda functions can be monitored and troubleshooted – AWS CloudWatch
- Lambda Execution Role – IAM Role
  - Grants the Lambda function permissions to AWS services/resources
  - Some sample managed policies for Lambda:
    - *AWSLambdaBasicExecutionRole* – upload logs to CloudWatch logs
    - *AWSLambdaKinesisExecutionRole* – read from kinesis
    - *AWSLambdaDynamoDBExecutionRole* – read from DynamoDB streams
    - *AWSLambdaSQSQueueExecutionRole* – read from SQS
    - *AWSLambdaVPCAccessExecutionRole* – deploy Lambda functions in VPC
    - *AWSXRayDaemonWriteAccess* – upload trace data to X-Ray

- When event source mapping is used to invoke function – Lambda uses the execution role to read event data
- Best practice – create one Lambda Execution Role per Function

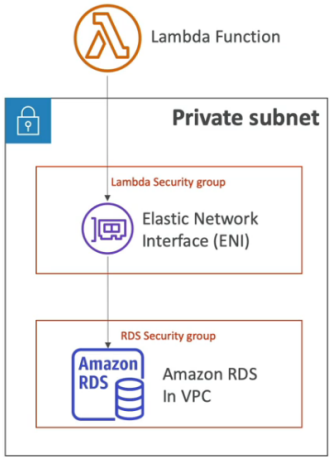
- Lambda Resource Based Policies – to give other accounts and AWS services permission to use Lambda resources
  - Similar to S3 bucket policies for S3 bucket
  - An IAM principal can access Lambda
    - If the IAM policy attached to the principal authorizes it – user access
    - Or if the resource-based policy authorizes it – service access
  - When an AWS service S3 calls Lambda function – the resource-based policy gives it access
  - Lambda permissions to write to CloudWatch logs – Lambda Execution Role



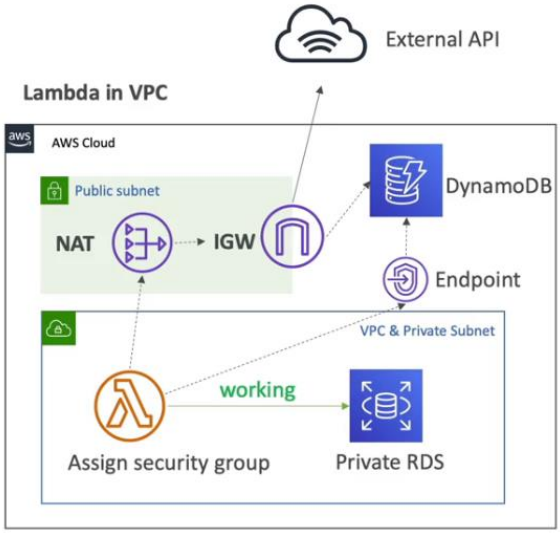
- By default, Lambda function is launched outside of user's own VPC – in an AWS-owned VPC
  - Therefore, it cannot access resources in user's VPC



- To make Lambda access resources within user's VPC
  - User must define the VPC ID, the Subnets, and Security Groups for Lambda to access
  - Lambda will create an ENI in user's subnet – behind the scenes
  - *AWSLambdaVPCAccessExecutionRole* is required



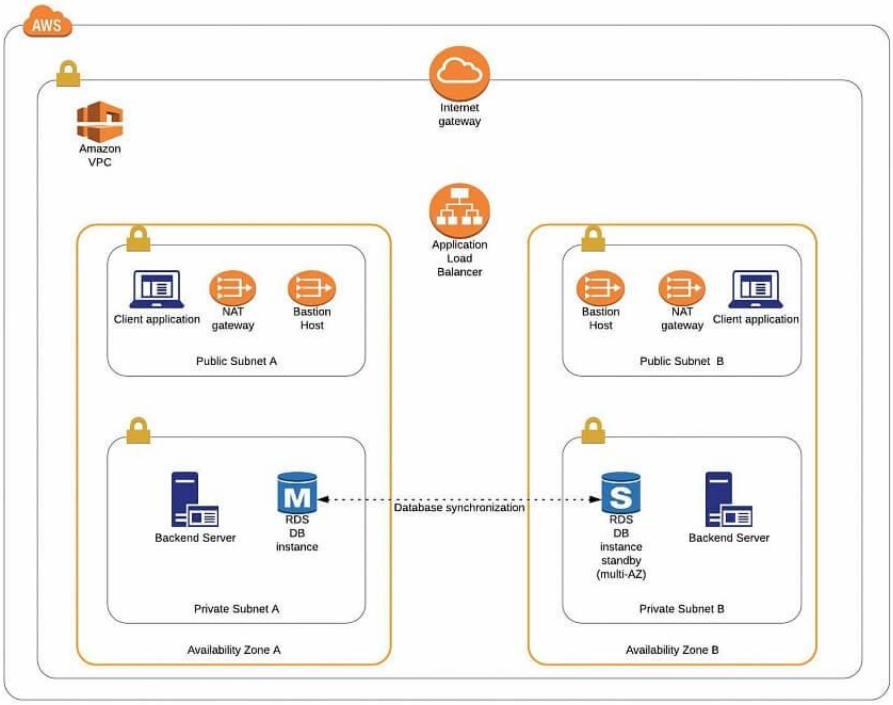
- A Lambda function in user's VPC does not have internet access
  - Deploying a Lambda function in a public subnet does not give it internet access or a public IP
  - Deploying a Lambda function in a private subnet gives it internet access if user has a NAT Gateway/Instance
  - User can use VPC endpoints to privately access AWS services without a NAT
  - Lambda – CloudWatch logs works even without endpoint or NAT Gateway



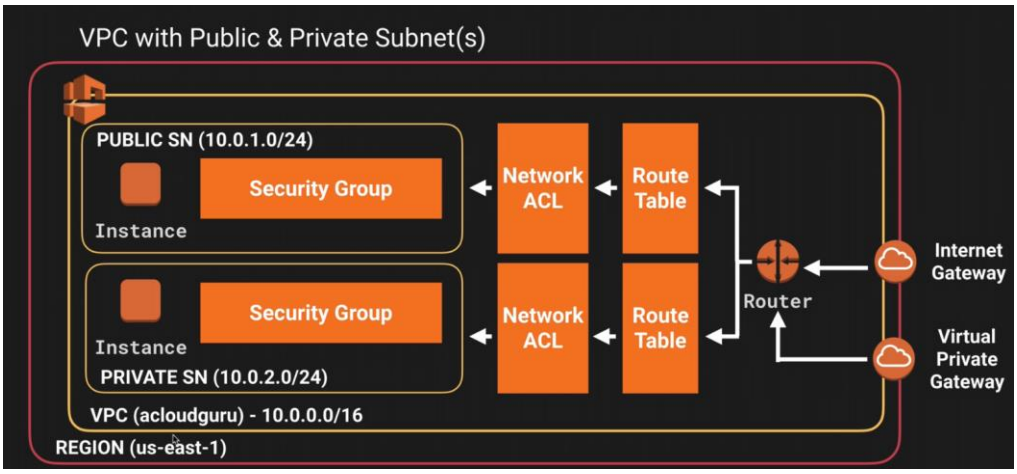
Amazon VPC:

- Amazon Virtual Private Cloud
- Virtual network service provided by AWS
- Allows to create a logically isolated section within AWS cloud, similar to traditional network in user's own data center

- It enables the launching of AWS resources, such as EC2 instances, databases, load balancers, in a private, secure, and customizable network environment
- Key features of AWS VPC include:
    - Subnets – VPCs can be divided into multiple subnets to segregate resources and control network traffic – 1 subnet = 1 AZ
    - Security – VPC provides built-in security features like NACLs (stateless) and security groups (stateful) to control inbound and outbound traffic
    - Routing – routing tables can be configured to control the flow of traffic between subnets and to the internet
    - Connectivity – VPC supports connectivity options like VPN and Direct Connect to connect the VPC with on-premises network
- It allows users to define IP address ranges for their resources, choose their own network topology, and configure network gateways for internet access
- It provides the ability to create and manage multiple VPCs, enabling the isolation of different environments such as development, staging, and production
- User can configure route tables between subnets
- User can create internet gateway and attach it to VPC



- User can easily customize the network configuration of Amazon VPC – for example, user can create a public-facing subnet for webservers that has access to the internet, and place backend systems such as databases or application servers in a private subnet with no internet access
- Additionally, user can create a Hardware VPN connection between corporate data center and VPC and leverage the AWS cloud as an extension of corporate data center

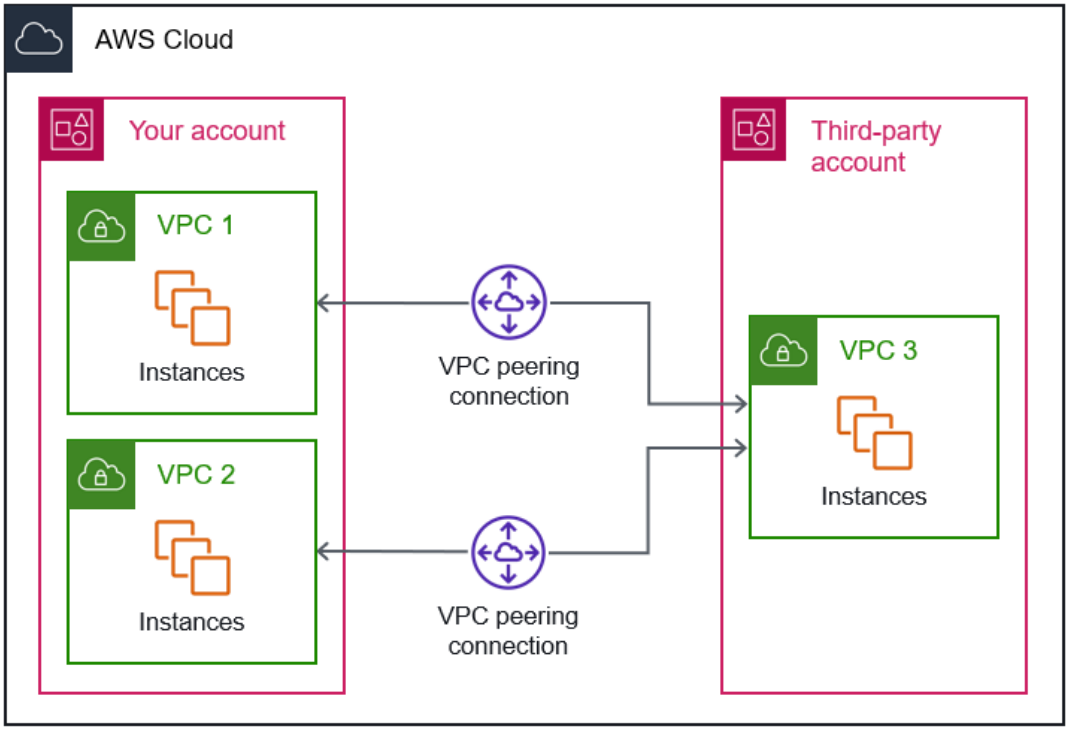


Default VPC:

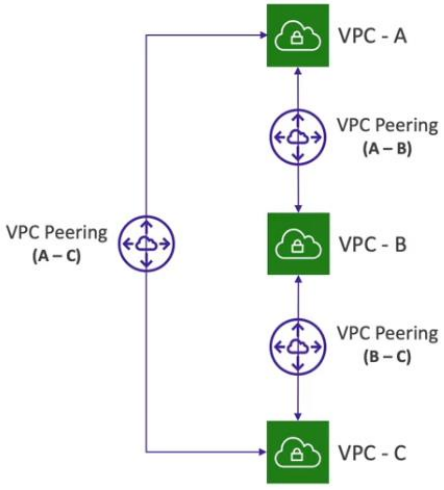
- A pre-configured VPC that is automatically created for every AWS account in each region
- It simplifies the process of launching resources by providing a ready-to-use network infrastructure
- The default VPC includes default subnets, route tables, security groups, and network ACLs
- It is designed to be convenient for beginners and quick prototyping, but it may not suit all use cases
- Instances launched in default VPC have automatic public IP assignment and can communicate with the internet by default
- The default VPC can be modified, and additional resources can be added or removed as needed
- User can choose to delete the default VPC if they no longer need it, but it is not possible to restore it once deleted

VPC Peering:

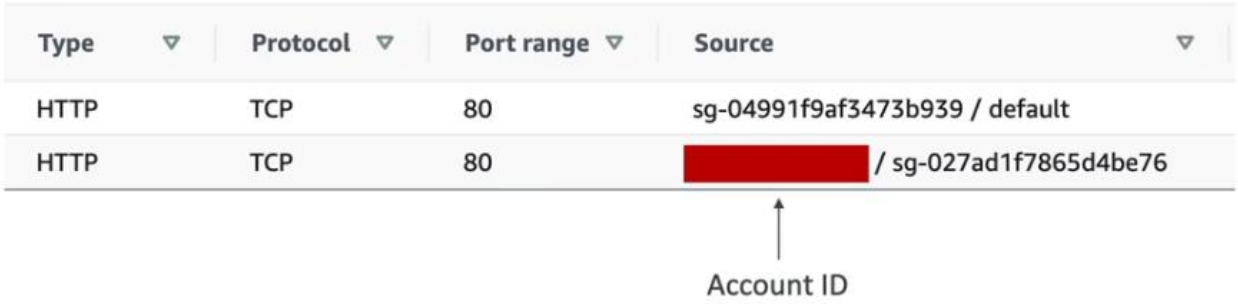
- VPC peering is a networking connection between two VPCs in the same AWS region or different AWS accounts
- It allows the VPCs to communicate with each other using private IP addresses as if they were on the same network
- Allows users to connect one VPC with another via a direct network route using private IP addresses – must not have overlapping CIDRs



- VPC peering operates at the network layer (layer 3) of the OSI model and can be used for various services like EC2 instances, RDS databases, and Lambda functions
- It supports routing updates – allowing the exchange of routing information between the peered VPCs
- Peering is in a start configuration – no transitive peering – like 1 VPC peers with 4 others

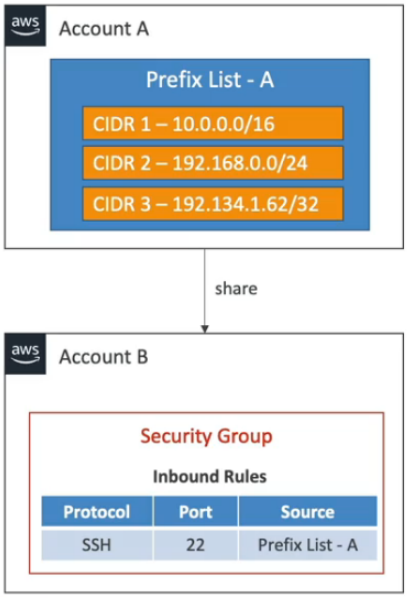


- Security group in a peered VPC can be referenced (work cross accounts – same region)



Security Groups:

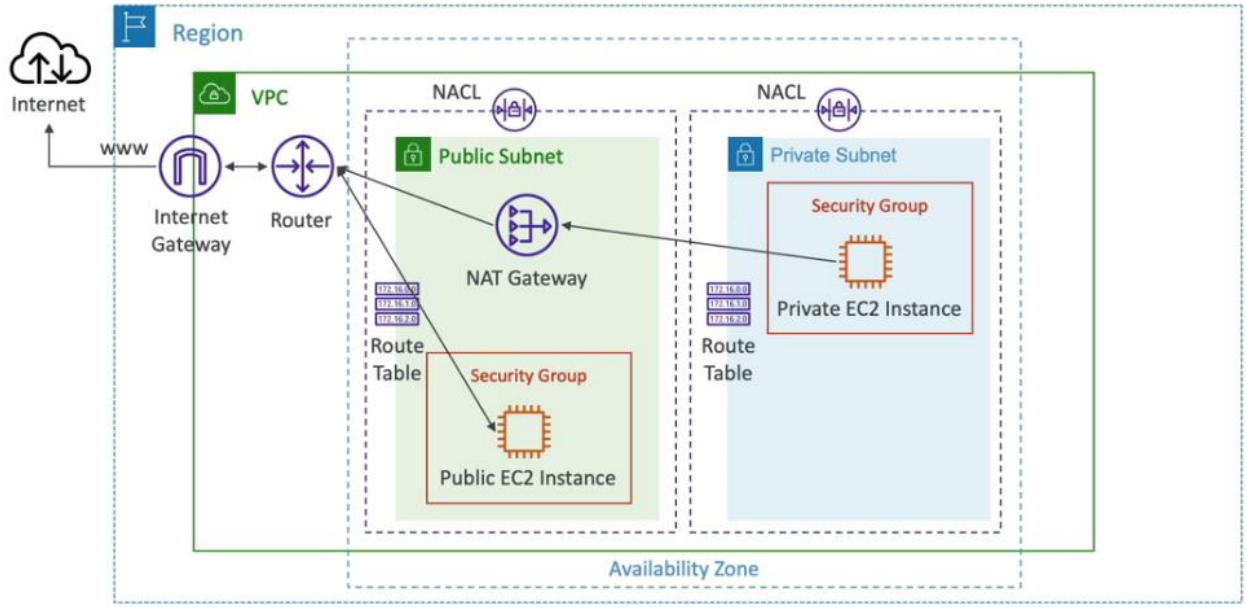
- AWS Security Groups are virtual firewalls that control inbound and outbound traffic from AWS resources
- They act as a network level access control for EC2 instances, RDS instances, and more
- Security Groups are stateful, allowing automatic inbound and outbound traffic based on rules
- Rules in Security Groups specify protocols, ports, IP ranges, and other attributes for traffic control
- Security Groups can refer other security groups for easy management and resource communication
- Multiple Security Groups can be associated with a single resource for granular traffic control
- Changes to Security Groups rules take effect immediately, without instance restart
- Security Groups can be assigned during resource creation and or modified later
- They work in tandem with NALCs that operate at subnet level
- They provide an additional layer of security by enforcing fine-grained traffic rules for resources within a VPC
- Regarding Outbound rules of Security Groups – default is allowed 0.0.0.0/0 anywhere
    - It can be removed, and specific prefixes can be allowed only
- Managed Prefix List – a set of one or more CIDR blocks
    - Makes it easier to configure and maintain Security Groups and Route Tables



- Customer Managed Prefix List – set of CIDRs that user defines and managed by user
  - Can be shared with other AWS accounts or AWS Organization
  - Modify to update many Security Groups at once
- AWS Managed Prefix List – set of CIDRs for AWS service
  - User can't create, modify, share, or delete them
  - S3, CloudFront, DynamoDB, Ground Station etc.

NACLs:

- Network Access Control Lists – virtual firewalls at the subnet level in AWS
- They control inbound and outbound traffic based on user-defined rules
- NACLs operate at the subnet level, filtering traffic entering and leaving the subnet
- Each subnet is associated with a single NACL, but multiple subnets can share the same NACL
- NACLs are evaluated in a specific order, following numbered rules (lowest to highest) until a match is found – higher precedence with a lower number
  - First rule match will drive the decision
  - The last rule is an asterisk * and denies a request in case of no rule match
  - AWS recommends adding rules by increment of 100
- Rules in NACL can allow or deny traffic based on IP addresses, protocols, ports, and other attributes
- NACLs are stateless, meaning that response traffic must be explicitly allowed through outbound rules



- They provide a coarse-grained control layer for network traffic, complementing security groups
- NACLs can be set up to enable or block specific types of traffic at subnet level
- Changes to NACL rules take effect immediately, without the need of instance restarts
- Provides an additional layer of security for subnets within a VPC
- NACL cannot be associated with multiple VPCs – each NACL is associated with only one VPC
- VPC comes with default NACL
    - default NACL allows all inbound and outbound traffic by default with the subnets associated with
    - Recommendations – Do not modify the default NACL, instead create custom NACL
- If a subnet is not explicitly associated with an NACL, the subnet is automatically associated with a default NACL

Default NACL for a VPC that supports IPv4

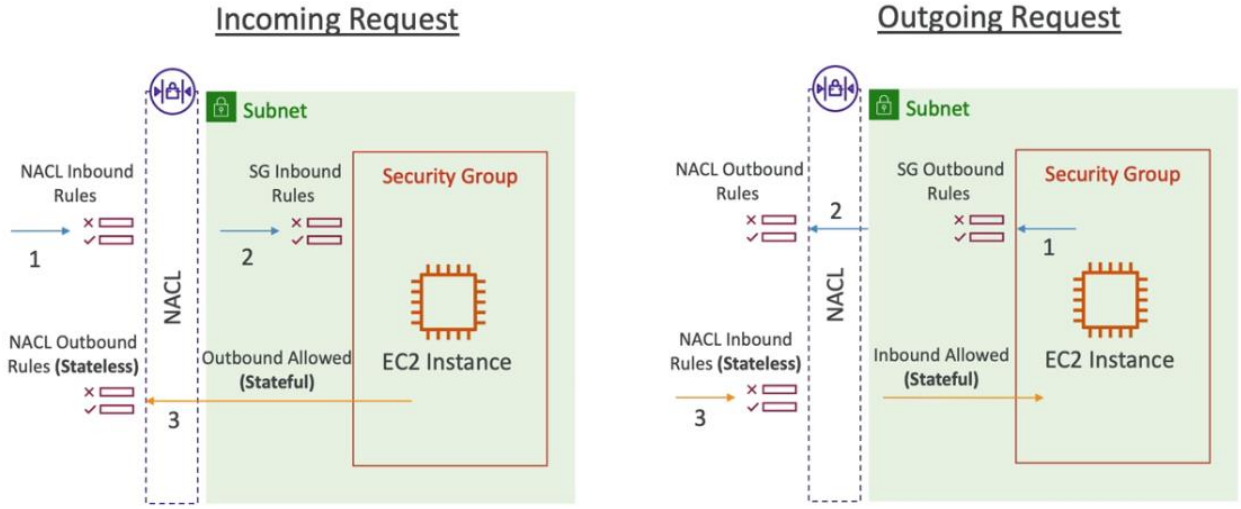
Inbound Rules

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

Outbound Rules

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

- Custom NACL denies all inbound and outbound traffic by default until rules are added



- Difference between Security Groups and NACLs can be explained as below:

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

Subnets:

- Subnets are subdivision of an AWS VPC's IP address range
- They allow for resource isolation and control within the VPC
- Each subnet is associated with a specific AZ within a region
- Subnets can have their own CIDR block, defining the IP address range for the subnet
- Instances launched within a subnet can communicate with other instances in the same subnet by default
- Subnets can be either public or private based on their route table configuration
- Public subnets have a route to an Internet Gateway, allowing instances to have direct internet access
- Private subnets do not have a route to the Internet Gateway, providing additional security by preventing direct internet access
- Communications between instances in private subnets and the internet can be facilitated through NAT Gateways or NAT instances
- Subnets can be associated with NACL for granular control over inbound and outbound traffic

- Subnets within VPC can span multiple AZs, providing high availability and fault tolerance
- Subnets allow for the segmentation of resource within a VPC based on their functional requirements

Route Tables:

- They control the routing of network traffic within a VPC and between the subnets
- Each subnet within a VPC is associated with a specific Route Table
- Route Tables contain a set of rules, known as routes, that determine where network traffic is directed
- Routes can be defined to route traffic within the VPC, to the internet, or to other connected networks
- The main/default Route Table is automatically created when the VPC is created
- Additional custom Route Tables can be created to tailor routing for specific subnets or use cases
- Each subnet must be associated with a Route Table, and a subnet can be associated with only one Route Table at a time.
- Route Tables can be edited to add, modify, or delete routes as needed.
- They can also be associated with network ACLs for fine-grained control over inbound and outbound traffic
- Route Tables are evaluated in order, and the most specific route (matching the destination IP) takes precedence
- They provide flexibility and control for directing network traffic within the VPC and enabling connectivity with external networks

Internet Gateway:

- An Internet Gateway is a horizontally scalable gateway in AWS VPC
- It enables communication between instances in the VPC and the internet
- The Internet Gateway acts as a bridge between the VPC and public internet
- Instances in public subnet can have direct internet access through the Internet Gateway
- Internet Gateway provides a public IP address to instances for outbound communications
- It performs NAT for outbound traffic from instances in the VPC
- Inbound traffic from the internet can be directed to instances in the VPC through appropriate security group rules
- An Internet Gateway is associated with a VPC and is used by default with the main/default route table
- Multiple subnets within a VPC can leverage the same Internet Gateway for internet connectivity
- An Internet Gateway is stateful, allowing response traffic to return to the instances seamlessly
- It does not impose any bandwidth constraints and can handle a large volume of network traffic

NAT Gateways:

- Network Address Translation Gateways
- It is a managed service provided by AWS to enable instances in private subnets to access the internet

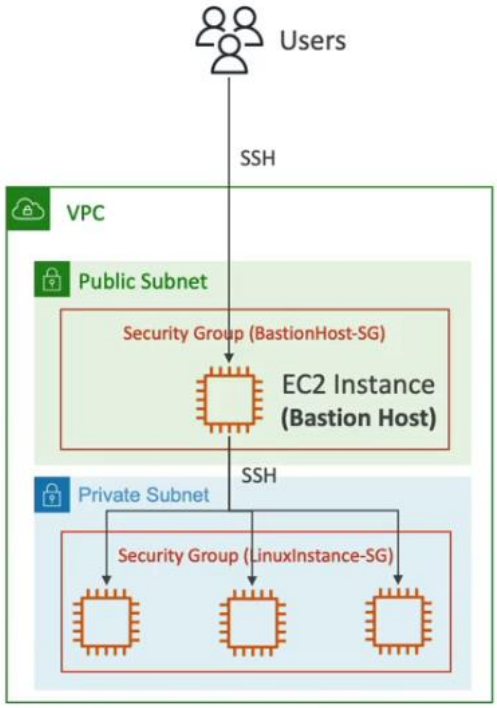
- NAT Gateway provides outbound internet connectivity for instances in private subnets while maintaining security
- It performs NAT to translate private IP addresses of instances to a public IP address when communicating with the internet
- Instances in private subnets use the NAT Gateway as a proxy to access resources on the internet
- NAT Gateway uses Elastic IP addresses for outbound communication – when NAT Gateway is created, it automatically assigns an EIP to it, which remains constant even if the NAT Gateway is replaced or restarted
- NAT Gateway can handle high throughput and is horizontally scalable to meet the demand of the workload
- It is highly available and redundant, automatically scaling to accommodate traffic
- It allows instances in private subnets to communicate with the internet, but inbound connections from the internet are not allowed
- NAT Gateway is stateful, allowing response traffic from the internet to reach instances in private subnets
- Inbound and outbound traffic must be explicitly allowed in the associated NACLs for the private subnets, as the NAT Gateway acts as a managed service and does not have its own associate NACL
- It simplifies the management and configuration of outbound internet access for private instances in the VPC
- NAT Gateway is a pay-as-you-go service, and its cost is based on the data processed and the duration of usage

NAT Instances:

- NAT instances act as a gateway between resources within private subnets and the internet
- They allow private instances to communicate with the internet while keeping them protected from direct inbound access
- NAT instances translate the private IP addresses of resources to the public IP addresses assigned to the NAT instance when communicating within the internet
- They can be launched from AMI provided by AWS or created from a custom AMI
- NAT instances require the configuration of appropriate routing rules and security group settings to enable communication between private subnets and NAT instance
- NAT Instance must be in public subnet
- When creating NAT Instance, Source/Destination Check on the instance needs to be disabled
- NAT Instance can use either an Elastic IP address or a public IP address associated with the instance – if public IP address is used, it may change if the instance is stopped or restarted
- Inbound and outbound traffic must be explicitly allowed in the associated NACLs for both the NAT instance's security group and the private subnets
- They can be deployed in high availability configurations across multiple AZs for redundancy
- NAT instances can impose some limitations in terms of network throughput and scalability compared to newer AWS services like NAT Gateways
- They can be monitored using Amazon CloudWatch for metrics such as data transfer, CPU utilization, and network packets

Bastion Hosts:

- A highly secured and hardened instance that provides controlled access to private resources within a VPC
- It acts as a gateway for SSH or RDP connections to user's private instances, allowing to securely administer them
- It is placed in a publicly accessible subnet and acts as a bridge between the internet and user's private instances
- It is typically designed with minimal software and services installed to reduce the attach surface



- Configured with strict security group rules to restrict inbound traffic to only authorized IP addresses or network
    - Bastion Host security group must allow inbound from the internet on 22 or RDP port from restricted CIDR – for example public CIDR of user's corporation
    - Security group of EC2 instances must allow the security group of the Bastion Host, or private IP of the Bastion Host
- Users connect to Bastion Host using SSH or RDP and then use Bastion Host as a proxy to establish connections to the private instances
- It can be configured with logging and monitoring to track user activities
- It can be deployed as a standalone instance or in a high availability configuration with redundancy and load balancing

NAT Instances VS NAT Gateways:

- NAT Gateway is a managed service provided by AWS, while NAT Instance is an EC2 instance configured as a NAT device

- NAT Gateway automatically scales up to 45Gbps, whereas NAT Instance's performance is limited to EC2 instance type
- NAT Gateway is highly available within the AZ, while NAT Instance requires manual configuration for high availability
- NAT Gateway does not require managing underlying EC2 instances, whereas NAT Instance needs configuration and maintenance of the EC2 instance
- NAT Gateway does not provide SSH access or control over the underlying infrastructure, whereas NAT Instance allows more control and access
- NAT Gateway does not require a separate security group, while NAT Instance requires appropriate security group configurations
- Setting up and configuring NAT Gateway is easier compared to configuring and managing the NAT Instance
- NAT Gateway supports VPC Flow Logs for monitoring and troubleshooting, while NAT Instance requires additional configuration for monitoring
- NAT Gateway uses EIP for outbound communication, while NAT Instance can use either EIP or public IP address associated with the instance
- NAT Gateway does not require explicit NACL configuration, while NAT Instance requires explicit NACL configuration for both the NAT Instance's security group and the private subnets

NAT Instance VS Bastion Instance:

- NAT Instance are used for performing Network Address Translation for instances in private subnets, providing outbound internet connectivity while Bastions, on the other hand, Bastion Instance acts as a jump hosts or dedicated instances for secure remote access to instance in private subnets
- NAT instance requires manual configuration and management, while Bastions enforce security measures such as authentication and access control
- NAT Instances are typically deployed in public subnets and require security groups and NACL for traffic control, while Bastions are deployed in public subnets for inbound access
- NAT Instances enable outbound internet access, while Bastions enable secure remote access to instances within the VPC
- NAT Instances can be used for multiple VPCs by configuring appropriate routing, while Bastions enhance security by acting as an entry point with limited access to private subnet instances
- NAT Instance is used to provide internet traffic to EC2 instances in private subnet, while Bastions is used to securely administer EC2 instances (using SSH or RDP) in private subnets

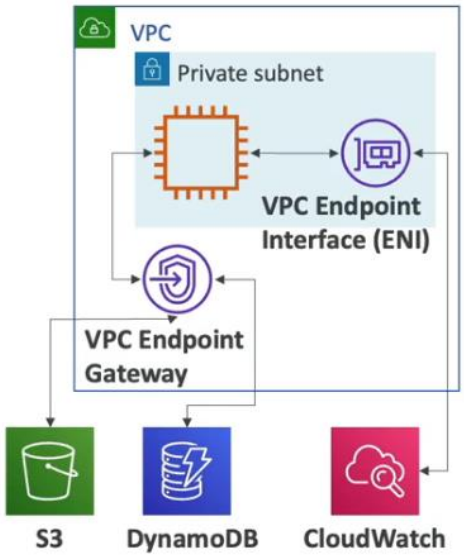
Virtual Private Gateway:

- It is a component of AWS VPC that enables secure communication between user's VPC and on-premises network
- It acts as a VPN concentrator on the AWS side of the VPN connection
- The VPG is associated with a specific VPC and is responsible for handling encrypted VPN traffic
- It provides an endpoint for establishing secure IPSec VPN connections over the public internet
- With a VPG, VPC can securely extend its network to the AWS Cloud and access resources within the VPC

- Both route-based and policy-based VPNs are supported by the VPGs
- Route-based VPNs use static routes to determine the network traffic flow between the VPC and on-premises network
- Policy-based VPNs use access control policies to determine which network traffic is allowed to pass through the VPN
- VPGs can be associated with customer gateways, which are the on-premises VPN devices or software endpoints
- The VPN connection between the VPG and customer gateway is encrypted using IPSec
- VPGs provide a secure and reliable way to connect to the VPC to the on-premises network, enabling hybrid cloud architectures

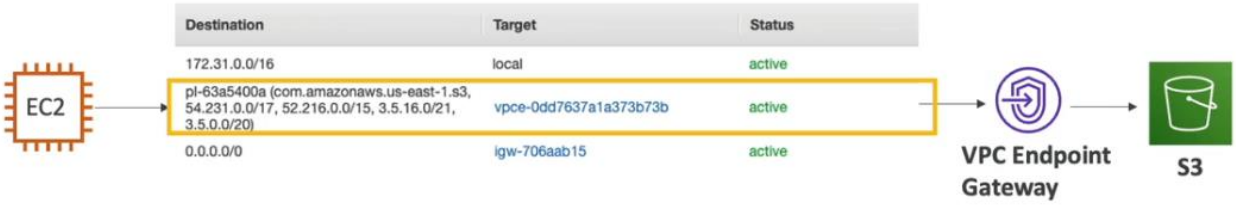
VPC Endpoints:

- VPC endpoints provide a secure and private connection between a VPC and supported AWS services or third-party services
- They allow access to these services without the need to traverse the public internet
- VPC endpoints are virtual devices that are horizontally scalable and highly available
- They facilitate communication between instances within the VPC and the endpoint service without requiring public IP addresses

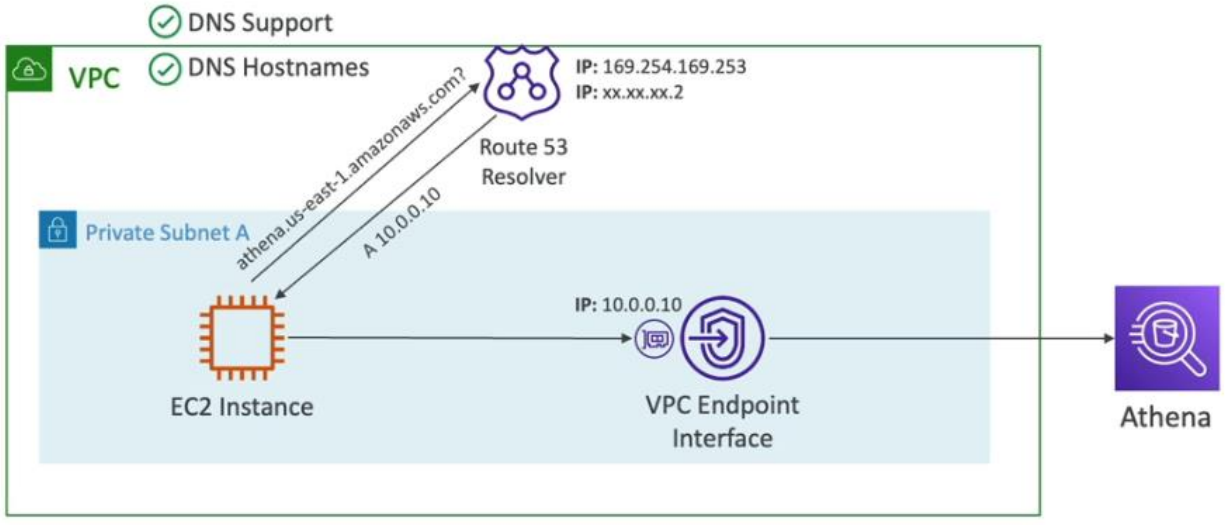


- They provide a highly available and scalable method to access services privately, avoiding exposure to the public internet
- Security is enhanced as traffic remains within the AWS network, eliminating the need for internet gateway or NAT devices
- Network performance is improved by reducing latency and avoiding bandwidth constraints associated with internet-based communication
- Configuring and managing VPC endpoints is straightforward through the AWS Management Console, CLI or API
- In case of issues – check DNS setting resolution in the VPC, check route tables

- Two types of VPC endpoints exist – Gateway endpoints and Interface endpoints
- Gateway endpoints – used for connecting to AWS services such as Amazon S3 and DynamoDB
    - Only works for S3 and DynamoDB – must create one gateway per VPC
    - Must update route table entries – no security groups
    - Gateway is defined at VPC level
    - DNS resolution must be enabled in the VPC

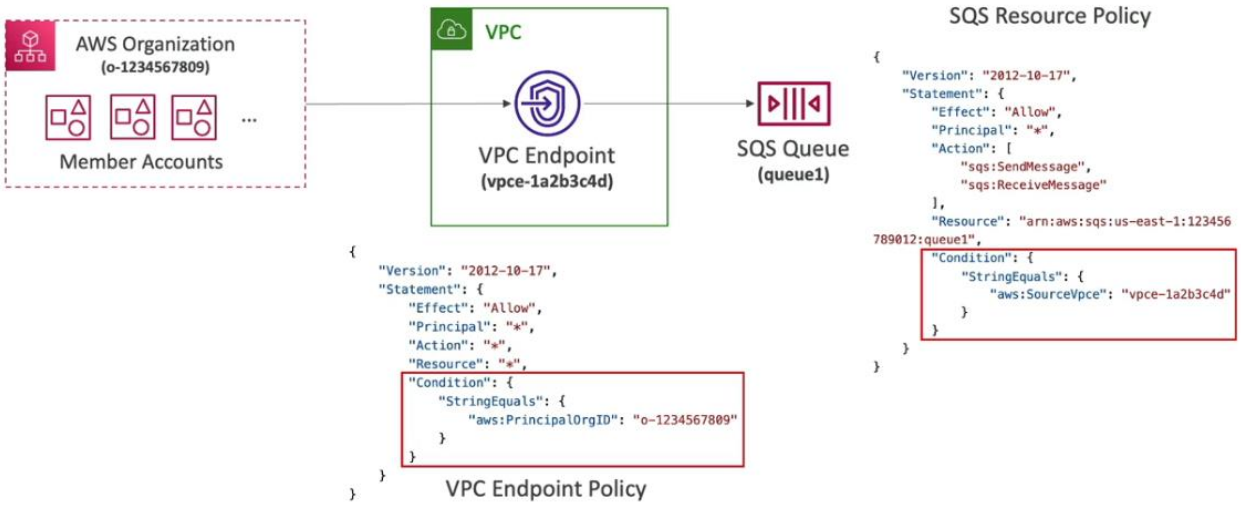


    - The same public hostname for S3 can be used
    - Gateway endpoint cannot be extended out of a VPC – VPN, DX, TGW, peering
- Interface endpoints - used for connecting to all other AWS services or third-party services except DynamoDB over the AWS network – known as AWS Private Link
    - Interface endpoints operate at the network interface level and have their own IP addresses within the VPC's subnet
    - Leverage Security Groups for security
    - Provision an ENI that will have a private endpoint interface hostname
    - Private DNS – setting when endpoint is created
        - The public hostname of a service will resolve to the private endpoint interface hostname
        - VPC setting 'Enable DNS hostname' and 'Enable DNS support' must be true
    - These interfaces can be accessed from Direct Connect and Site-to-Site VPN

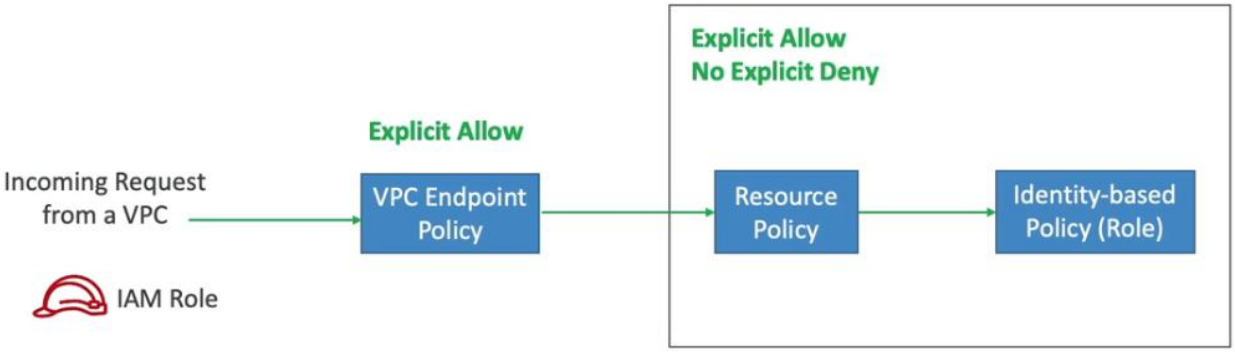


- VPC Endpoint Policy – controls which AWS principals (AWS accounts, IAM users, IAM roles) can use the VPC endpoint to access AWS services

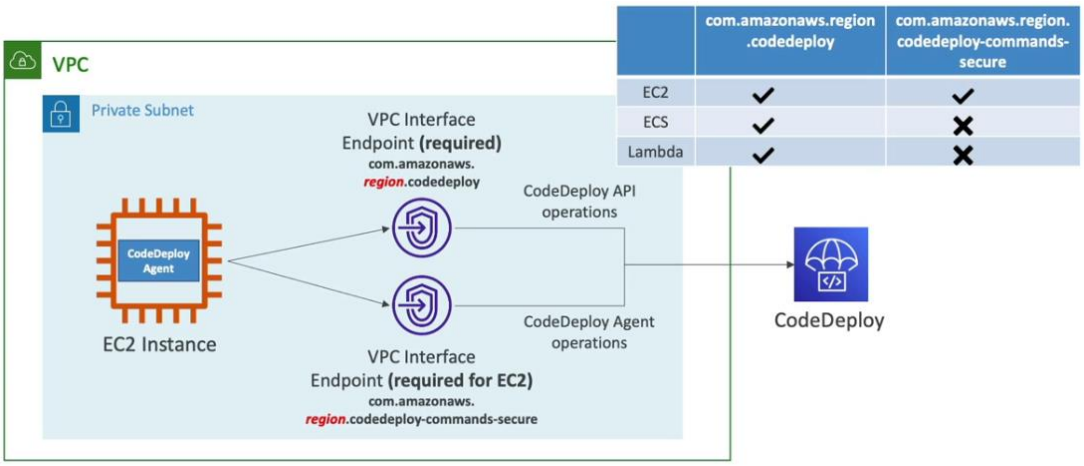
- Can be attached to both Interface Endpoint and Gateway Endpoint
- Can restrict specific API calls on specific resources
- Doesn't override or replace identity-based policies or service-specific policies – S3 bucket policies etc.
- Can use *aws:PrincipleOrgId* to restrict access only within the organization



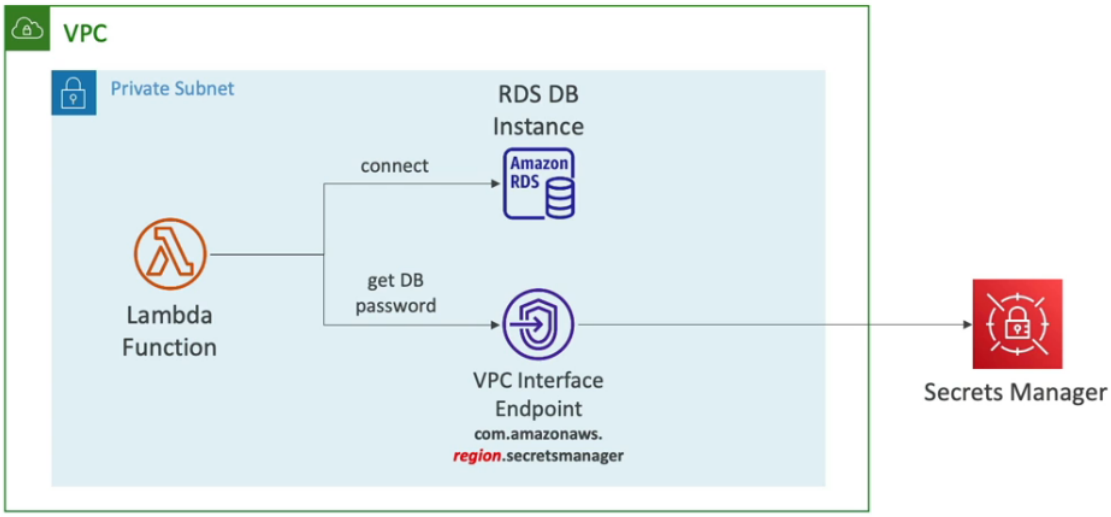
- VPC Endpoint Policy + Resource-based Policy + IAM Role – Authorization Logic



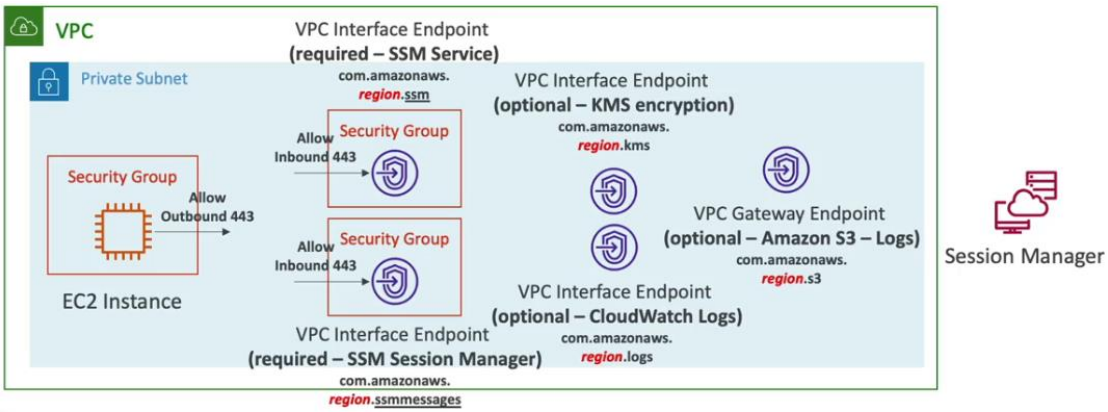
- Some examples of VPC Endpoint could be as below:
  - **CodeDeploy** – VPC Endpoint



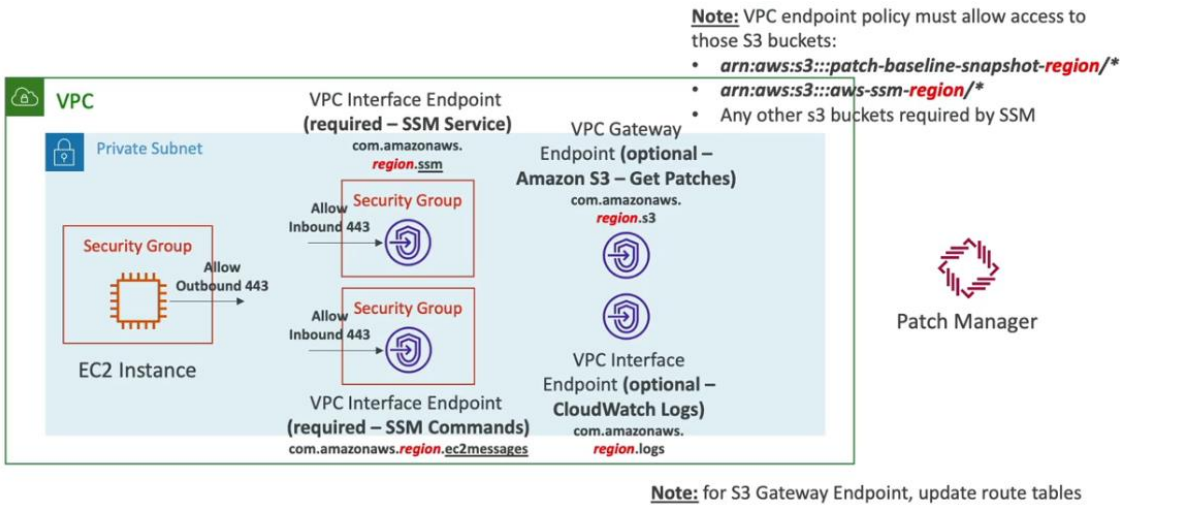
- **Secrets Manager** – VPC Endpoint



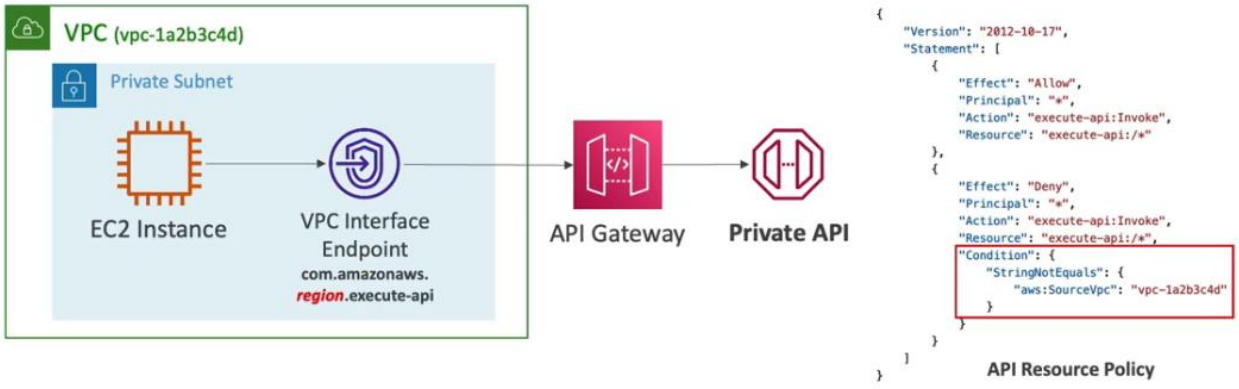
- **SSM Session Manager** - VPC Endpoint



- **Patch Manager** - VPC Endpoint



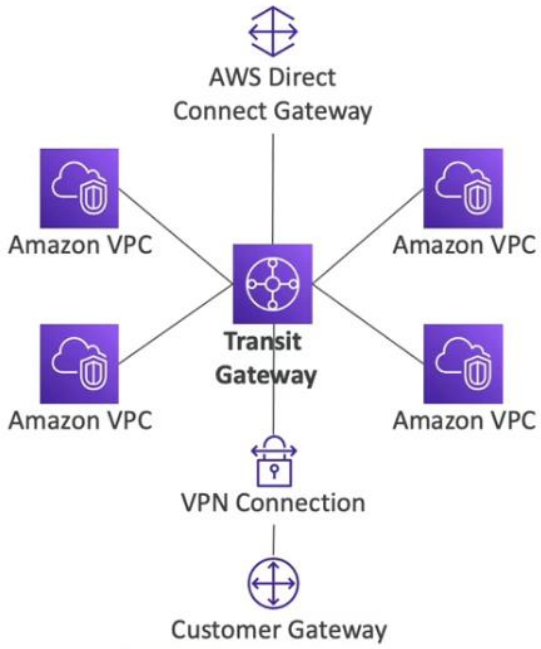
- **API Gateway** - VPC Endpoint
  - Private REST APIs can only be accessed using VPC Interface Endpoint
  - VPC Endpoint policies can be used together with API Gateway resource policies
  - Restrict access to private APIs from VPC and VPC Endpoints using resource policies



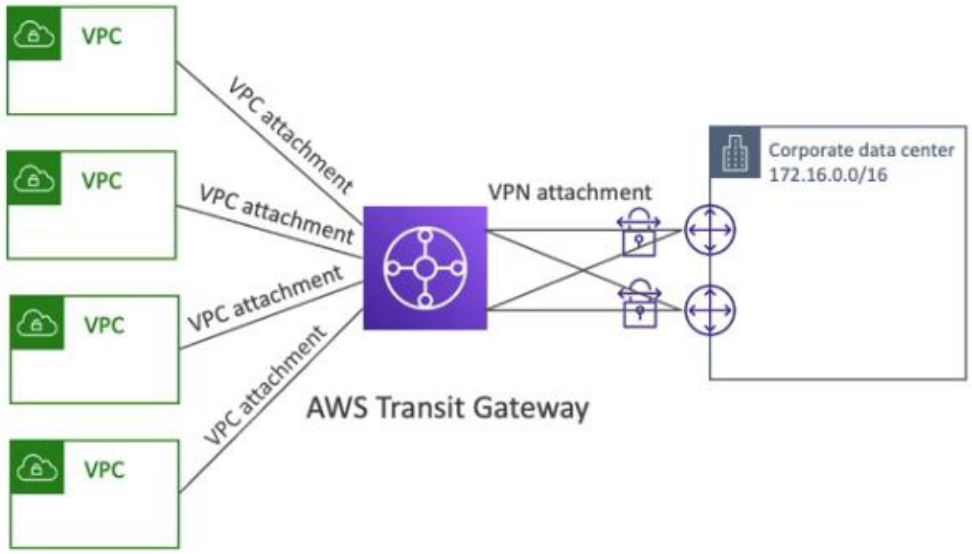
VPC Transit Gateway:

- VPC Transit Gateway is a scalable and centralized hub that enables connectivity between multiple VPCs and on-premises networks
- It acts as a transit hub, allowing communication between VPCs and on-premises networks without the need for individual direct connections
- VPCs and on-premises networks can attach to the Transit Gateway using VPN connections, AWS Direct Connect, or VPC peering
- Traffic between attached networks flow through the Transit Gateway, simplifying network management and reducing the number of connections
- VPC Transit Gateway supports the sharing of VPC resources, such as subnets and security groups, across connected networks
- It provides a highly available and scalable solution for interconnecting large numbers of VPCs and on-premises networks

- Hub-and-spoke (star) connections – transitive peering

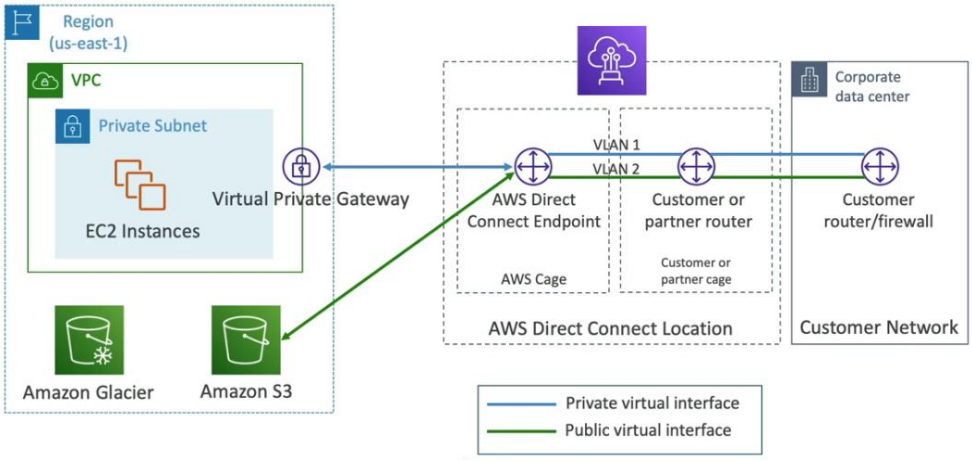


- Network routing between connected is managed through route tables associated with the Transit Gateway
- Any connected VPC is automatically available to every other connected network – Route tables control which VPCs can communicate
- Traffic between user's VPCs and the Transit Gateway is on the AWS network, inter-region traffic is encrypted
- It simplifies the network architecture, improves network performance, and reduces the overall complexity of managing the network connections
- Works with Direct Connect Gateway, VPN connections
- Regional resource – can work cross region
- Share cross-account using RAM (Resource Access Manager)
- The only service that supports the IP-Multicast
- Equal-Cost-Multiple-Path-Routing (ECMP) – routing strategy to allow to forward a packet over multiple best path
    - Use case: create multiple site-to-site VPN connections to increase the bandwidth

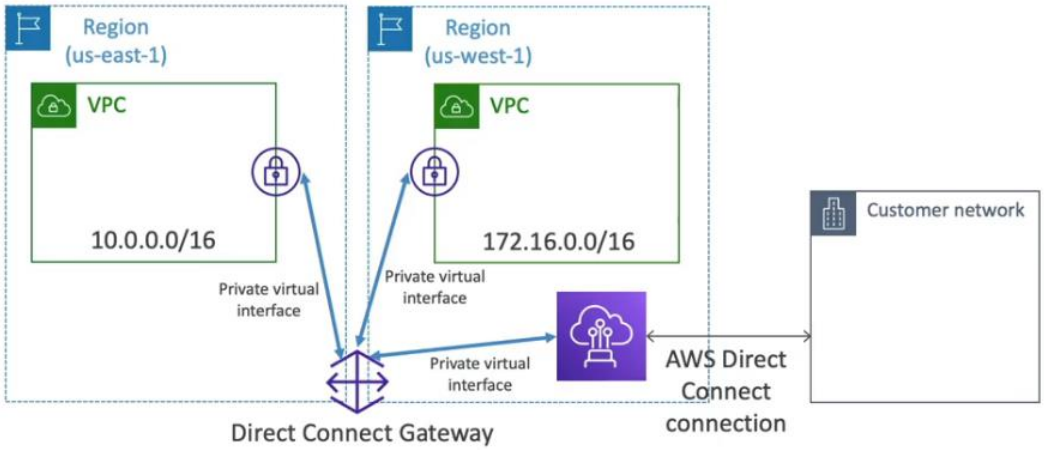


Direct Connect:

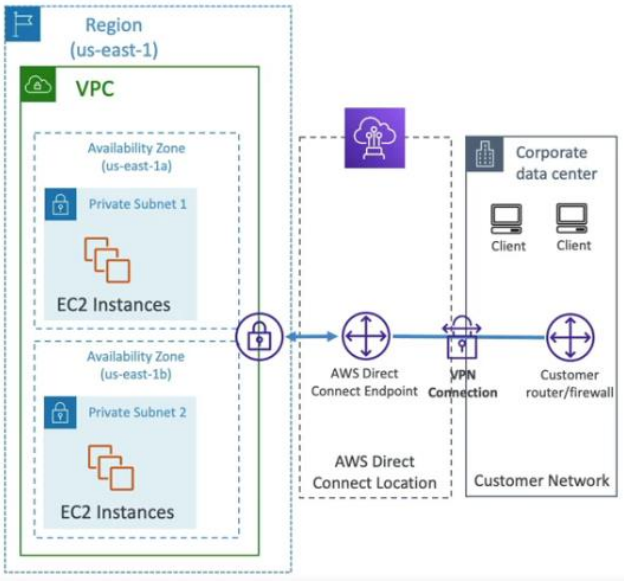
- AWS Direct Connect is a dedicated network connection service provided by AWS
- It establishes a private, high-bandwidth connection between an on-premises data center and AWS Direct Connect locations
    - Direct Connect connections can be established in AWS Direct Connect locations worldwide, including colocation facilities and data centers
- Direct Connect bypasses the public internet, offering more reliable and consistent network connection
- It provides a secure and low-latency connection for data transfer, suitable for large data transfers, real-time workloads, and sensitive applications
    - Direct Connect offers multiple connection speeds ranging from 1Gbps to 100Gbps
- It can be used to access services within a VPC, integrate on-premises infrastructure with AWS, or establish a hybrid cloud environment
- Users need to setup a Virtual Private Gateway on their VPC
- Users can access public resources (S3) and private resources (EC2) on same connection



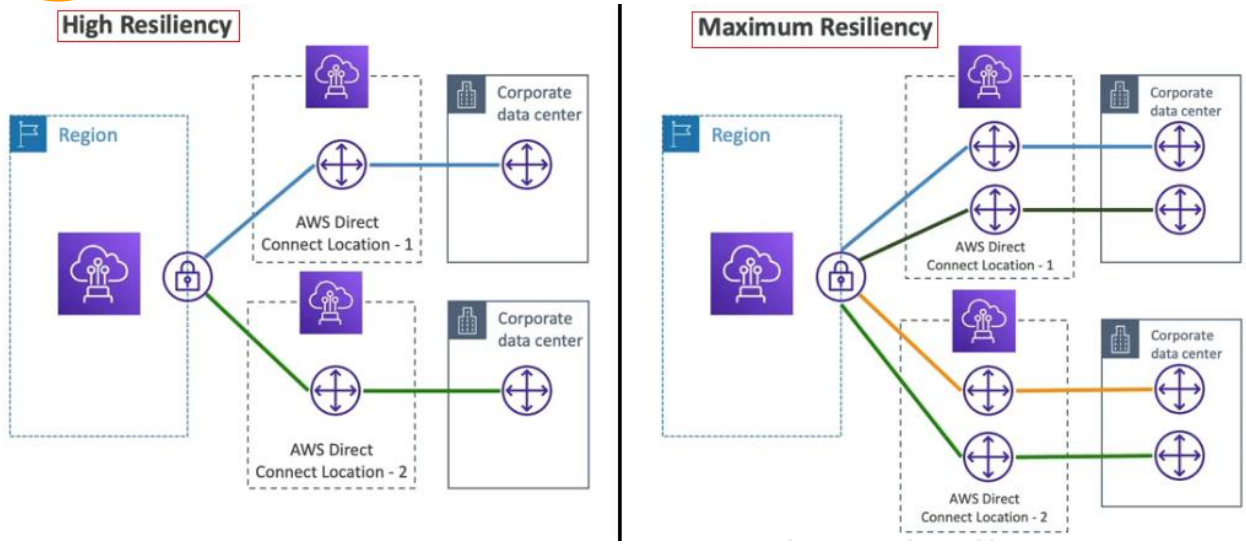
- Support both – IPv4 and IPv6
- Direct Connect Gateway – in case of setting up Direct connect to one or more VPCs in many different regions (same account)



- Connection types
  - Dedicated Connections – physical ethernet port dedicated to a customer
  - Hosted Connections – capacity can be added or removed on demand
- Direct Connect – Encryption
  - Data in transit is not encrypted but private
  - AWS Direct Connect + VPN – provides an IPSec-encrypted private connection

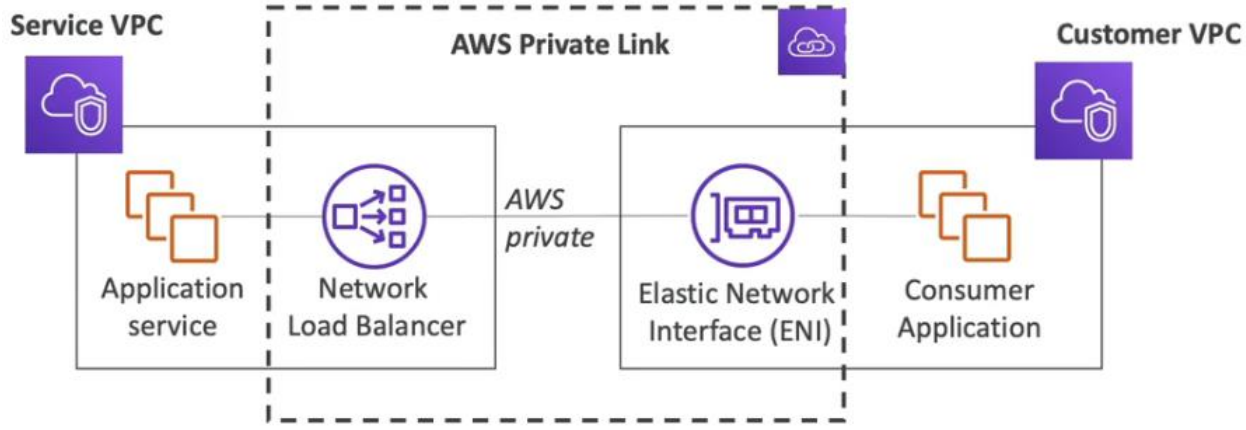


- Direct Connect – Resiliency
  - High resiliency for Critical workloads – one connection at multiple locations
  - Maximum resiliency for Critical workloads – separate connections on separate devices in more than one location



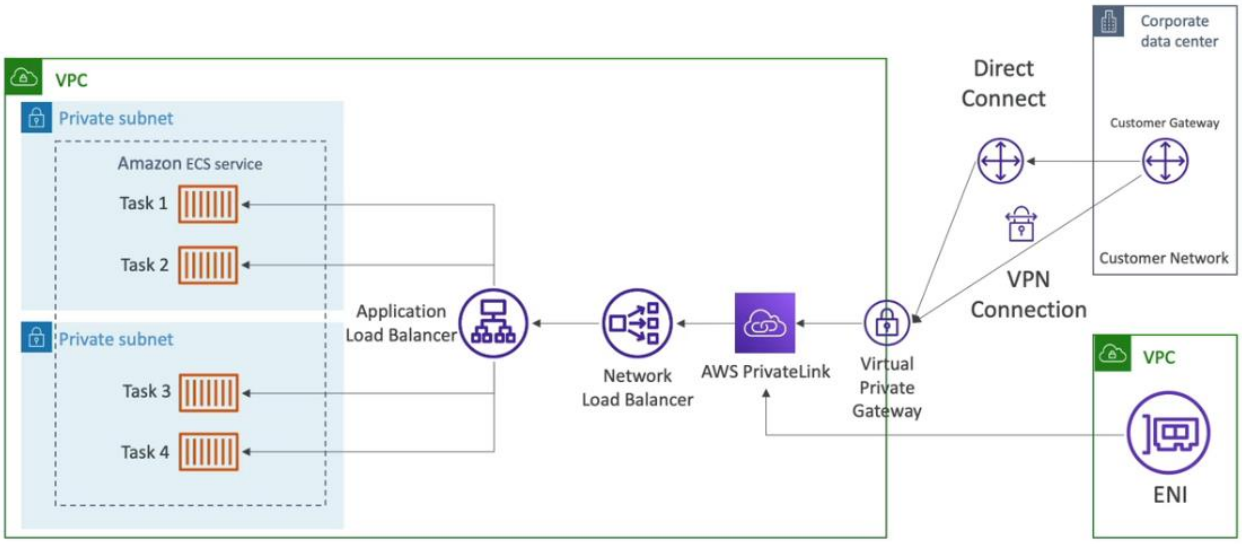
AWS Private Link:

- AWS Private Link enables private connectivity between VPCs and AWS services or supported third-party services
- Most secure & scalable way to expose a service to thousands of VPC (own or other accounts)
- Requires Elastic ENIs, NLBs, GWLB, and PrivateLink endpoints to establish private connections over the AWS network
- If the NLB is in multiple AZs, and the ENIs in multiple AZs, the solution is fault-tolerant



- Does not require VPC peering, internet gateway, NAT, route tables etc.
- PrivateLink endpoints are created in the VPC and associated with the desired service
- Each PrivateLink endpoint is assigned a private IP address within the VPC's subnet
- PrivateLink provides enhanced security by keeping traffic within the AWS network, reducing exposure to external threats
- It supports communication with various AWS services, including AWS S3, DynamoDB, and AWS Lambda, as well as select third party services
- Connectivity to PrivateLink endpoints can be controlled using security groups and VPC endpoint policies

- PrivateLink supports both TCP and UDP traffic for the services it enables connectivity to
- It simplifies network architecture by eliminating the need for public IP addresses, NAT gateways, or VPN connections
- Configuration and management of PrivateLink endpoints can be done using the AWS Management Console, CLI or SDKs
- AWS PrivateLink & ECS – architecture example

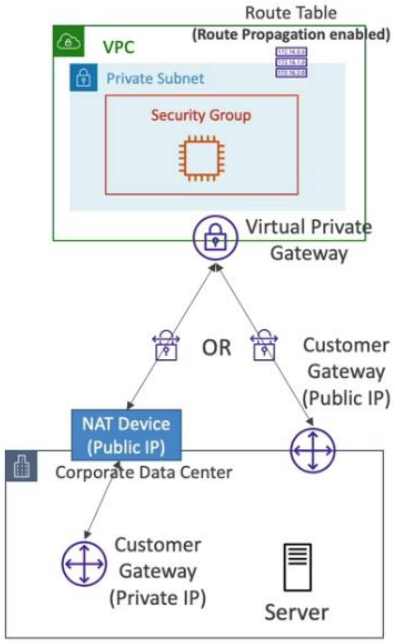


AWS VPN:

- Supports two types of VPN connections – Site-to-Site VPN and AWS Client VPN
- AWS Site-to-Site VPN – establishes encrypted IPsec tunnels between the on-premises network and the VPC
    - Encryption and authentication are provided using IPsec VPN protocols
    - Secure communication is enabled over the public internet
    - Various VPN devices and software are supported for compatibility
    - Multiple VPN connections can be configured for redundancy
    - AWS site-to-site VPN is used for establishing secure connections between on-premises networks and VPCs, while AWS Client VPN is used for secure remote access to VPN resources VPC resources for individual users
    - Site-to-site VPN connects networks, whereas Client VPN connects individual user devices
    - IP address ranges, routing, and security policies can be defined
    - Customer gateway devices and virtual private gateways are used for connection establishment
        - Virtual Private Gateway
            - VPN concentrator on the AWS side of the VPN connection
            - Virtual Gateway is created and attached to the VPC from which user wants to create the site-to-site VPN connection
            - Possible to customize ASN

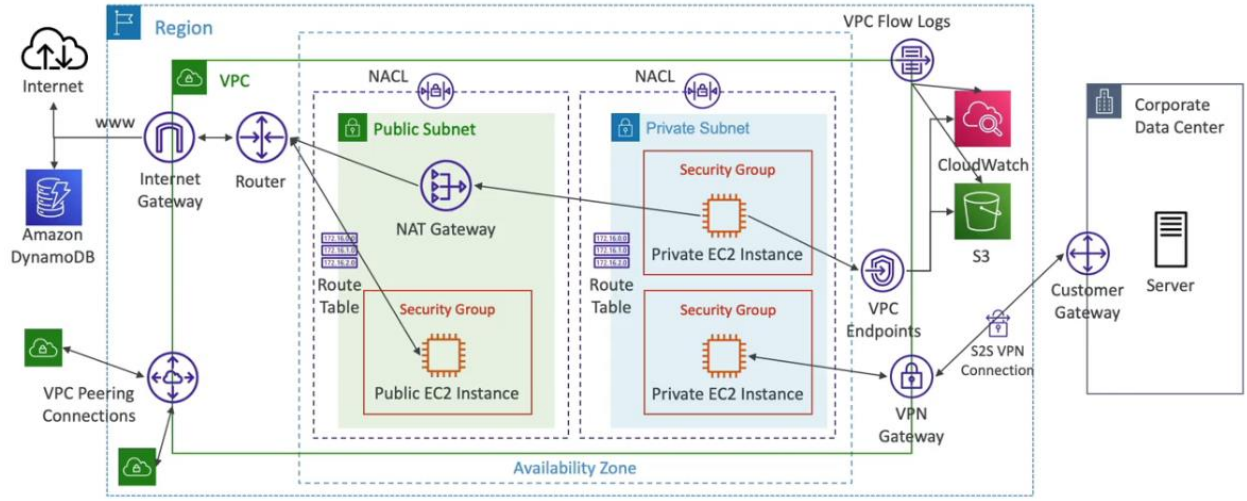
- Enable Route Propagation for the VPG in route table that is associated with the subnets

- Customer Gateway
  - Software application or physical device on customer side of the VPN connection
  - What IP to use?
    - Public Internet-routable IP address of user's CGW device
    - If it's behind the NAT device that's enabled for NAT traversal, use the public IP address of the NAT device

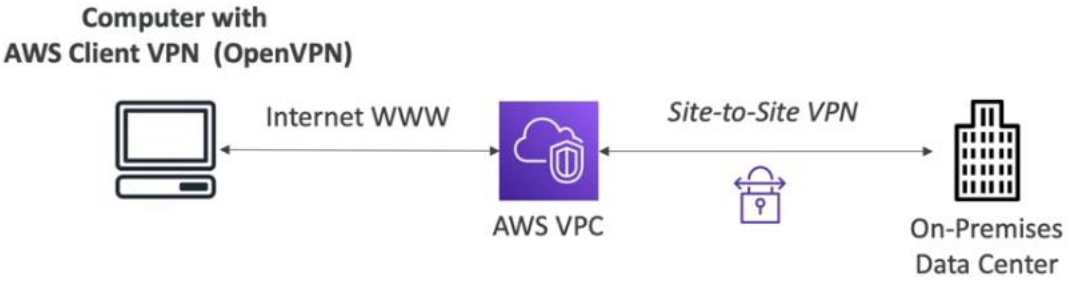


  - If users want to ping their EC2 instances from on-premises, they have to add ICMP protocol on the inbound of their security groups

- Site-to-site VPN is suitable for hybrid cloud environments or extending on-premises network
- Monitoring and management are done through AWS Management Console, CLI or API

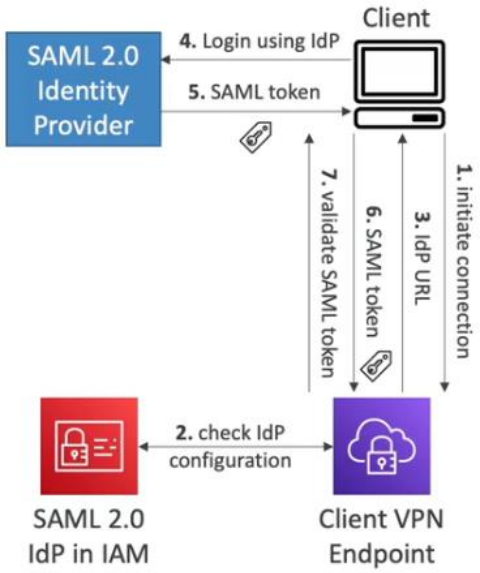


- AWS Client VPN – provides secure remote access to VPC resources
  - Users can connect securely from anywhere using standard VPN clients
  - Encrypted tunnels are established between user devices and the VPC
  - IPsec VPN protocols are used for encryption and authentication
  - Access is controlled through user authentication and authorization
  - Compatible with a wide range of VPN clients and devices
  - Flexible configuration options for IP address ranges and routing
  - Goes over public internet
  - Allow users to connect to their EC2 instances over a private IP (just if they were in the private VPC network)



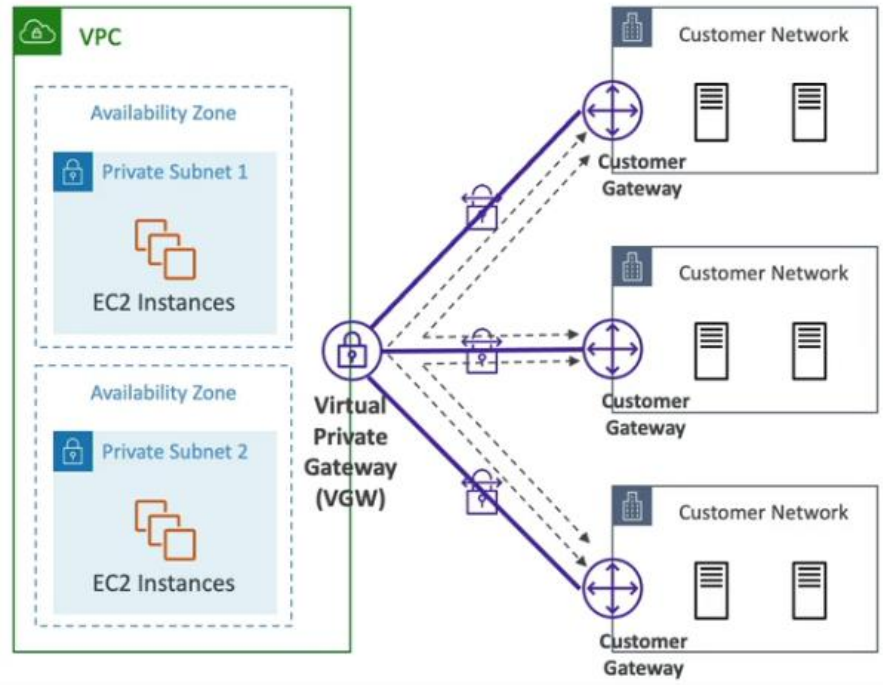
  - Monitoring and management can be done through AWS Management Console, CLI, or API
  - Client VPN – Authentication Types:
    - Active Directory Authentication – authenticate against Microsoft Active Directory (user-based)
      - AWS Managed Microsoft AD or on-premises AD through AD connector
      - Supports MFA
    - Mutual Authentication – use certificates to perform the authentication (certificate-based)
      - Must upload the server certificate to AWS Certificate Manager
      - One client certificate for each user (recommended)

- Single Sign-On – supports IAM identity center /AWS SSO
  - Authenticate against SAML 2.0-based identity provider (user-based)
  - Establish trust relationship between AWS and the identity provider
  - Only one identity provider at a time

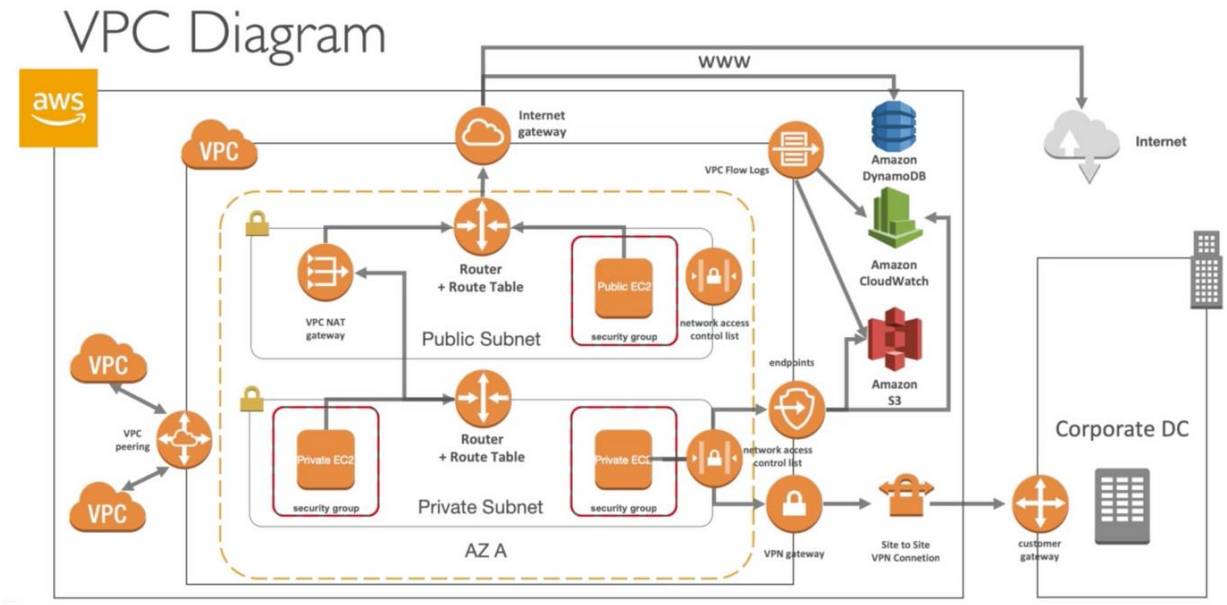


AWS VPN CloudHub:

- Provides secure communication between multiple sites – for multiple VPN connections
- Low-cost hub-and-spoke model for primary or secondary network connectivity between different locations (VPN only)



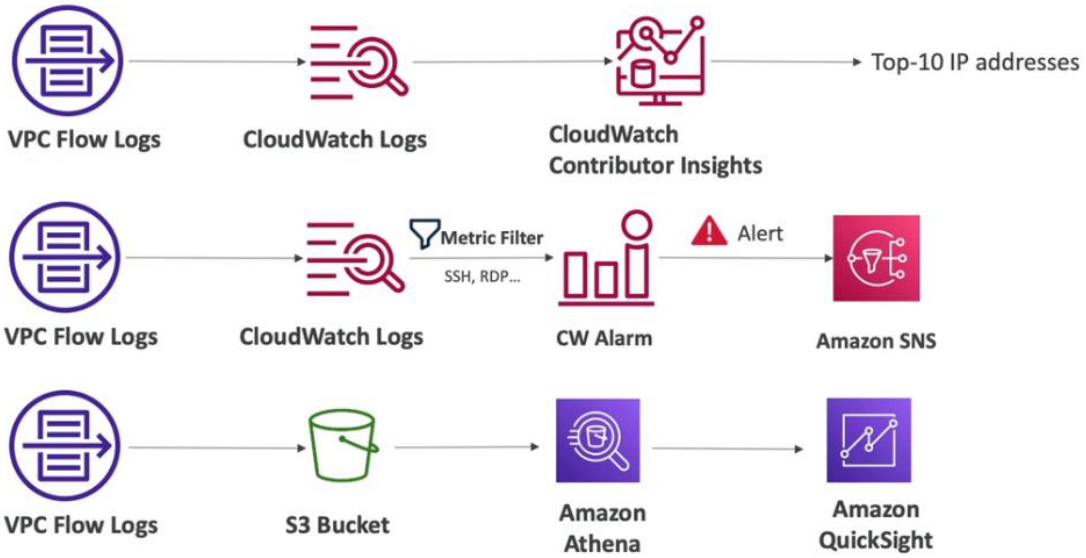
- It's a VPN connection so it goes over the public internet
- To set it up, connect multiple VPN connections on the same VGW, setup dynamic routing and configure route table – supports Border Gateway protocol
- It simplifies the network connectivity by allowing users to establish a single VPN connection from hub VPC to the on-premises data centers or branch offices
- The hub VPC acts as a central point of connectivity, while spoke VPCs represent the on-premises networks or branch offices
- Traffic between the hub and spoke VPCs is isolated and does not traverse other spoke VPCs, ensuring secure and private communication



VPC Flow Logs:

- A feature that captures information about the IP traffic flowing in and out of a VPC
- It provides detailed visibility into network traffic patterns and helps with monitoring, troubleshooting, and security analysis
- Flow Logs capture information such as source and destination IP addresses, ports, protocols, packet counts, action (success or failure) and more
- Flow Log can be enabled at the VPC level, subnet level, or network interface level
- They can be stored in Amazon CloudWatch Logs or Amazon S3 for analysis and retention – Athena on S3 or CloudWatch Logs Insight
- Flow Log can be used to analyze network traffic, detect unusual activity, and investigate security incidents
- They can help in compliance audits and meeting regulatory requirements
- Flow Logs do not capture payload data or decrypt encrypted traffic

- They can be used in conjunction with other AWS services like AWS CloudTrail and AWS Config for enhanced visibility and security
- Flow Logs can be created, managed, and monitored using AWS Management Console, CLI, or API
- Flow Logs cannot be enabled for VPCs that are peered within the VPC unless the peered VPC is in the same AWS account
- Flow Logs cannot be directly tagged
- The configuration of a Flow Log cannot be modified once it is created
- The traffic that is not captured by VPC Flow Logs:
    - Traffic to and from IP address 169.254.169.254, which is reserved for Amazon EC2 instance metadata service
    - Traffic to and from IP address 169.254.169.123 for Amazon Time Sync service
    - Traffic to the reversed IP addresses (10.0.0.1) of the default VPC router, used for internal VPC routing
    - Traffic for Windows Instance license activation, including communication with Microsoft activation servers
    - DHCP traffic
    - Traffic to Amazon DNS server – custom DNS server traffic is logged
    - Traffic between VPC Endpoint ENI and Network Load Balancer ENI
- It does not provide packet inspection capabilities
- Some of VPC flow logs architecture examples are below:

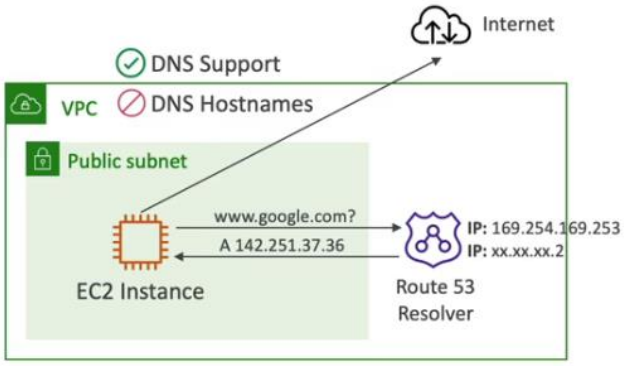


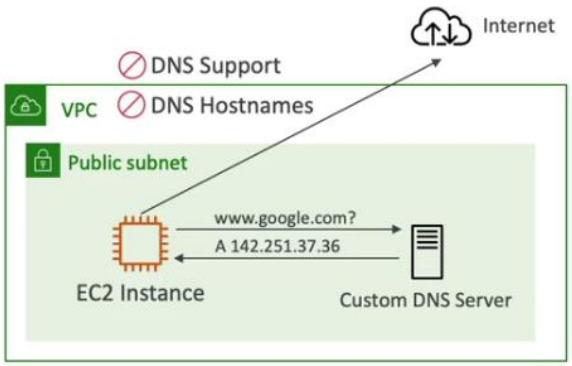
Amazon DNS:

- Amazon DNS server within a VPC provides DNS resolution and hostname resolution services for resources within the VPC
- When user creates a VPC, new VPC automatically includes an Amazon provided DNS server which is used to resolve public DNS hostnames
- It automatically assigns DNS names to EC2 instances launched in the VPC, making them accessible using the assigned DNS name
- Amazon DNS resolves private DNS names for resources within the VPC without the need for public IP addresses
- Is supports both forward and reverse DNS resolution for internal resources
- Amazon DNS server uses DHCP options sets to configure DNS setting for VPCs
- DNS resolution requests within the VPC are handled by the local Amazon DNS server, reducing latency, and improving performance
- Amazon DNS server supports both IPv4 and IPv6 addressing
- It provides DNS caching to improve resolution speed and reduce the load on DNS servers
- Custom DNS domain names can be associated with Amazon DNS server within the VPC using DNS resolution rules
- Amazon DNS server integrates with other AWS services, allowing seamless resolution of DNS names for resource such as load balancers and RDS instances within the VPC
- It is used for DNS hostname resolution for instances in user's VPC which are communicating over the internet
- The Amazon DNS server uses one of the reserved IP addresses in user's VPC CIDR range:
    - In a subnet CIDR block of 10.0.0.0/16
        - 10.0.0.0 is the network address
        - 10.0.0.1 is user's VPC router
        - 10.0.0.2 is the DNS server
        - 10.0.0.3 is reserved for future use
        - 10.0.0.255 is usually the network broadcast address but as broadcast is not supported in AWS that is also reserved
- If user doesn't want to use the Amazon provided DNS server and instead user wants to use a custom DNS server, user can disable this in the setting of VPC

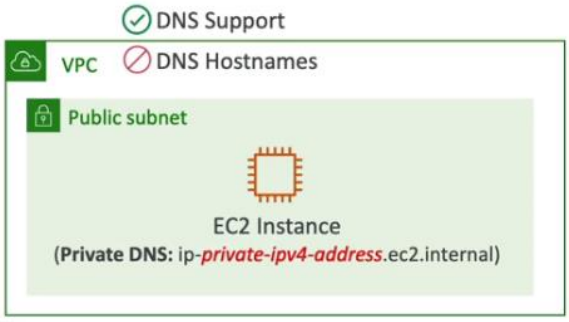
DNS Resolution in VPC:

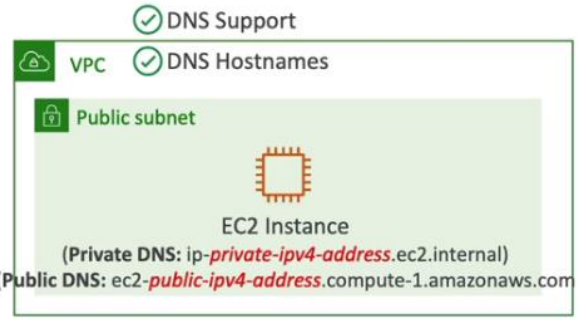
- DNS Resolution (enableDnsSupport) – decides if DNS resolution from Route53 resolver server is supported for VPC
    - True by default – it queries the Amazon provider DNS server at 169.254.169.253 or the reserved IP address at the base of the VPC IPv4 network range plus two (.2)



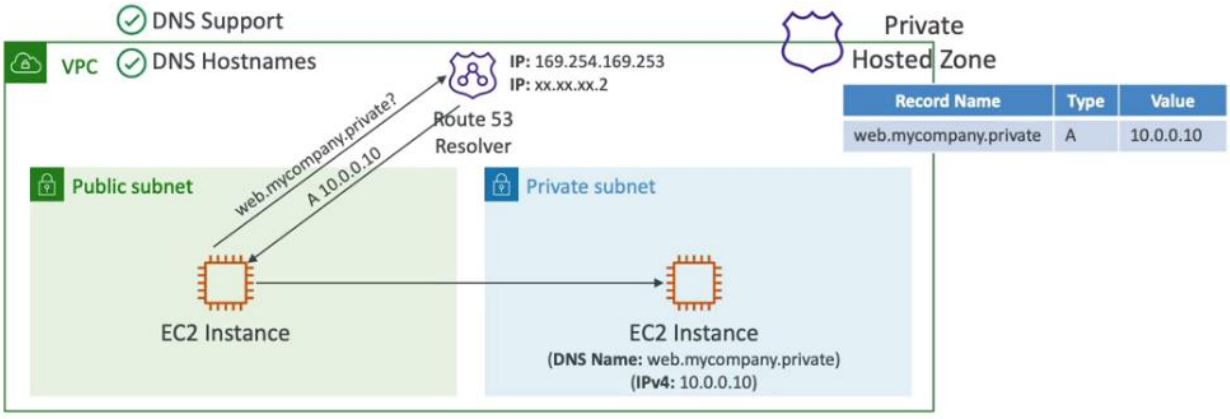


- DNS Hostnames (enableDnsHostnames)
  - Be default, true – default VPC
  - By default, false – newly created VPCs
  - Won't do anything unless enableDnsSupport is true
  - If true, it assigns public hostname to EC2 instance if it has a public IPv4





- If we use custom DNS domain names in a private hosted zone in Route53, we must set both these attributes (enableDnsSupport and enableDnsHostnames) to true – can be explained via below diagram

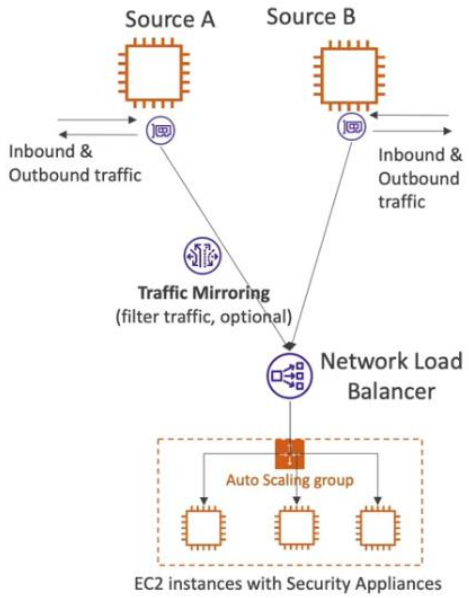


VPC Traffic Mirroring:

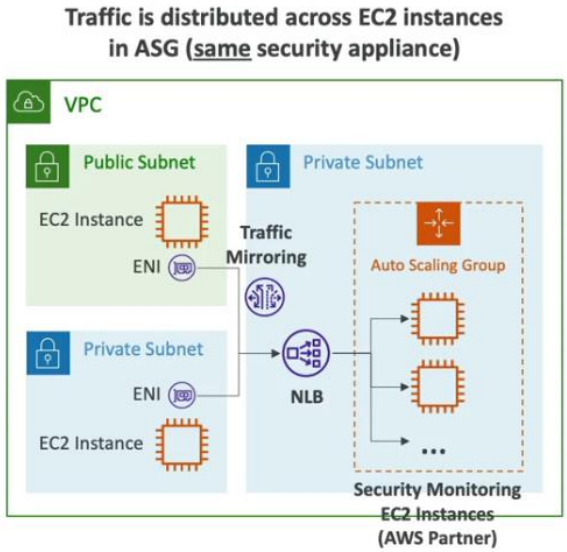
- Allows to capture and inspect network traffic within a VPC
  - Captures from – ENI (Source)

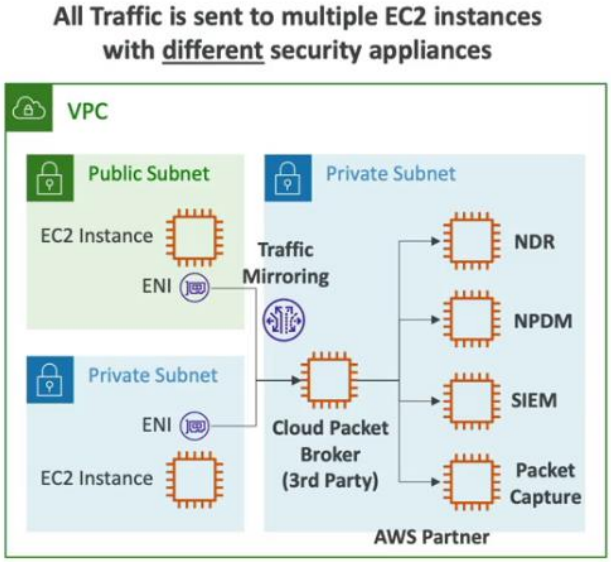
- To – an ENI or NLB (Target)
- Source and Target can be in the same VPC or different VPCs (VPC Peering)

- It enables to mirror traffic from ENIs of EC2 instances, NLBs, and ENIs of NAT Gateway to a designated destination
- Use case – analyze network traffic for inspection, threat monitoring, anomalies detection, troubleshooting, forensic and security purposes/compliance requirements
- Supports both inbound and outbound traffic mirroring
- Selective filters can be configured to capture specific traffic based on protocols, ports, IP addresses, or other criteria

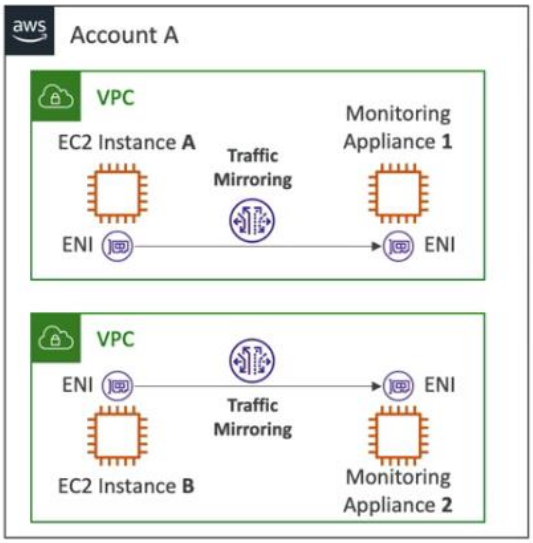


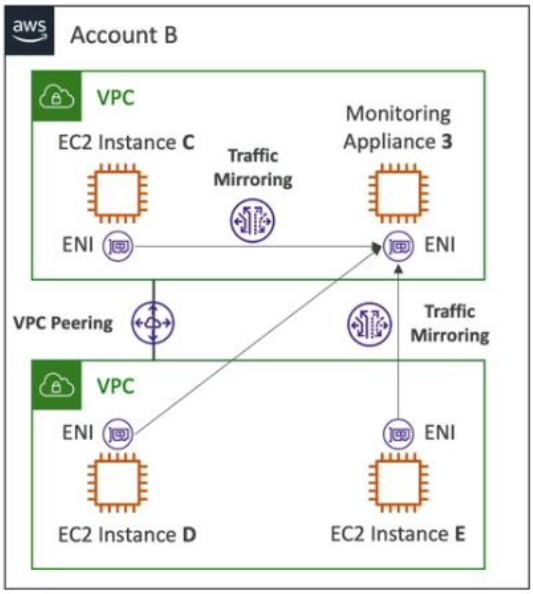
- Some of the VPC Traffic Mirroring architectures could be as below:



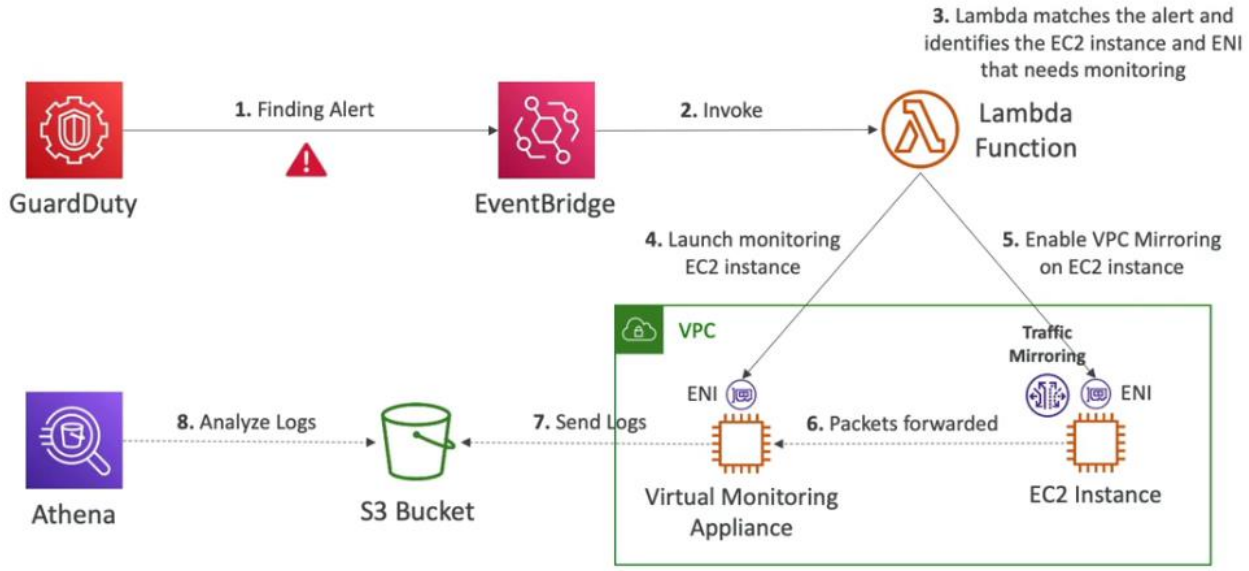


- Distributed VPC Traffic Mirroring architecture could be as below:

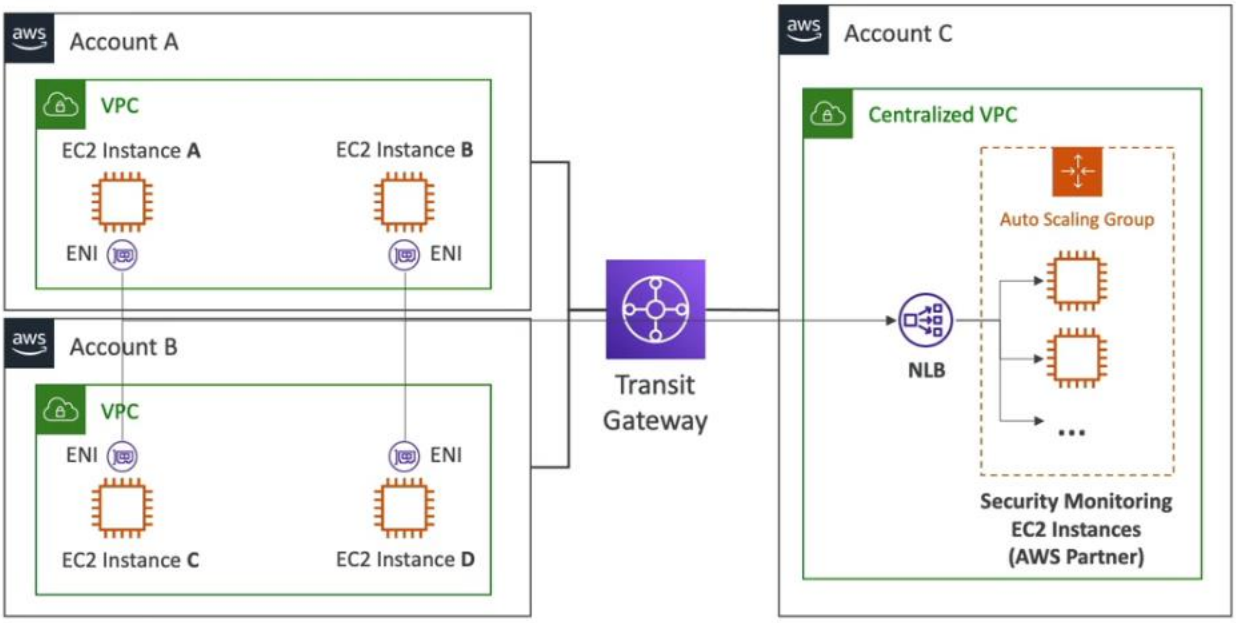




- Automation of VPC traffic mirroring example:



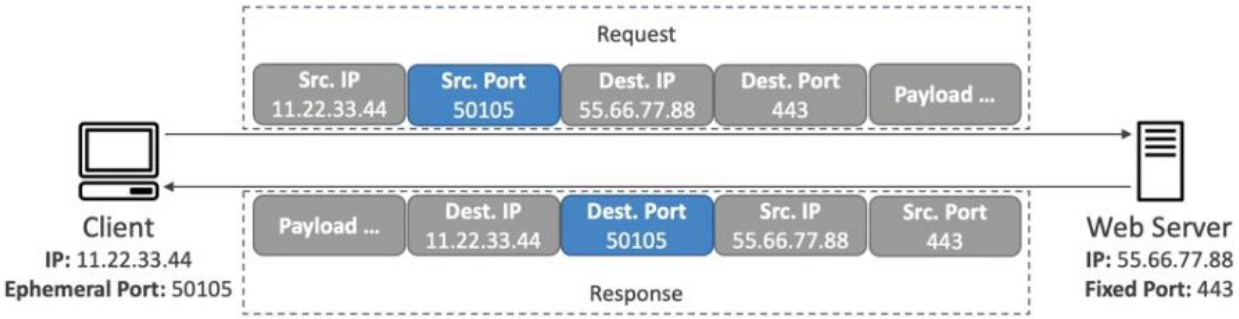
- VPC traffic mirroring centralized deployment model example:
    - Higher transfer data costs due to transit gateway



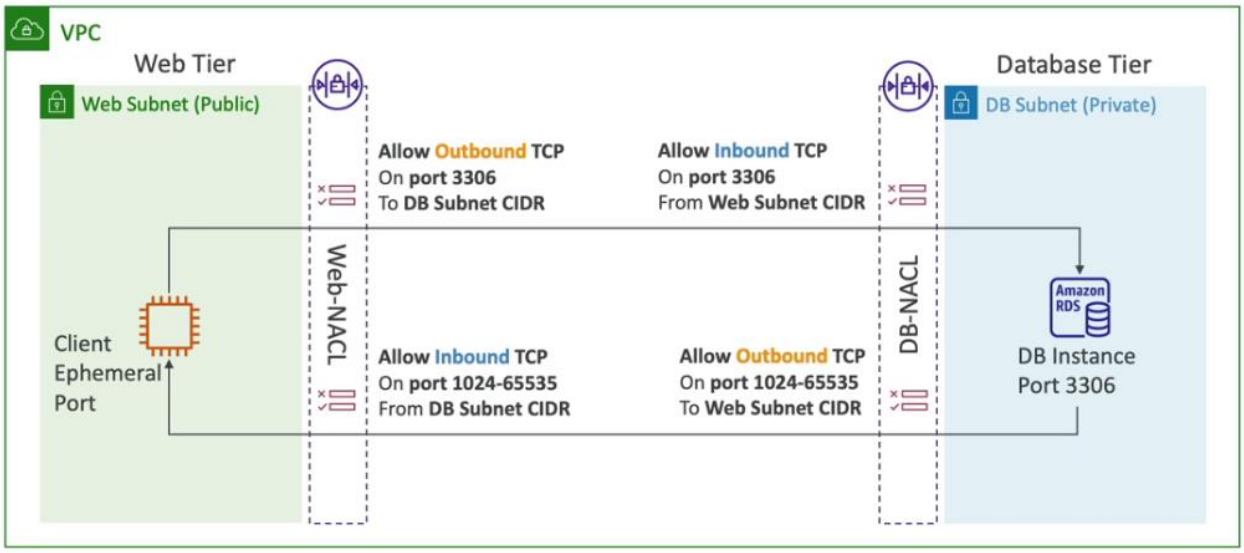
Ephemeral Ports:

- An ephemeral port is utilized IP communication and serves as a short-lived transport protocol port
- If a request is initiated by an instance in the VPC, the network ACL should include an inbound rule to permit traffic to the specific ephemeral ports associated with the instance types
    - Clients connect to a defined port, and expect a response on ephemeral port
- Different operating systems use different port range, examples:
    - IANA & Microsoft Windows – 49152 – 65535

- Many Linux Kernels – 32768 – 60999

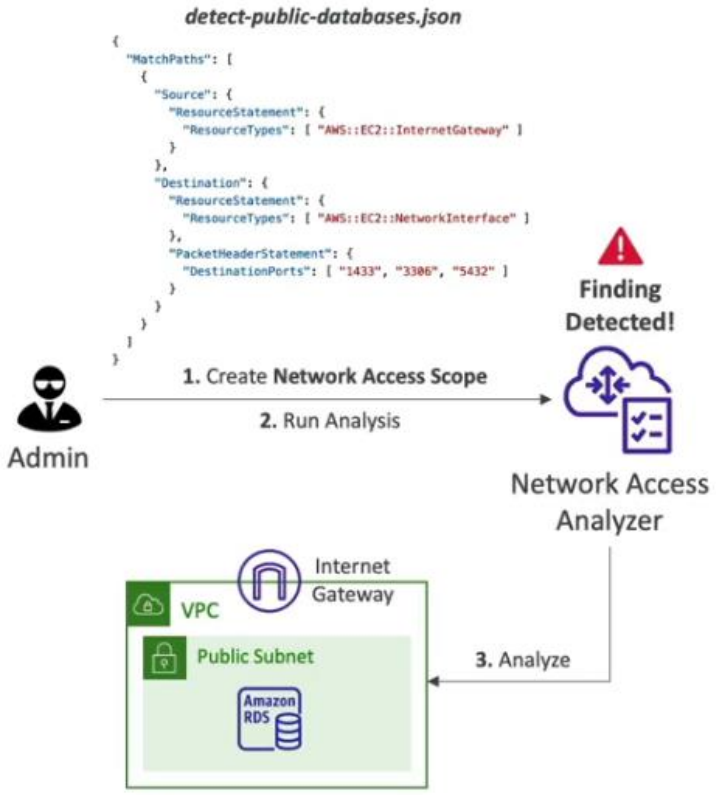


- NACL with ephemeral ports:



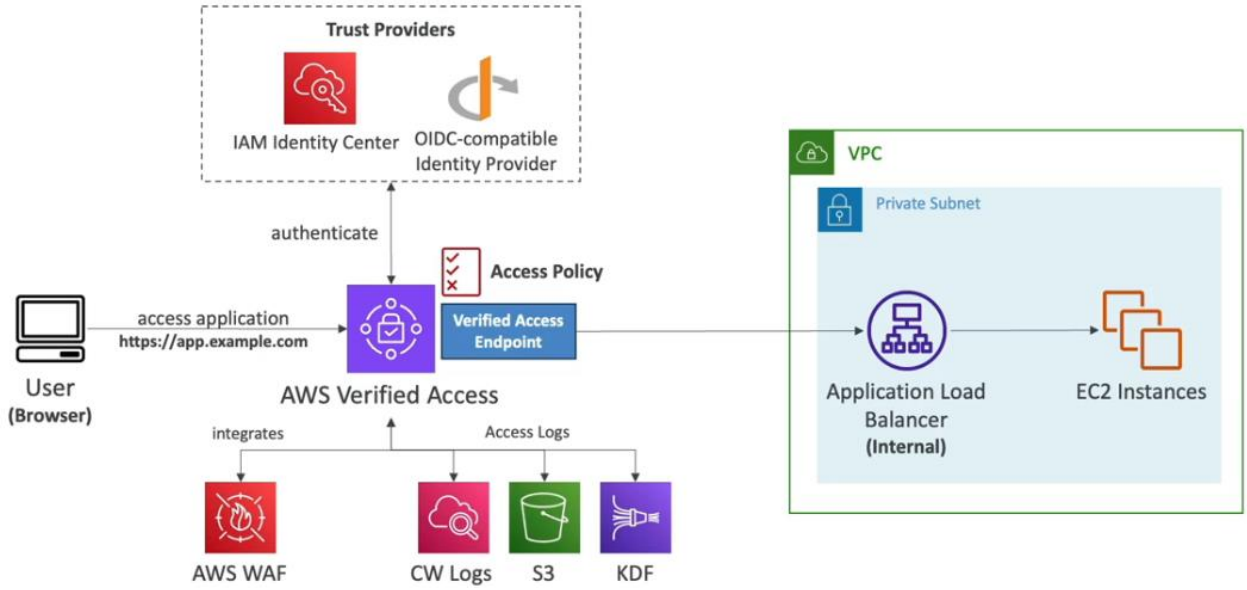
VPC Network Access Analyzer:

- Helps user to understand potential network paths to/from user's resources
- Define network access requirements – for example identify publicly available resources
- Evaluate against them and find issues / demonstrate compliance – for example match against the configuration of user's resources
- Network Access Scope – json document contains conditions to define network security policy – for example detect public database



AWS Verified Access:

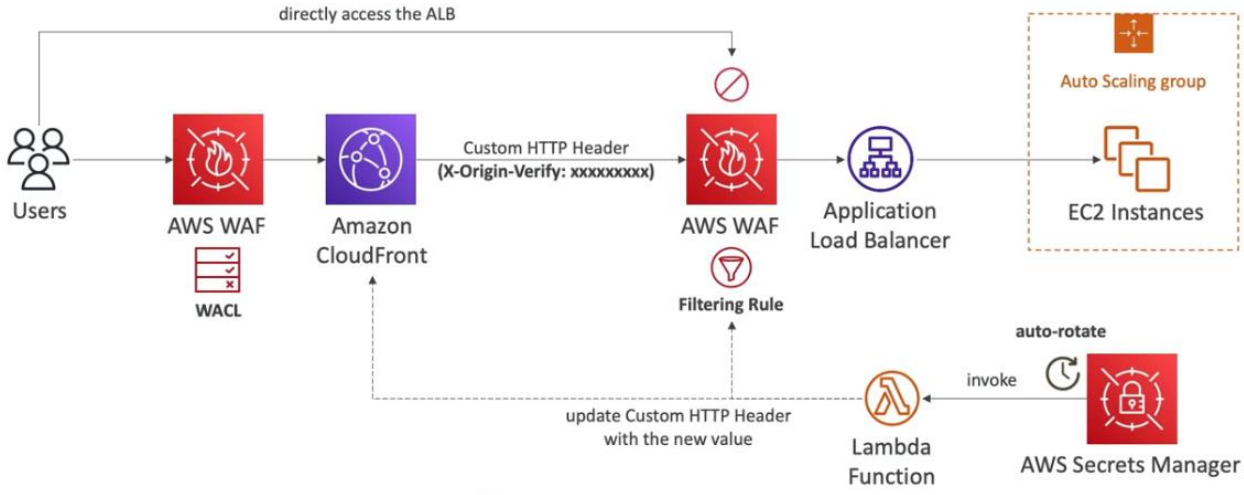
- Provides secure access to corporate applications without a VPN
- Remotely connect to applications form any network
- Implement zero trust principles
- Go beyond network security – user identity, device security state
- Fine-grained access to applications
- Validates each application request in real-time
- Authenticate users against IAM identity Center or 3rd party OIDC IdP (OKTA, CrowdStrike etc.)
- Integrates with AWS WAF to prevent threats – SQLi, XSS etc.
- Logs to S3, CloudWatch Logs, Kinesis Data Firehose, or 3rd Party
- Can send traffic to ALB or ENI



AWS WAF:

- A web application firewall that lets monitor HTTP and HTTPS requests that are forwarded to Amazon CloudFront or an Application Load Balancer
- AWS WAF also let control access to content
- Can configure what IP addresses are allowed to make this request or what query string parameters need to be passed for the request to be allowed – and then the ALB or CloudFront will either allow this content to be received or to give a http 403 code
- At its most basic level, AWS WAF allows 3 different behaviors:
    - Allow all requests except the ones that are specified
    - Block all requests except the ones that are specified
    - Count the requests that match the properties that are specified
    - CAPTCHA
    - Challenge
- Additional protection against web attacks using conditions can be provided using WAF. User will define the conditions by using characteristics of web requests such as the following:
    - IP addresses that request originate from
    - Country that request originate from
    - Values in request header
    - Strings that appear in requests, either specific strings or strings that match regular expression patterns
    - Length/size of requests
    - Rate limiting based rules – to count occurrence of events
    - Presence of SQL code that is likely to be malicious – SQL injection
    - Presence of a script that is likely to be malicious – XSS
- Web ACL can be defined

- Application Load Balancers integrates with WAF at regional level, CloudFront at global level
    - Rules need to be associated with AWS resources to be able to make it work
- WAF integrates with both ALB and CloudFront as well as API Gateway – it does not integrate with EC2 directly, nor Route53, or any other services
    - Can be deployed on AppSync – to protect GraphQL APIs
    - Network Load Balancer does not support WAF
- Amazon IP reputation list is a managed rules group available in WAF.
    - This rule group "Inspects for a list of IP addresses that have been identified as malicious actors and bots by Amazon threat intelligence."
- Enhance CloudFront Origin Security with AWS WAF & AWS Secrets Manager



- AWS WAF can be used to protect websites not hosted in AWS via CloudFront – CloudFront supports custom origins outside of AWS
- IPs can be blocked at a /8, /16, /24 and /32 level – IPv4 and IPv6 are supported
- Offers preconfigured managed rulesets, including AWS Managed Rules for protection against known attack patterns
- Provides detailed logs and metrics to analyze traffic patterns, detect potential attacks, and monitor overall application security
- Generates real-time alerts and notifications when suspicious or malicious activities are detected, enabling quick response
- Leverages threat intelligence from AWS and third-party sources to stay updated with emerging threats and patterns
- AWS WAF does not provide the packet inspection capabilities
- WAF supports 10,000 entries per IP set, and you can use multiple IP sets
- AWS WAF can protect the Cognito User Pool's Hosted UI from web exploits, but it does not control or influence user registration or authentication directly in the user pool
- AWS WAF does not add HTTP security headers to responses.

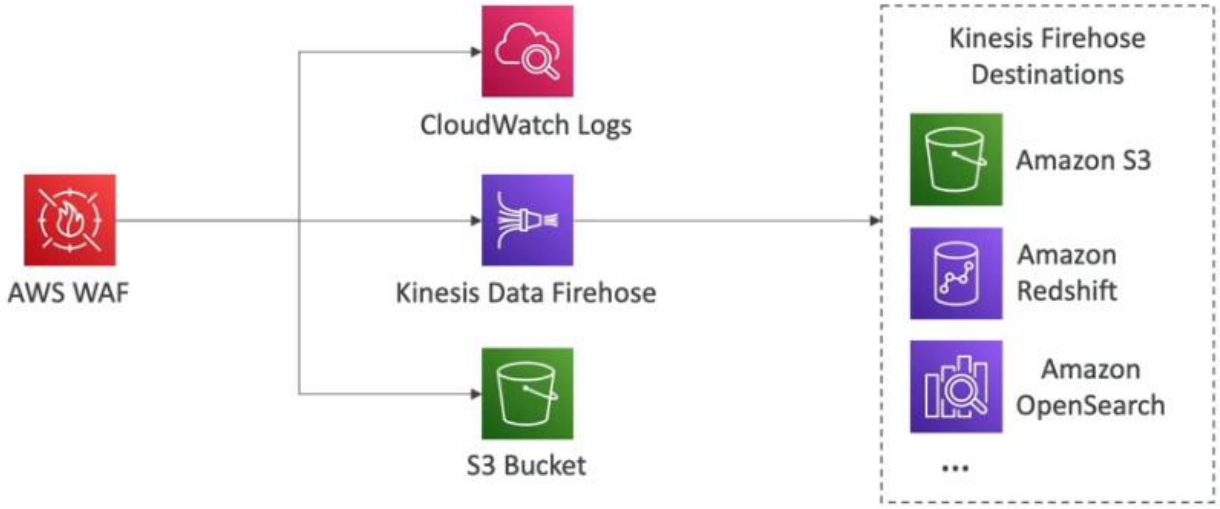
WAF Managed Rules:

- Library of over 190 managed rules

- Ready-to-use rules managed by AWS and AWS Marketplace Sellers
- Baseline Rule Groups – general protection from common threats
  - *AWSManagedRulesCommonRuleSet*
  - *AWSManagedRulesAdminProtectionRuleSet*
- Use-case specific Rule Groups – protection from many AWS WAF use cases
  - *AWSManagedRulesSQLiRuleSet*
  - *AWSManagedRulesWindowsRuleSet*
  - *AWSManagedRulesPHPRuleSet*
  - *AWSManagedRulesWordPressRuleSet*
- IP reputation Rule Groups – block requests based on source (e.g., malicious IPs)
  - *AWSManagedRulesAmazonIpReputationList*
  - *AWSManagedRulesAnonymousIpList*
- Bot control managed Rule Groups – block and manage requests from bots
  - *AWSManagedRulesBotControlRuleSet*

Web ACL Logging:

- Logs can be sent to an:
  - Amazon CloudWatch Logs log group – 5 Mbps
  - Amazon S3 – 5 minutes interval
  - Amazon Kinesis Data Firehose – limited by Firehose quotas



AWS Shield:

- A managed DDoS protection service by AWS
- If offer two tires:
  - Shield Standard – included for free
  - Shield Advanced – additional subscription – 3000$ per month

- AWS Shield utilizes the scalability and mitigation capabilities of the AWS global network infrastructure
- It employs machine learning algorithms to analyze and mitigate malicious traffic patterns
- AWS Shield provides automatic protection against common DDoS attack techniques
- Protects against SYN/UDP Floods, reflection attacks, and other layer 3/layer 4 attacks
- Users can customize protection rules and policies to meet specific application requirements
- Real-time alerts and notifications are generated during attack for immediate response
- AWS Shield seamlessly integrates with other AWS services like CloudFront, Route53, and Elastic Load Balancing

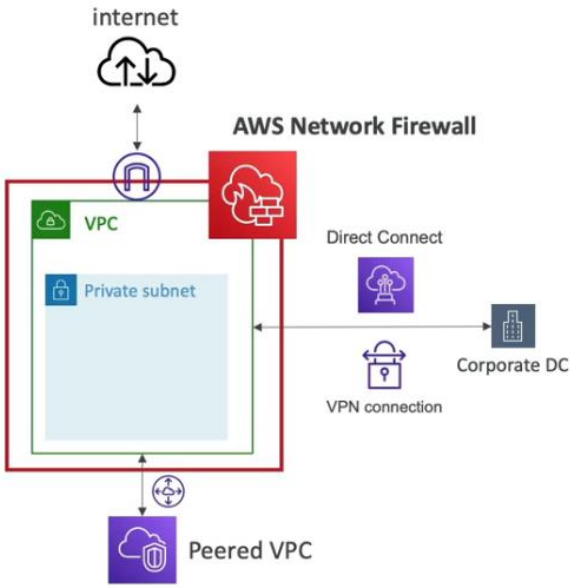
Shield Advanced:

- Shield Advanced - the subscription-based tier, offers enhanced DDoS protection, cost protection, and 24/7 access to DDoS experts for personalized support
    - Always-on
    - Flow-based monitoring of network traffic
    - Active application monitoring to provide near real-time notifications of DDoS attacks
    - Protect the AWS bill against higher fees due to ELB, Amazon CloudFront, and Amazon Route53 usage spikes during DDoS attacks
- Shield Advanced CloudWatch Metrics – helps to detect if there's a DDoS attack happening
    - DDoSDetected – indicates whether a DDoS event is happening for a specific resource
    - DDoSAttackBitsPerSecond – number of bits per second during a DDoS event for a specific resource
    - DDoSAttackPacketsPerSecond – number of packets per second during a DDoS event for a specific resource
    - DDoSAttackRequestsPerSecond – number of requests per second during a DDoS event for a specific resource

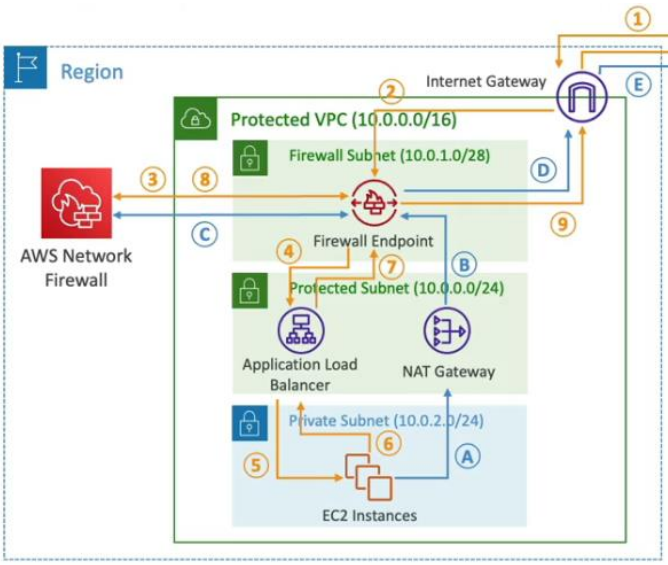
AWS Network Firewall:

- Protect the user's entire Amazon VPC
- From layer 3 to layer 7 protection
- Inspect any kind of traffic:
    - VPC to VPC traffic
    - Outbound to internet
    - Inbound from internet
    - To/from DirectConnect and Site-to-Site VPN
- Internally the AWS Network Firewall uses the AWS Gateway Load Balancer
- Rules can be centrally managed cross-account by AWS Firewall Manager to apply to many VPCs
- Supports thousands of rules
    - IP & Port – example: 10,000 of IPs filtering
    - Protocol – example: block the SMB protocol for outbound communications
    - Stateful domain list rule groups – example: only allow outbound traffic *.abc.com or third-party software repo

- General pattern matching using regex
- Traffic filtering – allow, drop, or alert for the traffic that matches the rule
- Active flow inspection – to protect against network threats with intrusion-prevention capabilities
- Send logs of rule matches to S3, CloudWatch Logs, Kinesis Data Firehose



- Can be configured to forward traffic to other AWS services for further analysis, monitoring or processing – Lambda, S3, CloudWatch
- Supports third-party threat intelligence feed
- Can be managed via AWS Management Console, CLI or SDK
- Some Network Firewall deployment architectures



IGW Ingress Route Table

Destination	Target
10.0.0.0/16	local
10.0.0.0/24	vpce-id

Inspection Subnet Route Table

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

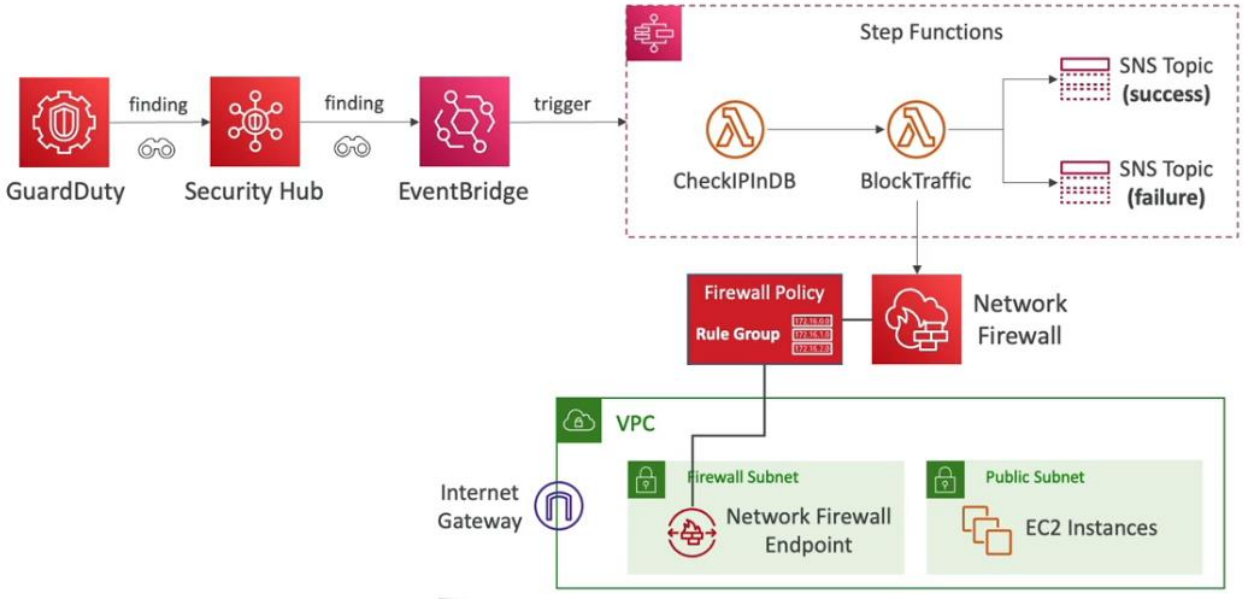
Protected Subnet Route Table

Destination	Target
10.0.0.0/16	local
0.0.0.0/0	vpce-id

Private Subnet Route Table

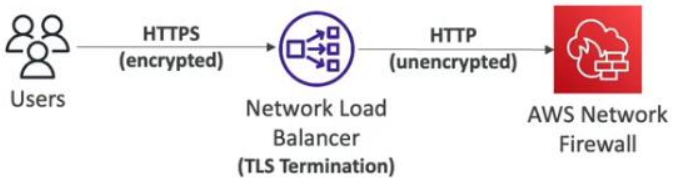
Destination	Target
10.0.0.0/16	local
0.0.0.0/0	natgw-id

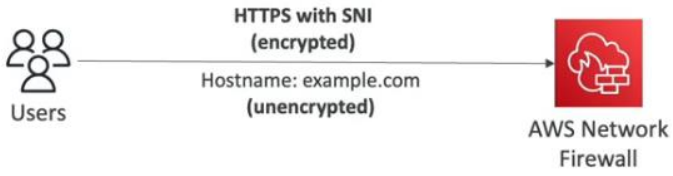
- Deployment with other security services and forwarding notifications of the incident response via Network Firewall



Encrypted Traffic:

- AWS Network Firewall does not support deep packet inspection
  - First decrypt the traffic using Network Load Balancer – where deep packet inspection can be done
  - If using HTTPS with SNI, blocking based on the SNI Hostname content (which is unencrypted) can be done





AWS Firewall Manager:

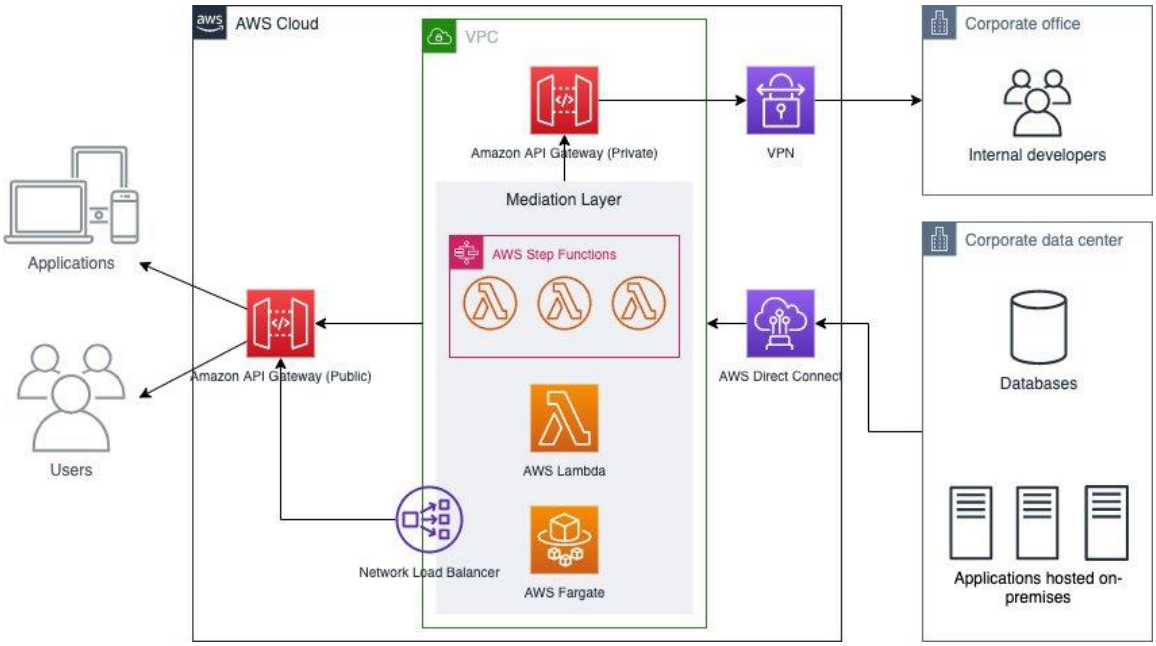
- A security management service that allows centralized management and configuration of firewall rules across multiple AWS accounts and resources

- It provides a simple and scalable way to enforce and maintain firewall rules at scale
- It integrates with AWS WAF, AWS Shield, and VPC security groups to provide comprehensive network protection
- Allows to define security policies and rulesets centrally and apply them to multiple accounts and resources
- Supports rule-grouping, allowing to define and enforce multiple rules together for better security posture
- Supports integration with AWS Organizations to manage and protect resources across multiple accounts
- Provides a global rule group that can be applied to multiple AWS regions for consistent policy enforcement
- Supports custom rule creation and management, allowing to define organization-specific rules
- Provides detailed logging and monitoring capabilities to track rule enforcement and security events
- Designed to simplify and streamline the management of firewall rules, reducing operational complexity and enhancing security posture
- Rules are applied to new resources as they are created across all and future accounts in the user's organization – good for compliance
- Security Policy – common set of security rules
    - WAF rules – ALB, API Gateways, CloudFront
    - AWS Shield Advanced – ALB, CLB, NLB, Elastic IP, CloudFront
    - Security Groups – for EC2, ALB, and ENI resources in VPC
    - AWS Network Firewall – VPC level
    - Amazon Route53 Resolver DNS Firewall
    - Policies are created at the region level

API Gateway:

- Fully managed service that enables developers to create, publish, maintain, monitor, and secure APIs at any scale
- Allows to create RESTful APIs and easily integrate them with various AWS services, such as Lambda functions, DynamoDB table, or HTTP endpoints
- Provides a simple process for deploying and versioning APIs, allowing to manage changes and updates effectively
- It offers features for managing the lifecycle of APIs, including stages, environments, and API versions

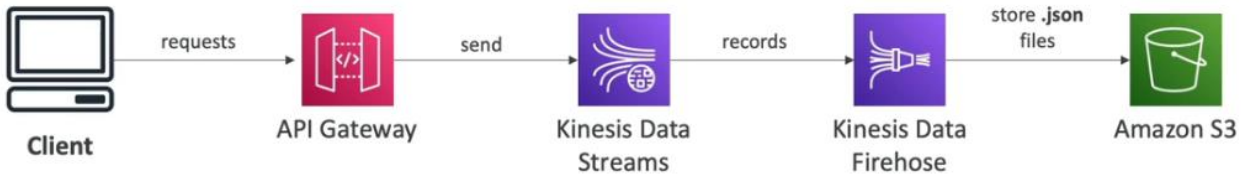
- Supports various authentication mechanisms, including IAM, Lambda authorizers, and custom authentication
- Allows to define fine-grained authorization policies to control access to APIs, including IAM policies, resource policies, and custom authorizers
- Supports for the web socket protocol
- Provides detailed monitoring and logging capabilities, allowing to track API usage, performance, and errors using CloudWatch Logs and metrics
- Offers built-in caching to improve API performance and reduce the load on backend systems
- Automatically scales to handle high volumes of API traffic and provides regional redundancy for high availability
- Allows to create and publish developer portals with documentation, SDK generation, and API key management to facilitate API consumption and collaboration
- Integrates with AWS services like AWS WAF, AWS CloudTrail, and AWS X-Ray, providing enhanced security, auditing, and tracing capabilities



Integrations High Level:

- Lambda Functions
  - Invoke Lambda function
  - Easy way to expose REST API backed by AWS Lambda
- HTTP
  - Expose HTTP endpoint in the backend
  - Example: internal HTTP API on premises, ALB,..
  - Why? Add rate limiting, caching, user authentications, API keys, etc.
- AWS Service
  - Expose any AWS API through the API Gateway
  - Example: start an AWS step function workflow, post a message to SQS

- Why? Add authentication, deploy publicly, rate control
- For example – AWS service integration kinesis data streams



API Gateway Endpoint Types:

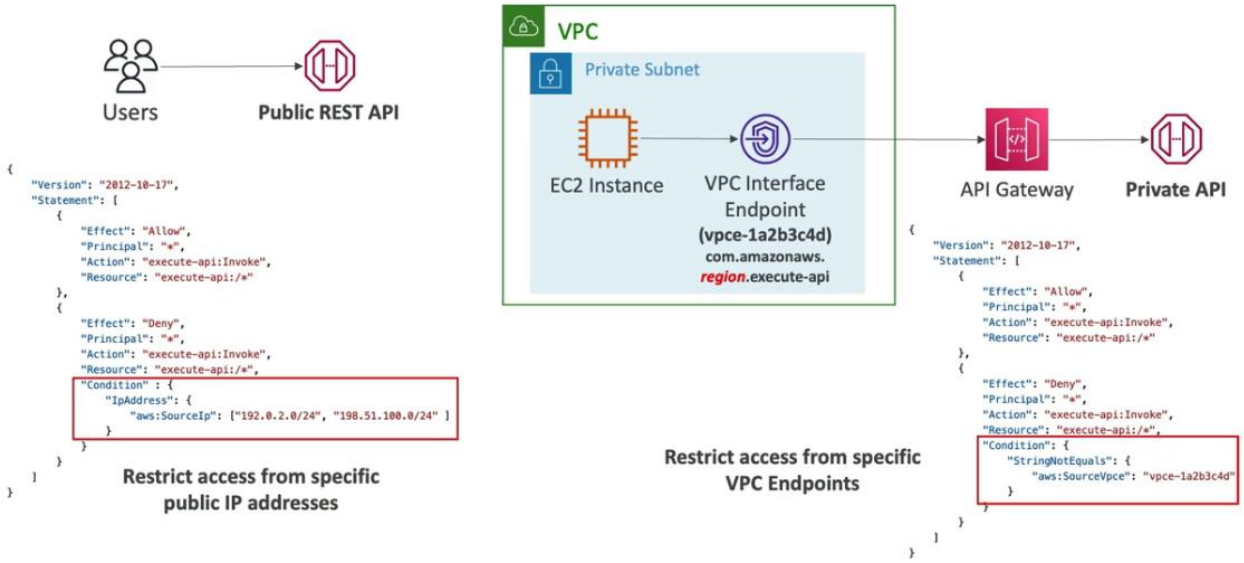
- Edge-Optimized (default) – for global clients
  - Requests are routed through the CloudFront Edge locations (improve latency)
  - The API Gateway still lives in only one region
- Regional – for clients within the same region
  - Could manually combine with CloudFront (more control over the caching strategies and the distribution)
- Private – only accessible within user's private network
  - Can only be accessed from user's VPC using an interface VPC Endpoint (ENI)
  - Use a resource-based policy to define access

API Gateway Security:

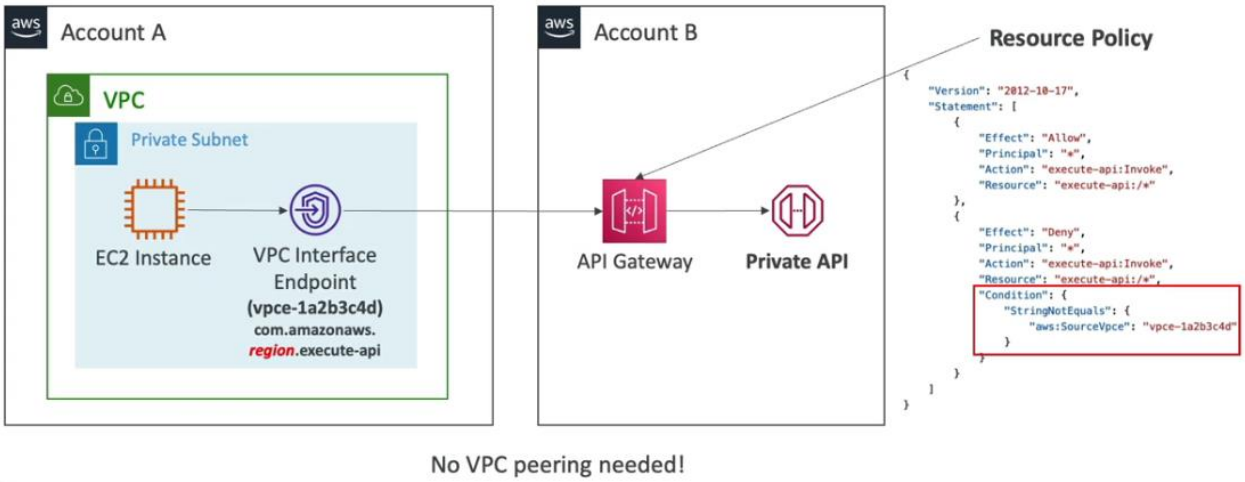
- User Authentication through
  - IAM roles – useful for internal applications
  - Cognito – identify for external users like mobile users
  - Custom Authorizer – user's own logic via lambda functions
- Custom Domain Name HTTPS – security through integration with ACM
  - If using edge-optimized endpoint – certificate must be in us-east-I
  - If using regional endpoint – certificate must be in the API Gateway region
  - Must setup CNAME or A-alias record in Route53

API Gateway Resource Policy:

- Users accessing public REST API or Private API being accessed from EC2 instance



- Cross VPC Same-Region Access



API Gateway Throttling:

- A feature that allows to control and limit the rate of requests made to an API deployed in Amazon API Gateway
- Throttling helps prevent an API from being overwhelmed by too many requests by enforcing rate limits and request quotas
- Enforces steady-state request rate limits and burst limits – stead-state limits define the maximum number of requests per second, while bursts limits allow short bursts of requests that exceed the steady-state rate
- When a client exceeds the defined rate limits or quotas, API gateway fails the limit-exceeding requests and returns a 429 Too Many Requests error response to the client
- API Gateway limits the bursts to 10,000 requests across all APIs within an AWS account – this helps prevent a single API from consuming excessive resources during peak periods

- Throttling can be configured at different levels, such as the API, stage, or method level, using usage plans which allow to define rate limits and quotas for different API consumers based on their API Key, identity, or other criteria
- One API Gateway that is overloaded and not limited can cause the other APIs to be throttled

API Gateway Caching:

- Allows to cache API responses at CloudFront edge locations
- Caching reduces backend server load and improve response time for subsequent identical requests
- Cache settings can be configured at the API, stage, or method level with customizable cache key parameters
- Support for cache invalidation ensures data freshness when underlying data changes
- Can control caching behavior using cache control headers like max-age, no-cache, or private
- Cached responses are encrypted and communicated securely between API Gateway and CloudFront
- Can monitor cache hits, misses, and latency using CloudWatch metrics
- Can configure caching for specific response headers, status codes, or request parameters
- When a user enables caching for a stage, API Gateway caches responses from the user's endpoint for a specific TTL period, in seconds
- The default TTL value for API caching is 300 seconds – the maximum TTL value is 3600 seconds – TTL = 0 means caching disabled

Amazon Athena:

- An interactive query service that allows to analyze data stored in Amazon S3 using standard SQL syntax – built on Presto
- It eliminates the need to set up and manage complex infrastructure for data analysis - serverless
- It automatically scales based on the size of the data and query complexity, providing fast query performance
- It supports a wide range of data formats, including CSV, JSON, Parquet, ORC, Avro and more
- Users will pay for the queries they run, with no upfront costs or data loading requirements
    - 5$ per TB of data scanned
- It integrates seamlessly with other AWS service
- It provides schema-on-read functionality, allowing to query structured and unstructured data without predefining a schema
- It provides easy integration with popular business intelligence tools and visualization platforms for data analysis and reporting – Quicksight

- It offers built-in security features, including integration with IAM for fine-grained access control
- It supports query result caching for improved performance and cost optimization

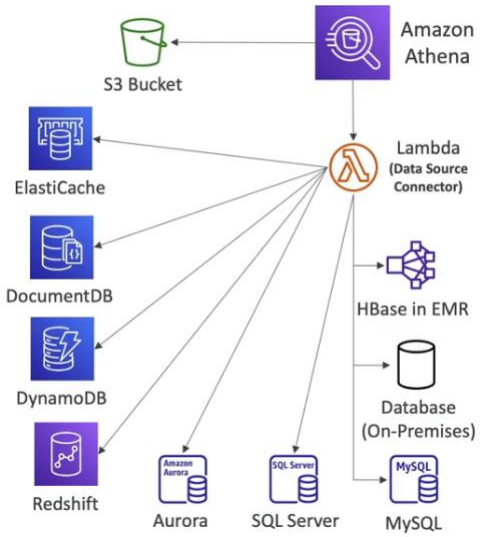
Performance Improvement:

- Use columnar data – for cost saving (less scans) – fast query performance
  - Apache Parquet or ORC is recommended
  - Use Glue to convert data to Parquet or ORC
- Compress data – for smaller retrievals, reduce the amount of data scanned > improve query performance
  - bzip2, gzip, lz4, snappy, zlip, zstd,..
- Query result caching – reuse the results of frequent or expensive searches
- Improve query planning – collect and analyze table statistics to provide query optimizer with accurate metadata
- Data partition pruning – skip irrelevant partitioning during query execution
  - Partition data sets in S3 for easy querying on virtual columns
- Use bigger files – (>128 MB) many smaller files are not recommended – minimize the overhead

Federated Query:

- Allows to query data from different data sources using a single query
- Enables to join data across multiple data sources – S3, RDS, and more without the need for data movement or ETL processes
- Supports various connectors , including JDBC and ODBC to connect to data sources
- Virtual tables can be defined in Athena that represent the external data sources and can be queries alongside the native Athena tables
- Federated query leverages the Athena query engine to optimize performance and parallelize query performance

- Supports pushdown predicates and aggregations to minimize data transfer and improve query performance

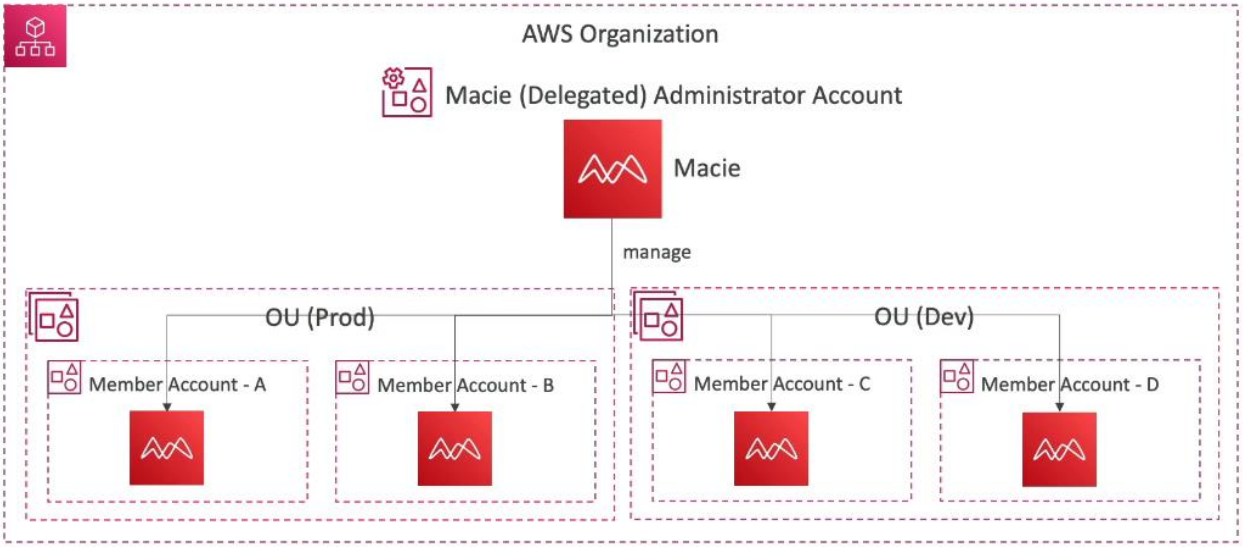


AWS Macie:

- Automatically scans and discovers sensitive data within the AWS environment, such as PII, financial data, and intellectual property
- It classifies and labels sensitive data using machine learning algorithms, providing insights into the type of data and its associated risks
- It helps enforce data protection policies by identifying and alerting on data exposures, access anomalies, and unauthorized sharing of sensitive information
- It continuously monitors data access patterns, changes, and suspicious activities to identify potential security risks and data breaches

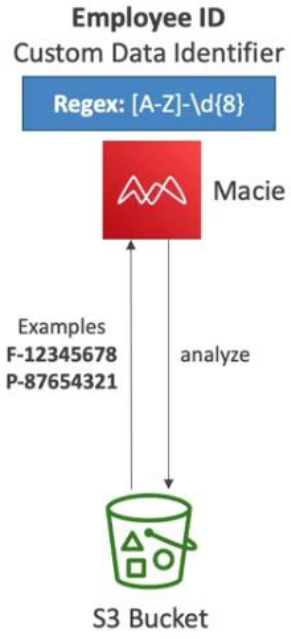
- Allows to define custom policies based on organization's specific data protection requirements, ensuring compliance with regulations and industry best practices
- It can also analyze CloudTrail logs
- It integrates with various AWS services, including S3, AWS Glue, AWS Lambda, and IAM, to provide comprehensive data protection across AWS infrastructure

- It provides detailed visibility into data usage, access patterns, and generates reports to support auditing and compliance requirements
- Macie scales automatically to handle large data volumes and offers automation capabilities to streamline data protection workflows
- With ongoing machine training and updates, Macie improves its accuracy in identifying sensitive data and detecting emerging data protection risks
- Great for PCI DSS and preventing ID theft
- Macie analyzes the content of files to determine their types, such as documents, images, videos, or archives – PDF, JSON, Excel, TAR, or Zip file, source code, XML
- It uses machine learning algorithms to identify common themes or topics present in the data, allowing for a deeper understanding of the data's context – AmEx, Visa, MasterCard credit card keywords, banking or financial keywords, hacker and web exploitation keywords
- Considers the file extension or format to infer the type of data it contains – .bin, .c, .bat, .exe, .html, .sql
- Can apply regular expressions patterns to detect specific patterns for format of data, such as credit card numbers, social security numbers, or email addresses – aws_secret_key, RSA Private Key, SWIFT codes, Cisco Router Config
- Supports multi-account strategy



Data Identifiers:

- Two types: Managed & Custom
- Managed Data Identifiers – pre-configured patterns and rules used by Amazon Macie to identify sensitive data within AWS account
- Include a wide range of pre-defined data types – credit card numbers, social security numbers, email addresses, IP addresses, and more
- They use regular expressions and algorithms to match and identify sensitive data patterns
- Custom Data Identifiers – can be customized and expanded to meet specific data identification requirements by users



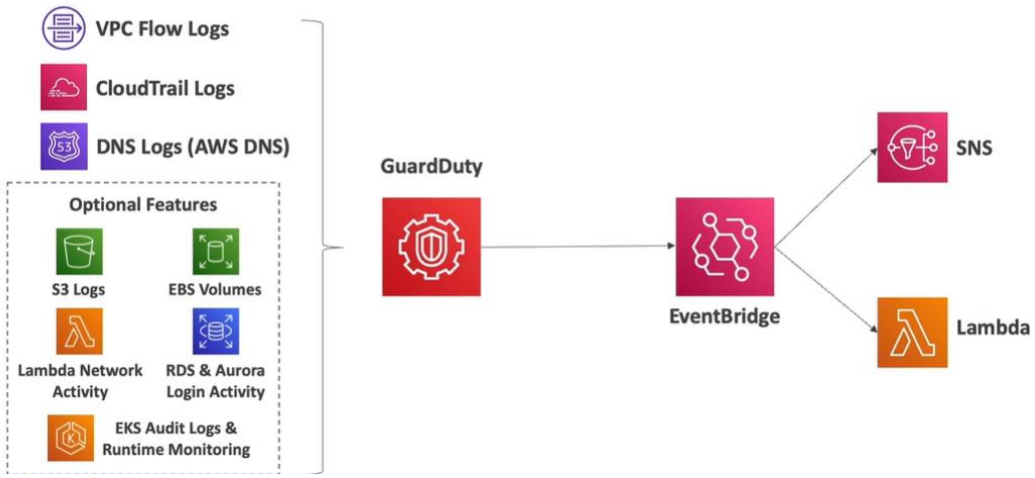
- Macie Managed Data Identifiers are regularly updated by AWS to include to improve patterns and improve detection accuracy
- Used by Macie to scan and analyze data stored in AWS – S3, RDS, Redshift etc.
- Allow List – can be used to define a text pattern to ignore

Macie Findings:

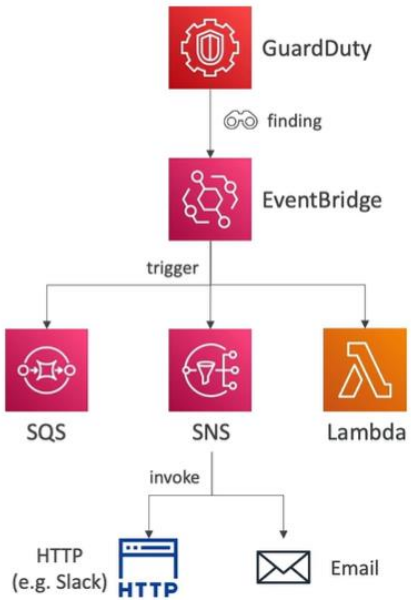
- A report of potential issue or sensitive data that Macie found
- Each finding has a severity rating, affected resource, datetime
- Sensitive Data Discovery Result – a record that logs details about the analysis of an S3 object
    - Configure Macie to store the results in S3, then query using Athena
- Suppression Rules – set of attributes-based filter criteria to archive findings automatically
- Findings are stored for 90 days
- Review findings using AWS console, EventBridge, Security Hub
- Finding Types could be:
    - Policy Findings – a detailed report of policy violation or issue with the security of S3 bucket
        - Default encryption is disabled
        - Bucket is public
        - Detect changes only after Macie is enabled
        - Policy:IAMUser/S3BucketEncryptionDisabled, Policy:IAMUser/S3BucketPublic
    - Sensitive Data Findings – a detailed report of sensitive data that's found in S3 buckets
        - Private keys (Credentials), credit card numbers(Financial) etc.
        - *SensitiveData:S3Object/Credentials*, *SensitiveData:S3BOject/Financial*
        - *SensitiveData:S3Object/CustomIdentifier*

AWS GuardDuty:

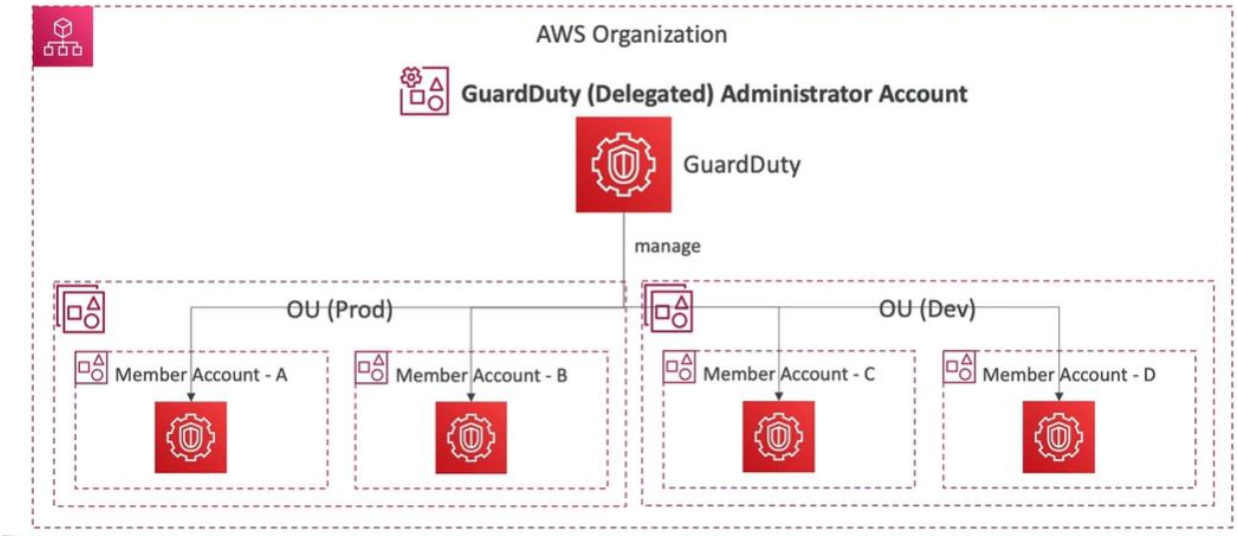
- Uses machine learning and anomaly detection techniques to analyze event logs and network traffic data, identifying potential security threats and unauthorized activities
- Provides real-time monitoring, analyzing data form various sources, including VPC Flow Logs, CloudTrail logs, and DNS logs, to detect suspicious behavior and security risks



- Automatically analyzes and correlates findings from multiple sources, aggregating security alerts and providing a consolidated view of potential threats
- Leverages threat intelligence feeds from AWS, partner sources, like ProofPoint, CrowdStrike etc. and the broader security community to enhance its detection capabilities and identify known malicious activities
- Detect network-based threats such as port scanning, unusual network traffic patterns, and compromised instances, or host-based threats
- Can detect unusual API calls, unauthorized deployments, attempts to disable CloudTrail logging etc.
- Alerts appear in the GuardDuty console and CloudWatch Events
- Integrates seamlessly with AWS services, requiring no additional software or agent installation
- Can add trusted IP lists for whitelisting IP addresses – GD will not generate findings
- Threat lists where known malicious IPs can be added if required – GD will generate findings
- Offers interactive visualization and detailed findings, helping security team understand the context and impact of detected threats
- Can be integrated with AWS Lambda, CloudWatch Events, or other automation tools to automate response actions, such as blocking IP addresses or isolating compromised resources
- Response can be automated to security issues revealed by GuardDuty findings using Event Bridge

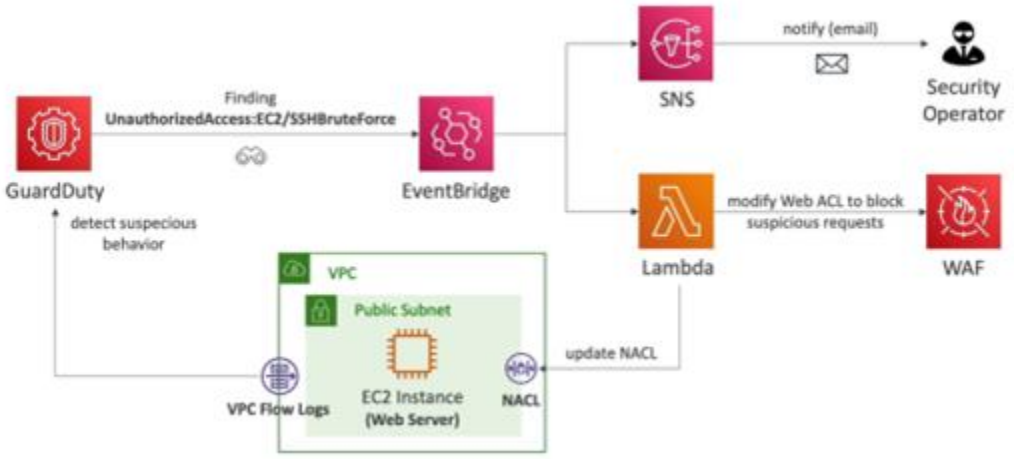


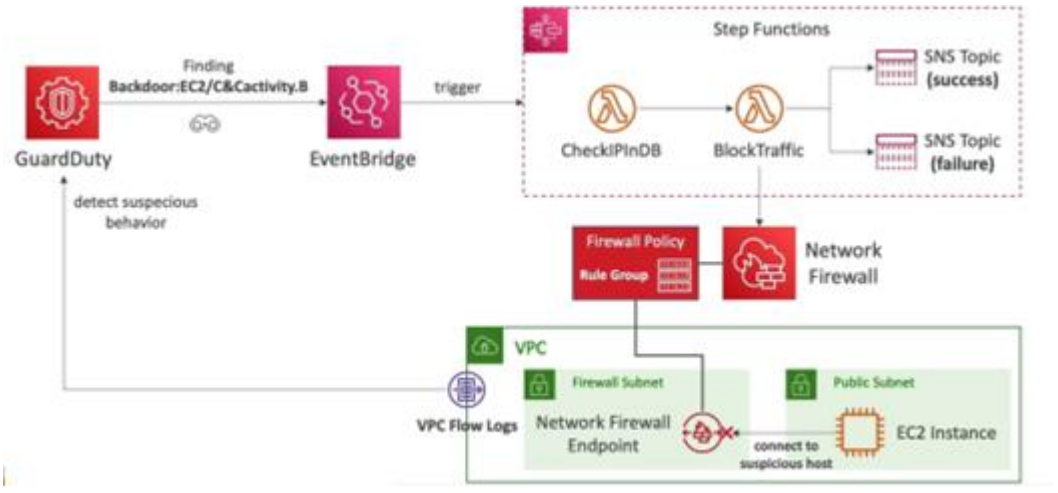
- Centralize threat detection across multiple AWS accounts
    - Events are published to both administrator account and the member account that it is originated from



- Fully managed service with a pay-to-go pricing model – 30 days free and charges are based on:
    - Quantity of CloudTrail events
    - Volume of DNS and VPC Flow Log data
- Takes 7-14 days to set a baseline – what is normal behavior in the AWS account
- GuardDuty pulls independent streams of data directly from CloudTrail logs (management event, data event), VPC Flow Logs or EKS Logs
- Each finding has a severity value (High, Medium, Low)

- Finding naming convention – *ThratPurpose:ResourceTypeAffected/ThreatFamilyName.DetectionMechanism!Artifact*
  - ThreatPurpose – primary purpose of threat (Backdoor, Cryptocurrency)
  - ResourceTypeAffected – which AWS resource is the target (EC2, S3)
  - ThreatFamilyName – describes the potential activity (NetworkPortUsual)
  - DetectionMechanism – method GuardDuty detecting the finding (TCP, UDP)
  - Artifact – describes the resource that is used in the malicious activity (DNS)
- Finding Types can be:
  - EC2 – *UnauthorizedAccess:EC2/SSHBruteForce*, *CryptoCurrency:EC2/BitCoinToo.B!DNS*
  - IAM – *Stealth:IAMUser/CloudTrailLoggingDisabled*, *Policy:IAMUser/RootCredentialUsage*
  - Kubernetes – *CredentialAccess:Kubernetes/MaliciousIPCaller*
  - Malware Protection – *Execution:EC2/SuspiciousFile*, *Execution:ECS/SusiciousFile*
  - RDS – *CredentialAccess:RDS/AnomalousBehavior.SuccessfulLogin*
  - S3 – *Policy:S3/AccountBlockPublicAccessDisabled*, *PenTest:S3/KaliLinux*



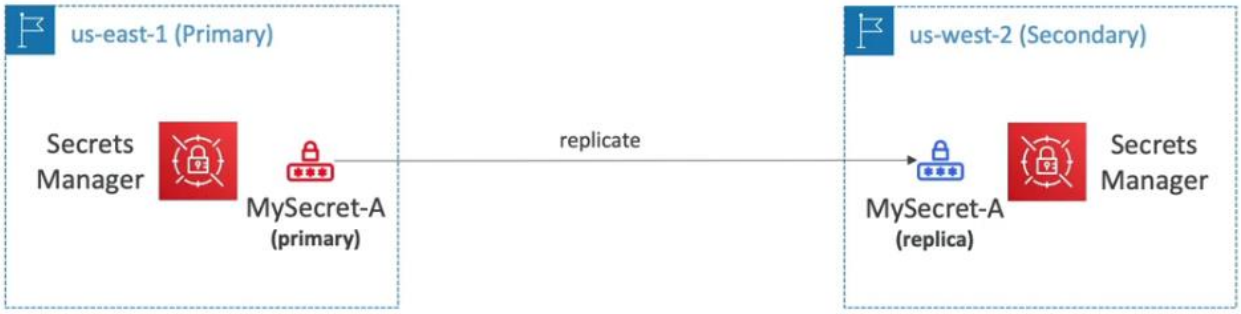


AWS Secrets Manager:

- Provides a secure and encrypted repository to store sensitive information, such as API keys, database credentials, and other secrets
- Offers a centralized location to manage secrets across different applications, services, and AWS accounts, reducing complexity and improving security
- Seamlessly integrates with multiple AWS services, allowing to secretly retrieve and use secrets in applications and infrastructure – MySQL, PostgresSQL, Aurora etc.
- Can be controlled using IAM policies, providing granular control over who can access and modify the store secrets
- It also generates detailed logs and metrics, enabling monitoring, auditing, and compliance efforts
- Offers integration capabilities with third-party tools, allowing for automated secret retrieval and management in different workflows
- Can also securely stores the configuration data
- Uses encryption in-transit and at rest using KMS
- The first rotation will happen immediately, so any applications which rely on this secret need to be updated to retrieve it from Secrets Manager otherwise they will no longer be able to access the database
- Built-in integration with RDS, MySQL, PostgresSQL, Aurora
- 0.40$ per secret per month – 0.05$ per 10,000 API calls

Multi-region Secrets:

- Replicate secrets across multiple AWS regions
- Secrets Manager keeps read replicas in sync with the primary secret

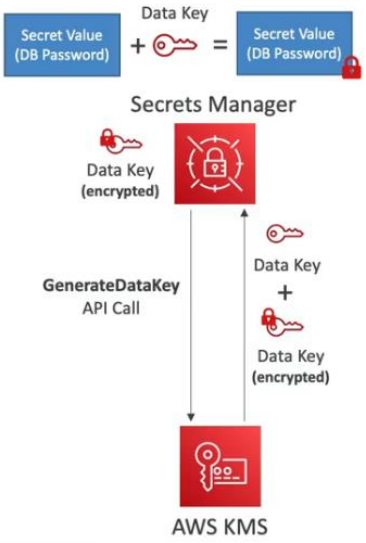


- Ability to promote a read replica Secret to a standalone Secret
- Use case – multi-region apps, disaster recovery strategies, multi-region DB etc.

KMS with Secrets Manager:

- Secrets Manager uses KMS to encrypt/decrypt every version of every Secret value
- Each secret value is encrypted with a unique data key – Envelope Encryption
- Specify the KMS key or use AWS Managed Key – aws/secretsmanager
- Works only with Symmetric KMS Keys

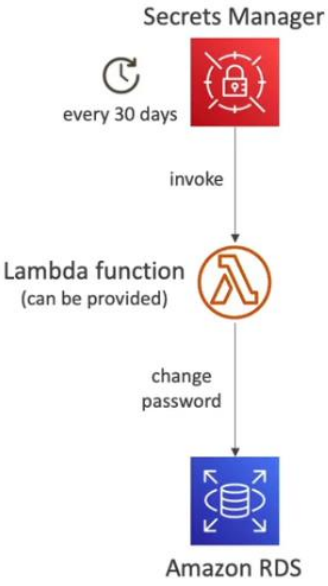
- Encryption process takes place in Secrets Manager



- **Note:** you must have access to both the KMS key and the Secret in Secrets Manager

Secrets Rotation:

- Automatically and periodically update a Secret
- Automated password rotation for integrated databases
    - RDS, RedShift, DocumentDB, other databases etc.
- Credentials are changed in the Secret and the databases
- Secrets Manager use Lambda function to rotate Secrets
- When Secret Rotation is enabled, the secret is rotated immediately

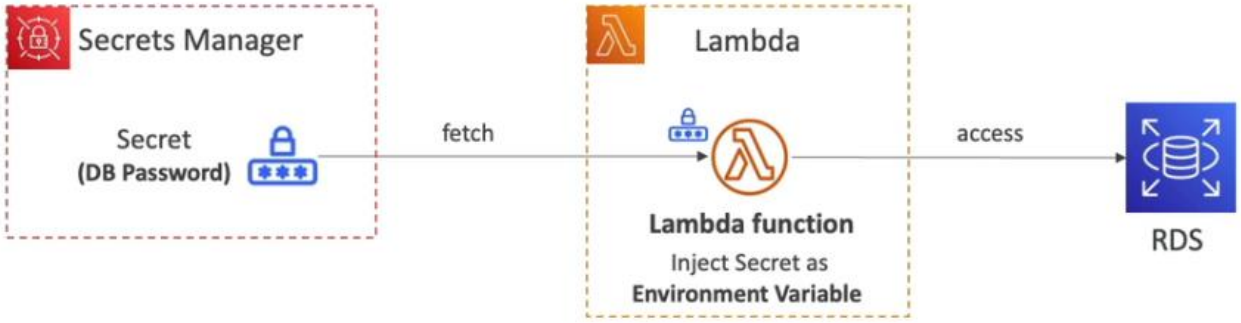


- **Note:** Lambda function must have access to both Secrets Manager and database

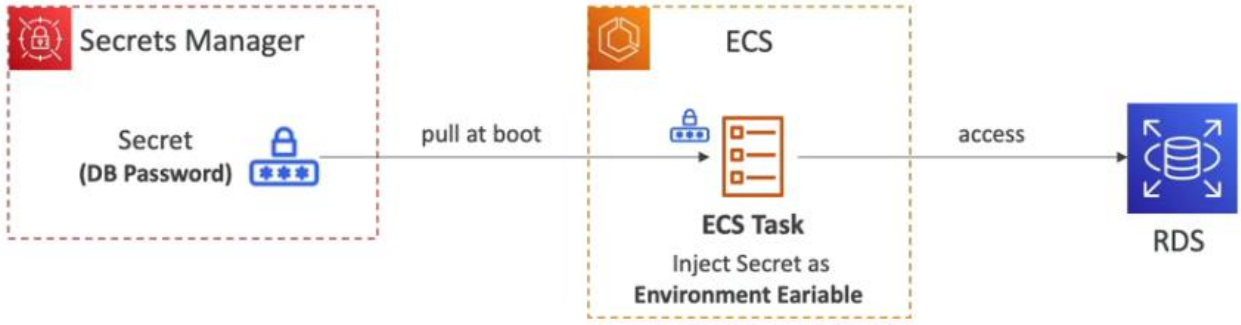
- If Lambda in VPC private subnet, use VPC Endpoints or NAT Gateway

Secrets Manager's Integrations:

- With **Lambda**



- With **ECS**



Resource Policy for Secrets Manager:

- Specify who can access a Secret and what actions an IAM identity can perform

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

- Use cases – grant access to a single Secret for multiple users
  - Enforcing permissions – adding an explicit deny to a Secret
  - Sharing a Secret between AWS accounts

AWS SES:

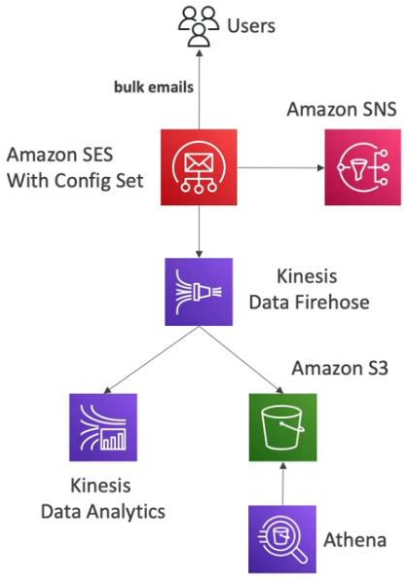
- A cloud-based email sending and receiving service provided by AWS – globally at scale
- Enables to send transactional, marketing, and notification emails reliably and efficiently

- Provides a simple and flexible API for sending emails programmatically, integrating seamlessly with applications and systems – uses a standard SMTP interface
- SES offers tools to monitor the deliverability of emails, providing insights into bounces, complaints, and other email-related metrics
- Can be used as a mailbox for receiving incoming emails, processing them, and triggering actions based on their content
- Designed to handle high email volumes and scale to meet the demands of applications and business
- Manages email sender reputation, ensuring optimal deliverability and reducing the chances of emails being marked as spam
- Includes mechanisms to filter out unwanted or malicious content from incoming or outgoing emails, enhancing security and compliance
- Integrates seamlessly with other AWS services, such as AWS Lambda, S3, allowing to trigger actions and store email-related data
- Pay-as-you-go pricing model
- Supports DKIM and SPF
- Provides features like encryption, secure transmission, and data residency options – Compliance

Configuration Sets:

- When configuring access to SES for EC2 instances – configure the security groups associated with EC2 instance to allow connections to the SES SMTP endpoint
    - Port 25 is the default
    - To avoid timeouts, use either 587 or 2587
- Event Destinations

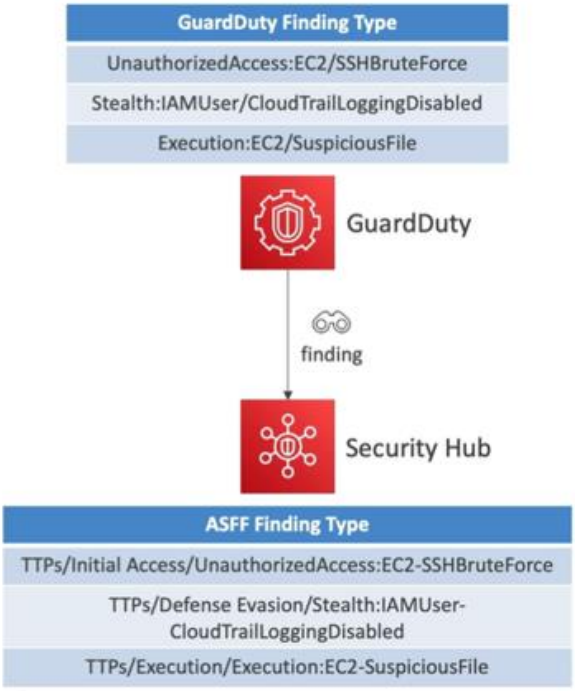
- Kinesis Data Firehose – number of sends, deliveries, opens, clicks, bounces, and complaints for each email
- SNS – for immediate feedback on bounce and complaint information



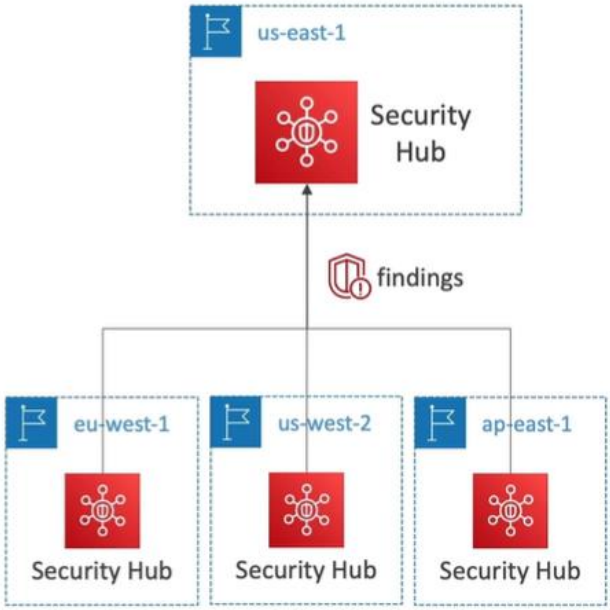
- IP Pool Management – use IP pools to send particular types of emails

AWS Security Hub:

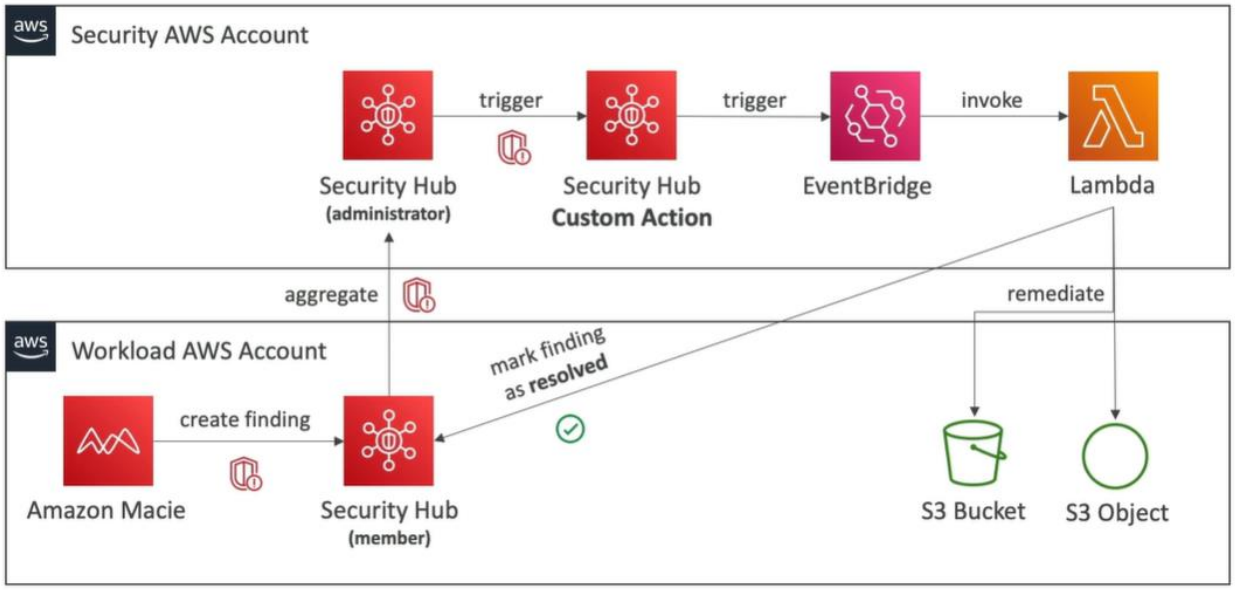
- A comprehensive security service provided by AWS for centralized security management and compliance monitoring
- Collects security findings and alerts from various AWS services, such as GuardDuty, Inspector, and Macie, as well as third-party solutions, providing a unified view of security across the AWS environment
  - Archiving a GuardDuty finding will not update the finding in Security Hub
  - Findings are sent in AWS Security Finding Format (ASFF)

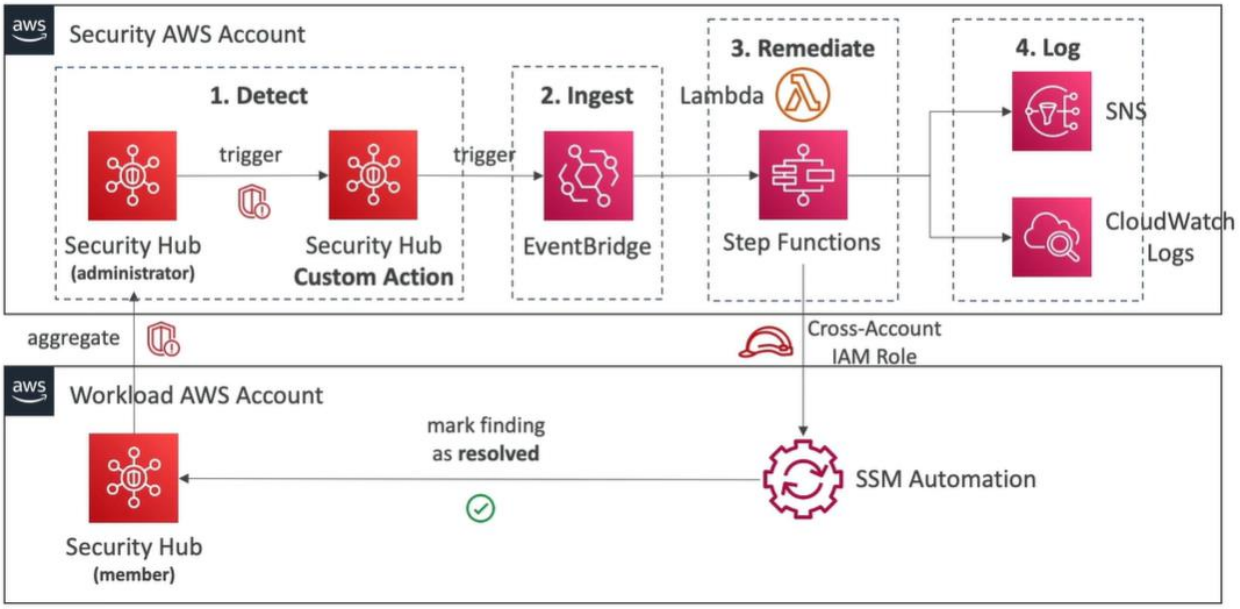


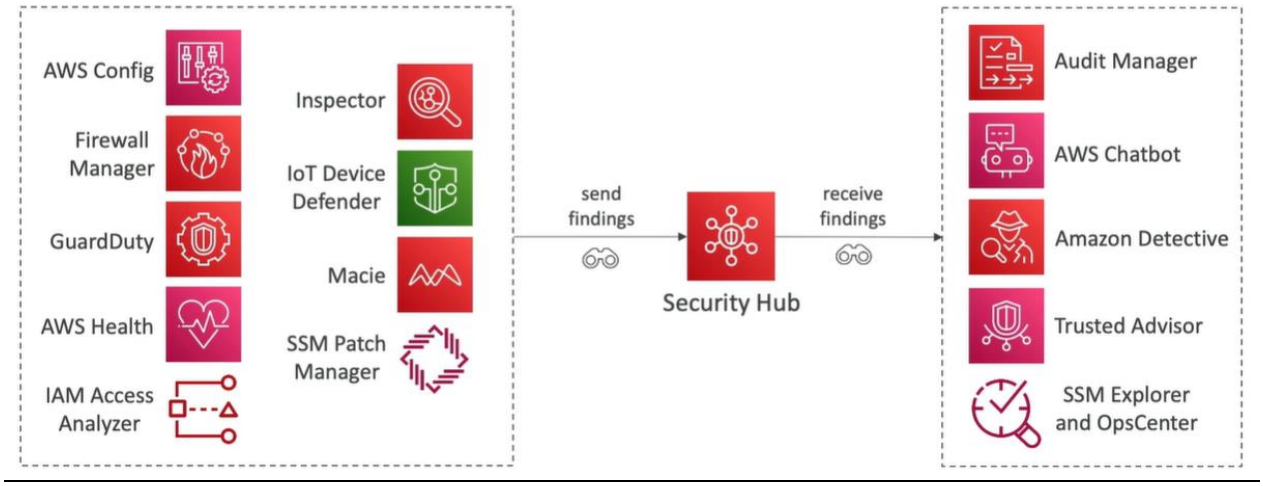
- Continuously monitors resources and accounts for security vulnerabilities, misconfigurations, and potential threats
- AWS Config service is must to be enabled first for Security Hub
- Performs automatic security checks based on industry standards and best practices, such as CIS AWS Foundations Benchmark, PCI-DSS, generating actionable findings and recommendations
- Prioritizes security findings based on severity and provides actionable insights and recommendations to address the identified risks
- Integrates with other AWS services, including CloudTrail, IAM, and S3, as well as third-party security tools
- Offers customizable dashboards and compliance reports, enabling to visualize and track the security posture and compliance status of AWS environment
- Supports automated remediation actions through integration with AWS Systems Manager Automation
- Provides centralized security management across multiple AWS accounts and regions, allowing to gain insights and actions at scale

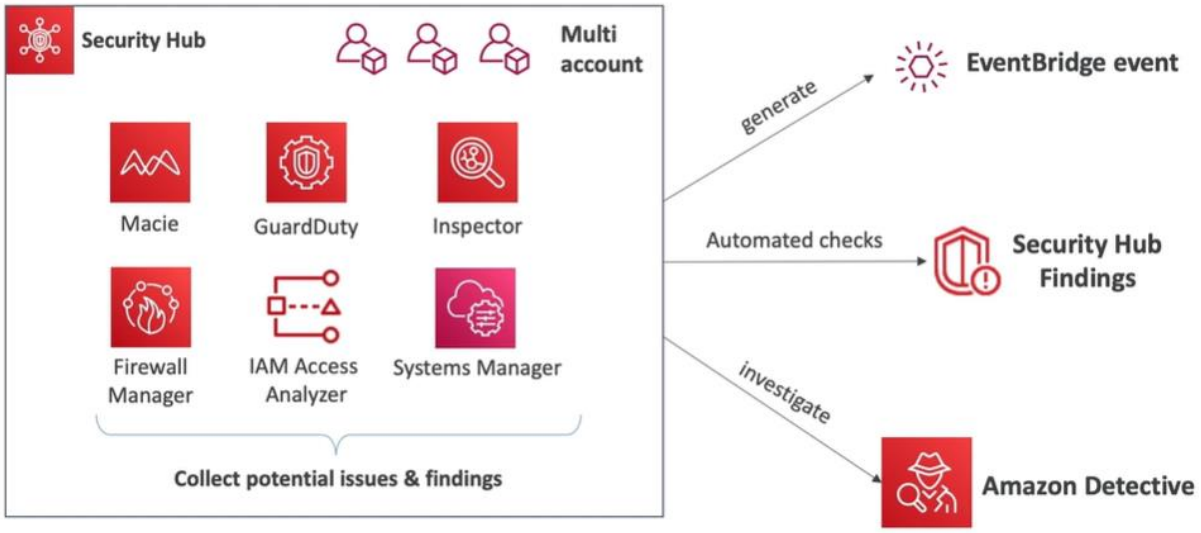


- Offers APIs and event integrations, enabling automation, orchestration, and integration with existing security tools and workflows
- 30-days free trail – regional service – first 10,000 free
- Each finding is an individual issue or a potential security issue – group of findings is Insight
    - Insight – collection of related findings that identifies a security area that requires attention and intervention
    - Brings findings from across finding providers
    - Each Insight defined by a Group By statement and optional filters
- Custom Actions – helps to automate Security Hub with Event Bridge
    - Allows to create actions for response and remediation to selected finding within the Security Hub Console


Security AWS Account
trigger
trigger
invoke
Security Hub
(administrator)
Security Hub
Custom Action
EventBridge
Lambda
aggregate
remediate
Workload AWS Account
mark finding
as resolved
create finding
Amazon Macie
Security Hub
(member)
S3 Bucket
S3 Object


Security AWS Account
1. Detect
2. Ingest
3. Remediate
4. Log
trigger
trigger
Lambda
SNS
CloudWatch
Logs
Security Hub
(administrator)
Security Hub
Custom Action
EventBridge
Step Functions
aggregate
Cross-Account
IAM Role
Workload AWS Account
mark finding
as resolved
SSM Automation
Security Hub
(member)

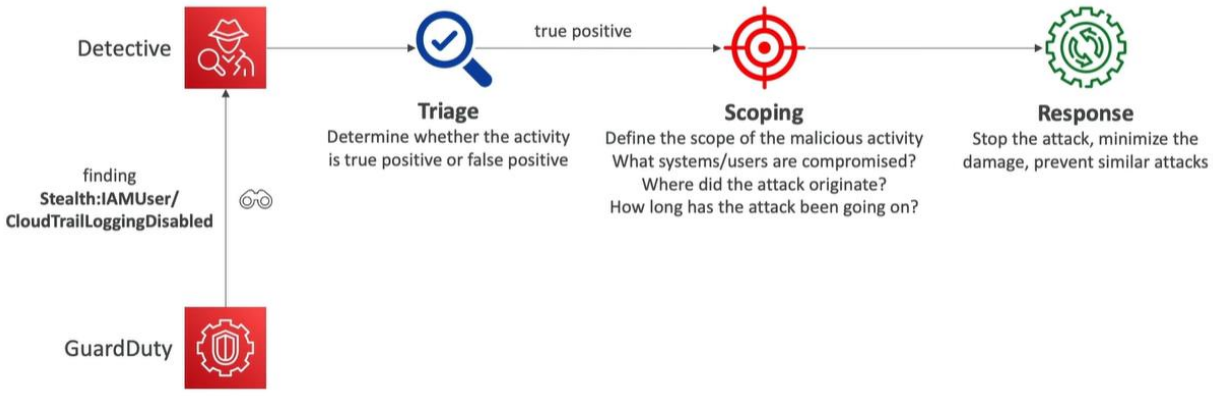




Amazon Detective:

- Provides automated analysis and visualization of log data to help detect and investigate
- Sometimes security findings require deeper analysis to isolate the root cause and take actions – Amazon Detective can help finding it using ML and Graphs
- Uses machine learning algorithms to identify patterns, anomalies, and potential security threats in the log data
- Automatically collects and process events from VPC Flow Logs, CloudTrail, GuardDuty and create a unified view
- Produce visualization with details and context as well as interactive graphs to get to the root cause and then provides actionable insights for remediation
- It generates security findings and recommendations based on the analyzed log data

- It provides a centralized dashboard for monitoring and managing security findings across multiple AWS accounts
- It provides detailed forensic information, including IP address attribution, to aid in the investigation process
- It integrates with other AWS services to enhance the overall security posture of the environment

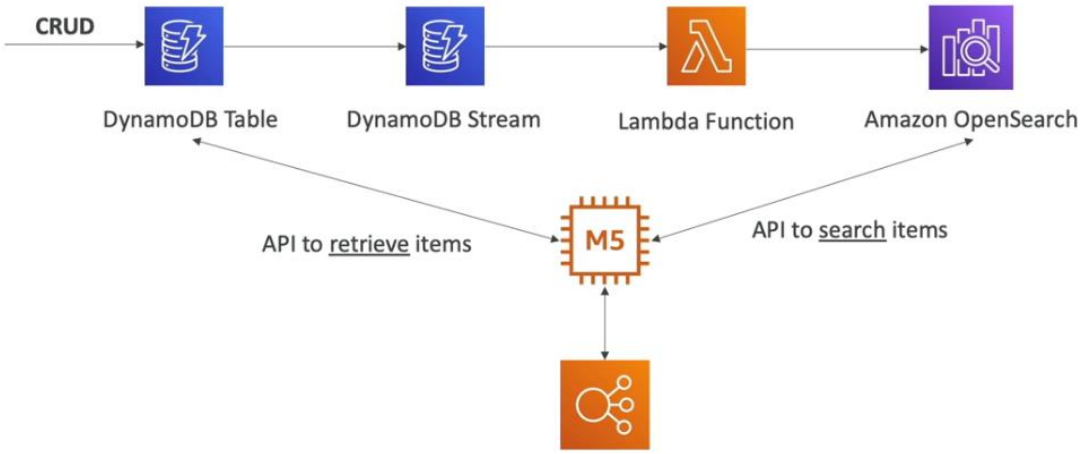


Amazon OpenSearch:

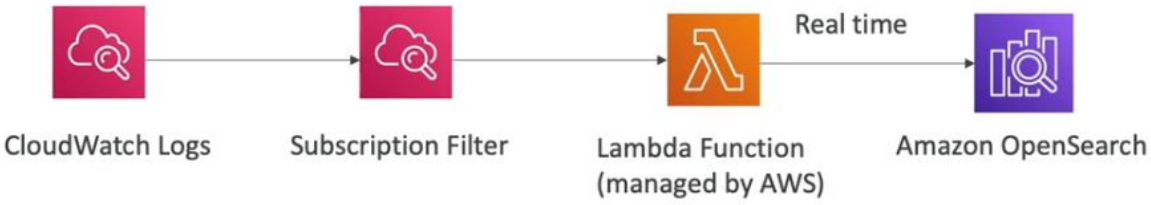
- A fully managed, scalable, and secure search service offered by AWS
- Based on the popular open-source search and analytics engine – Elasticsearch
- Log volume of data can be indexed and searched in near real-time with high performance and reliability
- Provides rich querying capabilities, including full-text search, fuzzy search, and geospatial search
- It supports powerful analytics features – aggregation, histograms, and time series analysis
- It integrates with other AWS services seamlessly:
    - With CloudWatch – for monitoring
    - With IAM – for access control
    - With CloudTrail – for audit logging
- It offers automated backups, automated software patching, and automated scaling to handle fluctuating workloads
- It provides built-in security features such as encryption at rest, access control policies, and integration with IAM
    - Security through Cognito & IAM, KMS encryption, TLS
- Two modes – managed cluster, serverless cluster
- Does not natively support SQL (can be enabled via plugin)
- Ingestion from Kinesis Data Firehose, AWS IoT, and CloudWatch Logs

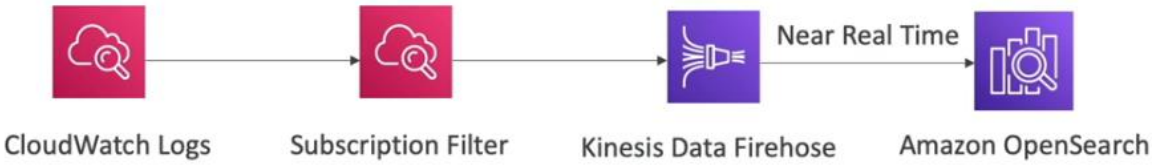
OpenSearch Patterns: (Architectures)

- DynamoDB

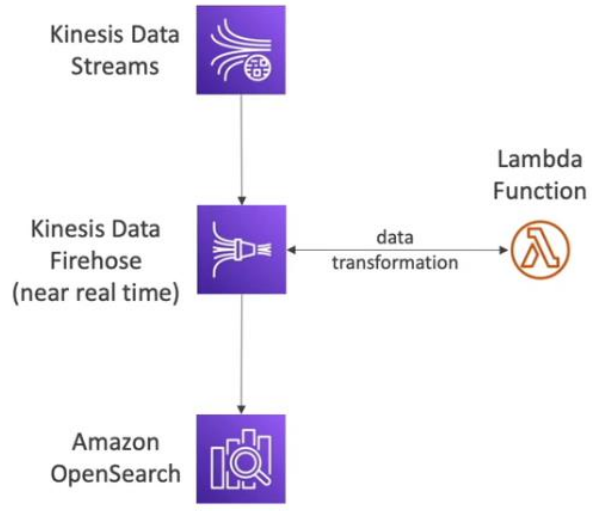


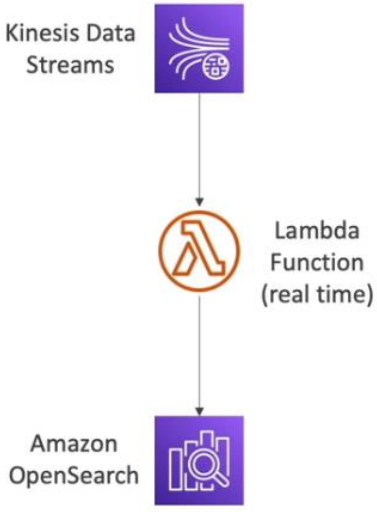
- CloudWatch Logs:





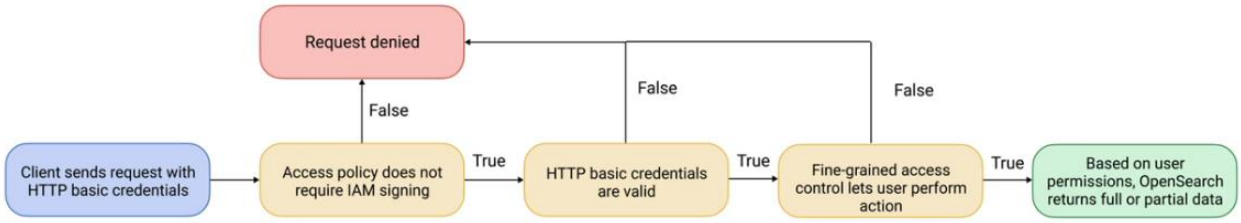
- Kinesis Data Streams & Kinesis Data Firehose:





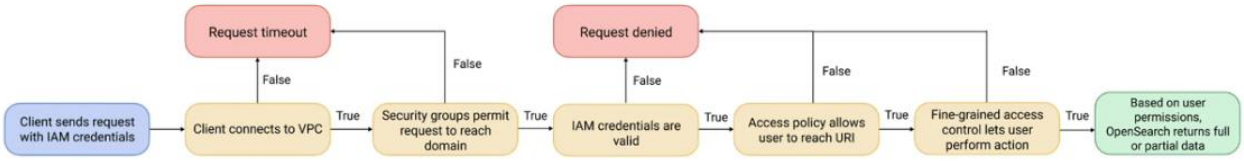
OpenSearch Public Access:

- Accessible from the internet with a public endpoint
- Access is restricted using access policies, identity-based policies, and IP-based policies



OpenSearch VPC Access:

- VPC, Subnets, Security Groups, and IAM Role need to be specified
- VPC Endpoints and ENIs will be created (IAM Role)
- User needs to use VPN, Transit Gateway, managed network, or proxy server to connect to the domain
- Access can be restricted using access policies and identity-based policies



Domain Access Policy:

- Specifies which actions a principal can perform on the domains sub-resources (indexes, APIs etc.)

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

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

AWS Glue:

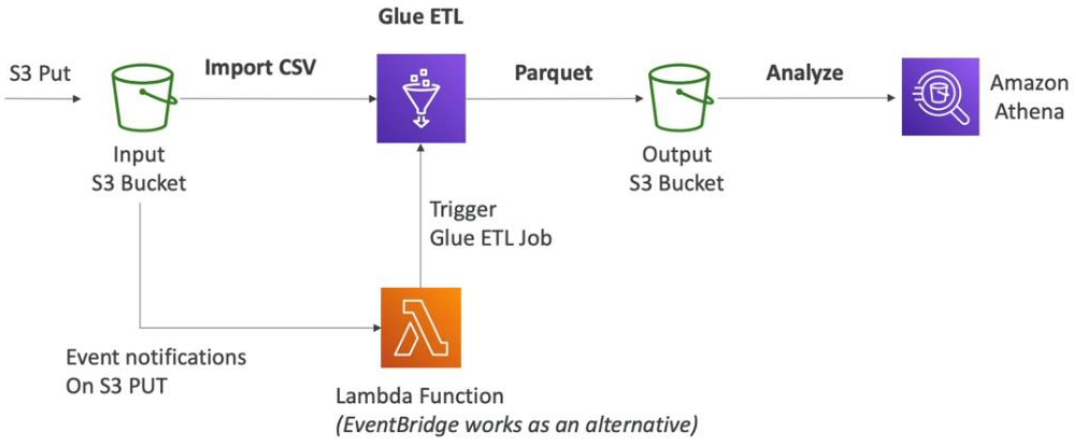
- Fully managed ELT (extract, load and transform) service that makes it easy to prepare and load data for analytics
- Provides a serverless environment for running ELT jobs
- Can automatically discover and catalog data from various sources – databases, data lakes, streaming platforms

- Offers built-in data transformation capabilities – data enriching
- Integrates with other AWS services for data processing and analysis – S3, RedShift, Athena

- Supports various data formats – define data schemas, perform schema evaluation, handle complex data structures
- Can generate code in Python or Scala – can be customized and optimized to meet specific data processing requirements
- Provides features for data lineage and metadata management – governance of data pipelines
- Glue Job Bookmarks – prevent re-processing old data
- Glue Elastic Views – combine and replicate data across multiple data storage using SQL (serverless)
- Glue DataBrew – clean and normalize data using pre-built transformation
- Glue Studio – create, run, and monitor ETL jobs in Glue (new GUI)
- Glue Streaming ETL – built on Apache Spark Structured Streaming (compatible with kafka, kinesis data streaming)

Data Conversion:

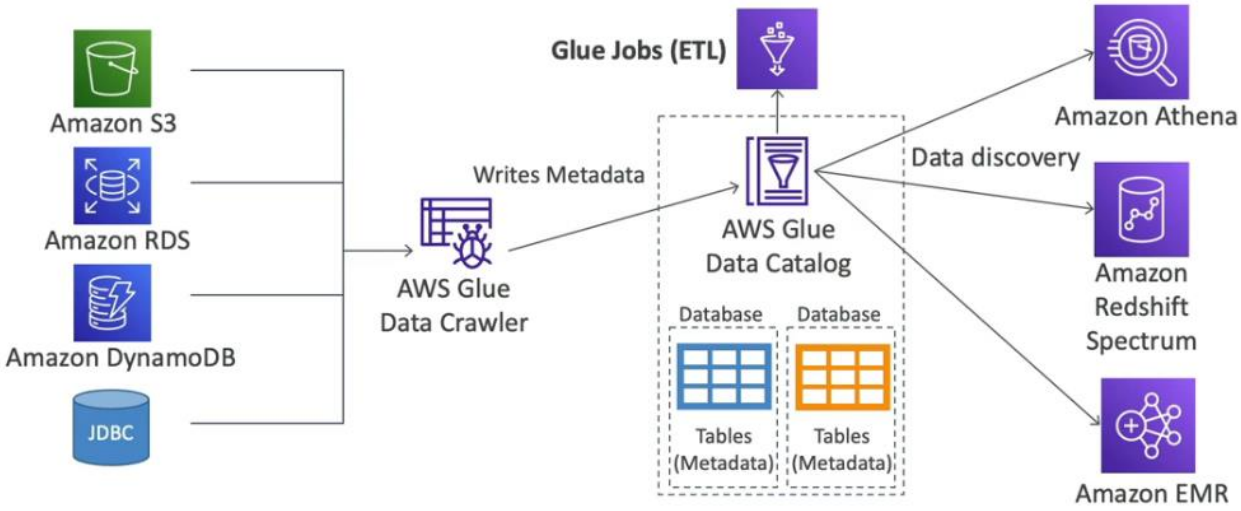
- Convert data into Parquet Format – columnar storage file format that is optimized for big data processing and analytics
    - Designed to efficiently store and query large amount of structured and semi-structured data



Glue Data Catalog:

- Catalog of datasets – acts as a central repository for storing and managing metadata
- Serves as a metadata repository for AWS Glue, as well as other AWS services – Athena, RedShift, AWS Glue ETL jobs

- Provides a unified view of metadata across various data sources and formats
- Metadata – table schemas, column schemas, data types, partitioning info etc.
- AWS Glue Data Crawler - Offers a crawler functionality that automatically discovers and catalogues data from various sources by scanning the data and inferring its structure and schema

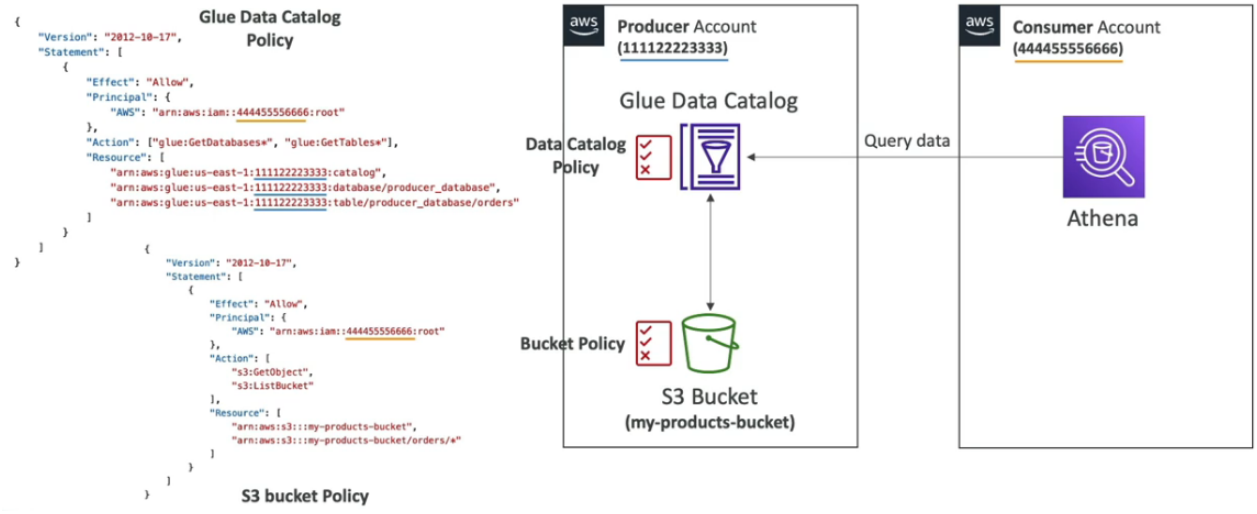


Glue Security:

- Encryption at rest using KMS for databases, tables, Job bookmarks
- Encryption in transit using TLS
- Identity-based policies – grant access to Glue resources by attaching policies to IAM identities

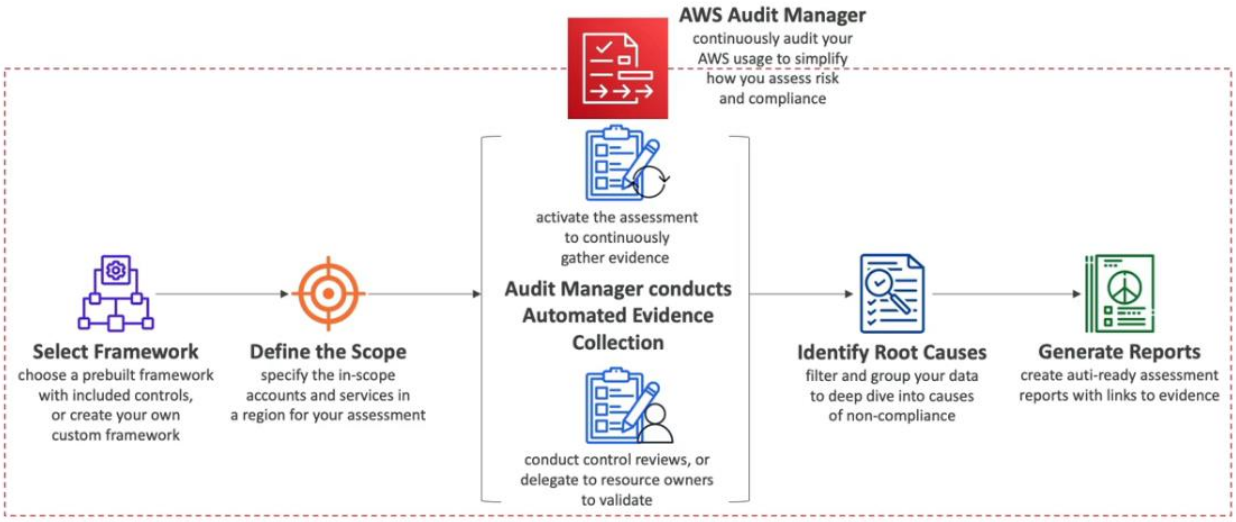
```
{
   "Version": "2012-10-17",
   "Statement": [
     {
       "Effect": "Allow",
       "Action": [
         "glue:GetTable",
         "glue:GetTables",
         "glue:GetDatabase",
         "glue:GetDatabases"
       ],
       "Resource": [
         "arn:aws:glue:us-west-2:123456789012:catalog",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

- Resource-based policies – policy attached to Glue Data Catalog to grant access to IAM identities (used for cross-account access)
- To access Centralized Data Catalog – Glue data catalog policy will be used along with resource-based policy depending on which resource being accessed – e.g., S3 Bucket policy



AWS Audit Manager:

- Helps to automate and streamline the auditing process of AWS resources and compliance frameworks – reduces manual effort
- Generates reports of compliance alongside evidence folders
- Provides built-in frameworks based on industry standards and regulations – PCI DSS, HIPPA, GDPR, and more
- Collects evidence from various AWS services including AWS Config, AWS CloudTrail, IAM, and more to assess the compliance of user's resources
- Allows to define and customize controls to align with user's specific compliance requirements
- Can conduct assessments results, track control implementation progress, and generate reports to demonstrate the compliance to auditors
- Continuously monitor changes to user's AWS resources and automatically updates the assessment results accordingly
- Provides a centralized dashboard where users can view and manage all their assessments, control sets, and findings
- Provides built-in collaboration features, allowing multiple users to collaborate on assessments, share findings, and track remediation actions
- It can integrate with other AWS services to enhance auditing capabilities – AWS Security Hub, AWS Systems Manager, AWS Lambda, AWS Config, Control Tower, AWS CloudTrail, License Manager etc.
- Run over multi-account via integration with AWS Organizations



AWS Artifact:

- A portal that provides access to AWS compliance reports and security documents
- Offers a centralized repository of accessing and downloading important AWS compliance reports, such as SOC reports, PCI DSS Attestations of Compliance, and ISO certifications
- Provides a self-service portal for customers to easily access and download compliance reports, saving time and effort in gathering the necessary documentation for audits and assessment
- Ensures the availability of up-to-date compliance reports, as it automatically updates the repository with the latest updates
- Simplifies the process of verifying AWS's compliance with various industry standards and regulations by providing a consolidated location for obtaining compliance documentation
- Maintains a secure and tamper-proof environment for storing and accessing compliance reports, ensuring the integrity and confidentiality of sensitive information
- Offers an easy-to-use web interface for searching, browsing, and downloading compliance reports

AWS Compliance:

- AWS Compliance encompasses compliance with a wide range of industry standards and regulations, such as ISO 27001, PCI DSS, HIPPA, GDPR, and more
- HIPPA – safeguard and ensure data privacy, security, and integrity of healthcare organizations
- NIST – helps organizations establish effective security controls and practices and provides cybersecurity framework to manage and mitigate risks
- ISO 27001 – sets the requirements for an information security management system and demonstrates a systematic approach to managing sensitive information and mitigating risks
- FedRAMP – provides a standardized approach to security assessment, authorization, and continuous monitoring for cloud services used by U.S government agencies

- PCI DSS – ensures secure handling of cardholder data by merchants and service providers and requires the implementation of specific security controls to protect payment card information
- SAS 70 – assesses the effectiveness of internal controls for service organizations
- SOC1 – assesses the internal controls for financial transactions and reporting
- FISMA – establishes information security standards for federal government agencies and requires agencies to implement security controls and undergo regular assessments
- FIPS 140-2 – specifies requirements for cryptographic modules used to protect sensitive information and validates the security and integrity of cryptographic operations

PCI DSS:

- Install and maintain a firewall configuration to protect cardholders' data
- Do not use vendor-supplied default passwords and security parameters
- Protect stored cardholders' data with encryption
- Restrict access to cardholders' data on a need-to-know basis
- Regularly test security systems and processes
- Maintain a policy that addresses information security for all personal

HIPPA:

- Implement physical safeguards to protect electronic protected health information
- Implement technical safeguards, including access controls and encryption, to protect health information
- Develop and enforce policies and procedures to ensure compliance with HIPPA requirements
- Conduct regular risk assessments and implement measures to mitigate identified risks
- Train employees on HIPPA compliance and their role in protecting health information
- Maintain documentation of HIPPA compliance efforts, including policies and procedures and incident response plans

ISO 27001:

- Establish a management framework to define the scope and objectives of information security management system
- Conduct a systematic risk assessment and implement appropriate controls to mitigate identified risks
- Develop and enforce security policies and procedures to address information security requirements
- Continuously monitor, measure, and evaluate the effectiveness of information security management system
- Conduct internal audit to assess compliance with ISO 27001 requirements
- Plan implement corrective actions to address identified non-conformities

AWS Acceptable Use Policy – AUP

- Governs user's use of the service offered by AWS
- User may not use for:
    - Illegal or fraudulent activity
    - Violate the rights of others


  - Threaten, terrorism, violence, or other serious harm
  - Child sexual abuse content or activity
  - Violate the security, integrity, or availability for other networks and computers
  - Distribute, publish, or facilitate the unsolicited mass emails (spams)
- AWS will remove or disable any content that violates this policy

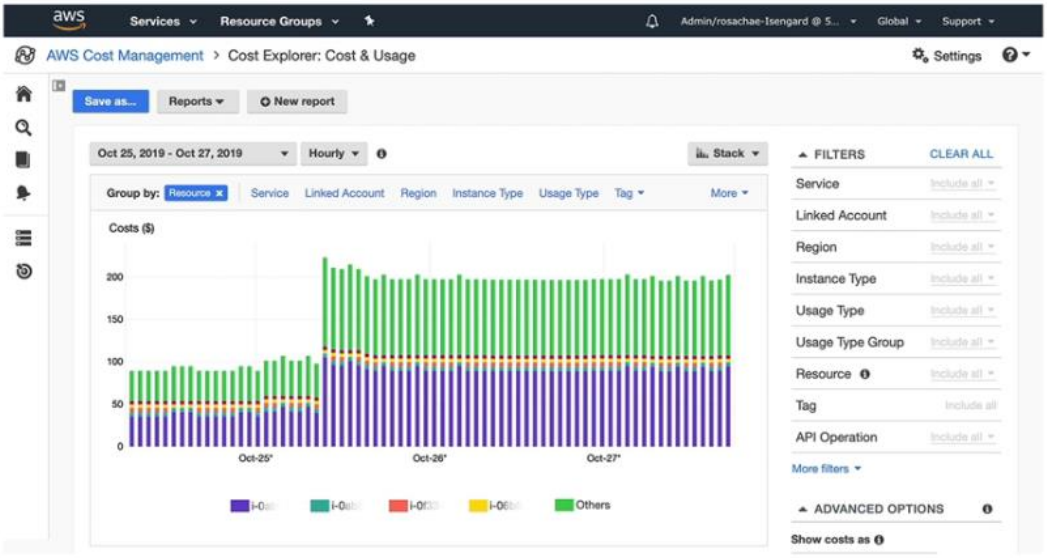
AWS Abuse Report

- To report if AWS resources are used for illegal or abusive purposes
  - Spam
  - Port scanning
  - DDoS attacks
  - Intrusion attempts
  - Hosting prohibiting content
  - Distributing malware
- If you receive an email that your AWS resources are used for illegal activity:
  - Respond to the email and explain how you're preventing this
  - If you don't respond within 24 hours, AWS might suspend your AWS account

AWS Cost Explorer:

- A tool that provides insights into user's AWS usage and costs
- Allows to analyze and visualize user's AWS spending patterns over time – total costs and usage across all accounts
- Offers preconfigured reports and customizable filters to help user understand cost drivers and identify the areas of cost optimization
- Provides cost forecasts to estimate future spending based on historical data
- Supports cost allocation tags to categorize and track costs by applications, departments, or other dimensions
- Integrates with AWS Budgets and AWS Cost Anomaly Detection to provide advanced cost monitoring and anomaly detection capabilities
- Helps users optimize their AWS costs by identifying opportunities for rightsizing, Reserved instance purchases, and eliminating idle resources
- Monthly Cost by AWS services – example dashboard

- Hourly & Resource level – example dashboard



- Forecast usage – example dashboard

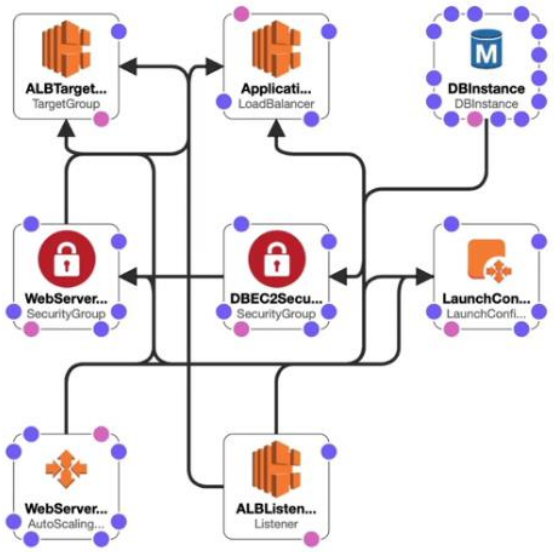
AWS Cost Anomaly Detection:

- A service that automatically identifies and notifies of unusual spending patterns in AWS account
- Uses machine learning algorithms to analyze user's historical cost and usage data
- It learns user's unique, historic spend patterns to detect one-time cost spike and/or continuous cost increase – user doesn't need to define threshold
- Then it detects anomalies that deviate significantly from user's expected spending pattern
- Helps to identify cost spikes, unexpected changes in usage, or unusual resource consumption
- It provides timely alerts and recommendations to investigate and address potential cost issues
- Helps users proactively manage their AWS costs and prevent unexpected financial surprises
- Monitor AWS services, member accounts, cost allocation tags, or cost categories
- Sends user the anomaly detection report with root-cause analysis
- User can get notified with individual alerts or daily/weekly summary – using SNS

AWS CloudFormation:

- A service that allows to define and provision AWS infrastructure resources in a declarative template format

- Provides a way to create, update, and delete resources as a single, consistent unit – stack
    - The CloudFormation will create in the right order, with the exact configuration that user specifies
    - Simplifies and automates the process of deploying and managing infrastructure
- Supports wide range of AWS resource types – most of them are supported
- Templates can be written in JSON or YAML format and can be stored in version control systems for easy management
- It handles resource dependencies – ensures that resources are created and updated in the correct order
- CloudFormation stacks can be created and managed using the Console, CLI or SDK
- Supports infrastructure-as-code principles
    - No resources are manually created – excellent for control
    - Changes to the infrastructure are reviewed through code
- Cost – each resource within the stack is tagged with an identifier so we can easily see how much a stack cost us
    - We can estimate the cost of our resources using the CloudFormation template – saving strategies
- Productivity – ability to destroy and re-create an infrastructure on the cloud on the fly
    - Automate generation of Diagram for user's templates
    - Declarative programming – no need to figure out ordering and orchestration
- Don't need to re-invent the wheel
    - Leverage existing templates on the web
    - Leverage the documentation
- Example – WordPress CloudFormation Stack



CloudFomration Drift:

- CloudFormation Drift detection is a feature that helps to identify configuration changes made to resources managed by CloudFormation stacks

  - Configuration updates
  - Resources replacements
- Compares the current state of resources within their expected state defined in the CloudFomration template
- Actions can be taken to reconcile the drift by updating the stack or resources to bring them back into desired state
- Supports drift detection for nested stacks – identify changes at different levels of infrastructure

Stack Termination Protection:

- It prevents accidental or undesired/unauthorized termination of CloudFormation stacks – provides an additional layer of control and safety
- When termination protection is enabled, user cannot delete the stack using CloudFormation APIs or the AWS Management Console
- To delete a stack with termination protection enabled, protection needs to be disabled first and then proceed with deletion

Stack Policies:

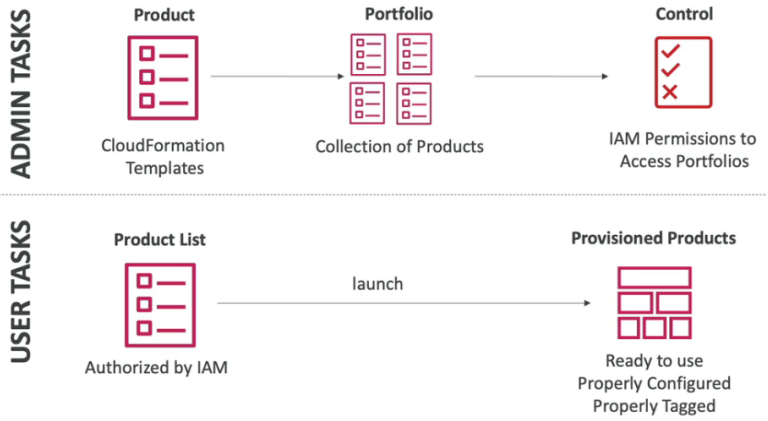
- To control the updates that can be made to resources within stack
- Allows to define fine-grained permissions and restrictions on resource modification, deletion, or replacement
- Can explicitly specify which resources can be update, which properties are immutable and the actions that are allowed or denied
- Can be used to restrict modifications based on resource types, logical IDs, or specific property value

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "Update:*",
            "Principal": "*",
            "Resource": "*"
        },
        {
            "Effect": "Deny",
            "Action": "Update:*",
            "Principal": "*",
            "Resource": "LogicalResourceId/CriticalSecurityGroup"
        },
        {
            "Effect" : "Deny",
            "Action" : "Update:*",
            "Principal": "*",
            "Resource" : "*",
            "Condition" : {
              "StringEquals" : {
                "ResourceType" : ["AWS::RDS::DBInstance"]
              }
            }
        }
    ]
}
```

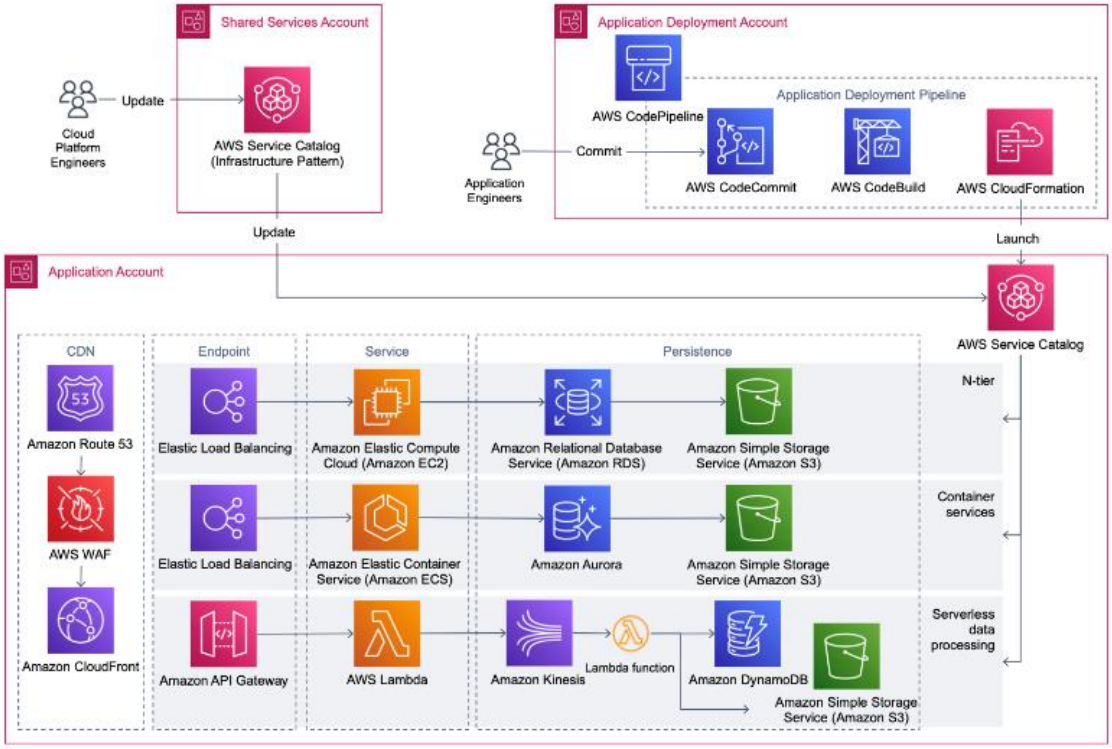
- Stack policies are cumulative – multiple policies can be applied on stack and the resulting permissions and restrictions are combined
- Changes that violate the Stack Policy will be prevented and result in a stack update failure
- We cannot specify a user in the principal clause of the stack policy

AWS Service Catalog:

- Enables organizations to create and manage catalogs of approved IT services on AWS
- Administrators can define and publish products, including AWS resources, software, and services to a centralized catalog
- Users can browse the catalog and launch approved products
- Helps enforce compliance, control cost

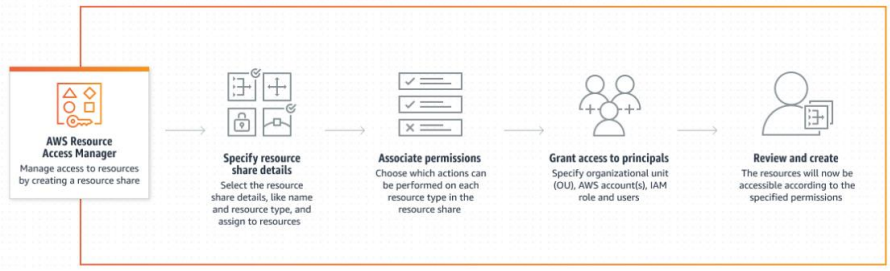


- A little complicated architecture diagram explaining the use case of AWS Service Catalog

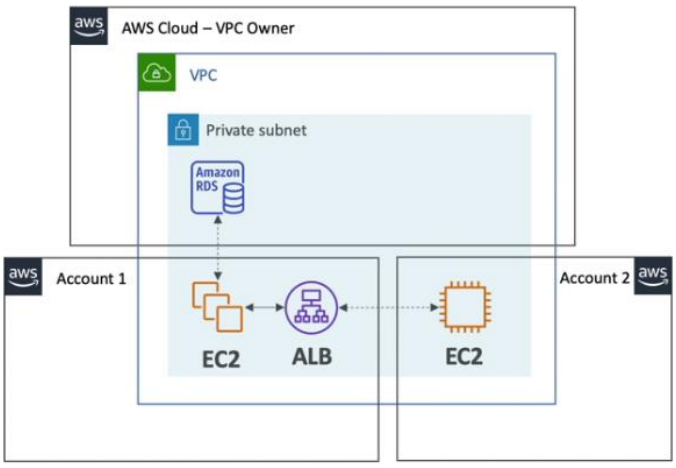


AWS RAM:

- Resource Access Manager – allows users to share AWS resources that they own with other AWS accounts
- Enables resource sharing within Organization or with specific accounts outside of Organization
    - If integrated with AWS Organizations
    - Provides centralized control and visibility over shared resources
- Simplifies resource management – reduces the need for resource duplication



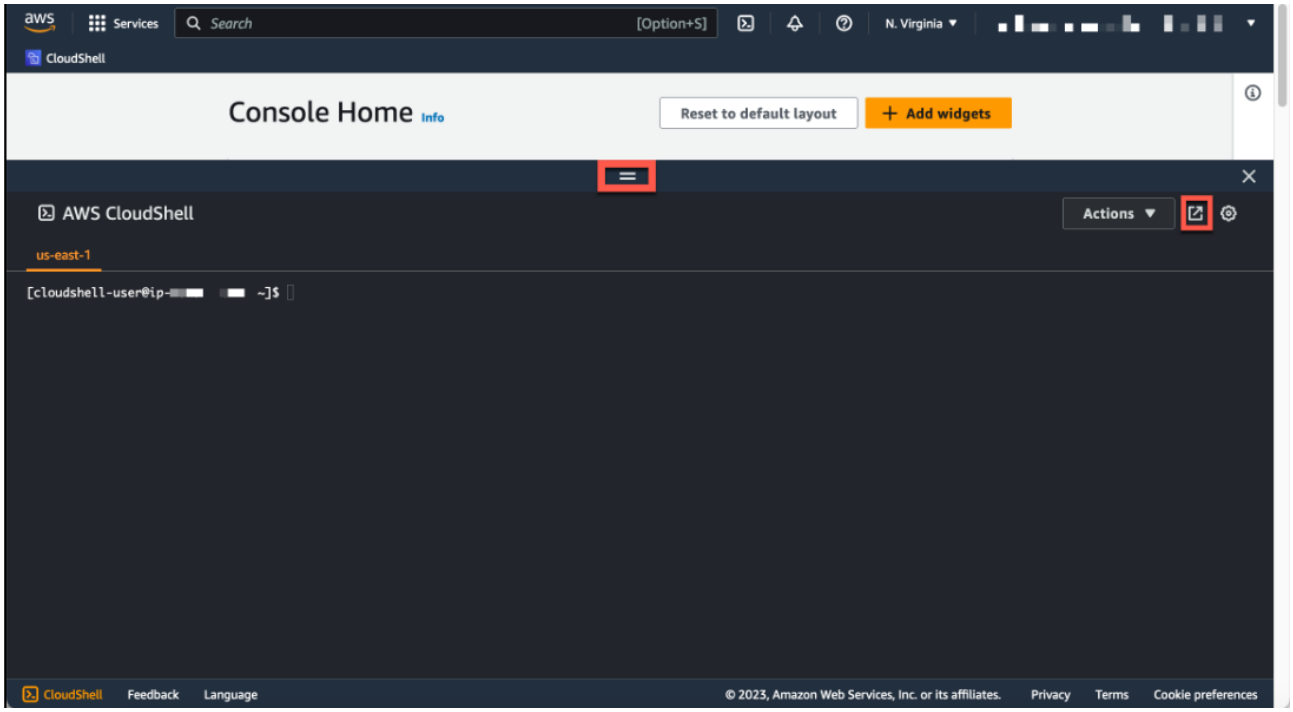
- A lot of resources can be shared like:
    - VPC subnets – allow to have all the resources launched in the same subnets
        - Must be from same AWS Organizations
        - Cannot share security groups and default VPC
        - Security groups from other accounts can be referenced
        - Participants can manage their own resources in there
        - Participants can't view, modify, delete resources that belong to other participants or owner
        - Anything deployed in VPC can talk to other resources in the VPC
        - Applications are accessed easily across accounts – using Private IP



    - Route53 Resolver Rules
    - AWS Transit Gateway
    - License Manager Configurations and many more

AWS CloudShell:

- A browser-based pre-authenticated shell that can be launched directly from AWS Management Console
- Offers a CLI directly within the AWS Management Console
- Eliminates the need for local installation and configuration of CLI tools on user machines
- Run AWS CLI commands and run scripts without leaving browser
- Pre-installed common libraries and common command-line tools
    - AWS CLI, ECS CLI, AWS SDKs for Python and Node.js
    - Bash, PowerShell, zsh, vi, git, npm, pip etc.
- Pre-configured AWS credentials – inherit from the logged-in user
- Persistent storage with up to 1 GB per AWS region
- Runs on Amazon Linux 2
- Users can customize to their exact preference – screen layouts, display text size, theme etc.
- Session restores functionality – resumes the session until the shell is stopped
- No additional charge



CloudShell Security:

- Protected by specific security features
    - Permissions management with IAM – grant and deny permissions to AWS CloudShell users using IAM policies


```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "CloudShellUser",
        "Effect": "Allow",
        "Action": [
            "cloudshell:*"
        ],
        "Resource": "*"
    }, {
        "Sid": "DenyUploadDownload",
        "Effect": "Deny",
        "Action": [
            "cloudshell:GetFileDownloadUrls",
            "cloudshell:GetFileUploadUrls"
        ],
        "Resource": "*"
    }]
}
```

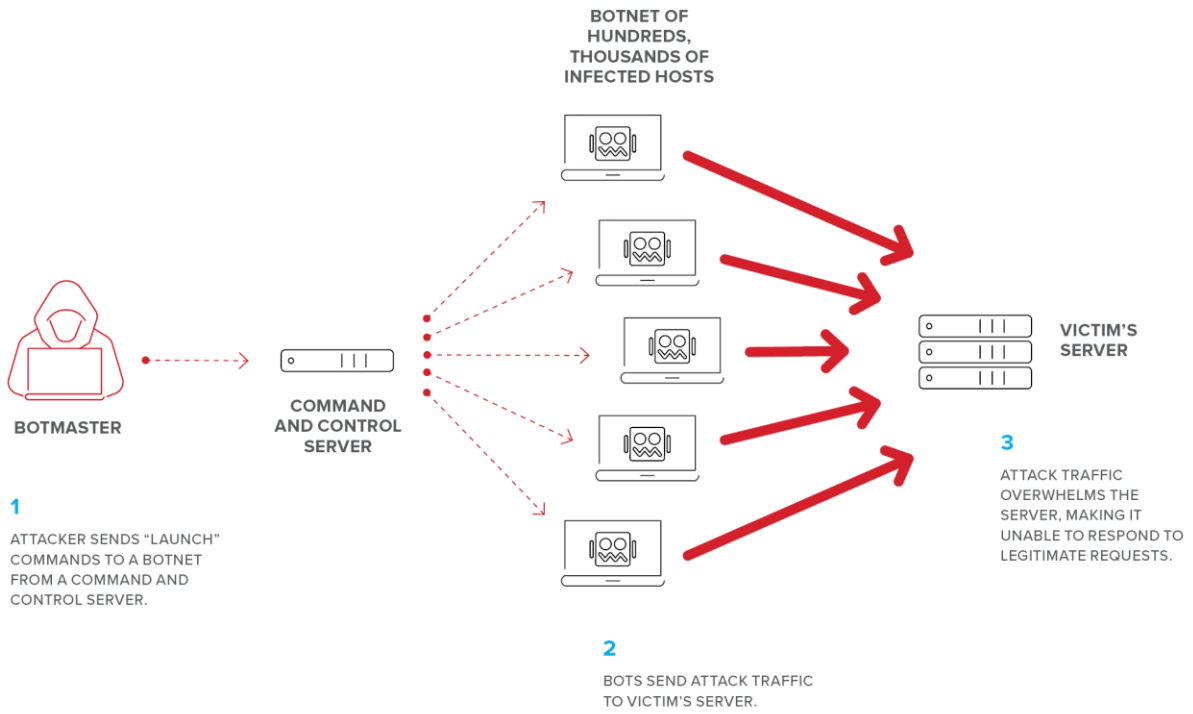
- Shell session management – automatically stopping inactive and long running sessions
- Safe paste for text input – enabled by default
    - Verifies that the multiline text that you want to paste into the shell doesn't contain malicious scripts

Attacks & Mitigations:

Attacks:

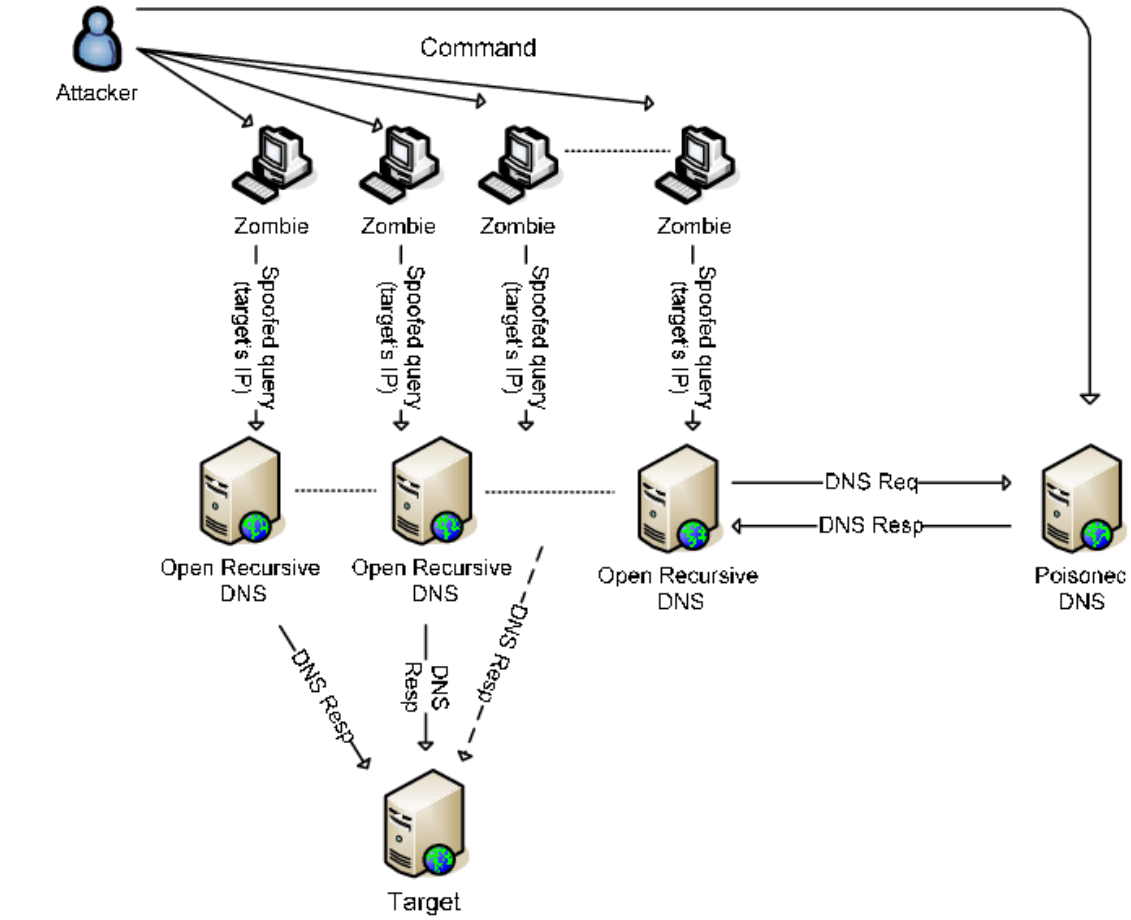
DDoS Attacks:

- DDoS attack involve overwhelming a target system with a massive amount of traffic or requests, making it unable to respond to legitimate users
- The attack is distributed because it uses multiple compromised computers or devices, knows as a botnet, to flood the target
- The compromised devices are often infected with malware that allows them to be controlled remotely by the attacker, forming a network of “zombie” devices
- The attacker typically hides their identity by using techniques such as IP address spoofing or routing the attack through multiple proxy servers
- There are different types of DDoS attacks including the volumetric attacks, which flood the network with a high volume of traffic; TCP/IP protocol attacks, which exploit the vulnerabilities in the network protocols; and application-layer attacks, which target specific applications or services
- DDoS attack can disrupt or completely shut down targeted websites or online services, causing financial loss, reputational damage, and inconvenience to users



Amplification/Reflection Attacks:

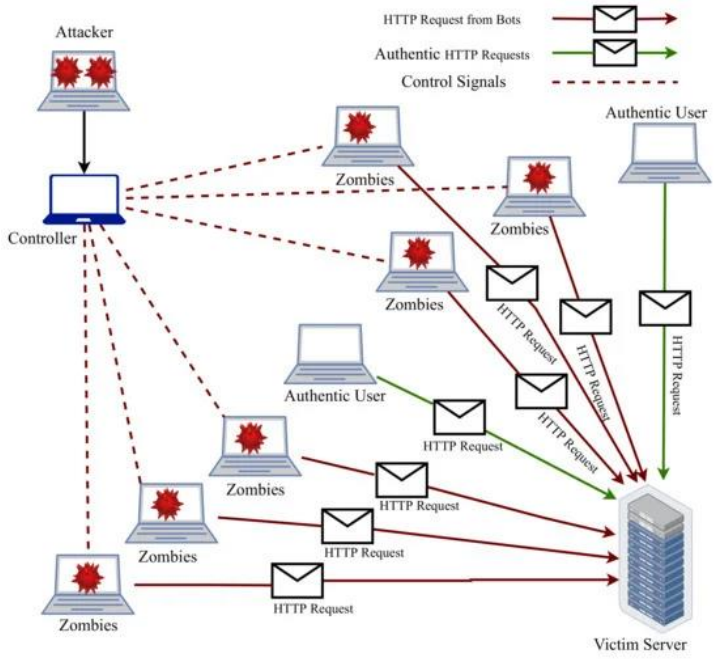
- Utilize legitimate servers or devices unwitting amplifiers to generate a large volume of traffic towards the target
- The attacker spoofs the source IP address or their requests to appear as the victim's IP address
- The attacker sends a small request to vulnerable server or device, such as a DNS server or NTP server, with the victim's IP address as the source
- The vulnerable server or device responds to the request by sending a much larger response to the victim's IP address
- The amplification effect occurs because the response from the vulnerable server is significantly larger than the original request
- The attacker can launch a relatively small number of requests, but the amplification factor can result in a substantial volume of traffic overwhelming the target
- Amplification/reflection attacks can achieve high amplification ratios, ranging from several times to hundreds or even thousands of times in original request size
- These attacks exploit weaknesses in protocol like DNS, NTP, SNMP, SSDP, and others that allow the use of spoofed IP addresses and generate large responses



- NTP Amplification – a DDoS attack type that exploits the vulnerable NTP servers to generate a large volume of traffic towards the target
    - Attackers spoof the victim's IP address and send small request to NTP servers, which respond with larger responses, causing traffic amplification
    - The amplification traffic overwhelms the target, causing disruption to its normal functioning

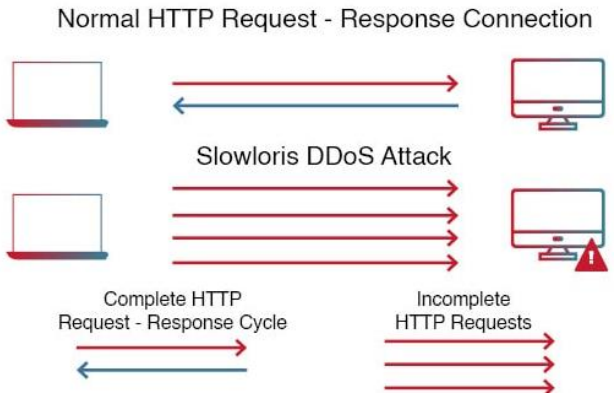
Flood of Get Requests Attacks:

- The attacker uses a botnet or multiple compromised devices to send a high volume of HTTP GET requests to the target server
- Each GET request is a simple HTTP request sent to a specific URL or resource on the server, typically without any legitimate purpose
- The goal is to exhaust the server's resources, such as CPU, memory, or network bandwidth, by overwhelming it with a flood of simultaneous GET requests
- The attacker often spoofs the source IP addresses of the requests to make tracing and filtering more challenging
- The flood of GET requests consumes the server's processing power and network resources, making it unable to respond to legitimate user requests



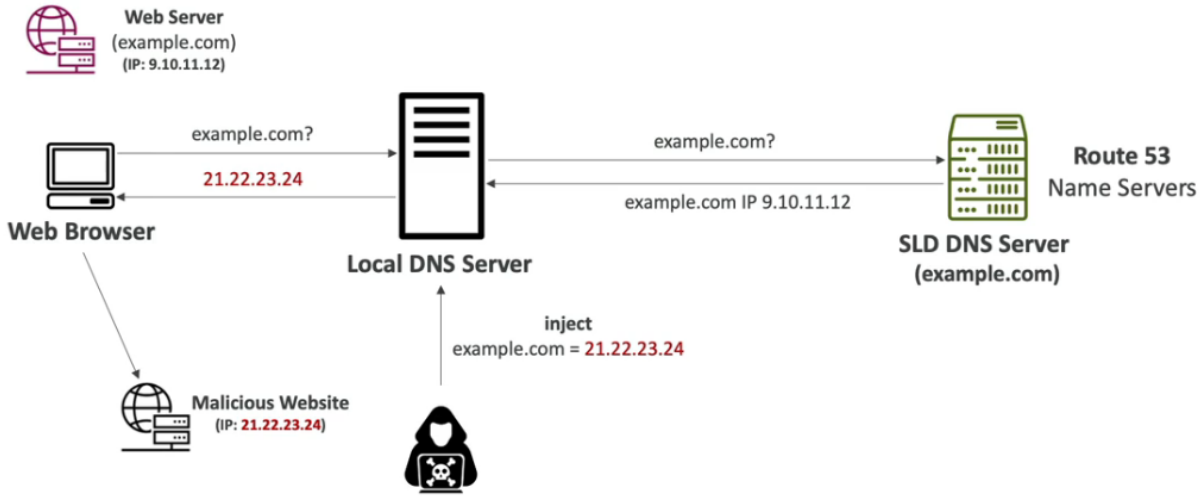
Slowloris Attack:

- In a Slowloris attack, the attacker establishes multiple connections to the target web server, gradually sending incomplete HTTP requests
- The attacker purposefully keeps those connections open by periodically sending small chunks of data, preventing the server from closing them
- By keeping connections open for an extended period, the attacker exhausts the server's maximum concurrent connections limit, effectively denying service to legitimate users
- Slowloris attack exploit the server's resource allocation, as each connection consumes server resources such as memory and thread capacity
- The attack is called "Slowloris" because it simulates a slow-moving, persistent attack, similar to slow movement of sloth
- Unlike traditional DDoS attacks that flood with high volume of traffic, Slowloris attacks are more focused on exhausting server resources



DNS Poisoning (spoofing):

- Form of a cyber attack that manipulates the DNS resolution process to redirect users to malicious websites
    - Attacker manipulates DNS records to associate legitimate domain names with incorrect IP address
    - Users are unknowingly redirected to fraudulent websites, often resembling the legitimate ones



Mitigations:

Generic:

- Minimize the attack surface area – limit the number of potential entry points
    - Utilize AWS VPC to segment and isolate resources into separate subnets
    - Configure AWS Security Groups to restrict inbound and outbound traffic based on specific ports, protocols, and IP ranges
    - By minimizing the attack surface area, user can limit the exposure to just a few hardened entry points
- Be ready to scale to absorb the attack – these measures can help the infrastructure dynamically scale and absorb the impact of DDoS attack minimizing the attack's success
    - Design the infrastructure to scale as, and when, it is needed
    - Scaling can be horizontally or vertically
        - The attack is spread over the larger area
        - Attackers then have to counterattack, taking up more of their resources
        - Scaling can help buy more time to analyze the attack and to respond with the appropriate countermeasures
        - Scaling has the added benefit of providing with additional levels of redundancy
- Safeguard exposed resources – need to take additional measures to restrict access and protect those entry points without interrupting legitimate end user traffic
    - Three resources can provide this control and flexibility:

- Amazon CloudFront – by distributing content across a global network of edge locations, it can absorb DDoS traffic closer to the source reducing the impact on origin servers
  - Geographic Restriction/Blocking – whitelists/blacklists
  - Origin Access Identity – restrict access to S3 bucket so that people can only access S3 using CloudFront URLs
- Amazon Route53 – can implement rate limiting, which restricts the number of requests from a particular IP address, or a specific geographic location and it also integrates with AWS Shield
  - Alias Record Sets – to immediately redirect traffic to CloudFront distribution, or to a different ELB with higher capacity – No DNS change
  - Private DNS – manage internal DNS names for application resources without exposing to the public internet
- WAFs – by creating rules and conditions, WAF can be configured to filter out DDoS attack traffic before it reaches to the application, and it works in conjunction with Route53 and CloudFront to provide comprehensive protection

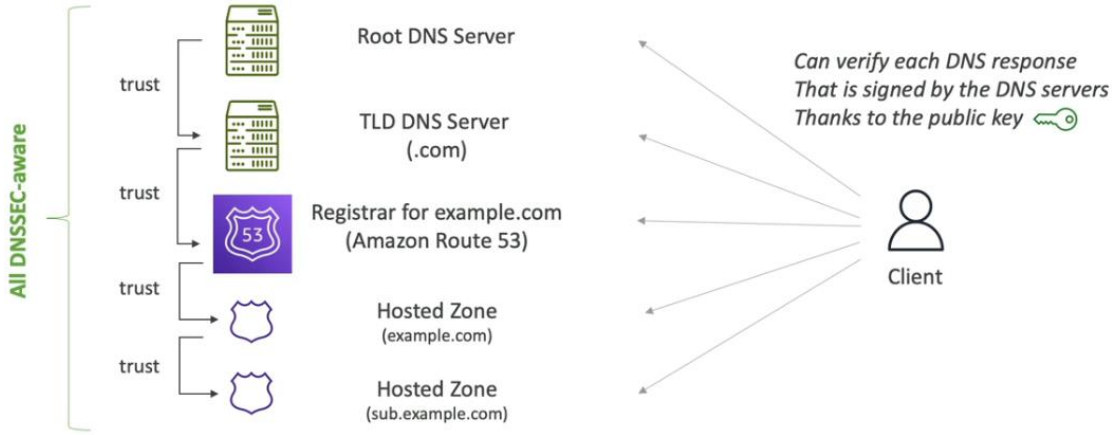
- Learn normal behavior
  - Establish a baseline of normal behavior for network and application traffic
  - Monitor traffic patterns and identify significant deviations from the baseline
  - Detect and understand expected and unexpected spikes in incoming or outgoing traffic
  - Identify abnormal connection durations or patterns
  - Trigger alerts or mitigation actions when anomalies are detected
  - Implement rate limiting, traffic filtering, or activation of DDoS protection services
  - Apply access controls and secure vulnerable services
- Create a plan for attacks
  - Validate architecture design for security and resilience
  - Understand costs and techniques for increased resiliency
  - Establish clear communication channels and contacts
  - Enable faster response and mitigation
  - Minimize downtime and expedite recovery
  - Enable continuous improvement and adaptation to be evolving attacks

DNS Poisoning:

- Route53 – DNSSEC (DNS Security Extension)
  - A protocol for securing DNS traffic, verifies DNS data integrity and origin
  - Works only with Public Hosted Zones
  - Route53 supports both DNSSEC for Domain Registration and Signing
- DNSSEC Signing – validate that DNS response came from Route53 and has not been tempered with
  - Route53 cryptographically signs each record in the Hosted Zone
  - Two keys:
    - Managed by User – Key-signing Key (KSK) – based on asymmetric CMK in AWS KMS

- Managed by AWS – Zone-signing Key (ZSK)

- When enabled, Route 53 enforces a TTL of one week for all records in the Hosted Zone
  - Records that have TTL less than one week are not affected
- Steps to enable DNSSEC on a hosted zone
  - Step1 – prepare for DNSSEC signing
    - Monitor zone availability – through customer feedback
    - Lower TTL for records – recommended 1 hour
    - Lower SOA minimum for 5 minutes
  - Step2 – enable DNSSEC signing and create a KSK
    - Enable DNSSEC in Route53 for the hosted zone – Console or CLI
    - Make Route53 create a KSK in the console and link it to a customer managed CMK
  - Step3 – establish a chain of trust
    - Create a chain of trust between the hosted zone and the parent hosted zone
    - By creating a Delegation Signer (DS) record in the parent zone
    - It contains a hash of the public key used to sign DNS records



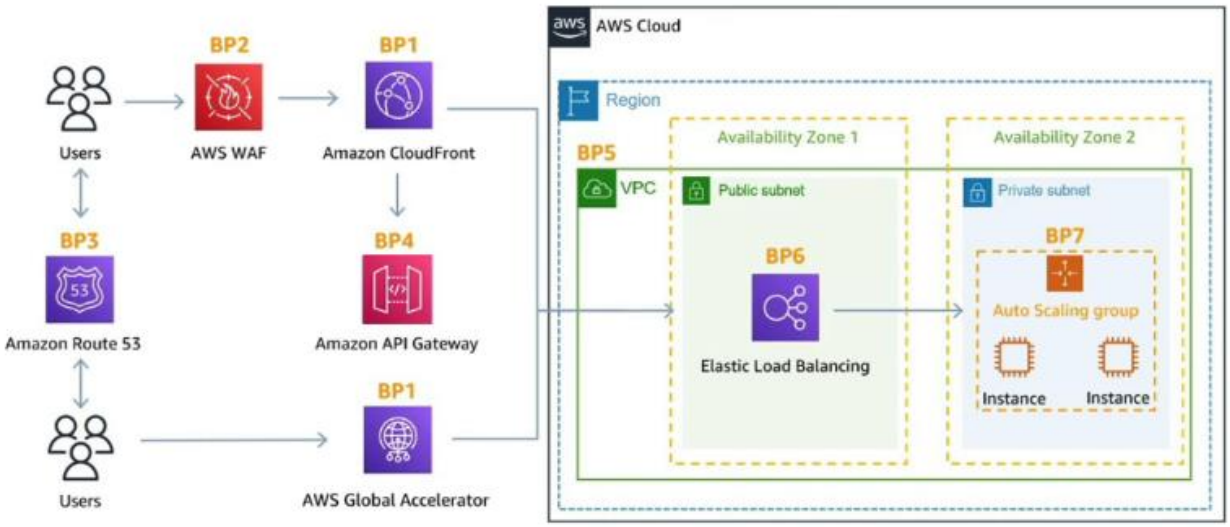
  - Step4 – monitor for errors using CloudWatch Alarms
    - Create CloudWatch alarms for *DNSSECInternalFailure* and *DNSSECKeySigningKeysNeedingAction*

DDoS:

- Technologies can be used to mitigate DDoS attack:
  - CloudFront – acts as distributed CDN to absorb and distribute traffic
  - Route53 – offers intelligent DNS routing and traffic management, redirecting and filtering traffic
  - ELB's – distributes traffic across multiple instances, scaling resources dynamically
  - WAFs – inspect and filter HTTP requests, blocking malicious traffic and known attack patterns
  - Autoscaling – automatically adjust resources capacity based on demand to handle increased traffic and distributing the load

- CloudWatch – providing real-time visibility into traffic patterns, monitor and analyze AWS services
- API Gateway – leveraging features such as rate limiting, throttling, and integration with WAF to filter
- Shield – managed DDoS protection service
- Global Accelerator – providing advanced traffic routing

- AWS best practices for DDoS resiliency Edge Locations Mitigations – see the below diagram to understand the below explanation
  - CloudFront – BPI
    - Web application delivery at the edge
    - Protection from DDoS common attacks – SYN floods, UDP reflection etc.
  - Global Accelerator – BPI
    - Access application from the edge
    - Integration with Shield for DDoS protection
    - Helpful if backend is not compatible with CloudFront
  - Route 53 – BP3
    - Domain name resolution at the edge
    - DDoS protection mechanism on DNS
  - Infrastructure layer defense – BPI, BP3, BP6
    - Protect Amazon EC2 against high traffic
    - That includes using Global Accelerator, Route53, CloudFront, ELB



  - EC2 with Auto Scaling – BP7
    - Helps scale in case of sudden traffic surges including a flash crowd or a DDoS attack
  - Elastic Load Balancing – BP6
    - ELB scales with the traffic increases and will distribute the traffic to many EC2 instances
  - Detect and filter malicious web requests – BPI, BP2

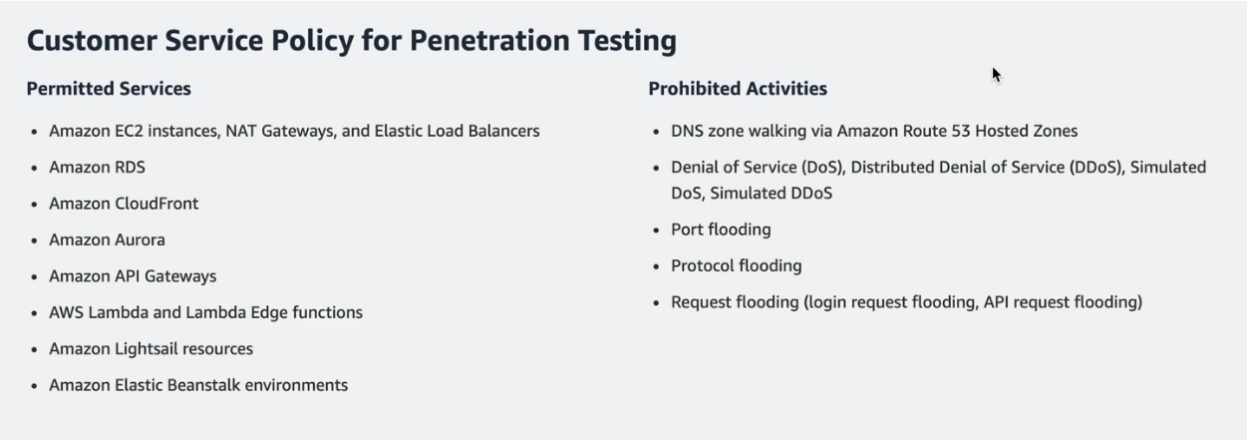
- CloudFront cache static content and serve it from edge locations, protecting backend
- AWS WAF is used on top of CloudFront and ALB to filter and block requests based on request signatures
- WAF rate-based rules can automatically block the IPs of bad actors
- Use managed rules on WAF to block attacks based on IP reputation, or block anonymous IPs
- CloudFront can block specific geographies

- Shield Advanced – BPI, BP2, BP6
  - Shield Advanced automatic application layer DDoS mitigation automatically creates, evaluates, and deploy AWS WAF rules to mitigate layer 7 attacks
- Obfuscating AWS resources – BPI, BP4, BP6
  - Using CloudFront, API Gateway, ELB to hide backend resources (Lambda functions, EC2 instances)
- Security Groups and NACLs – BP5
  - Use security groups and NACLs to filter traffic based on specific IP at the subnet or ENI level
  - Elastic IPs are protected by AWS Shield Advanced
- Protecting API endpoints – BP4
  - Hide EC2, Lambda, elsewhere
  - Edge-optimized mode or CloudFront + regional mode – more control for DDoS
  - WAF + API Gateway – bursts limits, header filtering, use API keys

Security Assessment & Attacks Simulation

AWS Penetration Testing:

- AWS has specific policies and guidelines regarding penetration testing to ensure the safety and integrity of their services – below are the details of permitted and prohibited activities



Customer Service Policy for Penetration Testing

Permitted Services

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

Prohibited Activities

- DNS zone walking via Amazon Route 53 Hosted Zones
- Denial of Service (DoS), Distributed Denial of Service (DDoS), Simulated DoS, Simulated DDoS
- Port flooding
- Protocol flooding
- Request flooding (login request flooding, API request flooding)

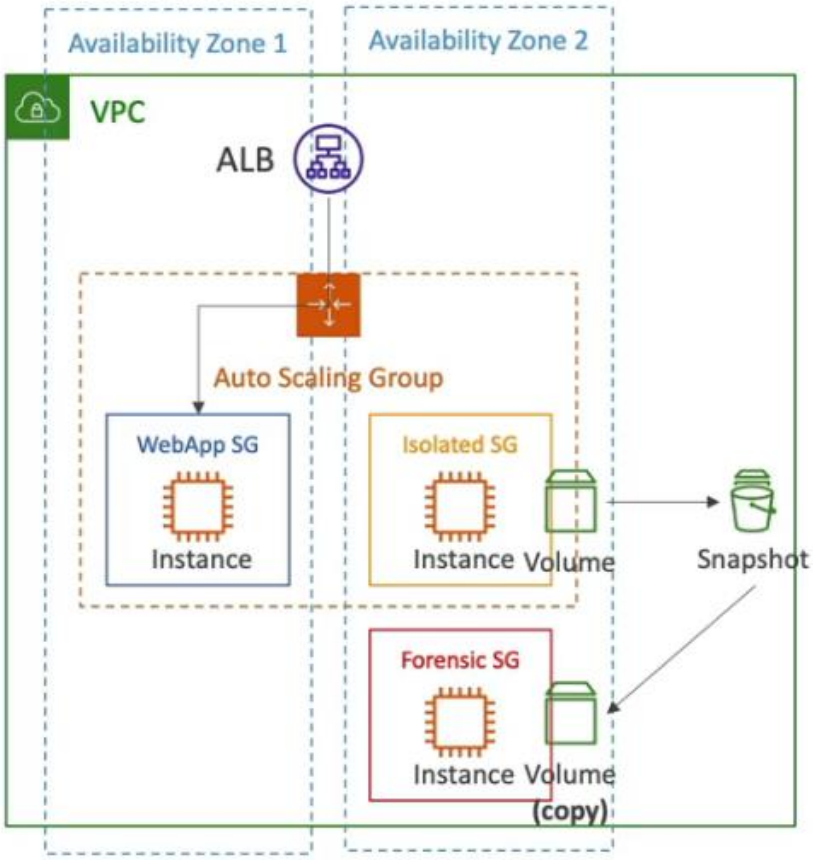
DDoS Simulation Testing on AWS:

- Controlled DDoS attack which enables users to evaluate the resiliency of their applications and to practice event response
- Must be performed by approved AWS DDoS Test Partner
- The target can be either Protected Resource or Edge-Optimized API Gateway that is subscribed to Shield Advanced
- Attach must not be originated from AWS resources
- Attack must not exceed 20 Gbps
- Attack must not exceed 5 million packets per second for CloudFront and 50,000 packets per second for any other service

Incident Response:

EC2 has been hacked. What should the user do?

- Capture the instance metadata – capture the necessary information
- Isolate the compromised instance – disconnect the compromised instance from the network to prevent further damage and limit the attacker's access
    - Replace the instance' Security Group – no outbound traffic authorized
    - Detach instance from Auto Scaling Group – suspend processes
    - Deregister the instance from any ELB
    - This process can be automated via Lambda
- Take a snapshot of the EBS volume – create a snapshot of the EBS volume attached to the compromised instance to preserve evidence for forensic analysis
    - Tag the compromised EC2 instance – ticket id etc.
- Deploy the instance into an isolated environment – launch a new instance in a completely isolated environment, ideally in a separate VPC with no internet access, place the instance in a private subnet to minimize external exposure

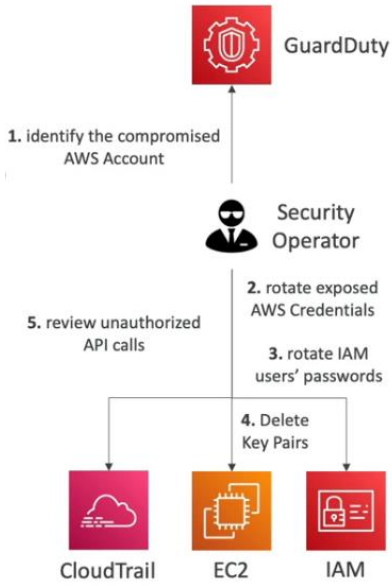


- Access the instance using a forensic workstation – connect securely to the isolated instance using forensic workstation dedicated to analyzing compromised system
- Investigate logs to determine the compromise – review logs (Windows/Linux) on the compromised instance to identify any suspicious activities or potential entry points used by the attacker
- Perform forensic analysis – conduct a thorough forensic analysis to understand the scope of the compromise, identify attack vectors, compromised accounts, and potential backdoor
    - Memory capture can be automated using SSM Run command
- Preserve evidence – document any suspicious activities or logs related to the compromise, as they may be useful for forensic analysis and investigation
- Review security groups and NACLs – access and update security groups and NACLs to ensure that only necessary ports and services are accessible
- Change credentials – reset passwords and access keys for all user accounts associated with the compromised instance, including the root account
- Monitor and detect future incidents – enable comprehensive logging and monitoring, including CloudWatch Logs and AWS CloudTrail, to track and detect any future security incidents or unauthorized access attempts
- Communicate and learn from the incident – notify relevant stakeholders about the incident, including management, IT Teams, and affected users. Conduct a pos-incident review to analyze

the root causes, evaluate the effectiveness of the response plan, and identify areas for improvement in security practices and incident response procedures

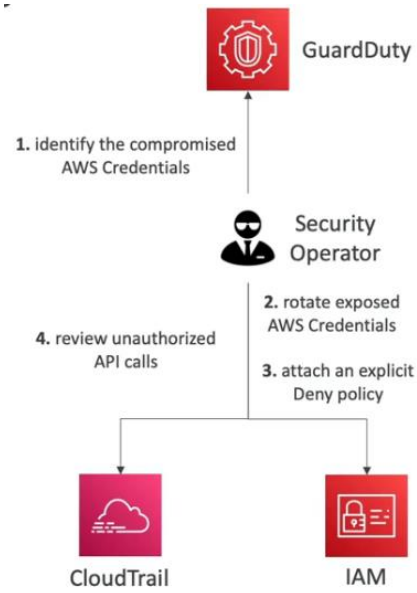
User has leaked his access keys. What should users do?

- Revoke compromised access keys – immediately revoke the leaked AWS Access Keys to prevent unauthorized access
- Disable affected access keys – disable any remaining compromised access keys associated with the affected account
- Scan repositories for other potential leaks – conduct a thorough search across repositories to identify any additional instances of leaked access keys or sensitive information
- Monitor for unauthorized activity – Monitor AWS account activity logs and CloudTrail for any suspicious or unauthorized actions related to compromised access keys
- Review AWS resources – like deleted unauthorized resources
- Learn from the incident – conduct a post-incident analysis to identify gaps in security practices and update policies and procedures accordingly to prevent similar incidents in the future



AWS credentials compromised. What should the user do?

- Identify the affected IAM user – using GuardDuty
- Rotate the exposed AWS credentials – to prevent further unauthorized credentials
- Invalidate temporary credentials – by attaching an explicit Deny policy to the affected IAM user with an STS date condition
- Check for the unauthorized activity – in CloudTrail logs associated with compromised credentials
- Assess the scope of compromise – by evaluating which AWS services and resources the compromised credentials had access to
- Review your AWS resources – like deleted unauthorized resources



- Conduct a thorough investigation – to identify the root cause using Amazon Detective or any other available logs
- Implement security best practices – such as fine-grained IAM polices, regularly rotating the keys, enabling MFA
- Learn from the incident – create and share the lesson learnt summary

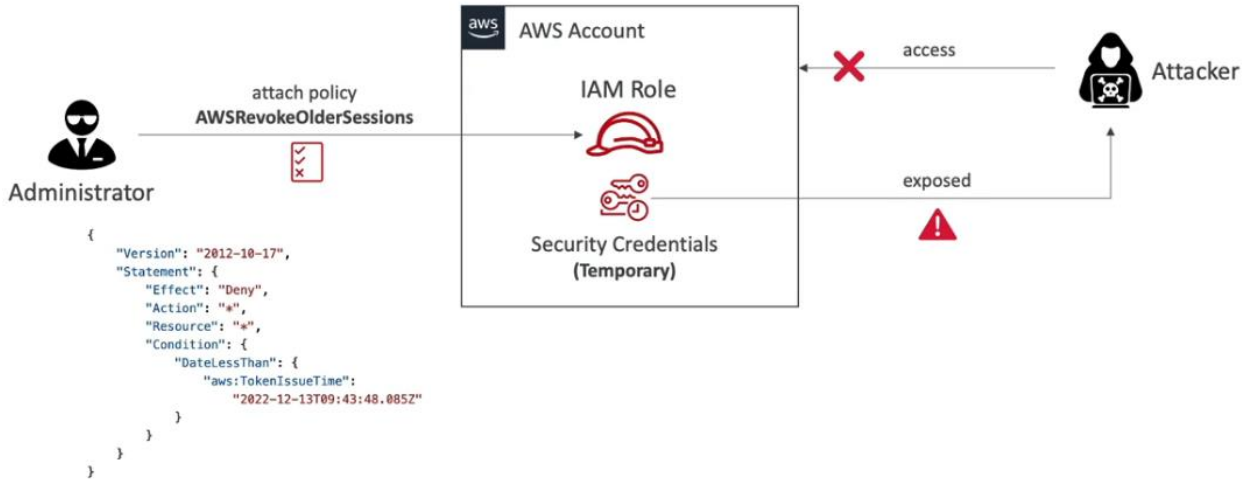
IAM Role temporary credentials are compromised. What should the users do?

- Users usually have a long session duration time – 12 hours for example
- If credentials are exposed, they can be used for the duration of the session
    - Immediately revoke all permissions to the IAM policy to the IAM role's credentials issued before a time
    - AWS attaches a new inline IAM policy to the IAM role that denies all permissions if the token is too old – forces users to reauthenticate

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

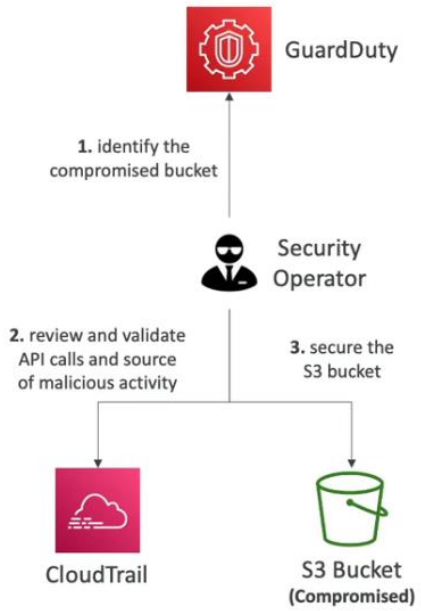
AWSRevokeOlderSessions Policy

    - Doesn't affect any user who assumes the IAM role after you revoke sessions – no worries about deleting the policy



Sensitive data stored in S3 buckets is accidentally exposed due to misconfigured access controls. What should the user do?

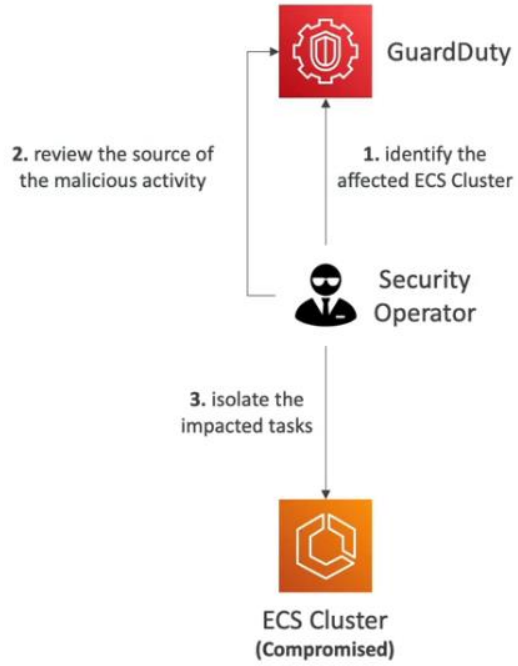
- Identify and assess the exposure – determine the extent of exposure by identifying the affected S3 buckets and specific data that has been exposed
- Containment and isolation – immediately restrict access to exposed S3 buckets by modifying the access control settings to ensure only authorized users can access the data
- Identify the source of malicious activity – IAM/user role and the API calls using CloudTrail or Amazon Detective
- Notification and communication – notify the appropriate internal stakeholders, such as IT security teams, data owners, and legal departments, to ensure timely response and collaboration
- Remediation and access control review – remediate the misconfigured access controls to prevent further exposure. Conduct a thorough review of all S3 bucket access controls and apply the principle of least privileges to restrict access to only necessary entities
    - S3 block public access settings
    - Se Bucket Policies and User Policies
    - VPC Endpoints for S3
    - S3 pre-signed URLs
    - S3 Access Points
    - S3 ACLs



- Data integrity and cleanup – verify the integrity of the exposed data and assess if any unauthorized changes or modifications have occurred. Take necessary steps to clean up or remove unauthorized copies or instances of the exposed data
- Breach notification (if required) – if the exposed data includes PII or falls under specific regulatory requirements, follow relevant breach notification laws and regulations to inform affected individuals or regulatory authorities
- Lesson learnt – perform a post-incident analysis to understand the root cause of the misconfiguration, identify gaps in security controls or processes, and update policies and procedures accordingly to prevent similar incidents in the future

ECS Cluster has been compromised. What should the user do?

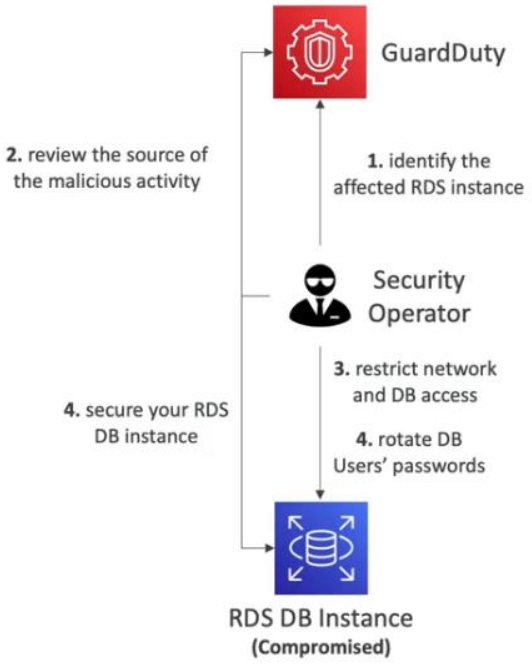
- Identify the compromised ECS Cluster – using GuardDuty
- Identify the source of malicious activity – container image, tasks etc.
- Stop the compromises ECS tasks – after identification, stop them immediately to prevent further unauthorized activity
- Isolate the impacted tasks – deny all ingress/egress traffic using security groups
- Isolate the compromised instance – remove the compromised instances from the ECS cluster to prevent them from affecting other resources
- Investigate the compromise – analyze logs, security groups, and other relevant data to determine the extent of the compromise, identify the attack vector, and understand the impact on the environment
- Evaluate the presence of malicious activity – malware



- Mitigate the vulnerabilities – patch any known vulnerabilities or misconfigurations that may have been exploited to gain unauthorized access
- Change access credentials – reset access keys, IAM roles, and other credentials associated with the compromised ECS cluster to prevent further unauthorized access
- Implement preventive measures – implement security best practices
- Learn from the incident – conduct a post-incident analysis

RDS database has been compromised. What should the user do?

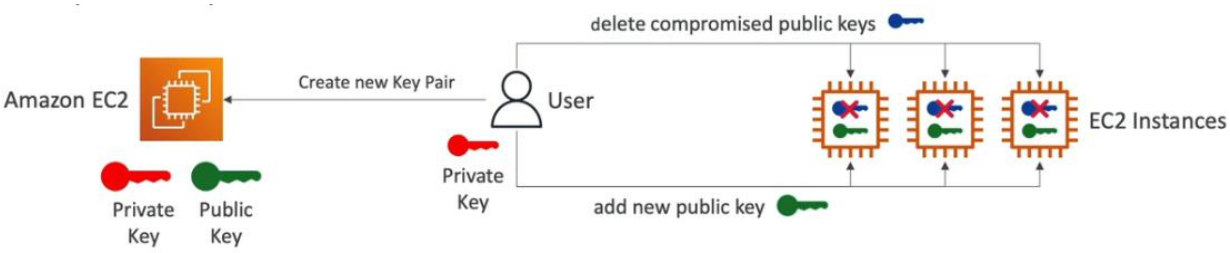
- Identify the affected DB instance and DB user – using GuardDuty
- Isolate the compromised database – isolate it from the network to prevent further unauthorized access and data exfiltration
    - Restrict network access – Security Groups, NACLs
    - Restrict the DB access for the suspected DB user
- Rotate the DB credentials – ensure that the compromised credentials are no longer valid
- Investigate the compromise – review DB audit logs to identify leaked data
    - Analyze network traffic
    - Determine the attack vector
    - Assess the impact on the database and associated systems
- Secure the RDS DB instance – recommended settings
    - Use Secrets Manager to rotate the DB password
    - Use IAM DB Authentication to manage DB user's access without passwords
    - Modify the Security Groups, network configurations, access controls to make it more restrict and secure



- Learn from the incident – conduct a post-incident analysis

EC2 private key pairs have been exposed. What should users do?

- Remove all the public keys – in ~./ssh/authorized_keys file on EC2 instances
- Create a new key pair
- Update the public key – in EC2 instance above mentioned directory



- Investigate unauthorized activity – in the relevant logs
- Implement best practices – MFA and logging
- Automate – use SSM run command to automate the add/delete public keys on EC2 instance process
- Learn from the incident – conduct a post-incident analysis

Troubleshooting Scenarios:

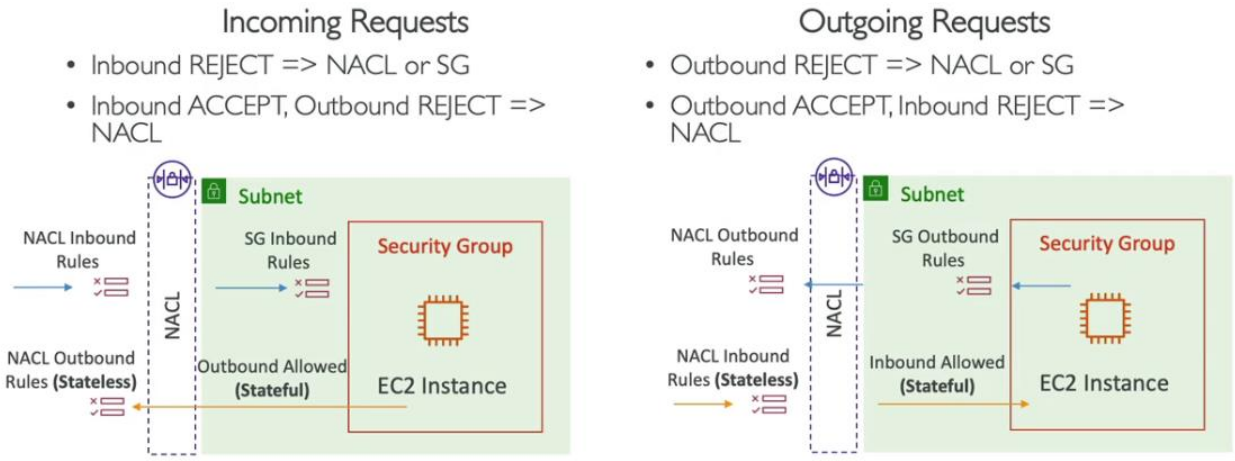
Troubleshoot Security Groups Issues:

- Examine the inbound and outbound rules of the Security Group associated with the affected instances

- Verify that the necessary ports, protocols, and IP range are correctly configured
- Look for overly permissive rules that allow unrestricted access to resources
- Adjust the Security Group rules to ensure the principle of least privilege and restrict access to only necessary resource

Troubleshoot Security Group and NACL issues:

- Via VPC Flow Logs – look at the "Action" field
- Observe the incoming requests and outgoing requests as explained below:



User not able to read from CloudWatch Dashboard:

- Ensure that the user has the necessary permissions to access CloudWatch resources
- Check their IAM policy to confirm they have the required actions and resource-level permissions for CloudWatch dashboards
- Confirm that their IAM policy grants them access to the dashboards – explicit or inherited permissions on the specific dashboards
- Check the region if the selected region is same as the CloudWatch dashboard is created
- Ensure that the user has selected the appropriate time range – also check if the filters are correct

No data in CloudWatch Dashboard form the instance:

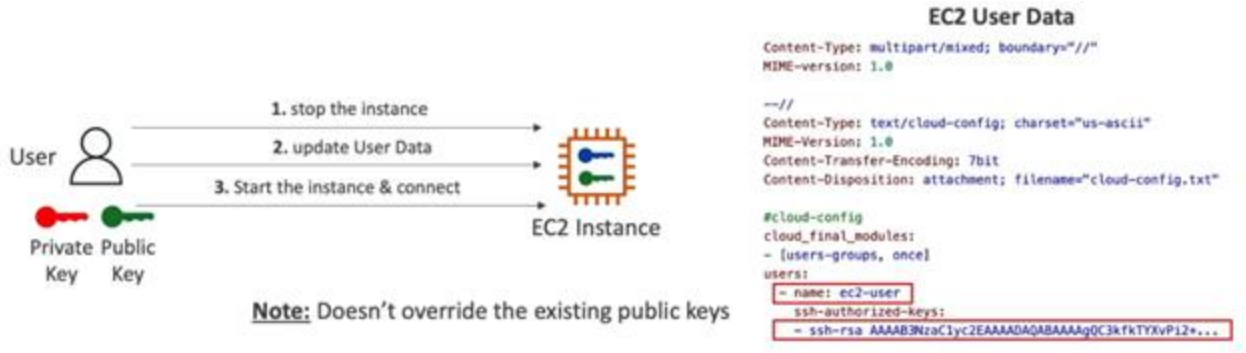
- Check if the agent is installed on the instance
- Verify that the agent is properly configured and running
- Ensure that the IAM role assigned to the instance has the necessary permissions to publish metrics and logs to CloudWatch – IAM policy
- Ensure that the instance has outbound connectivity to CloudWatch service – security groups, NACLs, network settings etc.

EC2 Key Pair/Password is lost:

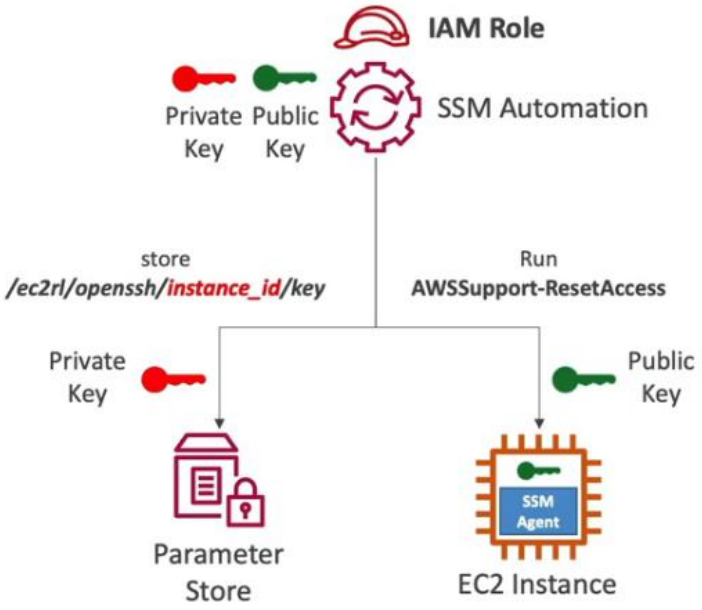
Linux – Lost key pair

- Using EC2 User Data:
  - Create a new key pair, then copy the public key

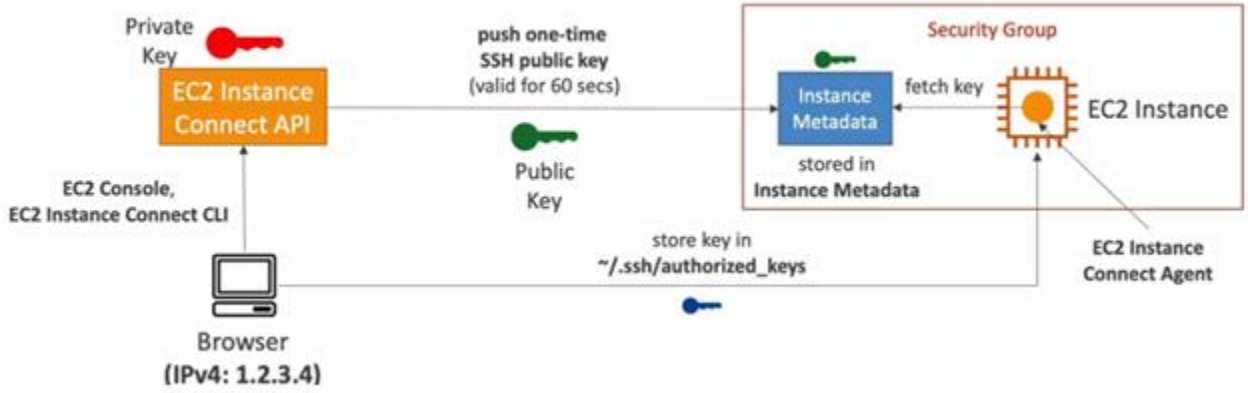
- Stop the instance, update the EC2 User Data (cloud-config format)
- Start the instance and connect with the private key
- Delete the EC2 user data



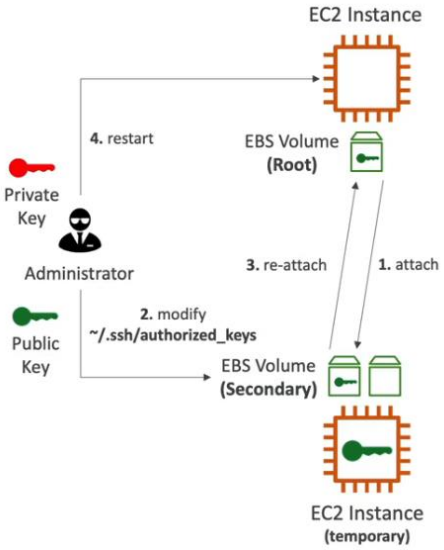
- Using Systems Manager:
  - Use AWSSupport-ResetAccess Automation Document
  - Will create and apply a new key pair
  - Works for both linux and windows
  - The private key stored encrypted in SSM Parameter Store /ec2rl/openssh/instance_id/key
  - Must have SSM agent installed



- Using EC2 Instance Connect:
  - EC2 Connect agent must be installed
  - Connect using EC2 Instance Connect temporary session
  - Store permanent new publish SSH key into ~/.ssh/authorized_keys

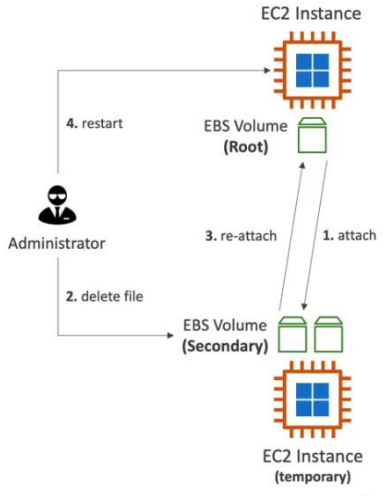


- Using EBS Volume Swap:
  - Create a new key pair
  - Stop the original EC2 instance
  - Detach the EBS root volume
  - Launch a new temporary instance
  - Attach the EBS volume to a temporary EC2 instance as a secondary volume
  - Add the new public key to ~/.ssh/authorized_keys on the volume
  - Re-attach the volume to the original instance, then restart the instance

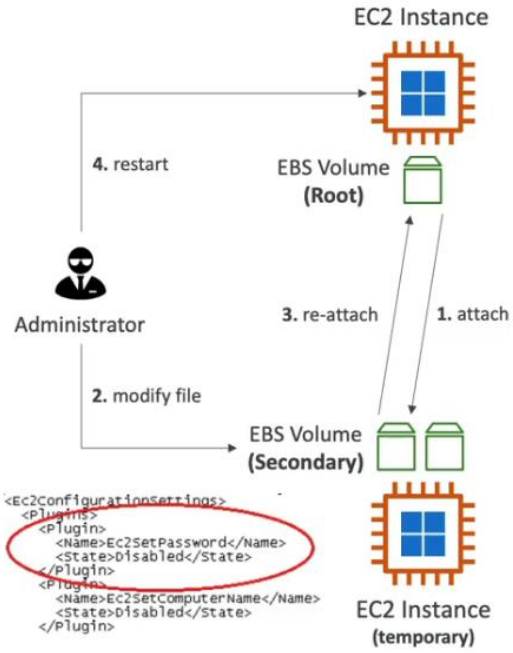


Windows – lost password

- Using EC2Launch – v2:
  - Verify EC2Launch v2 service is running – windows AMIs with EC2Launch v2 service
  - Detach EBS root volume
  - Attach the volume to a temporary instance as a secondary volume
  - Delete file %ProgramData%/Amazon/EC2Launch/state/.run-once
  - Re-attach the volume to the original instance, then restart the instance, you will be able to set a new password

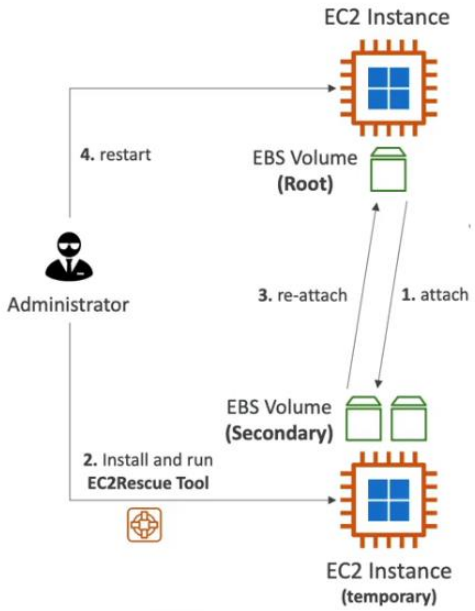


- Using EC2Config – Windows AMIs before Windows Server 2016
    - Verify EC2Config service is running
    - Detach the EBS root volume
    - Attach the volume to a temporary instance as a secondary volume
    - Modify the \Program Files\Amazon\EC2ConfigService\Settings\config.xml
    - Set EC2SetPassword to Enabled
    - Reattach the volume to the original instance, then restart the instance

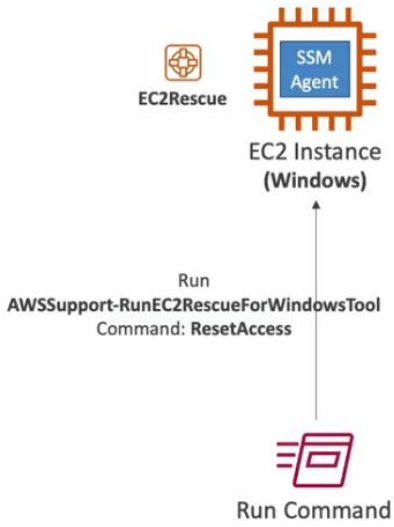


- Using EC2Launch – Windows Server 2016 and later AMIs
    - Detach the EBS root volume
    - Attach the volume to a temporary instance as a secondary volume
    - Download and install EC2Rescue Tool for Windows Server

- Select Offline Instance Options -> Diagnose and Rescue -> Reset Administrator Password
- Reattach the volume to the original instance, then restart the instance



- Using System Manager – must have SSM agent installed
  - Method 1 – use AWSSupport-RunEC2RescueForWindowsTool Run Command document
    - Install and run EC2RescueTool for Windows Server
    - Command is set to ResetAccess
  - Method 2 – Use AWSSupport-ResetAccess Automation document
    - Works for both Windows and Linux
  - Method 3 – Manually AWS-RunPowerShellScript Run Command document
    - Command – net user Administrator Password@123



Issue in Lambda not terminating the instance on CloudWatch Event trigger:

- Check the code and logic in the Lambda function to ensure it is designed to terminate instances properly
- Validate the configuration of the CloudWatch Event rule to ensure it triggers the Lambda function correct
- Confirm that the IAM role associated with the Lambda function has the necessary permissions to terminate instances
- Ensure the CloudWatch Event rule's IAM policy grants the necessary permissions to invoke the Lambda function

CloudTrail logs are not appearing in S3 bucket:

- Ensure that the CloudTrail is enabled and properly configured to send logs to S3 bucket
- Verify the provided S3 bucket name is correct
- Confirm that the S3 bucket has the necessary permissions to receive and store CloudTrail logs
- Ensure that the IAM role associated with CloudTrail has the necessary permissions to write the logs to that S3 bucket
- Check the bucket policy to ensure it allows CloudTrail to write logs to the bucket

External Auditor is not able to read the CloudTrail logs saved in S3 bucket:

- Ensure that the S3 bucket hosting the CloudTrail logs has the necessary permissions to allow access to the Auditor's user
- Confirm that the IAM policies associated with the Auditor's user grant the required permissions to access the S3 bucket
- Review the S3 bucket policy to ensure it grants the necessary permissions to the Auditor's user
- Double-check that the Auditor's user is correctly associated with the policies granting access to S3
- If the S3 bucket is encrypted using KMS, validate that the KMS key permissions allow the Auditor's user to decrypt data

Troubleshooting a secure network infrastructure:

- Ensure that the subnets are correctly configured with the appropriate IP ranges and associated with the correct route tables
- Check the NACL rules for both the public and private subnets ensuring they allow necessary inbound and outbound traffic
- Review the Security Group rules for the instances in both private and public subnets ensuring they allow the required traffic
- Validate the route table configurations for both subnets ensuring they are correctly associated with the respective subnets and contain appropriate routing rules
- Check that the Internet Gateway and Virtual Private Gateway are properly attached to the VPC and configured to handle traffic flow between the subnets and external networks
- Verify the allow/deny traffic in VPC Flow Logs

Troubleshooting authentication and authorization – Conflicting Policies:

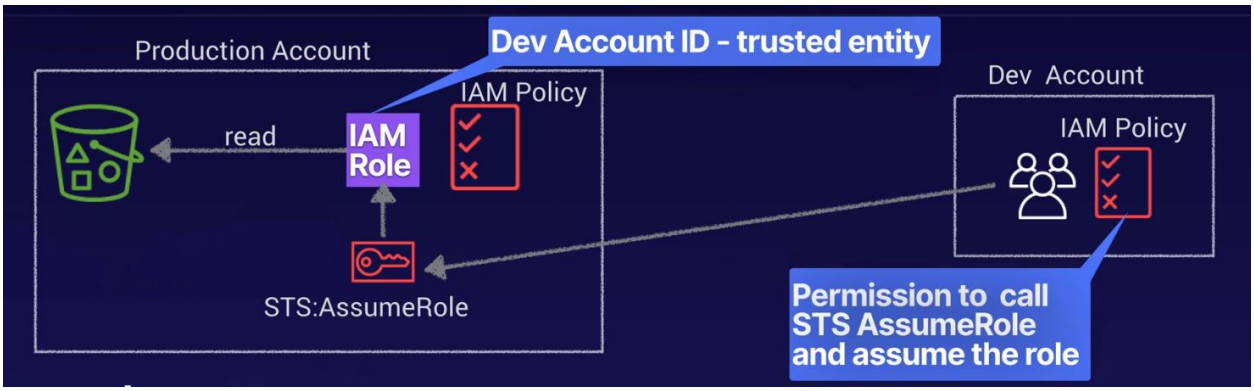
- All actions are denied by default, make sure that there is explicit allow for the action that user wants to perform
- Explicit deny will always override an allow
- With multiple policies in play (IAM Policy, S3 Bucket Policy, S3 ACL, Key Policy etc.) an action will only be allowed if no method explicitly denies and at least one method explicitly allows access
- If using AWS organization – check in case, there is a Permission Boundary prevent the action

Troubleshooting authentication and authorization – Identity Federation:

- Use the correct API for the Job:
  - *sts:AssumeRoleWithWebIdentity* – authentication by a web identity provider
  - *sts:AssumeRoleWithSAML* – authentication by a SAML compliant ID provider (AD)
  - *sts:AssumeRole* – authentication by AWS
- Check that the IAM role associated with federated identities have the necessary permissions
- Verify that the policies attached to the roles allow desired actions and resource access
- Ensure that the role mappings and attribute mappings are correctly defined

Troubleshooting Cross Account Roles – Using STS:AssumeRole:

- Check external account has permissions to call *sts:AssumeRole* – Dev Account IAM policy



- Check the external account is trusted AND has permission to perform the action that is being attempted – Prod Account Role
- Double-check the ARN or name of the target role and ensure it is accurately specified in the AssumeRole request
- Ensure that the IAM user or role making the request has the necessary IAM permissions, including *sts:AssumeRole*, for both source and target account
- Verify and update the trust policy of the role to include the correct accountID or entity details
- Confirm that the external ID provided in the AssumeRole request matches the expected value specified in the target role's trust policy
- Include the MFA-related parameters in the AssumeRole request when required by the target role's configuration

Troubleshooting Lambda access issues:

- Ensure that the IAM execution role associated with the Lambda function has the necessary permissions to access the target AWS services
- Check that the IAM policies attached to the role grant the required permissions for the specific actions – e.g. s3:PutObject, *logs:CreateLogGroup*, ec2:TerminateInstances, *kms:Encrypt* etc.
- Check the resource policies (e.g. S3 bucket policy, CMK key policy) to ensure that the Lambda function's role is granted appropriate access
- Review the resource policies for any explicit denies or restrictions that may prevent the Lambda function from accessing the resources
- If the Lambda function is configured to run within a VPC, verify that it has the necessary VPC subnet, security group, and route table configurations
- Ensure that Lambda function has access to Internet Gateway or relevant VPC Endpoints for accessing the AWS services
- If the Lambda function requires access to CMK for encryption or decryption operations, verify that the function has the necessary permissions to use CMK
- Confirm that the CMK policy allow the Lambda function's role to access and use the key
- If Lambda function interacts with the AWS Secrets Manager to retrieve sensitive information, ensure that the function's execution role has the appropriate permissions to access the secret
- Check the Secrets Manager secret policy to confirm that the Lambda function's role is allowed to retrieve the secret value

Troubleshooting access to a CMK:

- Check the key policy of the CMK to ensure that the appropriate IAM users, roles, or AWS accounts are granted the necessary permissions
- Review the key policy for any explicit denies or restrictions that may prevent access to the CMK
- Ensure that the IAM entity attempting to access the CMK is included in the key policy with the required permissions (*kms:Encrypt*, *kms:Decrypt*)
- Ensure that the IAM entity (user or role) attempting to access the CMK has the necessary permissions in the IAM policy
- Verify that the IAM policy attached to the entity grants the required KMS actions (*kms:Encrypt*, *kms:Decrypt*) and specifies the CMK ARN in the resource section
- If the IAM entity is an IAM role, check the trust policy associated with the role to ensure that it allows the required AWS services to assume the role

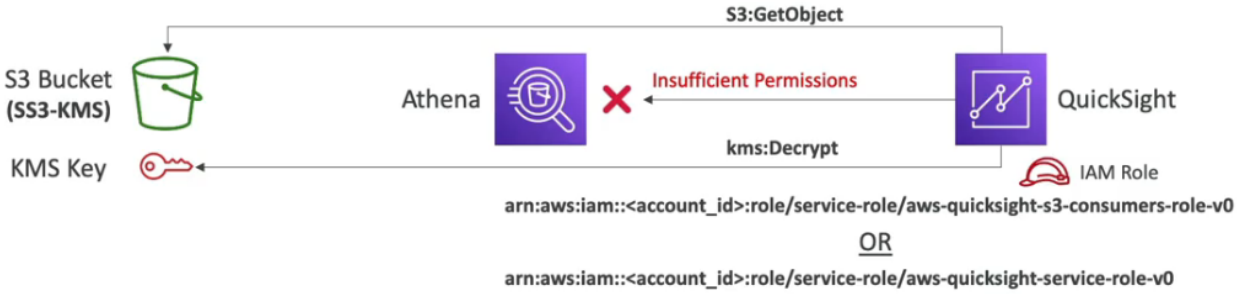
- Key Policy – resource-based policy attached to the CMK
- IAM Policy – assigned to User, Group, or Role

Unified CloudWatch Agent – Troubleshooting:

- Update to the latest CloudWatch agent version
- Test connectivity to the CloudWatch Logs endpoint logs
    - Check Security Groups
    - Check NACLs
- Review account, region, and log group configurations
- Check IAM permissions
- Verify the system time on the instance is correctly configured
- Check CloudWatch Agent logs at */opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log*

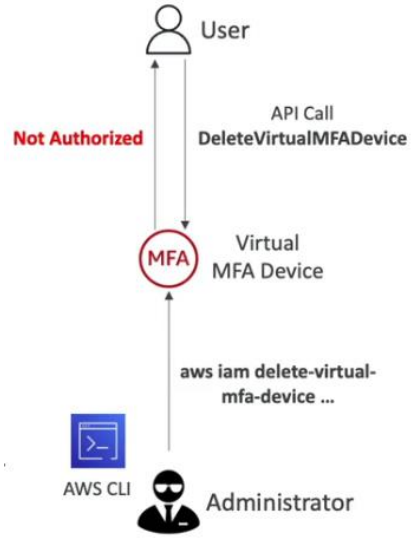
Amazon Athena Troubleshooting:

- Insufficient permissions when using Athena with QuickSight
- If the data in S3 bucket is encrypted using AWS KMS key (SSE-KMS), then QuickSight IAM role must be granted access to decrypt with the key (KMS:Decrypt)



Not authorized to perform iam:DeleteVirtualMFADevice:

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



Error: AWS was not able to validate the provided access credentials:

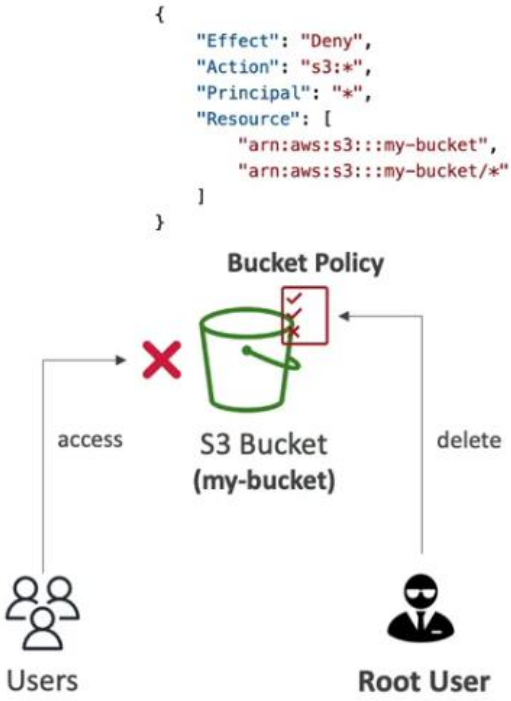
- Use the regional STS endpoint (any region) which will return STS tokens version 2
    - Use the closet regional endpoint for lowest latency
- OR by default, the AWS STS calls to the STS global endpoint issues session tokens which are of version 1 (default AWS regions)
    - Configure STS global endpoint to issue STS tokens version 2 (All AWS regions)

ECR – HTTP 403 (Forbidden) or no basic Auth Credentials:

- This error might happen form the *docker push* and *docker pull* commands even if you have successfully authenticated to Docker using *aws ecr get-login-password*
- Reasons could be any of following:
    - you have authenticated to a different AWS region
    - you have authenticated to an ECR repository you don't have permissions for
    - authentication token has expired – valid for 12 hours

Regain Access to Locked S3 Buckets:

- If users have incorrectly configured their S3 bucket policy to deny access to everyone – Deny s3:*, Principal:*
    - Users must delete the S3 bucket policy using the AWS account root user
    - **Note:** Deny statements in IAM policies do not affect the root account



Can't start an EC2 instance with Encrypted EBS volume:

- Reasons could be:
  - KMS Key might be disabled or deleted
  - EBS service might not have the required permissions to use KMS Key
- Resolution:
  - Make sure the KMS key exists and is enabled
  - Make sure that EBS service has the required permissions to create KMS Grants in behalf of the specified principal
  - *kms:GrantIsForAWSResource* – allows AWS services to create Grants

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

Network Load Balancer Troubleshooting:

- Registered Target is not in service – health check can be failing

- Target's security group does not allow traffic on health check port/protocol from the NLB
- Target's subnet NACL does not allow traffic from NLB

- Requests are not routed to targets
  - Target's security group does not allow traffic
    - Must allow traffic from Client IP addresses – if client IP Preservation *Enabled*
    - Must allow traffic from NLB IP addresses – if client IP Preservation *Disabled*
  - Target's Subnet NACL does not allow traffic
  - Targets are in an AZ that is not enabled
  - EC2 instance is in a peered VPC
    - Then EC2 instances must be registered by IP addresses – not by instance IDs

EC2 Image Builder – Access Denied Error:

- Reasons of this access denied error with status code 403 could be
  - Instance profile role doesn't have the required permissions
  - Instance profile is missing permissions required for logging to S3

- Reasons are:
  - Make sure the following Managed policies added to instance profile role
    - *AmazonSSMManagedInstanceCore*
    - *EC2InstanceProfileForImageBuilder*
    - *EC2InstanceProfileForImageBuilderECRContainerBuilds*
  - Add a policy to instance profile role that grants s3:PutObject permissions to write to S3 bucket

References:

ACloudGuru

- AWS Certified Security – Specialty 2020

Udemy

- By Stephane Maarek

- By Zeal Vora
- By Neal Davis

AWS

- Documentation
- Ramp-up guide
- Workshops
- Skill Builder
- Whitepapers
    - Introduction to AWS Security
    - Security Overview of AWS Lambda
    - Security Overview of Amazon API Gateway
    - Security Pillars – AWS Well-Architected Framework
    - AWS Best Practices for DDoS Resiliency
    - Secure Content Delivery with Amazon CloudFront
    - Security of AWS CloudHSM Backups
    - Security Practices for Multi-Tenant SaaS Applications using Amazon EKS
- Incident Response Technical Guide
- Implementing AWS WAF Technical Guide
- Security Blogs