AWS Security & Encryption

Why Encryption?

Encryption in Flight (TLS / SSL)

- **Data is encrypted before being sent** and decrypted upon receipt
- Prevents **Man-In-The-Middle (MITM) attacks**
- Ensures data integrity and confidentiality during transmission

Key Concepts

- **TLS Certificates** are used to enable HTTPS and provide encrypted communication
- Commonly used in:
  - Secure websites (HTTPS)
  - API communications
  - Client-server interactions

Example Workflow

1. **Client** sends credentials (e.g., username, password)
2. Data is **encrypted using TLS** before transmission
3. **Server** receives and **decrypts** the data securely

This ensures that sensitive information (like `admin` / `supersecret`) is **not readable in transit**.

Server-Side Encryption at Rest

- **Data is encrypted after it is received** by the server
- **Data is decrypted before it is sent** back to the requester
- The data is **stored in encrypted form**, typically using a **data key**
- The encryption and decryption processes rely on a **key management system**, and the server must have access to these keys

Key Concepts

- Used in AWS services such as **Amazon S3**
- Ensures that data stored in the cloud remains protected even if unauthorized access to the storage layer occurs
- Keys are often managed via **AWS KMS** or other key management tools

Example Workflow

1. **Client** sends data to the AWS service (e.g., S3) over HTTP(S)
2. The service encrypts the object using a **data key**
3. The encrypted object and the data key (securely managed) are stored
4. When the object is requested, the service decrypts it before returning the response

Client-Side Encryption

- **Data is encrypted on the client-side** before being sent to the server
- The server **never sees the unencrypted data**
- Only the **receiving client** is capable of decrypting the data
- Ensures that even if the server is compromised, **data remains unreadable**

Key Concepts

- The **server does not have access** to the decryption keys
- Often used in highly secure environments where end-to-end encryption is required
- Can leverage **Envelope Encryption**:
  - Encrypt data with a **data key**
  - Then encrypt the data key with a **master key**

Example Workflow

1. **Client** encrypts the data using a client-side data key
2. The encrypted object (and optionally encrypted data key) is **stored** on a service (e.g., S3, FTP)
3. The receiving **client retrieves and decrypts** the object using the correct key

This method ensures **maximum confidentiality** as data is never exposed to the server in plaintext.

AWS KMS (Key Management Service)

Overview

- **KMS is the central service for encryption** in AWS
- Anytime you hear "encryption" related to an AWS service, it's likely backed by KMS

Features

- **Key Management**:
  - AWS handles creation, rotation, and storage of encryption keys
  - KMS keys can be customer-managed or AWS-managed
- **Access Control**:
  - Fully integrated with **IAM** to control key usage
  - Enables fine-grained permissions over who can use or manage encryption keys
- **Auditability**:
  - **AWS CloudTrail** logs all KMS key usage for auditing and compliance
- **Service Integration**:
  - Seamlessly integrated into most AWS services, including:
    - **EBS**
    - **S3**
    - **RDS**
    - **SSM Parameter Store**
- **API Access**:
  - KMS encryption and decryption is also accessible via:
    - AWS SDKs
    - AWS CLI

Best Practices

- **Never store secrets in plaintext**, especially in code
- Use KMS to encrypt secrets and store them safely, for example in:
  - **Environment variables**

    - **Configuration files**

KMS Key Types

Terminology

- **KMS Keys** is the new name for **KMS Customer Master Keys (CMKs)**

1. Symmetric Keys

- Uses **AES-256** encryption
- A **single key** is used for both encryption and decryption
- **Most AWS services** that integrate with KMS use symmetric keys
- You **never get direct access** to the key itself
    - Must use the **KMS API** to perform cryptographic operations

2. Asymmetric Keys

- Based on **RSA** or **ECC** key pairs
- Involves:
    - **Public Key** for encryption or signature verification
    - **Private Key** for decryption or signing
- The **public key is downloadable**
- The **private key is not accessible in plaintext**
- Useful for:
    - External applications that need encryption but **cannot call the KMS API directly**

Use Cases

- **Symmetric**: Default choice for AWS service integrations
- **Asymmetric**: Ideal for client-side encryption and digital signature scenarios

AWS KMS (Key Management Service) – Key Types and Costs

Types of KMS Keys

- **AWS Owned Keys** *(Free)*:

    - Used by services like **SSE-S3**, **SSE-SQS**, and **SSE-DynamoDB** as default encryption keys
    - Not visible or manageable by the customer

- **AWS Managed Keys** *(Free)*:

    - Automatically created and managed by AWS for services
    - Example aliases: `aws/rds` , `aws/ebs`

- **Customer Managed Keys (CMKs)**:

    - Created and managed by the customer in AWS KMS
    - **$1 per key per month**
    - Allows custom policies, key rotation, and more control

- **Imported Customer Keys**:

    - Keys created outside AWS and imported into KMS
    - Also **$1 per key per month**

    - Useful for compliance scenarios requiring external key material

KMS API Pricing

- **$0.03 per 10,000 API calls** to KMS

Key Rotation

- **AWS Managed Keys**:
    - **Automatically rotated** every 1 year
- **Customer Managed Keys**:
    - Can be **automatically rotated** (if enabled)
    - Also supports **on-demand rotation**
- **Imported Keys**:
    - **Manual rotation only**
    - Requires switching to a new key via an alias

Copying Snapshots Across Regions (with KMS Encryption)

Scenario

When copying an **EBS Snapshot** from one AWS region to another, and the snapshot is **encrypted with KMS**, special handling of encryption keys is required.

Example Workflow

Source Region: `eu-west-2`

- **EBS Volume** is encrypted using **KMS Key A**
- The **EBS Snapshot** is also encrypted with **KMS Key A**

Destination Region: `ap-southeast-2`

- To copy the snapshot, AWS uses **KMS ReEncrypt**
- A **new KMS Key B** is used in the destination region
- The copied **EBS Snapshot** and **EBS Volume** in the new region are encrypted with **KMS Key B**

Key Points

- KMS keys are **region-specific**
- Cross-region snapshot copies must **re-encrypt** the data using a **KMS key in the destination region**
- You must **specify or create** a KMS key (`KMS Key B`) in the target region for encryption

KMS Key Policies

Purpose

- KMS Key Policies **control access to encryption keys**, functioning similarly to **S3 bucket policies**
- **Unlike other services**, you **cannot access a KMS key** without a key policy

Default KMS Key Policy

- Automatically created if you don't define one
- Grants **full access** to the key for the **root user** of the AWS account

Custom KMS Key Policy

- Lets you explicitly define:
    - **Users and roles** who can use the KMS key
    - **Administrators** who can manage the key (e.g., enable rotation, delete, etc.)
- Essential for enabling **cross-account access** to KMS keys

Key Takeaway

- **Access to KMS keys is managed exclusively via key policies**
- IAM permissions alone are **not sufficient** without an appropriate KMS key policy

Copying Snapshots Across AWS Accounts (with KMS Encryption)

Steps for Cross-Account Encrypted Snapshot Sharing

1. **Create a Snapshot**
    - Encrypt it using a **Customer Managed KMS Key (CMK)** in the source account
2. **Update KMS Key Policy**
    - Add a **policy to authorize access** from the target AWS account
    - This allows the target account to use the key to decrypt the snapshot
3. **Share the Encrypted Snapshot**
    - Use the AWS console, CLI, or SDK to share the snapshot with the target account
4. **Copy the Snapshot in the Target Account**
    - Create a **copy of the shared snapshot**
    - **Encrypt it using a CMK** owned by the target account
5. **Create a Volume from the Snapshot**
    - Use the newly copied and re-encrypted snapshot to create an EBS volume in the target account

Key Notes

- Cross-account sharing of encrypted snapshots **requires KMS key policies**
- The snapshot must be **re-encrypted with a CMK** in the target account before it can be used

KMS Multi-Region Keys

Overview

- **Multi-Region Keys (MRKs)** allow you to replicate KMS keys across multiple AWS regions
- Each key copy shares the **same key material and key ID**, but resides in a **different region**

Primary and Replica Keys

- **Primary Key**:
    - Created in the original region (e.g., `us-east-1`)
    - Can be used to create **replica keys** in other regions
- **Replica Keys**:

  - Exist in other regions (e.g., `us-west-2` , `eu-west-1` , `ap-southeast-2`)
  - Automatically **synchronized** with the primary key
  - Share the same key ID (e.g., `mrk-1234abcd12ab34cd56ef1234567890ab`)

ARN Example

- `us-east-1` Primary Key:

arn:aws:kms:us-east-1:111122223333:key/mrk-1234abcd12ab34cd56ef1234567890ab

- `us-west-2` Replica Key:

arn:aws:kms:us-west-2:111122223333:key/mrk-1234abcd12ab34cd56ef1234567890ab

Use Case

- Enables **cross-region encryption operations** with **consistent key material**
- Ideal for:
- **Disaster recovery**
- **Global applications** requiring encrypted data in multiple regions

KMS Multi-Region Keys – Key Features and Use Cases

Key Features

- **Identical KMS Keys** exist in multiple AWS Regions
- Keys have the same:
  - **Key ID**
  - **Key material**
  - **Automatic rotation settings**
- **Encrypt in one region** and **decrypt in another**
  - No need to re-encrypt data or make cross-region API calls
- **Each key is managed independently**, even though they are synchronized
- Multi-Region keys follow a **Primary + Replica** model, but they are **not global** entities

Use Cases

- **Global client-side encryption**
- **Encryption for Global DynamoDB tables**
- **Encryption for Global Aurora databases**

These keys simplify cryptographic operations in applications that span multiple AWS regions.

DynamoDB Global Tables and KMS Multi-Region Keys – Client-Side Encryption

Overview

- **Client-side encryption** allows you to encrypt specific attributes of a DynamoDB item before sending it to AWS.
- With **Amazon DynamoDB Encryption Client**, data is encrypted client-side and replicated as-is using **Global Tables**.

Integration with KMS Multi-Region Keys

- Encrypt data using a **Primary Multi-Region Key (MRK)** in one region (e.g., `us-east-1`)

- The encrypted data is replicated to another region (e.g., `ap-southeast-2`) via **Global Table Replication**
- In the destination region, a **Replica MRK** (same key material) is used to decrypt the data using **low-latency local KMS API calls**

Benefits

- Reduces latency by avoiding cross-region KMS calls
- Maintains data confidentiality during replication
- Enables regional access control and compliance
- Decryption only possible if the client has access to the appropriate **KMS API and key policy**

Workflow Summary

1. **Client app** encrypts an attribute (e.g., SSN) with **primary MRK**
2. Encrypted attribute is stored in **DynamoDB table** in `us-east-1`
3. Data is replicated to `ap-southeast-2` via **Global Table**
4. **Client app** in `ap-southeast-2` retrieves the encrypted attribute
5. The client decrypts it using the **replica MRK** in the same region

Global Aurora and KMS Multi-Region Keys – Client-Side Encryption

Overview

- With the **AWS Encryption SDK**, specific attributes can be encrypted **client-side** before storing them in an **Aurora database**.
- When combined with **Aurora Global Databases**, this encrypted data is **replicated across regions**.

Integration with KMS Multi-Region Keys

- The data is encrypted using a **Primary Multi-Region Key (MRK)** in one region (e.g., `us-east-1`)
- The encrypted column is replicated to other regions (e.g., `ap-southeast-2`)
- In the target region, clients can **decrypt using the Replica MRK** via **low-latency local KMS API calls**

Benefits

- Maintains **data confidentiality** even during global replication
- Enables **secure decryption** only for authorized clients with access to the key
- Prevents even **database administrators** from reading sensitive data
- Reduces latency for cryptographic operations by using **region-local KMS keys**

Example Workflow

1. **Client app** encrypts a column (e.g., SSN) using the **primary MRK**
2. Encrypted data is stored in the Aurora table in `us-east-1`
3. Aurora **Global Database replication** sends data to `ap-southeast-2`
4. Client in `ap-southeast-2` retrieves the encrypted column
5. Decryption is performed using the **replica MRK** in the same region

S3 Replication – Encryption Considerations

Default Replication Behavior

- **Unencrypted objects** and objects encrypted with **SSE-S3**:
    - Replicated by default without additional configuration
- **Objects encrypted with SSE-C** (Customer-Provided Key):

    - Can be replicated, but the key must be provided by the client for both encryption and decryption

SSE-KMS Encrypted Objects

- **Additional steps are required** for replicating objects encrypted with **SSE-KMS**:

  1. **Enable SSE-KMS replication option** in the S3 replication configuration
  2. **Specify the destination KMS key** to be used for encrypting objects in the target bucket
  3. **Adapt the KMS key policy** for the destination key to allow replication
  4. Grant the IAM role used for replication:
     - `kms:Decrypt` on the **source KMS key**
     - `kms:Encrypt` on the **target KMS key**

- Be aware of **KMS throttling**:

  - KMS has service quotas (e.g., API call limits)
  - If you hit the limit, request an **increase via AWS Service Quotas**

Multi-Region Keys

- You can use **multi-region KMS keys**, but:
  - **Amazon S3 treats them as independent keys**
  - The object is still **decrypted with the source key** and then **encrypted with the target key**

Summary

Using SSE-KMS with S3 replication requires **explicit permissions and configuration**, and even with multi-region keys, S3 performs full decrypt–reencrypt operations.

AMI Sharing Process – Encrypted via KMS

Step-by-Step Process

1. **Create AMI in Source Account**

   - The AMI is **encrypted with a KMS Key** from the source account

2. **Modify Image Attribute**

   - Use the AWS CLI or console to **add a Launch Permission**
   - This grants the **target AWS account** permission to launch instances from the AMI

3. **Share KMS Key**

   - The **KMS key** used to encrypt the AMI's snapshots must be **shared** with the target account or IAM role
   - Done by updating the **KMS key policy**

4. **Assign Required Permissions**

   - The target IAM role/user must have the following permissions:
     - `kms:DescribeKey`
     - `kms:ReEncrypt*`
     - `kms:CreateGrant`
     - `kms:Decrypt`

5. **Launch Instance in Target Account**

    - The target account can **launch an EC2 instance** from the shared AMI
    - Optionally, the EBS volumes can be **re-encrypted using a KMS key** owned by the target account

Summary

Sharing encrypted AMIs across accounts involves:

- **Sharing both the AMI and the KMS key**
- **Setting correct permissions**
- **(Optionally) re-encrypting volumes during instance launch**

AWS SSM Parameter Store

Overview

- Provides **secure, scalable, and serverless storage** for:
    - Application configuration data
    - Secrets (e.g., passwords, tokens, API keys)

Key Features

- **Encryption with AWS KMS**:
    - Optional seamless integration with KMS for encrypting parameter values
- **Version Tracking**:
    - Automatically tracks changes to parameters
    - Allows rollback to previous versions if needed
- **IAM-Based Security**:
    - Access is controlled using **IAM policies**
    - Supports fine-grained permission management
- **Notifications**:
    - Integrated with **Amazon EventBridge** to trigger alerts or workflows on parameter changes
- **CloudFormation Integration**:
    - Parameters can be defined and used within **CloudFormation templates**

Use Cases

- Store **plaintext or encrypted configuration** values
- Provide secure, centralized secrets management for serverless and containerized apps
- Enable automated workflows triggered by config changes

Example Flow

1. **Application** retrieves parameter
2. **SSM Parameter Store** checks IAM permissions
3. If encrypted, **AWS KMS** decrypts the value
4. Value is returned to the application

SSM Parameter Store – Hierarchy

Parameter Hierarchy Structure

SSM Parameter Store supports a **hierarchical naming structure**, allowing better organization and access control:

/my-department/ my-app/ dev/ db-url db-password prod/ db-url db-password other-app/ /other-department/

Special Namespaces

- **Secrets Manager Integration**:
  - `/aws/reference/secretsmanager/secret_ID_in_Secrets_Manager`
  - Reference secrets stored in **AWS Secrets Manager**
- **Public Parameters**:
  - `/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2`
  - Used to retrieve the latest Amazon Linux AMIs

Access by Environment

- **Dev Lambda Function**:
  - Uses `GetParameters` or `GetParametersByPath` to fetch `/my-department/my-app/dev/...` parameters
- **Prod Lambda Function**:
  - Fetches `/my-department/my-app/prod/...` parameters

Benefits

- Easier separation of environments (dev, prod)
- Supports **access control per path**
- Enables dynamic configuration and secret retrieval

SSM Parameter Store – Standard vs Advanced Tiers

Feature	**Standard Tier**	**Advanced Tier**
Total number of parameters allowed	10,000 per AWS account and region	100,000 per AWS account and region
Maximum size of a parameter value	4 KB	8 KB
Parameter policies available	No	Yes
Cost	Free	$0.05 per advanced parameter per month

Key Differences

- **Advanced parameters** support:
  - Larger values
  - More parameters
  - Additional features such as **parameter policies** (e.g., expiration)

- Use **Standard** for basic needs without cost
- Use **Advanced** when needing enhanced features and scalability

Parameter Policies (Advanced Parameters Only)

Overview

Parameter policies are available **only for advanced parameters** in AWS SSM Parameter Store. They enable automated lifecycle management for sensitive data such as credentials.

Supported Policies

- **Expiration**:
  - Automatically **deletes a parameter** after a defined TTL (Time To Live)
  - Helps ensure sensitive data is rotated or removed on time
- **ExpirationNotification**:
  - Triggers an **EventBridge notification** before the parameter expires
  - Useful for alerting and initiating automated workflows (e.g., secret rotation)
- **NoChangeNotification**:
  - Triggers an **EventBridge event** when a parameter hasn't been updated in a specified period
  - Helps enforce regular updates for compliance or security

Additional Notes

- You can **assign multiple policies** to a single parameter
- These policies enhance **security automation** and **operational awareness**

AWS Secrets Manager

Overview

- A **newer AWS service** designed specifically for storing **secrets**
- Suitable for storing:
  - Passwords
  - API keys
  - Database credentials
  - Any sensitive configuration value

Key Features

- **Automatic Secret Rotation**:
  - Force secrets to rotate **every X days**
  - Rotation is performed using **AWS Lambda functions**
- **Automated Secret Generation**:
  - On rotation, a new secret can be **programmatically generated and stored**
- **RDS Integration**:
  - Deep integration with **Amazon RDS**:

        - **MySQL**
        - **PostgreSQL**
        - **Aurora**

- **Encryption**:
    - All secrets are **encrypted using AWS KMS**
    - Allows fine-grained access control and audit logging

Primary Use Case

- While Secrets Manager can be used broadly, it is **especially useful for managing RDS credentials** and ensuring they are securely rotated and managed.

AWS Secrets Manager – Multi-Region Secrets

Overview

- **AWS Secrets Manager** supports **replicating secrets across multiple AWS regions**
- Designed to support **multi-region applications** and **disaster recovery**

Key Features

- **Replication**:
    - Secrets created in a **primary region** (e.g., `us-east-1`) can be **replicated** to one or more **secondary regions** (e.g., `us-west-2`)
    - Replication keeps **read replicas in sync** with the primary secret

- **Promotion**:
    - A **read replica** can be promoted to a **standalone secret**, if needed
    - Useful in disaster recovery scenarios where the primary region is unavailable

Use Cases

- **Multi-region applications**
- **Disaster recovery strategies**
- **Access to secrets for multi-region databases**

Example

- `MySecret-A` in `us-east-1` is the **primary**
- It is replicated to `us-west-2` as a **read-only replica**
- Both regions can now access the secret with **low latency**

AWS Certificate Manager (ACM)

Overview

- **AWS Certificate Manager (ACM)** simplifies the **provisioning, management, and deployment** of TLS certificates
- Ensures **in-flight encryption** for services such as websites and APIs (via HTTPS)

Key Features

- **Public and Private TLS Certificates**:

  - Supports both types for different use cases
  - **Public certificates are free of charge**
- **Automatic Renewal**:
  - ACM automatically renews certificates before expiration, reducing operational overhead

Integrations

- ACM certificates can be directly integrated with:
  - **Elastic Load Balancers (CLB, ALB, NLB)**
  - **CloudFront Distributions**
  - **Amazon API Gateway**

Limitations

- **ACM cannot be used directly with EC2 instances**
  - You cannot extract the certificate to install it manually
  - Not compatible with use cases requiring local installation on EC2 or Auto Scaling groups

Example Use Case

1. ACM provisions a TLS certificate
2. The certificate is automatically applied to an **Application Load Balancer**
3. Traffic from clients uses **HTTPS**
4. The ALB forwards **HTTP traffic** to EC2 instances

This provides secure HTTPS endpoints without manual certificate management.

ACM – Requesting Public Certificates

Steps to Request a Public Certificate

1. **List Domain Names**
   - Specify the domains to secure:
     - **FQDN (Fully Qualified Domain Name)**: e.g., `corp.example.com`
     - **Wildcard Domain**: e.g., `*.example.com` (secures all subdomains)
2. **Select Validation Method**
   - **DNS Validation** (Recommended for automation):
     - Add a **CNAME record** to your DNS (e.g., via Route 53)
   - **Email Validation**:
     - Sends confirmation emails to contact addresses from the domain's WHOIS record
3. **Verification Process**
   - DNS or Email validation may take a few hours
   - Once verified, the certificate is issued
4. **Automatic Renewal**
   - ACM **automatically renews** public certificates issued through ACM
   - Renewal begins **60 days before the certificate expires**

Best Practices

- Use **DNS validation** for scalability and automation
- Ensure access to DNS management (especially if using third-party DNS providers)
- Monitor certificate status in the ACM console or via CloudWatch metrics

ACM – Importing Public Certificates

Key Features

- **Manual Certificate Import**:

  - You can **generate TLS certificates externally** and **import them into ACM**
  - Useful for using certificates from third-party Certificate Authorities (CAs)

- **No Automatic Renewal**:

  - Imported certificates **must be renewed manually**
  - You must **re-import** the new certificate before the previous one expires

Expiration Notifications

- ACM sends **daily expiration events** starting **45 days prior to expiration**
- The notification period is **configurable**
- Events are emitted through **Amazon EventBridge**, enabling automation

Monitoring and Compliance

- **AWS Config** provides a managed rule:
  - `acm-certificate-expiration-check`
  - Checks for certificates nearing expiration
  - You can configure the threshold in days for compliance

Example Automation Flow

1. **ACM EventBridge rule** detects an upcoming expiration
2. Triggers a **Lambda function** to handle the event
3. Lambda could notify via **SNS** or enqueue a task in **SQS**
4. **AWS Config** rule checks and flags **non-compliant certificates**

This approach helps prevent outages due to expired certificates by enabling proactive alerting and automation.

ACM – Integration with Application Load Balancer (ALB)

Overview

- AWS Certificate Manager (ACM) can be **integrated with ALBs** to provide **TLS/SSL termination**
- TLS certificates are **provisioned and managed by ACM**, eliminating manual work

Example Architecture

- **EC2 instances** running in an **Auto Scaling Group**
- **Application Load Balancer (ALB)** handles incoming traffic

TLS Flow

1. **Client sends HTTP request**
2. ALB has a **redirect rule** from HTTP to HTTPS

3. ACM-provisioned certificate is used to establish **HTTPS connection**
4. ALB forwards the **decrypted HTTP traffic** to backend EC2 instances

Benefits

- Simplified TLS certificate management (ACM handles provisioning and renewal)
- Enhanced security with **HTTPS enforced via redirect rule**
- Scalable and fault-tolerant architecture using **Auto Scaling Group**

Best Practice

- Always configure **HTTP → HTTPS redirection** at the ALB level
- Use **ACM-managed certificates** to ensure automated renewal and security compliance

API Gateway – Endpoint Types

1. Edge-Optimized (Default)

- Designed for **global clients**
- API Gateway is deployed in a single region, but requests are routed through **CloudFront edge locations**
- This **improves latency** and global performance

2. Regional

- Intended for **clients within the same AWS Region**
- API Gateway is deployed regionally and accessed directly
- You can **manually integrate with CloudFront** to gain more control over:
    - Caching behavior
    - Custom domain settings
    - Distribution settings

3. Private

- Accessible **only from within your VPC** using an **interface VPC endpoint (Elastic Network Interface - ENI)**
- Requires a **resource policy** to control access and define which VPCs or principals can call the API

Summary

Endpoint Type	Optimized For	Access Scope	Notes
Edge-Optimized	Global clients	Public via CloudFront	Latency improved via edge locations
Regional	In-region clients	Public within region	Can integrate manually with CloudFront
Private	Internal VPC access	VPC-only (via VPC endpoint)	Needs resource policy

ACM – Integration with API Gateway

Custom Domain Name Setup

API Gateway allows the use of **custom domain names** with support for **ACM-provided TLS certificates**.

Edge-Optimized Endpoint (Default)

- **For global clients**
- Requests are routed via **CloudFront edge locations** to reduce latency
- **API Gateway resides in a single AWS region**, but uses CloudFront globally
- **TLS certificate must be in `us-east-1`** (CloudFront region)
- Configure DNS:
  - Use **CNAME** or preferred **A-Alias record** in **Route 53**

Regional Endpoint

- **For clients in the same region**
- No CloudFront distribution involved by default
- **TLS certificate must be imported into API Gateway** in the **same region** as the API Stage
- Configure DNS:
  - Use **CNAME** or preferred **A-Alias record** in **Route 53**

Summary Table

Endpoint Type	TLS Certificate Region	DNS Setup	Notes
Edge-Optimized	`us-east-1`	CNAME or A-Alias (Route 53)	Optimized for global access
Regional	Same as API Stage	CNAME or A-Alias (Route 53)	Direct in-region access, more control

AWS WAF – Web Application Firewall

Overview

- **AWS WAF** is a **Layer 7 (HTTP)** firewall designed to protect **web applications** from common web exploits such as:
  - SQL injection
  - Cross-site scripting (XSS)
  - Bot traffic
  - Other OWASP Top 10 threats
- Operates at **Layer 7** of the OSI model (application layer), while Layer 4 covers transport (TCP/UDP)

Deployment Targets

AWS WAF can be deployed on:

- **Application Load Balancer (ALB)**
- **Amazon API Gateway**
- **Amazon CloudFront**
- **AWS AppSync** (GraphQL APIs)
- **Amazon Cognito User Pool**

Benefits

- Protects against a wide range of **application-level threats**

- **Customizable rules** and **managed rule groups** from AWS and third parties
- Integrates with other AWS services to provide **centralized protection** for web traffic

AWS WAF – Web ACLs and Rules

Web ACL (Web Access Control List)

A Web ACL defines a set of **rules** that control **allow/deny/monitor** behavior for web requests.

Types of Rules

- **IP Set**:
  - Can include up to **10,000 IP addresses**
  - For more IPs, use **multiple rules**
- **String Matching**:
  - Inspect **HTTP headers**, **body**, or **URI strings**
  - Protect against common attacks like:
    - **SQL Injection**
    - **Cross-Site Scripting (XSS)**
- **Size Constraints**:
  - Block or allow requests based on size of headers, body, etc.
- **Geo-Match**:
  - Allow or block requests based on **country**
- **Rate-Based Rules**:
  - Count number of matching requests over time
  - Useful for **DDoS protection** and throttling

Rule Groups

- A **Rule Group** is a **reusable set of rules** that can be added to one or more Web ACLs
- Helps organize and apply common protection strategies across applications

Regionality

- **Web ACLs are Regional**, meaning they apply to services within a specific region
- Exception: **CloudFront** Web ACLs are **global**, since CloudFront is a global service

WAF – Fixed IP with Load Balancer

Key Concepts

- **AWS WAF does not support Network Load Balancer (NLB)**, which operates at **Layer 4 (TCP/UDP)**
- WAF is only supported with **Application Load Balancer (ALB)**, which operates at **Layer 7 (HTTP)**

Requirement: Fixed IP

To achieve a **fixed IP address** while using **WAF with an ALB**, use **AWS Global Accelerator**.

Architecture

- **Global Accelerator**:
  - Provides a **fixed IPv4 address** (e.g., `1.2.3.4`)
  - Routes traffic intelligently and globally to the closest AWS region
- **Application Load Balancer (ALB)**:
  - Hosts your application
  - Has **WAF WebACL attached** for Layer 7 protection
- **AWS WAF**:
  - Must be **in the same region** as the ALB
  - Applies Web ACL rules to incoming HTTP(S) requests

Benefits

- Enables **fixed public IPs** for applications protected by **WAF**
- Maintains **Layer 7 security** while achieving **IP stability**
- Allows **global, low-latency access** via AWS Global Accelerator

AWS Shield – DDoS Protection

What is a DDoS Attack?

- **Distributed Denial of Service (DDoS)**: Overwhelming a system with **simultaneous requests**, aiming to make a service unavailable
- Attacks can occur at:
  - **Layer 3/4**: Network/Transport (e.g., SYN floods, UDP floods)
  - **Layer 7**: Application (e.g., HTTP floods)

AWS Shield Standard

- **Free and automatically enabled** for all AWS customers
- Protects against common network and transport layer attacks such as:
  - **SYN/UDP floods**
  - **Reflection attacks**

AWS Shield Advanced

- **Paid service**: $3,000/month per organization
- Offers **enhanced protection** for:
  - Amazon EC2
  - Elastic Load Balancer (ELB)
  - Amazon CloudFront
  - AWS Global Accelerator
  - Amazon Route 53

Additional Features:

- **24/7 access to AWS DDoS Response Team (DRP)**
- **Cost protection** against billing spikes caused by DDoS attacks
- **Automated Layer 7 protection**:
  - Automatically creates, evaluates, and deploys **AWS WAF rules** to mitigate application-layer attacks

---

Summary Table

Feature	AWS Shield Standard	AWS Shield Advanced
Cost	Free	$3,000/month
Protection Scope	Layer 3/4 (basic)	Layer 3–7 (advanced + app layer)
Coverage	All AWS customers	EC2, ELB, CloudFront, Route 53, Accelerator
WAF Integration	No	Yes (auto mitigation rules)
Cost Protection	No	Yes
DRP Access	No	24/7 access

AWS Firewall Manager

Overview

AWS Firewall Manager is a **centralized security management tool** that enables you to manage **firewall rules across multiple AWS accounts** within an AWS Organization.

Features

- Define a **Security Policy** (a set of common security rules)
- Automatically apply rules to:
  - **AWS WAF** (for ALB, API Gateway, CloudFront)
  - **AWS Shield Advanced** (for ALB, CLB, NLB, Elastic IPs, CloudFront)
  - **Security Groups** (for EC2, ALB, and ENI resources in VPC)
  - **AWS Network Firewall** (at the VPC level)
  - **Amazon Route 53 Resolver DNS Firewall**

Key Capabilities

- Rules are applied to **new resources automatically** as they are created
- Helps enforce **compliance** across **all current and future accounts** in the Organization
- Policies are **created at the regional level**

Benefits

- Simplifies firewall and security rule management at scale
- Enforces consistent security posture across all accounts
- Ideal for organizations with **many AWS accounts** and **centralized security teams**

WAF vs. Firewall Manager vs. Shield

Overview

AWS provides multiple security services that work together to protect applications:

Service	Description
AWS WAF	Web Application Firewall for **granular Layer 7 protection** on a per-resource basis

AWS Shield	Protects against **DDoS attacks** (Layer 3/4 and Layer 7 with Advanced tier)
Firewall Manager	**Centralized management** of security policies (WAF, Shield, SGs) across multiple AWS accounts

---

When to Use Each

- Use **AWS WAF** when:
  - You want **custom rules** for HTTP(S) traffic
  - You need protection for **specific resources** (e.g., ALB, API Gateway)
- Use **Firewall Manager** when:
  - You want to **enforce WAF rules across all accounts** in an AWS Organization
  - You want to **automate protection** for newly created resources
- Use **AWS Shield Advanced** when:
  - You require **DDoS protection** beyond the standard level
  - You want access to the **Shield Response Team (SRT)**
  - You need **advanced DDoS reporting** and **cost protection**

Combined Use

- These services are **complementary**, not mutually exclusive
- A recommended setup for enterprise environments:
  - **WAF**: Define and apply Web ACLs
  - **Shield Advanced**: For DDoS defense and support
  - **Firewall Manager**: To manage and deploy WAF and Shield policies across accounts

AWS Best Practices for DDoS Resiliency – Edge Location Mitigation

BP1 – Amazon CloudFront

- Delivers **web applications at the edge**
- Protects against common DDoS attacks:
  - **SYN floods**
  - **UDP reflection**
- Distributes traffic to **edge locations**, absorbing attack volume before reaching the origin

BP1 – AWS Global Accelerator

- Provides **edge access** to applications
- Offers **DDoS protection** via integration with **AWS Shield**
- Useful when your **backend is not compatible with CloudFront**

BP3 – Amazon Route 53

- Provides **DNS resolution at the edge**
- Includes built-in **DDoS protection mechanisms**
- Helps maintain **high availability** and **low-latency DNS queries**

Using **CloudFront**, **Global Accelerator**, and **Route 53** together enables a layered defense strategy for distributing and protecting traffic at the edge before it reaches your backend services.

AWS Best Practices for DDoS Resiliency – Infrastructure Layer

Infrastructure Layer Defense

- **Best Practices: BP1, BP3, BP6**
- Use services that distribute and absorb traffic before reaching EC2:
  - **AWS Global Accelerator**
  - **Amazon Route 53**
  - **Amazon CloudFront**
  - **Elastic Load Balancing (ELB)**

Protecting Amazon EC2

1. Elastic Load Balancer (BP6)

- Scales automatically with incoming traffic
- Distributes load across multiple EC2 instances
- Acts as a buffer to prevent a single instance from being overwhelmed

2. Amazon EC2 with Auto Scaling (BP7)

- Scales out in response to **sudden traffic spikes**
- Supports both legitimate flash crowds and DDoS bursts
- Ensures availability and resilience during large traffic events

---

These practices help defend applications running on EC2 by combining **scalable infrastructure** and **edge services** that filter, absorb, and distribute traffic intelligently.

AWS Best Practices for DDoS Resiliency – Application Layer Defense

Key Practices

- **Detect and filter malicious web requests** using:
  - **CloudFront** (BP1, BP2): caches static content at edge, shielding the backend
  - **AWS WAF**: sits on top of CloudFront and Application Load Balancer (ALB)

AWS WAF Capabilities

- Filters and blocks traffic based on:
  - Request signatures
  - **Rate-based rules**: automatically block IPs with high request rates
  - **Managed rules**: use IP reputation databases or block anonymous IPs

Additional Mitigation Tools

- **CloudFront** can block traffic by geographic location
- **AWS Shield Advanced**:
  - Supports automatic Layer 7 (application layer) mitigation
  - Dynamically creates, evaluates, and deploys **WAF rules** in response to attacks

---

These strategies help protect against **HTTP floods and bot-driven attacks**, preserving backend performance and application availability.

AWS Best Practices for DDoS Resiliency – Attack Surface Reduction

Key Strategies

- **Obfuscate AWS Resources** (BP1, BP4, BP6):
  - Hide backend resources like **EC2 instances** and **Lambda functions**
  - Use **CloudFront**, **API Gateway**, and **Elastic Load Balancing** as a protective edge layer
- **Protect API Endpoints** (BP4):
  - Avoid exposing direct access to EC2 or Lambda
  - Use:
    - **Edge-optimized API Gateway** for global access
    - **Regional API Gateway + CloudFront** for enhanced DDoS control
- **Use Security Groups and Network ACLs (NACLs)** (BP5):
  - Filter traffic based on **specific IP addresses**
  - Enforce rules at **subnet** or **ENI (Elastic Network Interface)** level
- **Elastic IPs**:
  - Automatically protected by **AWS Shield Advanced**
- **WAF + API Gateway Enhancements**:
  - Apply **rate limiting (burst limits)**
  - Filter requests using **headers**
  - Enforce **API key usage**

These measures help minimize the visible attack surface and make your infrastructure harder to target directly.

Amazon GuardDuty

Overview

- **Amazon GuardDuty** is a **threat detection service** that continuously monitors your AWS accounts for **malicious or unauthorized behavior**
- It uses:
  - **Machine Learning**
  - **Anomaly detection**
  - **Third-party threat intelligence**

Key Features

- **Easy Setup**:
  - One-click activation
  - No software installation required
  - Includes a **30-day free trial**
- **Data Sources**:

  - **CloudTrail Event Logs**: Detects unusual API calls and unauthorized actions
  - **CloudTrail Management Events**: Tracks actions like VPC creation, trail creation
  - **CloudTrail S3 Data Events**: Monitors access to S3 objects (get, list, delete)
  - **VPC Flow Logs**: Detects unusual internal network activity
  - **DNS Logs**: Identifies EC2 instances exfiltrating data via DNS queries
- **Optional Inputs**:
  - **EKS Audit Logs**
  - **RDS & Aurora logs**
  - **EBS volume activity**
  - **Lambda logs**
  - **Additional S3 Data Events**

Integration

- **EventBridge**:
  - Create rules to trigger alerts on GuardDuty findings
  - Targets can include **Lambda functions** or **SNS topics**

Use Case

- Can detect and alert on **cryptocurrency mining attacks**, including a **dedicated finding** for them

Amazon GuardDuty – Architecture Summary

Core Data Sources

- **VPC Flow Logs**: Detects suspicious network traffic patterns
- **CloudTrail Logs**: Monitors API activity across AWS services
- **DNS Logs (AWS DNS)**: Identifies potential domain-based threats

GuardDuty Components

- **GuardDuty Engine**:
  - Ingests and analyzes data from core AWS logs
  - Applies threat intelligence, anomaly detection, and ML-based analysis
- **EventBridge Integration**:
  - Allows you to route findings for automated response
  - Triggers can invoke **Lambda functions**, **SNS**, or other actions

Optional Features

- **EKS Audit Logs & Runtime Monitoring**
- **RDS & Aurora Login Activity**
- **S3 Logs**
- **EBS Volume Activity**
- **Lambda Network Activity**

These optional inputs provide **deep visibility** into specific services and enhance the overall security posture of your AWS environment.

Amazon Inspector

Overview

- **Amazon Inspector** is an automated security assessment service that helps improve the security and compliance of AWS workloads.

Use Cases

- **EC2 Instances**:
  - Requires the **SSM Agent**
  - Assesses:
    - **Unintended network accessibility**
    - **OS vulnerabilities**
- **Container Images (Amazon ECR)**:
  - Assesses container images **at the time of push**
- **Lambda Functions**:
  - Detects vulnerabilities in:
    - **Function code**
    - **Package dependencies**
  - Runs assessments **upon deployment**

Integration

- **Reporting**:
  - Findings are sent to **AWS Security Hub**
  - Findings can be forwarded via **Amazon EventBridge**

Architecture Summary

- Inspector leverages:
  - **SSM Agent** for EC2 scans
  - **Real-time scans** for ECR and Lambda
  - **Event-driven assessments**

Amazon Inspector – Evaluation Details

Scope of Evaluation

- Applies only to:
  - **EC2 instances**
  - **Container Images (ECR)**
  - **Lambda functions**

Assessment Focus

- **Continuous scanning**, triggered **only when needed**
- **Package Vulnerabilities**:
  - Applies to **EC2, ECR, and Lambda**
  - Uses a **CVE (Common Vulnerabilities and Exposures)** database
- **Network Reachability**:
  - Evaluated only for **EC2 instances**

- Each vulnerability is assigned a **risk score** to help with **prioritization**

Amazon Macie

Overview

- **Amazon Macie** is a fully managed **data security** and **privacy** service.
- Uses **machine learning** and **pattern matching** to:
  - Discover sensitive data in AWS (especially **PII** – Personally Identifiable Information)
  - Alert you to potential data security risks

Key Features

- Automatically scans **S3 buckets** for sensitive data
- Provides **detection**, **classification**, and **protection** capabilities
- Generates findings that can be integrated with **Amazon EventBridge** for:
  - Notifications
  - Automation workflows