AWS
SIC – Security, Identity & Compliance

For the AWS SIC – The Major services would be IAM & KMS, we will be learning those as an Individual Services, so - in this lecture, we will learn all the minor service which are part of the AWS SIC category.

Those Services are

- AWS Cognito
- AWS Certificate Manager (ACM)
- AWS Secrets Manager (ASM)
- Security Token Service (STS)
- AWS Private Certificate Authority (PCA)
- AWS Directory Services (ADS)
- AWS Web Application Firewall (WAF)

AWS Cognito

Basic Terms that we need to know – before we jump into the AWS Cognito

1. Authentication:

- The process of verifying the identity of a user or service. It ensures that the entity attempting to access a system is who they claim to be.
- **Example**: Logging into an application using a username and password.

2. Authorization:

- The process of determining what an authenticated user or service is allowed to do. It defines permissions and access levels.
- **Example**: Granting a user access to certain resources or features based on their role.

3. User Pool:

- A user directory in Amazon Cognito that helps you manage user registration, authentication, and account recovery.
- **Example**: A pool where users sign up and log in, and their profiles are stored.

4. Identity Pool:

- A service in Amazon Cognito that allows you to grant your users access to other AWS services. It enables the use of federated identities from social providers, enterprise providers, or Cognito user pools.
- **Example**: Allowing users to upload files to S3 after authenticating via Google or Facebook.

5. JWT (JSON Web Token):

- A compact, URL-safe means of representing claims to be transferred between two parties. It is commonly used for authentication in web applications.
- **Example**: The token you receive after logging in, which can be used to prove your identity when making API requests.

6. Federated Identity:

- A system that allows users to access multiple applications with a single set of credentials. AWS Cognito supports federated identities through various identity providers like Facebook, Google, etc.
- **Example**: Logging into multiple services (e.g., AWS services and third-party applications) using a single Google account.

7. MFA (Multi-Factor Authentication):

- A security system that requires more than one method of authentication from independent categories of credentials to verify the user's identity.
- **Example**: Using both a password and a one-time code sent to your phone to log into an account.

8. IAM (Identity and Access Management):

- A web service that helps you securely control access to AWS services and resources. It enables you to create and manage AWS users and groups and use permissions to allow or deny their access to resources.
- **Example**: Creating policies that allow users authenticated via Cognito to access specific AWS resources.

9. OAuth 2.0:

- An authorization framework that enables third-party applications to obtain limited access to a user's resources without exposing their credentials.
- **Example**: Allowing an app to access your Google Drive files without sharing your Google password.

10. OpenID Connect (OIDC):

- An identity layer on top of OAuth 2.0 that provides a way to verify the identity of a user based on the authentication performed by an authorization server, as well as obtain basic profile information.
- **Example**: Using your Google account to sign into an application, which uses OIDC to verify your identity.

11. Tokens:

- **Access Token**: Grants permission to access certain resources.
- **ID Token**: Contains user profile information.
- **Refresh Token**: Used to obtain new access or ID tokens without re-authentication.

12. Scopes:

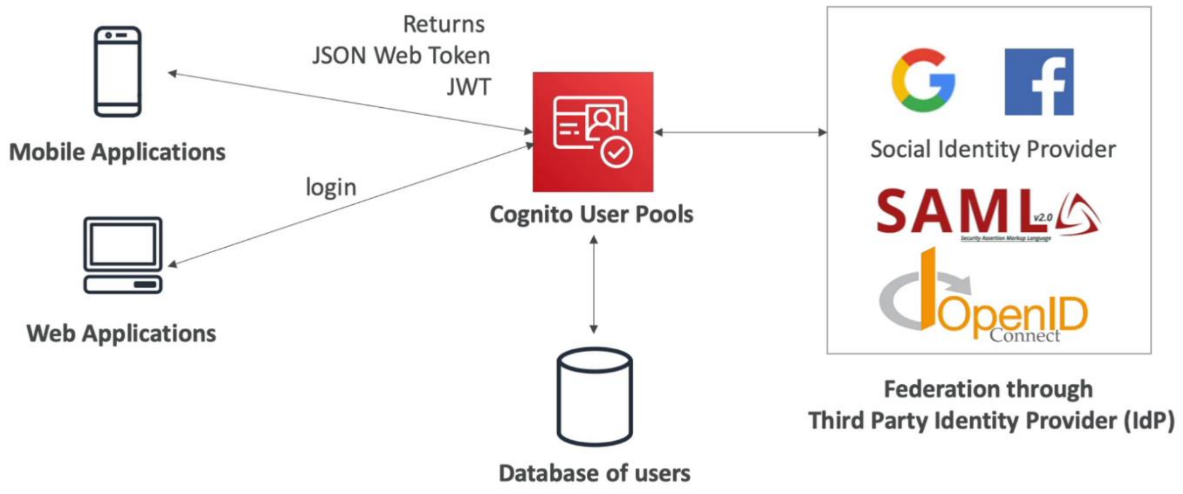
- Used to define what access the application is requesting for the user. In OAuth 2.0 and OIDC, scopes determine the level of access granted to an application.
- **Example**: Requesting read-only access to a user's email address.

13. Claims:

- Statements about an entity (typically, the user) and additional metadata. Claims are contained within JWTs and include information like the user's ID, roles, and more.
- **Example**: A claim in an ID token that specifies the user's email address.

Working Model

Basic reference diagram



This is how it goes

1. User Authentication via Identity Provider (IDP)

- **Step 1:** The user initiates the login process by selecting an Identity Provider (IDP) such as Google, Facebook, or an enterprise IDP.
- **Step 2:** The user is redirected to the chosen IDP's login page, where they enter their credentials (username and password).
- **Step 3:** Upon successful authentication, the IDP generates an authentication token (often a JWT) and redirects the user back to your application with this token.

2. Exchange IDP Token for AWS Cognito Tokens

- **Step 4:** Your application sends the IDP's token (JWT) to AWS Cognito's Identity Pool or User Pool.
- **Step 5:** AWS Cognito verifies the token with the IDP and, if valid, issues its own set of tokens, which usually include:
  - **ID Token:** Contains user identity details like name, email, etc.
  - **Access Token:** Grants permission to access certain AWS resources.
  - **Refresh Token:** Used to obtain new tokens when the current ones expire.

3. Access AWS Resources

- **Step 6**: The application uses the Access Token to request resources from AWS services (e.g., S3, DynamoDB). The Access Token includes the necessary claims that authorize the user's actions.
- **Step 7**: AWS services verify the Access Token provided by Cognito and, if valid, allow the requested operation (e.g., uploading a file to S3).

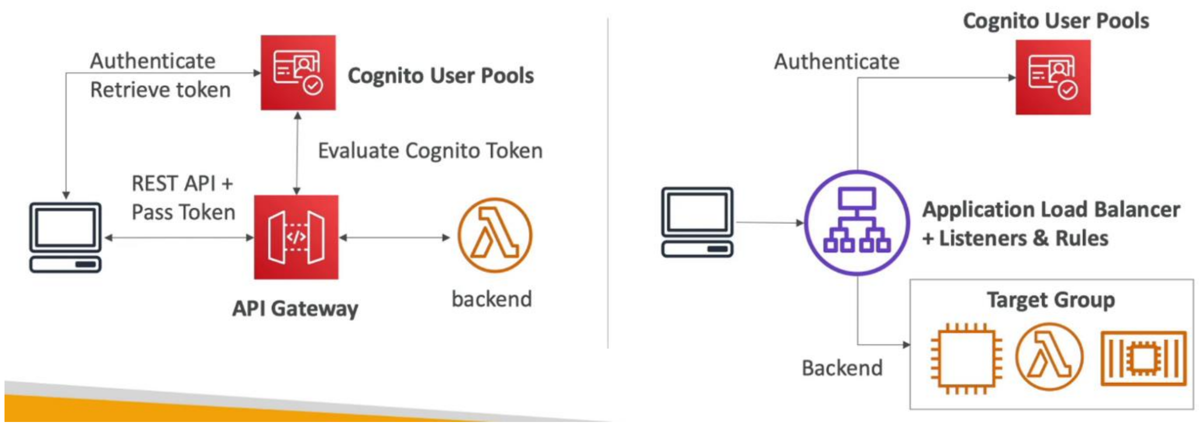
4. Token Refresh (Optional)

- **Step 8**: When the Access Token expires, the application can use the Refresh Token to request a new Access Token and ID Token from AWS Cognito, without requiring the user to log in again.

5. User Logout

- **Step 9**: When the user logs out, the application can invalidate the tokens, either by deleting them locally or by notifying AWS Cognito to revoke them.

Seamless integration of Cognito with API gateway and Application load balancer

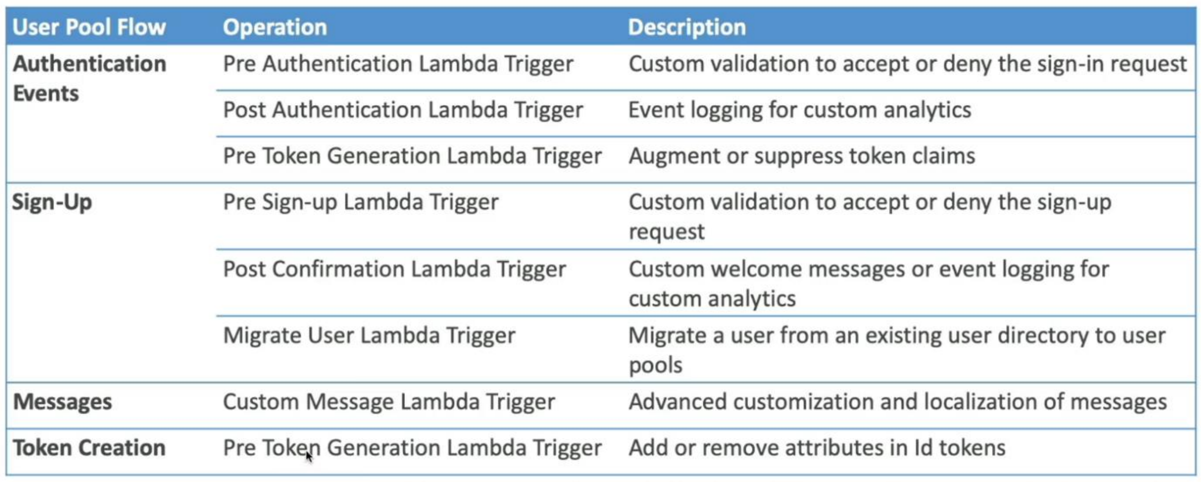


Cognito Hosted UI

- AWS created login and signup page (can be customized using images and CSS)
- To host the hosted UI on a custom domain, we must create an ACM certificate in `us-east-1`

Lambda Triggers

CUP Cognito User Pools (CUP) can invoke lambda functions synchronously on certain events.



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

AWS Certificate Manager

AWS Certificate Manager (ACM) is a service that simplifies the process of provisioning, managing, and deploying SSL/TLS certificates for use with AWS services and your internal connected resources. These certificates are crucial for securing communications between your application and its users, ensuring data integrity, confidentiality, and authenticity.

- Provision, manage and deploy TLS Certificates easily
- **Supports both public and private TLS certificates**
- Free for public TLS certificates
- TLS certificates can be loaded on:
  - ELB
  - API Gateway
  - CloudFront Distributions
- If we already have a certificate, we can load them to ACM. Otherwise, we can request a certificate from ACM directly.
- **ACM issued certificates are valid for 13 months.** They are also renewed automatically.
- **Imported certificates are not automatically renewed** and would need to be imported after getting renewed from the 3rd party.
- An ACM certificate that was validated using DNS validation will automatically renew if the certificate is still being using by an AWS service 60 days prior to its expiration and has an ACM-provided CNAME that is accessible via public DNS. If the certificate is not being used or if the CNAME is not correct, ACM will not automatically validate the DNS and will send notifications starting at 45 days prior to the expiration date.
- For regions where ACM is not supported, **IAM Certificate Store** can be used to import SSL certificates issued by a 3rd party.

AWS Secrets Manager (ASM)

AWS Secrets Manager is a service designed to help you securely manage and retrieve secrets, such as database credentials, API keys, and other sensitive information, used by your applications. Managing secrets manually can be complex and prone to security risks, especially as your applications scale. AWS Secrets Manager automates this process, enhancing security, compliance, and efficiency.

Use Cases:

- **Managing Database Credentials**: Store and rotate database credentials for Amazon RDS, ensuring that your applications always use the latest credentials without manual intervention.
- **Securing API Keys**: Securely store API keys for third-party services and rotate them periodically to minimize security risks.
- **Dynamic Credentials**: Provide your applications with dynamic, short-lived credentials that are automatically rotated, reducing the window of exposure if credentials are compromised.

Intro Parameters

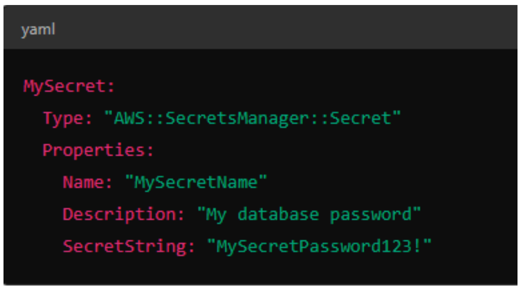
- For storing secrets only
- **Mandatory encryption** using KMS
- **Each secret can have multiple key-value pairs**
- **Ability to force rotation of secrets every n days** (not available in Parameter Store)
- **Well integrated with SQL databases** like MySQL, PostgreSQL, RDS and Aurora to store DB username and password
- **Automated secret rotation using Lambda** (needs IAM permission)
- Mostly used for **RDS authentication**
  - need to specify the username and password to access the database
  - link the secret to the database to allow for automatic rotation of database login info
- Can create custom secrets
- **Secrets are retained after deletion for 7 - 30** (default) days (waiting period)

What are the ways or methods, we need to use – to create or read a secret from the secrets manager while we are using the AWS Cloud Formation Template

Method 01: Using the AWS::SecretsManager::Secret Resource

This CloudFormation resource type is used to create a new secret in AWS Secrets Manager as part of your CloudFormation stack.

Use Case: When you need to define a new secret directly in your CloudFormation template.



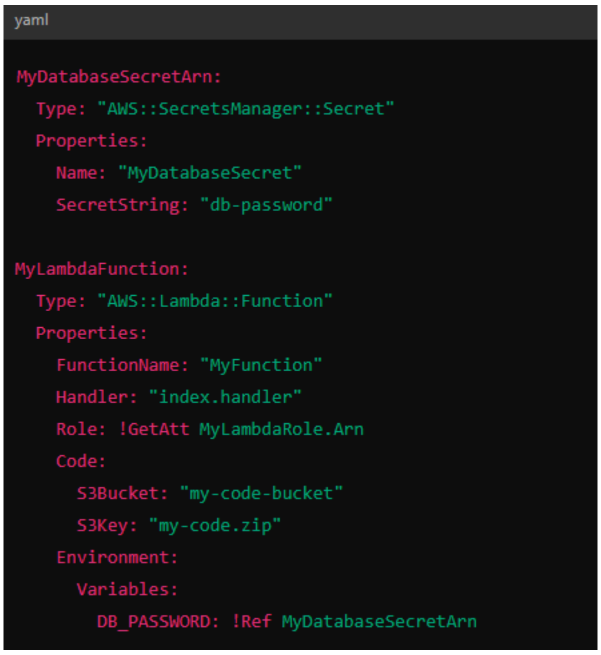
```yaml
MySecret:
  Type: "AWS::SecretsManager::Secret"
  Properties:
    Name: "MySecretName"
    Description: "My database password"
    SecretString: "MySecretPassword123!"
```

- You can specify a SecretString or use a SecretBinary to store secrets.
- The secret can then be referenced in other parts of the template by using the secret's ARN.

Method 02: Referencing Secrets Using the !Ref or Fn::GetAtt Functions

You can reference existing secrets by their ARN, which can be retrieved using the !Ref intrinsic function or the Fn::GetAtt function.

Use Case: When you need to use a secret that is already managed in AWS Secrets Manager.



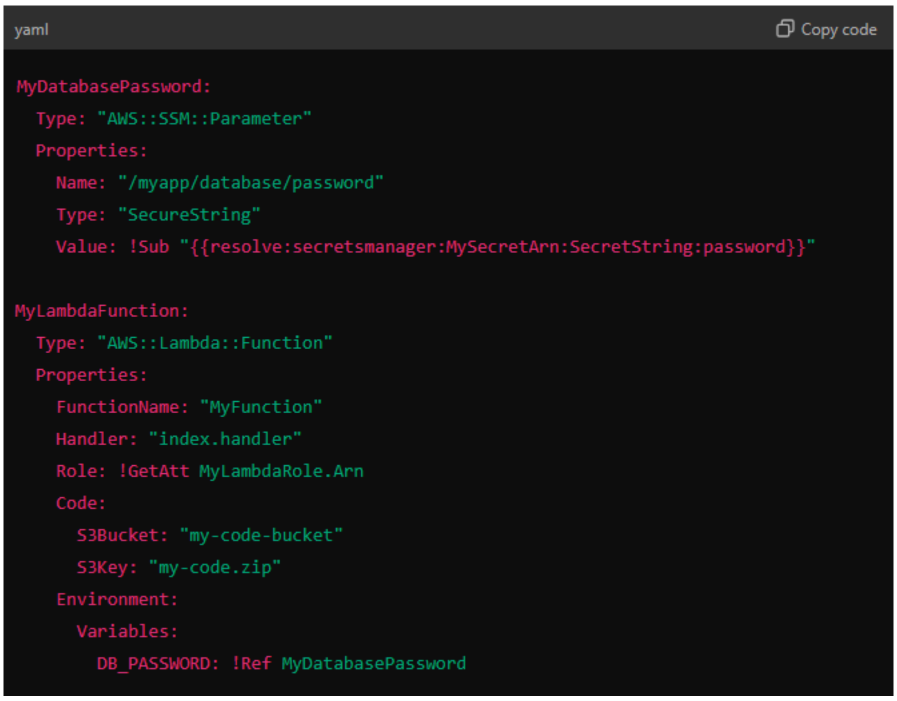
```yaml
MyDatabaseSecretArn:
  Type: "AWS::SecretsManager::Secret"
  Properties:
    Name: "MyDatabaseSecret"
    SecretString: "db-password"

MyLambdaFunction:
  Type: "AWS::Lambda::Function"
  Properties:
    FunctionName: "MyFunction"
    Handler: "index.handler"
    Role: !GetAtt MyLambdaRole.Arn
    Code:
      S3Bucket: "my-code-bucket"
      S3Key: "my-code.zip"
    Environment:
      Variables:
        DB_PASSWORD: !Ref MyDatabaseSecretArn
```

- !Ref returns the secret's ARN, which can be used in environment variables or as input to other resources.
- Fn::GetAtt can be used to get specific attributes, such as the SecretString.

Method 03: Using AWS Systems Manager (SSM) Parameter Store with Secrets Manager

- If your secret is also stored in the SSM Parameter Store, you can use the AWS::SSM::Parameter resource to retrieve it.
- **Use Case:** When secrets are shared or managed through SSM, and you want to reference them in CloudFormation.



```yaml
MyDatabasePassword:
  Type: "AWS::SSM::Parameter"
  Properties:
    Name: "/myapp/database/password"
    Type: "SecureString"
    Value: !Sub "{{resolve:secretsmanager:MySecretArn:SecretString:password}}"

MyLambdaFunction:
  Type: "AWS::Lambda::Function"
  Properties:
    FunctionName: "MyFunction"
    Handler: "index.handler"
    Role: !GetAtt MyLambdaRole.Arn
    Code:
      S3Bucket: "my-code-bucket"
      S3Key: "my-code.zip"
    Environment:
      Variables:
        DB_PASSWORD: !Ref MyDatabasePassword
```

- SecureString in SSM can be backed by Secrets Manager.
- You can use !Ref to retrieve the parameter value.

AWS Security Token Service

AWS Security Token Service (STS) is a web service that allows you to request temporary, limited-privilege credentials for AWS Identity and Access Management (IAM) users or federated users. These temporary credentials can be used to access AWS resources securely without exposing long-term credentials.

In one line → Used to grant limited and temporary access (15 min - 1 hour) to AWS resources.

Use Cases:

- **Temporary Access to Resources**: Developers or administrators can use STS to request temporary credentials for accessing AWS services, reducing the risk of exposing permanent IAM credentials.
- **Cross-Account Role Assumption**: Organizations can securely manage access to resources across multiple AWS accounts by using STS to assume roles with specific permissions in other accounts.
- **Federated Access**: Businesses can integrate AWS resources with their existing identity management systems, providing federated users with secure, temporary access to AWS resources.
- **Mobile and Web Applications**: Applications can use STS to request temporary credentials that allow them to securely interact with AWS services, such as uploading files to S3 or querying a DynamoDB table.

STS APIs

AWS Security Token Service (STS) provides several APIs that you can use to request temporary security credentials, manage cross-account access, and federate users from external identity providers. These APIs are crucial for scenarios where you need temporary, limited-privilege access to AWS resources.

- AssumeRole - assume an IAM Role within your account or **cross account** and return temporary credentials for that role
- AssumeRoleWithSAML - return credentials for users logged in with SAML
- AssumeRoleWithWebIdentity - return credentials for users logged in via a web identity provider like Facebook, Google, OIDC compatible IDP, etc.
- GetSessionToken - get session token for MFA
- GetFederationToken - obtain temporary credentials for a federated user
- GetCallerIdentity - return details about the IAM user or role
- DecodeAuthorizationMessage - decode error message when an AWS API is denied

AWS Private Certificate Authority

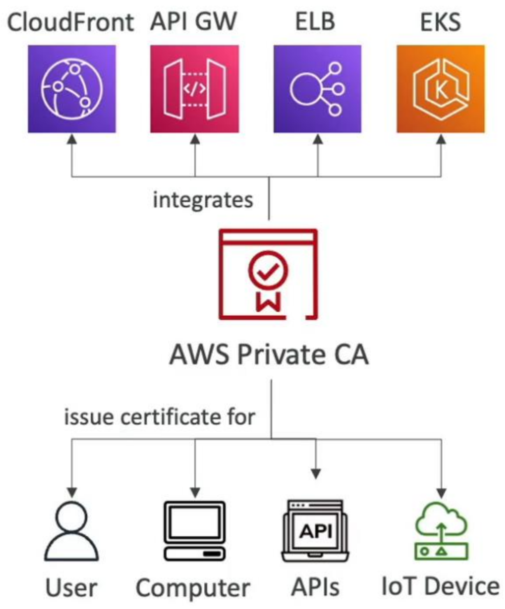
AWS Private Certificate Authority (AWS Private CA) is a managed service that enables you to create and manage private certificates for your organization. These certificates are used to secure communications and establish trust within your private network or between your services. Unlike public certificates, which are used for public-facing websites and services, private certificates are used internally within an organization.

- **Used to create Private CA** (root or subordinate) which can then issue and deploy **X.509 certificates** (can only be used by applications, cannot be used to create other certificates)
- Private TLS certificates are trusted only within your organization (not the public internet)
- Private CA integrates with ELB, API Gateway, CloudFront and **EKS** to load private certificates.
- Usually used to build a **Public Key Infrastructure (PKI)** within an enterprise.

Use Cases:

- **Securing Internal APIs and Microservices**: Issue and manage private certificates to secure communication between internal services and APIs, ensuring that data transmitted over the network is encrypted and authenticated.
- **Device Authentication in IoT**: Use AWS Private CA to issue certificates for IoT devices, ensuring that only trusted devices can connect to your network and communicate with your services.
- **VPN and Remote Access**: Use private certificates to secure VPN connections, ensuring that only authorized users can access your internal network securely.
- **Internal Websites and Applications**: Issue private SSL/TLS certificates for internal websites, dashboards, or applications that are not exposed to the public internet.

Reference


CloudFront
API GW
ELB
EKS
integrates
AWS Private CA
issue certificate for
User
Computer
APIs
IoT Device

AWS Directory Services

AWS Directory Service is a managed service that allows you to run various types of directories on the AWS Cloud, including Microsoft Active Directory (AD). It provides seamless integration between your on-premises directory infrastructure and the cloud, enabling centralized user management, authentication, and resource access control across both environments.

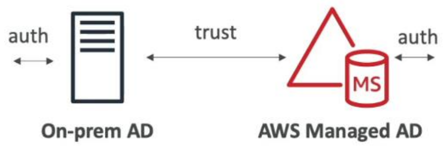
Use Cases:

- **Enterprise Applications:** Run enterprise applications that require Active Directory for user authentication and access control, such as Microsoft SharePoint, Exchange, or custom applications that rely on AD.
- **Cloud-Based Workspaces**: Use AWS Managed Microsoft AD to authenticate and manage user access to Amazon WorkSpaces, providing secure, remote desktop environments for your users.
- **File Sharing and Storage**: Integrate AWS Directory Service with Amazon FSx for Windows File Server, enabling seamless file sharing and storage with directory-based access control.
- **Single Sign-On (SSO):** Implement SSO for cloud and on-premises applications using AWS Directory Service in conjunction with AWS Single Sign-On (SSO) or other identity providers.

Intro Parameters

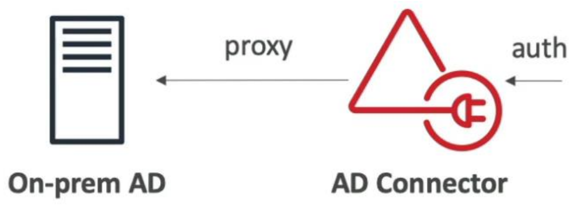
- Database of objects (User Accounts, Computers, Printers, File Shares, Security Groups) on a Windows server. Users can login on any system in the network.
- Objects are organized in **trees**. A group of trees is called **forest**.
- Used to extend the AD network by involving services like EC2 to be a part of the AD to share login credentials.

AWS Managed Microsoft AD



- Login credentials are shared between on-premise and AWS managed AD
- **Manage users on both AD (on-premise and on AWS managed AD)**
- Supports MFA
- Establish trust connections with your on-premise AD
- Supports **directory-aware workloads on AWS**

AD Connector



- AD connector will proxy all the requests to the on-premise AD
- **Users are managed on the on-premise AD only**
- Does not support directory-aware workloads on AWS

Simple AD

- AD-compatible managed directory on AWS (**cannot be joined with on-premise AD**)
- **Users are managed on the AWS AD only**
- Use when you don't have an on-premise AD

AWS
SIC – Security, Identity & Compliance

AWS Web Application Firewall (WAF)

AWS WAF stands for **AWS Web Application Firewall.** It is a security service provided by Amazon Web Services that helps protect your web applications from common web exploits and attacks.

Key Features:

- **Custom Rules:** Create custom rules to block or allow requests based on various conditions, such as IP addresses, HTTP headers, and URI paths.
- **Managed Rules:** Use AWS-managed rule groups to quickly deploy protection against known threats without having to manually configure rules.
- **IP Whitelisting/Blacklisting:** Specify IP addresses or address ranges to be allowed or blocked from accessing your web applications.
- **Rate-Based Rules:** Limit the number of requests from a single IP address to protect against rate-based attacks or abuse.
- **Integration with AWS CloudWatch:** Monitor and set alarms for WAF metrics and logs to track the effectiveness of your security policies and respond to potential threats.

Use Cases:

- **Web Application Protection:** Safeguard your web applications from common vulnerabilities and attacks, ensuring that your applications remain secure and operational.
- **API Security:** Protect APIs exposed through Amazon API Gateway or Application Load Balancer from malicious requests and abuse.
- **Geographic Restrictions:** Restrict access to your web application based on geographic locations or IP addresses.
- **Compliance Enforcement:** Impleme

AWS
SIC – Security, Identity & Compliance

AWS Scenario based Questions

https://lisireddy.medium.com/aws-sic-security-identity-compliance-scenario-based-questions-b1268600dabb

Wish you the best …! Happy Learning ..!

Yours' Love (@lisireddy across all the platforms)