Advanced Identity in AWS

AWS Organizations

- Global service
- Allows management of multiple AWS accounts
- The main account is called the **management account**
- Other accounts are **member accounts**
- A member account can belong to only one organization
- **Consolidated Billing** enables a single payment method for all accounts
- Benefits from **aggregated usage pricing** (e.g., volume discounts on EC2, S3)
- **Reserved Instances** and **Savings Plans** discounts are shared across accounts
- An **API** is available to automate the creation of AWS accounts

AWS Organizations – Hierarchy

Root Organizational Unit (OU)

- At the top of the hierarchy is the **Root OU**, which contains all accounts in the organization.

Management Account

- The **Management Account** is created by default and is responsible for managing the organization.

Organizational Units (OUs)

- OUs are used to group accounts for easier management.
- Example OUs:
  - **OU (Dev)**: Contains accounts related to development environments
  - **OU (Prod)**: Contains accounts used for production workloads
  - **OU (HR)** and **OU (Finance)**: Contain accounts specific to HR and Finance departments

This hierarchical structure allows applying policies and access controls at different levels of the organization.

Organizational Units (OU) – Examples

Organizational Units (OUs) can be structured in various ways depending on business needs. Below are common examples:

Business Unit Structure

- **Management Account**
  - **Sales OU**
    - Sales Account 1
    - Sales Account 2
  - **Retail OU**
    - Retail Account 1
    - Retail Account 2
  - **Finance OU**
    - Finance Account 1
    - Finance Account 2

Used when departments or business units require isolated AWS accounts for billing and access control.

Environmental Lifecycle Structure

- **Management Account**
  - **Prod OU**
    - Prod Account 1
    - Prod Account 2
  - **Dev OU**
    - Dev Account 1
    - Dev Account 2
  - **Test OU**
    - Test Account 1
    - Test Account 2

Ideal for separating environments (production, development, testing) for better resource governance.

Project-Based Structure

- **Management Account**
  - **Project 1 OU**
    - Project 1 Account 1
    - Project 1 Account 2
  - **Project 2 OU**
    - Project 2 Account 1
    - Project 2 Account 2
  - **Project 3 OU**
    - Project 3 Account 1
    - Project 3 Account 2

Useful when organizing resources around projects for tracking costs, isolating environments, and managing permissions.

AWS Organizations – Advantages and Security

Advantages

- Supports **Multi Account** architecture versus using one account with multiple VPCs
- Facilitates cost allocation and tracking using **tagging standards**
- **CloudTrail** should be enabled on all accounts and logs sent to a central S3 bucket
- **CloudWatch Logs** can be centralized in a single logging account
- Use **Cross Account Roles** for administrative tasks across accounts

Security – Service Control Policies (SCP)

- SCPs are **IAM-like policies** applied at the OU or account level to **restrict Users and Roles**
- **Do not apply to the Management Account**, which retains full administrative privileges
- SCPs require an **explicit allow** from the root down through each OU in the path to the target account
- By default, SCPs **deny everything** unless explicitly allowed, similar to IAM

SCP Hierarchy – Example

OU (Root)

- The root contains all OUs and accounts.
- **Management Account**:
  - Not affected by SCPs
  - Has full administrative access

OU (Sandbox)

- Applies: `FullAWSAccess` + `Deny S3`
- Affects:
  - **Account A**:
    - Full AWS access
    - **Denied**: S3 (via Sandbox OU), EC2 (explicit)
  - **Account B** and **Account C**:
    - Full AWS access
    - **Denied**: S3 (via Sandbox OU)

OU (Test)

- Applies: `FullAWSAccess` + `Deny EC2`
- Affects:
  - **Account D**:
    - Can access everything **except EC2**

OU (Workloads)

- Applies: `FullAWSAccess`

OU (Prod)

- Applies: `Allow EC2`
- Affects:
  - **Account E** and **Account F**:
    - Full AWS access (due to inherited policies)

Notes

- SCPs (Service Control Policies) are inherited from the OU hierarchy.
- Explicit Deny in any level overrides Allow.
- SCPs do not apply to the management account.

SCP Examples

SCP Strategies

There are two main strategies for implementing Service Control Policies (SCPs):

- **Blocklist Strategy**:
  - Grant full access and explicitly **deny specific services or actions**
  - Example: FullAWSAccess + Deny S3
- **Allowlist Strategy**:
  - Deny everything by default and **explicitly allow only specific services or actions**

    - Example: Allow only EC2 and S3 actions

For detailed examples, refer to the official AWS documentation:
Example SCPs – AWS Organizations User Guide

IAM Conditions

IAM policies can include conditions to enforce additional security controls.

`aws:SourceIp`

- Restricts **from which IP address** or IP range the API calls can be made.
- Useful for allowing access only from corporate networks or specific IPs.

`aws:RequestedRegion`

- Restricts **to which AWS Region** the API calls can be made.
- Helps enforce compliance by limiting actions to approved geographic locations.

`ec2:ResourceTag`

- Restricts access based on **tags assigned to EC2 resources**
- Useful for enforcing policies that allow or deny actions only on resources with specific tags

`aws:MultiFactorAuthPresent`

- Ensures that the **user is authenticated using MFA**
- Helps enforce stronger security by requiring Multi-Factor Authentication for sensitive actions

IAM for S3

Bucket-Level Permissions

- The `s3:ListBucket` permission applies to:
    - `arn:aws:s3:::test`
    - This grants the ability to list the contents of the **bucket** (not the objects inside)

Object-Level Permissions

- Permissions like `s3:GetObject` , `s3:PutObject` , and `s3:DeleteObject` apply to:
    - `arn:aws:s3:::test/*`
    - These grant access to perform actions on the **objects within the bucket**

Resource Policies & `aws:PrincipalOrgID`

`aws:PrincipalOrgID`

- Can be used in **resource policies** to restrict access to only those **accounts that are part of a specific AWS Organization**
- Helps enforce organization-wide access control at the resource level

Example Use Case

- A resource (e.g., **S3 Bucket: `2022-financial-data`**) includes a policy with `aws:PrincipalOrgID`
- Only **member accounts** of the specified **AWS Organization (`o-yyyyyyyyyy`)** are granted access
- **Users outside the Organization** are denied access automatically

IAM Roles vs Resource-Based Policies

Cross-Account Access Options

There are two primary ways to grant cross-account access:

1. **Resource-Based Policy**

    - Attach a policy directly to a resource (e.g., S3 bucket)
    - Grants access to users or roles from another account

    Example:

    - A user in **Account A** accesses an **S3 bucket** in **Account B** via the bucket's resource policy

2. **IAM Role as a Proxy**

    - Create a role in the target account (e.g., Account B)
    - Users from another account (e.g., Account A) **assume the role**
    - Role permissions determine what the user can do

    Example:

    - A user in **Account A** assumes a **role in Account B** to interact with an S3 bucket

Key Differences

- **Assuming a Role**:

    - When a user, application, or service assumes a role, it **replaces its original permissions** with those defined in the role.
    - This is often used for delegation of access across accounts.

- **Resource-Based Policy**:

    - The principal (user, application, etc.) **retains their original permissions**.
    - Access is granted **without assuming a new role**.

Example Use Case

- A user in **Account A** needs to:
    - **Scan a DynamoDB table** in Account A
    - **Dump the data to an S3 bucket** in Account B

This is achievable using a resource-based policy on the S3 bucket.

Supported AWS Services

- Amazon S3 buckets
- Amazon SNS topics
- Amazon SQS queues
- And other services that support resource-based policies

Amazon EventBridge – Security

When an EventBridge rule triggers an action, it must have the correct permissions to interact with the target service.

Permission Models

- **Resource-Based Policy**:
  - Applied directly to the target resource
  - Used for services like:
    - AWS Lambda
    - Amazon SNS
    - Amazon SQS
    - Amazon S3 buckets
    - API Gateway
  - Example: A Lambda function with a policy that allows invocation by EventBridge
- **IAM Role**:
  - EventBridge assumes the role to interact with the target
  - Required for services like:
    - EC2 Auto Scaling
    - Systems Manager Run Command
    - ECS task

Choosing between a resource-based policy and an IAM role depends on the type of target used in the rule.

IAM Permission Boundaries

Overview

- **IAM Permission Boundaries** are an advanced feature used to define the **maximum permissions** an IAM user or role can be granted.
- They **do not apply to groups**.
- The boundary is defined using a **managed policy**.

How it Works

An IAM entity (user or role) must have:

- An **IAM policy** that grants permissions
- A **permission boundary** that allows those permissions

Both conditions must be met. If either denies access, the action is not allowed.

Example

If the boundary **does not allow** a certain permission, even if the IAM policy does, the permission is **not granted**.

IAM Permission Boundary

+

IAM Policy

Effective Permissions (intersection only)

IAM Permission Boundaries – Use Cases

Integration with AWS Organizations

- IAM Permission Boundaries can be used **in combination with Service Control Policies (SCPs)** from AWS Organizations to implement layered access controls.

Common Use Cases

- **Delegate Responsibilities**:
  - Allow non-admin users to perform tasks like creating new IAM users, but **only within defined boundaries**
- **Developer Autonomy with Limits**:
  - Allow developers to self-manage their permissions
  - Prevent privilege escalation (e.g., cannot assign themselves admin-level permissions)
- **User-Specific Restrictions**:
  - Useful for restricting a **single IAM user** rather than applying broad restrictions at the account level

For more information, refer to the official AWS IAM documentation on Permission Boundaries

IAM Policy Evaluation Logic

IAM evaluates policies using a specific logic to determine whether a request is **allowed** or **denied**.

Key Evaluation Rules

1. **By Default, All Requests Are Denied**
   - No action is allowed unless explicitly permitted.
2. **An Explicit Allow Overrides the Default Deny**
   - If a policy explicitly allows an action, it can proceed.
3. **An Explicit Deny Overrides Any Allow**
   - If a policy explicitly denies an action, it is always denied — even if another policy allows it.
4. **Multiple Policy Types Are Evaluated Together**
   - IAM evaluates:
     - Identity-based policies
     - Resource-based policies
     - Permissions boundaries
     - Session policies
     - Service Control Policies (SCPs)

For the complete evaluation logic and detailed flowchart, refer to the official AWS documentation:
Policy Evaluation Logic – AWS IAM User Guide

Example IAM Policy – Evaluation Questions

Use IAM policy evaluation logic to answer the following:

- **Can you perform `sqs:CreateQueue`?**
- **Can you perform `sqs:DeleteQueue`?**
- **Can you perform `ec2:DescribeInstances`?**

How to Evaluate

To answer these questions, consider:

- Are the actions explicitly allowed in the identity-based policy?
- Is there a permissions boundary that restricts these actions?
- Are there any explicit denies from SCPs or resource-based policies?
- Are the conditions (e.g. region, MFA, tags) met?

You must evaluate the **effective permissions** resulting from all applicable policy layers.

AWS IAM Identity Center (Successor to AWS SSO)

Overview

- Provides **single sign-on (SSO)** access for users across:
  - All **AWS accounts** in your AWS Organization
  - **Business cloud applications** (e.g., Salesforce, Box, Microsoft 365)
  - **SAML 2.0-enabled applications**
  - **EC2 Windows Instances**

Identity Providers

- **Built-in identity store** within IAM Identity Center
- **External identity providers** supported:
  - Active Directory (AD)
  - OneLogin
  - Okta

AWS IAM Identity Center – Login Flow

Login Flow Overview

- The **AWS IAM Identity Center** handles the authentication and access management for users across AWS accounts and applications.
- Users sign in once using their **IAM Identity Center credentials**.
- After authentication, they are granted access to:
  - **Multiple AWS accounts** within the organization
  - **Cloud applications** and **SAML 2.0-enabled apps**
  - **EC2 Windows Instances** (if configured)

This centralized login process simplifies access management and enhances security by enforcing consistent identity policies.

AWS IAM Identity Center – Architecture Overview

Core Components and Integrations

- **AWS IAM Identity Center** acts as the central point for identity and access management across AWS and third-party applications.

Connected Systems

- **AWS Organization**: Enables centralized user access across all AWS accounts
- **Business Cloud Apps**: Provides SSO access (e.g., Salesforce, Box, Microsoft 365)
- **Custom SAML 2.0-enabled Applications**: Supports integration for internal or external SAML apps
- **Windows EC2 Login**: Enables user login to Windows instances via IAM Identity Center

Identity Sources

- **IAM Identity Center Built-in Identity Store**: Default internal directory
- **External Identity Providers**:
  - Active Directory (on-premises or cloud)
  - Third-party services (e.g., Okta, OneLogin)

Functionality

- **Permission Sets**: Define fine-grained access for users per AWS account
- **Browser Interface**: Users access the portal via a browser for SSO
- **SSO**: Single Sign-On experience for all connected applications and services

IAM Identity Center – Permissions Assignment Example

Structure Overview

- **IAM Identity Center** is configured in the **Management Account** of an **AWS Organization**.
- The organization includes:
  - **OU (Development)**
    - Dev Account A
    - Dev Account B
  - **OU (Production)**
    - Prod Account A
    - Prod Account B

User Groups and Permissions

- A user **Group** called **Developers** includes:
  - **Bob**
  - **Alice**
- Two **Permission Sets** are defined:
  - `ReadOnlyAccess`
  - `FullAccess`

Assignment Logic

- IAM Identity Center **assigns** the appropriate **Permission Sets** to users and accounts:
  - For example, Bob might be assigned `ReadOnlyAccess` on Dev Account A
  - Alice might be assigned `FullAccess` on Prod Account B

This allows fine-grained, account-specific access control via centralized IAM Identity Center configuration.

AWS IAM Identity Center – Fine-Grained Permissions and Assignments

Multi-Account Permissions

- **Manage access across multiple AWS accounts** within your AWS Organization.
- Use **Permission Sets**:
    - Collections of one or more IAM policies
    - Assigned to users and groups to define their access in specific AWS accounts

Application Assignments

- Enable **SSO access to SAML 2.0-enabled business applications** such as:
    - Salesforce
    - Box
    - Microsoft 365
- Requires configuration of URLs, certificates, and metadata for SAML integration

Attribute-Based Access Control (ABAC)

- Grants permissions based on **user attributes** stored in the IAM Identity Center Identity Store
- Example attributes: `cost center`, `title`, `locale`, etc.
- **Use Case**:
    - Define permissions once using tags and policies
    - Modify access dynamically by updating user attributes instead of policies

Example: Database Admins

- A **Permission Set** called `DB Admins` is assigned to:
    - **Dev Account** (access to RDS)
    - **Prod Account** (access to Aurora)
- Users in the `Database Admins` group assume roles in the corresponding accounts with appropriate permissions

What is Microsoft Active Directory (AD)?

Overview

- **Microsoft Active Directory (AD)** is available on any Windows Server with **Active Directory Domain Services** enabled.
- It functions as a **centralized database** of directory objects, including:
    - User Accounts
    - Computers
    - Printers
    - File Shares
    - Security Groups

Key Features

- Provides **centralized security management**:
    - Create and manage user accounts
    - Assign permissions and policies

Structure

- Objects in AD are organized into **trees**
- A **group of trees** forms a **forest**
- Authentication and access control are handled by a **Domain Controller**

AWS Directory Services

AWS Managed Microsoft AD

- Fully managed **Microsoft Active Directory** in AWS
- Supports **Multi-Factor Authentication (MFA)**
- Allows you to **manage users locally**
- Can establish **trust relationships** with on-premises Active Directory

AD Connector

- Functions as a **Directory Gateway (proxy)** to forward requests to on-premises AD
- Users are authenticated via the **on-premises AD**
- Supports **MFA**
- No user data is stored in AWS

Simple AD

- An **AD-compatible managed directory** hosted in AWS
- Suitable for small environments
- **Cannot establish trust** with on-premises AD

Summary of Use Cases

Service	Description	MFA	Trust with On-Prem AD
AWS Managed Microsoft AD	Fully functional AWS-hosted AD	Yes	Yes
AD Connector	Proxy to on-prem AD	Yes	N/A
Simple AD	Lightweight standalone AD	Yes	No

IAM Identity Center – Active Directory Setup

Integration Options

1. AWS Managed Microsoft AD

- Direct integration with **IAM Identity Center**
- **Out-of-the-box** support
- Simplifies identity management within AWS

2. Self-Managed Active Directory (On-Premises)

- **Two Integration Options**:
  - **Two-Way Trust**:
    - Establish a trust relationship between a **self-managed AD** and an **AWS Managed Microsoft AD**
    - IAM Identity Center connects via AWS Managed AD
  - **AD Connector**:

        - Acts as a **proxy** to redirect authentication requests to the **on-premises AD**
        - No directory data is stored in AWS

These options allow IAM Identity Center to leverage existing Active Directory environments for centralized user management and authentication.

AWS Control Tower

Overview

- AWS Control Tower provides an **easy way to set up and govern** a secure, multi-account AWS environment.
- It leverages **AWS Organizations** to create and manage accounts.

Benefits

- **Automated Setup**:
    - Quickly configure a multi-account environment with AWS best practices.
- **Policy Management with Guardrails**:
    - Use **guardrails** (predefined policies) to enforce governance.
    - Guardrails can be preventive or detective.
- **Continuous Compliance**:
    - Automatically detect and remediate policy violations.
- **Compliance Monitoring**:
    - Monitor your environment's compliance through an **interactive dashboard**.

AWS Control Tower – Guardrails

Purpose

- Guardrails provide **ongoing governance** for environments managed by AWS Control Tower.
- They help enforce compliance and security best practices across all **member accounts**.

Types of Guardrails

- **Preventive Guardrails**:
    - Implemented using **Service Control Policies (SCPs)**
    - Example: Restrict usage of specific AWS Regions across all accounts
- **Detective Guardrails**:
    - Implemented using **AWS Config**
    - Example: Detect untagged resources

Example Workflow for Detective Guardrail

1. **AWS Config** identifies a non-compliant resource (e.g., missing tags)
2. **SNS** triggers a notification
3. **Lambda** function is invoked
4. Lambda function **notifies an admin** and/or **remediates** the issue (e.g., by adding tags)

This allows for **automatic detection and remediation** of policy violations.