Identity & Access Management

Learning Objectives for this Session

- Introduction to IAM in AWS
- IAM Resources: Users, Groups, Roles and Policies
- IAM Users and Groups
- IAM Roles
- IAM Policies
- IAM Best Practices
- Advanced IAM Features
- Monitoring IAM Activities

Introduction to IAM

- Identity and Access Management (IAM) is a fundamental part of managing security in AWS. It allows you to control access to AWS services and resources securely, ensuring that only authorized users and applications have the right permissions to perform specific actions.
- With IAM, you can create users, groups, roles, and policies to control who can access your AWS services and what actions they are allowed to perform. Implementing IAM effectively helps protect your cloud infrastructure and ensures compliance with security best practices.
- IAM Allows Organizations to enable granular and role based access control. By default design of IAM ensures least privilege access.
- IAM being a global service which enables organization's to manage all IAM Security principle or resources centrally.
- IAM is entirely free of cost.

IAM Resources: Users, Groups, Roles and Policies

IAM in AWS is a centralized service designed to help you manage users and their permissions.

AWS IAM operates on a system of permissions and policies. Here's a breakdown of the core components:

- **Users** - Represent individual people or applications. Each user can have unique security credentials and policies attached to them. An User can be authenticated using Username and Password or Access Key Id & Secret Key.
- **Groups** - Collections of users with similar access requirements. You can assign policies to groups instead of individuals to simplify management.
- **Roles** - Allow temporary access for users or services. For example, an EC2 instance can assume a role to access an S3 bucket without needing direct permissions.
- **Policies** - JSON documents that define the permissions (allow or deny) for resources and actions within AWS. Policies are attached to users, groups, or roles to grant them specific permissions.

IAM Users and Groups

IAM Users represent individual identities within AWS, allowing them to interact with AWS services. Each user is unique and associated with credentials for secure access. Ideal for granting specific access to individual employees, applications, or services that need to interact with AWS resources.

User Permissions

- Assigning Policies to Users: Use AWS-managed or custom policies to grant specific permissions to users.
- Least Privilege Principle: Grant only the permissions required to perform specific tasks, minimizing potential security risks.

IAM Groups are collections of users that simplify permission management by allowing policies to be applied to multiple users simultaneously. A "Developers" group can have permissions to EC2 and S3 services, and all users in the group inherit those permissions.

Access Keys consist of an Access Key ID and a Secret Access Key, enabling programmatic access to AWS services via SDKs, CLI, or APIs.

- Avoid sharing access keys.
- Regularly rotate keys to reduce exposure risks.
- Delete unused access keys.
- Use IAM roles instead of access keys wherever possible.

IAM Roles

IAM Roles in AWS are secure identities designed to grant temporary permissions to entities like AWS services, applications, or users. Unlike IAM users, roles do not have long-term credentials but use temporary security tokens provided by the AWS Security Token Service (STS). Enable access to AWS resources without embedding credentials in applications or scripts.

- Cross-Account Access: Facilitate secure resource access between AWS accounts.
- Roles for EC2 Instances: Allow EC2 instances to perform actions on AWS resources (e.g., accessing an S3 bucket or DynamoDB table).
- Roles for AWS Lambda: Enable Lambda functions to access resources such as S3 buckets, DynamoDB, or Secrets Manager without embedding credentials.
- Service-to-Service Communication: Allow AWS services like ECS or SageMaker to interact securely with other services.
- Applications Running Outside AWS: Use roles with an external identity provider (e.g., SAML, OpenID) for secure access to AWS resources.

Best Practices for IAM Roles

- Follow the Principle of Least Privilege
- Use IAM Policy Conditions
- Monitor Role Usage
- Regularly Review and Rotate Trusted Entities

Inline vs. Managed Policies

AWS Identity and Access Management (IAM) provides two types of policies:

- Inline Policies:
  - Attached directly to a single IAM identity (user, group, or role).
  - Used for fine-grained, specific permissions.
  - Useful for tightly coupled, one-off access scenarios.
  - Easy to customize for specific needs.
  - Harder to manage at scale since they are not reusable.
- Managed Policies:
  - Managed Policies are of two type managed by AWS.
  - Standalone policies created by AWS or the user.
  - Can be attached to multiple IAM identities.
  - AWS-managed policies are maintained by AWS (e.g., AmazonS3ReadOnlyAccess).
  - Customer-managed policies allow full control over permissions.
  - Reusable, easier to update and manage across identities.
  - Slightly less flexible for one-off scenarios.

JSON Policy Structure:

AWS policies are written in JSON and consist of the following components:

- Statements: Each policy contains one or more statements enclosed in {}.
- Actions: Specifies the API operations that are allowed or denied. Example: "s3:GetObject" or "ec2:DescribeInstances".
- Resources: Defines the specific AWS resources to which the policy applies. Example: "arn:aws:s3:::my-bucket/*".
- Effect: Specifies whether the action is allowed or denied. Possible values: "Allow" or "Deny".

Different Ways to Access AWS

AWS provides multiple methods to interact with its cloud services, enabling users to choose the most suitable interface for their tasks and skill levels. Here's an overview of the key access methods:

- AWS Management Console (Graphical User Interface - GUI): A browser-based interface to interact with AWS services visually.
- AWS Command Line Interface (CLI): A text-based interface that allows you to manage AWS resources by running commands in a terminal.
- AWS Cloud Development Kit (CDK): An open-source software development framework for defining cloud resources using programming languages like Python, TypeScript, Java, and C#.

AWS provides flexible options for accessing and managing resources, ensuring users of all experience levels can interact effectively with the platform. GUI is ideal for quick setups, CLI is suited for automation, and CDK empowers developers to manage infrastructure with code.

IAM Best Practices

- Implementing Security Best Practices:
  - Enabling Multi-Factor Authentication (MFA): MFA requires a second authentication factor, like a mobile app or hardware device. MFA is enabled to protecting root accounts and critical IAM users/roles.
  - Regularly Rotating Access Keys: Generate new access keys before deleting old ones. Use AWS SDKs or CLI for temporary session-based credentials instead of long-term keys.
  - Auditing IAM Permissions Using IAM Access Analyzer. IAM Access Analyzer Identifies public or cross-account access to resources like S3 buckets, KMS keys, and IAM roles. Generates actionable findings to tighten security.
  - Generating and Analyzing the Credential Report. IAM Credential Report provides a detailed summary of credentials for all IAM users.

Advanced IAM Features

Advanced IAM Features

- IAM Access Analyzer
  - Identifies public or cross-account access to resources like S3 buckets, KMS keys, and IAM roles.
  - Provides detailed findings with recommendations to secure resources.
  - Identifying Resources with Public or Cross-Account Access
- Identity Federation
  - Allows external identities to access AWS resources via identity providers.
  - Supported methods:
    - SAML 2.0: Integrate with corporate directories (e.g., Active Directory).
    - OpenID Connect (OIDC): Authenticate using services like Google or Okta.
  - AWS SSO: Centralized access control for multiple AWS accounts.
- Resource-Based Policies
  - Attach policies directly to resources like:
    - S3 Buckets: Bucket policies for fine-grained access control.
    - KMS Keys: Policies to encrypt and control access to data.
    - SNS Topics: Policies to manage topic subscriptions and message publishing.

Monitoring IAM Activities

AWS IAM Activities can be monitored by various ways. Some of them are listed below:

- AWS CloudTrail records all API calls and management actions made within an AWS account, including IAM-related activities.
- Amazon CloudWatch: Set up alarms for specific IAM API actions, such as "DeleteUser or "ChangePassword".
- AWS Config monitors configuration changes to IAM resources.

Understanding IAM Limits

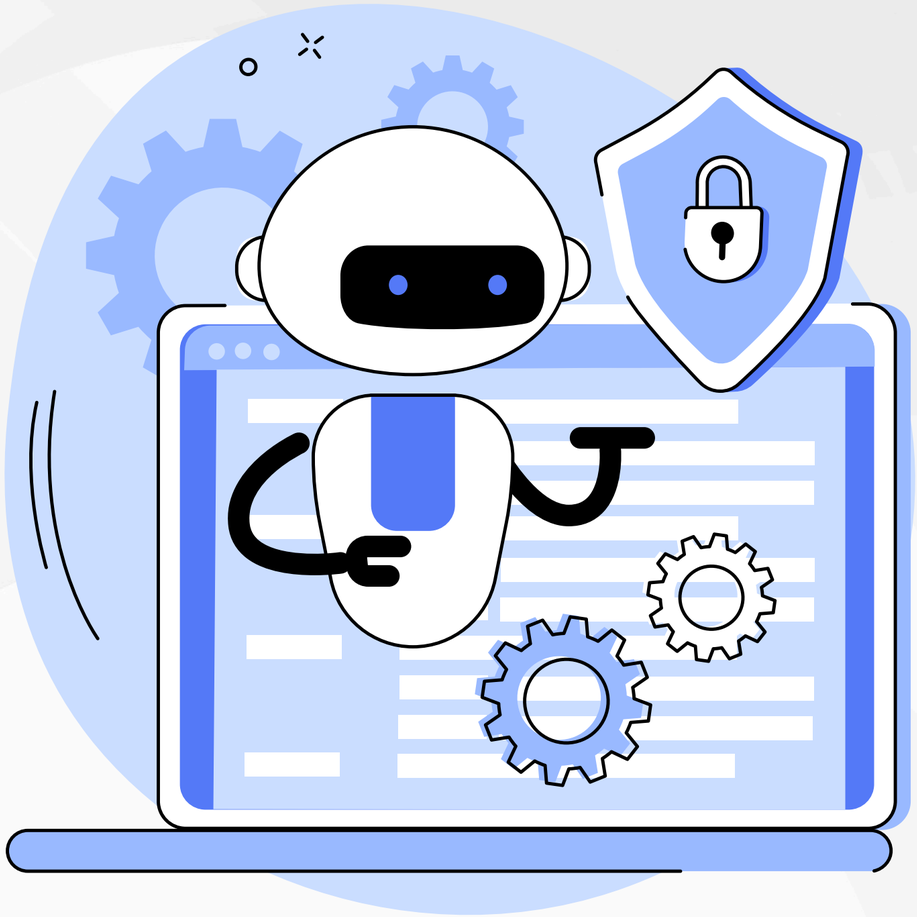
Maximum Number of Users, Groups, and Policies

- AWS IAM comes with default limits to ensure account security and resource optimization:
  - Maximum 5,000 IAM users per account.
  - Maximum 300 IAM groups per account.
  - Maximum 1,500 managed policies per account.

Requesting Limit Increases

- For scaling needs, request increases via the AWS Support Center.

Summary of IAM



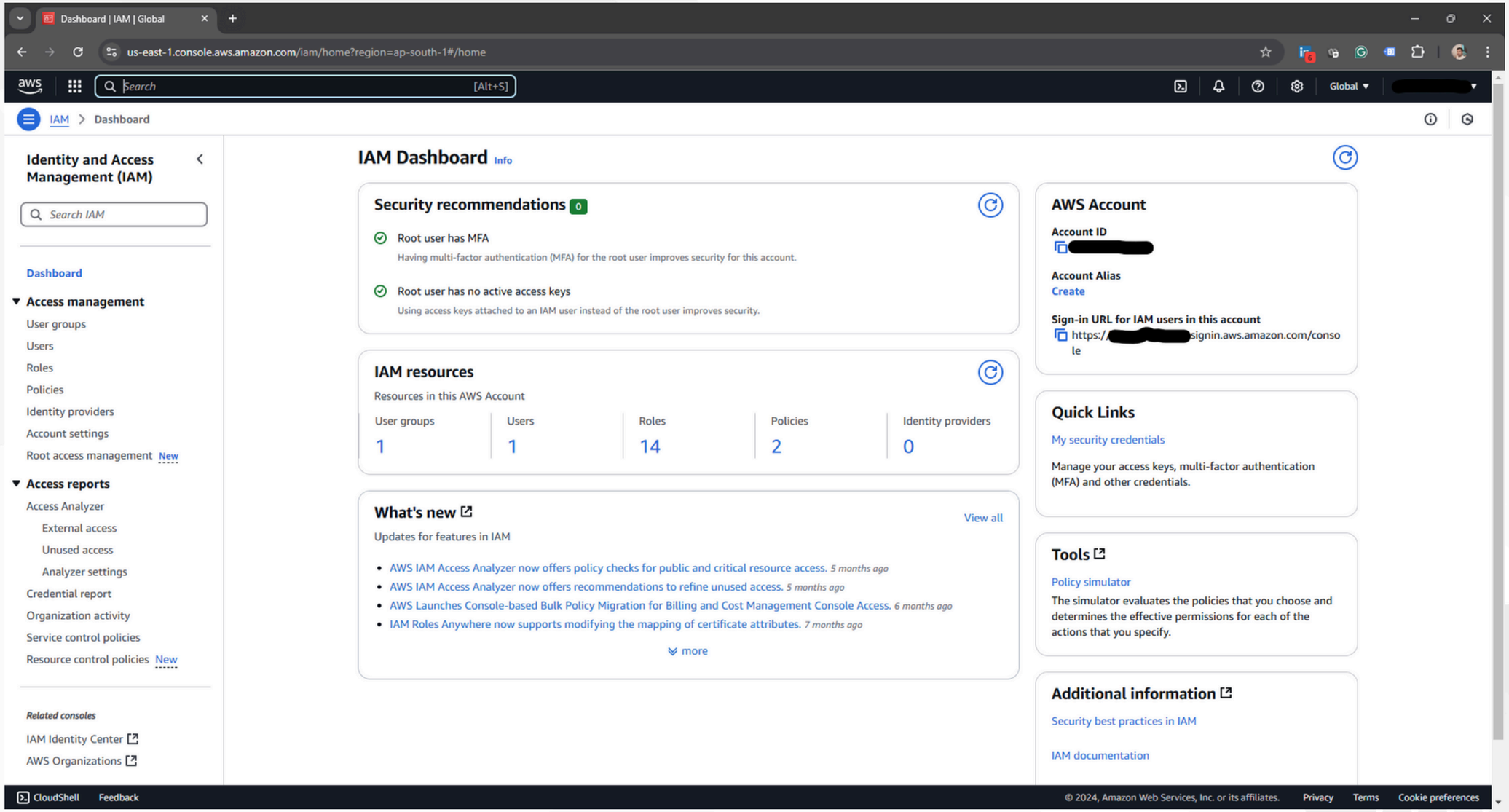
IAM enables us to manage access to AWS resources securely by defining who can access which resources and under what conditions. AWS IAM is the foundation of secure and efficient cloud access management, ensuring both flexibility and compliance with best practices.

IAM supports identity federation, resource-based policies, and role-based access control for cloud-native environments. We can align IAM configuration with Organization policies using AWS Config and also monitor activities in AWS API Level using Cloud Trail and Cloud watch.

To Access AWS API we can either authenticate to the GUI or Access programmatically using CLI/SDK. An IAM User can authenticate using either Username and Password or Access key id and Secret Key. It is best practice to rotate them periodically.

We can Use Advanced feature of IAM to analyze over exposed or cross account access given.

AWS IAM Dashboard



Hands-on Lab

- Create an IAM User, Group, Role and Customer Managed Policy.
- Enable MFA for the IAM User and Root User.
- Assign Customer Managed Policies to groups, users and roles.
- Understand AWS Access Key and Secret Key.
- Setup AWS CLI and create/update user from CLI.

Thank You