AWS Identity & Access Management (IAM)

Sri Balaji

Senior Associate Cloud Engineer and Corporate Trainer

DotNetTricks

Agenda

- AWS IAM - Introduction
- IAM Features
- IAM Infrastructure Elements
- IAM Identities
- IAM Best Practices
- IAM Summary

AWS IAM: Identity and Access Management

- IAM = Identity and Access Management, Global service.
- Free Service
- Allows users to manage access to AWS services and resources.
- Enables to Create and manage users, groups, and use permissions to allow and deny access to resources.
- Users are people within your organization, and can be grouped.

IAM Features

- Centralized Control of your account
- Shared access to your AWS account
- Granular permissions (Different level of access to users)
- Secure access to AWS resources
- Multi-factor authentication (MFA)
- Identity federation (Login From Facebook, LinkedIn, and AD)
- Integrated with many AWS services

Who Uses IAM?

- **IT admins** - An IT admin can use AWS IAM to control access of apps and app resources, based on the business requirements.
- **App developers** - An app developer can use AWS IAM for adding single sign-on (SSO) functionality into to your app where a user can access your app using existing credentials.
- **Users** - A User can manage his identity using AWS IAM.

For example, self-service password reset enables users to change or reset their password with no involvement from an IT administrator or help desk.

IAM Policy

- A policy is a document which defines one or more permissions.
- IAM Policies are stored as JSON documents in AWS.
- IAM Policy can be attached to a user, group and role.
- IAM policy consists of :
  - Version
  - Id
  - Statement

IAM Policy Structure

- Version: Specify the version of the policy language that you want to use. As a best practice, use the latest 2012-10-17 version.
- Id: an identifier for the policy (optional)
- Statement: one or more individual statements (required)
- Statement consists of
  - Sid
  - Effect
  - Principal
  - Action
  - Resource
  - Condition

IAM Policy Structure - Contd..

- Sid: an identifier for the statement (optional)
- Effect: Use **Allow** or **Deny** to indicate whether the policy allows or denies access.
- Principal: account/user/role to which this policy applied to.
- Action: list of actions this policy allows or denies
- Resource: list of resources to which the actions applied to
- Condition: conditions for when this policy is in effect (optional)

Note:

- If you want to define more than one permission for an entity, you can use multiple statements in a single policy.
- You can also attach multiple policies.

Authentication

- Authentication is how you sign in to AWS using your credentials.
- To access AWS, you must be authenticated (signed in to AWS) to send a request to AWS.
- AWS Management Console - Username, password + MFA
- AWS Command Line Interface (CLI) - Access keys,Secret Access Key
- AWS Software Developer Kit (SDK) for code: Protected by access keys

Authorization

- During authorization, IAM uses values from the request context to check for matching policies and determine whether to allow or deny the request.
- In all the policies, the code looks for a Deny statement that applies to the request. This is called an explicit deny.
- If the code finds even one explicit deny that applies, the code returns a final decision of **Deny**.
  - By default, all requests are denied.
  - An explicit allow overrides this default.
  - An explicit deny overrides any allows.

IAM Identities

- **IAM Users** - The person or service who uses the IAM user to interact with AWS.
- **IAM Groups** - A group can contain many users, and a user can belong to multiple groups. Groups can't be nested, they can contain only users, not other groups.
- **IAM Roles** - An AWS identity with permission policies that determine what the identity can and cannot do in AWS.

IAM Policy Simulator

- Enables you to test and troubleshoot identity-based policies, IAM permissions boundaries, Organizations service control policies (SCPs), and resource-based policies.
- The simulator does not make an actual AWS service request, so you can safely test requests that might make unwanted changes to your live AWS environment.
- If you edit a policy inside the simulator, these changes affect only the simulator. The corresponding policy in your AWS account remains unchanged.

Note: https://policysim.aws.amazon.com

IAM Roles for Services

- IAM Roles will provide some AWS services which will need to perform actions on your behalf.
- You can use roles to delegate access to users, applications, or services that don't normally have access to your AWS resources.
- We will assign permissions to AWS services with IAM Roles.
- An IAM user in the same/different AWS account as the role
- An external user authenticated by an external identity provider (IdP) service that is compatible with SAML 2.0 or OpenID Connect.

Common Roles in AWS:

- EC2 Instance Roles
- Lambda Function Roles
- ECS Task Role

IAM Security

IAM Access Advisor

- Provides last accessed information to help you identify unused permissions so that you can remove them.
- You can use last accessed information to refine your policies.
- This helps you with the best practice of least privilege.
- You can view information about for entities or policies that exist in IAM.
- Access Advisor is at **user**-level.

IAM Credentials Report

- You can generate and download a credentials report in IAM.
- Lists all users in your account and the status of their various credentials, including passwords, access keys, and MFA devices.
- Credentials Report is at **account**-level

Multi Factor Authentication - MFA

- Users have access to your account and can possibly change configurations or delete resources in your AWS account .
- You want to protect your Root Accounts and IAM users.
- MFA adds extra security because it requires users to provide unique authentication from an AWS supported MFA mechanism.
- You can enable MFA for IAM users or the AWS account root user.
- MFA = password you know + security device you own .
- Virtual MFA Device
- U2F Security Key
- Hardware MFA Device
- SMS text message based MFA

NOTE: If a password is stolen or hacked, the account is not compromised.

AWS CLI (Command Line Interface)

- AWS Command Line Interface (AWS CLI) is an open source tool provided by AWS.
- Enables you to interact with AWS services using commands in your command-line shell.
- Direct access to the public APIs of AWS services
- To access AWS services with the AWS CLI, you need an AWS account, IAM credentials, and an IAM access key pair.
- When running AWS CLI commands, the AWS CLI needs to have access to those AWS credentials.
- Alternative to using AWS Management Console

Note: https://github.com/aws/aws-cli

Shared Responsibility Model for IAM

User's Responsibility

- Users, Groups, Roles, Policies management and monitoring
- Enable MFA on all accounts
- Rotate all your keys often
- Use IAM tools to apply appropriate permissions
- Analyze access patterns & review permissions

AWS Responsibility:

- Infrastructure
- Compliance validation
- Configuration analysis
- Vulnerability analysis

IAM Best Practices

- Don’t use the root account except for AWS account setup
- One physical user = One AWS user
- Assign users to groups and assign permissions to groups
- Create a strong password policy
- Enforce the use of Multi Factor Authentication (MFA)
- Use IAM Roles for giving permissions to AWS services
- Use Access Keys for Programmatic Access (CLI / SDK)
- Audit permissions of your account with the IAM Credentials Report

Important: “Never share IAM secret key & Access Keys”

IAM - Summary