Lecture 2: IAM

Maharishi International University

Department of Computer Science

M.S. Thao Huy Vu

Maharishi International University - Fairfield, Iowa

Agenda

- IAM
- Group
- User
- Policy
- Best practices for IAM security

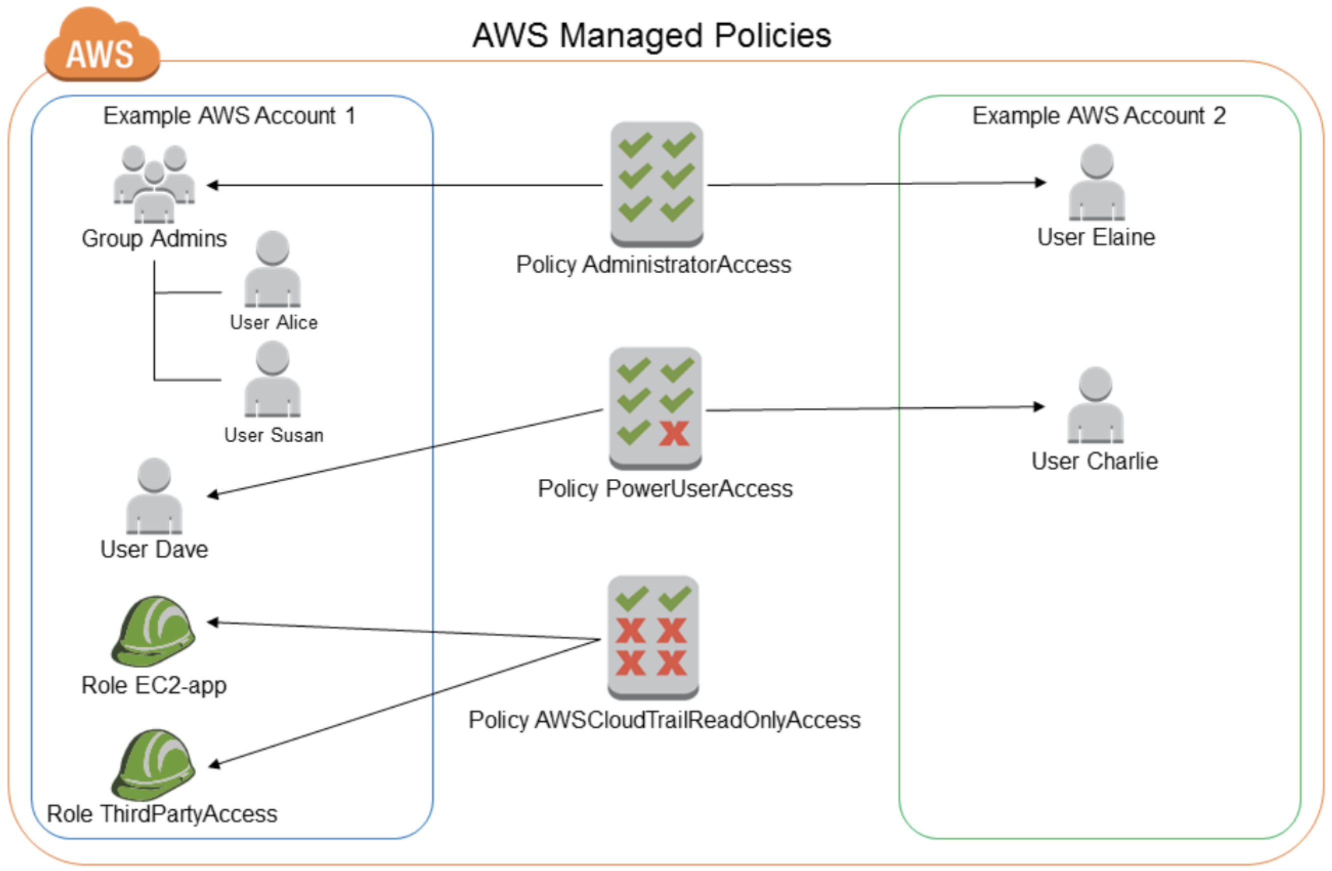
IAM - Introduction

- AWS Identity and Access Management (IAM)
- Manage access to AWS services and resources **securely**
- Create and manage AWS users
- Allow/deny access to AWS resources
- Ensure only **authenticated** and **authorized** users can access resources

IAM – Core components

- Users
- Group
- Role
- Policy

IAM - Examples



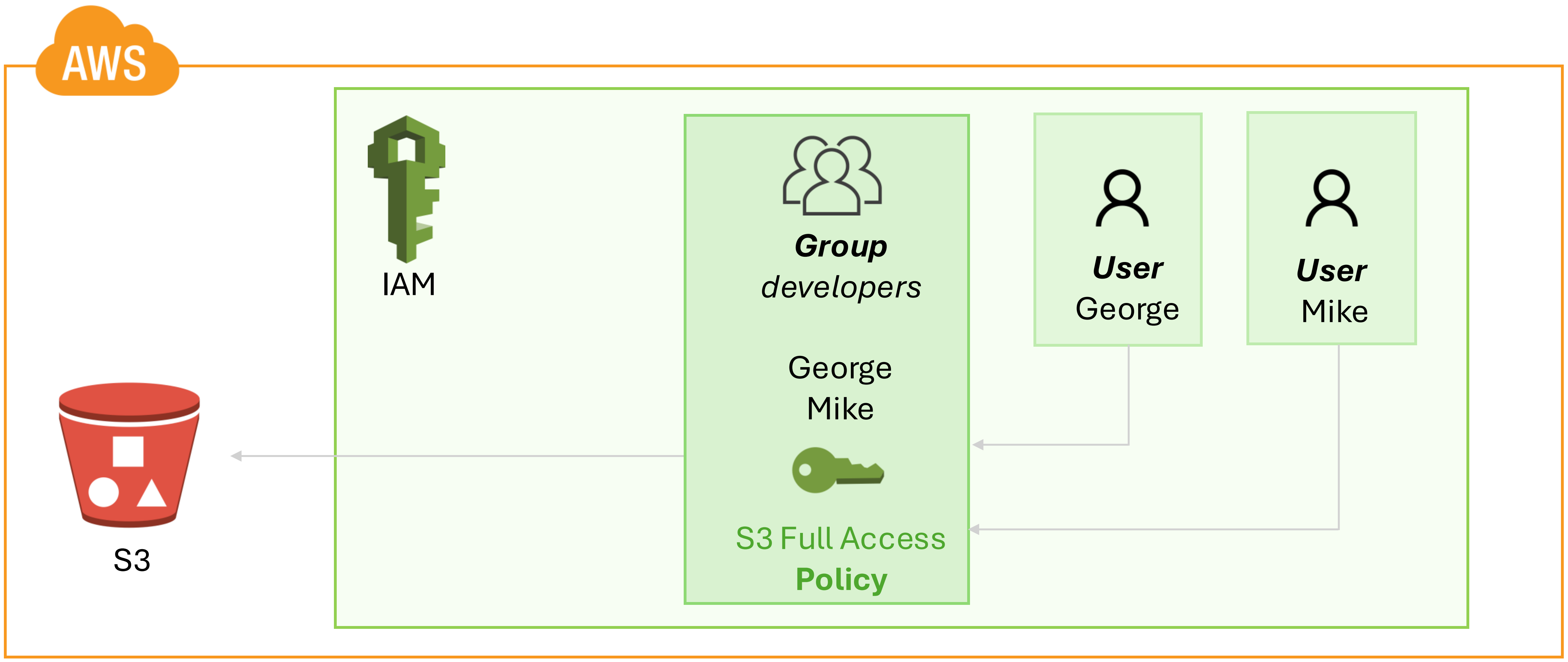
IAM – Key Features

- Centralized Control of AWS Account: Given a central area to manage users and security credentials.
- Shared Access to your AWS account: Grant other people permission to use resources.
- Granular Permissions: Grant different permissions to different people for different resources in details
- Identity Federation: Support for identity federation like Active Directory, Facebook, LinkedIn, etc. That means allowing users to access resources in AWS using their existing credentials.
- Multifactor Authentication (MFA): Add two-factor authentication.
- Temporary Access for Users: Provide users with temporary access to AWS resources.

Users and Groups

- Users
  - An entity represents a **person** or **service** that interacts with AWS resources.
  - Provide individual **credentials** for accessing AWS services, ensuring accountability and traceability.
  - Each has a name, credentials (passwords, access keys), and permissions assigned through policies.
- Groups
  - A **collection** of users with specific permissions.
  - Manage permissions at a group level rather than at the individual user level.
  - Groups have policies attached to them which apply to all users in the group.

AWS IAM Group



IAM Roles

- **Identity**: An IAM role is an IAM identity that is created with specific permissions.
- **Assumable**: It is intended to be assumable by anyone who needs it.
- **Temporary Credentials**: Provides temporary security credentials for users or services that need to perform specific tasks.
- **Separation of Permissions**: Separates permissions from users and can be assumed by trusted entities within or outside the AWS account.

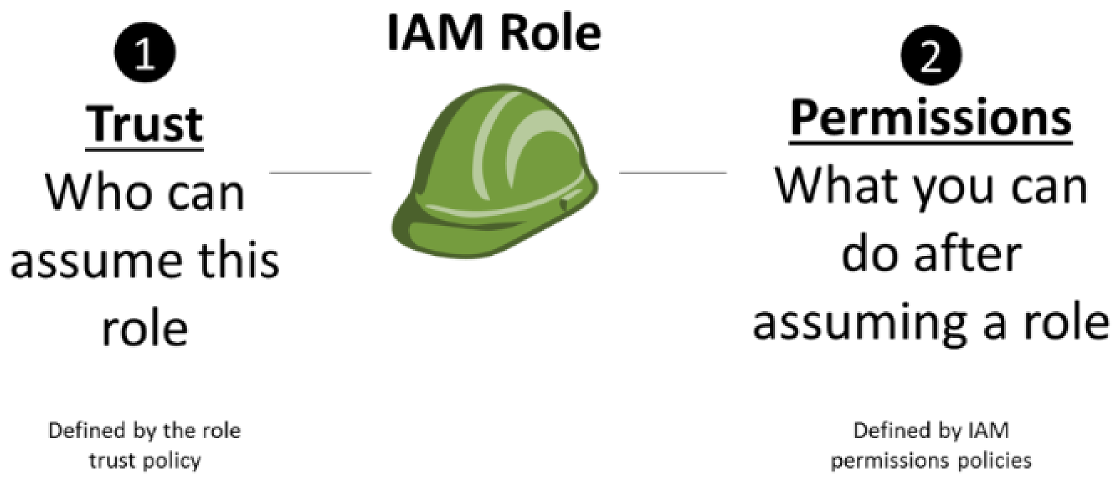
IAM roles

- Components:
  - Trust Policy: Defines who is allowed to assume the role.
  - Permission Policy: Defines what actions and resources the role has access to.
- Common Use Cases:
  - EC2 Instance Roles: Allow EC2 instances to access AWS services.
  - Cross-Account Access: Allow users from one AWS account to access resources in another account.
  - Service Roles: Allow AWS services like Lambda or ECS to access other AWS services.

IAM role

An IAM user and role have one thing in common, permission policies.

When the service in the trust policy assumes the role, AWS STS (Security Token Service) returns **temporary** tokens (access key id, secret access key, and **session token**), and those tokens are rotated automatically.


IAM Role
1
Trust
Who can assume this role
Defined by the role trust policy
2
Permissions
What you can do after assuming a role
Defined by IAM permissions policies

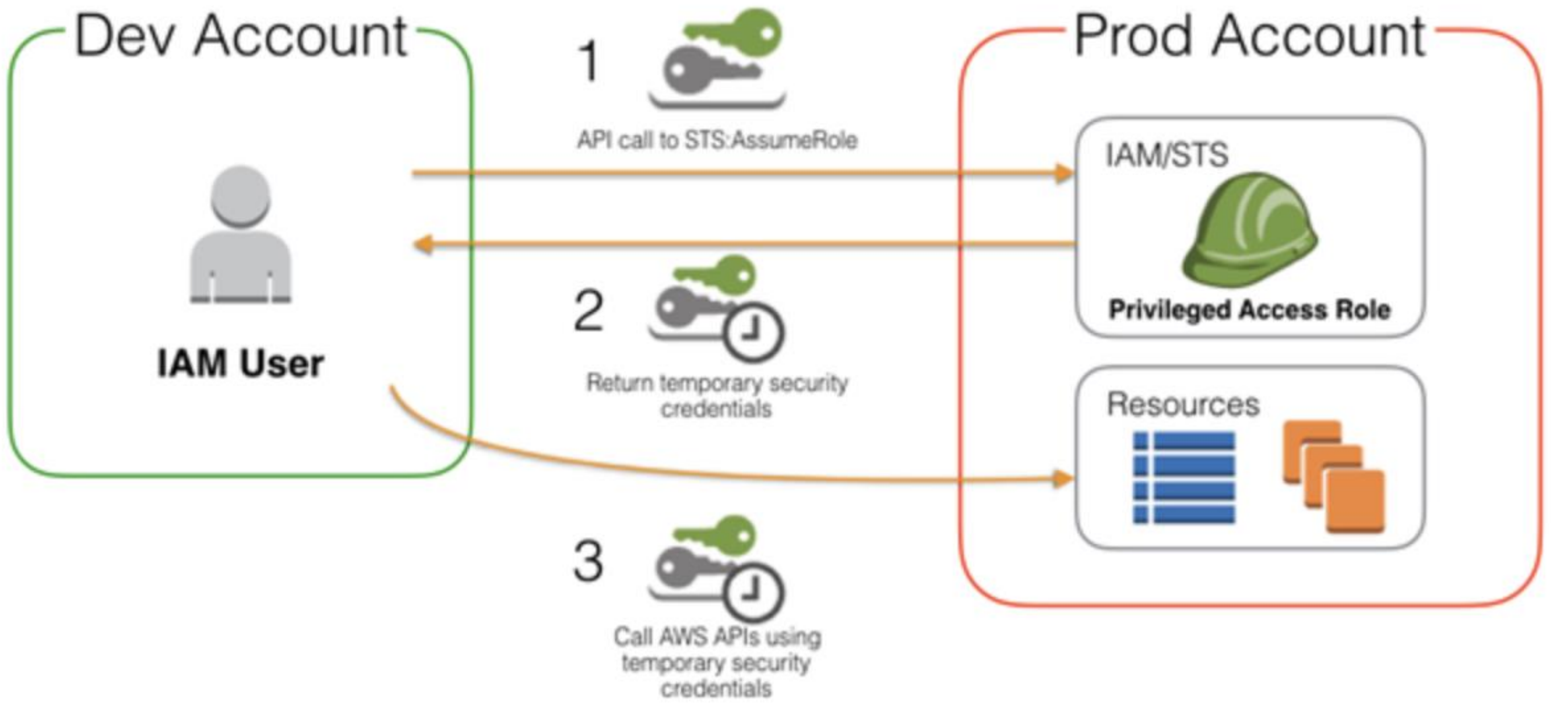
AWS Security Token Service (AWS STS)

AWS **STS** is a web service that enables you to **request temporary security credentials** for **IAM roles** or **federated users**. These credentials are used to access AWS services securely.

The temporary credentials consist of:

1. **Access key ID** - A unique identifier for the temporary credentials like username.
2. **Secret access key** - **A secret string l**ike password.
3. **Session token** - A token used to **validate** and **authenticate** the temporary credentials.
4. **Duration** - **R**ange from **15 minutes (minimum)** to **12 hours (default for most use cases).**

AWS Security Token Service (AWS STS)


Dev Account
IAM User
1
API call to STS:AssumeRole
2
Return temporary security credentials
3
Call AWS APIs using temporary security credentials
Prod Account
IAM/STS
Privileged Access Role
Resources

https://blog.knoldus.com/deep-dive-aws-temporary-security-credentials-assumerole-and-iam-role

IAM Role – Automatic Credential Process

- AWS services (e.g., **EC2**) automatically make an **implicit `AssumeRole` call** to **AWS STS** to fetch temporary credentials.
- Temporary credentials include **Access Key ID**, **Secret Access Key**, and **Session Token**.
- This process happens **behind the scenes** without developer intervention.
- The credentials are used to access resources (e.g., **S3**) securely.
- AWS handles **automatic credential rotation** every few hours, eliminating the need for manual rotation.
- Simplifies and secures resource access for AWS-managed services.

IAM User vs Role

IAM User	**IAM Role**
IAM entity assigned to a **person**.	IAM entity assigned to another **service**. It has a trust policy that specifies what services can use the role.
Credentials are permanent. Not recommended.	**Credentials are temporary**. Credentials are generated by AWS STS. It has an extra token "aws_session_token".
Not a good practice. Feasible for single user accounts and a small team in 10	Federated users are a practical approach that uses IAM role and STS to get access to AWS

Identity Providers and Federation

- Identity Provider (IdP):
  - Creates, maintains, and manages identity information for principals (users or services).
  - Provides authentication services to relying applications within a federated or distributed network.
- Federation: A process of
  - **Establishing trust relationships** between different identity domains (e.g., companies, organizations, or service providers).
  - Allowing users from one domain to access resources in another domain without creating separate accounts.
  - Uses standardized protocols like **SAML**, **OIDC**, or **OAuth** to enable authentication and identity sharing.

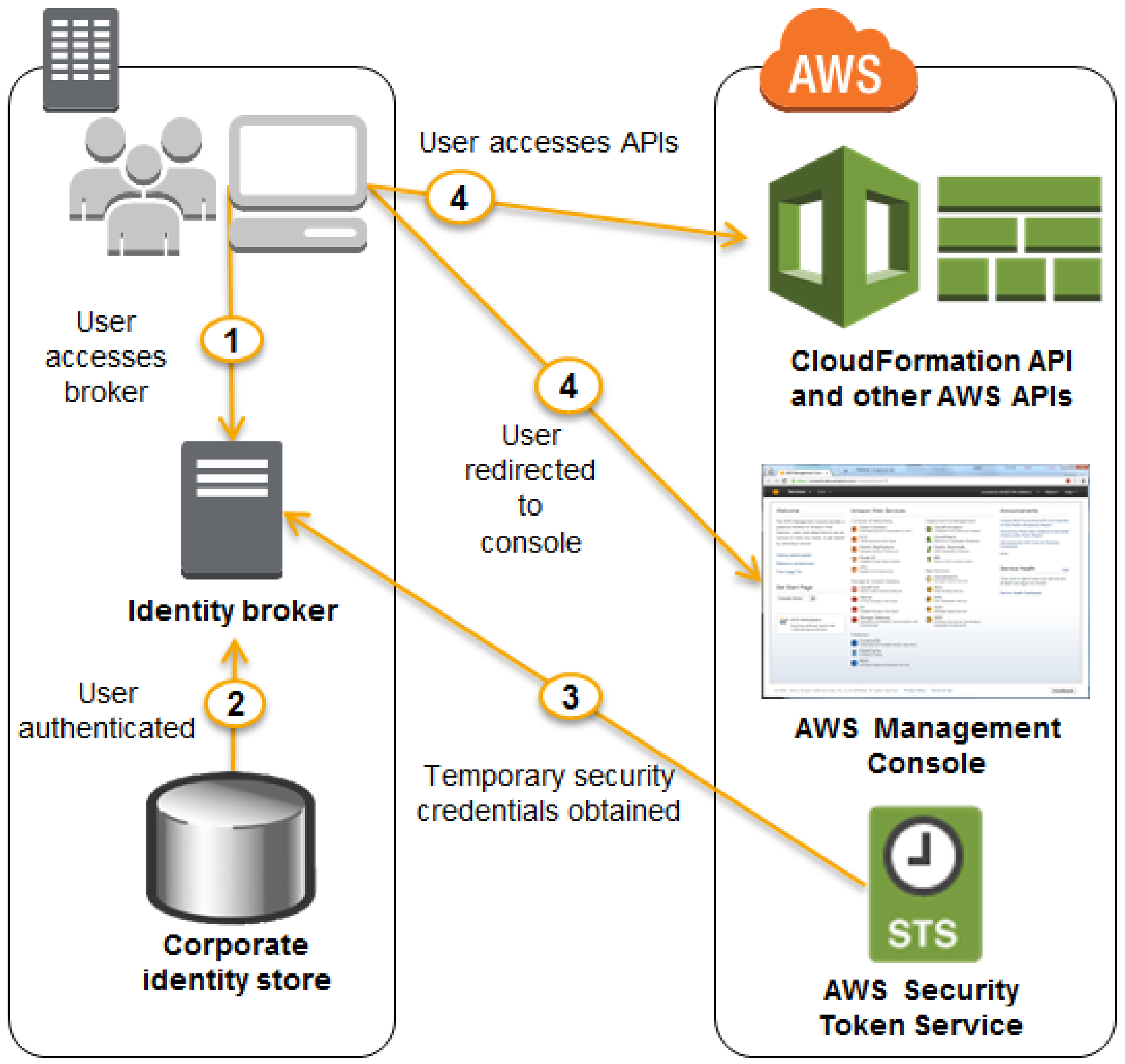
Identity Providers and Federation

- Purpose
  - Single Sign-On (SSO): Enables users to sign in once and access multiple systems, including AWS, without needing multiple credentials
  - Centralized Identity Management: Allows organizations to manage user identities and access from a central directory, such as Microsoft Active Directory, Google, or other third-party identity providers.
- Use cases
  - **Enterprise Integration**: Integrate AWS with corporate directories.
  - **Partner Access**: Grant temporary access to external partners using their own credentials.
  - **SSO for AWS Resources**: Simplify access to the AWS Management Console, CLI, SDK, and internal applications.

Identity Federation in AWS

- **Identity Federation**:
  - Enables external identities from corporate IdPs (e.g., Active Directory, Okta) to access AWS resources securely.
  - Uses protocols like **SAML** or **OIDC** for federated login.
- **Federated Users (External Identities)**:
  - Managed outside AWS (e.g., corporate directories or external IdPs).
  - Access AWS resources via IAM Role.

Federated users and STS


AWS
User accesses APIs
4
CloudFormation API and other AWS APIs
User accesses broker
1
Identity broker
4
User redirected to console
AWS Management Console
User authenticated
2
3
Temporary security credentials obtained
Corporate identity store
STS
AWS Security Token Service

Policy

- A policy is a JSON document that defines permissions to allow or deny actions on AWS resources.
- Control Access: Ensure users and roles have the appropriate access to AWS resources.
- Allows for fine-grained access control, improving security and adherence the principle of least privilege
- Enables the use of conditions to dynamically adjust permissions based on factors such as time, IP address, or multi-factor authentication(MFA).
- Conditions allow to specify when a policy is in effect, using key-value pairs to control access dynamically.
- Use cases
    - Restricting Actions by IP: Allowing access only from specific IP ranges.
    - Time-based Access: Granting access during specific times or days.
    - Resource-level permissions: Specifying permissions for particular resources or subsets of resources.

Policy

- **Identity-based** policies – Attach policies to IAM identities (users, groups, or roles)
  - Managed Policies:
    - AWS Managed Policies: Predefined policies created and managed by AWS
    - Customer Managed Policies: Created and managed by users.
    - Shared
  - Inline policies:
    - Attach directly to a single identity and belong to this identity.
    - Not shared.
- **Resource-based** policies:
  - Attach directly to a single resource(such as S3, SQS, SNS) and belong to this resource.
  - Have a **principal** element that defines who can the actions defined.

Policy Syntax

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow" | "Deny",
      "Action": "service:action" | ["service:action1", "service:action2"],
      "Resource": "arn:aws:service:region:account-id:resource" | ["arn:aws:service:resource"],
      "Principal": {"AWS": "arn:aws:iam::123456789012:user/UserName" | "Service": ... | "Federated":},
      "Condition": {
        "ConditionOperator": {
          "ConditionKey": "ConditionValue"
        }
      }
    }
  ]
}
```

Policy Syntax

```
{
  "NotAction": "service:action" | [
    "service:action1",
    "service:action2"
  ],
  "NotResource": "arn:aws:service:region:account-id:resource" | [
    "arn:aws:service:resource1",
    "arn:aws:service:resource2"
  ]
}
```

Policy Syntax

```
{
  "NotPrincipal": {
    "AWS": "arn:aws:iam::account-id:root" | [
      "arn:aws:iam::account-id:root",
      "arn:aws:iam::123456789012:user/JohnDoe"
    ],
    "Service": "ec2.amazonaws.com" | [
      "ec2.amazonaws.com",
      "lambda.amazonaws.com"
    ],
    "Federated": "cognito-identity.amazonaws.com" | [
      "cognito-identity.amazonaws.com",
      "www.amazon.com"
    ]
  }
}
```

Policy Example

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::my-example-bucket/*"
    }
  ]
}
```

Policy Example

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:StartInstances",
      "Resource": "arn:aws:ec2:us-west-2:123456789012:instance/*",
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": "203.0.113.0/24"
        }
      }
    }
  ]
}
```

Policy Example

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "NotAction": "s3:DeleteObject",
      "Resource": "arn:aws:s3:::example-bucket/*"
    },
    {
      "Effect": "Allow",
      "NotPrincipal": { "AWS": "arn:aws:iam::123456789012:role/Admin"},
      "Action": "ec2:StartInstances",
      "Resource": "*"
    }
  ]
}
```

Multiple Statements or Policies

- Everything is denied **by default** unless explicitly allowed.
- If **any applicable policy** (identity-based or resource-based) explicitly **allows** an action, AWS considers allowing it.
- If **any** policy explicitly **denies** the action, that **deny takes precedence**, even if other policies allow it.
- AWS evaluates **all relevant statements and policies simultaneously**

AWS Identity Center (AWS SSO)

- AWS Identity Center is a centralized service for managing user access to multiple AWS accounts, cloud applications, and custom apps using **Single Sign-On (SSO)**.
- **AWS Identity Center** focuses on managing **user access and roles** across AWS accounts and apps.
- **Centralized User Management:** Manage user access across AWS accounts and third-party applications.
- **Single Sign-On (SSO):** One-click access to AWS accounts, CLI, and cloud apps.
- **Integration with Identity Providers:** Federate with corporate directories (e.g., Active Directory, Okta).
- **Permission Sets:** Predefined roles with fine-grained permissions.

AWS Organizations

- AWS Organizations is a service that enables centralized management of multiple AWS accounts, offering consolidated billing, governance, and resource sharing across your organization.
- **AWS Organizations** focuses on managing **accounts, billing, and governance** within a multi-account structure.
- **Account Management:** Create and manage multiple AWS accounts under a single organization.
- **Service Control Policies (SCPs):** Enforce governance and compliance across accounts.
- **Consolidated Billing:** Centralized billing with cost tracking for individual accounts.
- **Resource Sharing:** Share resources like Reserved Instances across accounts.

AWS CLI

- AWS Command Line Interface (AWS CLI) is a unified tool to manage AWS services.
- Installation: https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html

AWS CLI

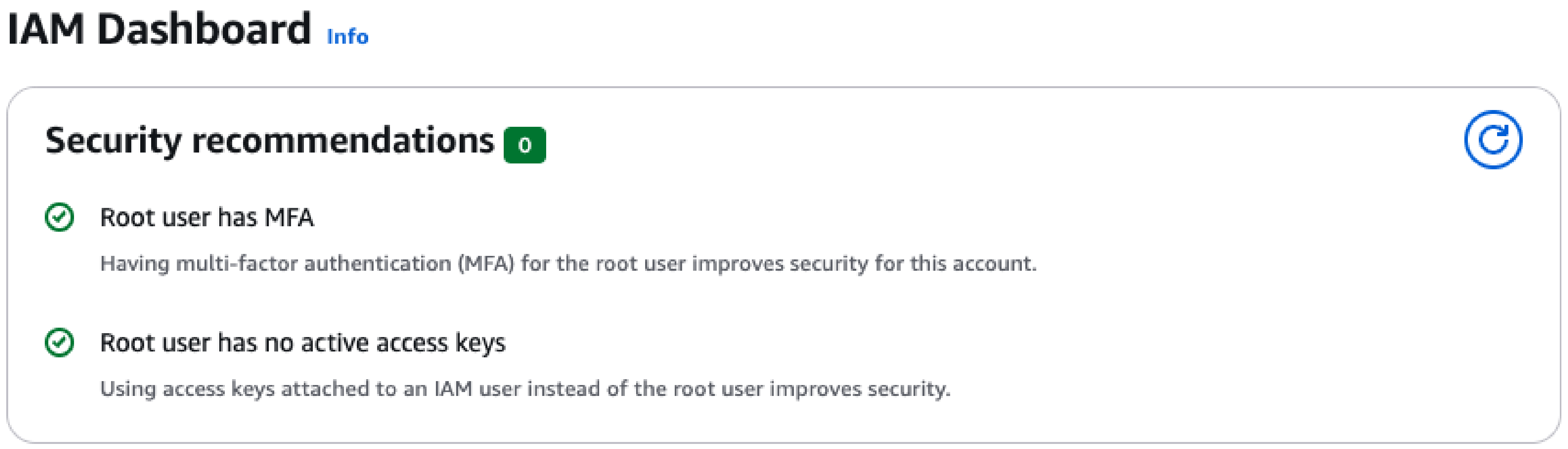
- Common commands

```
aws configure
aws --version
aws iam list-users
aws s3 ls
aws ec2 describe-instances
aws lambda list-functions
aws apigateway get-rest-apis
aws rds describe-db-instances
aws docdb describe-db-clusters
```

Best practices for IAM Security

1. Enable Multi-Factor Authentication (MFA) for users.
2. Use a user for deployment.
3. Never use the root account for deployment.
4. Disable the access keys of the root account.
5. Least Privilege Principle
6. Use Groups to Assign Permissions
7. Use IAM Roles for Applications
8. Regularly Rotate Credentials
9. Implement Strong Password Policies
10. Monitor and Audit IAM Activity

Root Account



IAM Dashboard Info

Security recommendations 0

✓ Root user has MFA

Having multi-factor authentication (MFA) for the root user improves security for this account.

✓ Root user has no active access keys

Using access keys attached to an IAM user instead of the root user improves security.

Least Privilege Principle

- Least Privilege Principle: Grant only the permissions necessary for users to perform their job functions.
- Actions:
  - Regularly review and audit permissions.
  - Use IAM policies to grant specific permissions rather than broad ones.
  - Create Roles with limited permissions for specific tasks.

Least Privilege Principle

- S3ReadOnlyAccess

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
    "Effect": "Allow",
    "Action": [
     "s3:GetObject",
     "s3:ListBucket"
    ],
    "Resource": [
     "arn:aws:s3:::example-bucket",
     "arn:aws:s3:::example-bucket/*"
    ]
  }
 ]
}
```

Use IAM Roles for Applications

- Provide temporary credentials for applications running on AWS services without embedding long-term credentials in code.
- Create IAM roles with specific permissions for your applications
- Assign roles to AWS services that need to access resources

Monitor and Audit IAM Activity

- Monitoring: Continuous tracking of user activities and API calls to detect and respond to security threats.
- Auditing: Systematic review of user activities and resource configurations to ensure compliance with security policies and standards.
- Purpose:
    - Security: Detect unauthorized access or misconfigurations that could lead to security breaches
    - Compliance: Ensure adherence to regulatory requirements and internal policies
    - Accountability: Maintain detailed records of user actions for accountability and troubleshooting

AWS CloudTrail

- Logs, monitors, and retains account activity related to actions across AWS infrastructure.
- **Event History**:
  - Provides a detailed history of **AWS API calls** for your account.
  - Includes API calls made through:
    - AWS Management Console.
    - AWS SDKs.
    - AWS CLI (Command Line Interface).
    - Other AWS services.
- **Trail Configuration**:
  - You can create a **trail** to log events and deliver them to an Amazon S3 bucket for long-term storage.
  - Trails can also be integrated with **Amazon CloudWatch Logs** for near-real-time monitoring and alerting.

AWS Config

- AWS Config:
  - Enables you to **assess, audit, and evaluate** the configurations of your AWS resources.
  - Continuously monitors and records your AWS resource configurations and allows you to automate the evaluation of recorded configurations against desired configurations.
- **Configuration History**:
  - Provides a detailed **timeline of changes** to the configurations of AWS resources.
  - Helps you understand the state of resources at any point in time for troubleshooting or auditing.
- **Compliance Management**:
  - Automates the evaluation of resource configurations against custom **compliance rules** or **AWS Config Managed Rules**.
  - Enables continuous compliance monitoring and helps ensure resources adhere to organizational policies and security requirements.

IAM Access Analyzer

- IAM Access Analyzer:
  - Identify the resources in your organization and accounts, such as Amazon S3 buckets or IAM roles, that are shared with an external entity.
  - Continuously monitors and logs findings for new and updated policies.
- **Policy Analysis**:
  - Analyzes resource-based policies to identify permissions granted to external entities outside your account or organization.
  - Helps you validate least-privilege access and prevent unintended sharing of sensitive resources.
- **Continuous Monitoring**:
  - Automatically updates findings when resource policies are created or modified.
  - Provides actionable insights to review and remediate access issues.

References

- https://docs.aws.amazon.com/
- ChatGPT: https://chatgpt.com/
- Google AI: https://gemini.google.com/app