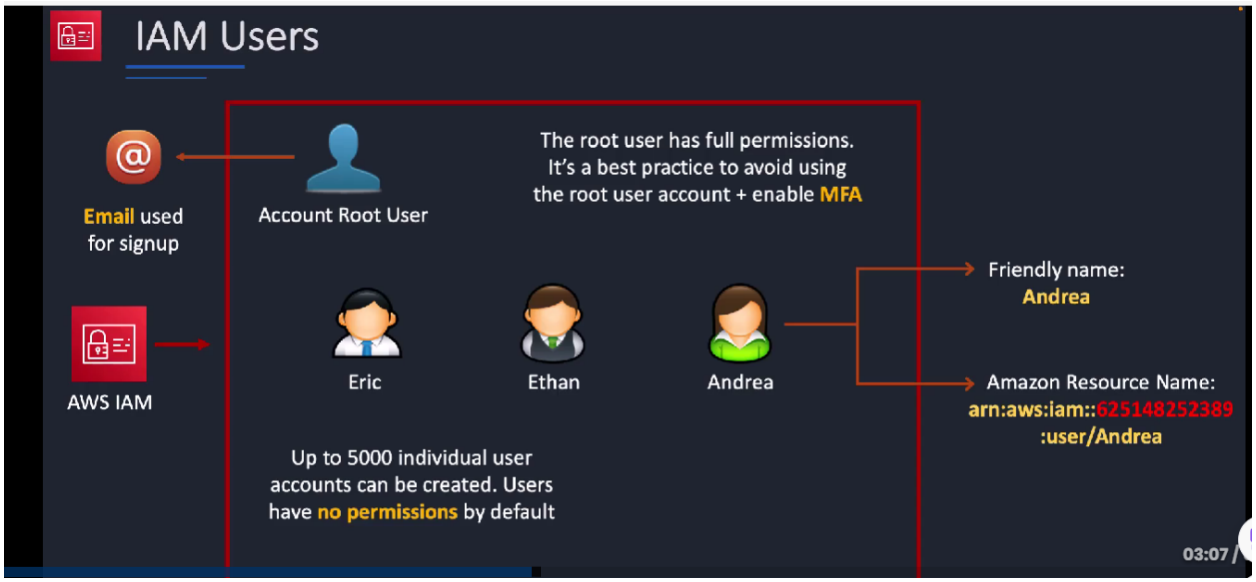
AWS Identity and Access Management (IAM) is a service provided by Amazon Web Services (AWS) that helps you securely control access to AWS resources. IAM enables you to manage users, groups, and permissions within your AWS environment. Here are some key concepts and features of AWS IAM:

1. Users:Users represent individuals or entities within your organization who interact with AWS resources. Each user has a unique set of security credentials.



2. Groups: Groups are collections of users, and they make it easier to manage permissions for multiple users simultaneously. Instead of attaching policies to individual users, you can attach policies to groups.

3. Roles:IAM roles are similar to users but are not associated with a specific person. Roles are meant to be assumed by entities such as AWS services, EC2 instances, or applications. Roles have policies attached to them to define the permissions.

4. **Policies:Policies are JSON documents that define permissions. They are attached to users, groups, or roles and specify what actions are allowed or denied on what resources. AWS provides predefined policies that you can use, and you can also create custom policies.

5. **Permissions:** Permissions define what actions users, groups, or roles are allowed or denied to perform on AWS resources. These actions include things like creating or deleting resources, and they are defined in IAM policies.

AWS STS, or Amazon Web Services Security Token Service, is a web service that enables you to request temporary, limited-privilege credentials for AWS Identity and Access Management (IAM) users or for users that you authenticate (federated users).

The main purpose of AWS STS is to provide a way to assume a role and obtain temporary security credentials that can be used to access other AWS services. This is useful in scenarios where you want to grant temporary access to your AWS resources to entities or applications that are not directly associated with an IAM user or role in your AWS account.

Here are some key concepts related to AWS STS:

1. Identity Federation: AWS STS supports identity federation, allowing you to grant temporary access to your AWS resources to users authenticated by an external identity provider (IdP), such as Active Directory, Facebook, or Google.

2. Roles:You can define IAM roles with specific permissions, and then use AWS STS to assume those roles. This enables you to delegate permissions to users, applications, or services without having to share long-term security credentials.

3. Temporary Security Credentials: When you assume a role, AWS STS provides you with temporary security credentials (access key, secret key, and session token). These credentials have a limited lifespan, typically ranging from a few minutes to several hours, reducing the risk associated with long-term credential exposure.

4. Cross-Account Access:AWS STS allows users in one AWS account to assume a role in another account, enabling cross-account access to resources.

To use AWS STS, you typically make API requests to the `AssumeRole` operation, specifying the Amazon Resource Name (ARN) of the role to assume. You then receive temporary security credentials that can be used to make subsequent AWS API requests.

Here's a basic example of assuming a role using the AWS CLI:

```bash
aws sts assume-role --role-arn
arn:aws:iam::account-id-with-role-to-assume:role/role-name --role-session-name
"session-name"
```

This command assumes the specified IAM role and returns temporary security credentials that can be used for subsequent AWS API requests.

Keep in mind that AWS services and features are updated regularly, so it's a good idea to refer to the [official AWS STS documentation](https://docs.aws.amazon.com/STS/latest/APIReference/welcome.html) for the most up-to-date information and usage details.

AWS Multi-Factor Authentication (MFA) is a security feature that adds an extra layer of protection to your AWS account. It requires users to present two or more separate forms of identification (factors) to gain access. The primary factors are typically something you know (e.g., a password), and the secondary factor is something you have, such as an MFA device.

Here's how AWS MFA works:

1. Enable MFA for an IAM User:
   - You can enable MFA for an IAM user in the AWS Management Console or by using the AWS Command Line Interface (CLI) or SDKs. To enable MFA, you associate a virtual MFA device or a hardware MFA device with the IAM user.

2. Virtual MFA Device:
   - A virtual MFA device is an app that generates time-based one-time passwords (TOTPs). Common apps include Google Authenticator or Authy. When you enable MFA for an IAM user, you scan a QR code with the MFA app, and it generates a time-based code that you enter as a confirmation.

3.Hardware MFA Device:

- Alternatively, you can use a hardware MFA device, which is a physical device that generates time-based codes. AWS supports various MFA devices, and you can associate one with an IAM user.

4. MFA-Protected API Calls:
- Once MFA is enabled for an IAM user, the user is required to provide both their regular IAM user credentials (username and password) and the current time-based code from the MFA device when signing in or making privileged API calls.

5. AWS CLI with MFA:
- When using the AWS CLI with an IAM user that has MFA enabled, you typically generate temporary security credentials by assuming a role with the `--serial-number` and `--token-code` parameters. This includes the ARN of the MFA device and the current time-based code.

Example AWS CLI command to assume a role with MFA:
```bash
aws sts assume-role --role-arn arn:aws:iam::account-id:role/role-name --role-session-name "session-name" --serial-number arn:aws:iam::account-id:mfa/user-name --token-code code-from-mfa
```

MFA provides an extra layer of security, especially for users with administrative privileges or access to sensitive resources. It helps protect against unauthorized access even if the user's password is compromised.

Keep in mind that AWS MFA is just one part of a comprehensive security strategy, and it's important to implement other security best practices, such as regular password updates, least privilege principles, and monitoring for suspicious activities in your AWS environment.

7. Identity Federation: IAM allows you to grant access to your AWS resources to users who are authenticated by an external identity provider (IdP). This is useful for integrating AWS with your existing identity management systems.

8. Access Key and Secret Access Key: Users can have access keys associated with their IAM accounts, which are used for programmatic access to AWS resources via APIs. Access keys consist of an access key ID and a secret access key.

9. IAM Dashboard:AWS provides a web-based console for IAM, allowing administrators to manage users, groups, roles, and policies through a graphical interface.

IAM is a crucial component of AWS security, allowing you to implement the principle of least privilege, where users and entities are granted only the permissions they need to perform their tasks, and no more. Properly configuring IAM is essential for securing your AWS environment.

AWS POLICIES

Amazon Web Services (AWS) provides a variety of services and features that allow users to configure and manage their infrastructure. AWS uses a set of policies to control access to its resources and services. These policies are defined and enforced using AWS Identity and Access Management (IAM). IAM policies are written in JSON (JavaScript Object Notation) format and consist of statements that define the permissions for different actions on AWS resources.

Here are some key types of policies in AWS:

Identity-Based Policies:

- User Policies: These policies are attached to IAM users and define what actions the user is allowed or denied to perform.
- Group Policies: IAM groups allow you to organize IAM users and apply policies to all members of a group. Group policies specify the permissions for all users within that group.
- Role Policies: IAM roles are used to delegate access to AWS resources to users, applications, or services that normally don't have access. Role policies define the permissions for the actions that can be taken when someone assumes the role.

Resource-Based Policies:

- Bucket Policies: In Amazon S3, bucket policies are resource-based policies that define permissions for the bucket itself. These policies can control access at the bucket level and are written in JSON.
- Queue Policies: In Amazon Simple Queue Service (SQS), you can attach policies to queues to control who can send messages to the queue, receive messages, or perform other actions.

Managed Policies:

- AWS provides a set of managed policies that you can directly attach to IAM users, groups, or roles. Examples include "AmazonS3FullAccess" or "AmazonEC2ReadOnlyAccess." These policies are predefined by AWS and are designed to cover common use cases.

Inline Policies:

- These policies are directly embedded into an IAM user, group, or role and are part of the IAM resource's definition. They are defined in the same JSON document as the IAM entity to which they are attached.

Permission Boundaries:

- A permission boundary is an advanced feature in IAM that sets the maximum permissions that an IAM entity (user or role) can have. It helps

to control the maximum permissions granted to a user, even if the policies attached to that user provide broader permissions.

AWS Organizations Policies:

- AWS Organizations allow you to consolidate multiple AWS accounts into an organization that you create and centrally manage. Service control policies (SCPs) are used within AWS Organizations to set fine-grained permissions across the organization.

IAM policies use a combination of elements to specify permissions, including the "Effect" (Allow or Deny), "Action" (the specific API actions), "Resource" (the AWS resource or resources to which the actions apply), and "Condition" (optional conditions that must be met for the policy to be in effect).

It's important to carefully craft policies to ensure that users and services have the appropriate level of access to AWS resources while maintaining security and compliance. Regularly reviewing and auditing policies is also essential to ensure that access permissions align with organizational requirements.

In AWS Identity and Access Management (IAM), permission boundaries are an advanced feature that sets the maximum permissions a user or role can have. They define the extent to which an IAM entity (user or role) can delegate access to other entities. By setting a permission boundary, you can control the maximum level of permissions that can be granted, even if policies attached to the entity would allow broader access.

Here are key points to understand about permission boundaries in AWS:

Setting a Permission Boundary:

- When you create or modify an IAM user or role, you can specify a permission boundary. This is done by attaching an IAM policy to the user or role that includes a "iam:PermissionsBoundary" statement. The value of this statement is the Amazon Resource Name (ARN) of the policy that acts as the permission boundary.

Maximum Permissions:

- The permission boundary acts as a limit on the permissions that an IAM entity can have. Even if the policies directly attached to the user or role grant broader permissions, the effective permissions are limited by the permission boundary.

Policy Evaluation:

- When AWS evaluates permissions for an IAM entity, it considers both the policies attached directly to the entity and the permissions boundary. The effective permissions are the intersection of the two sets of permissions.

Use Cases:

- Permission boundaries are useful in scenarios where you want to delegate administrative responsibilities but still maintain control over the maximum permissions granted. For example, you might have a central security team that defines the permission boundaries for other teams within the organization.

Delegated Administration:

- By using permission boundaries, you can delegate the ability to create and manage IAM users or roles to different teams or individuals without giving them unrestricted access. This helps in enforcing organizational policies and ensuring a least privilege access model.

Hierarchy of Permissions:

- IAM policies can be complex, especially in organizations with multiple teams and roles. Permission boundaries add an additional layer of control, creating a hierarchy of permissions that helps prevent unintended broadening of access.

AWS Organizations Integration:

- Permission boundaries can be used in conjunction with AWS Organizations. When you have an organization in AWS Organizations, you can set service control policies (SCPs) at the root level to further restrict what actions can be performed by IAM entities within the organization.

It's important to note that setting permission boundaries requires careful consideration of the organization's structure and policies. Regular audits of permission boundaries and IAM policies are recommended to ensure that access controls align with security and compliance requirements.

IAM policy evaluation in AWS involves several steps to determine the permissions granted to a user, group, or role. The process considers policies attached to the entity, policies attached to other entities in the resource hierarchy, and any conditions specified in the policies. Here's an overview of the IAM policy evaluation process:

Request Initiation:

- The process begins when an AWS service or user initiates a request to perform an action (e.g., launching an EC2 instance or reading an S3 bucket).

Authentication and Authorization:

- The user or service making the request must be authenticated and authorized. Authentication confirms the identity of the requester, while

authorization determines whether the requester has the necessary permissions to perform the requested action.

User Context:

- AWS uses the identity context of the requester, including the user, group memberships, and any assumed roles, to understand the permissions associated with the entity making the request.

Policy Evaluation:

- IAM policies are evaluated in a specific order to determine the permissions granted to the entity making the request. The process involves the following steps:
    - Identity Policies: Policies attached directly to the user or group are evaluated first.
    - Resource Policies: Policies attached directly to the resource (e.g., S3 bucket policies) are evaluated next.
    - IAM Role Policies: If the user or group assumed a role, the policies attached to that role are evaluated.
    - Permissions Boundary: If a permissions boundary is set for the user or role, it is considered in the evaluation.

Policy Conditions:

- Policies can include conditions that must be met for the policy to be in effect. These conditions can be based on factors such as the current time, the source IP address, or the use of multi-factor authentication. If a condition is present, it is evaluated to determine if it is satisfied.

Deny Overrides Allow:

- If any policy explicitly denies the requested action, the denial takes precedence over any allows. In other words, an explicit deny in any policy

overrides any allows, providing a security principle known as "deny by default."

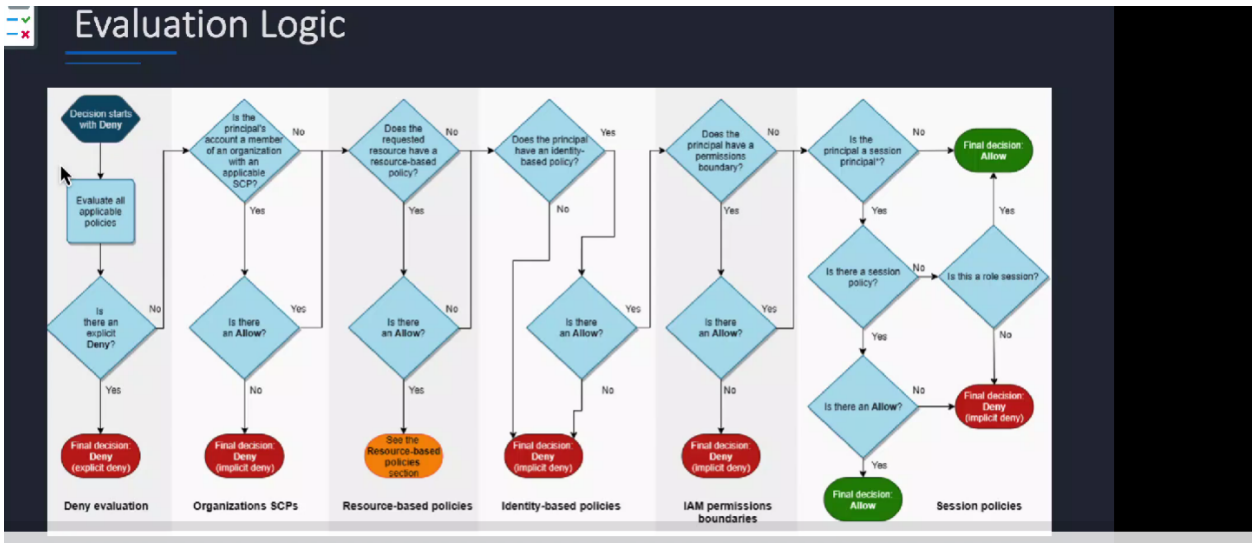
Effective Permissions:

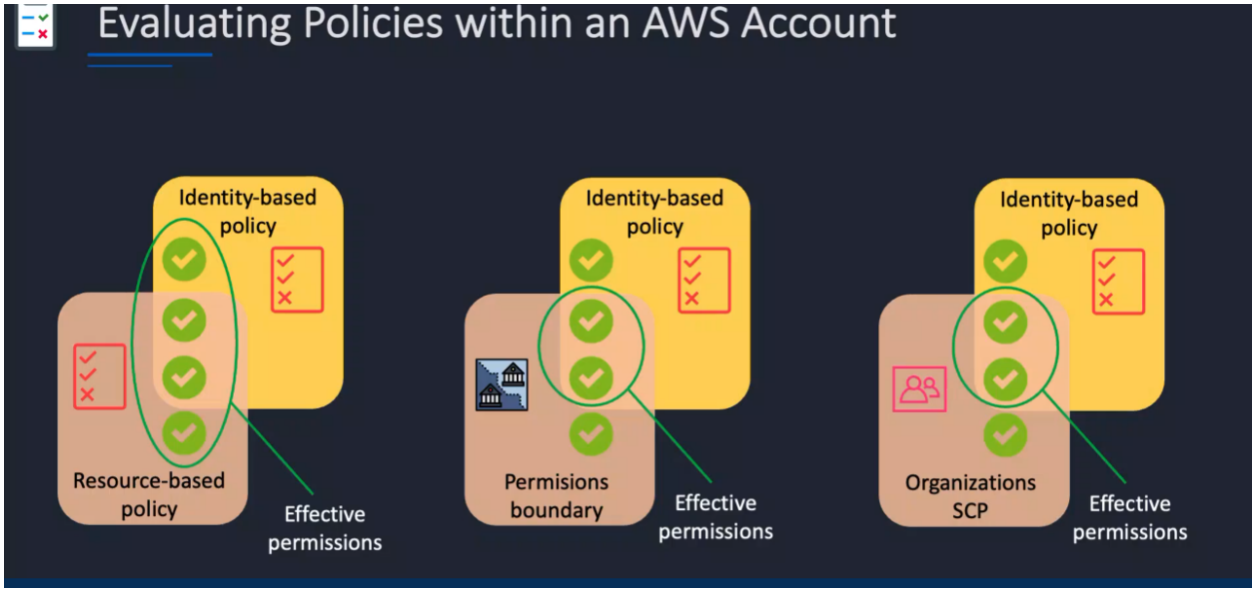
- The effective permissions for the request are the result of combining the permissions granted by all policies, factoring in conditions and explicit denies. These effective permissions are used to determine whether the requested action is allowed or denied.

Request Outcome:

- Based on the evaluation, the request is either allowed or denied. If allowed, the requested action is performed. If denied, the requester receives an error indicating insufficient permissions.

This process ensures that permissions are evaluated in a comprehensive and granular manner, considering the entire policy set and any conditions specified. Regularly reviewing and auditing IAM policies is crucial to maintaining a secure and compliant access control model in AWS.




Evaluating Policies within an AWS Account
Identity-based policy
Resource-based policy
Effective permissions
Identity-based policy
Permisions boundary
Effective permissions
Identity-based policy
Organizations SCP
Effective permissions