

AWS Cloud Security Foundations– AWS-CSF

Dr. Ibrahim Gomaa

Lecture 7 – Securing Access to Cloud Resources



Lecture 7: Securing Access to Cloud Resources

Topics

- IAM fundamentals
- Authenticating with IAM
- Authorizing with IAM
- Examples of authorizing with IAM
- Additional authentication and access management services
- Using AWS Organizations

Demo

- Amazon S3 Cross-Account Resource-Based Policy

Lab

- Using Resource-Based Policies to Secure an S3 Bucket

IAM Fundamentals



AWS Identity and Access Management (IAM)

- Securely shares and controls individual and group access to your AWS resources
- Integrates with many other AWS services
- Supports federated identity management
- Supports granular permissions
- Supports multi-factor authentication (MFA)
- Provides identity information for assurance



AWS Identity and
Access Management
(IAM)

What IAM provides

- **Authentication**

- **Who** is requesting access to the AWS account and the resources in it?
- It's important to establish the identity of the requester through credentials.
- The requester could be a *person* or an *application*; IAM calls them *principals*.

- **Authorization**

- After the requester has been authenticated, **what** should they be allowed to do?
- IAM checks for policies that are relevant to the request to determine whether to allow or deny the request.



IAM overview



User



Group



Role

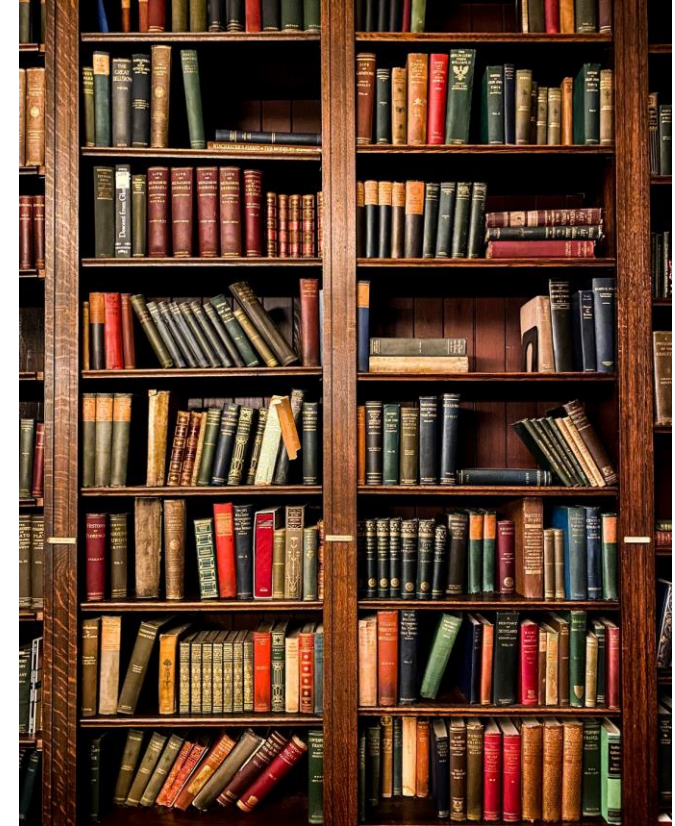


IAM policy

- A person or application that can authenticate with an AWS account
- A collection of IAM users who are granted identical authorization
- An identity used to grant a temporary set of permissions to make AWS service requests
- The document that defines which resources can be accessed and the level of access to each resource

IAM terminology

- *IAM entity*: Used by AWS for authentication (users and roles)
- *IAM identity*: Used to identify and group
 - You can attach a policy to an IAM identity (user, group, or role).
- *IAM resource*: The user, group, role, policy, and identity provider objects that are stored in IAM
 - You can add, edit, and remove resources from IAM.
- *Principal*: A person or application that uses the AWS account root user, IAM user, or IAM role to sign in and make requests to AWS



Requests in IAM

A request is made any time **a principal attempts to use the AWS** Management Console, application programming interface (API), or AWS Command Line Interface (AWS CLI).

The request contains the following information:

- **Actions or operations:** What the principal wants to perform
- **Resources:** The object upon which the actions or operations are performed
- **Principal:** The person or application that sends a request by using a user or role
- **Environment data:** The IP address, user agent, Secure Sockets Layer (SSL) enabled status, or time of day
- **Resource data:** Data related to the resource being requested

Service endpoints

- To connect to an AWS service, you must use the URL of the entry point for that service, known as an *endpoint*.
- The AWS software development kits (SDKs) and AWS CLI use the default endpoint for each service in an AWS Region.
- You can specify alternate endpoints for API requests based on configuration requirements.



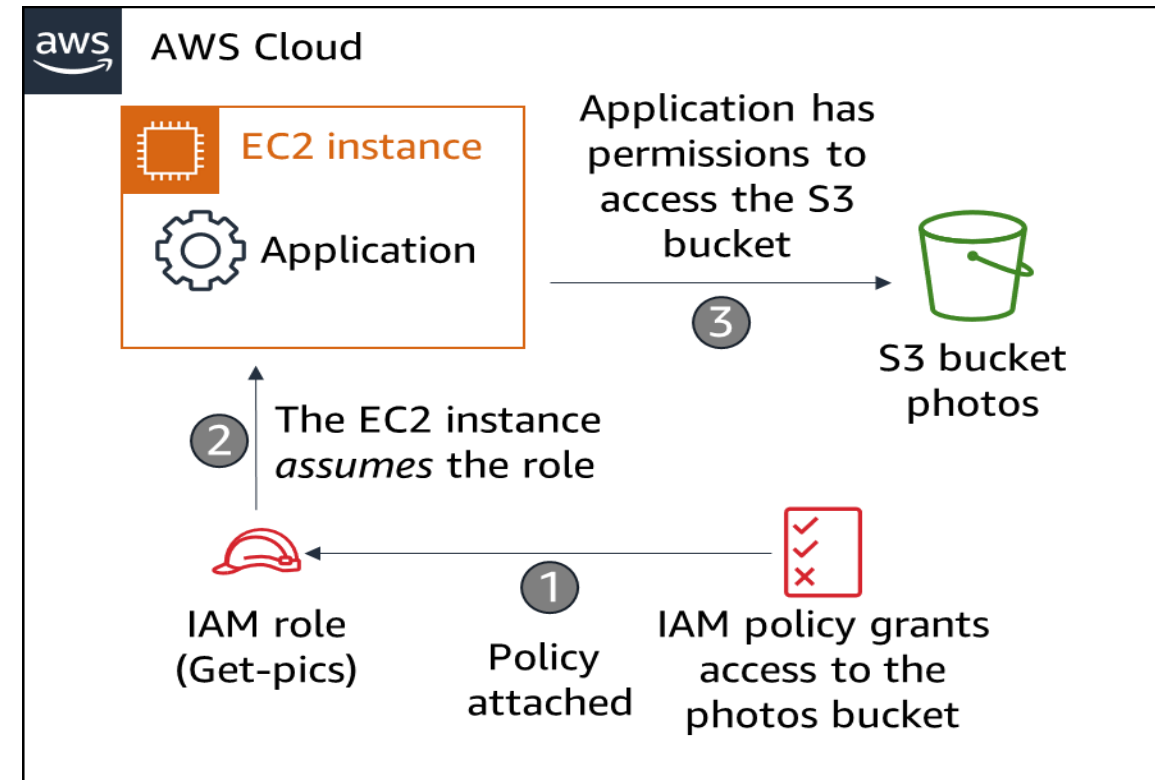
An *example* of an Amazon Elastic Compute Cloud (Amazon EC2) endpoint is `ec2.us-east-1.amazonaws.com`, which is the US East (N. Virginia) service endpoint for Amazon EC2

Authenticating with IAM



AWS Identity and Access Management (IAM)

- IAM role characteristics
 - It provides *temporary* security credentials.
 - A role is **not** uniquely associated with one person.
 - A *person, application, or AWS service* can assume a role.
 - A role is often used to delegate access.
 - The AWS Security Token Service (AWS STS) issues temporary security credentials.
- Common use cases
 - Applications that run on Amazon Elastic Compute Cloud (Amazon EC2)
 - Cross-account access for an IAM user
 - Mobile applications



Authenticating with IAM

Authenticate as an IAM user to gain access

When you define an **IAM user**, you select what *types of access* the user is permitted to use.

Programmatic access

- Authenticate using:
 - Access key ID
 - Secret access key
- Provides AWS CLI and AWS SDK access



AWS CLI



AWS Tools
and SDKs

AWS Management Console access

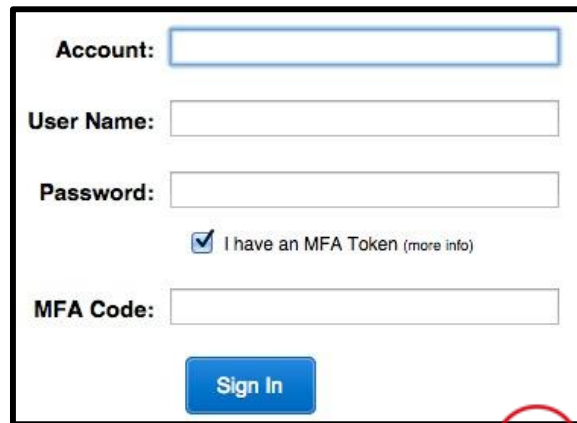
- Authenticate using:
 - 12-digit Account ID or alias
 - IAM username
 - IAM password
- If enabled, **multi-factor authentication (MFA)** prompts for an authentication code.



AWS Management
Console

IAM credentials for authentication

User name and password
(console access)



AWS Management Console



Option

Access key ID and secret access key
(programmatic access)

```
~$ aws configure
AWS Access Key ID [*****O22A]:
AWS Secret Access Key [*****4m8i]:
Default region name [ap-southeast-1]:
Default output format [json]:
```

AWS CLI

ACCESS KEY ID

AKIAIOSFODNN7EXAMPLE

SECRET KEY

wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY

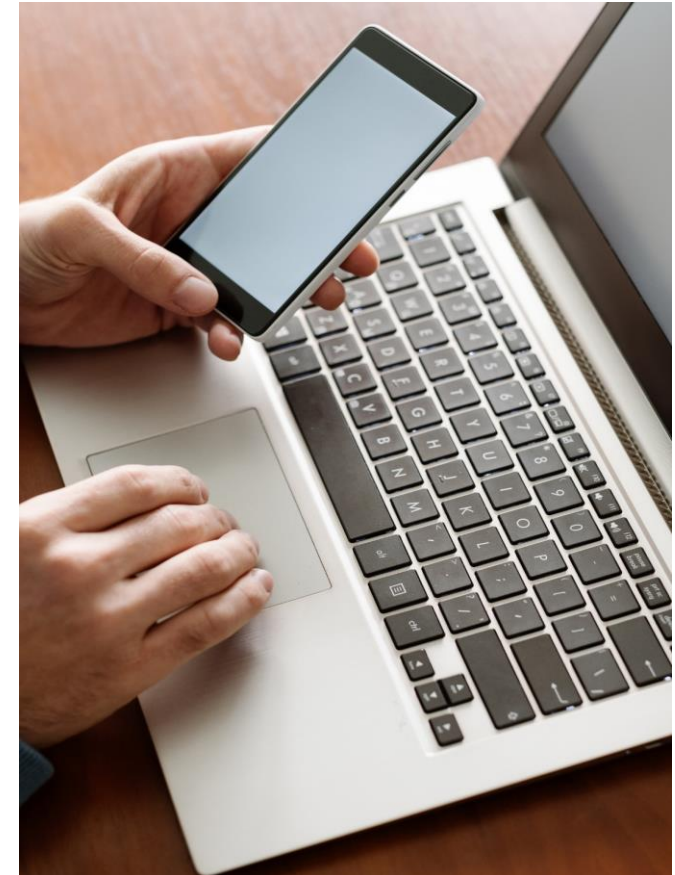
API access



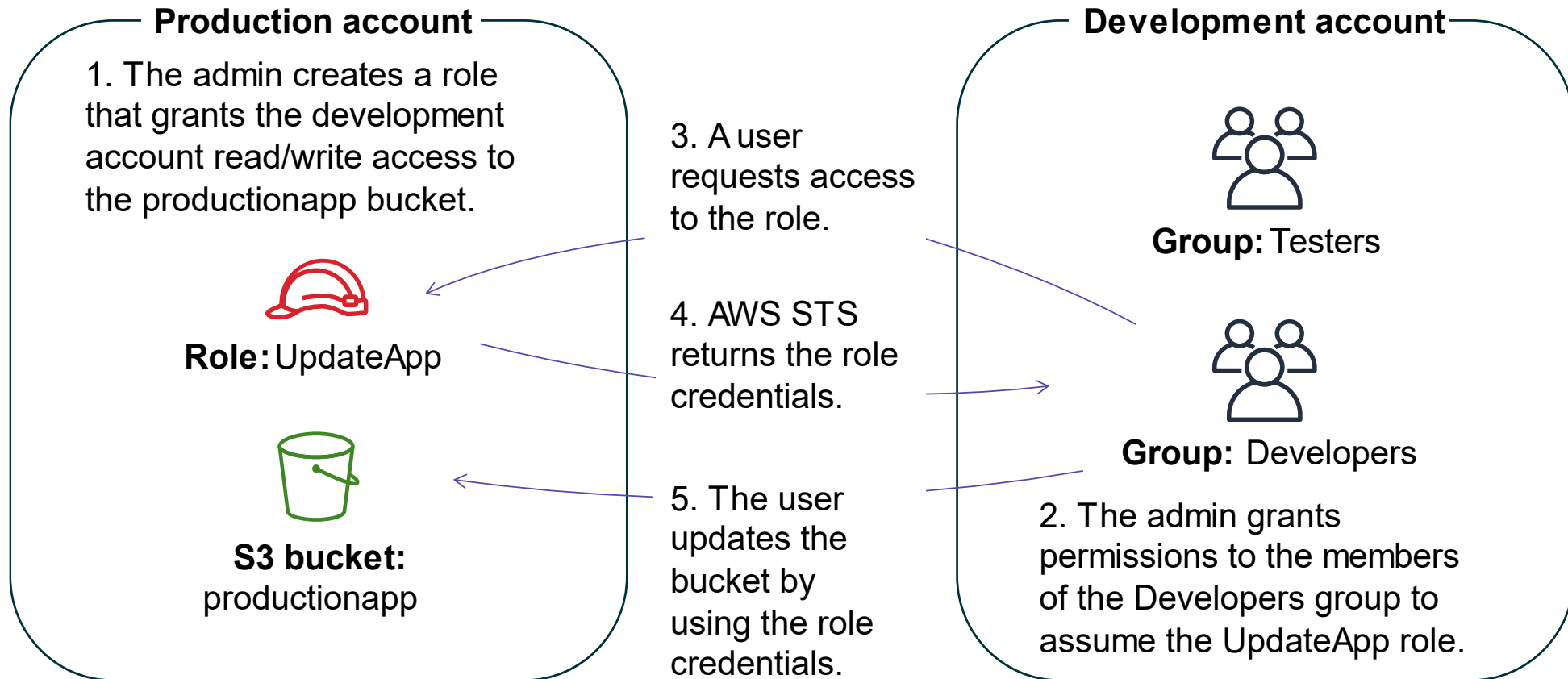
Option

Multi-factor authentication (MFA)

- Adds an extra layer of protection on top of your username and password
 - Users prompted for an authentication code
 - Can be hardware based or a virtual device
- Activate MFA for:
 - AWS Management Console users
 - AWS API users (requires temporary security credentials)
- Examples of MFA devices:
 - Security keys devices (YubiKey, Gemalto)
 - Applications (Google Authenticator, Authy)
 - Hardware devices



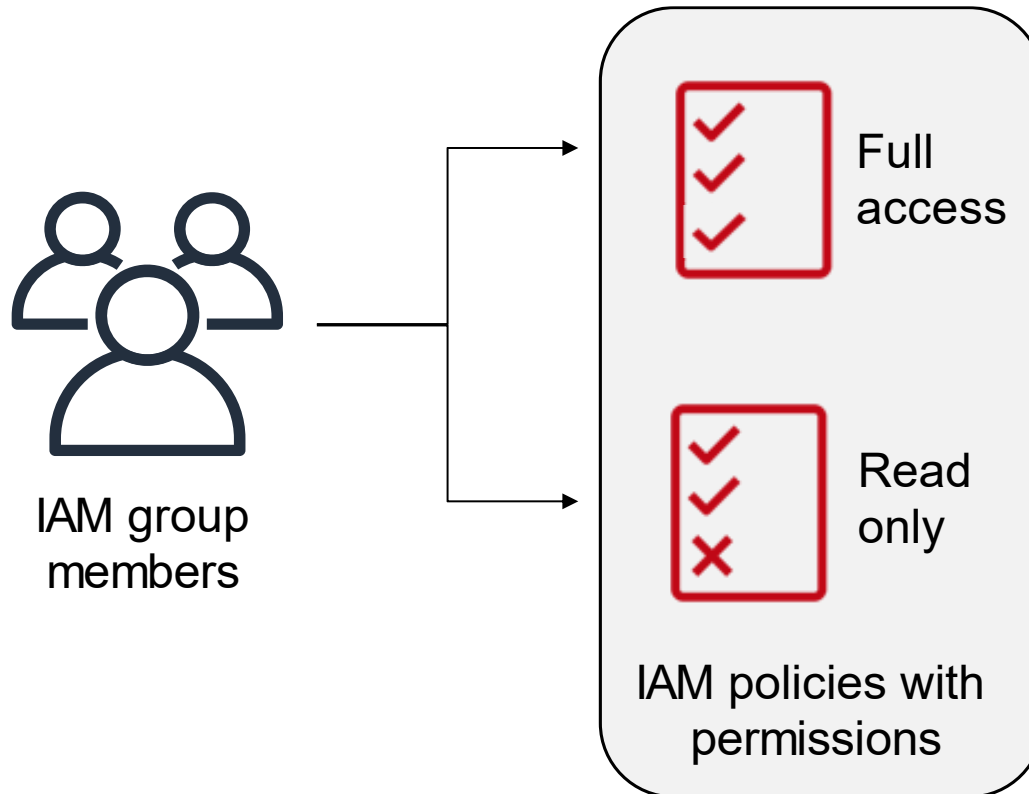
Authentication scenario



Authorizing with IAM



Principle of least privilege



Best practices:

- Start by granting the minimum AWS account permissions that are needed for the job role.
- Grant additional access as needed.

Policies and permissions

AWS currently supports six types of policies:

Identity-based policies: These policies are attached to IAM identities and grant permissions to the identity.

Resource-based policies: These are attached to resources and grant permissions to the principal specified in the policy.

Permissions boundaries: These define the maximum permissions that the identity-based policies can grant to an entity. Permissions boundaries do not grant permissions.

AWS Organizations service control policy (SCP): This defines the maximum permissions for account members of an organization or organizational unit (OU). You can use SCPs to limit the permissions of identity-based or resource-based policies, but they cannot be used to grant permissions.

Access control lists (ACLs): You can use an ACL to control which principals in other accounts can access the resource to which the ACL is attached. You cannot use an ACL to grant permissions to entities that reside within the same account. ACLs are the only policy type that doesn't use the JSON document format.

Session policies: These are used with the AWS CLI or AWS API to create a temporary session for a role or federated user. You can use session policies to limit the permissions that a role or a user's identity-based policies grant to a session. Session policies limit permissions but cannot grant them.

Identity-based versus resource-based policies



Identity-based policies

- Attached to a user, group, or role
- Types of policies
 - AWS managed
 - policies are created and managed by AWS.
 - Customer managed
 - policies that you create and manage in your AWS account
 - Inline
 - policies that you create and manage, and that are embedded directly into a single user, group, or role



Resource-based policies

- Attached to AWS resources
 - Example: Attach to an Amazon S3 bucket
- Always an inline policy

Identity-based and resource-based policies

Identity-based policies

What does a particular identity have access to?

Carlos	Resource	Read	Write	List
	Resource X	Allow	Allow	Allow

Richard	Resource	Read	Write	List
	Resource Y	Allow	N/A	N/A
	Resource Z	Allow	N/A	N/A

Managers	Resource	Read	Write	List
	Resource X	N/A	N/A	Allow
	Resource Y	N/A	N/A	Allow
	Resource Z	N/A	N/A	Allow

Resource-based policies

Who has access to a particular resource?

Resource X	User	Read	Write	List
	Ana	Allow	Allow	Allow
	Akua	Allow	Allow	Allow
	Mary	Allow	N/A	Allow
	Mateo	N/A	N/A	Allow

Resource Y	User	Read	Write	List
	Paulo	Allow	Allow	Allow
	Nikki	Allow	N/A	N/A
	Mateo	N/A	Allow	Allow

Managed and inline IAM policies

Managed policies

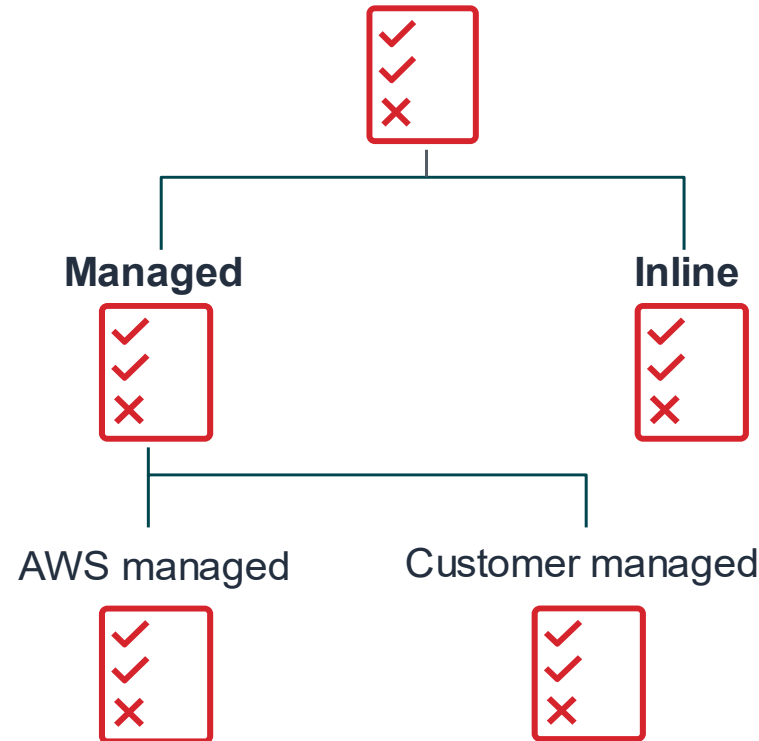
- Are standalone, *identity-based* policies
- Can be attached to multiple users, groups, and roles



Features:

- Reusability
- Central change management
- Versioning and rollback
- Permissions management that can be delegated to others
- Provides the use of *permissions boundaries*

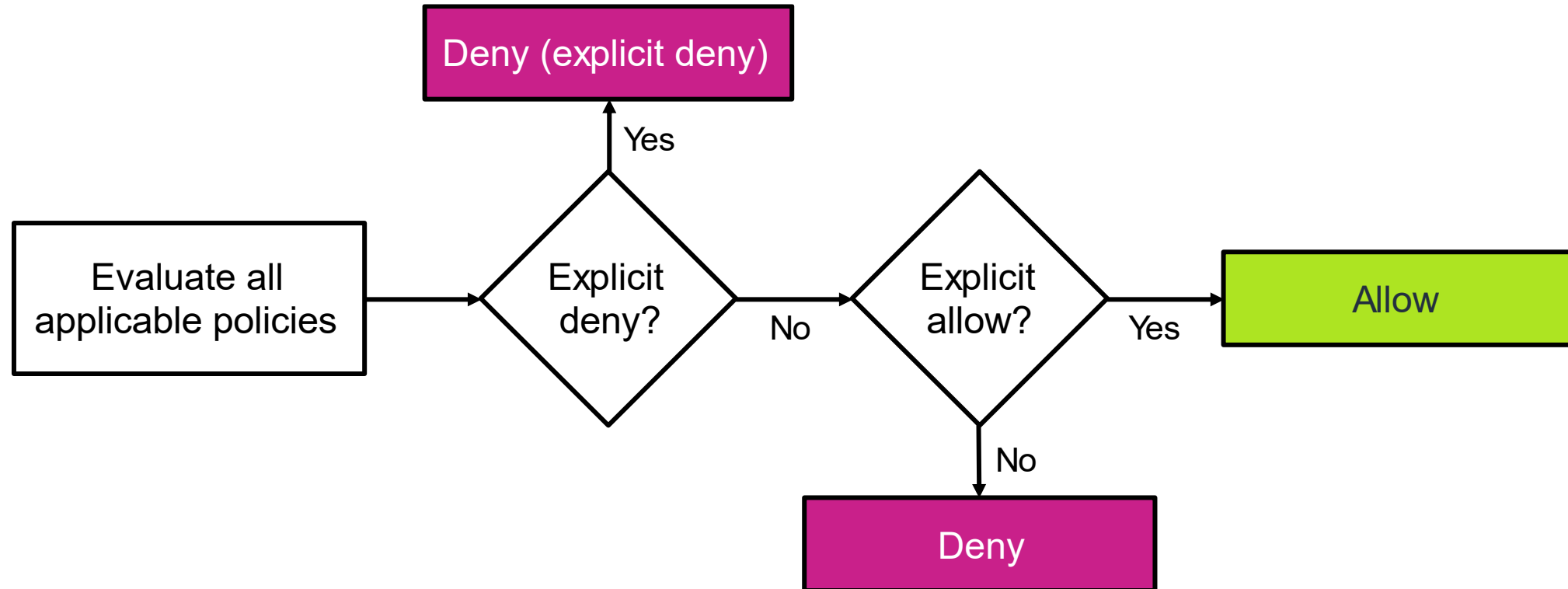
IAM policies



Inline policies

- Are embedded in a principal entity (for example, user, group, or role)
- ↓
- Are useful for a strict 1:1 relationship between a policy and the entity

Evaluation logic for IAM policies



Examples of authorizing with IAM



IAM policy document structure

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "effect",
    "Action": "action",
    "Resource": "arn",
    "Condition": {
      "condition": {
        "key": "value"
      }
    }
  }]
}
```

- **Effect:** Effect can be either *Allow* or *Deny*
- **Action:** Type of access that is allowed or denied
 "Action": "s3:GetObject"
- **Resource:** Resources that the action will act on
 "Resource": "arn:aws:sqs:us-west-2:123456789012:queue1"
- **Condition:** Conditions that must be met for the rule to apply
 "Condition" : {
 "StringEquals" : {
 "aws:username" : "johndoe"
 }
 }

ARNs and wildcards

- Resources are identified by using Amazon Resource Name (ARN) format
 - Syntax – *arn:partition:service:region:account:resource*
 - Example – "Resource": "arn:aws:iam::123456789012:user/mmajor"
- You can use a **wildcard** (*) to give access to all actions for a specific AWS service
 - Examples –
 - "Action": "s3:*"
 - "Action": "iam:*AccessKey*"



Examples of authorizing with IAM

IAM policy example

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": ["dynamodb:*", "s3:*"],
    "Resource": [
      "arn:aws:dynamodb:region:account-number-without-hyphens:table/table-name",
      "arn:aws:s3:::bucket-name",
      "arn:aws:s3:::bucket-name/*"
    ]
  },
  {
    "Effect": "Deny",
    "Action": ["dynamodb:*", "s3:*"],
    "NotResource": [
      "arn:aws:dynamodb:region:account-number-without-hyphens:table/table-name",
      "arn:aws:s3:::bucket-name",
      "arn:aws:s3:::bucket-name/*"
    ]
  }
]
```

Explicit allow gives users access to a specific DynamoDB table and...

...Amazon S3 buckets.

Explicit deny ensures that the users cannot use any other AWS actions or resources other than that table and those buckets.

An explicit deny statement **takes precedence** over an allow statement.



Examples of authorizing with IAM

Example: Identity-based policy

```
{  
  "version": "2018-10-17",  
  "statement": {  
    "effect": "Allow",  
    "action": [  
      "iam:*LoginProfile",  
      "iam:*AccessKey*",  
      "iam:*SSHPublicKey*"  
    ],  
    "resource": "arn:aws:iam::account-id-without-  
hyphens:user/${aws:username}"  
  }  
}
```

These are
the actions
allowed by
the policy

These are the AWS
resources that the allowed
actions can be performed on

Examples of authorizing with IAM

Example: Identity-based policy

```
{  
  "version": "2018-10-17",  
  "statement": {  
    "effect": "Allow",  
    "action": [  
      "iam:*LoginProfile",  
      "iam:*AccessKey*",  
      "iam:*SSHPublicKey*"  
    ],  
    "resource": "arn:aws:iam::account-id-without-  
hyphens:user/${aws:username}"  
  }  
}
```

These are
the actions
allowed by
the policy

These are the AWS
resources that the allowed
actions can be performed on

Examples of authorizing with IAM

Example: Cross-account, resource-based policy

```
{
  "version": "2018-10-17",
  "statement": {
    "sid": "AccountBAccess1",
    "principal": {"AWS": "111122223333"},
    "effect": "Allow",
    "action": "s3:*",
    "resource": [
      "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
      "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
    ]
  }
}
```

Account B principal allowed by Account A to make the request

Actions allowed by Account A

Resources shared by Account A

Examples of authorizing with IAM

Example IAM policy: Allow statement

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": ["dynamodb:*", "s3:*"],
    "Resource": ["arn:aws:dynamodb:region:account-number-without-hyphens:table/coursenotes",
      "arn:aws:s3:::course-notes-web",
      "arn:aws:s3:::course-notes-mp3/*"]
  },
  {
    "Effect": "Deny",
    "Action": ["dynamodb:*", "s3:*"],
    "NotResource": ["arn:aws:dynamodb:region:account-number-without-hyphens:table/coursenotes",
      "arn:aws:s3:::course-notes-web",
      "arn:aws:s3:::course-notes-mp3/*"]
  }
]
```

Allows the entity to perform any DynamoDB or S3 action on this DynamoDB table and these S3 buckets

Example IAM policy: Deny statement

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": ["dynamodb:*", "s3:*"],
    "Resource": ["arn:aws:dynamodb:region:account-number-without-hyphens:table/coursenotes",
      "arn:aws:s3:::course-notes-web",
      "arn:aws:s3:::course-notes-mp3/*"]
  },
  {
    "Effect": "Deny",
    "Action": ["dynamodb:*", "s3:*"],
    "NotResource": ["arn:aws:dynamodb:region:account-number-without-hyphens:table/coursenotes",
      "arn:aws:s3:::course-notes-web",
      "arn:aws:s3:::course-notes-mp3/*"]
  }
]
```

Ensures that the entity can't perform any action on any DynamoDB table or S3 bucket except the tables and buckets that the policy specifies

Examples of authorizing with IAM

Example IAM policy: Permissions boundary

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:*",
        "cloudwatch:*",
        "ec2:*"
      ],
      "Resource": "*"
    }
  ]
}
```

1

If policy 1 is attached to a user...

```
{
  "version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "iam:CreateUser",
    "Resource": "*"
  }
}
```

2

...followed by policy 2, then any attempt to create a user in IAM will fail because of the boundary restrictions placed by policy 1.

Activity: IAM policy analysis

Consider this IAM policy, then answer the questions as they are presented.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "iam:Get*",
      "iam:List*"
    ],
    "Resource": "*"
  }
}
```

1. Which AWS service does this policy grant you access to?
 - **ANSWER:** The IAM service.
2. Does it allow you to create an IAM user, group, policy, or role?
 - **ANSWER:** No. The access is limited to *get* and *list* requests. It effectively grants read-only permissions.
3. Go to <https://docs.aws.amazon.com/IAM/latest/UserGuide/> and in the left navigation expand *Reference > Policy Reference > Actions, Resources, and Condition Keys*. Choose *Identity And Access Management*. Scroll to the Actions Defined by Identity And Access Management list.

Name at least three specific actions that the iam:Get* action allows.

- **ANSWER:** iam:Get* allows many specific actions, including GetGroup, GetPolicy, GetRole, and others.

Additional authentication and access management services



Identity federation

- **Identity federation** is a system of trust between two parties to authenticate users and convey information that is needed to authorize resource access.
 - *Identity providers* are responsible for user authentication.
 - *Service providers* are responsible for resource access.
- Two AWS services are available to provide federation to AWS accounts and applications:
 - AWS Single Sign-On (AWS SSO)
 - AWS Identity and Access Management (IAM)

Additional authentication and access management services

AWS Single Sign-On (AWS SSO)

- Create or connect identities once and manage access centrally across your AWS accounts.
- AWS SSO provides a unified administration experience to define, customize, and assign fine-grained access.
- Users are provided a user portal to access all their assigned AWS accounts or cloud applications.
- You can flexibly configure access to run parallel to or replace AWS account access management by using IAM.



AWS Single Sign-On
(AWS SSO)

Additional authentication and access management services

AWS Directory Service

you can use your existing **on-premises** user credentials to **access cloud resources**, which simplifies the process to extend your existing Active Directory to AWS.

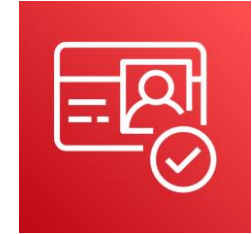
- AWS Directory Service for Microsoft Active Directory, also called AWS Managed Microsoft AD
- Facilitates directory-aware workloads and AWS resources to use managed Active Directory in the AWS Cloud
- Provides the ability to extend your existing Active Directory to AWS by using your existing on-premises user credentials to access cloud resources
- Supports Active Directory SSO to AWS applications by using a single set of credentials



AWS Directory
Service

Amazon Cognito

- Integrates user sign-up, sign-in, and access control with **web** and **mobile** applications
- Provides a secure identity store that can scale to millions of users with **Amazon Cognito user pools**
- Offers user sign-in through enterprise identity providers and social identity providers such as Apple, Google, Facebook, and Amazon
- Provides the ability to create unique identities for your users and federate them with identity providers through **Amazon Cognito identity pools**



Amazon Cognito

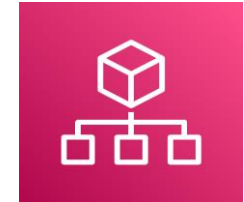
User pools are directories that provide sign-up and sign-in options for your application users
identity pools, you can grant your users access to AWS services through temporary, limited-privilege AWS credentials.

Using AWS Organizations



AWS Organizations

- Account management service that you can use to **consolidate multiple AWS accounts** into a centrally managed organization
- Includes account creation and management as well as **consolidated billing capabilities**
- Provides for hierarchical grouping of accounts
- Supports centralized policy control over AWS services and API actions using service control policies (SCPs)
- Integrates with IAM and other services



AWS Organizations

Example: SCP

- Prevent member accounts from leaving the organization:

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Deny",  
      "Action": [ "organizations:LeaveOrganization" ],  
      "Resource": "*" }  
  ]  
}
```

Thanks

