AWS Academy Cloud Architecting Module 03 Student Guide

Version 3.0.0

200-ACACAD-30-EN-SG

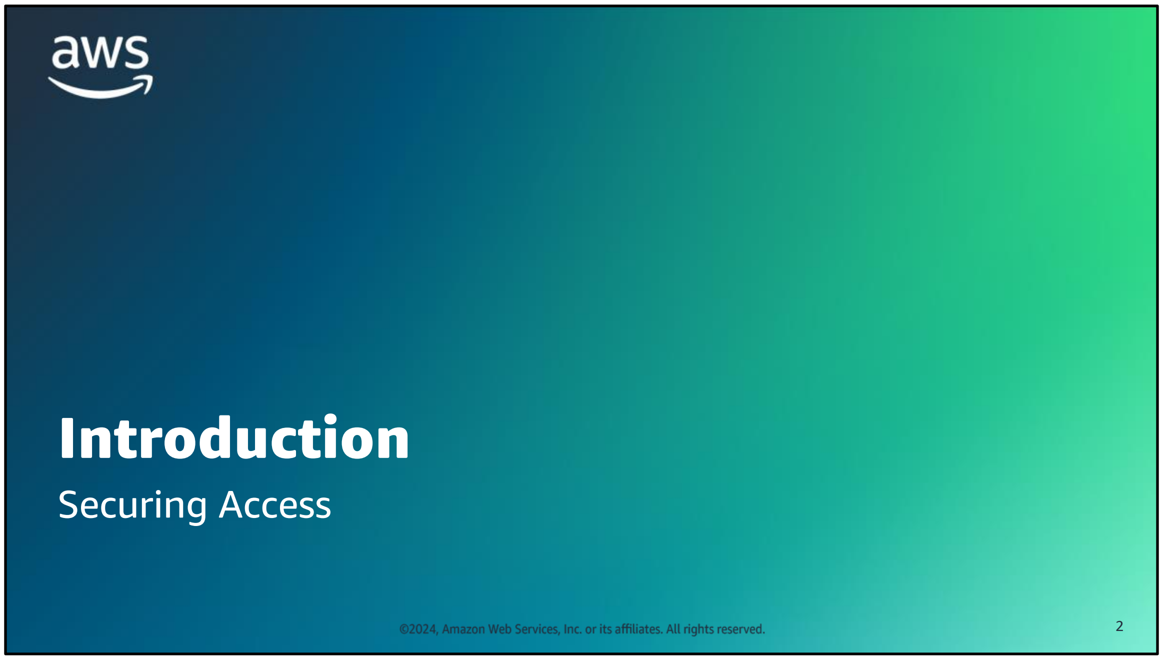
Contents

Securing Access

AWS Academy Cloud Architecting

1

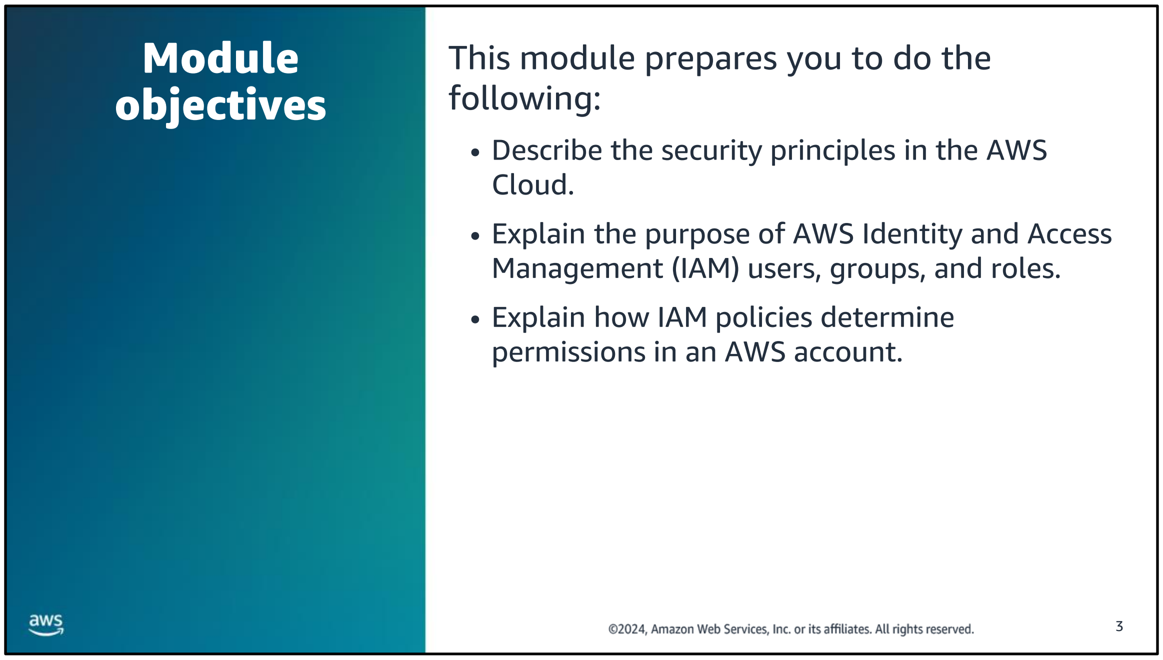
Welcome to the Securing Access module. This module reviews key security principles and concepts for securing access to your cloud resources.



Introduction

Securing Access

This introduction section describes the content of this module.

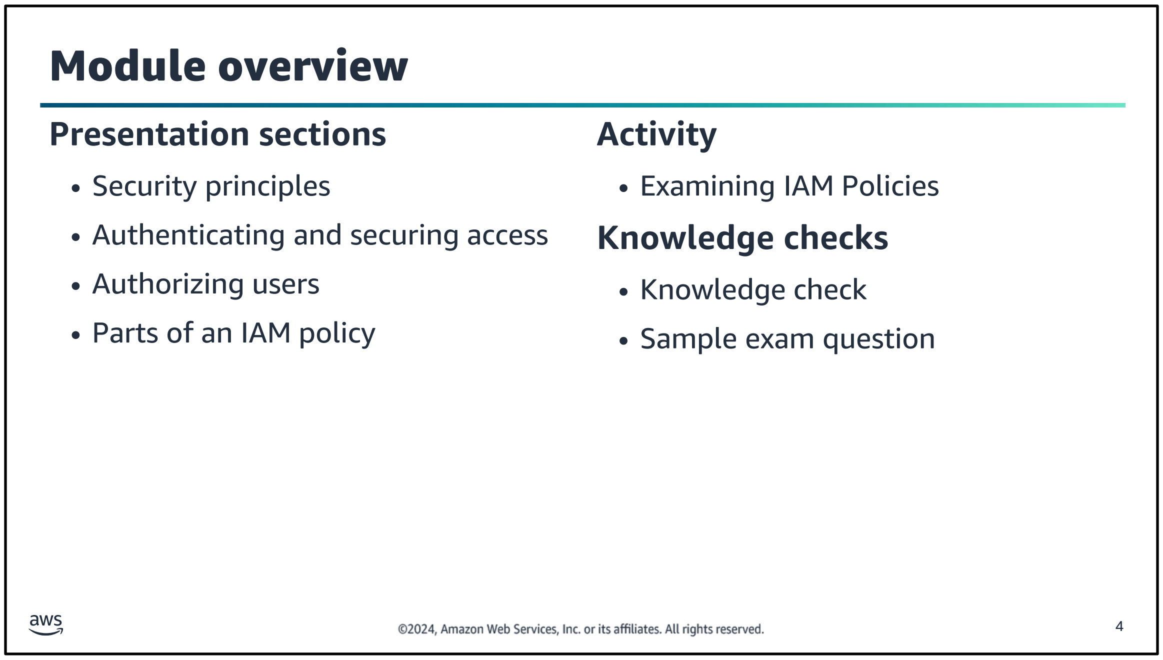


Module objectives

This module prepares you to do the following:

- Describe the security principles in the AWS Cloud.
- Explain the purpose of AWS Identity and Access Management (IAM) users, groups, and roles.
- Explain how IAM policies determine permissions in an AWS account.

This module reviews key concepts that are related to security in the Amazon Web Services (AWS) Cloud. This includes cloud security principles and the use of AWS Identity and Access Management (IAM) users, groups, roles, and policies to secure access to cloud resources.



Module overview

Presentation sections

- Security principles
- Authenticating and securing access
- Authorizing users
- Parts of an IAM policy

Activity

- Examining IAM Policies

Knowledge checks

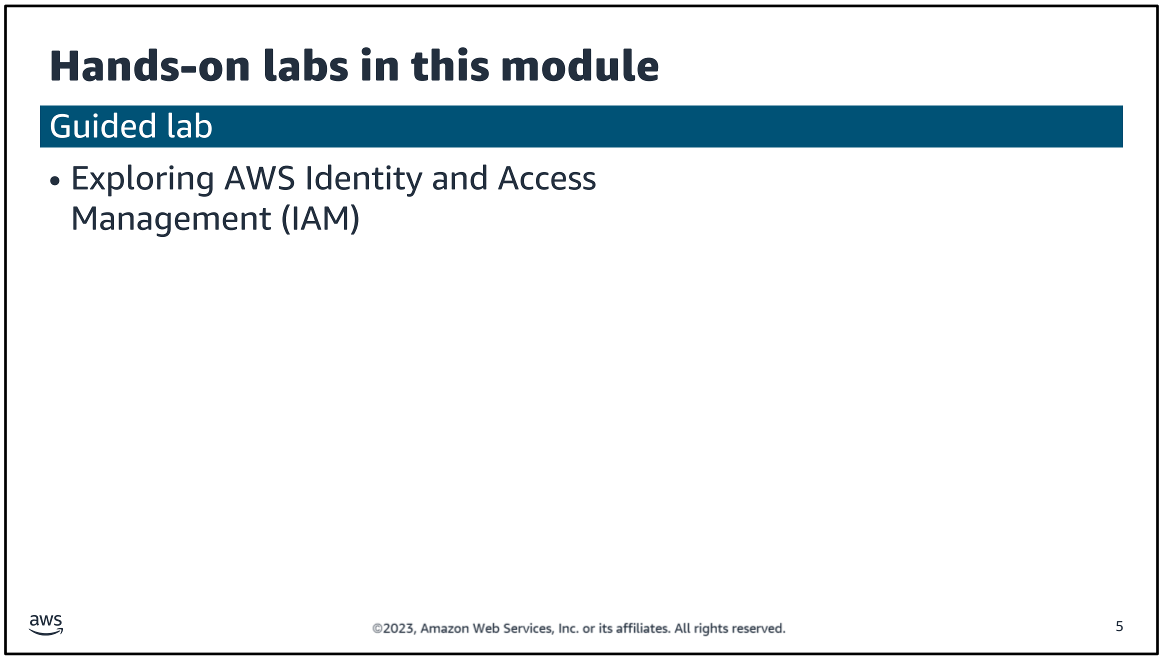
- Knowledge check
- Sample exam question

4

The objectives of this module are presented across multiple sections.

You will also participate in an activity to examine IAM policies and answer questions about the access that the policies provide.

The module wraps up with a sample exam question and an online knowledge check that covers the presented material.
The lab in this module is described on the next slide.



Hands-on labs in this module

Guided lab

- Exploring AWS Identity and Access Management (IAM)

5

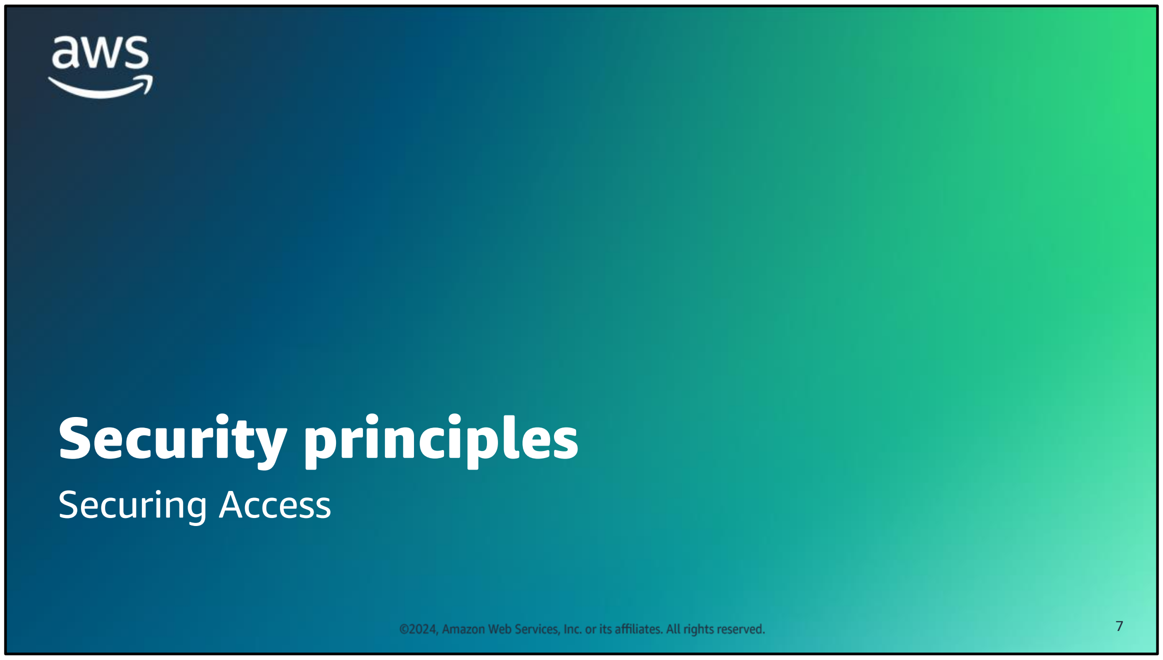
This module includes the guided lab listed. Additional information about this lab is included in the student guide where the lab takes place, and detailed instructions are provided in the lab environment.

As a cloud architect:

- I need to apply security best practices when I grant access to cloud resources so that I can minimize the risk of unwanted access at each layer of an architecture.

6

This slide asks you to take the perspective of a cloud architect as you think about how to secure access to your cloud resources. Keep these considerations in mind as you progress through this module. Remember that the cloud architect should work backward from the business need to design the best architecture for a specific use case. As you progress through the module, consider the café scenario in the course as an example business need, and think about how you would address these needs.

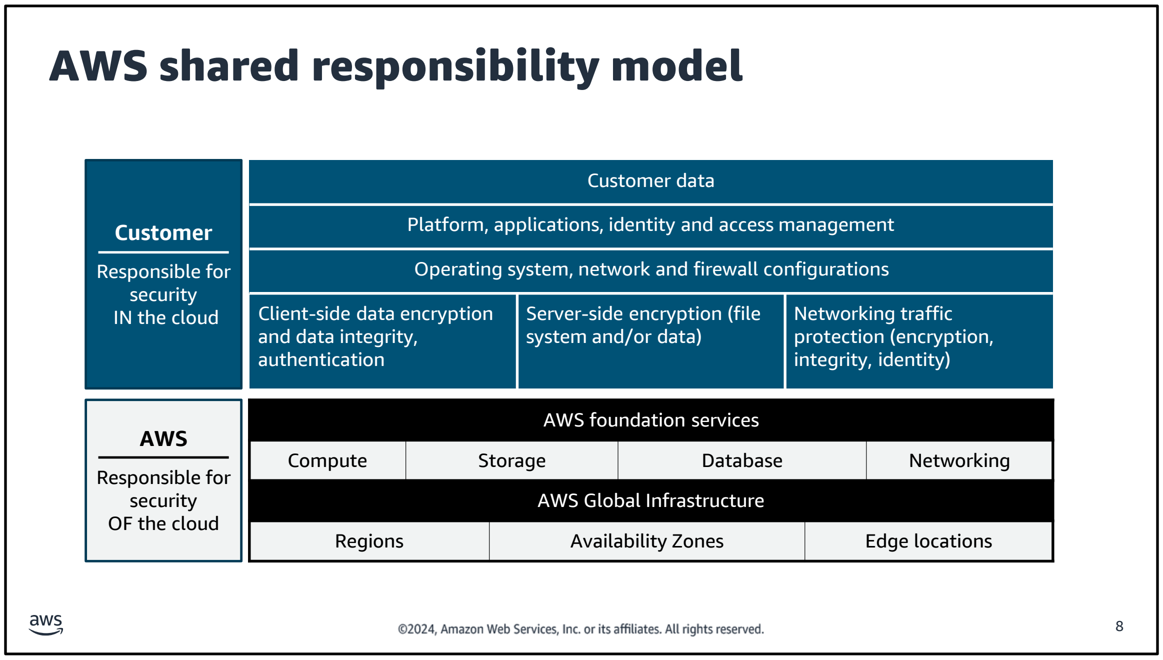


Security principles

Securing Access

This section reviews the following principles, which are important for securing your AWS architectures:

- Consider the AWS shared responsibility model when you determine how you need to apply security to your cloud architectures.
- Use the security pillar of the Well-Architected Framework to identify the design principles that apply to your architectures.
- Always apply the principle of least privilege when you secure your AWS resources.



AWS shared responsibility model

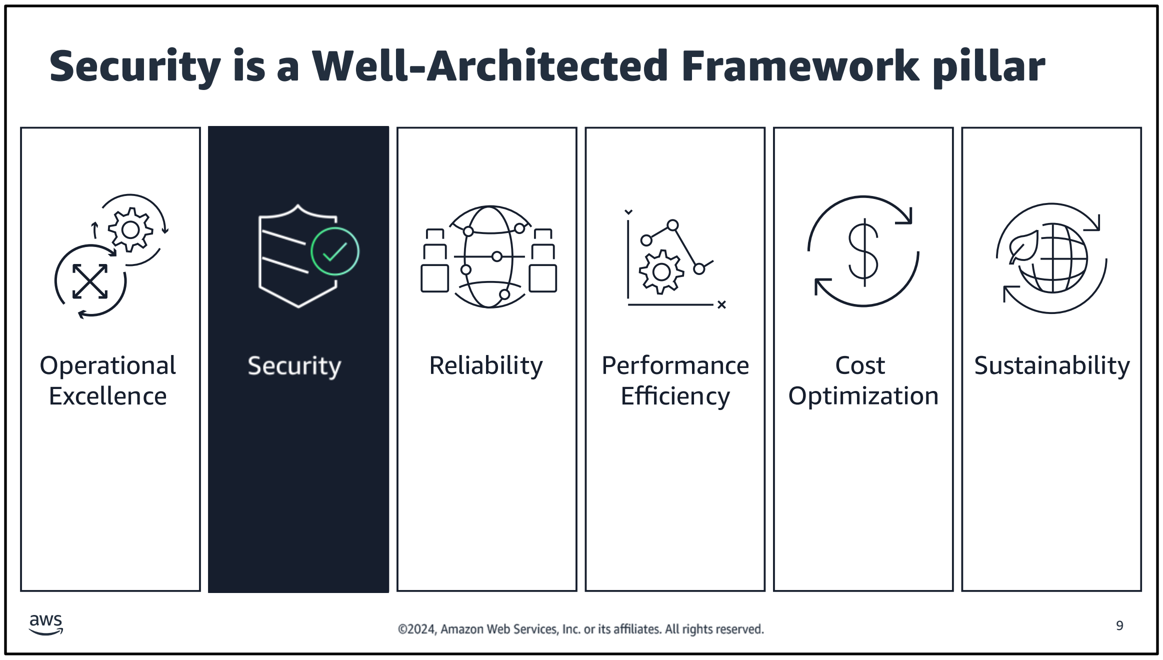
Customer
Responsible for security IN the cloud
Customer data
Platform, applications, identity and access management
Operating system, network and firewall configurations
Client-side data encryption and data integrity, authentication
Server-side encryption (file system and/or data)
Networking traffic protection (encryption, integrity, identity)
AWS
Responsible for security OF the cloud
AWS foundation services
Compute
Storage
Database
Networking
AWS Global Infrastructure
Regions
Availability Zones
Edge locations

8

Image description: Shared responsibility model of customer and AWS responsibilities. The customer is responsible for security *in* the cloud. This includes customer data. Platform, applications, identity and access management. Operating system, network and firewall configurations. Client-side data encryption and data integrity, authentication. Server-side encryption of the file system and data. Networking traffic protection, to include encryption, integrity, and identity. AWS is responsible for security *of* the cloud. This includes the AWS foundation services for compute, storage, databases, and networking, and the AWS Global Infrastructure, which includes Regions, Availability Zones, and Edge Locations. **End description.**

Security and compliance are shared responsibilities between AWS and our customers. AWS operates, manages, and controls security *of* the cloud. This responsibility includes securing components, from the host operating system and virtualization layer to the physical security of the facilities where services operate. AWS is responsible for protecting the global infrastructure that runs all the services that are offered in the AWS Cloud. This infrastructure is composed of the hardware, software, networking, and facilities that run AWS Cloud services.

You assume responsibility and management *in* the cloud. The security steps that you must take depend on the services that you use and the complexity of your system. Customer responsibilities include selecting and securing operating systems that run on Amazon Elastic Compute Cloud (Amazon EC2) instances, and securing the applications that are launched on AWS resources. Customers must also select and handle the configuration of security groups, firewalls, and networks. Customers are also responsible for managing accounts and their data, including encryption options.



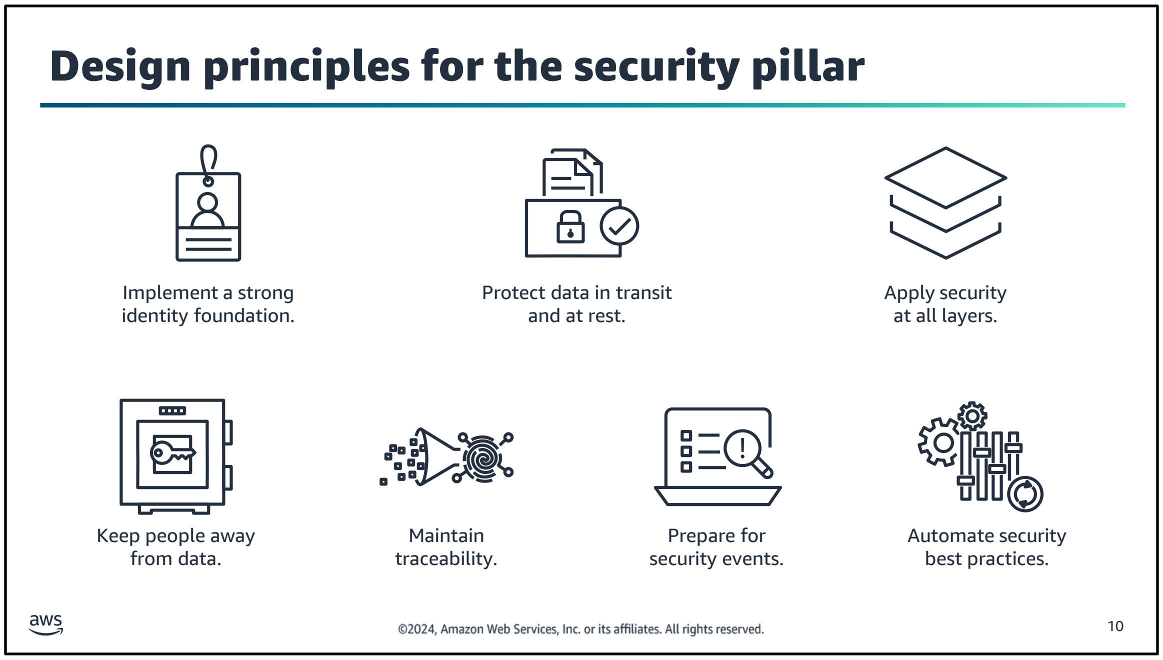
Security is a Well-Architected Framework pillar

The AWS Academy Cloud Foundations course introduced the Well-Architected Framework. You might find it helpful to open the website now to look at how the pillars are organized.

Security is one of the six pillars of the framework. The other pillars are operational excellence, reliability, performance efficiency, cost optimization, and sustainability. The framework provides best practices and design guidance across each pillar to help you make choices and review existing architectures.

You can use the AWS Well-Architected Tool to help you implement best practices that are part of the Well-Architected Framework. The AWS WA Tool is a self-service tool that provides you with on-demand access to current AWS best practices. These best practices can help you build solutions that meet the well-architected security design principles as well as design principles that are associated with the other pillars.

Links to the AWS Well-Architected Framework site, and to an appendix of questions and best practices that can help you follow the framework, are provided on the content resources page of your online course.



Design principles for the security pillar

Implement a strong identity foundation.
Protect data in transit and at rest.
Apply security at all layers.
Keep people away from data.
Maintain traceability.
Prepare for security events.
Automate security best practices.

©2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

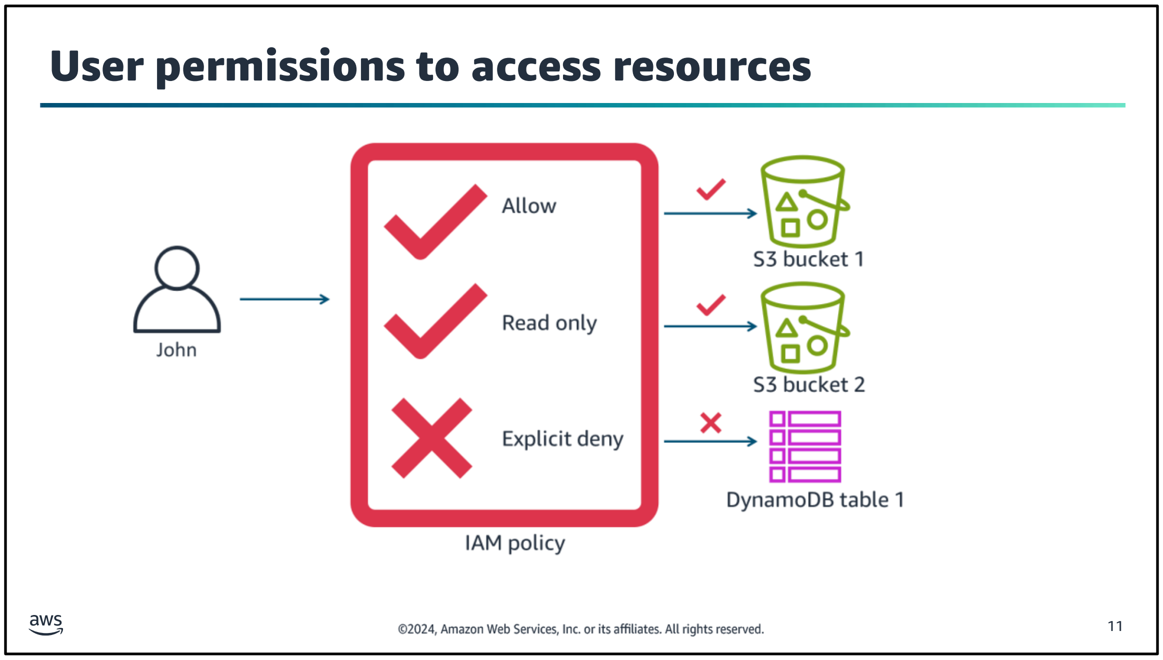
10

Your architecture will present a much stronger security presence if you follow these design principles from the security pillar of the Well-Architected Framework:

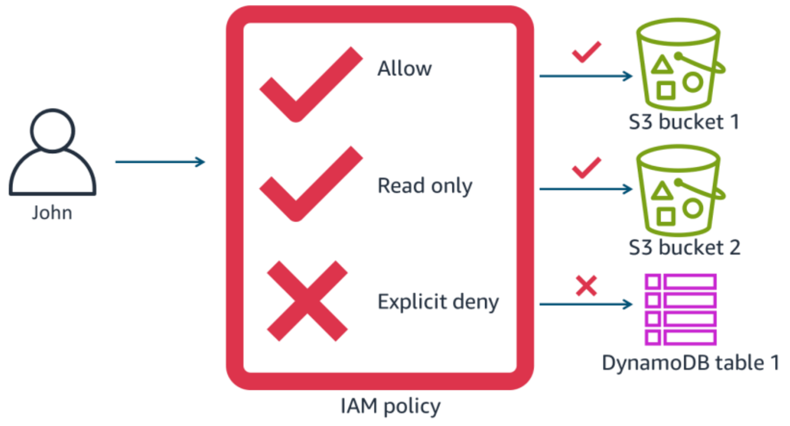
- **Implement a strong identity foundation:** Implement the principle of least privilege and enforce separation of duties with appropriate authorization for each interaction with your AWS resources. Centralize identity management, and aim to eliminate reliance on long-term, static credentials.
- **Protect data in transit and at rest**: Classify your data into sensitivity levels, and use mechanisms, such as encryption, tokenization, and access control, where appropriate.
- **Apply security at all layers:** Apply a defense-in-depth approach with multiple security controls, and apply it to all layers (for example, edge of network, virtual private cloud [VPC], load balancing, every instance and compute service, operating system, application, and code).
- **Keep people away from data:** Use mechanisms and tools to reduce or eliminate the need for direct access or manual processing of data. This reduces the risk of mishandling or modification and human error when handling sensitive data.
- **Maintain traceability:** Monitor, alert, and audit actions and changes to your environment in real time. Integrate log and metric collection with systems to automatically investigate and take action.
- **Prepare for security events:** Prepare for an incident by having incident management and investigation policies and processes that align to your organizational requirements. Run incident response simulations, and use tools with automation to increase your speed for detection, investigation, and recovery.
- **Automate security best practices:** Automated software-based security mechanisms improve your ability to securely scale more rapidly and cost-effectively. Create secure architectures, which includes implementing controls that are defined and managed as code in version-controlled templates.

The security pillar of the Well-Architected Framework describes how to take advantage of cloud technologies to protect data, systems, and assets in a way that can improve your security posture. For more information, see the Security Pillar of the AWS Well-Architected Framework link on the content resources page of your online course.

The next few slides review key aspects of the first two design principles.

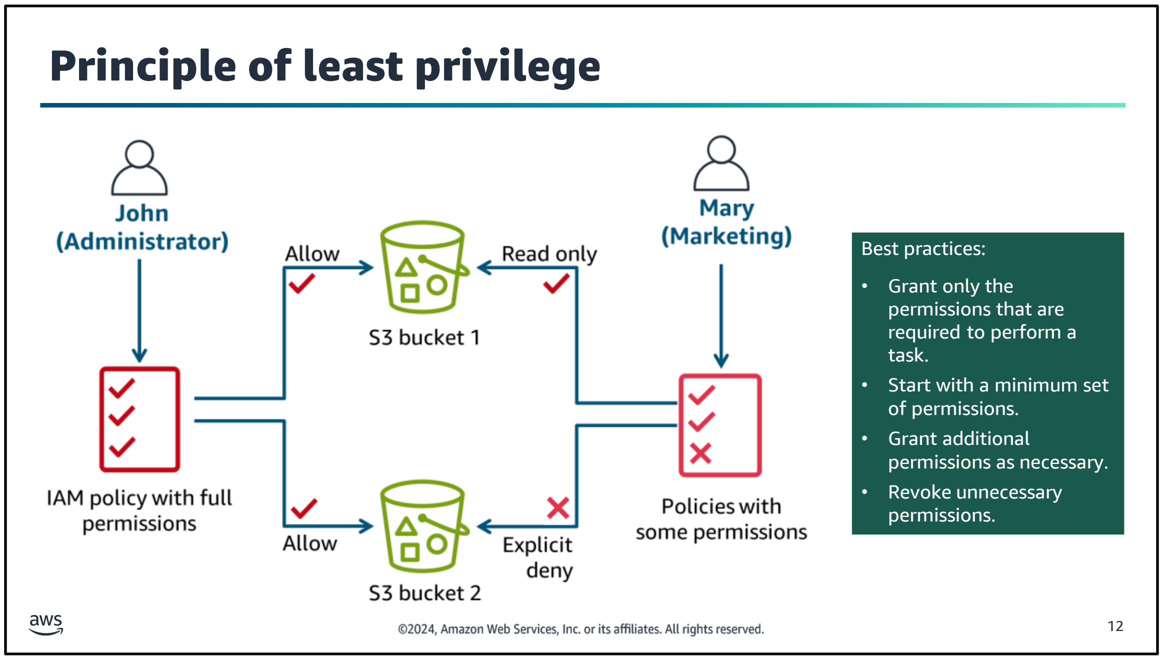


User permissions to access resources



11

Part of implementing a strong identity foundation is using policies to grant or deny access to your AWS resources, such as Amazon Simple Storage Service (Amazon S3) buckets. In this example, John can read, write, and delete objects in S3 bucket 1. However, he can only read the objects in S3 bucket 2, and he is explicitly denied access to a specific Amazon DynamoDB table.



Principle of least privilege

Best practices:

- Grant only the permissions that are required to perform a task.
- Start with a minimum set of permissions.
- Grant additional permissions as necessary.
- Revoke unnecessary permissions.

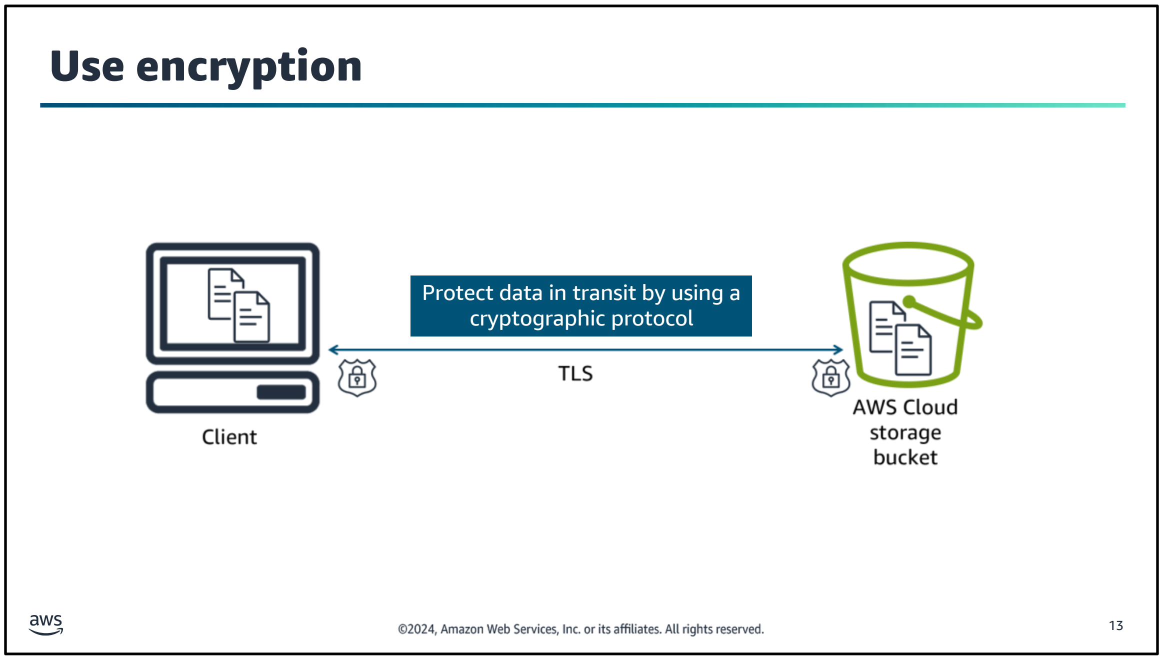
Image description: Diagram of John, an administrator user, receiving permissions from a policy. The policy allows full access to S3 buckets 1 and 2. Mary, a marketing user, receives read-only access to S3 bucket 1 and is explicitly denied access to S3 bucket 2 through policies. **End description.**

Another key aspect to implementing a strong identity foundation is to apply the principle of least privilege to the permissions that you set. This means granting only the permissions that are needed to perform a task.

It's more secure to start with a minimum set of permissions and grant additional permissions as needed. This provides better security than starting with permissions that are too permissive and then trying to restrict them later. To define the correct set of permissions, you must research what access is needed to accomplish a specific task.

When you create access policies, determine what your users need to do, and then craft policies that allow them to perform only those tasks. Similarly, create policies for individual resources that identify precisely who is allowed to access the resource, and allow only the minimal permissions for those users. For example, perhaps developers should be allowed to create EC2 instances in production environments but not to stop or terminate the instances.

For more information, see the Apply Least-Privilege Permissions link on the content resources page of your online course.



Use encryption

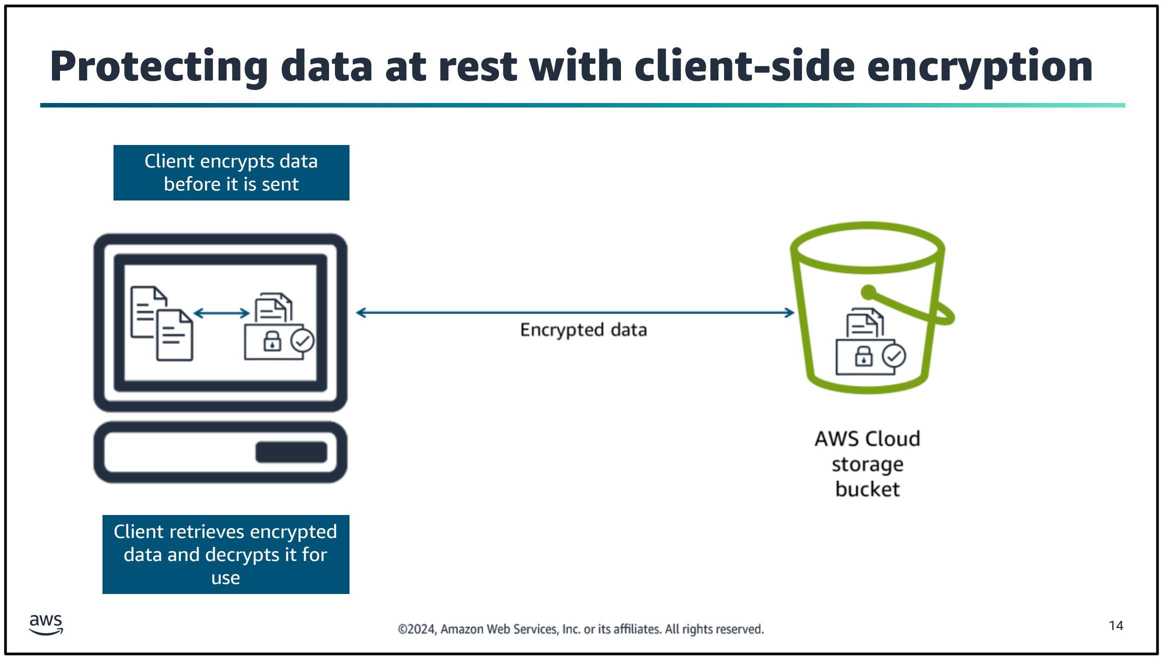
13

Image description: Diagram of protecting data in transit by using cryptographic protocol. TLS protects the data as it moves to and from the client and cloud storage. **End description.**

In addition to limiting who has permissions to access data, you should prevent access to data through encryption. This practice relates to the design principle of protecting data in transit and at rest.

Data in transit is data that is actively moving from one location to another, such as across the internet or through a private network. To protect data in transit, you protect data while it's traveling from network to network or being transferred from a local storage device to a cloud storage device. Wherever data is moving, protection measures are critical because data is often considered less secure while in motion.

An example of protecting data in transit is encrypting pictures as they are uploaded to a cloud service. The pictures don’t need to be encrypted; however, by using TLS, you can ensure the privacy of the transfer.



Protecting data at rest with client-side encryption

Client encrypts data before it is sent

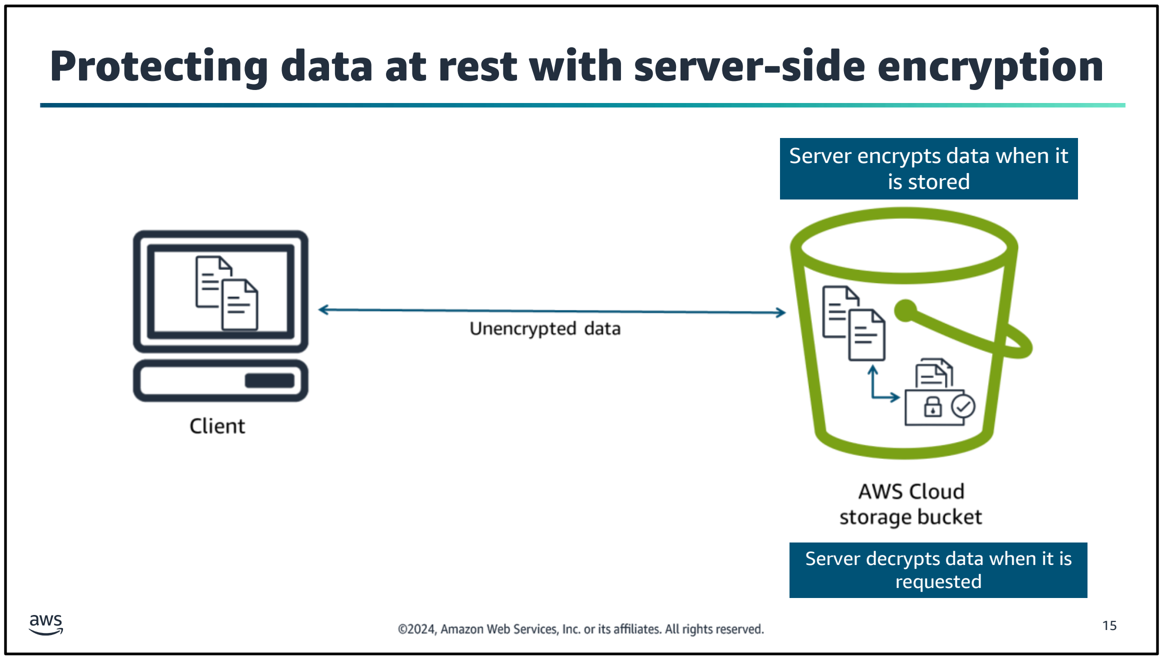
Client retrieves encrypted data and decrypts it for use

14

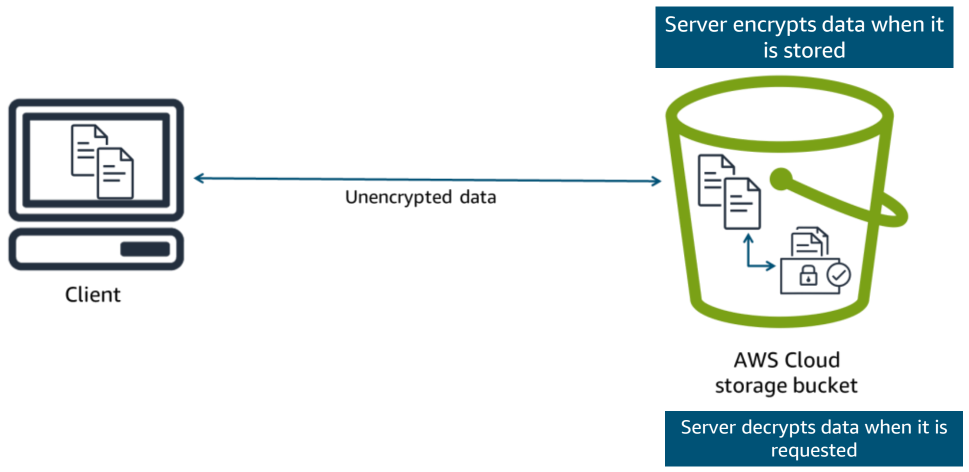
Image description: Diagram of client-side encryption. The client encrypts data before sending it. The encrypted data is sent through a secure pipe from the client to cloud storage. The client retrieves encrypted data from cloud storage and decrypts the data for use. **End description.**

Protecting data at rest means using encryption to protect data that is stored anywhere in your cloud solution. Client-side encryption provides end-to-end protection for your object, in transit and at rest, from its source to storage.

An example is encrypting data on a mobile device. Because mobile devices can be easily lost or stolen, encryption will ensure that data is scrambled and isn't accessible to unintended users.



Protecting data at rest with server-side encryption


Server encrypts data when it is stored
Unencrypted data
Client
AWS Cloud storage bucket
Server decrypts data when it is requested

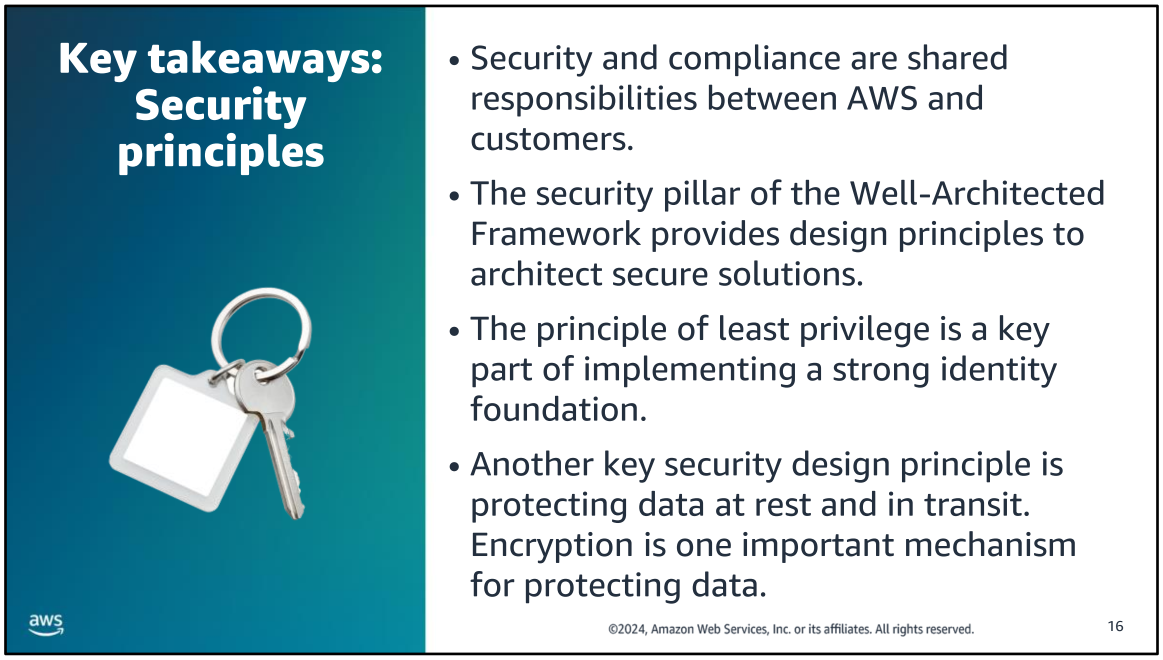
©2024, Amazon Web Services, Inc. or its affiliates. All rights reserved. 15

Image description: Diagram of server-side encryption. Unencrypted data is sent through a secure pipe. The server encrypts data when storing it and decrypts data when you request it. Data is returned to the client through the secure pipe. **End description.**

With server-side encryption, data is encrypted before it’s stored.

Amazon S3 provides server-side encryption. The service encrypts your data at the object level as it writes the data to disks in AWS data centers and decrypts the data when you access it.

An example is encryption of personal user data, such as a Social Security Number, that is stored in a cloud database.



Key takeaways: Security principles

- Security and compliance are shared responsibilities between AWS and customers.
- The security pillar of the Well-Architected Framework provides design principles to architect secure solutions.
- The principle of least privilege is a key part of implementing a strong identity foundation.
- Another key security design principle is protecting data at rest and in transit. Encryption is one important mechanism for protecting data.

16

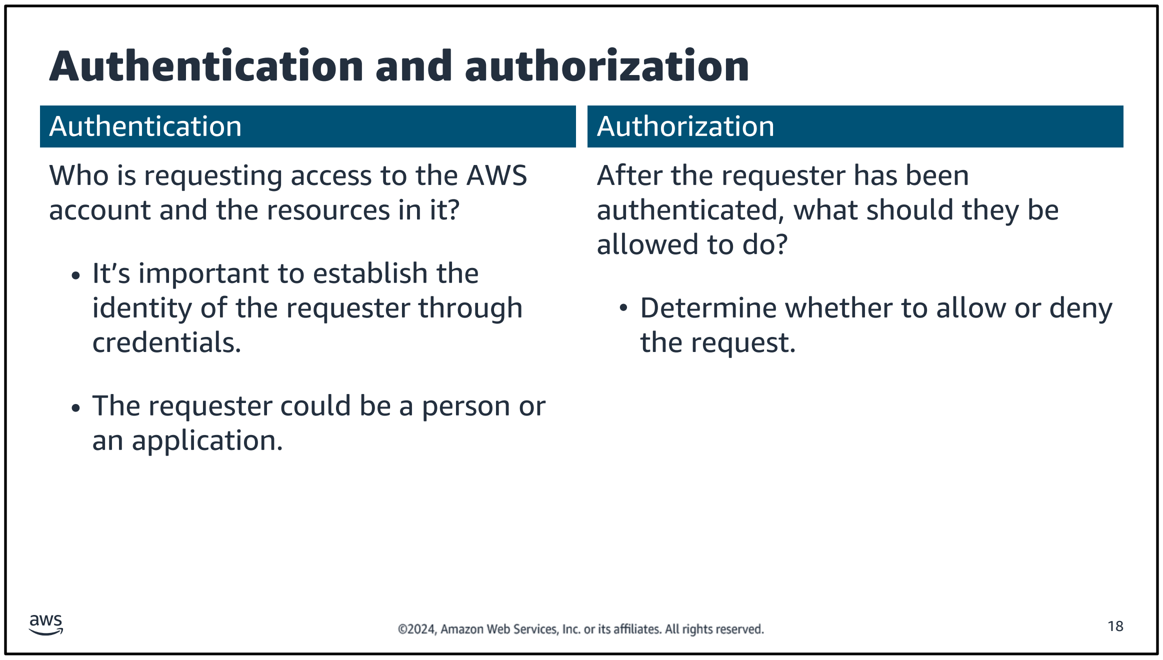
These key takeaways summarize this section.

Authenticating and securing access

Securing Access

17

This section covers authentication and the importance of securing access.



Authentication and authorization

Authentication	Authorization
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.	After the requester has been authenticated, what should they be allowed to do? • Determine whether to allow or deny the request.

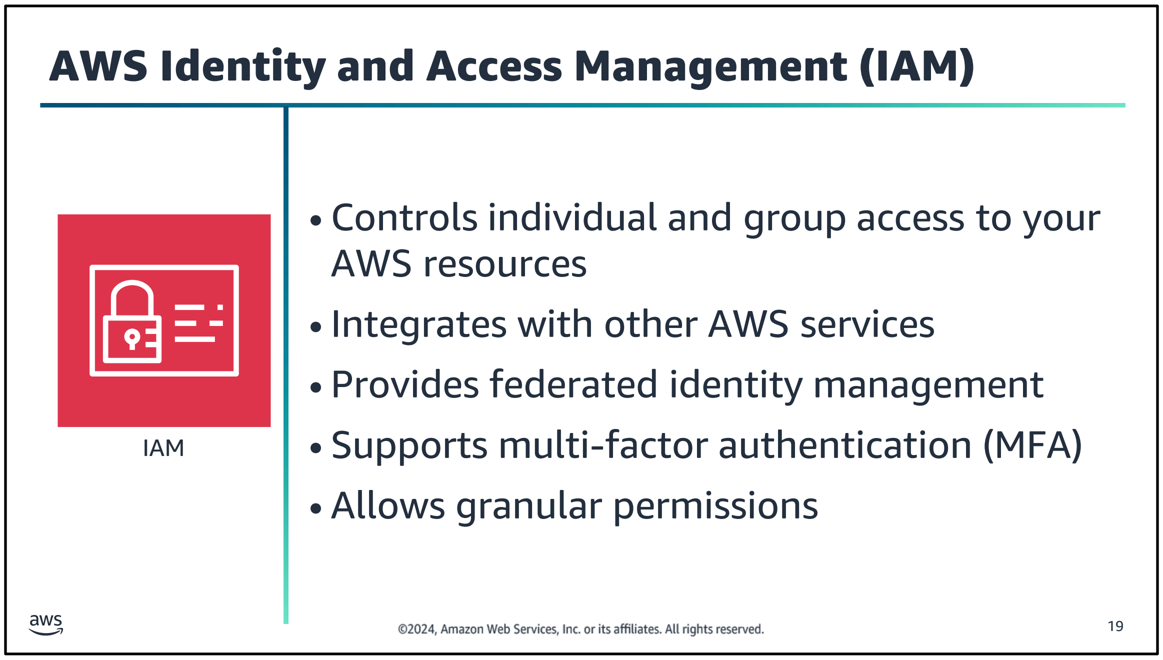
18

With *authentication*, you can use the identity of requesters to control *who* can use your AWS resources. With *authorization*, you can also use access management to control *what* resources requesters can use and in what ways.

To understand the fundamental aspects of authentication and authorization, consider a banking analogy. Think of a bank that allows their customers (users) to access their accounts online. Suppose that you are a bank customer. You have money in a checking account at that bank, and you want to pay a bill online. Should any random person be able to log in to your bank account and pay their bills with the money in your account? No, of course not!

Before the bank allows you to access your checking account, the bank must ensure that the person accessing the account is you. The bank wants to *authenticate* that you are who you claim to be. They typically accomplish this authentication by requiring you to enter your username and a password that is supposed to be known only to you. For extra security, they might have configured multi-factor authentication (MFA). Thus, in addition to typing in a password, you also must receive a code on your phone and enter that code on the bank's website.

Now, suppose that you are successfully logged in to the bank website (authenticated). Can you pay your bill by using the money in *another customer's* account? No, of course not! You don't have the *authorization* to access accounts that don't belong to you. However, you are authorized to access the money in *your* account, and you are authorized to pay bills by using *your* money.



AWS Identity and Access Management (IAM)

IAM

- Controls individual and group access to your AWS resources
- Integrates with other AWS services
- Provides federated identity management
- Supports multi-factor authentication (MFA)
- Allows granular permissions

aws ©2024, Amazon Web Services, Inc. or its affiliates. All rights reserved. 19

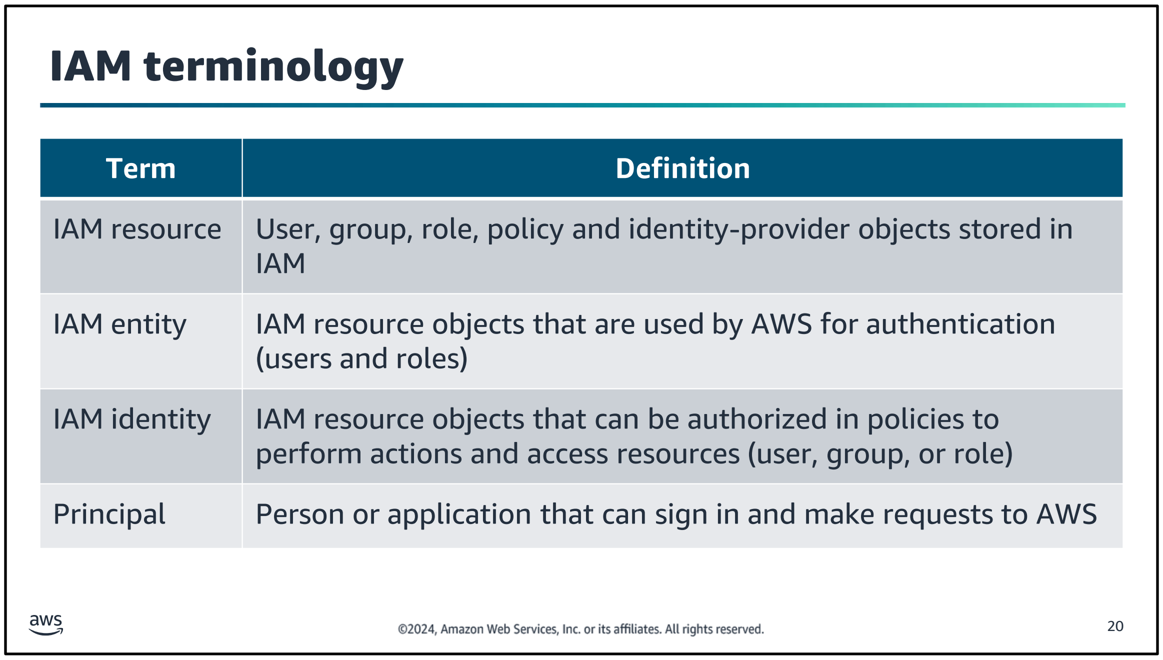
AWS Identity and Access Management is also known as IAM. With this service, you can configure fine-grained access control to AWS resources. You can use IAM to follow security best practices by granting unique security credentials to users and groups. These credentials specify which AWS service APIs and resources the users and groups can access. IAM is secure by default—users don't have access to AWS resources until permissions are explicitly granted.

IAM is integrated into most AWS services. You can define access controls from one place in the AWS Management Console, and they will take effect throughout your AWS environment.

You can use IAM to grant your employees and applications access to the console and to AWS service APIs by using your existing identity systems. AWS supports federation from corporate systems, such as Microsoft Active Directory, and standards-based identity providers.

IAM also supports multi-factor authentication (MFA). If MFA is enabled and an IAM user attempts to log in, they will be prompted for an authentication code. The authentication code is delivered to an MFA device, such as a hardware or virtual device. For example, Google Authenticator is a virtual MFA device that runs as an application on a user's smartphone.

You can create accounts that have permissions that are similar to the AWS account root user. However, a best practice is to create administrative accounts that grant only the account permissions that are needed. Follow the principle of least privilege. For example, ask yourself whether your database administrator (DBA) should be able to provision EC2 instances. If the answer is no, then provision accounts accordingly.

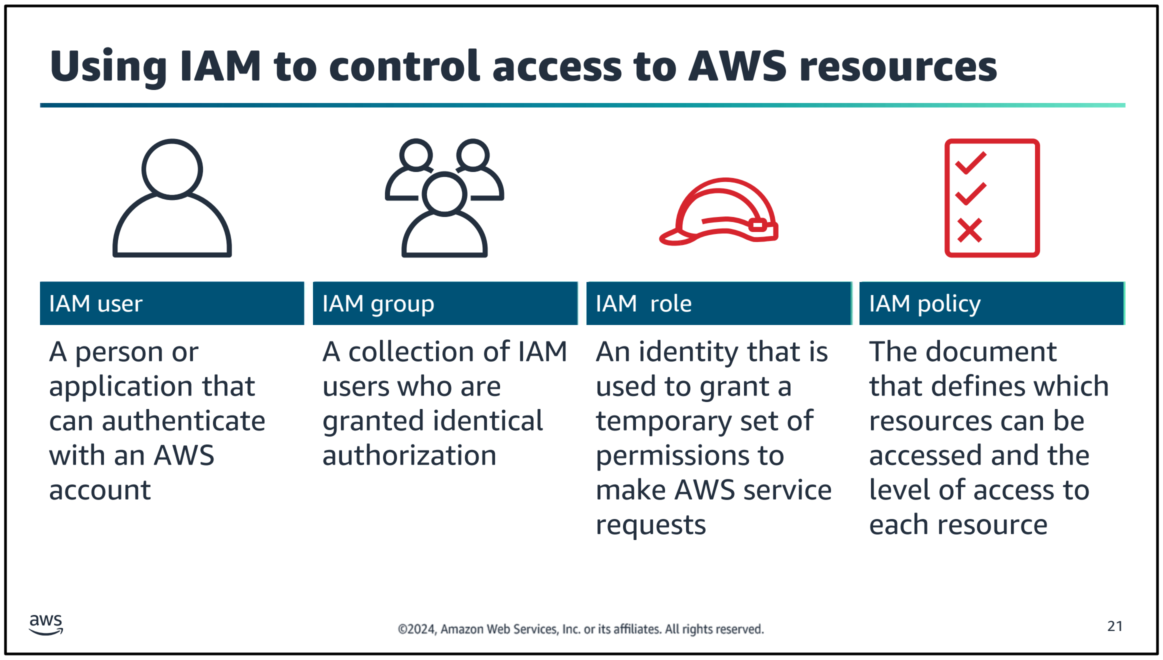


IAM terminology

Term	Definition
IAM resource	User, group, role, policy and identity-provider objects stored in IAM
IAM entity	IAM resource objects that are used by AWS for authentication (users and roles)
IAM identity	IAM resource objects that can be authorized in policies to perform actions and access resources (user, group, or role)
Principal	Person or application that can sign in and make requests to AWS

The following terms are specific to IAM, and you will use them throughout this course:

- An *IAM resource* is the user, group, role, policy, or identity provider object that is stored in IAM. As with other AWS services, you can add, edit, and remove resources from IAM.
- An *IAM entity* is the IAM resource object that AWS uses for authentication. Entities include IAM users and roles. Through an IAM entity, you can be provided the ability to sign in to the console and make programmatic requests to AWS services by using the APIs or AWS Command Line Interface (AWS CLI).
- An *IAM identity* is the IAM resource object that is used to identify and group. You can attach a policy to an IAM identity. Identities include users, groups, and roles.
- A *principal* is a person or application that uses the AWS account root user, an IAM user, or an IAM role to authenticate and make requests to AWS. Principals include assumed roles and federated users—external identities that don't have an AWS account.



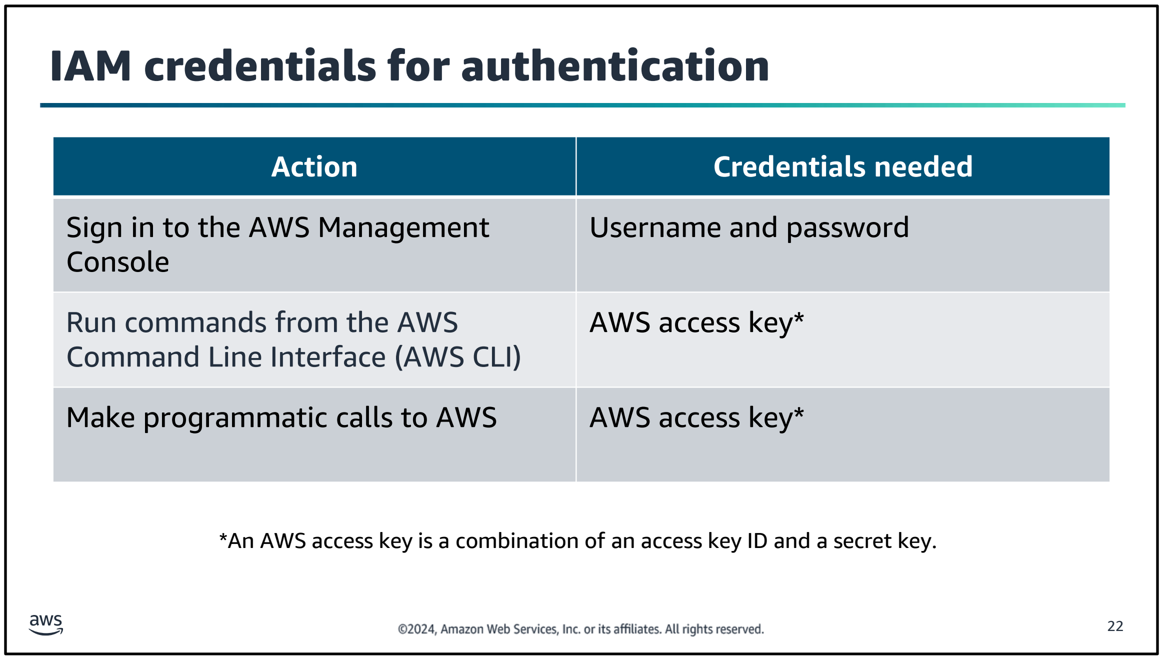
Using IAM to control access to AWS resources

IAM user	IAM group	IAM 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 that is 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

21

You can use IAM to control access to your AWS resources. You achieve this access control by creating *users*, *groups*, and *roles*. You can also enforce access control by attaching *policies* to IAM entities.

For IAM, these terms are defined as follows:
- An *IAM user* is an entity that you create in AWS to represent a person or application that interacts with AWS account services and resources. A user is given a permanent set of credentials, which stay with the user until a forced rotation occurs.
- An *IAM group* is a collection of IAM users. You can use groups to grant the same set of permissions to multiple users.
- An *IAM role* is similar to a user. It's an AWS identity that you can attach permissions policies to. These policies determine what the identity can and can't do in AWS. However, a role doesn't have long-term credentials (such as a password or access keys) associated with it. Instead, when a person or application assumes a role, they are provided with temporary security credentials for the role session.
- An *IAM policy* is a document that explicitly lists permissions. The policy can be attached to an IAM user, group, role, or any combination of these resources.



IAM credentials for authentication

Action	Credentials needed
Sign in to the AWS Management Console	Username and password
Run commands from the AWS Command Line Interface (AWS CLI)	AWS access key*
Make programmatic calls to AWS	AWS access key*

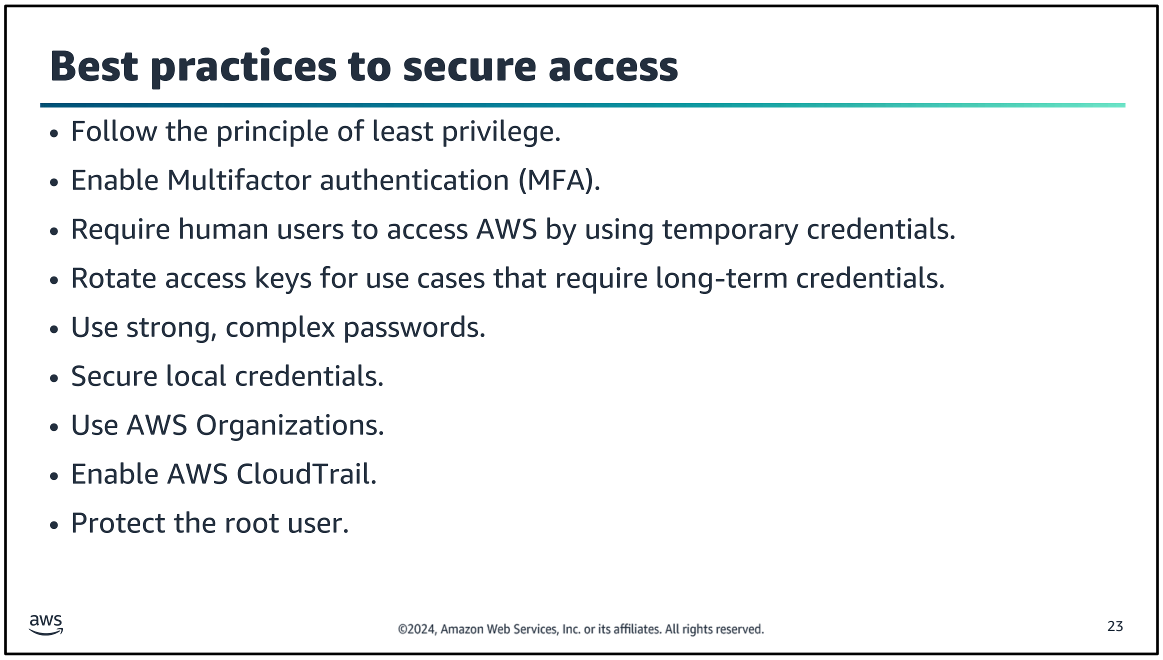
*An AWS access key is a combination of an access key ID and a secret key.

22

When you interact with AWS you must provide AWS credentials. Two primary types of credentials are used for authentication, and which credential type you use depends on how you access AWS:

- To authenticate from the console, you must sign in with your *username and password*.
- To authenticate through the AWS CLI, SDKs, or direct API calls, you must provide an AWS access key. The AWS access key is the combination of an *access key ID and a secret access key*. You cannot use a username and password to authenticate through these methods.

This is a very simplified introduction to using security credentials. For more information, see the AWS Security Credentials link on the content resources page of your online course.



Best practices to secure access

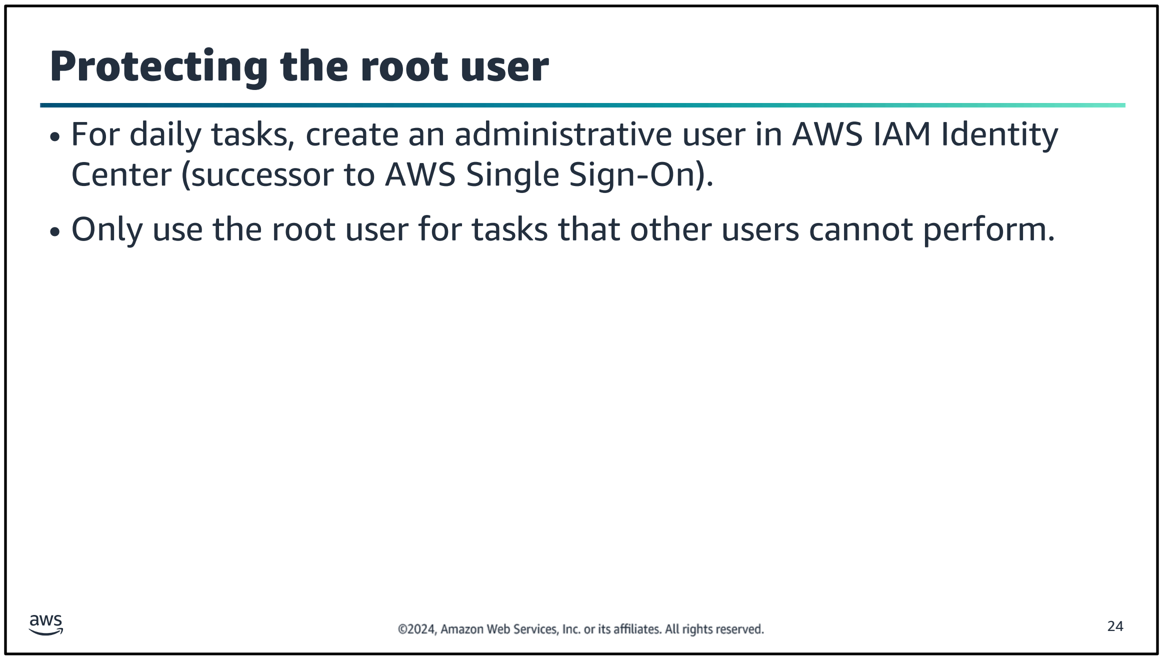
- Follow the principle of least privilege.
- Enable Multifactor authentication (MFA).
- Require human users to access AWS by using temporary credentials.
- Rotate access keys for use cases that require long-term credentials.
- Use strong, complex passwords.
- Secure local credentials.
- Use AWS Organizations.
- Enable AWS CloudTrail.
- Protect the root user.

23

The following are some best practices for securing access:

- **Follow the principle of least privilege:** Assign users, groups, and roles with the minimum necessary permissions to perform their tasks, and avoid providing unnecessary permissions.
- **Enable MFA:** MFA adds an extra layer of security for your root user by requiring a code that is generated by a hardware or software token during sign-in.
- **Require human users to access AWS by using temporary credentials:** Instead of granting long-term credentials to users, use an identity to provide federated access to AWS accounts by assuming IAM roles, which provide temporary credentials. For centralized access management, use AWS IAM Identity Center to manage access to your accounts and permissions within those accounts.
- **Rotate access keys:** for scenarios in which you need IAM users with programmatic access and long-term credentials, use access key last used information to rotate and remove access keys regularly.
- **Use strong, complex passwords:** Ensure that the root user and IAM users have strong, unique passwords that follow AWS password policy guidelines.
- **Secure local credentials:** Secure access keys, passwords, and MFA tokens by storing them in a secure password manager or hardware tokens.
- **Use AWS Organizations:** Use Organizations to consolidate multiple AWS accounts to manage billing, access control, and resources centrally.
- **Enable AWS CloudTrail:** By enabling CloudTrail, you can have a record of all the actions that are taken in an account, which makes it easier to identify possible security risks and comply with auditing requirements.
- **Protect the root user:** When you first create an AWS account, you have a single sign-in identity that has complete access to all AWS services and resources in the account. This is the root user, and it's important to minimize the risk of access through this account given its power. Limit the use of the root user and monitor activity using CloudTrail logs, AWS Trusted Advisor recommendations, and console sign-in events for any suspicious activity related to your root user.

By following these best practices, you can enhance the security of your account setup. For more details and a complete list of current best practices, see the link titled Security best practices in IAM on the content resources page of your online course.



Protecting the root user

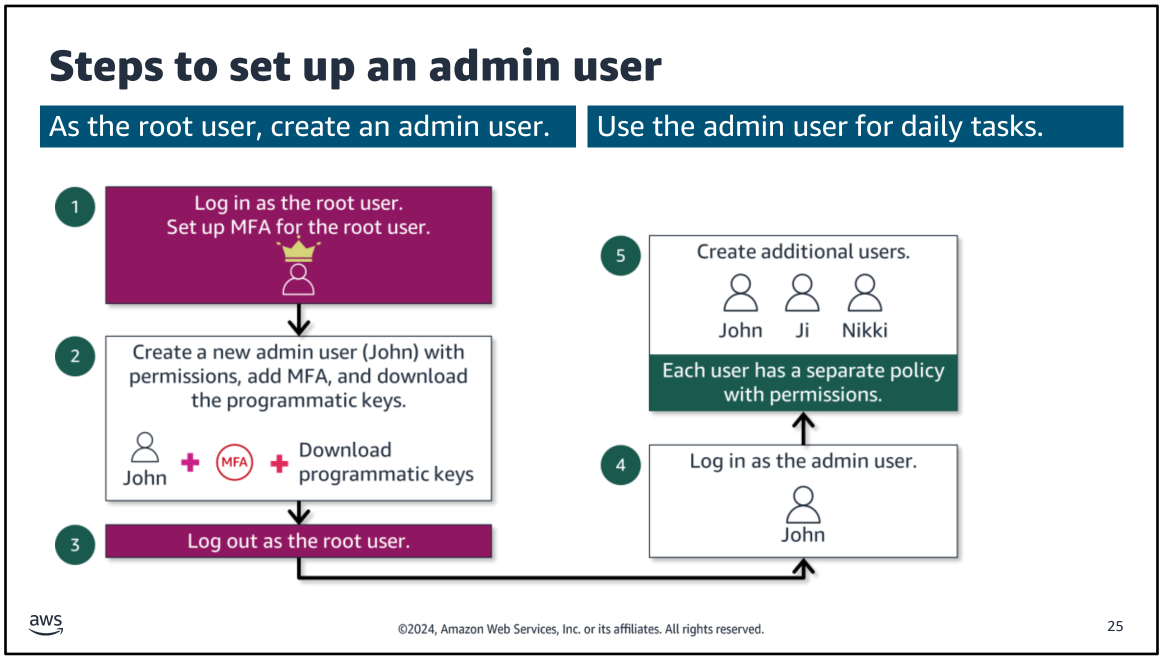
- For daily tasks, create an administrative user in AWS IAM Identity Center (successor to AWS Single Sign-On).
- Only use the root user for tasks that other users cannot perform.

24

Protect your root user credentials like you would protect your credit card number, or any other sensitive or secret data.

Rather than using the root user to do day-to-day administrative tasks, create an administrative user with permissions to perform most administrative tasks. If you use the AWS IAM Identity Center (successor to AWS Single Sign-On) to create the administrative user, temporary rather than long term credentials are used. For instructions, see the AWS IAM Identity Center User Guide link on the content resources page of your online course.

Only use the root user to perform those tasks that cannot be performed by any other user. For more information, see the Tasks That Require Root User Credentials link on the content resources page of your online course.



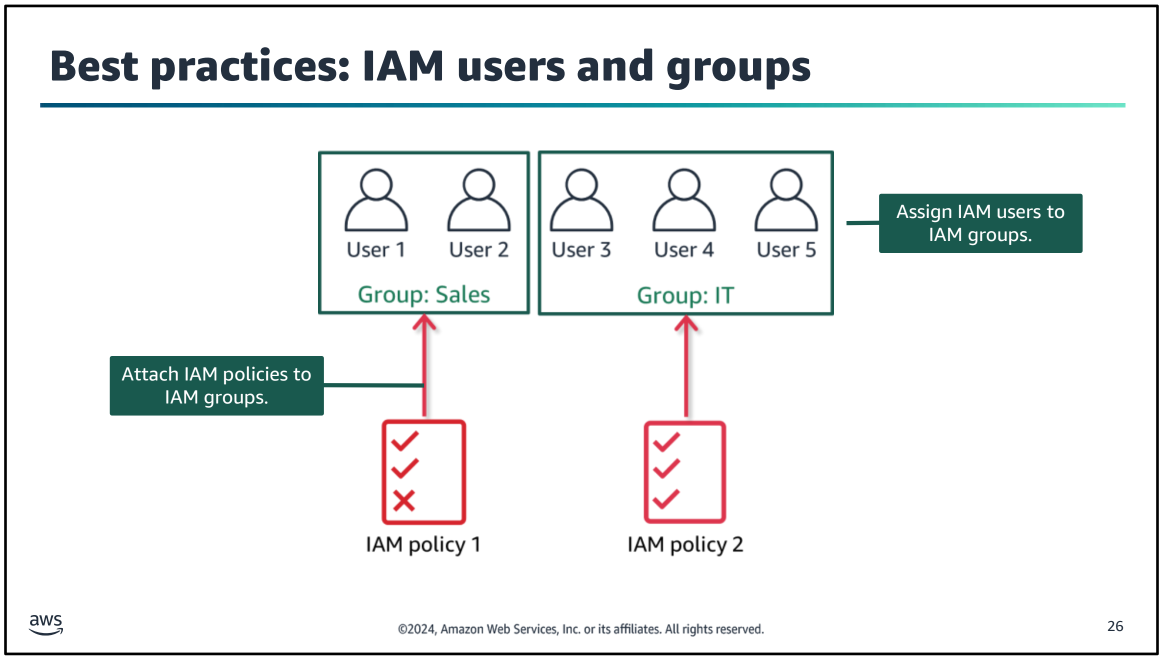
Steps to set up an admin user

As the root user, create an admin user.
Use the admin user for daily tasks.
1
Log in as the root user.
Set up MFA for the root user.
2
Create a new admin user (John) with permissions, add MFA, and download the programmatic keys.
John
MFA
Download programmatic keys
3
Log out as the root user.
4
Log in as the admin user.
John
5
Create additional users.
John
Ji
Nikki
Each user has a separate policy with permissions.
aws
©2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.
25

Instead of using the root user to perform administrative tasks, follow these steps to create an administrator (admin) user:

1. Log in as the root user, and set up MFA on the root user.
2. Create a new admin user, add MFA, and download the programmatic keys. In the example on the slide, John is the admin user.
3. Log out as the root user.
4. Log in as the admin user (in this example, John).
5. Create user accounts. Each user has a separate policy with permissions. In the example on the slide, user accounts are created for John, Ji, and Nikki.

For more information, see the Security Best Practices in IAM link on the content resources page of your online course.



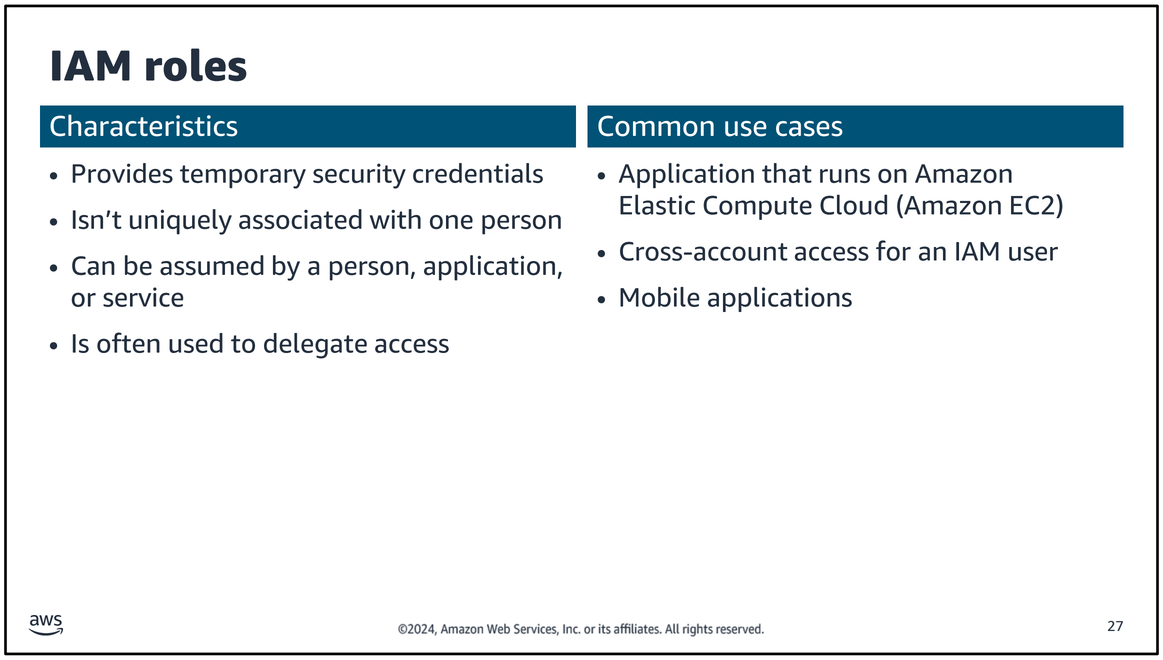
Best practices: IAM users and groups

26

Image description: Diagram of IAM policy 1 attached to an IAM group for sales users and IAM policy 2 attached to a group of IT users. IAM users 1 and 2 are members of the IAM sales group and receive the permissions from the policy that is attached to that group. IAM users 3, 4, and 5 are assigned to the IAM IT group and receive different permissions through the policy that is attached to that group. **End description.**

You can attach an IAM policy to an IAM user, group, role, or any combination of these resources.

To configure long-term access, a best practice is to attach IAM policies to IAM groups, and then assign IAM users to these IAM groups. An IAM user who is a member of an IAM group inherits the permissions that are attached to that group. You can also attach IAM policies directly to an IAM user to further customize the access that is granted through the group.



IAM roles

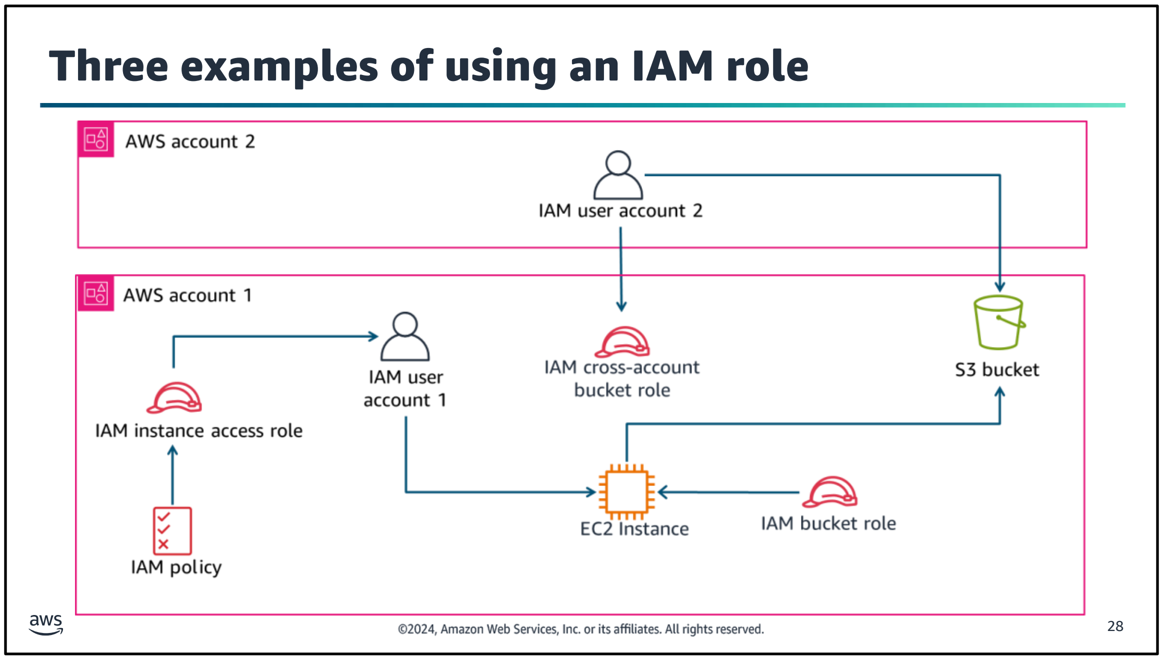
Characteristics	Common use cases
• Provides temporary security credentials • Isn't uniquely associated with one person • Can be assumed by a person, application, or service • Is often used to delegate access	• Application that runs on Amazon Elastic Compute Cloud (Amazon EC2) • Cross-account access for an IAM user • Mobile applications

27

By using an IAM role, you can define a set of permissions for the resources that a user or service needs to access. However, the permissions aren't attached to an IAM user or group. Instead, the permissions are attached to a role, and the user or service *assumes* the role. When a user assumes a role, the user's prior permissions are temporarily forgotten. AWS returns temporary security credentials that the user can then use to make programmatic requests to AWS. When you use IAM roles, you don't need to grant long-term security credentials to each entity that requires access to a resource.

The following examples describe situations when you might need to use temporary security credentials:
- An IAM user in one AWS account needs temporary access to resources in another AWS account.
- User identities that were authenticated by a system outside of AWS (federated users) need to perform AWS tasks and access your AWS resources. You don't want to create and manage IAM users for people outside your organization.
- A mobile app needs access to your AWS resources. You don't want to store AWS credentials with the application, where they can be difficult to rotate and users can potentially extract them.
- An application that runs on an EC2 instance needs access to AWS resources. You don't want to store long-term credentials locally.

When you need to provide an IAM user in another account or a federated user access to your AWS resources, create an IAM role with temporary credentials instead of creating an IAM user in your account. Similarly, for an application that runs on an EC2 instance and requires access to AWS resources, create an IAM role, add the role to an *instance profile*, and attach the profile to the instance. The application can then programmatically assume the role at runtime.

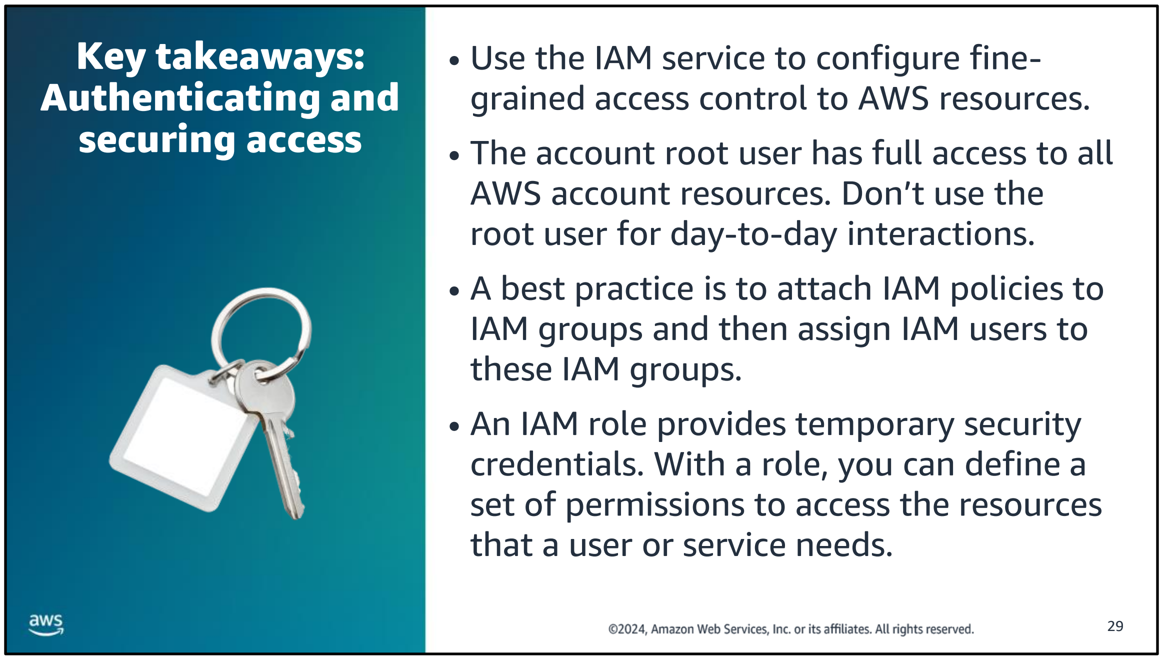


Three examples of using an IAM role

28

The diagram on this slide illustrates three uses cases of an IAM role:

- In the first use case, an IAM user in AWS account 1 needs temporary access to an application that runs on an EC2 instance. You first create an IAM policy with the necessary permissions. You then create an IAM role and attach the policy to the role. The IAM user can now assume the role and access the EC2 instance.
- In the second use case, an application that runs on an EC2 instance needs access to an S3 bucket. After you create an IAM role with the necessary permissions, you add the role to an instance profile and attach the profile to the instance. The application can then access the S3 bucket.
- In the third use case, an IAM user in AWS account 2 needs to access an S3 bucket in AWS account 1. You create a *cross-account* IAM role in account 1 that defines account 2 as a trusted entity. An IAM user in account 2 can then switch roles to the cross-account role and access the S3 bucket in account 1.

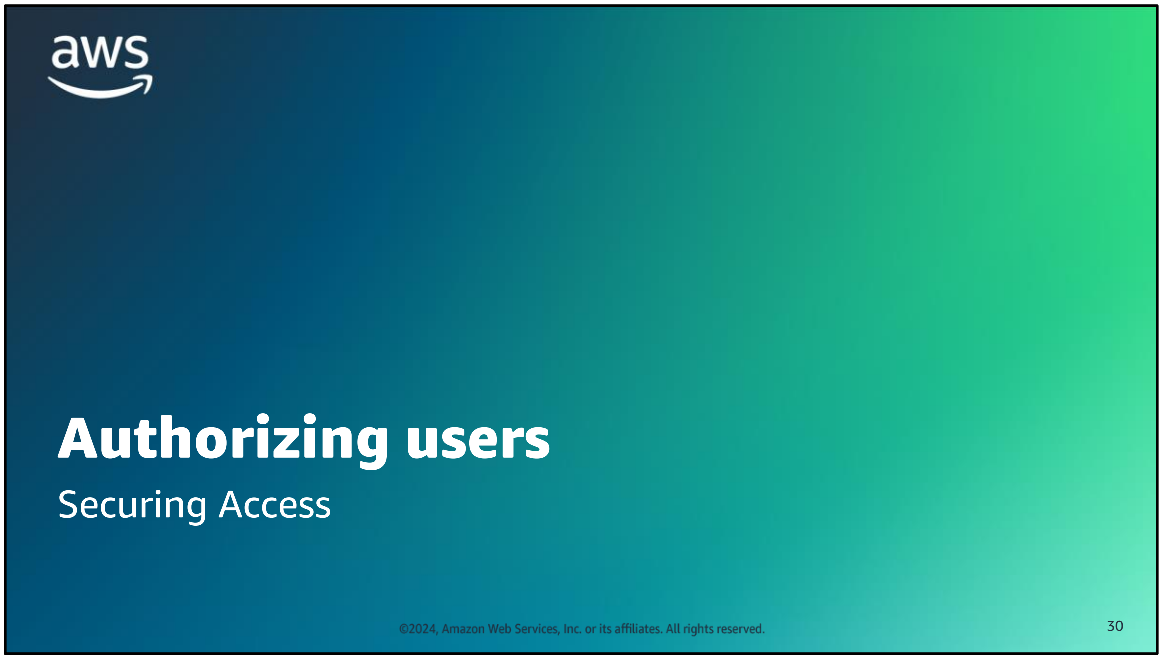


Key takeaways: Authenticating and securing access

- Use the IAM service to configure fine-grained access control to AWS resources.
- The account root user has full access to all AWS account resources. Don't use the root user for day-to-day interactions.
- A best practice is to attach IAM policies to IAM groups and then assign IAM users to these IAM groups.
- An IAM role provides temporary security credentials. With a role, you can define a set of permissions to access the resources that a user or service needs.

29

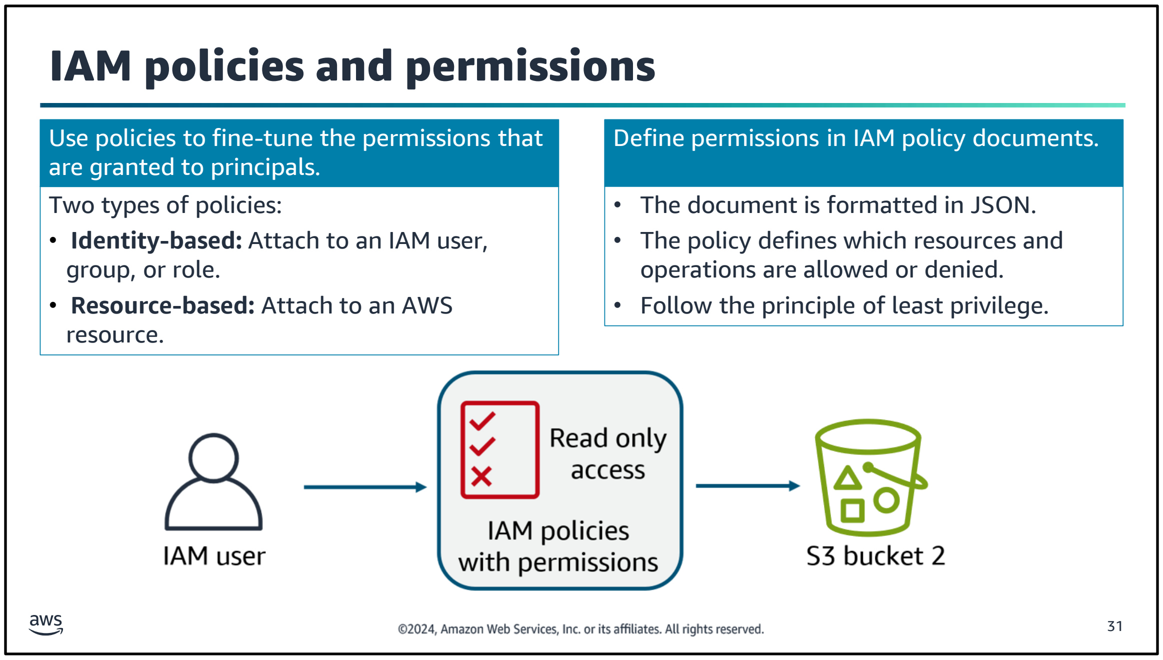
These key takeaways summarize this section.



Authorizing users

Securing Access

This section covers authorizing users.



IAM policies and permissions

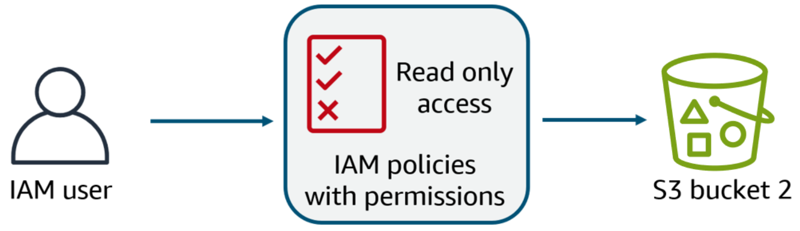
Use policies to fine-tune the permissions that are granted to principals.

Two types of policies:

- **Identity-based:** Attach to an IAM user, group, or role.
- **Resource-based:** Attach to an AWS resource.

Define permissions in IAM policy documents.

- The document is formatted in JSON.
- The policy defines which resources and operations are allowed or denied.
- Follow the principle of least privilege.



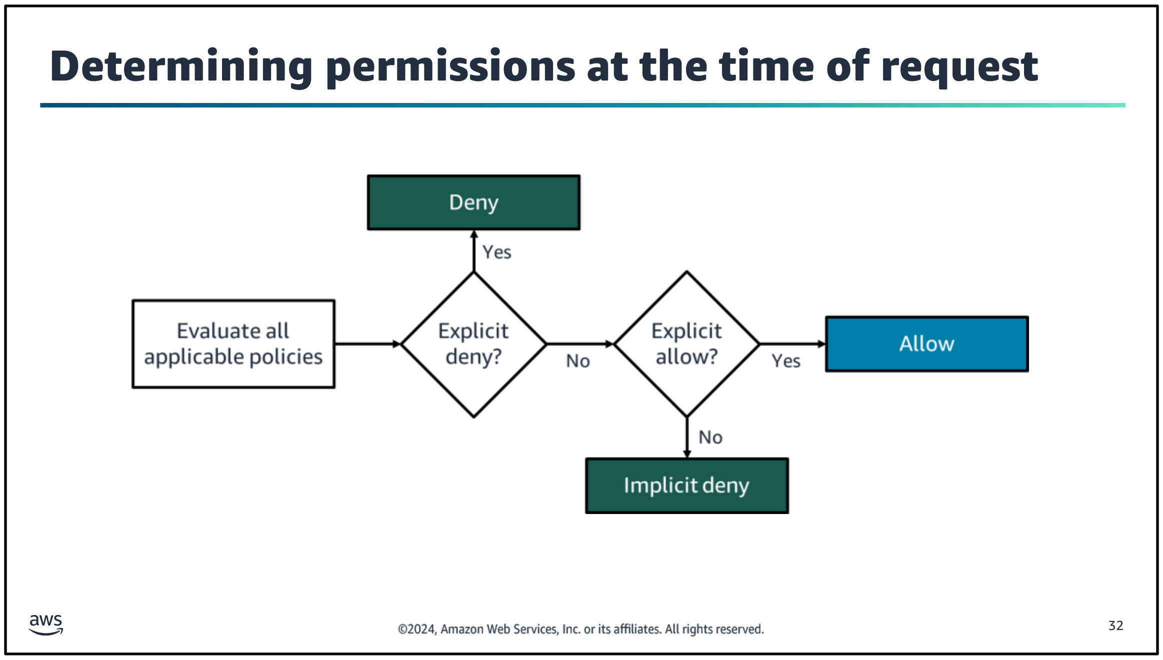
31

A policy is an object that defines permissions for the identity or resource that it's associated with. By using policies, you can fine-tune permissions that are granted to principals. Examples of principals include IAM users, IAM roles, and AWS services.

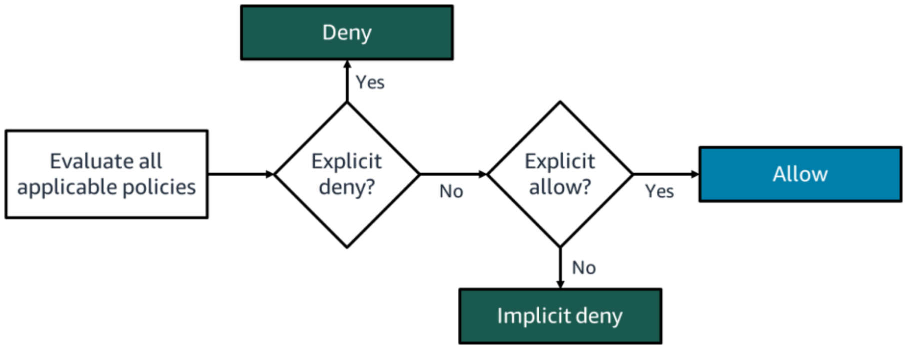
Two types of policies exist: identity-based policies and resource-based policies. The biggest difference is where they are applied. *Identity-based* policies are attached to an IAM user, group, or role and indicate what the user is permitted to do. *Resource-based* policies are attached to a resource and indicate what a specified user (or group of users) is permitted to do with the resource.

Most policies are stored in AWS as JSON documents. AWS evaluates these policies when an IAM principal makes a request. Permissions in the policies determine whether the request is allowed or denied.

A best practice is to follow the principle of least privilege, which was covered earlier in the module.



Determining permissions at the time of request



32

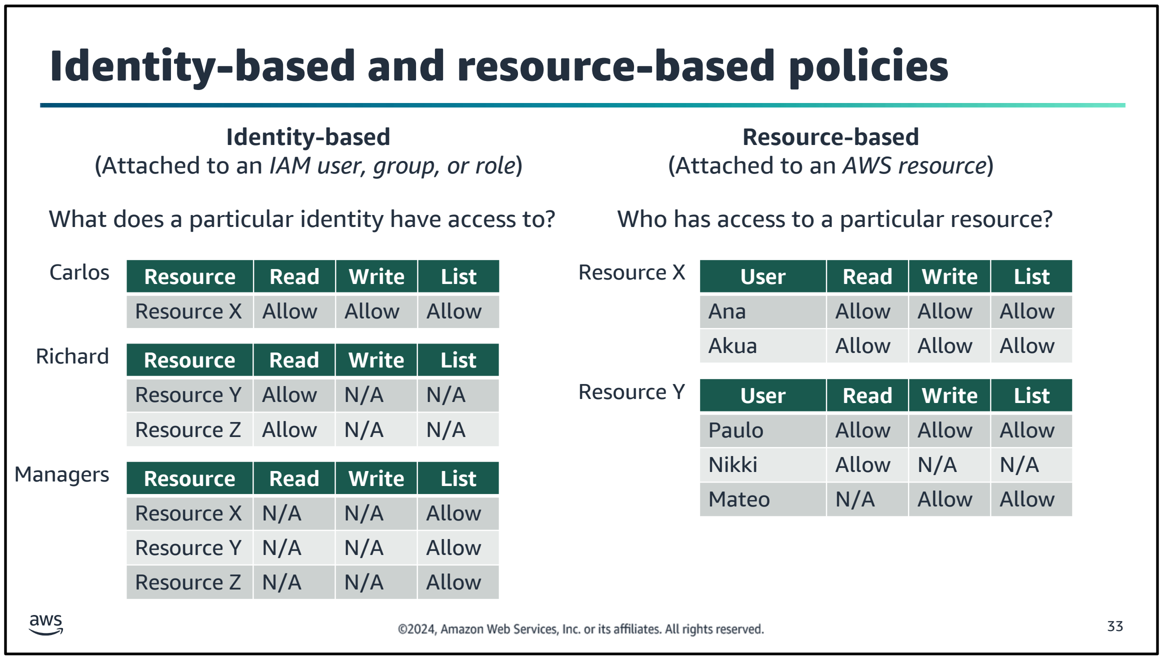
When IAM determines whether an action is allowed, the service first checks for the existence of any applicable explicit denial policy. If an explicit denial doesn't exist, then IAM checks for any applicable explicit allow policy. If an explicit deny or explicit allow policy doesn't exist, IAM reverts to the default and denies access. This process is referred to as an *implicit deny*. A user will be permitted to take the action only if the requested action is not explicitly denied and is explicitly allowed.

The diagram on the slide shows the logic that AWS uses to evaluate IAM policies. AWS evaluates all applicable policies and uses the following evaluation logic:

- By default, all requests are denied.
- An explicit allow overrides the default deny.
- An explicit deny overrides any explicit allow.

The order that the policies are evaluated in has no effect on the outcome of the evaluation. All policies are evaluated, and the result is always that the request is either allowed or denied. Suppose that a conflict occurs (one policy allows an action and another policy denies an action). Then, the most restrictive policy (that is, the policy that denies the action) is applied.

When you develop IAM policies, it can be difficult to determine whether access to a resource will be granted to an IAM entity. The IAM policy simulator is a helpful tool to test and troubleshoot IAM policies. For more information, see the Testing IAM Policies with the IAM Policy Simulator link on the content resources page of your online course.



Identity-based and resource-based policies

Identity-based
(Attached to an *IAM user, group, or role*)

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
(Attached to an *AWS resource*)

Who has access to a particular resource?

Resource X

User	Read	Write	List
Ana	Allow	Allow	Allow
Akua	Allow	Allow	Allow

Resource Y

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

33

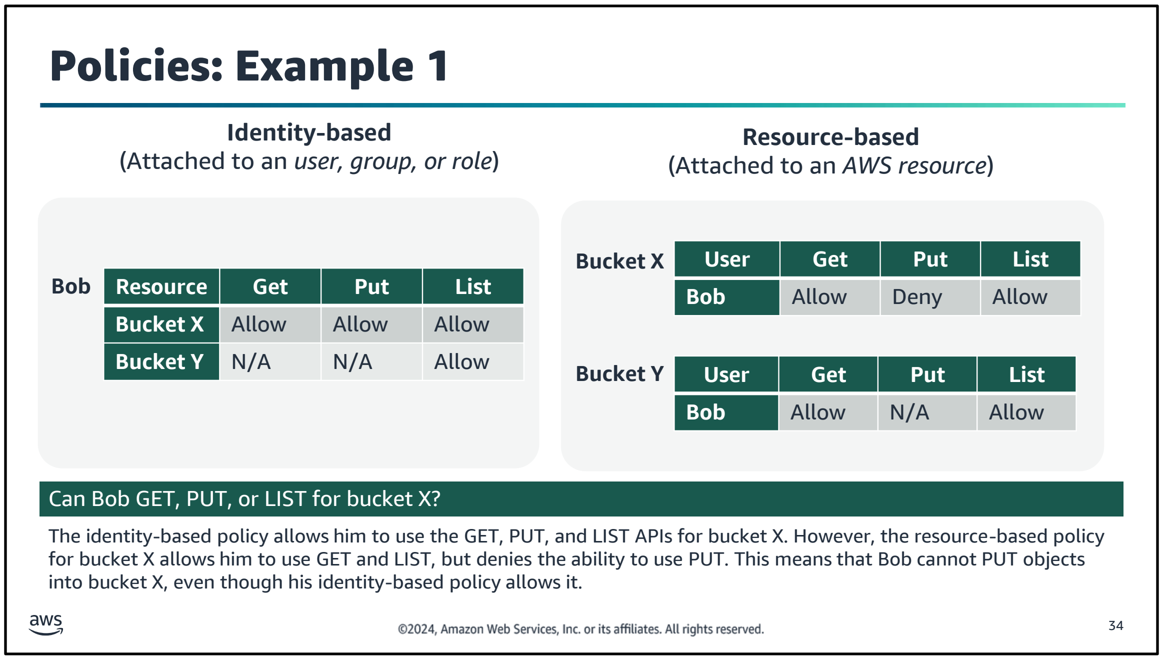
Note: In the tables on the slide, N/A means not applicable because the policy doesn't specify permissions for that particular action.

To reiterate, two types of IAM policies can grant permissions: identity-based policies and resource-based policies. The difference between the two types is where they are applied, which depends on the type of access that you're authorizing or restricting.

Identity-based policies are attached to an IAM user, group, or role. They indicate what that identity can do. For example, you could grant a user the ability to access a DynamoDB table.

Resource-based policies are attached to a resource. They indicate what a specified user (or group of users) is permitted to do with the resource. For example, you could use a resource-based policy to grant access to an S3 bucket or to grant cross-account access between two trusted AWS accounts.

For more information, see the Identity-Based Policies and Resource-Based Policies link on the content resources page of your online course.



Policies: Example 1

Identity-based
(Attached to an *user, group, or role*)

Bob

Resource	Get	Put	List
Bucket X	Allow	Allow	Allow
Bucket Y	N/A	N/A	Allow

Resource-based
(Attached to an *AWS resource*)

Bucket X

User	Get	Put	List
Bob	Allow	Deny	Allow

Bucket Y

User	Get	Put	List
Bob	Allow	N/A	Allow

Can Bob GET, PUT, or LIST for bucket X?

The identity-based policy allows him to use the GET, PUT, and LIST APIs for bucket X. However, the resource-based policy for bucket X allows him to use GET and LIST, but denies the ability to use PUT. This means that Bob cannot PUT objects into bucket X, even though his identity-based policy allows it.

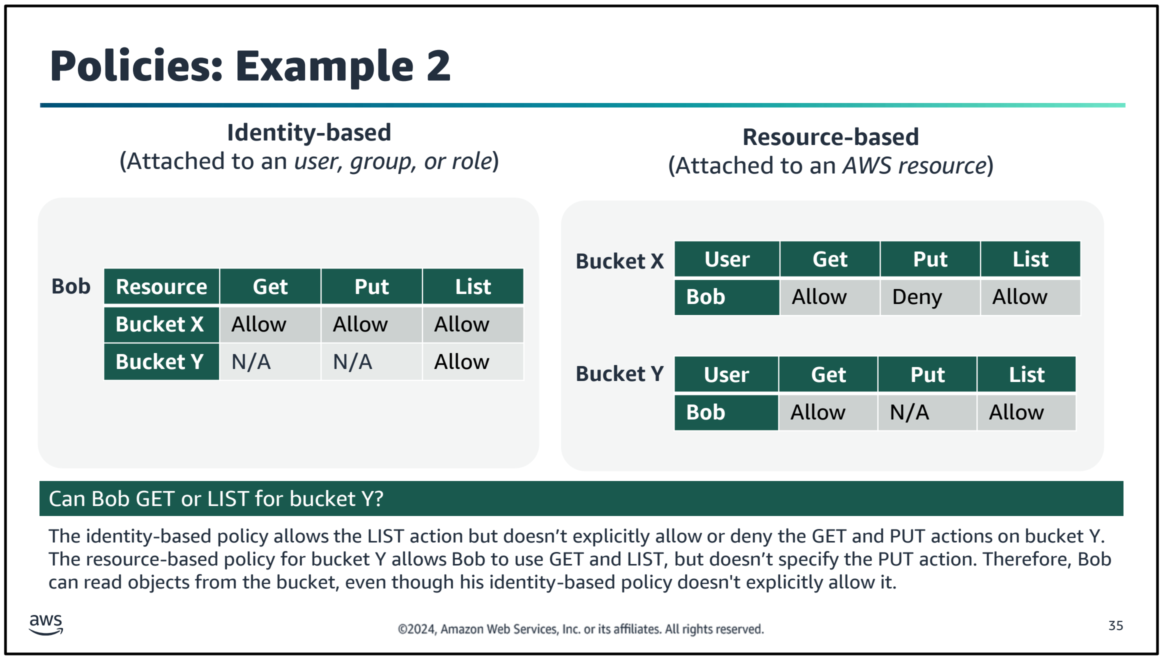
©2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

34

Note: In the tables on the slide, N/A means not applicable because the policy doesn't specify permissions for that particular action.

Amazon S3 supports two types of access control mechanisms: identity-based (or user based) and resource-based.

In the example on the slide, an IAM user named Bob has an identity-based policy attached. The policy allows him to use the GET, PUT, and LIST APIs for S3 bucket X. However, the resource-based policy for bucket X allows him to use GET and LIST, but denies the ability to use PUT. This means that Bob cannot PUT objects into bucket X, even though his identity-based policy allows it.



Policies: Example 2

Identity-based
(Attached to an *user, group, or role*)

Bob

Resource	Get	Put	List
Bucket X	Allow	Allow	Allow
Bucket Y	N/A	N/A	Allow

Resource-based
(Attached to an *AWS resource*)

Bucket X

User	Get	Put	List
Bob	Allow	Deny	Allow

Bucket Y

User	Get	Put	List
Bob	Allow	N/A	Allow

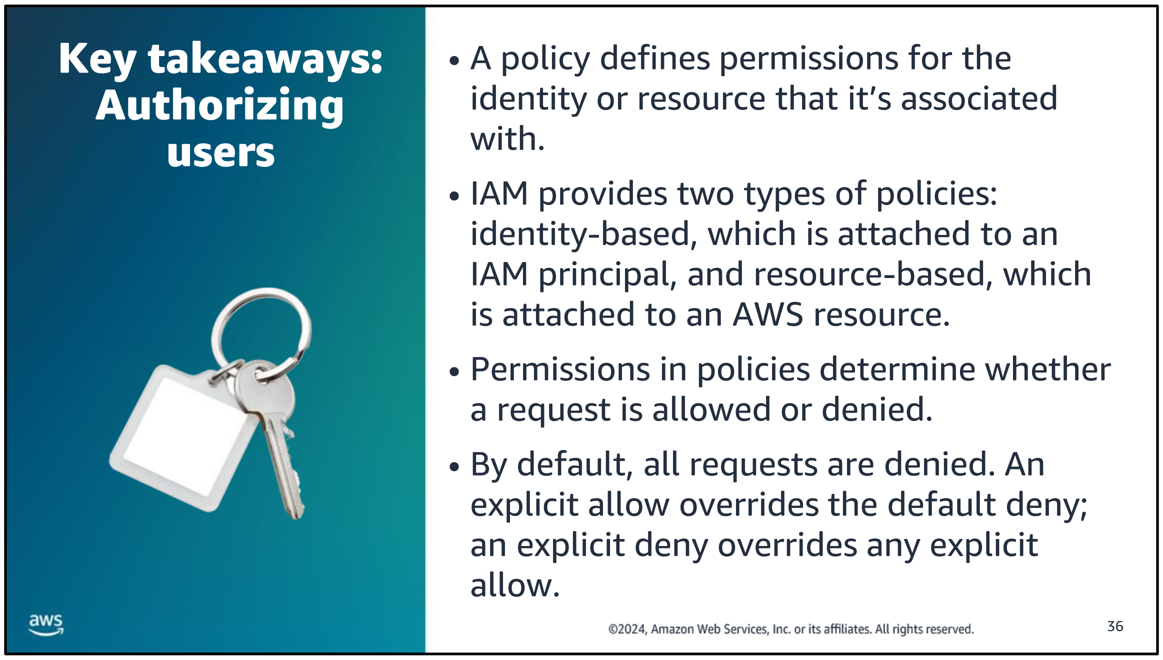
Can Bob GET or LIST for bucket Y?

The identity-based policy allows the LIST action but doesn't explicitly allow or deny the GET and PUT actions on bucket Y. The resource-based policy for bucket Y allows Bob to use GET and LIST, but doesn't specify the PUT action. Therefore, Bob can read objects from the bucket, even though his identity-based policy doesn't explicitly allow it.

35

Note: In the tables on the slide, N/A means not applicable because the policy doesn't specify permissions for that particular action.

The example on this slide focuses on Bob's permissions for bucket Y. Bob's identity-based policy allows the LIST action. The policy doesn't explicitly allow or deny the GET and PUT actions on bucket Y. The resource-based policy for bucket Y allows Bob to use GET and LIST, but doesn't specify the PUT action. Therefore, Bob can read objects from the bucket, even though his identity-based policy doesn't explicitly allow it.



Key takeaways: Authorizing users

- A policy defines permissions for the identity or resource that it's associated with.
- IAM provides two types of policies: identity-based, which is attached to an IAM principal, and resource-based, which is attached to an AWS resource.
- Permissions in policies determine whether a request is allowed or denied.
- By default, all requests are denied. An explicit allow overrides the default deny; an explicit deny overrides any explicit allow.

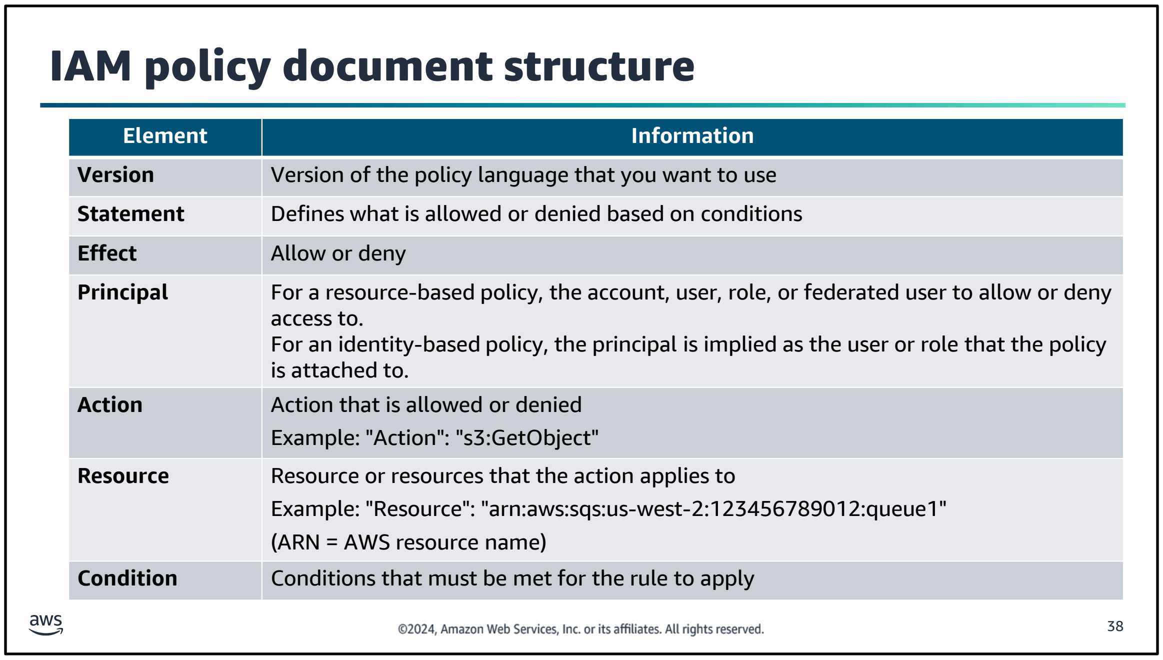
36

These key takeaways summarize this section.

Parts of an IAM policy

Securing Access

This section focuses on the structure and elements of an IAM policy.



IAM policy document structure

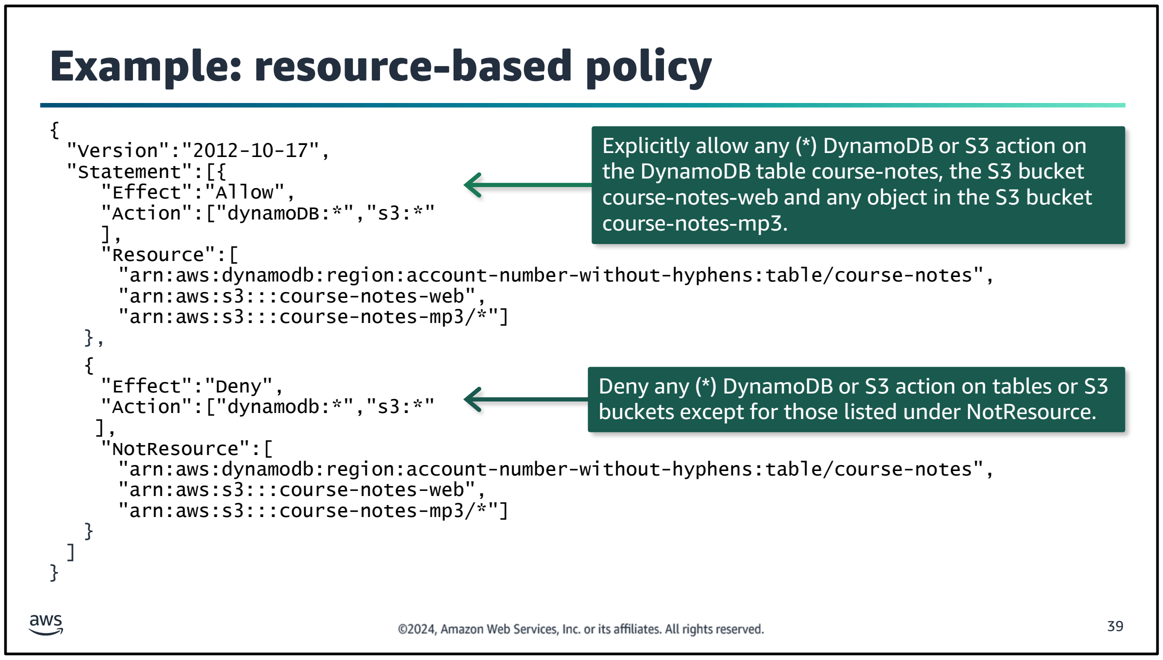
Element	Information
Version	Version of the policy language that you want to use
Statement	Defines what is allowed or denied based on conditions
Effect	Allow or deny
Principal	For a resource-based policy, the account, user, role, or federated user to allow or deny access to. For an identity-based policy, the principal is implied as the user or role that the policy is attached to.
Action	Action that is allowed or denied Example: "Action": "s3:GetObject"
Resource	Resource or resources that the action applies to Example: "Resource": "arn:aws:sqs:us-west-2:123456789012:queue1" (ARN = AWS resource name)
Condition	Conditions that must be met for the rule to apply

38

IAM policies are stored in AWS as JSON documents. Each statement includes information about a single permission. If a policy includes multiple statements, AWS applies a logical OR across the statements when evaluating them.

The following are common elements that are found in an IAM policy document:

- **Version:** Specify the version of the policy language that you want to use.
- **Statement:** Use this main policy element as a container for the following elements. You can include more than one statement in a policy:
- **Effect:** Use *Allow* or *Deny* to indicate whether the policy allows or denies access.
- **Principal:** If you create a *resource-based* policy, you must indicate the account, user, role, or federated user that you would like to allow or deny access to. If you are creating an IAM policy to attach to a user or role, you cannot include this element. The principal is implied as the user or role that the policy is attached to.
- **Action:** Include a list of actions that the policy allows or denies.
- **Resource:** If you are creating an IAM policy to attach to a user or role, you must specify a list of resources to which the actions apply. If you create a *resource-based* policy, this element is optional.
- **Condition (optional):** Specify the circumstances where the policy grants permissions.



Example: resource-based policy

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

Explicitly allow any (*) DynamoDB or S3 action on the DynamoDB table course-notes, the S3 bucket course-notes-web and any object in the S3 bucket course-notes-mp3.

Deny any (*) DynamoDB or S3 action on tables or S3 buckets except for those listed under NotResource.

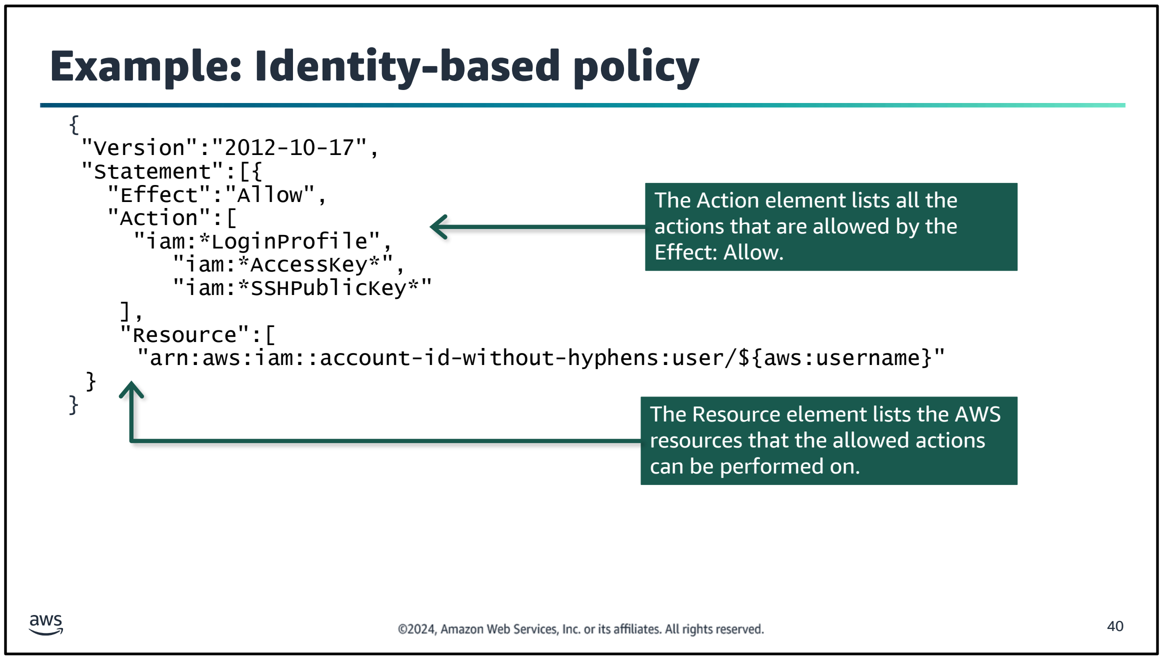
39

The policy uses a combination of an explicit allow and an explicit deny to limit access within an AWS account. The example IAM policy on this slide allows access only to the following resources:

- The DynamoDB table course-notes.
- The S3 buckets course-notes-web and course-notes-mp3.

The explicit deny ("Effect":"Deny") element uses the NotResource element to ensure that users cannot use any other DynamoDB or S3 actions or resources, except those that are specified in the policy. This is the case even if permissions have been granted in another policy. An explicit deny statement takes precedence over an allow statement.

In its totality, this IAM policy allows access to specific DynamoDB and Amazon S3 resources. It then explicitly denies access to any other DynamoDB or Amazon S3 resources in the account.



Example: Identity-based policy

```
{
 "Version":"2012-10-17",
 "Statement":[{
   "Effect":"Allow",
   "Action":[
     "iam:*LoginProfile",
        "iam:*AccessKey*",
        "iam:*SSHPublicKey*"
    ],
    "Resource":[
     "arn:aws:iam::account-id-without-hyphens:user/${aws:username}"
  }
 }
```

The Action element lists all the actions that are allowed by the Effect: Allow.

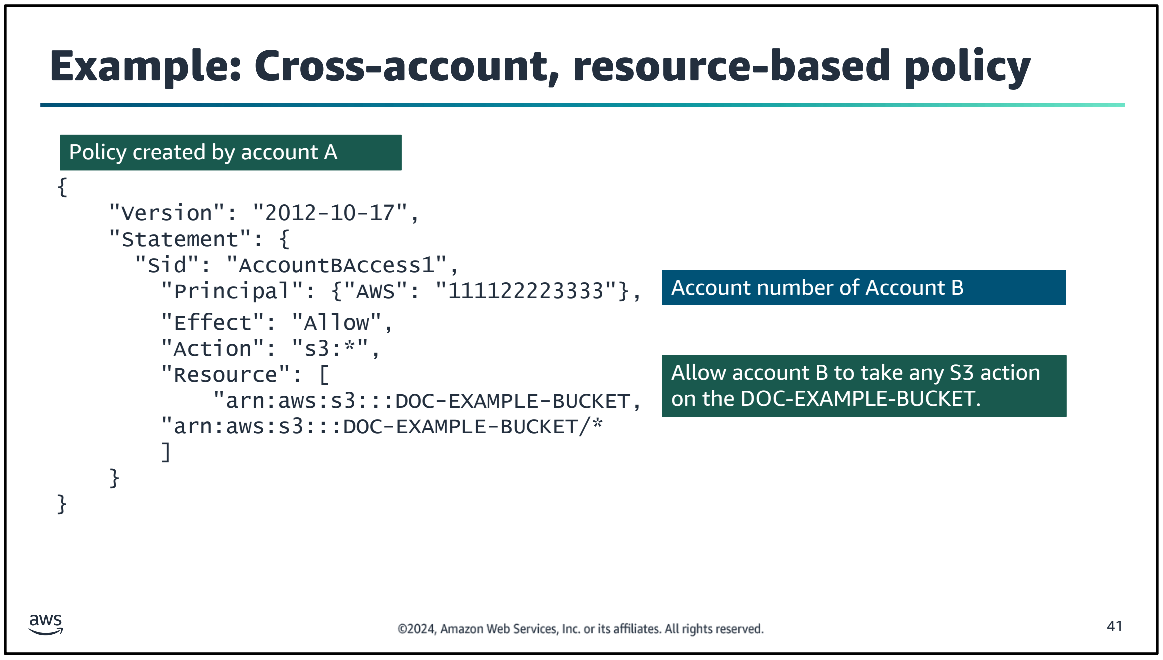
The Resource element lists the AWS resources that the allowed actions can be performed on.

40

After you apply an identity-based policy to a user, group, or role, the policy allows or denies the ability for the entity to perform specific actions.

The slide shows an example of an identity-based policy. If this policy was attached to an IAM user, the user would be able to perform the following actions:
- Create, delete, get, or update their own password by using IAM LoginProfile actions.
- Create, delete, list, or update their own access key by using IAM AccessKey actions.
- Create, delete, get, list, or update their own Secure Shell (SSH) keys by using IAM SSHPublicKey actions.

The actions in the policy include wildcards, which are indicated with an asterisk (*). This format provides a convenient way to include a set of related actions. Notice that the policy gets the AWS username dynamically, as defined in the *Resource* key.



Example: Cross-account, resource-based policy

Policy created by account A

```
{
    "Version": "2012-10-17",
    "Statement": {
      "Sid": "AccountBAccess1",
        "Principal": {"AWS": "111122223333"},
        "Effect": "Allow",
        "Action": "s3:*",
        "Resource": [
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET,
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*
        ]
    }
}
```

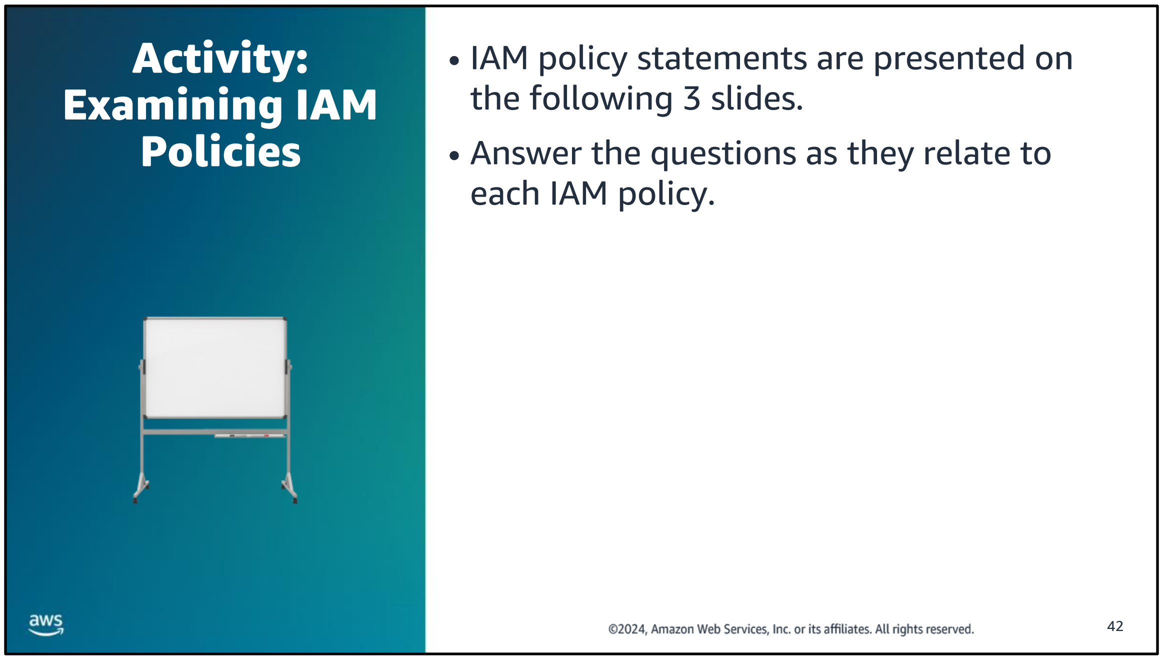
Account number of Account B

Allow account B to take any S3 action on the DOC-EXAMPLE-BUCKET.

41

The slide provides an example of a resource-based policy that allows cross-account access. In this scenario, account A created the resource-based policy. The policy grants account B access to perform *any* Amazon S3 API operation—which is indicated by the asterisk (*)—on the S3 bucket named DOC-EXAMPLE-BUCKET in account A.

This S3 bucket policy doesn't specify any IAM users, groups, or roles. Instead, it specifies account B (AWS account number 111122223333). Account B should create an IAM user policy to allow a user in account B to access account A's bucket.

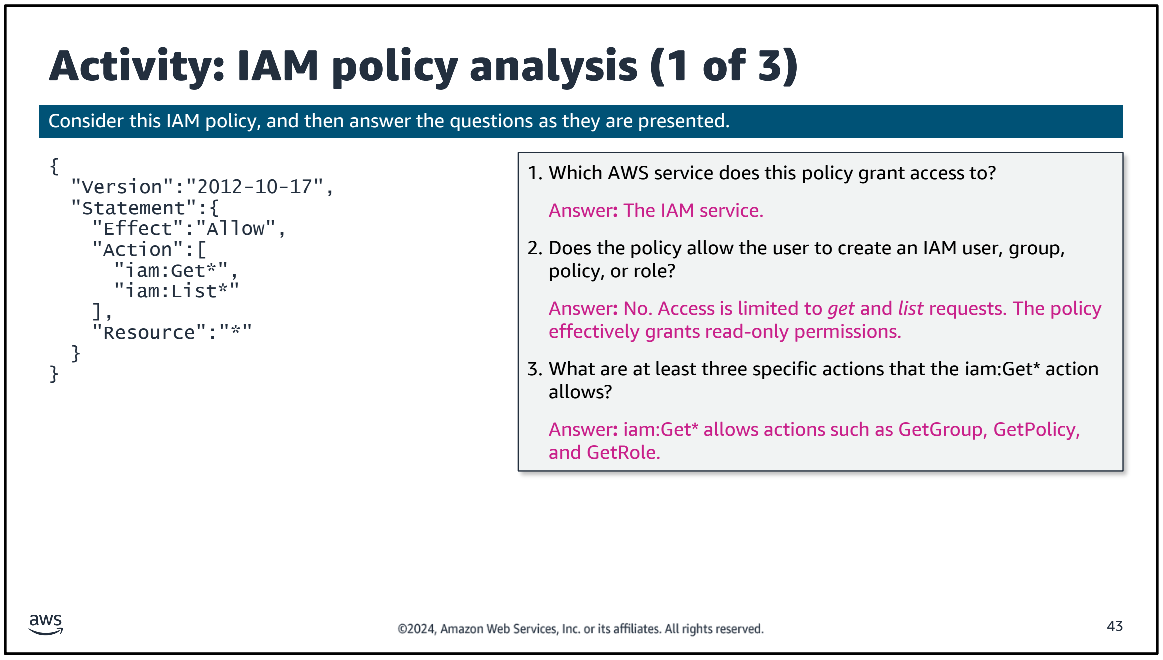


Activity: Examining IAM Policies

- IAM policy statements are presented on the following 3 slides.
- Answer the questions as they relate to each IAM policy.

42

In this educator-led activity, you will review a few examples of IAM policies. For each policy, you will be asked questions about what the policy allows you to do. Your educator will lead you in a discussion of each question and reveal the correct answers one at a time.



Activity: IAM policy analysis (1 of 3)

Consider this IAM policy, and 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 access to?

   Answer**:** The IAM service.

2. Does the policy allow the user to create an IAM user, group, policy, or role?

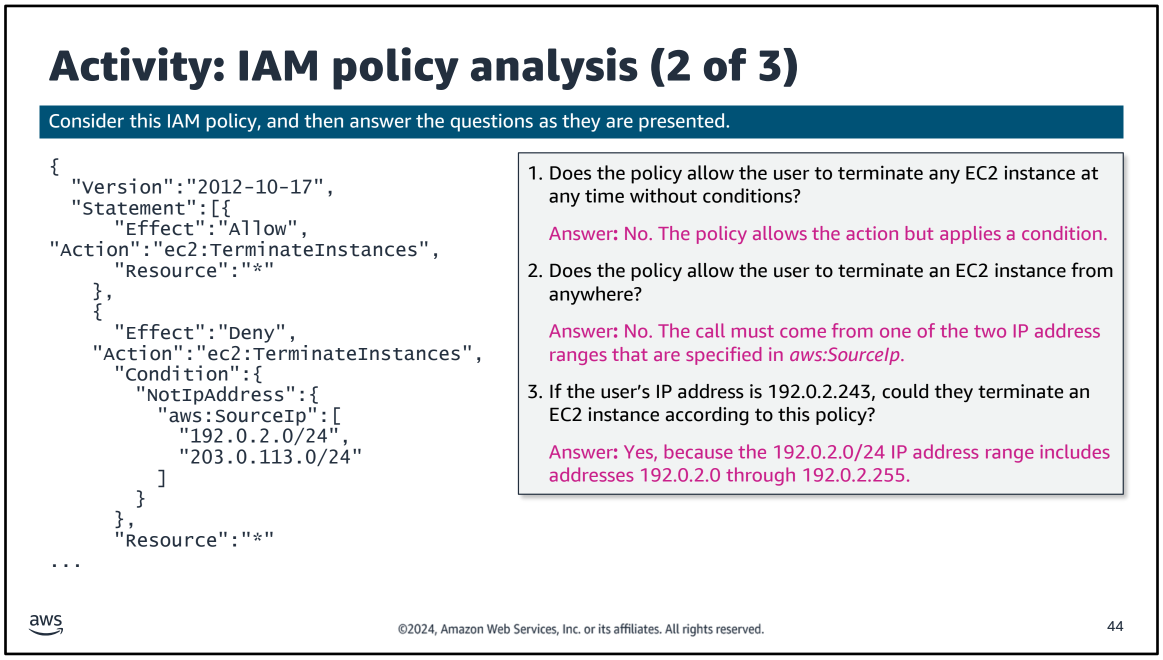
   Answer**:** No. Access is limited to *get* and *list* requests. The policy effectively grants read-only permissions.

3. What are at least three specific actions that the iam:Get* action allows?

   Answer**:** iam:Get* allows actions such as GetGroup, GetPolicy, and GetRole.

43

Look at the example IAM policy. Your educator will ask a series of questions to assess whether you understand what actions this policy will allow and deny. Use the link provided in the content resources page of your online course to answer the question.



Activity: IAM policy analysis (2 of 3)

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

```
{
  "Version":"2012-10-17",
  "Statement":[{
      "Effect":"Allow",
"Action":"ec2:TerminateInstances",
      "Resource":"*"
    },
    {
      "Effect":"Deny",
    "Action":"ec2:TerminateInstances",
      "Condition":{
        "NotIpAddress":{
          "aws:SourceIp":[
            "192.0.2.0/24",
            "203.0.113.0/24"
          ]
        }
      },
      "Resource":"*"
...
```

1. Does the policy allow the user to terminate any EC2 instance at any time without conditions?

   Answer**:** No. The policy allows the action but applies a condition.

2. Does the policy allow the user to terminate an EC2 instance from anywhere?

   Answer**:** No. The call must come from one of the two IP address ranges that are specified in *aws:SourceIp*.

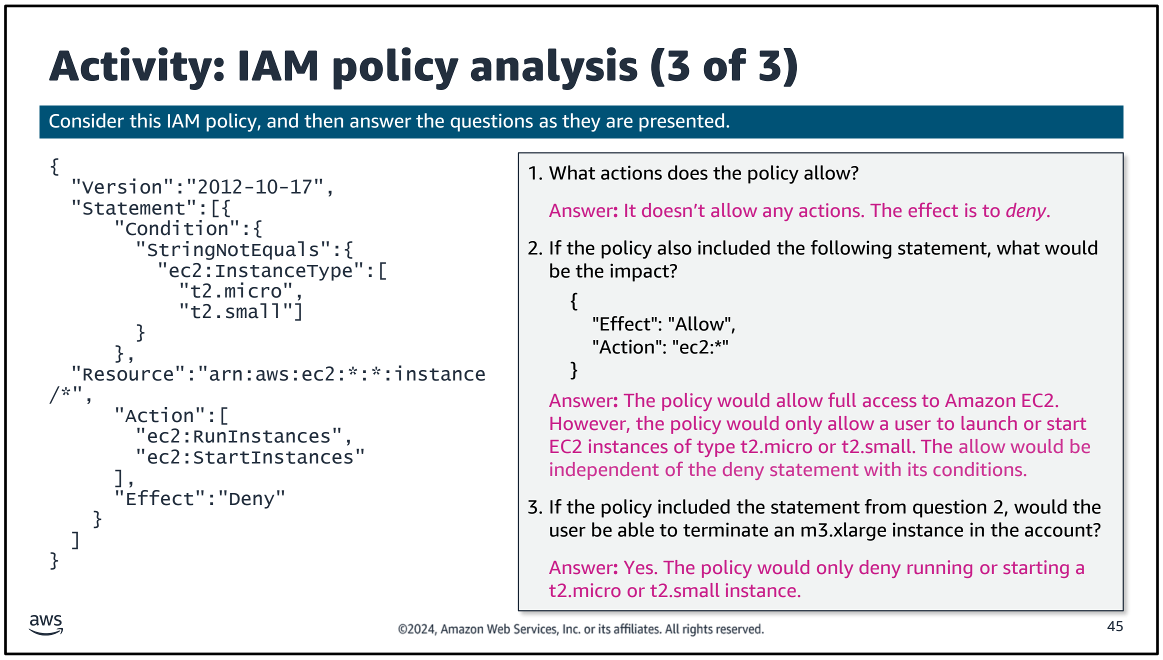
3. If the user's IP address is 192.0.2.243, could they terminate an EC2 instance according to this policy?

   Answer**:** Yes, because the 192.0.2.0/24 IP address range includes addresses 192.0.2.0 through 192.0.2.255.

44

Analyze this second example of an IAM policy. Your educator will ask a series of questions to assess whether you understand what actions this policy will allow and deny.

For question 3, to calculate the range of a CIDR block, you can use a CIDR to IPv4 Conversion tool resource located in the content resources page of your online course.



Activity: IAM policy analysis (3 of 3)

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

```
{
  "Version":"2012-10-17",
  "Statement":[{
      "Condition":{
        "StringNotEquals":{
          "ec2:InstanceType":[
            "t2.micro",
            "t2.small"]
        }
      },
  "Resource":"arn:aws:ec2:*:*:instance
/*",
      "Action":[
        "ec2:RunInstances",
        "ec2:StartInstances"
      ],
      "Effect":"Deny"
    }
  ]
}
```

1. What actions does the policy allow?

   Answer**:** It doesn't allow any actions. The effect is to *deny*.

2. If the policy also included the following statement, what would be the impact?

   ```
   {
       "Effect": "Allow",
       "Action": "ec2:*"
   }
   ```

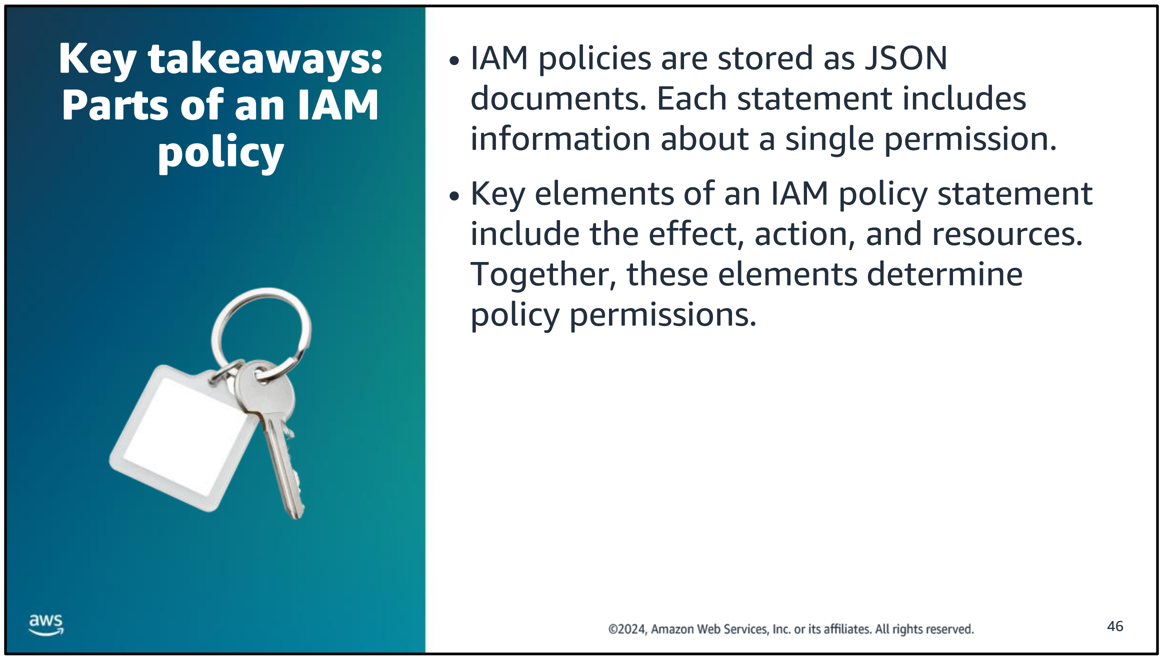
   Answer**:** The policy would allow full access to Amazon EC2. However, the policy would only allow a user to launch or start EC2 instances of type t2.micro or t2.small. The allow would be independent of the deny statement with its conditions.

3. If the policy included the statement from question 2, would the user be able to terminate an m3.xlarge instance in the account?

   Answer**:** Yes. The policy would only deny running or starting a t2.micro or t2.small instance.

45

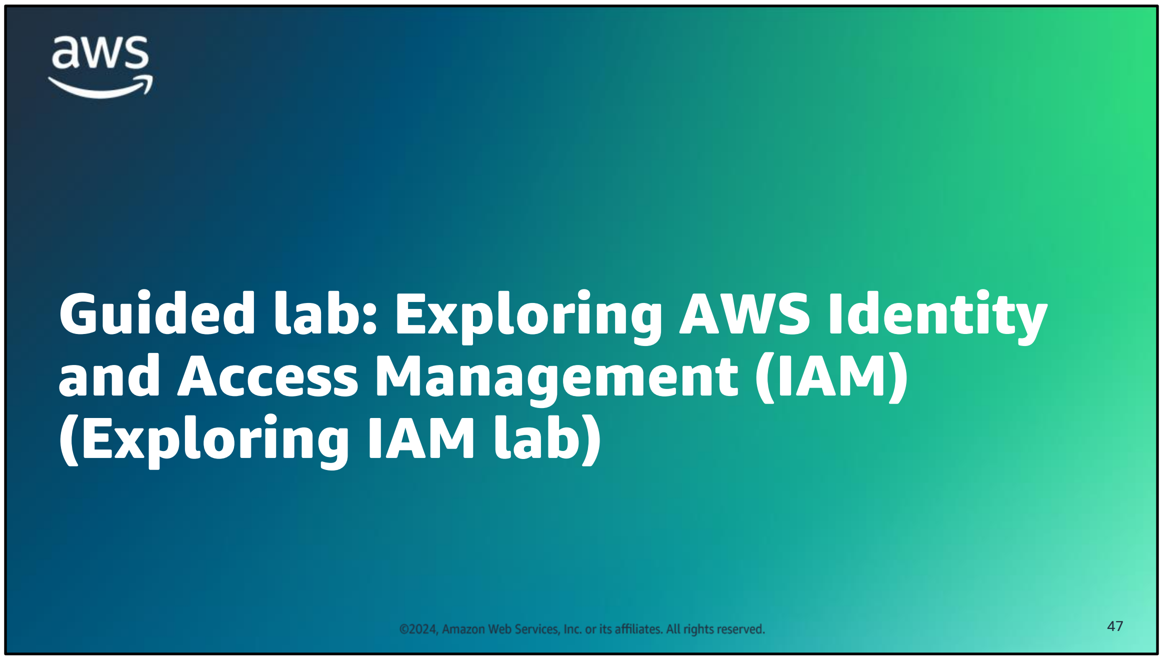
Observe this example of an IAM policy. Your educator will ask a series of questions to assess whether you understand what actions this policy will allow and deny.



Key takeaways: Parts of an IAM policy

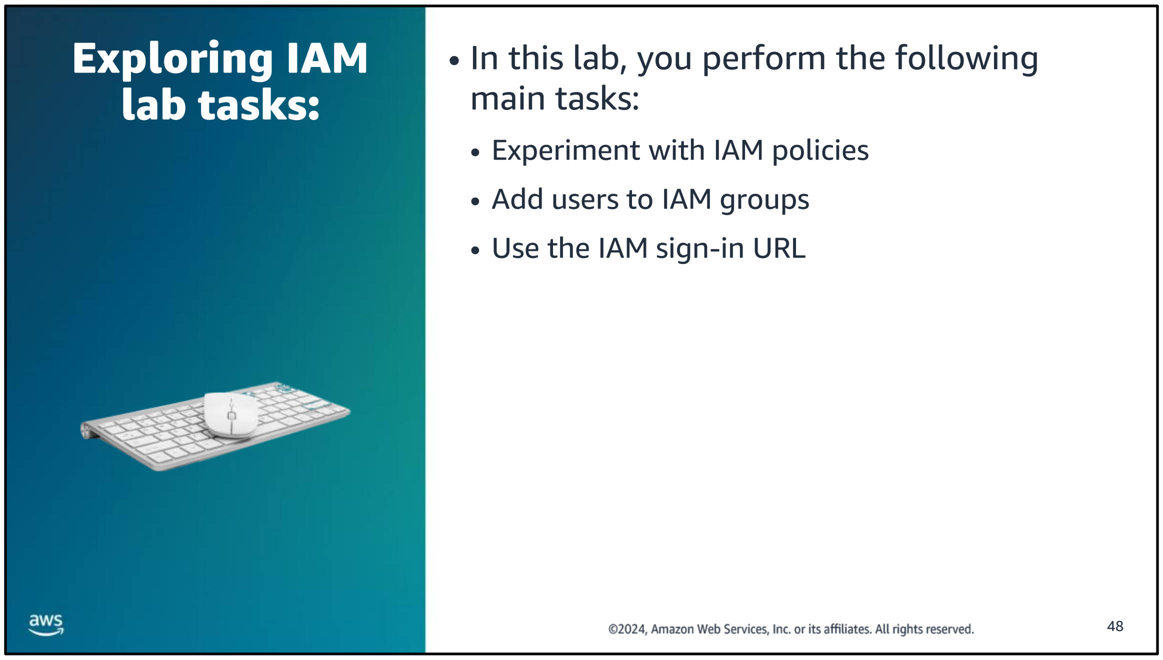
- IAM policies are stored as JSON documents. Each statement includes information about a single permission.
- Key elements of an IAM policy statement include the effect, action, and resources. Together, these elements determine policy permissions.

These key takeaways summarize this section.



Guided lab: Exploring AWS Identity and Access Management (IAM) (Exploring IAM lab)

You will now complete a lab. The next slides summarize what you will do in the lab, and you will find the detailed instructions in the lab environment.

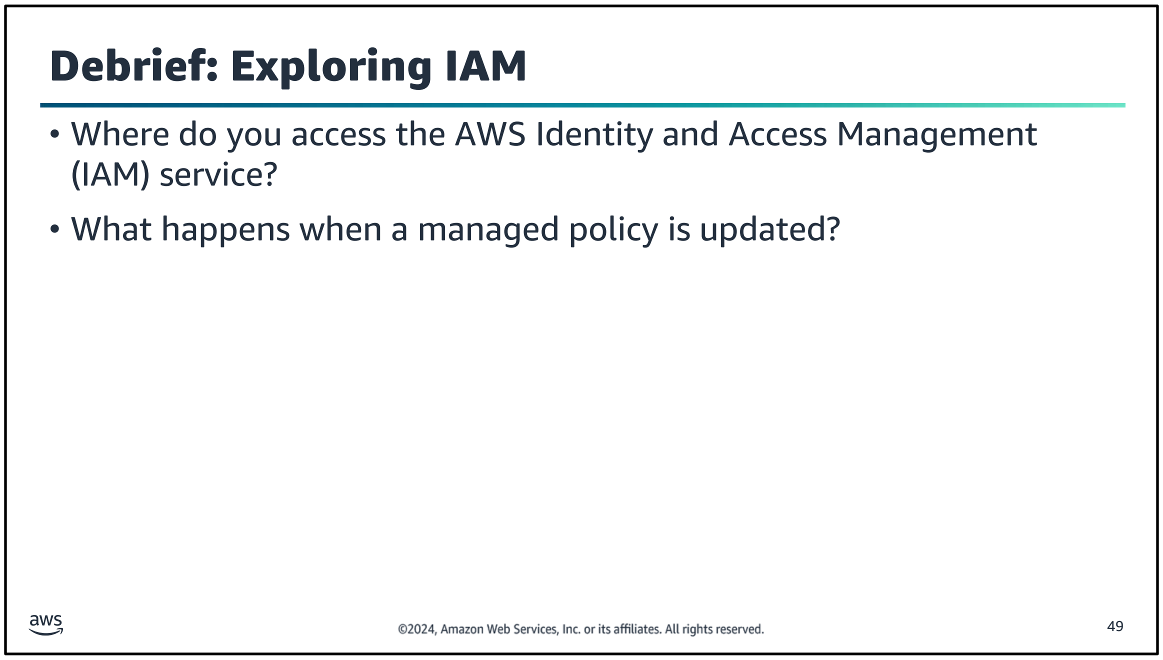


Exploring IAM lab tasks:

- In this lab, you perform the following main tasks:
  - Experiment with IAM policies
  - Add users to IAM groups
  - Use the IAM sign-in URL

48

Access the lab environment through your online course to get additional details and complete the lab.



Debrief: Exploring IAM

- Where do you access the AWS Identity and Access Management (IAM) service?
- What happens when a managed policy is updated?

©2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

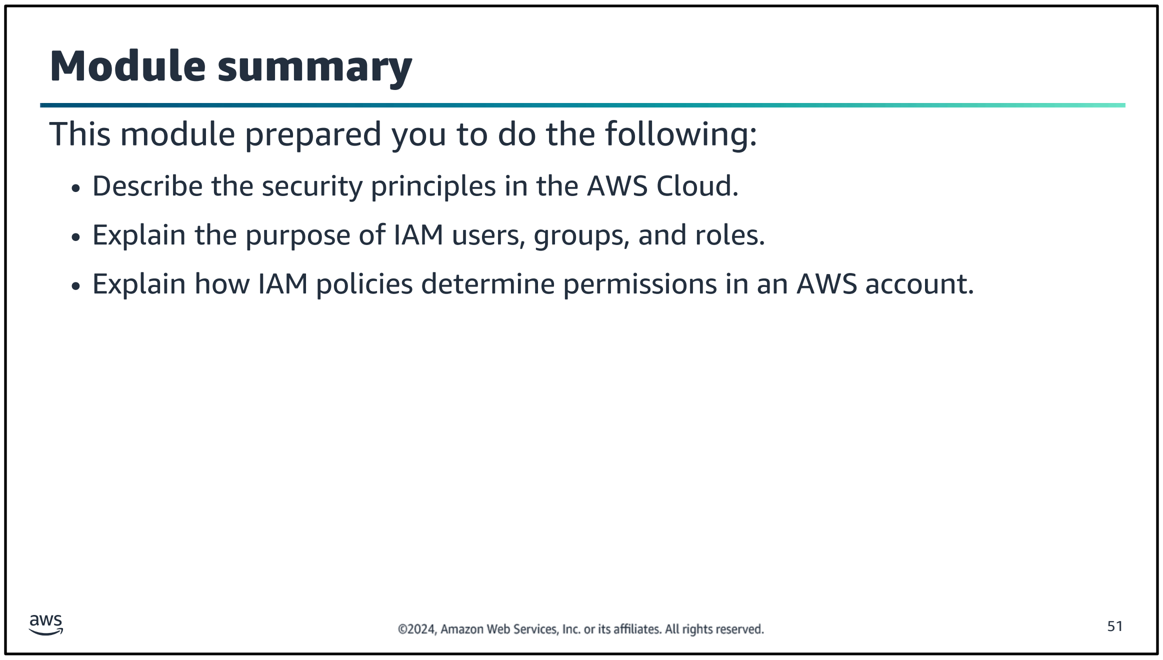
49

Your instructor might review these questions with you, or you might review them on your own. Use this opportunity to extend your thinking about the tasks that you performed during the lab.

Module wrap-up

Securing Access

This section summarizes what you have learned and brings the module to a close.



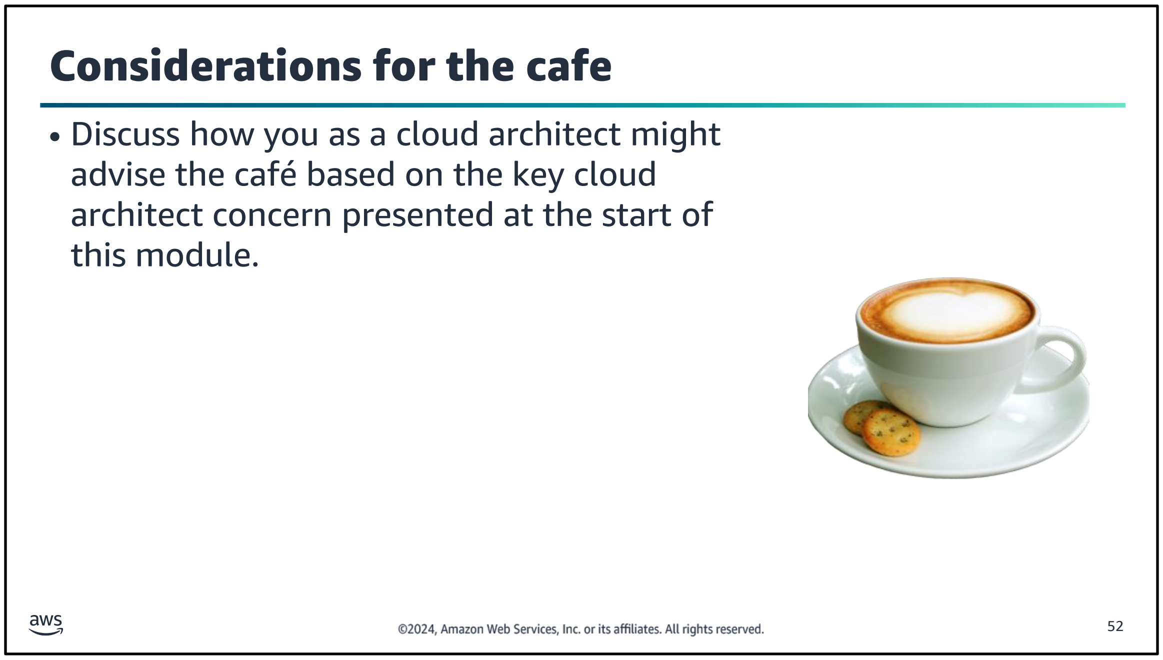
Module summary

This module prepared you to do the following:

- Describe the security principles in the AWS Cloud.
- Explain the purpose of IAM users, groups, and roles.
- Explain how IAM policies determine permissions in an AWS account.

51

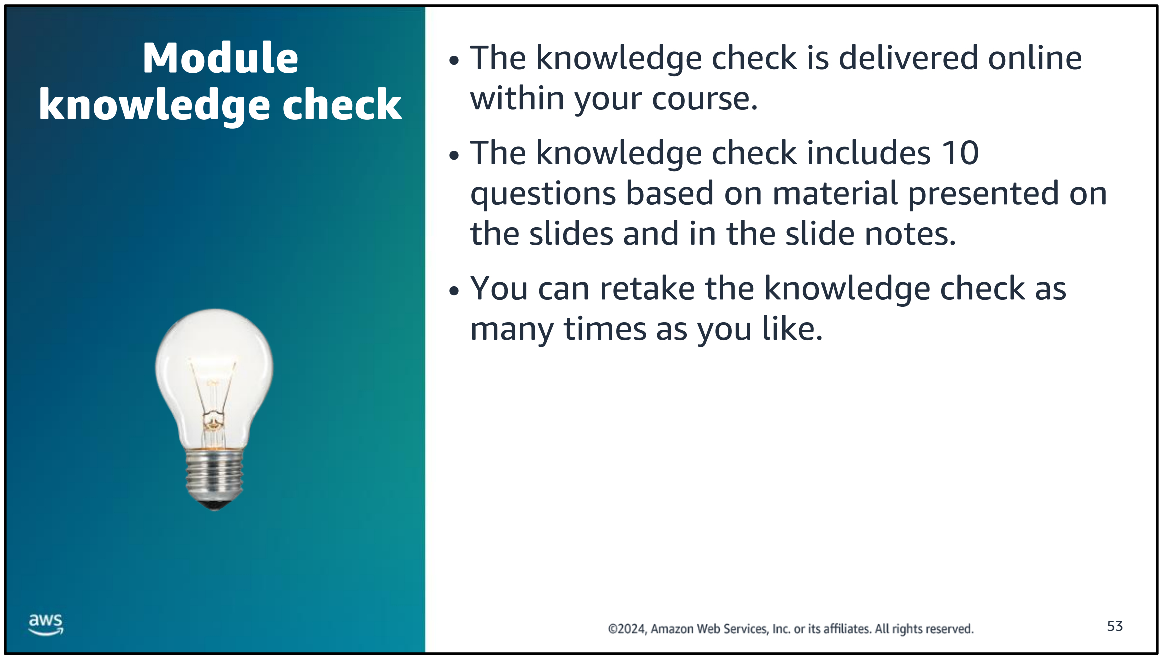
This module reviewed key security concepts that are applicable across all parts of your cloud solutions.



Considerations for the cafe

- Discuss how you as a cloud architect might advise the café based on the key cloud architect concern presented at the start of this module.

Use the cafe business as an example to apply what you've learned in this module to building a cloud architecture. What types of access considerations might be needed for the café staff?

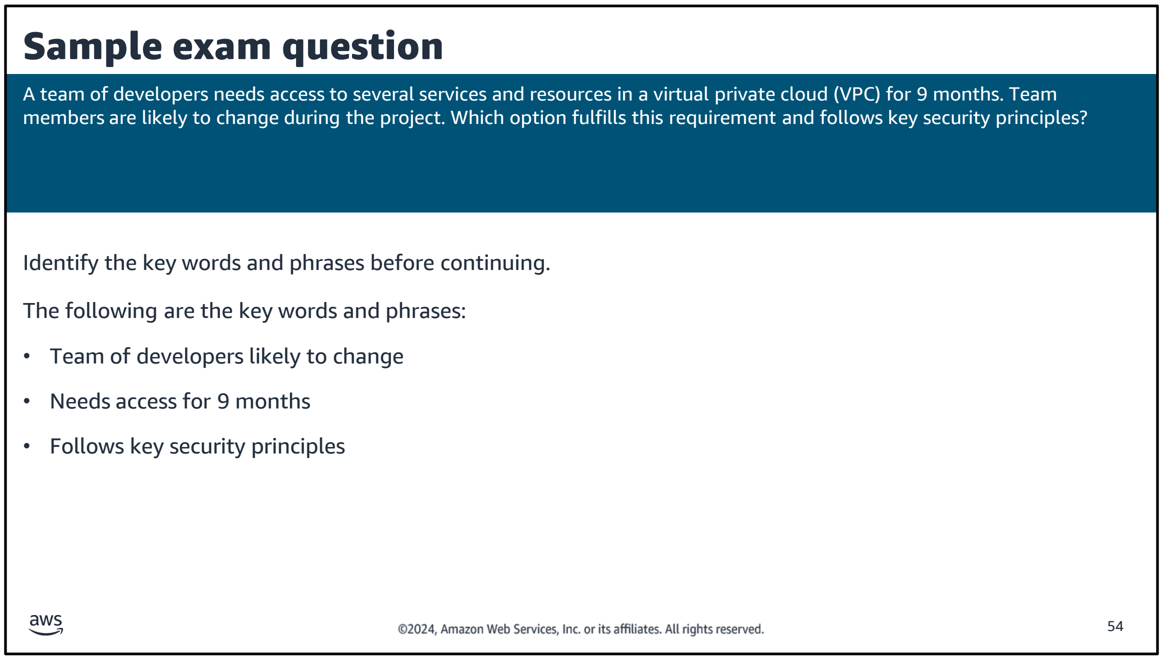


Module knowledge check

- The knowledge check is delivered online within your course.
- The knowledge check includes 10 questions based on material presented on the slides and in the slide notes.
- You can retake the knowledge check as many times as you like.

53

Use your online course to access the knowledge check for this module.



Sample exam question

A team of developers needs access to several services and resources in a virtual private cloud (VPC) for 9 months. Team members are likely to change during the project. Which option fulfills this requirement and follows key security principles?

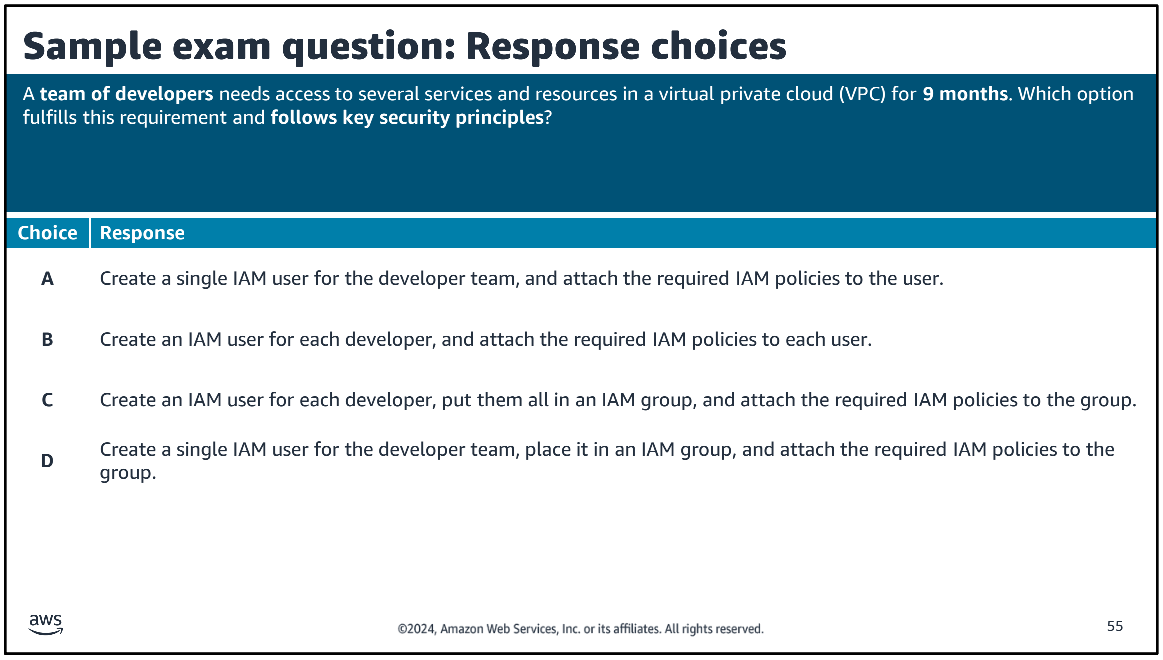
Identify the key words and phrases before continuing.

The following are the key words and phrases:

- Team of developers likely to change
- Needs access for 9 months
- Follows key security principles

©2024, Amazon Web Services, Inc. or its affiliates. All rights reserved. 54

The question identifies the need for a team of developers to access resources for a specific duration. The fact that they are a team of developers implies that the developers would need similar access to the resources. The specified duration suggests that these users won't require access after this period ends. The question also asks that you think in terms of key security principles

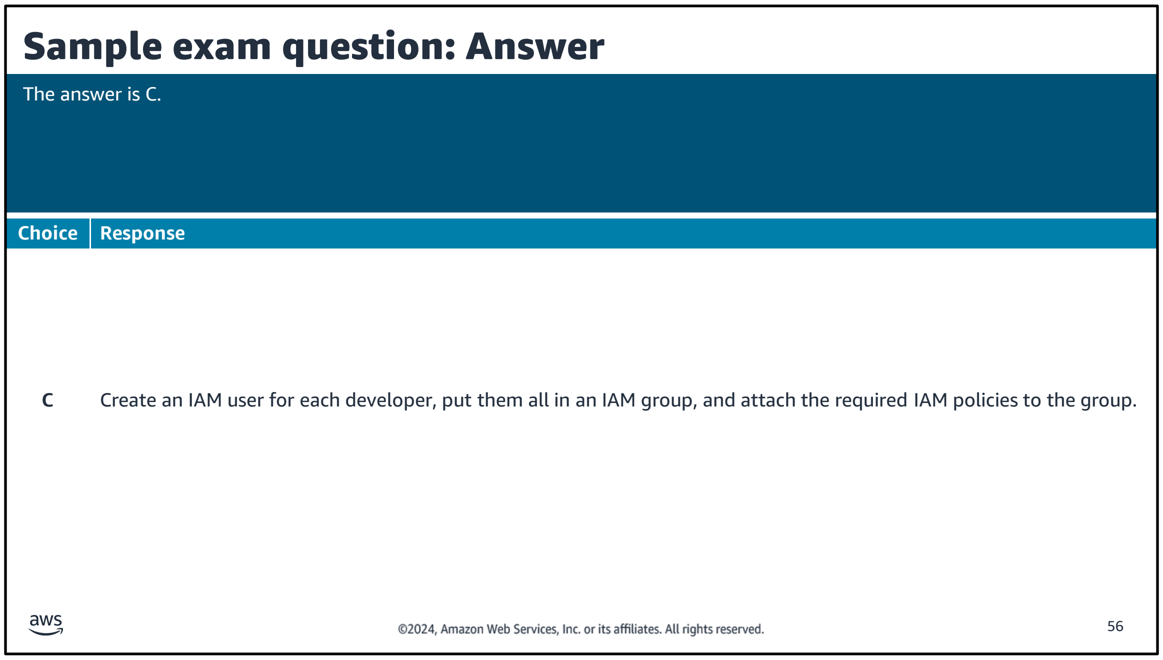


Sample exam question: Response choices

A **team of developers** needs access to several services and resources in a virtual private cloud (VPC) for **9 months**. Which option fulfills this requirement and **follows key security principles**?

Choice	Response
A	Create a single IAM user for the developer team, and attach the required IAM policies to the user.
B	Create an IAM user for each developer, and attach the required IAM policies to each user.
C	Create an IAM user for each developer, put them all in an IAM group, and attach the required IAM policies to the group.
D	Create a single IAM user for the developer team, place it in an IAM group, and attach the required IAM policies to the group.

Use the key words that you identified on the previous slide, and review each of the possible responses to determine which one best addresses the question.



Sample exam question: Answer

The answer is C.

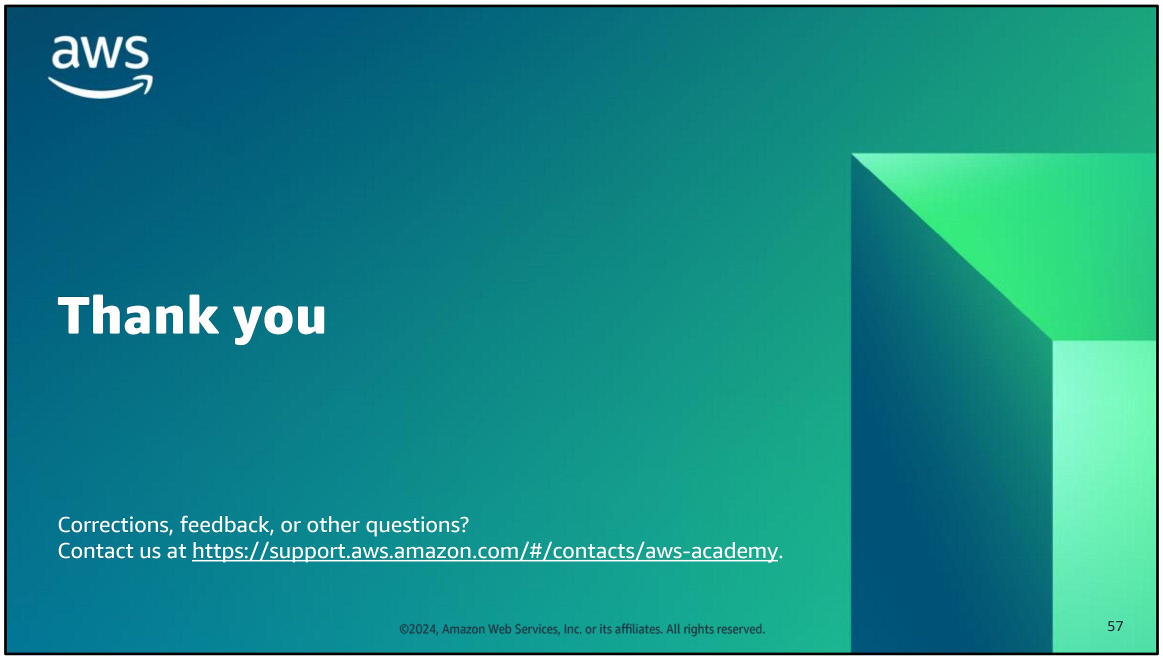
Choice	Response
C	Create an IAM user for each developer, put them all in an IAM group, and attach the required IAM policies to the group.

56

Choices A and D (creating a single shared IAM user for the team) shouldn't be implemented. Having multiple users share a single user account is generally a bad security practice. The password is shared by all users, and you lose individual accountability for actions that are performed by the user account.

Choice B (IAM users for each developer, and IAM policies attached to each user) will work but it is not the most efficient option for managing a team, especially when it is likely that developers will be added or removed during the project.

Choice C (putting users into a group, and attaching the policy to the group) is the best choice. Attaching a policy to a group applies the same access rules to all members of the group. This method also automatically applies the access rules to new users that are added to the group and removes those access rules from users that are removed from the group.



Thank you

Corrections, feedback, or other questions?
Contact us at https://support.aws.amazon.com/#/contacts/aws-academy.

57

That concludes this module. The Content Resources page of your course includes links to additional resources that are related to this module.