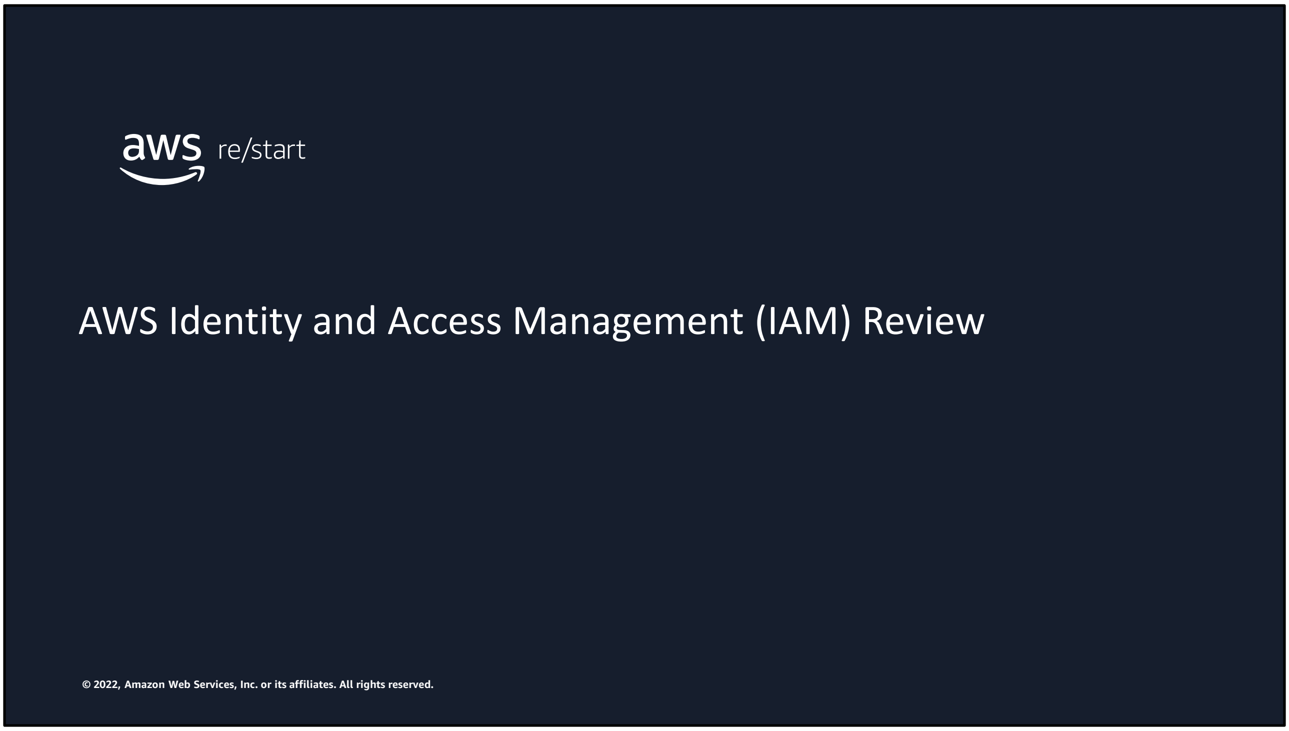
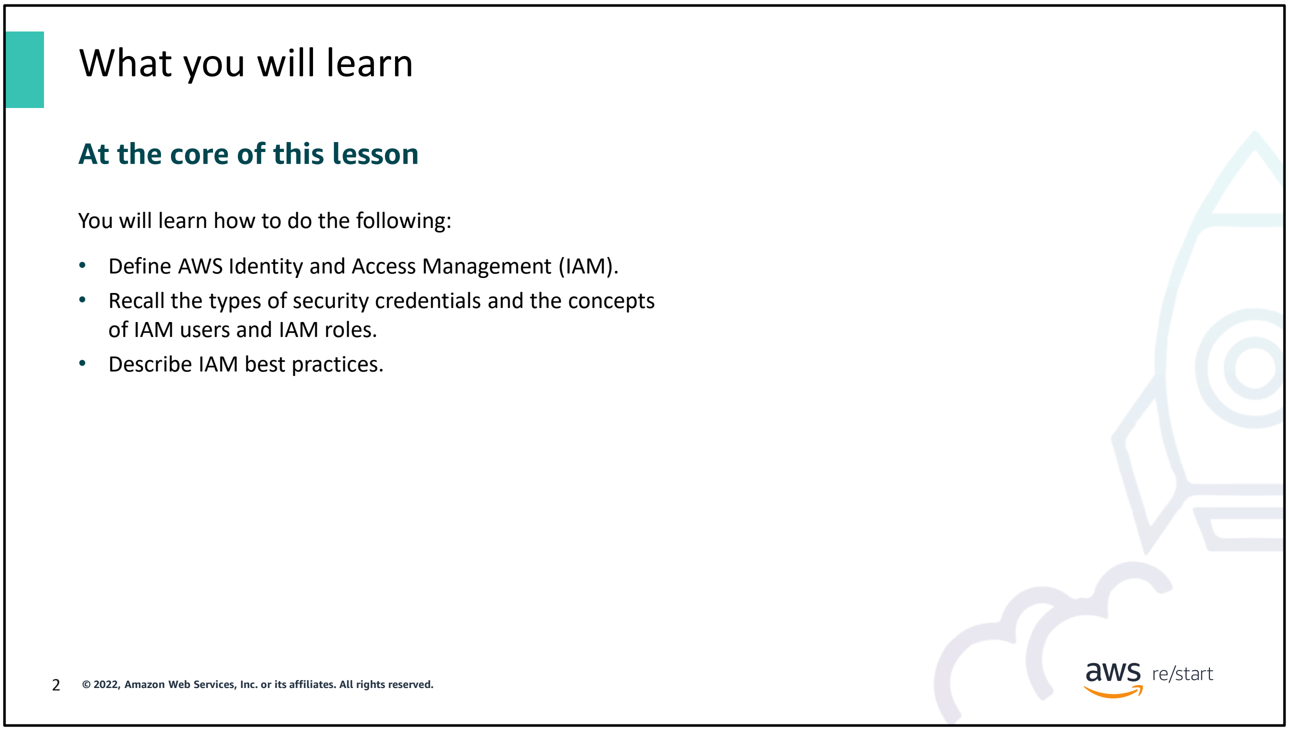


AWS Identity and Access Management (IAM) Review

Welcome to lesson titled AWS Identity and Access Management (IAM) Review.



What you will learn

At the core of this lesson

You will learn how to do the following:

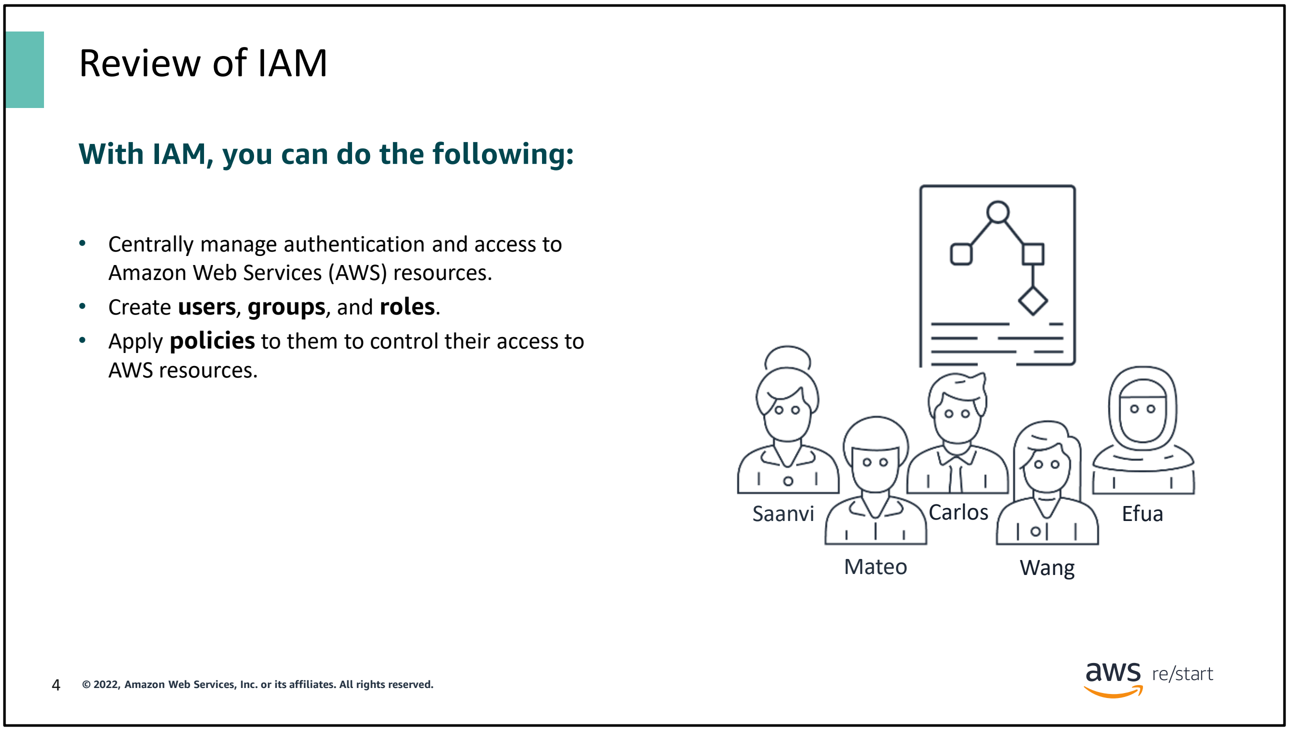
- Define AWS Identity and Access Management (IAM).
- Recall the types of security credentials and the concepts of IAM users and IAM roles.
- Describe IAM best practices.

At the end of this lesson, you will be able to do the following:

- Define AWS Identity and Access Management (IAM).
- Recall the types of security credentials and the concepts of IAM users and IAM roles.
- Describe IAM best practices.

IAM

Now, you'll review IAM, which you discussed earlier.



Review of IAM

With IAM, you can do the following:

- Centrally manage authentication and access to Amazon Web Services (AWS) resources.
- Create **users**, **groups**, and **roles**.
- Apply **policies** to them to control their access to AWS resources.

Saanvi
Carlos
Efua
Mateo
Wang

IAM is a service that helps securely control access to AWS resources. You can manage the resources that can be accessed and which actions can be performed. For example, who can terminate Amazon Elastic Compute Cloud (Amazon EC2) instances?

You can also define required credentials based on context, such as the following:
- Who can access which AWS services?
- What is the user or system allowed to do with the service?

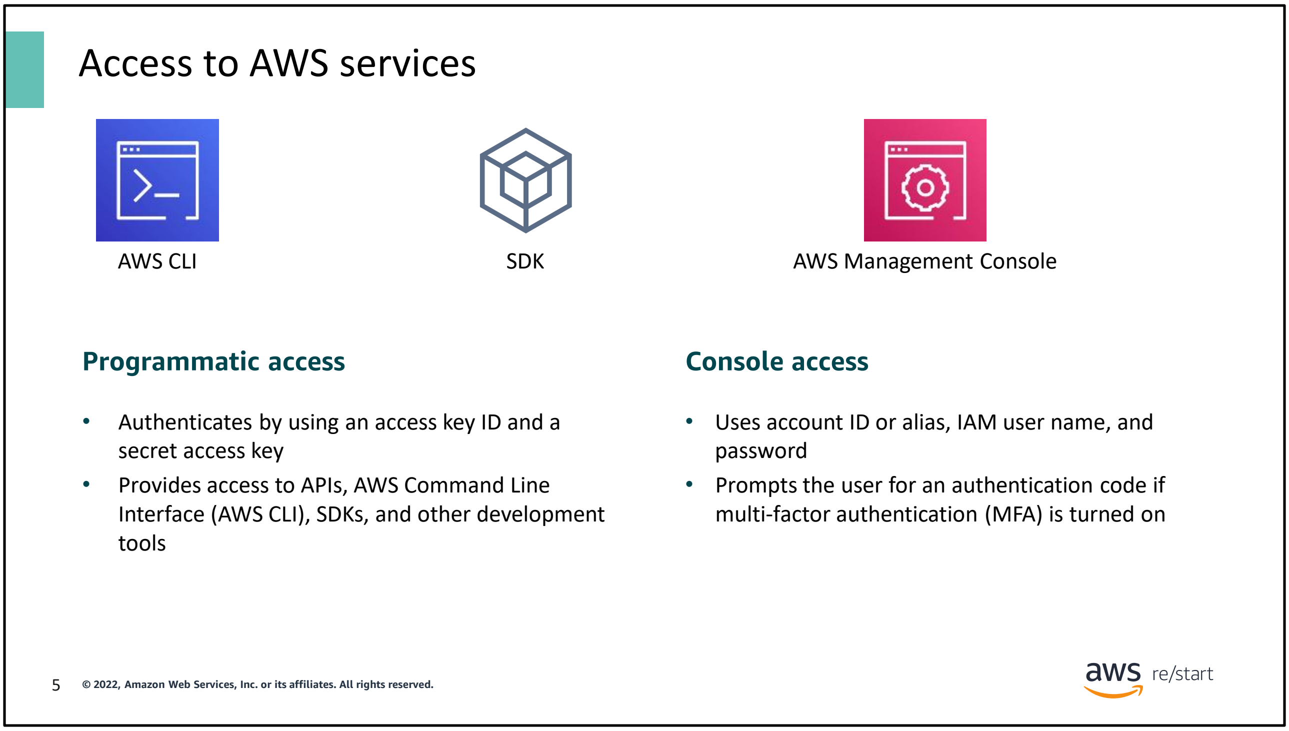
Use IAM to configure authentication, which is the first step, because it controls who can access AWS resources. IAM can also be used to authenticate resources. For example, applications and other AWS services use it for access.

IAM is used to configure authorization, which is based on knowing who the user is. Authorization controls which resources users can access and what they can do to or with those resources. IAM reduces the need to share passwords or access keys when you grant access rights. It also makes it easy to turn on or turn off a user’s access over time and as appropriate.

Think of IAM as the service that you can use to centrally manage who or what can launch, configure, manage, and remove resources.

A principal is a person or application that can make a request for an action or operation on an AWS resource.

IAM is offered as a feature of an AWS account at no charge.



Access to AWS services

AWS CLI

SDK

AWS Management Console

Programmatic access

- Authenticates by using an access key ID and a secret access key
- Provides access to APIs, AWS Command Line Interface (AWS CLI), SDKs, and other development tools

Console access

- Uses account ID or alias, IAM user name, and password
- Prompts the user for an authentication code if multi-factor authentication (MFA) is turned on

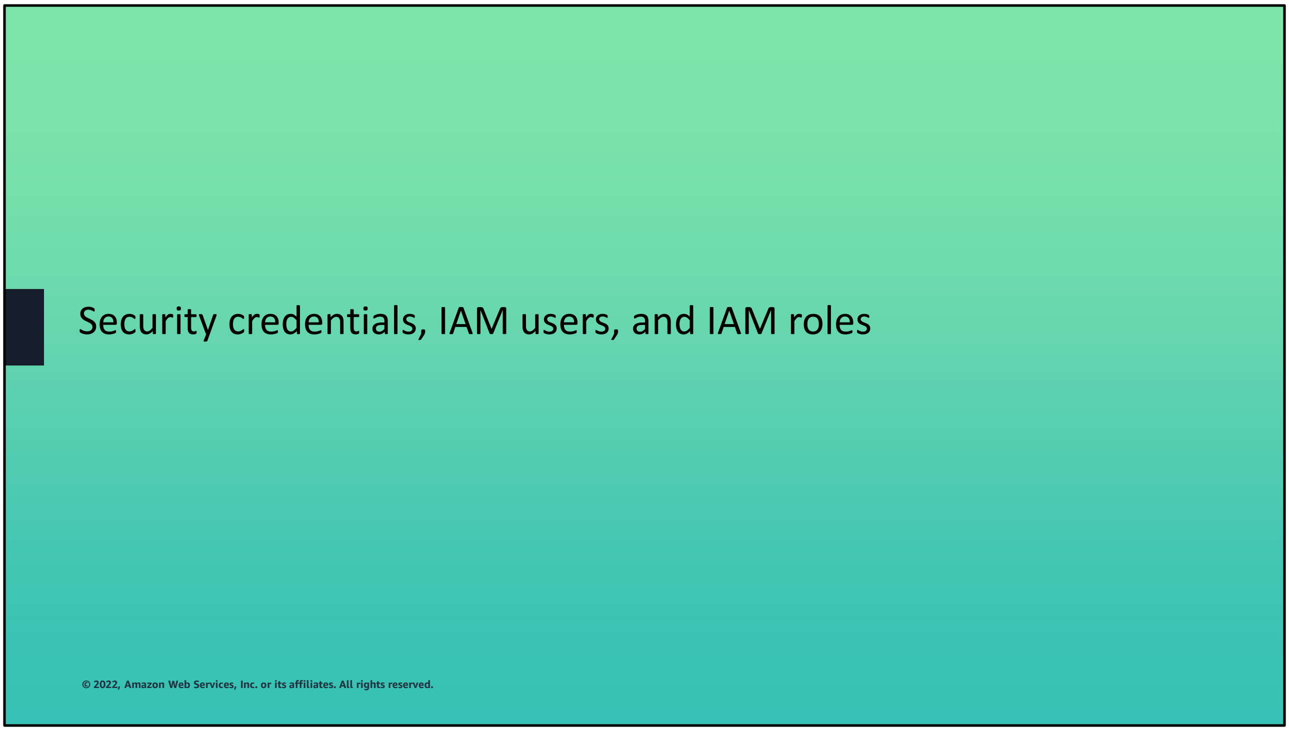
aws re/start

To provide programmatic access, create an access key ID and a secret access key, and offer them to the user. With these two items, the user will have access to the following:

- AWS Command Line Interface (AWS CLI)
- AWS APIs (directly or by using the AWS SDK)

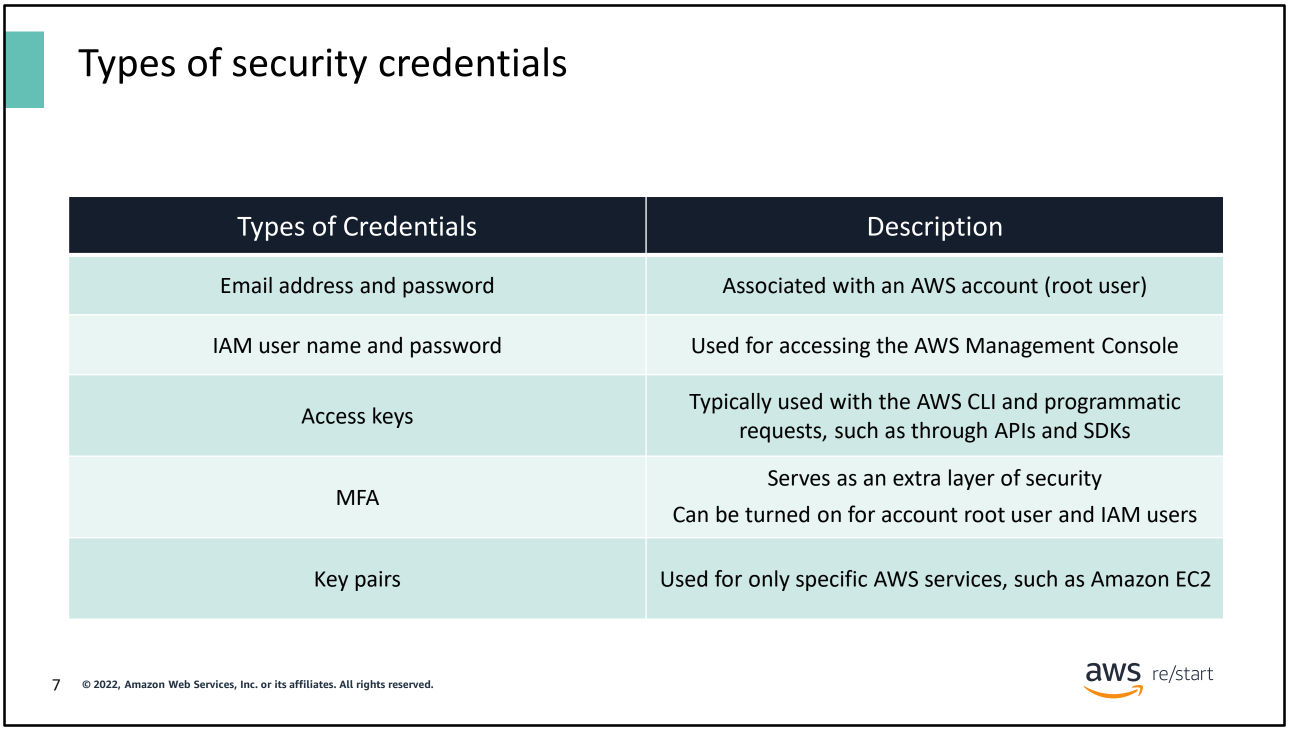
To provide AWS Management Console access, assign a user name and password to the user. The AWS Management Console provides a web interface for AWS.

For added security, AWS recommends that IAM user accounts have multi-factor authentication (MFA) turned on. If MFA is turned on, the user will be prompted for an additional authentication code after they provide their user name and password.



Security credentials, IAM users, and IAM roles

Next, you dive deeper into IAM.



Types of security credentials

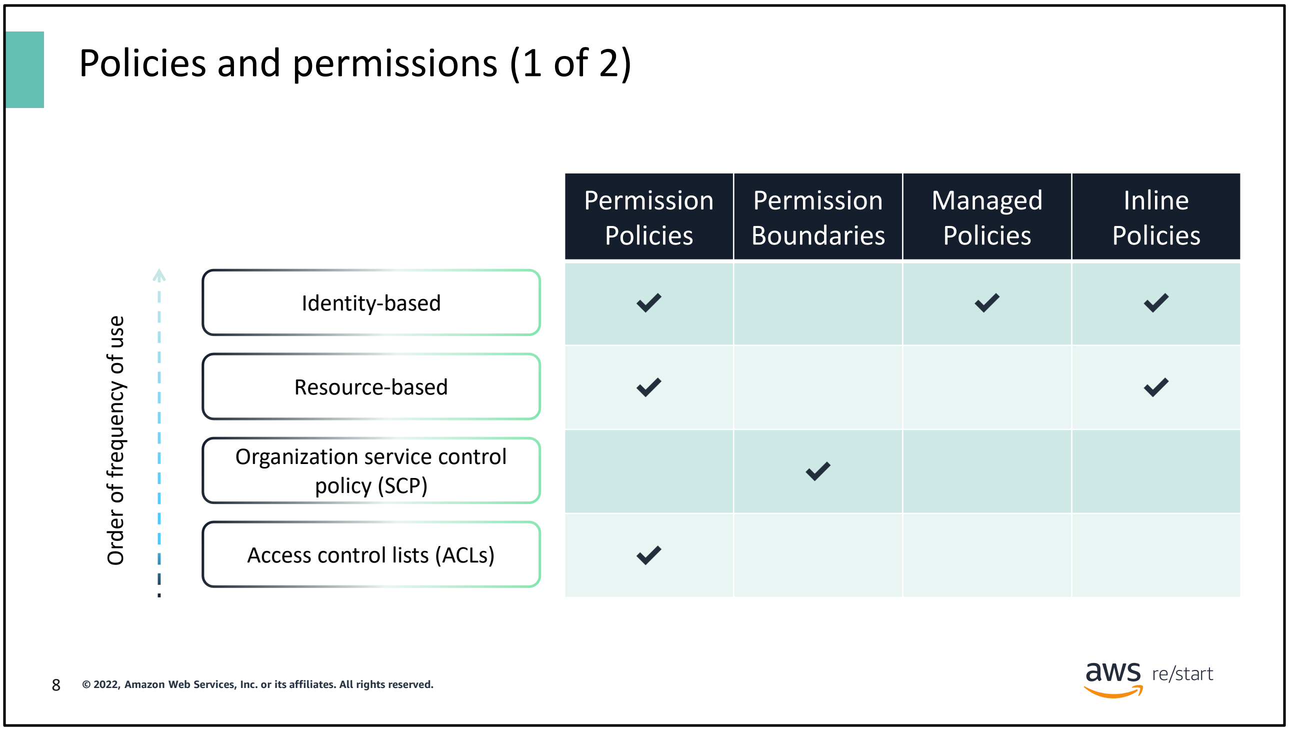
Types of Credentials	Description
Email address and password	Associated with an AWS account (root user)
IAM user name and password	Used for accessing the AWS Management Console
Access keys	Typically used with the AWS CLI and programmatic requests, such as through APIs and SDKs
MFA	Serves as an extra layer of security Can be turned on for account root user and IAM users
Key pairs	Used for only specific AWS services, such as Amazon EC2

This table summarizes the different types of security credentials.

Each AWS account has an assigned root user through the user account. The account root user has an assigned email address for the purposes of account recovery and communication. However, AWS recommends not using the root user for everyday tasks, even administrative tasks. Instead, follow the best practice of using the root user only to create an IAM user first. Then, securely lock away the root user credentials. Use these credentials only when you must perform the few account and service management tasks that you cannot accomplish in other ways.

As mentioned in the previous slide, the IAM user name and password security credentials are used to access the AWS Management Console, also called the console. Access keys can be used for programmatic access when they are generated for a user.

Finally, for added security, AWS recommends that you apply MFA on the account root user and on any defined IAM users.



Policies and permissions (1 of 2)

Order of frequency of use

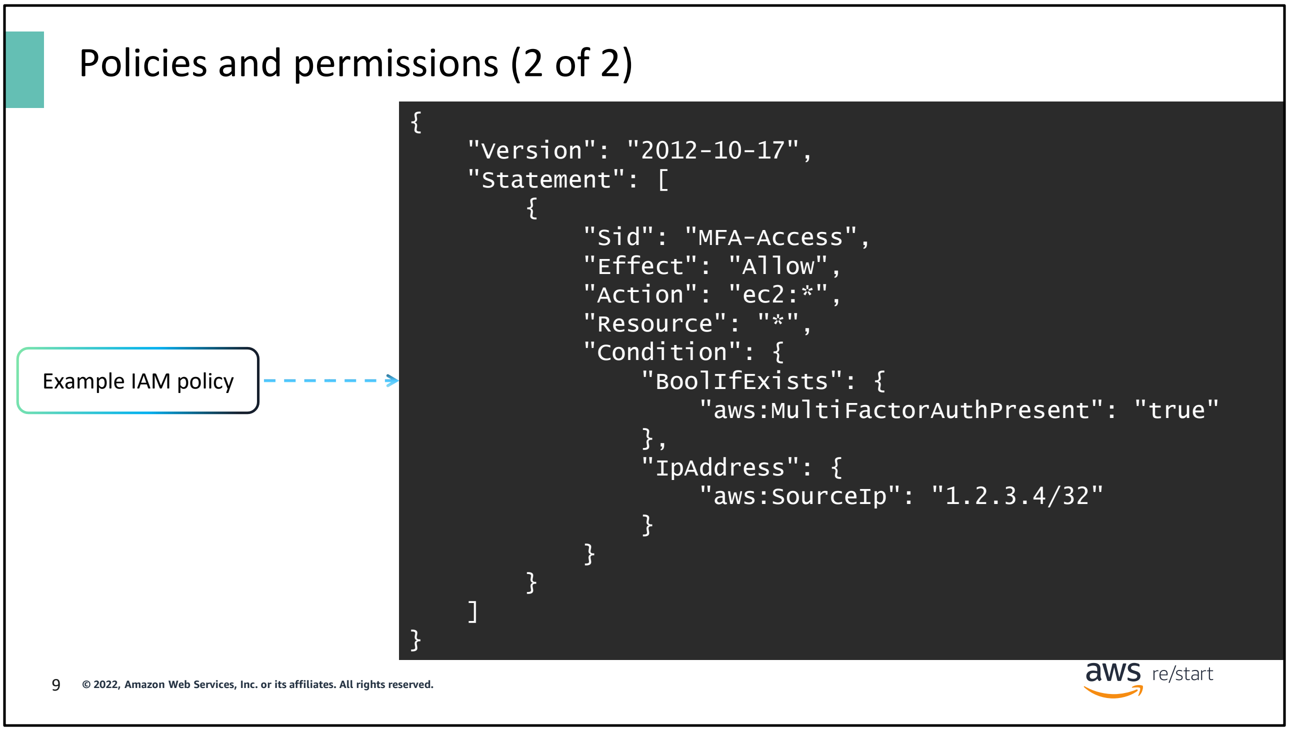
	Permission Policies	Permission Boundaries	Managed Policies	Inline Policies
Identity-based	✓		✓	✓
Resource-based	✓			✓
Organization service control policy (SCP)		✓		
Access control lists (ACLs)	✓			

aws re/start

Policy types indicate the type of permissions that are defined for users or resources. The following are the four policy types in order of popular usage:

1. **Identity-based policies** allow a user to attach managed and inline policies to IAM identities, such as users or the groups that users belong to. A user can also attach identity-based policies to roles. Identity-based policies are defined and stored as JSON documents.
2. **Resource-based policies** allow a user to attach inline policies to resources. The most common examples of resource-based policies are Amazon Simple Storage Service (Amazon S3) bucket policies and IAM role trust policies. Resource-based policies are JSON policy documents.
    - Resource-level permissions are available for only some AWS services and resources. They provide granular access control over specific objects within an AWS service. For example, a resource-based policy can list specific EC2 instances and specific Amazon Elastic Block Store (Amazon EBS) volumes.
    - Resource-level permissions do not always allow all actions. For example, for EC2 instances, actions such as *Reboot*, *Start*, *Stop*, and *Terminate* can be specified. However, actions such as *RunInstances* cannot be specified because you do not know the InstanceID before the RunInstances call is complete. Therefore, RunInstances applies to Amazon EC2 as a whole service but not to a specific resource.
3. **AWS Organizations service control policies (SCPs)** apply permissions boundaries to AWS Organizations, organizational units (OUs), or accounts. SCPs use the JSON format.

4. **Access control lists (ACLs)** can also be used to control which principals (that is, users or resources) can access a resource. ACLs are similar to resource-based policies although they are the only policy type that does not use the JSON policy document structure.



Policies and permissions (2 of 2)

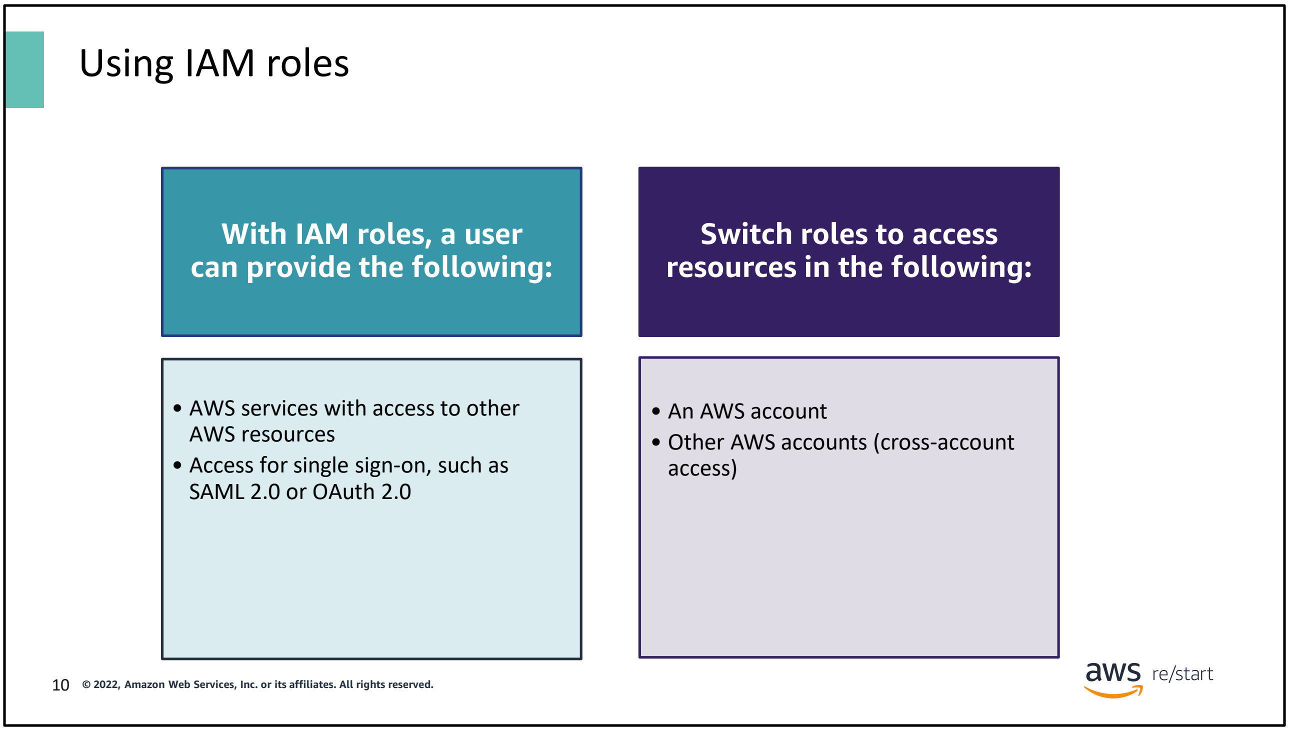
Example IAM policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "MFA-Access",
            "Effect": "Allow",
            "Action": "ec2:*",
            "Resource": "*",
            "Condition": {
                "BoolIfExists": {
                    "aws:MultiFactorAuthPresent": "true"
                },
                "IpAddress": {
                    "aws:SourceIp": "1.2.3.4/32"
                }
            }
        }
    ]
}
```

Remember that the policy evaluation logic involves the following steps:

1. Authenticate the principal that made the request.
2. Process the request context to determine which policies apply to the request.
3. Evaluate the applicable policies in order based on policy type.
4. Determine whether the request is allowed.

The evaluation order of the policies has no effect on outcome. For example, if a user has an explicit DENY in one of their policies, that DENY overrides any ALLOW.



Using IAM roles

With IAM roles, a user can provide the following:

- AWS services with access to other AWS resources
- Access for single sign-on, such as SAML 2.0 or OAuth 2.0

Switch roles to access resources in the following:

- An AWS account
- Other AWS accounts (cross-account access)

There are three ways to use a role:

- Interactively in the **IAM** section of the AWS Management Console
- Programmatically with the AWS CLI
- Through the AWS SDKs (API calls)

An application or a service such as Amazon EC2 can assume a role. It does so by requesting temporary security credentials for a role that the service can use to make programmatic requests to AWS.

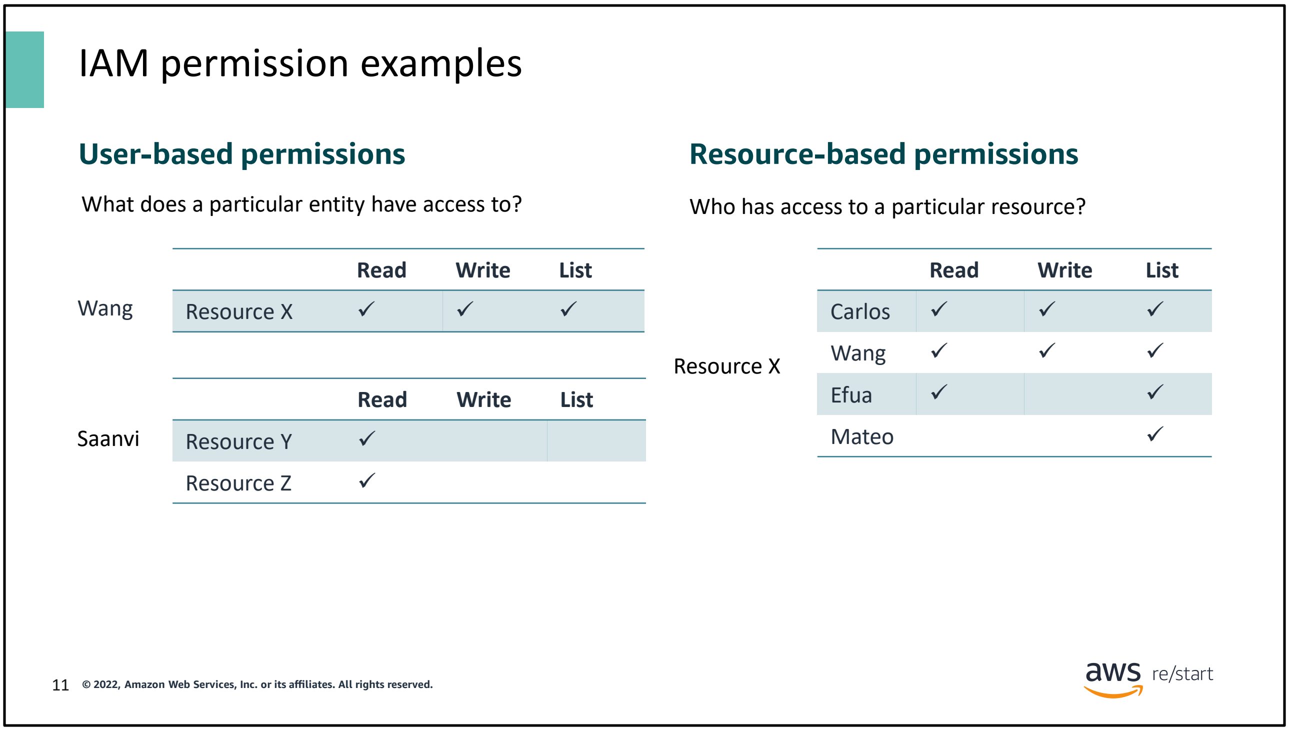
IAM roles also support single sign-on (SSO) solutions. For example, an IAM administrator could configure SAML 2.0 federation instead of creating IAM users in an AWS account. With an identity provider, user identities can be managed outside AWS. These external user identities could be granted permissions to access resources in the AWS account.

For more information, see the following resources in the *AWS Identity and Access Management User Guide*:

- "IAM roles" at https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html
- "How IAM roles differ from resource-based policies" at https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_compare-resource-

policies.html

- “Policies and permissions in IAM” at https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html



IAM permission examples

User-based permissions

What does a particular entity have access to?

Wang

	Read	Write	List
Resource X	✓	✓	✓

Saanvi

	Read	Write	List
Resource Y	✓		
Resource Z	✓		

Resource-based permissions

Who has access to a particular resource?

Resource X

	Read	Write	List
Carlos	✓	✓	✓
Wang	✓	✓	✓
Efua	✓		✓
Mateo			✓

A role defines permissions to resources. When the role is assigned to a user, the user gains the permissions defined in the role. Roles are intended to grant identity-based (or user-based) permissions.

The following information describes the user-based permissions example on this slide:

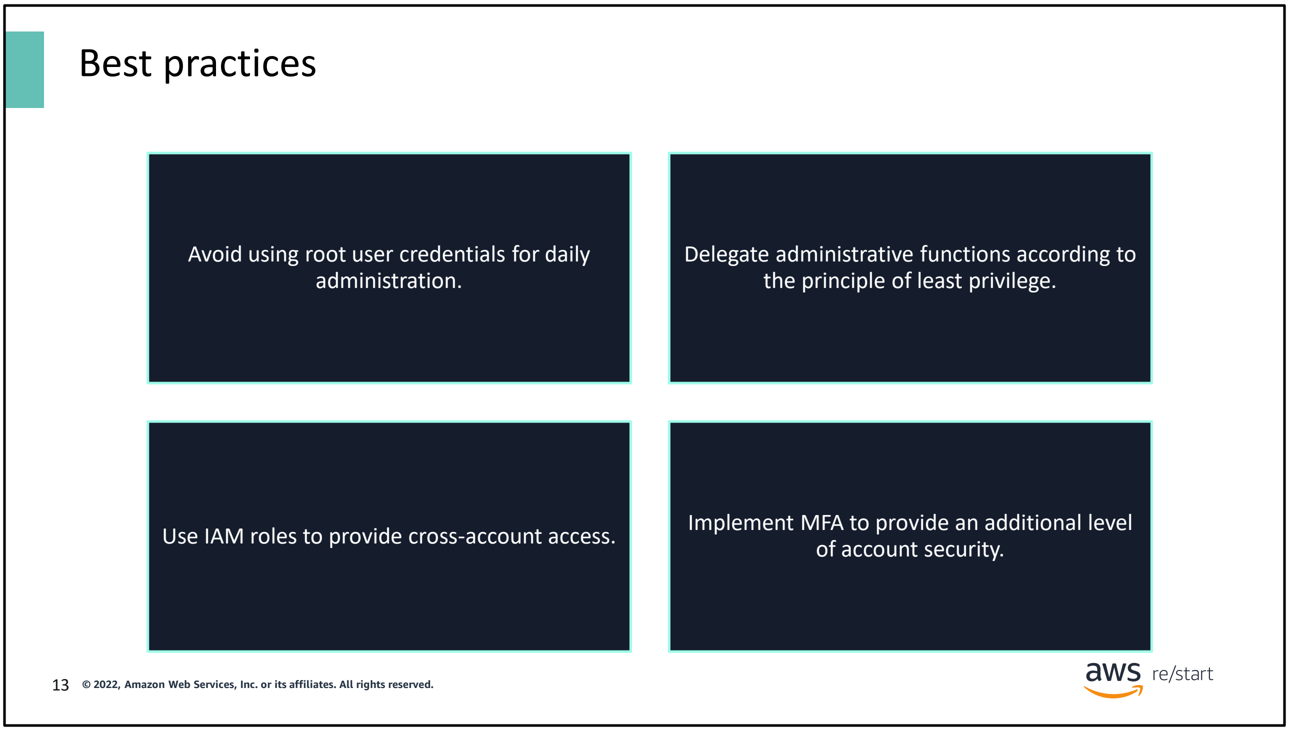
- The user Xiulan is assigned a role that gives Read, Write, and List access to Resource X.
- The user Saanvi is assigned a role that gives Read access to Resource Y and Resource Z.

You can also create permissions that are resource-based. These types of permissions are attached to a resource and specify who has access to a resource and what actions they can perform on it.

In the resource-based permissions example in this slide, Resource X can be accessed only by Carlos, Wang, Efua, and Mateo. Carlos and Wang have Read, Write, and List access. Efua has Read and List access. Mateo only has List access.

IAM best practices

Next, you will learn about IAM best practices.



Best practices

Avoid using root user credentials for daily administration.

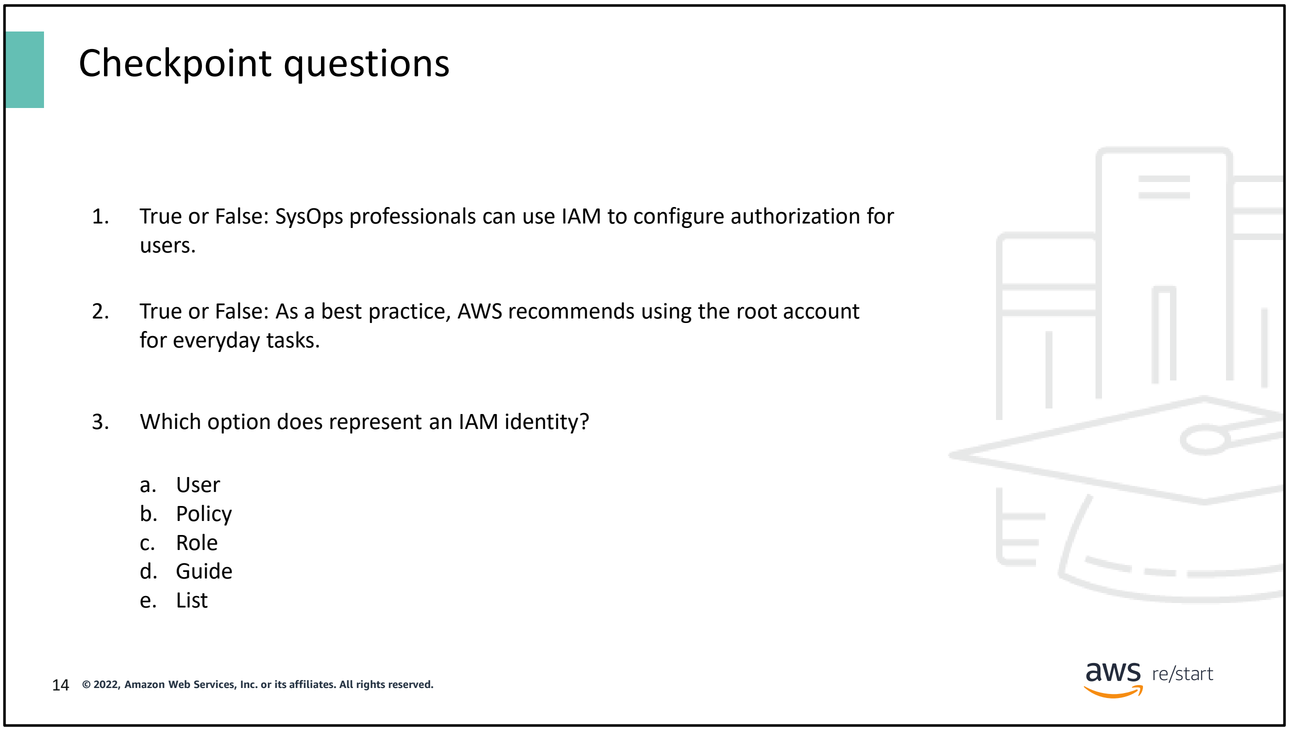
Delegate administrative functions according to the principle of least privilege.

Use IAM roles to provide cross-account access.

Implement MFA to provide an additional level of account security.

Best practices for IAM include the following:

1. Avoid using the account root user credentials for daily administration. Instead, when you set up a new AWS account, define at least one new IAM user. Then, grant the user or users access so that they can do most daily tasks by using these IAM user credentials.
2. Delegate administrative functions by following the principle of least privilege. Grant access to only services that are needed, and limit permissions within those services to only the parts that are needed. You can always grant additional rights over time if the need arises.
3. Use IAM roles to provide cross-account access. Other best practices for IAM mentioned earlier in this lesson include configuring strong password policies, turning on MFA for any privileged users, and rotating credentials regularly. For more information about best practices for IAM, see “Security best practices in IAM” at https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html.
4. MFA is a good practice to implement for security purposes.



Checkpoint questions

1. True or False: SysOps professionals can use IAM to configure authorization for users.

2. True or False: As a best practice, AWS recommends using the root account for everyday tasks.

3. Which option does represent an IAM identity?

   a. User
   b. Policy
   c. Role
   d. Guide
   e. List

The following are answers to the checkpoint questions:

1. True or False: IAM can be used to configure authorization for users.

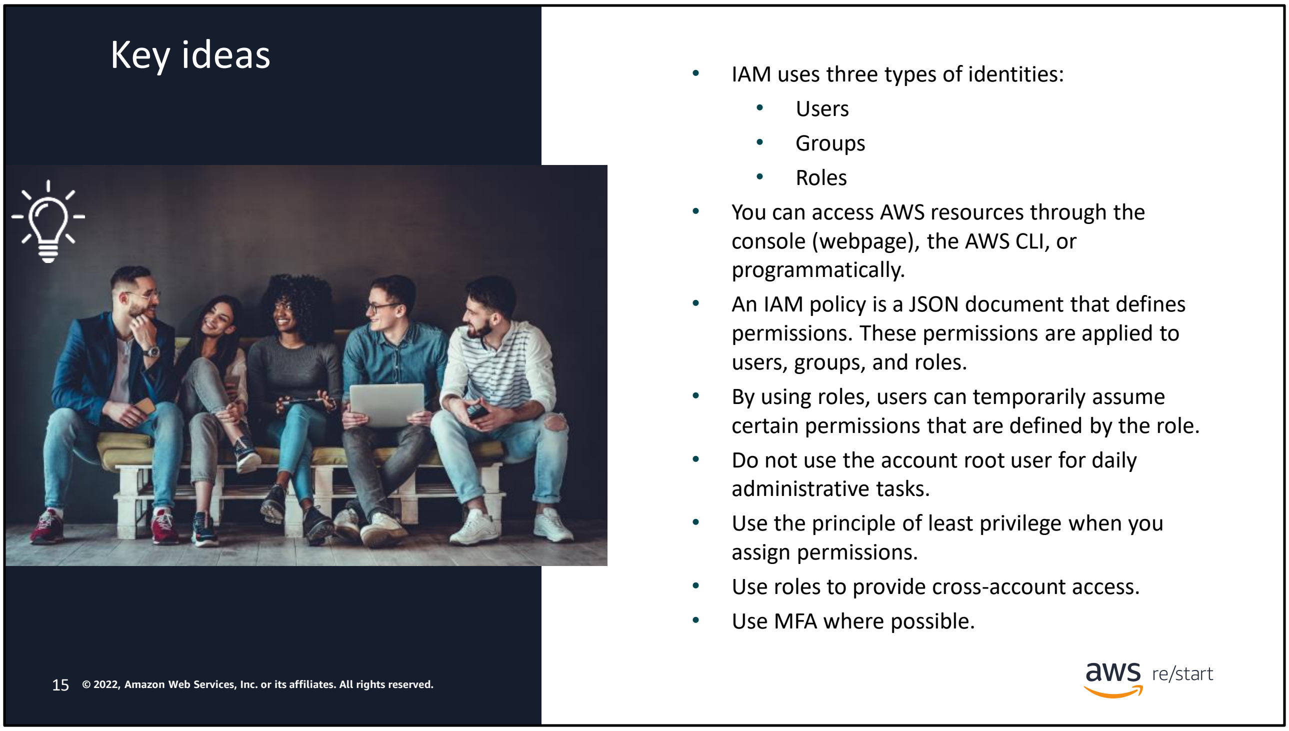
   True

2. True or False: As a best practice, AWS recommends using the root account for everyday tasks.

   False

3. Which option does not represent an IAM identity?

   Policy



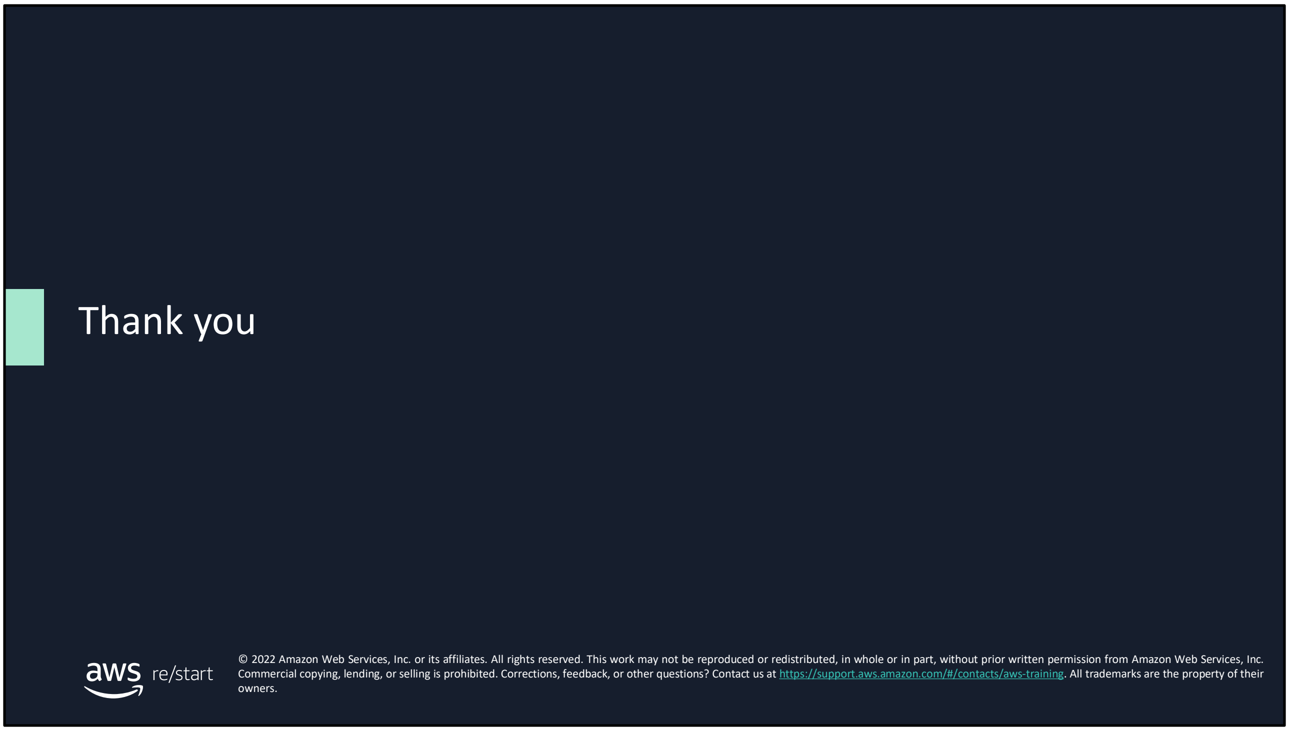
Key ideas

- IAM uses three types of identities:
  - Users
  - Groups
  - Roles
- You can access AWS resources through the console (webpage), the AWS CLI, or programmatically.
- An IAM policy is a JSON document that defines permissions. These permissions are applied to users, groups, and roles.
- By using roles, users can temporarily assume certain permissions that are defined by the role.
- Do not use the account root user for daily administrative tasks.
- Use the principle of least privilege when you assign permissions.
- Use roles to provide cross-account access.
- Use MFA where possible.

aws re/start

This lesson includes the following key takeaways:

- IAM uses three types of identities:
  - Users
  - Groups
  - Roles
- You can access AWS resources through the console (webpage), the AWS CLI, or programmatically.
- An IAM policy is a JSON document that defines permissions. These permissions are applied to users, groups, and roles.
- By using roles, users can temporarily assume certain permissions that are defined by the role.
- Do not use the account root user for daily administrative tasks.
- Use the principle of least privilege when you assign permissions.
- Use roles to provide cross-account access.
- Use MFA where possible.



Thank you

aws re/start

Thank you for completing this lesson.