**Module 6 introduction**

**Learning objectives**

In this module, you will learn how to:

* Explain the benefits of the shared responsibility model.
* Describe multi-factor authentication (MFA).
* Differentiate between the AWS Identity and Access Management (IAM) security levels.
* Explain the main benefits of AWS Organizations.
* Describe security policies at a basic level.
* Summarize the benefits of compliance with AWS.
* Explain additional AWS security services at a basic level.

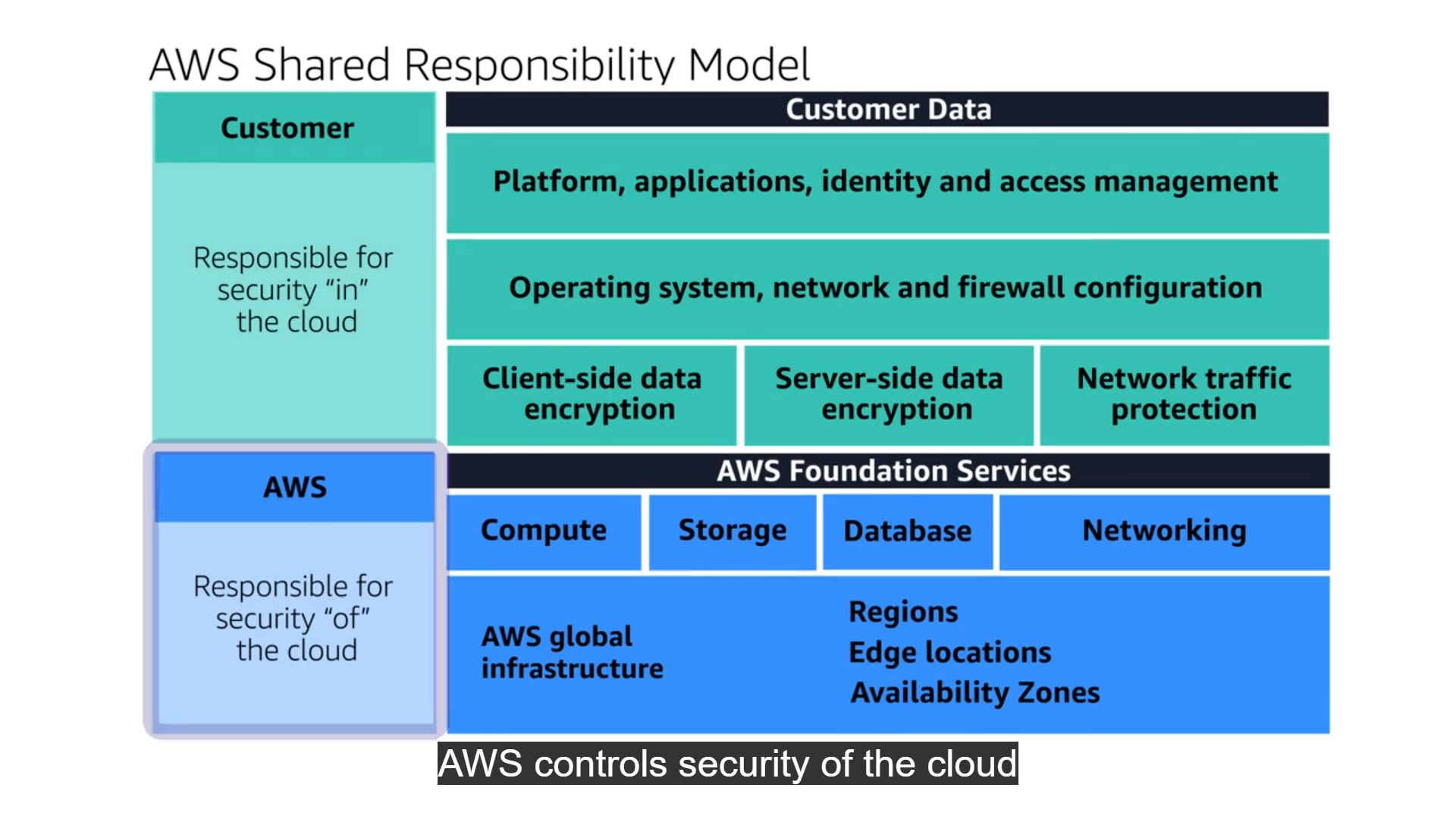
**Video transcript**

–

Looks like we're getting deeper into our AWS account. Things are running well. And I want to let you in on the security measures we have in place. By this, I mean, we have to describe the various security mechanisms we offer on the AWS Cloud, like the shared responsibility model.

With the shared responsibility model, AWS controls security of the cloud and customers control security in the cloud. We, as AWS, control the data centers, security of our services, and all the layers in this section. The next part are the workloads that AWS customers run in the cloud and those are the customer's responsibility to secure. It's something we share with customers to ensure security in the cloud.

Let's take a look at the various other security services, mechanisms, and features that AWS has to offer in this module. Stay tuned.



# Shared responsibility model

**Video transcript**

–

When it comes to securing your business on AWS, it's important to ask the question: Who is ultimately responsible for the security? Is it A: You, the customer? or B: AWS? And the correct answer is: yes. Both. Both are ultimately responsible for making sure that you are secure.

Now, if there's any security experts watching this right now you're probably shaking your head saying you can't have two different entities with the ultimate responsibility over a single object. That's not security, that's wishful thinking. At AWS, we agree completely. But for us, we don't look at your environment as a single object. Instead we see it as a collection of parts that build on each other. AWS is responsible for the security of some of the objects. Responsible 100% for those. The others, you are responsible 100% for their security. This is what's known as the shared responsibility model.

It's no different than securing a house. The builder constructed the house with four walls and a door. It's their responsibility to make sure the walls are strong and the doors are solid. It's your responsibility, the homeowner, to close and lock the doors.

It really is that simple on AWS as well. Take EC2, for instance. EC2 lives in a physical building, a data center that must be secured. It has a network and a hypervisor that supports your instances and their individual operating systems. On top of that operating system, you have your application, and that supports your data. So for EC2 and every service AWS offers, there's a similar stack of parts that build on top of each other. AWS is 100% responsible for some, you are responsible for the others.

So, starting with the physical layer. This is iron and concrete and fences and security guards. Someone has to own the concrete. Someone has to staff the physical perimeter, 24/7. This is AWS. On top of the physical layer we have our network and our hypervisor. Now, I'm not gonna go into details on how this is all secured, but basically we have reinvented those technologies to make them faster, stronger, tamper-proof.

But you don't have to just take our word for it. We have numerous third party auditors who have gone through the code and the way we build our infrastructure, and can provide the right documentation you need for your security compliance structures. Now, on top of all that, on EC2, you now get to pick what operating system you want to run.

This is the magic dividing line that separates our responsibility. AWS' responsibility and your responsibility. This is your operating system. You're 100% in charge of this. AWS does not have any backdoor into your system here. You and you alone have the only encryption key to log onto the root of this OS or to create any user accounts there. I mean, no more than a construction company would keep copies of your front door key, AWS cannot enter your operating system. And here's a hint. If someone from AWS calls and asks you for your OS key, it is not AWS.

Now that means your operations team is 100% responsible for keeping the operating system patched. If AWS discovers there are some new vulnerabilities in your version of Windows, let's say, we can certainly notify your account owner but we cannot deploy a patch. This is a really good thing for your security. This means no one can deploy anything that might break your system without your team being the ones that do it. Now, on top of that OS, you can run whatever applications you want. You own them. You maintain them.

Which takes us to the most important part of the stack, your data. Data. This is always your domain to control. And sometimes you might want to have your data open for everyone to see, like pictures on a retail website. Other times like banking or healthcare, yeah, not so much, not so much. AWS provides everyone with the tool set they need for their data to open it up to some authorized individuals, to everyone, to just a single person under specific conditions, or even lock it down so no one can access it. Plus, the ability to have ubiquitous encryption. That way even if you accidentally left your front door open, all anyone would see is unreadable encrypted content.

The AWS shared responsibility model is about making sure both sides understand exactly what tasks are ours. Basically, AWS is responsible for the security of the cloud and you are responsible for the security in the cloud. Together, you have an environment you can trust.

**The AWS shared responsibility model**

Throughout this course, you have learned about a variety of resources that you can create in the AWS Cloud. These resources include Amazon EC2 instances, Amazon S3 buckets, and Amazon RDS databases. Who is responsible for keeping these resources secure: you (the customer) or AWS?

The answer is both. The reason is that you do not treat your AWS environment as a single object. Rather, you treat the environment as a collection of parts that build upon each other. AWS is responsible for some parts of your environment and you (the customer) are responsible for other parts. This concept is known as the [**shared responsibility model**](https://aws.amazon.com/compliance/shared-responsibility-model/).

The shared responsibility model divides into customer responsibilities (commonly referred to as “security in the cloud”) and AWS responsibilities (commonly referred to as “security of the cloud”).

You can think of this model as being similar to the division of responsibilities between a homeowner and a homebuilder. The builder (AWS) is responsible for constructing your house and ensuring that it is solidly built. As the homeowner (the customer), it is your responsibility to secure everything in the house by ensuring that the doors are closed and locked.

To learn more, select the **+** symbol next to each category.

**Customers: Security in the cloud**

–

Customers are responsible for the security of everything that they create and put *in*the AWS Cloud.

When using AWS services, you, the customer, maintain complete control over your content. You are responsible for managing security requirements for your content, including which content you choose to store on AWS, which AWS services you use, and who has access to that content. You also control how access rights are granted, managed, and revoked.

The security steps that you take will depend on factors such as the services that you use, the complexity of your systems, and your company’s specific operational and security needs. Steps include selecting, configuring, and patching the operating systems that will run on Amazon EC2 instances, configuring security groups, and managing user accounts.

**AWS: Security of the cloud**

–

AWS is responsible for security *of*the cloud.

AWS operates, manages, and controls the components at all layers of infrastructure. This includes areas such as the host operating system, the virtualization layer, and even the physical security of the data centers from which services operate.

AWS is responsible for protecting the global infrastructure that runs all of the services offered in the AWS Cloud. This infrastructure includes AWS Regions, Availability Zones, and edge locations.

AWS manages the security of the cloud, specifically the physical infrastructure that hosts your resources, which include:

* Physical security of data centers
* Hardware and software infrastructure
* Network infrastructure
* Virtualization infrastructure

Although you cannot visit AWS data centers to see this protection firsthand, AWS provides several reports from third-party auditors. These auditors have verified its compliance with a variety of computer security standards and regulations

Which tasks are the responsibilities of customers? (Select TWO.)

* Maintaining network infrastructure

Correctly unchecked

* Patching software on Amazon EC2 instances

Correctly checked

* Implementing physical security controls at data centers

Correctly unchecked

* Setting permissions for Amazon S3 objects

Correctly checked

* Maintaining servers that run Amazon EC2 instances

Correctly unchecked

SUBMIT

**Correct**

The correct two response options are:

* **Patching software on Amazon EC2 instances**
* **Setting permissions for Amazon S3 objects**

The other three response options are tasks that are the responsibility of AWS.

# User permissions and access

**Video transcript**

–

In the coffee shop, every employee has an identity. They come into work in the morning and they'd log into the system to clock in, use the registers and manage the systems, running the coffee shop, day to day. We have the cash registers, the computers helping run the whole operation. Each person has unique access to these systems based on who they are.

If I have Rudy at the register taking orders and Blaine in the back, checking on the inventory levels on the computer, they have two different logins and two different sets of permissions. Rudy can run the cash register, but if he went to log into the inventory system, he wouldn't be allowed to do that.

You will want to scope your users permissions in AWS in a similar way. When you create an AWS account, you are given what is called the root account user. This root user is the owner of the AWS account and has permission to do anything they want inside of that account. This is like being the owner of the coffee shop.

In this situation, let's say I am the owner of the coffee shop. I can come into the shop, use my credentials to work the register, work the inventory system or any other system in the coffee shop. I cannot be restricted. With the AWS root user, you can access and control any resource in the account. You can spin up databases, EC2 instances, blockchain services, or literally whatever you want. Because that user is so powerful, we recommend that as soon as you create an AWS account and log in with your root user, you turn on multi-factor authentication, or MFA, to ensure that you need not only the email and password, but also a randomized token to log in.

That is great. But even with MFA turned on, in reality you don't want to use the root user for everything. I, as the coffee shop owner, don't give my level of access to all employees. Rudy on the cash register cannot access the inventory system, remember? You control access in a granular way by using the AWS service, AWS Identity and Access Management, or IAM.

In IAM, you can create IAM users. When you create an IAM user, by default, it has no permissions. The user can't even log into the AWS account at first, it has absolutely zero permissions. It can not launch an EC2 instance. It can not create an S3 bucket. Nothing. You have to explicitly give the user permission to do anything in that account. Remember, by default, all actions are denied. You have to explicitly allow any action done by any user. You give people access only to what they need and nothing else. This idea is called the least privilege principle.

The way that you grant or deny permission is to associate what is called an IAM policy to an IAM user. An IAM policy is a JSON document that describes what API calls a user can or cannot make. Let's look at this quick example. In this example, you can see we have a permission statement that has the effect as Allow, the action as s3:ListBucket. And the resource is a unique ID for an S3 bucket. So if I attach this policy to a user and that user could view the bucket "coffee\_shop\_reports" but perform no other action in this account. There were only two potential options for the effect on any policy. Either allow or deny. For action, you can list any AWS API call and for resource, you would list what AWS resource that specific API call is for. Now, as a business person, you wouldn't need to write these policies, but they are used all over in your AWS accounts.

One way to make it easier to manage your users and their permissions is to organize them into IAM groups. Groups are, well, they are groupings of users. You can attach a policy to a group and all of the users in that group will have those permissions. If you have a bunch of cashiers in the coffee shop, instead of individually granting them all access to the register. Instead, you can grant all cashiers access then just add each individual cashier to the group. Same idea with groups in IAM.

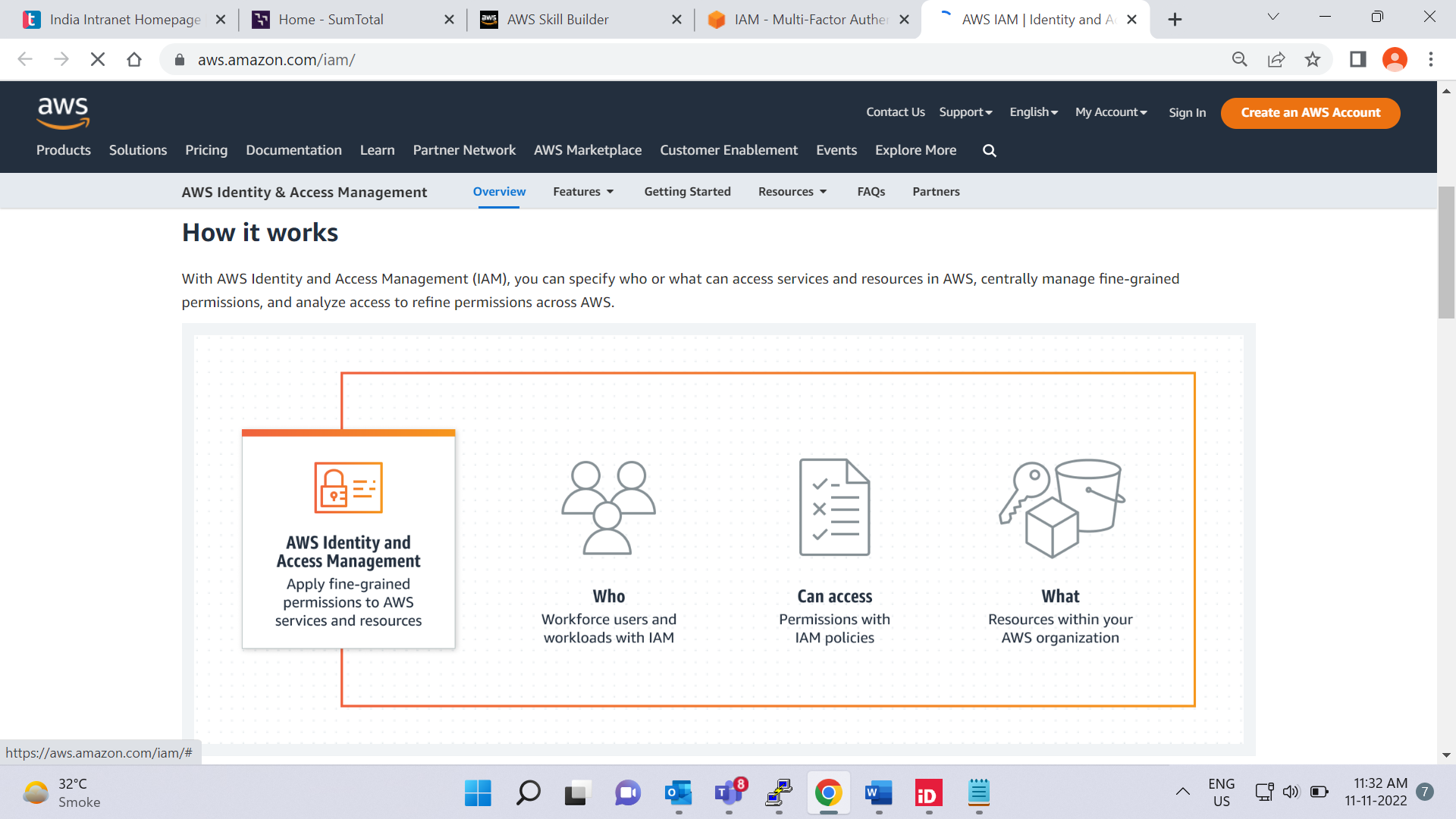
All right, so far with IAM, you have the root user, they can do anything. You have users which can be organized into groups. And you also have policies which are documents that describe permissions that you can then attach to users or groups. There is one other major identity in IAM, and it's called a role.

To understand the idea of roles, let's think about the coffee shop. As we know, Blaine works in the shop and depending on the staffing of the shop day to day, he might work the register or the inventory system or he might be the one cleaning up at the end of the day with access to no systems. I, as the owner, have the authority to assign these different roles to Blaine. His responsibilities and access are variable and change from day to day. Just because he worked on tracking inventory in the system yesterday, doesn't mean that he should be at any time. His role at work changes and is temporary in nature. The same type of idea exists in AWS. You can create identities in AWS that are called roles.

Roles have associated permissions that allow or deny specific actions. And these roles can be assumed for temporary amounts of time. It is similar to a user, but has no username and password. Instead, it is an identity that you can assume to gain access to temporary permissions. You use roles to temporarily grant access to AWS resources, to users, external identities, applications, and even other AWS services. When an identity assumes a role, it abandons all of the previous permissions that it has and it assumes the permissions of that role.

You can actually avoid creating IAM users for every person in your organization by federating users into your account. This means that they could use their regular corporate credentials to log into AWS by mapping their corporate identities to IAM roles. AWS IAM authentication and authorization as a service.

**AWS Identity and Access Management (IAM)**



[**AWS Identity and Access Management (IAM)**](https://aws.amazon.com/iam/) enables you to manage access to AWS services and resources securely.

IAM gives you the flexibility to configure access based on your company’s specific operational and security needs. You do this by using a combination of IAM features, which are explored in detail in this lesson:

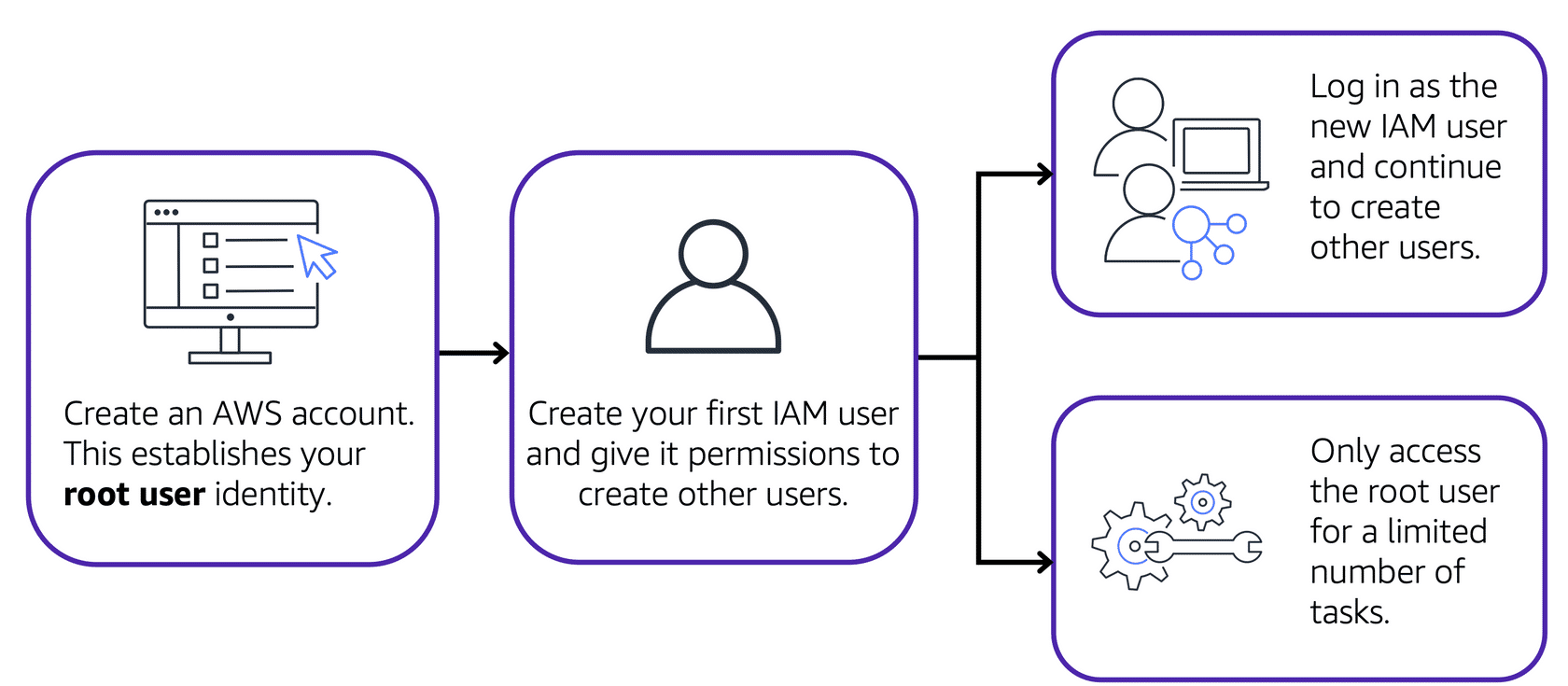
* IAM users, groups, and roles
* IAM policies
* Multi-factor authentication

You will also learn best practices for each of these features.

**AWS account root user**

When you first create an AWS account, you begin with an identity known as the [**root user**](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_root-user.html).

The root user is accessed by signing in with the email address and password that you used to create your AWS account. You can think of the root user as being similar to the owner of the coffee shop. It has complete access to all the AWS services and resources in the account.



Best practice:

Do **not** use the root user for everyday tasks.

Instead, use the root user to create your first IAM user and assign it permissions to create other users.

Then, continue to create other IAM users, and access those identities for performing regular tasks throughout AWS. Only use the root user when you need to perform a limited number of tasks that are only available to the root user. Examples of these tasks include changing your root user email address and changing your AWS support plan.

**IAM users**

An **IAM user** is an identity that you create in AWS. It represents the person or application that interacts with AWS services and resources. It consists of a name and credentials.

By default, when you create a new IAM user in AWS, it has no permissions associated with it. To allow the IAM user to perform specific actions in AWS, such as launching an Amazon EC2 instance or creating an Amazon S3 bucket, you must grant the IAM user the necessary permissions.

Best practice:

We recommend that you create individual IAM users for each person who needs to access AWS.

Even if you have multiple employees who require the same level of access, you should create individual IAM users for each of them. This provides additional security by allowing each IAM user to have a unique set of security credentials.

**IAM policies**

An **IAM policy** is a document that allows or denies permissions to AWS services and resources.

IAM policies enable you to customize users’ levels of access to resources. For example, you can allow users to access all of the Amazon S3 buckets within your AWS account, or only a specific bucket.

Best practice:

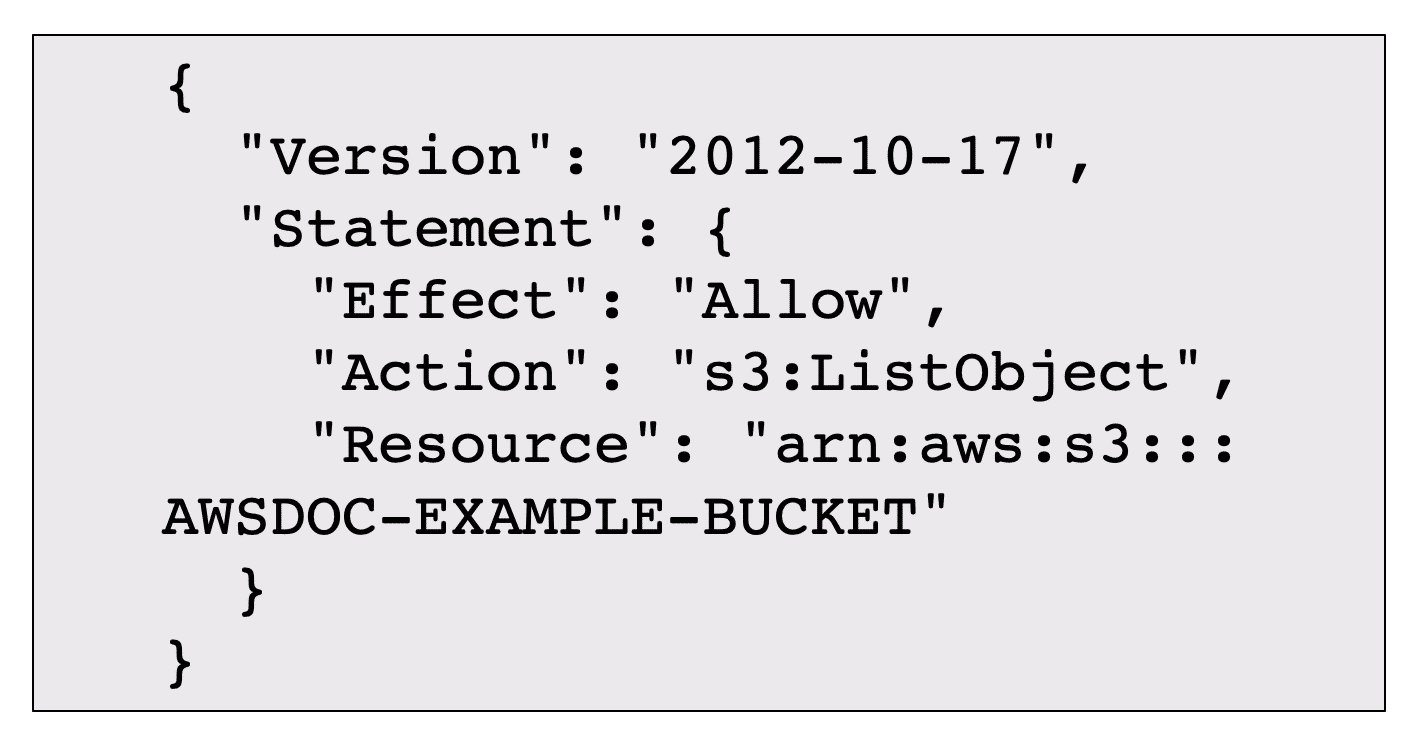
Follow the security principle of **least privilege** when granting permissions.

By following this principle, you help to prevent users or roles from having more permissions than needed to perform their tasks.

For example, if an employee needs access to only a specific bucket, specify the bucket in the IAM policy. Do this instead of granting the employee access to all of the buckets in your AWS account.

**Example: IAM policy**

Here’s an example of how IAM policies work. Suppose that the coffee shop owner has to create an IAM user for a newly hired cashier. The cashier needs access to the receipts kept in an Amazon S3 bucket with the ID: AWSDOC-EXAMPLE-BUCKET.



This example IAM policy allows permission to access the objects in the Amazon S3 bucket with ID: *AWSDOC-EXAMPLE-BUCKET*.

In this example, the IAM policy is allowing a specific action within Amazon S3: ListObject. The policy also mentions a specific bucket ID: AWSDOC-EXAMPLE-BUCKET. When the owner attaches this policy to the cashier’s IAM user, it will allow the cashier to view all of the objects in the AWSDOC-EXAMPLE-BUCKET bucket.

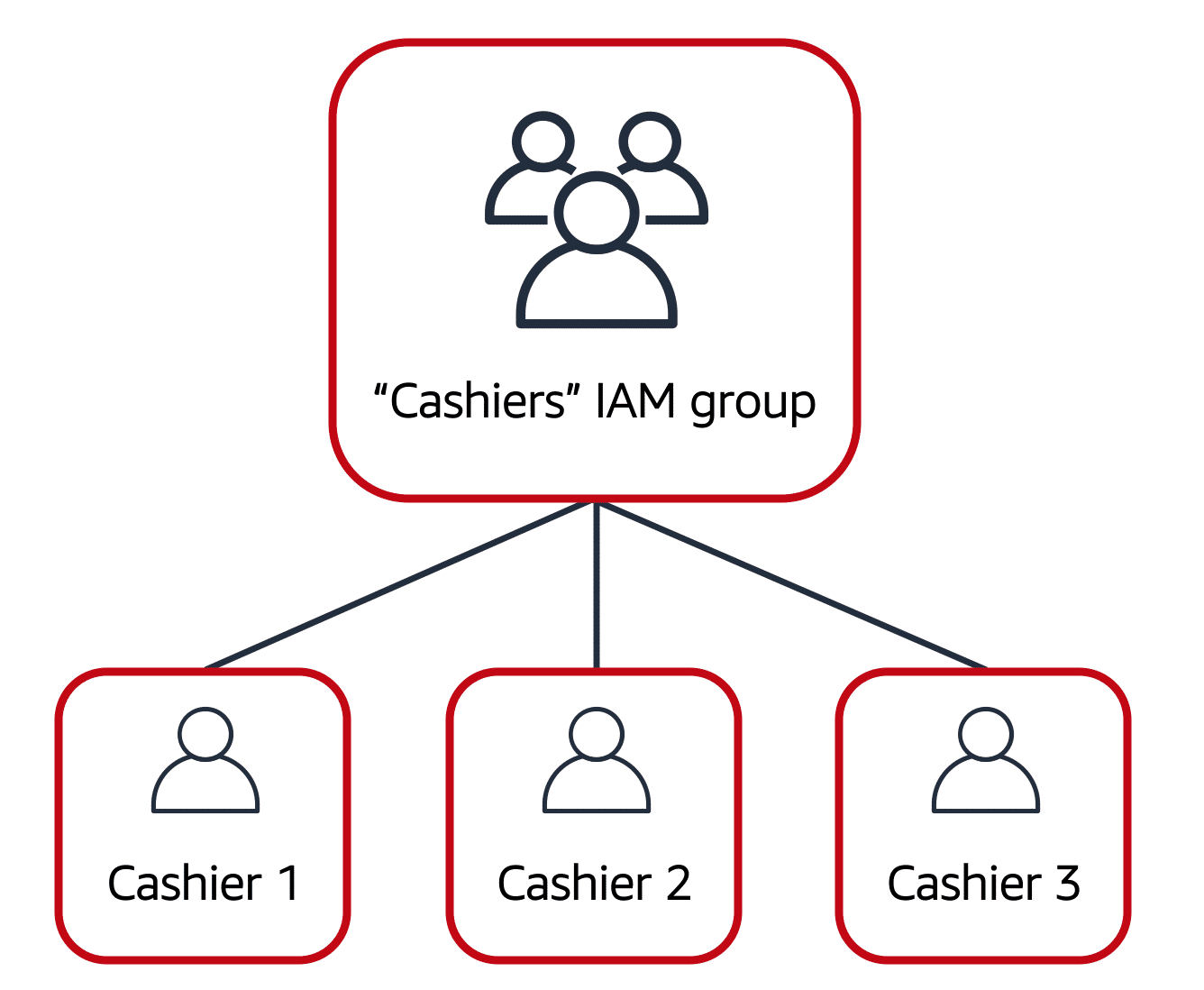
If the owner wants the cashier to be able to access other services and perform other actions in AWS, the owner must attach additional policies to specify these services and actions.

Now, suppose that the coffee shop has hired a few more cashiers. Instead of assigning permissions to each individual IAM user, the owner places the users into an [**IAM group**](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_groups.html).

**IAM groups**

An IAM group is a collection of IAM users. When you assign an IAM policy to a group, all users in the group are granted permissions specified by the policy.

Here’s an example of how this might work in the coffee shop. Instead of assigning permissions to cashiers one at a time, the owner can create a “Cashiers” IAM group. The owner can then add IAM users to the group and then attach permissions at the group level.



Assigning IAM policies at the group level also makes it easier to adjust permissions when an employee transfers to a different job. For example, if a cashier becomes an inventory specialist, the coffee shop owner removes them from the “Cashiers” IAM group and adds them into the “Inventory Specialists” IAM group. This ensures that employees have only the permissions that are required for their current role.

What if a coffee shop employee hasn’t switched jobs permanently, but instead, rotates to different workstations throughout the day? This employee can get the access they need through [**IAM roles**](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html).

**IAM roles**

In the coffee shop, an employee rotates to different workstations throughout the day. Depending on the staffing of the coffee shop, this employee might perform several duties: work at the cash register, update the inventory system, process online orders, and so on.

When the employee needs to switch to a different task, they give up their access to one workstation and gain access to the next workstation. The employee can easily switch between workstations, but at any given point in time, they can have access to only a single workstation. This same concept exists in AWS with IAM roles.

An IAM role is an identity that you can assume to gain temporary access to permissions.

Before an IAM user, application, or service can assume an IAM role, they must be granted permissions to switch to the role. When someone assumes an IAM role, they abandon all previous permissions that they had under a previous role and assume the permissions of the new role.

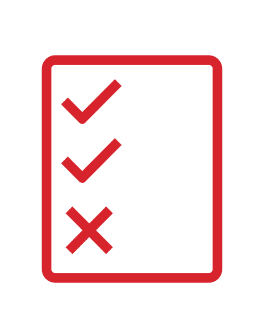
Best practice:

IAM roles are ideal for situations in which access to services or resources needs to be granted temporarily, instead of long-term.

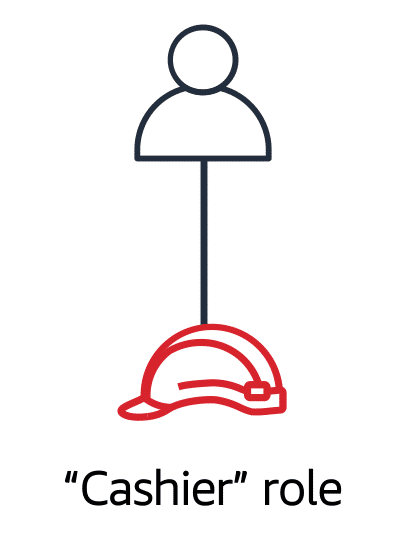
**Step 1**

## Example: IAM roles

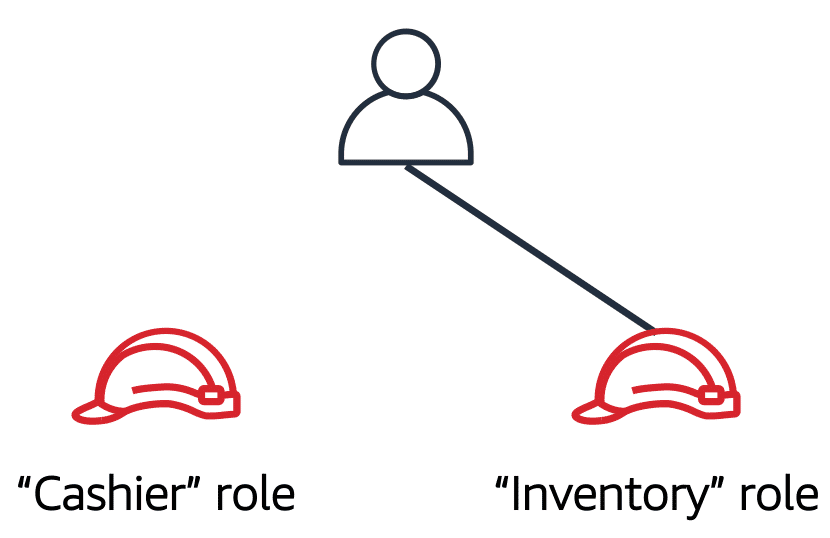
To review an example of how IAM roles could be used in the coffee shop



First, the owner grants the employee permissions to switch to a role for each workstation in the coffee shop.



The employee begins their day by assuming the “Cashier” role. This grants them access to the cash register system.



Later in the day, the employee needs to update the inventory system. They assume the “Inventory” role.

This grants the employee access to the inventory system and also revokes their access to the cash register system.

**Multi-factor authentication**

Have you ever signed in to a website that required you to provide multiple pieces of information to verify your identity? You might have needed to provide your password and then a second form of authentication, such as a random code sent to your phone. This is an example of [**multi-factor authentication**](https://aws.amazon.com/iam/features/mfa/).

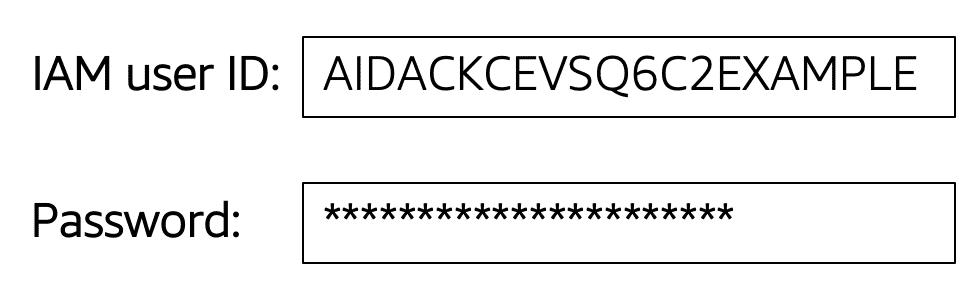
In IAM, multi-factor authentication (MFA) provides an extra layer of security for your AWS account.

## How multi-factor authentication works

To review the steps involved with multi-factor authentication, select **Start**.

START

**Step 1**



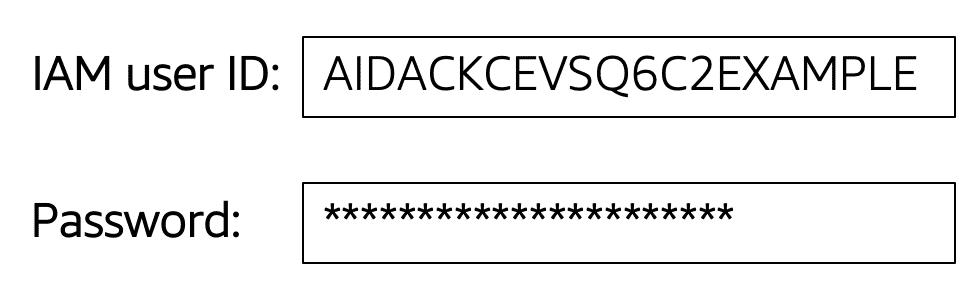
First, when a user signs in to an AWS website, they enter their IAM user ID and password.

## How multi-factor authentication works

To review the steps involved with multi-factor authentication, select **Start**.

START

**Step 1**

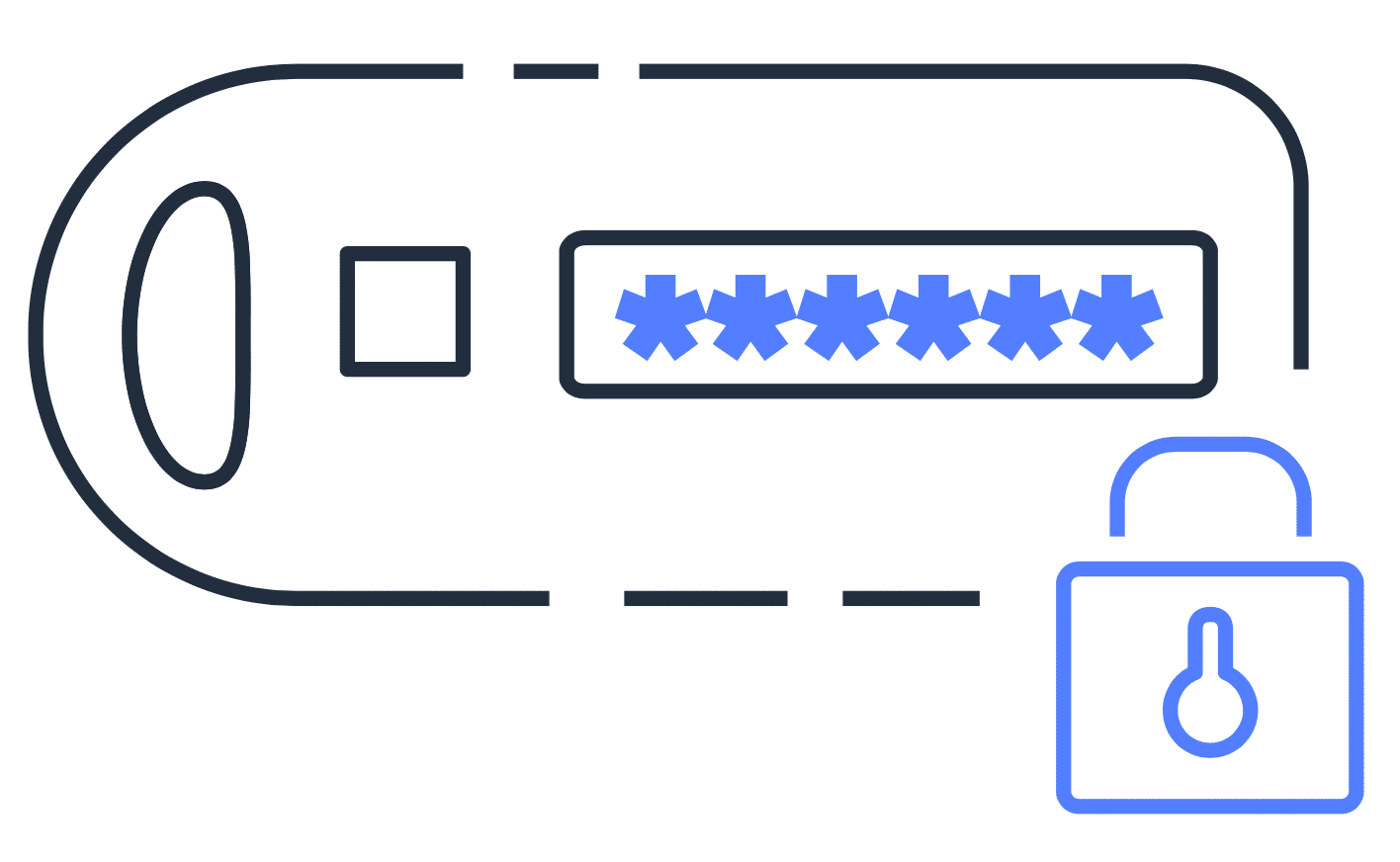


First, when a user signs in to an AWS website, they enter their IAM user ID and password.

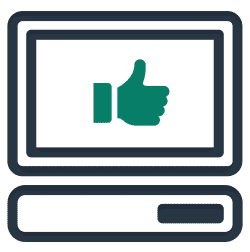
1

2

**Step 2**



Next, the user is prompted for an authentication response from their AWS MFA device. This device could be a hardware security key, a hardware device, or an MFA application on a device such as a smartphone.



When the user has been successfully authenticated, they are able to access the requested AWS services or resources.

You can enable MFA for the root user and IAM users. As a best practice, enable MFA for the root user and all IAM users in your account. By doing this, you can keep your AWS account safe from unauthorized access.

# AWS Organizations

**Video transcript**

–

With your first foray into the AWS Cloud, you most likely will start with one AWS account and have everything reside in there. Most people start this way, but as your company grows or even begins their cloud journey, it's important to have a separation of duties. For example, you want your developers to have access to development resources, have your accounting staff able to access billing information, or even have business units separate so that they can experiment with AWS services without effecting each other. So you start to add more cards for each person, whoever needs to onboard. And before you know it, you end up with a tangled bowl of AWS account spaghetti, not as tasty as you might imagine.

For example, you'll then have to keep track of Account A, F, and G, or maybe Account B has the wrong permissions and Account C has billing and compliance info. One way to install order and to enforce who is allowed to perform certain functions in what account is to make use of an AWS service called AWS Organizations.

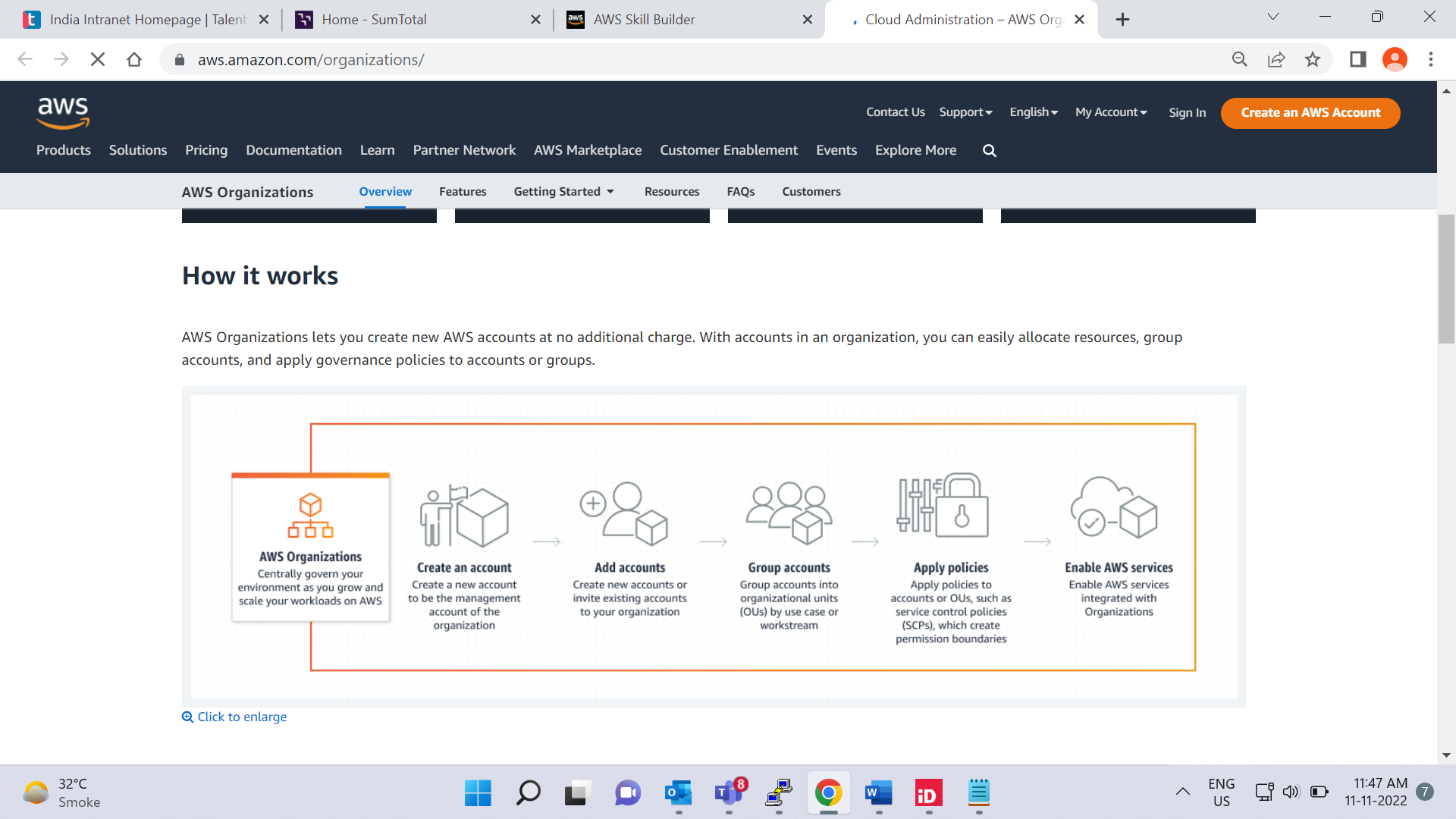
The easiest way to think of Organizations is as a central location to manage multiple AWS accounts. You can manage billing control, access, compliance, security, and share resources across your AWS accounts. Let's outline some of the main features of AWS Organizations, shall we?

The first is centralized management of all your AWS accounts. Think of all those AWS accounts, we had: A, B, C, F, G. Now you can combine them into an organization that enables us to manage the accounts centrally, and wow. Now we've found Accounts D and E in the process. Next up is consolidated billing for all member accounts. This means you can use the primary account of your organization to consolidate and pay for all member accounts. Another advantage of consolidated billing is bulk discounts. Cash money, indeed.

The next feature is that you can implement hierarchical groupings of your accounts to meet security, compliance, or budgetary needs. This means you can group accounts into organizational units, or OUs, kind of like business units, or BUs. For example, if you have accounts that must access only the AWS services that meet certain regulatory requirements, you can put those accounts into one OU, or if you have accounts that fall under the developer OU, you can group them accordingly.

One of the last main features we'll touch upon is that you have control over the AWS services and API actions that each account can access as an administrator of the primary account of an organization. You can use something called service control policies, or SCPs, to specify the maximum permissions for member accounts in the organization. In essence, with SCPs you can restrict which AWS services, resources, and individual API actions, the users and roles in each member account can access.

**AWS Organizations**



Suppose that your company has multiple AWS accounts. You can use [**AWS Organizations**](https://aws.amazon.com/organizations)to consolidate and manage multiple AWS accounts within a central location.

When you create an organization, AWS Organizations automatically creates a **root**, which is the parent container for all the accounts in your organization.

In AWS Organizations, you can centrally control permissions for the accounts in your organization by using [**service control policies (SCPs)**](https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_policies_scps.html). SCPs enable you to place restrictions on the AWS services, resources, and individual API actions that users and roles in each account can access.

Consolidated billing is another feature of AWS Organizations. You will learn about consolidated billing in a later module.

**Organizational units**

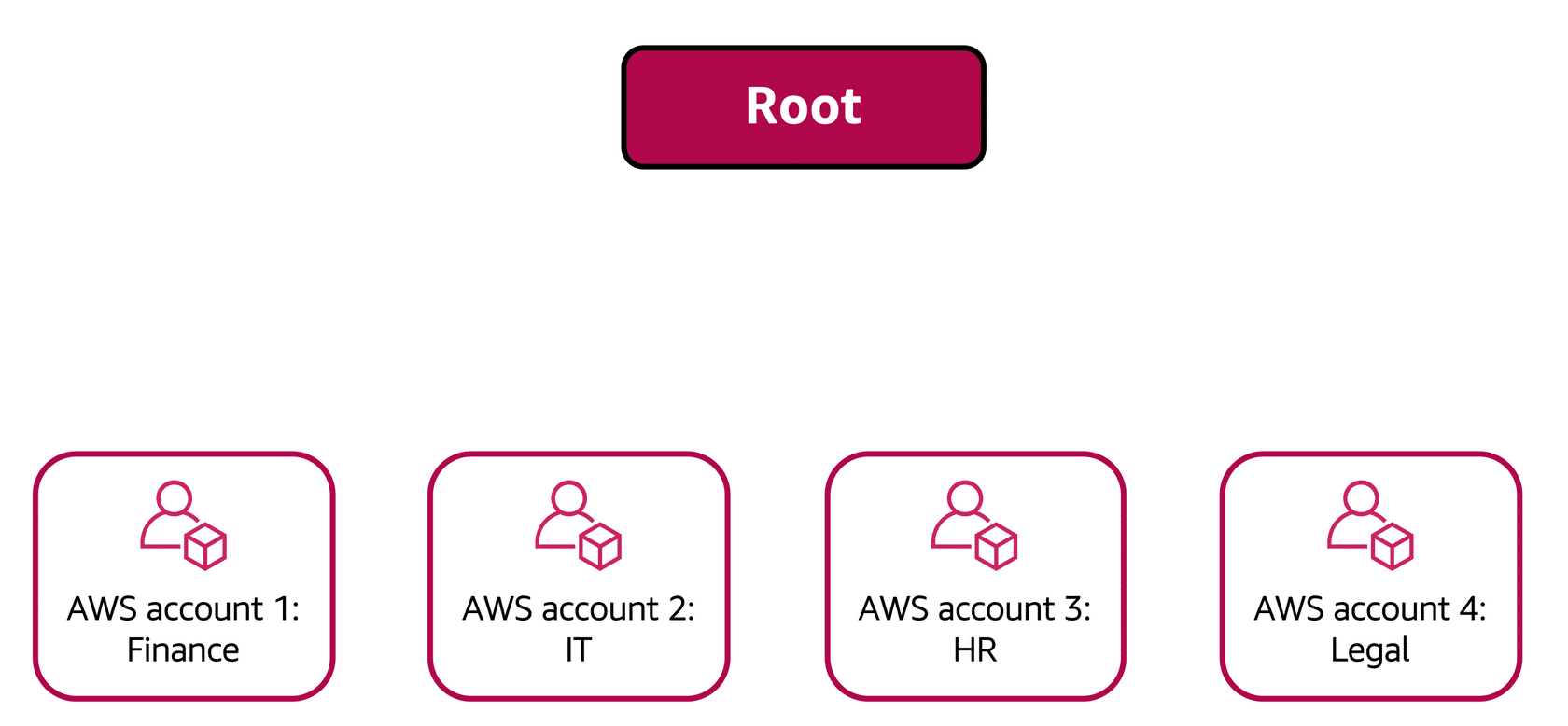
In AWS Organizations, you can group accounts into organizational units (OUs) to make it easier to manage accounts with similar business or security requirements. When you apply a policy to an OU, all the accounts in the OU automatically inherit the permissions specified in the policy.

By organizing separate accounts into OUs, you can more easily isolate workloads or applications that have specific security requirements. For instance, if your company has accounts that can access only the AWS services that meet certain regulatory requirements, you can put these accounts into one OU. Then, you can attach a policy to the OU that blocks access to all other AWS services that do not meet the regulatory requirements.

## xample: AWS Organizations

To review an example of how a company might use AWS Organizations

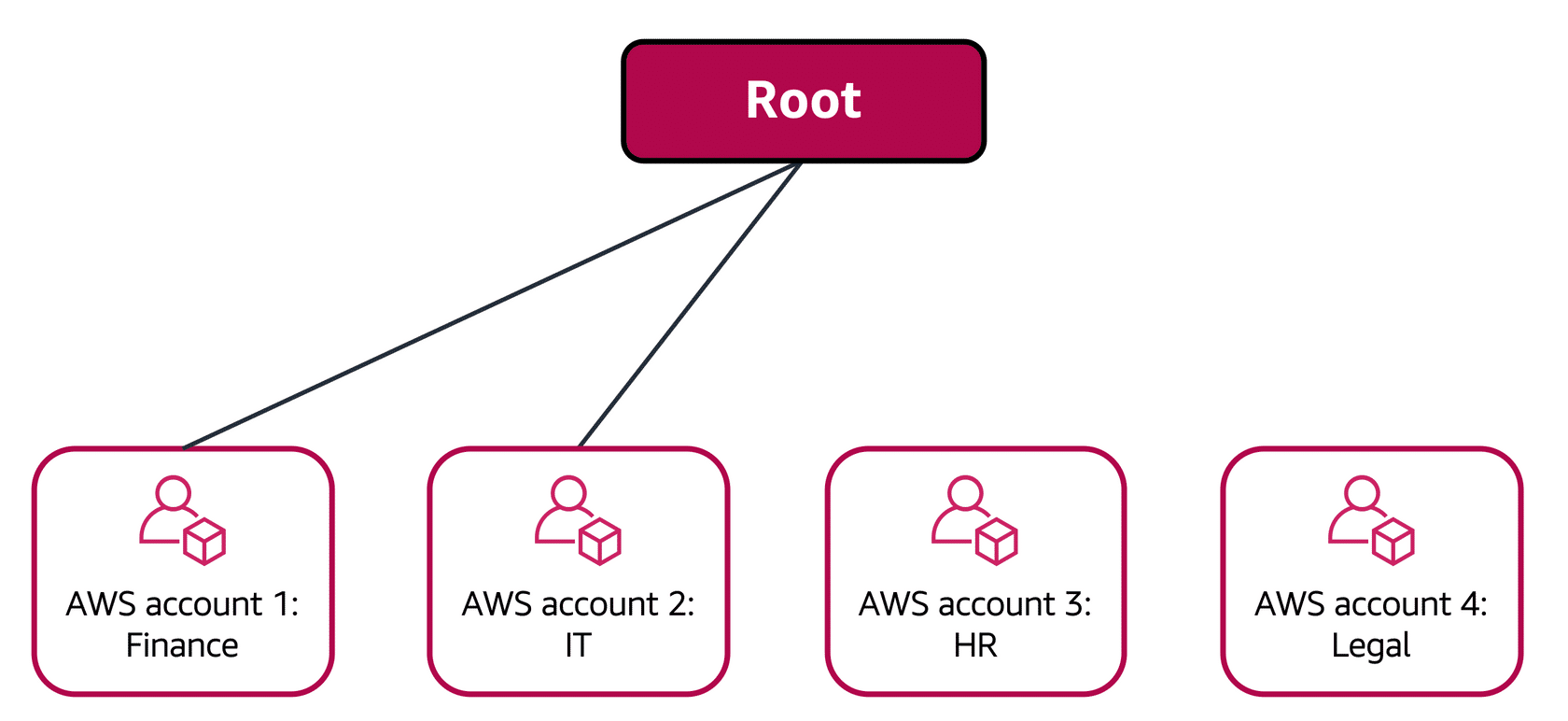
**Step 1**



Imagine that your company has separate AWS accounts for the finance, information technology (IT), human resources (HR), and legal departments. You decide to consolidate these accounts into a single organization so that you can administer them from a central location. When you create the organization, this establishes the root.

In designing your organization, you consider the business, security, and regulatory needs of each department. You use this information to decide which departments group together in OUs.

**Step 2**



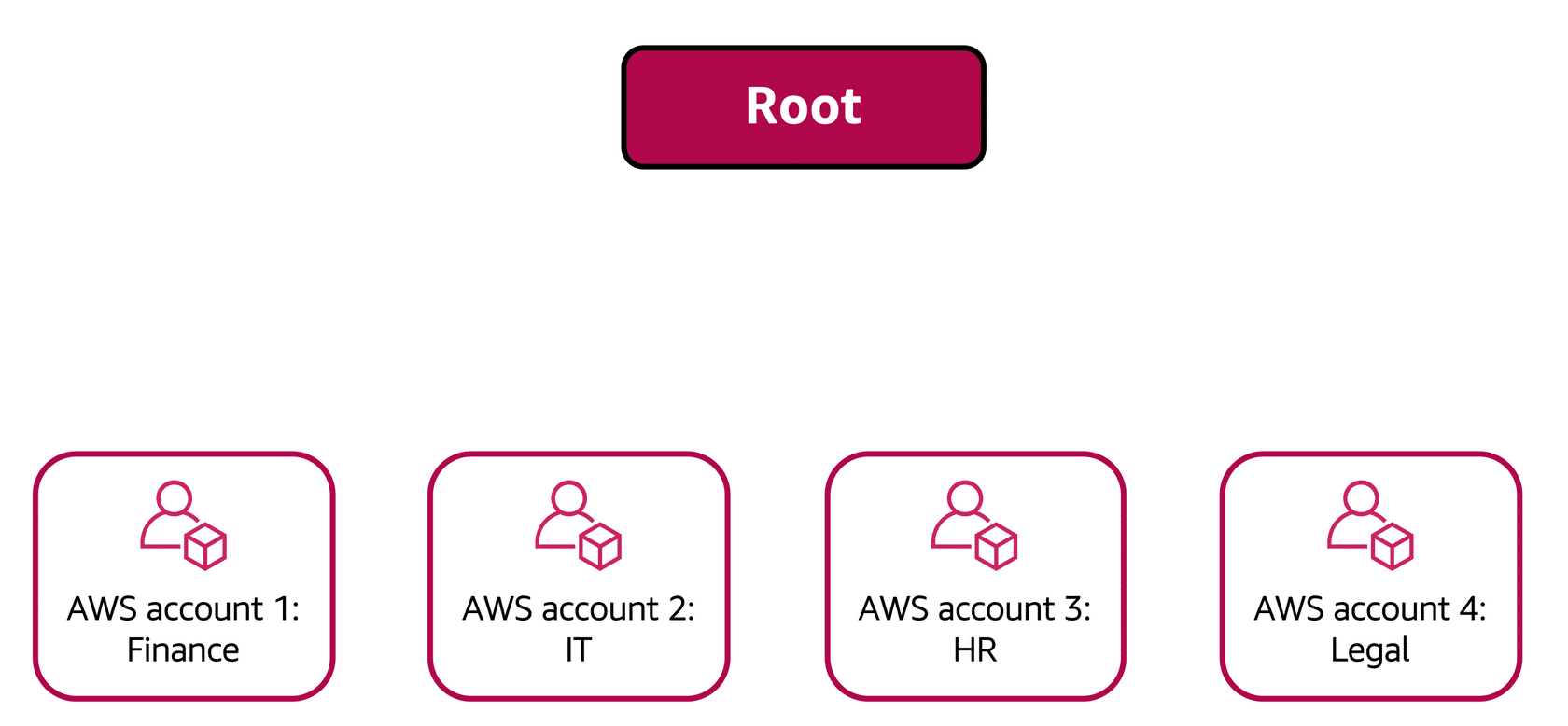
The finance and IT departments have requirements that do not overlap with those of any other department. You bring these accounts into your organization to take advantage of benefits such as consolidated billing, but you do not place them into any OUs.

## Example: AWS Organizations

To review an example of how a company might use AWS Organizations, select **Start**.

START

**Step 1**



Imagine that your company has separate AWS accounts for the finance, information technology (IT), human resources (HR), and legal departments. You decide to consolidate these accounts into a single organization so that you can administer them from a central location. When you create the organization, this establishes the root.

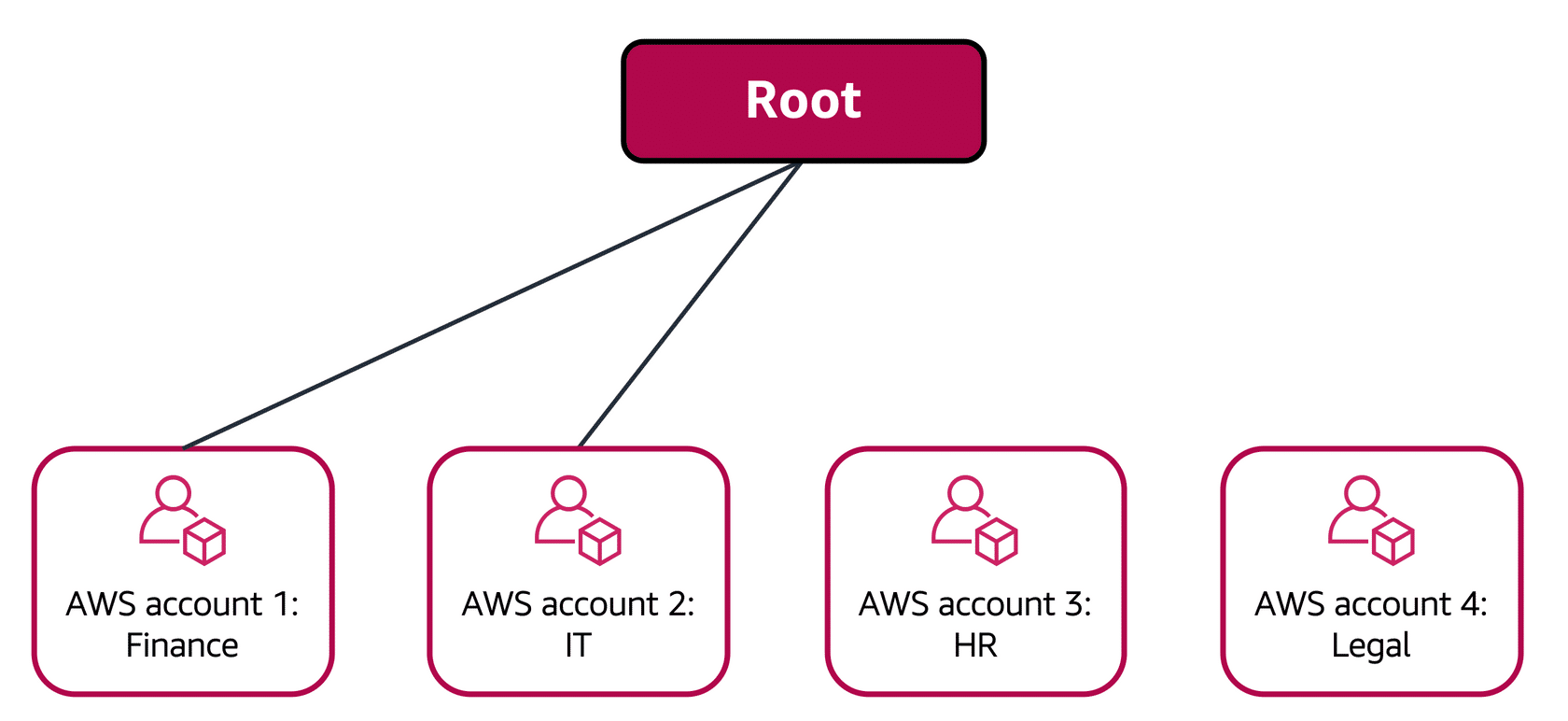
In designing your organization, you consider the business, security, and regulatory needs of each department. You use this information to decide which departments group together in OUs.

1

2

3

**Step 2**



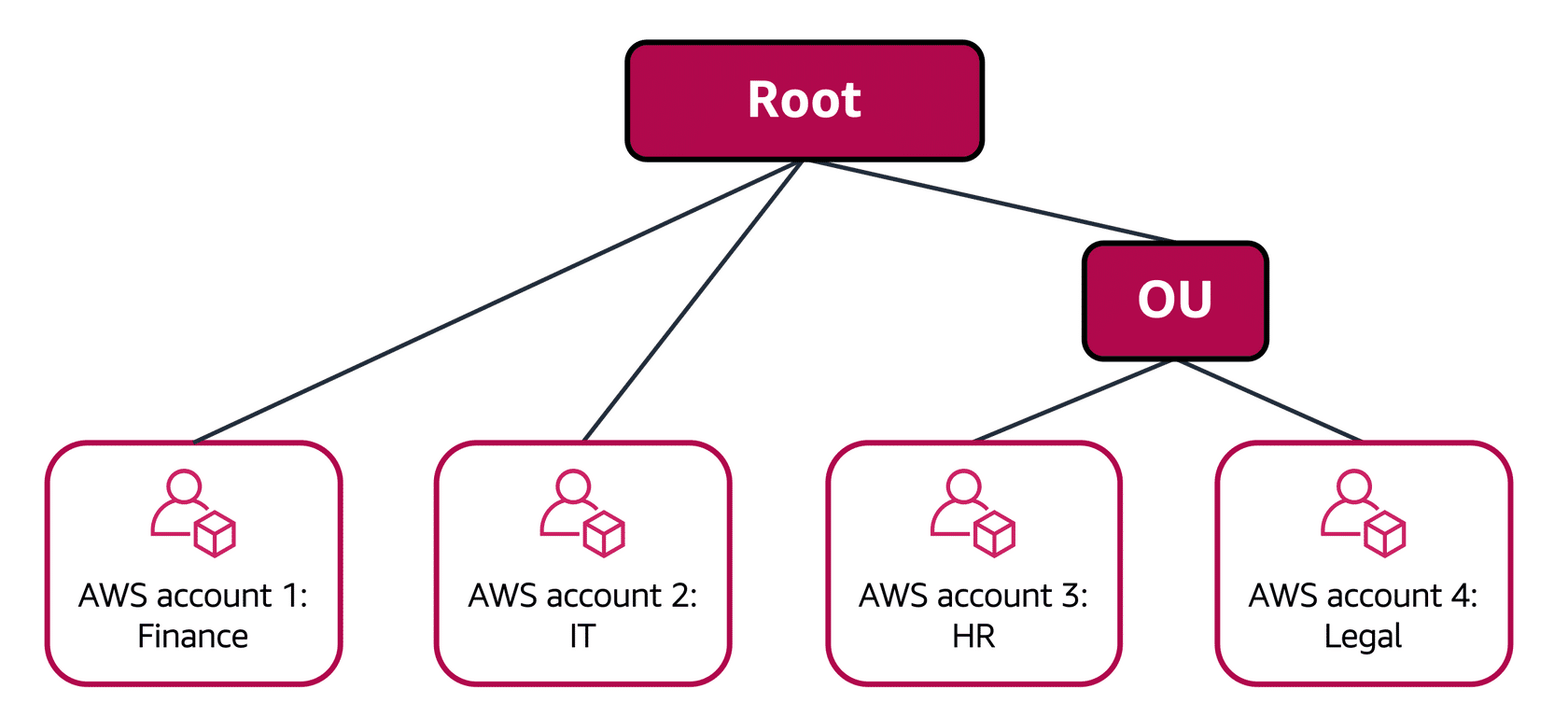
The finance and IT departments have requirements that do not overlap with those of any other department. You bring these accounts into your organization to take advantage of benefits such as consolidated billing, but you do not place them into any OUs.

1

2

3

**Step 3**



The HR and legal departments need to access the same AWS services and resources, so you place them into an OU together. Placing them into an OU enables you to attach policies that apply to both the HR and legal departments’ AWS accounts.

Even though you have placed these accounts into OUs, you can continue to provide access for users, groups, and roles through IAM.

By grouping your accounts into OUs, you can more easily give them access to the services and resources that they need. You also prevent them from accessing any services or resources that they do not need.

**Knowledge check**

You are configuring service control policies (SCPs) in AWS Organizations. Which identities and resources can SCPs be applied to? (Select TWO.)

* IAM users

Correctly unchecked

* IAM groups

Correctly unchecked

* An individual member account

Correctly checked

* IAM roles

Correctly unchecked

* An organizational unit (OU)

Correctly checked

SUBMIT

**Correct**

The correct two response options are:

* **An individual member account**
* **An organizational unit (OU)**

In AWS Organizations, you can apply service control policies (SCPs) to the organization root, an individual member account, or an OU. An SCP affects all IAM users, groups, and roles within an account, including the AWS account root user.

You can apply IAM policies to IAM users, groups, or roles. You cannot apply an IAM policy to the AWS account root user.