# Chapter 3

## Designing Secure Access to AWS Resources

This chapter covers the following topics:

- **Identity and Access Management (IAM)**
- **AWS IAM Users and Groups**
- **Creating IAM Policies**
- **IAM Roles**
- **AWS Organizations**
- **AWS Resource Access Manager**
- **AWS Control Tower**

This chapter covers content that's important to the following exam domain and task statement:

**Domain 1: Design Secure Architectures**

Task Statement 1: Design secure access to AWS resources

It's only natural to think that the same security issues and concerns you face on premises could occur—and possibly be more widespread—when operating in the AWS cloud. Amazon is continuously patching and updating its entire cloud infrastructure, managed services, and all other integral system components. The latest security bulletins published by AWS (see **https://aws.amazon.com/security/security-bulletins/**) indicate Amazon has needed to consider various security advisories on behalf of all AWS customers.

AWS is responsible for maintaining the security of its cloud infrastructure, defined by AWS as security *of* the cloud. Our job is to maintain everything we host and store in the cloud. This concept is security *in* the cloud.

Indeed, the job of securing customer resources hosted in the AWS cloud never ends, just as security issues continue to be discovered. Way back in 2014, security vulnerabilities with the Bash shell were found. Bash shell security issues weren't new; they just hadn't been discovered until 2014.

To be successful when taking the AWS Certified Solutions Architect – Associate (SAA-C03) exam, a good understanding of the tools and methods available for maintaining the security of workloads, administrators, and cloud services is required.

### "Do I Know This Already?"

The "Do I Know This Already?" quiz allows you to assess whether you should read this entire chapter thoroughly or jump to the "Exam Preparation Tasks" section. If you are in doubt about your answers to these questions or your own assessment of your knowledge of the topics, read the entire chapter. **Table 3-1** lists the major headings in this chapter and their corresponding "Do I Know This Already?" quiz questions. You can find the answers in **Appendix A**, "**Answers to the 'Do I Know This Already?' Quizzes and Q&A Sections.**"

**Table 3-1** "Do I Know This Already?" Section-to-Question Mapping

| Foundation Topics Section | Questions |
|---|---|
| Identity and Access Management (IAM) | 1, 2 |
| IAM Users and Groups | 3, 4 |
| Creating IAM Policies | 5, 6 |
| IAM Roles | 7, 8 |
| AWS Organizations | 9, 10 |
| AWS Resource Access Manager | 11, 12 |
| AWS Control Tower | 13, 14 |

**Caution**

The goal of self-assessment is to gauge your mastery of the topics in this chapter. If you do not know the answer to a question or are only partially sure of the answer, you should mark that question as wrong for purposes of the self-assessment. Giving yourself credit for an answer you correctly guess skews your

self-assessment results and might provide you with a false sense of security.

---

**1.** What process must happen before AWS IAM grants an IAM user access to requested AWS resources?

1. Authorization
2. Authentication
3. Access granted
4. Access denied

**2.** Which of the following entities is not controlled by AWS IAM?

1. IAM groups
2. IAM user
3. Externally authenticated user
4. Database authentication

**3.** What additional step can be added as a mandatory component when authenticating at AWS?

1. Password policies
2. Multi-factor authentication
3. Resource policies
4. Management policies

**4.** Which of the following AWS IAM entities cannot authenticate?

1. Management policies
2. IAM groups
3. Job policies
4. Secret keys

**5.** What type of permissions policy applies to resources?

1. IAM managed policy
2. Resource policy
3. Job function policy
4. Custom IAM policy

**6.** Which of the following options can be added to a policy as a conditional element?

1. Explicit denies
2. Tags
3. Explicit allows
4. Implicit allows

**7.** What essential component is not attached to an AWS IAM role?

1. Security policy
2. Credentials
3. Multi-factor authentication
4. Tags

**8.** What security service does an AWS IAM role interface with?

1. AWS CloudTrail
2. AWS Security Token Service
3. AWS Config
4. Amazon GuardDuty

**9.** How does an AWS organization help with managing costs?

1. Organizational units
2. Consolidated billing
3. Service control policy
4. Shared security services

**10.** Which of the following is the primary benefit of creating an AWS organization?

1. Lower costs for linked accounts
2. Centralized control of linked AWS accounts
3. Distributed control
4. Nesting OUs

**11.** What is a benefit of using AWS Resource Access Manager without AWS Organizations being deployed?

1. Replacement of network peering
2. Sharing of resources between AWS accounts
3. Sharing of resources between regions
4. Sharing of resources in different availability zones

**12.** Who is in charge of sharing resources using AWS Resource Access Manager?

1. The resource user
2. The principal ID
3. The AWS account root user
4. Any IAM administrator

**13.** How is governance carried out by AWS Control Tower?

1. Account Factory
2. Landing zone
3. Guardrails
4. Dashboard

**14.** What is the purpose of the AWS Control Tower Account Factory?

1. Apply mandatory guardrails
2. Provision new IAM accounts
3. Standardize the provisioning of new AWS accounts
4. Create OUs

## Foundation Topics

### Identity and Access Management (IAM)

The **Identity and Access Management (IAM)** service is used to deploy and maintain security using security policies and integrated security services. Security policies define what actions administrators (users and groups) and AWS cloud services can and can't carry out. IAM is a global AWS service that protects AWS resources located around the world across all AWS regions, with a global endpoint located at **https://sts.amazonaws.com**. There are different user types that require access to AWS resources, including an organization's AWS cloud administrators and end users who are not aware that they are even accessing the AWS cloud when they are using a SaaS application running on their phone. Many popular mobile SaaS applications have backend services hosted at AWS.

Identity and Access Management (IAM) (see **Figure 3-1**) was added as a feature to the AWS cloud on September 1, 2010, providing the capability for administrators to define the desired level of access to AWS workloads and the associated cloud services. Using the IAM dashboard, administrators must create IAM users, groups, and roles—no default users, groups, or roles are defined when an AWS account is created.

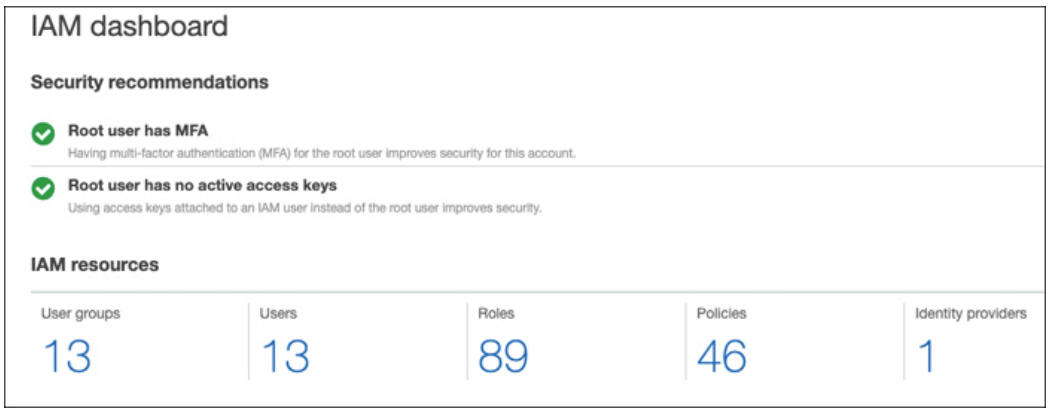


**Figure 3-1** The IAM Dashboard

IAM is a paranoid security service, and we *want* it to be paranoid. IAM's default mindset is explicit denial with no default access to any AWS cloud service defined.

AWS is responsible for building, securing, and maintaining all the physical components that comprise the cloud services available through the AWS cloud portal. When operating in the public cloud, both the customers and the cloud provider have defined responsibilities, which is defined as a *shared responsibility model*.

The responsibilities of AWS are described as *security of the cloud*—protecting the infrastructure that hosts and runs the AWS cloud services. Each AWS customer's responsibility is defined as *security in the cloud*. Each cloud service ordered by a customer will have a default security configuration applied, and from this point forward each customer assumes the responsibility of managing and maintaining the cloud service's current and future security configuration. When an EC2 instance is deployed, Amazon is responsible for launching and hosting the EC2 instance on the subnet and availability zone chosen by the customer and ensuring it is initially accessible only to the customer that requested the instance. In the future, a customer may decide to share the EC2 instance with other AWS customers or across the Internet; the exact level of security is the customer's choice—and responsibility. Each AWS service follows the defined shared responsibility model; each party's responsibilities are clearly laid out in AWS documentation. AWS adheres to the ISO/IEC 27001 security standards, which define security management best practices in managing information security through defined security controls.

Amazon cloud services that access the resources in your AWS account on your behalf are also governed by special IAM policies called *service-linked roles* that define the maximum permitted level of access for each cloud service. Identity and Access Management's main features are as follows:

- **Full integration with all AWS services:** Access to every AWS service can be controlled using IAM security.
- **Cost benefits:** There is no additional charge for using IAM security to control access to AWS resources.
- **Controlled access to your AWS account resources in all regions:** Each IAM user can be assigned security policies controlling access to AWS resources in any AWS account in any AWS region.
- **Granular permission control:** IAM can control access to AWS resources to a granular level, for example, defining a policy that allows an IAM user the singular task of viewing a load balancer's attributes.
- **Define the level of access AWS services have to resources:** When you order an AWS cloud service such as AWS Config, the service is granted access to your AWS account through an IAM security policy controlled by a service-linked role (see **Figure 3-2**), ensuring that the AWS service can only carry out the approved list of tasks.
- **Multi-factor authentication (MFA):** An additional layer of authentication can be added to any IAM user, including the root user of each AWS account. **Multi-factor authentication (MFA)** provides a security code—from a software or hardware device that is linked to your AWS account—that must be entered and verified in order for authentication to AWS to succeed.
- **Identity federation/SSO access to AWS services:** IAM roles allow **externally authenticated users**—whose identities have been federated from a corporate directory or from a third-party identity provider such as Google or Facebook—to have temporary access to select AWS services.

Key Topic

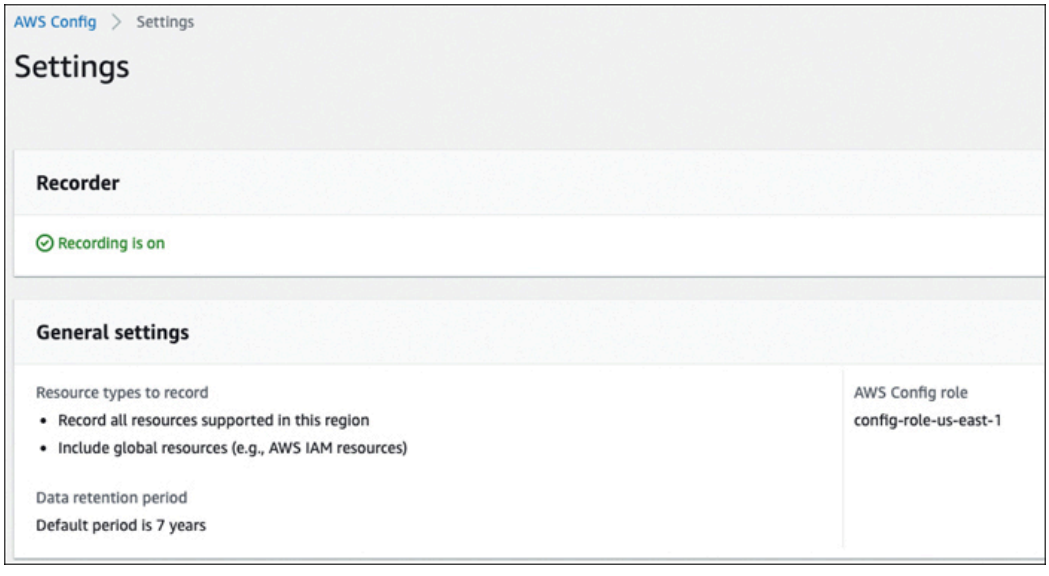

**Figure 3-2** AWS Config Service-Linked Roles Defined by IAM

> **Note**
>
> Because IAM is a global service, security changes and additions to the IAM service can take several minutes to completely replicate across the AWS cloud.


### IAM Policy Definitions

Each IAM policy is a set of rules that define the actions that each AWS entity can perform on specific AWS resources; *who* is allowed to do *what*. To understand IAM, we need to understand the following terms:

- **User:** Only defined IAM users within each AWS account and externally authenticated users with assigned roles can authenticate to an AWS account. An example of an externally authenticated user could be a corporate user who first authenticates to the corporate Active Directory network that also requires access to AWS resources. After AWS verifies the externally authenticated user's attached IAM role, temporary credentials are assigned, allowing the external authenticated user access to the requested AWS resources. Google and Facebook users are examples of externally authenticated users supported by IAM roles and AWS.
- **Group:** A group of IAM users can access AWS resources based on the IAM policies assigned to the IAM group they belong to.
- **Policy:** Each AWS service can be controlled by IAM policies created and managed by AWS or by custom policies created by customers.
- **Statement:** Policy statements define what actions are allowed or denied to AWS resources.
- **Principal:** The principal is an IAM user or application that can perform actions on an AWS resource.
- **Resource:** A resource is an AWS resource (such as compute, storage, networking, or managed services) where actions are performed.
- **Identity:** An identity is the IAM user, group, or role where an IAM policy is attached.
- **Entities:** IAM entities that can authenticate are an IAM user, which is assigned permanent credentials, or an IAM role, which does not have

attached credentials (that is, no password or permanent access keys). Temporary authentication credentials and session tokens are assigned to a role only after verification confirms that the identity is allowed to assume the policy assigned to the IAM role.

- **Role:** An IAM role provides temporary access to AWS resources based on the attached IAM policy.
- **Condition:** Specific conditions can be mandated. For example, a specific principal, IP address, date, or tag must be present before access is allowed. Conditions are optional.

### IAM Authentication


Before tasks can be performed on AWS resources, you must first be authenticated as an IAM user signing in with a recognized IAM username and password or have been granted access using an IAM Role. If multifactor authentication is enabled, you must also enter a numerical code during the authentication process before authentication is successful.

Authentication is also required when running commands or scripts using the AWS command-line interface (CLI) or software development kit (SDK). A valid **access key** and secret access key assigned to the IAM user account making the CLI or SDK request must be provided and validated before the command or script will execute.

When an IAM user account is created, two access keys are created; the first access key is the *ID key*, which is an uppercase alphabetic string of characters in the format AKIA*XXXXXXXXXXXX*, as shown in **Figure 3-3**. The second access key, the *secret access key*, is a Base64 string that is 40 characters in length.

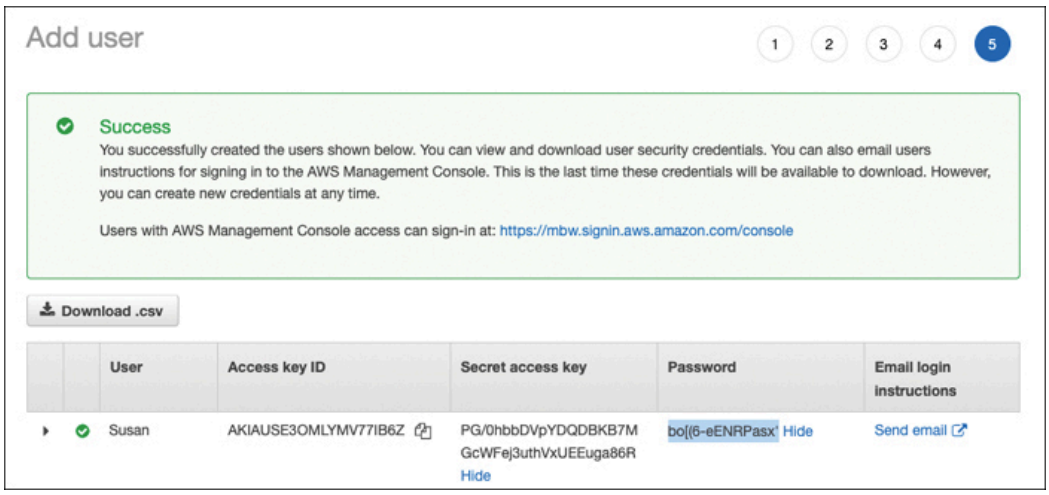


**Figure 3-3** IAM User Account Access Keys

External authentication uses a set of temporary access keys issued by the AWS Security Token Service (AWS STS). Temporary access keys issued from STS are in the format ASIA*XXXXXX*. External authentication is covered in more detail later in this chapter.

Several forms of authentication are supported by AWS, including the following:

- IAM users and groups carry out administrative actions on AWS services and resources.
- Single sign-on (SSO) is supported using a federated identity with Active Directory Domain Services (AD DS) or a third-party provider that supports the Security Association Markup Language (SAML) 2.0 protocol.
- SAML external authentication is supported by IAM using IAM roles, which are attached to the externally authenticated user after identity verification of their external identity by the AWS Security Token Service. Active Directory credentials are stored in two locations: on the Active Directory domain controllers on premises, and on domain controllers hosted at AWS that have been synchronized with a current copy of the organization's Active Directory credentials and attributes.
- Amazon Cognito authenticates and controls access to AWS resources from mobile applications using IAM policies and IAM roles using an identity store and application data synchronization.
- AWS supports external authentication from mobile applications using public identity providers, Facebook, Google, Login with Amazon, and providers that support OpenID Connect, which generates the authentication token that is presented to AWS STS. Verification of the external authentication token by STS results in temporary security credentials being provided for access to the desired AWS resources.

- IAM RDS database authentication is supported by the following database engines:
  - MariaDB 10.6, all minor versions
  - MySQL 8.0, minor version 8.0.23 or higher
  - MySQL 5.7, minor version 5.7.33 or higher
  - PostgreSQL 14, 13, 12, and 11, all minor versions
  - PostgreSQL 10, minor version 10.6 or higher
  - PostgreSQL 9.6, minor version 9.6.11 or higher
  - PostgreSQL 9.5, minor version 9.5.15 or higher

**Requesting Access to AWS Resources**

Only after authentication is successful are IAM users allowed to request access to AWS resources. The following IAM components work together when requesting access to AWS resources:

- **Principal:** The principal defines which IAM user or external user with an assigned IAM role has requested access.
- **Operations:** Only after each request is authenticated and authorized are operations and actions to the requested AWS resource approved. Operations are always API calls executed from the AWS Management Console, through AWS Lambda function, a CLI command or script, or an AWS SDK using RESTful calls.
- **Actions:** Actions define the specific task, or tasks, the principal has requested to perform. Actions might be for information (**List** or **Get** requests) or to make changes (such as creating, deleting, or modifying).
- **Resource:** Every AWS resource is identified with a unique Amazon Resource Name (ARN), as shown in **Figure 3-4**.
- **Environmental data:** Environmental data indicates where the request originated (for example, from a specific IP address range) and can provide additional required security information such as the time of day.
- **Resource data:** Resource data provides additional details about the resource being accessed, such as a specific Amazon S3 bucket, Amazon DynamoDB table, or a specific tag attached to the AWS resource being accessed.

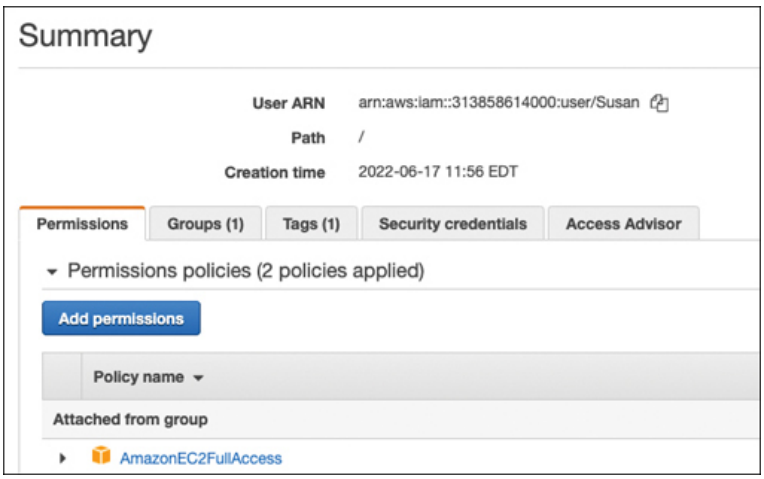


**Figure 3-4** An Amazon Resource Name (ARN)

### The Authorization Process

IAM reviews each request against the attached policies of the principal requesting authorization and determines whether the request will be allowed or denied, as shown in **Figure 3-5**. Note that the principal might also be a member of one or more **IAM groups**, which will increase the number of assigned policies that need to be evaluated before authorization is approved or denied. The evaluation logic of IAM policies follows these strict rules:

- By default, all requests are implicitly denied; there are no implicit permissions. Actions are not allowed without an explicit allow.
- Policies are evaluated for an explicit deny; if found the action is denied.
- An explicit allow overrides the implicit deny, allowing the action to be carried out.
- An explicit deny denies a requested action.

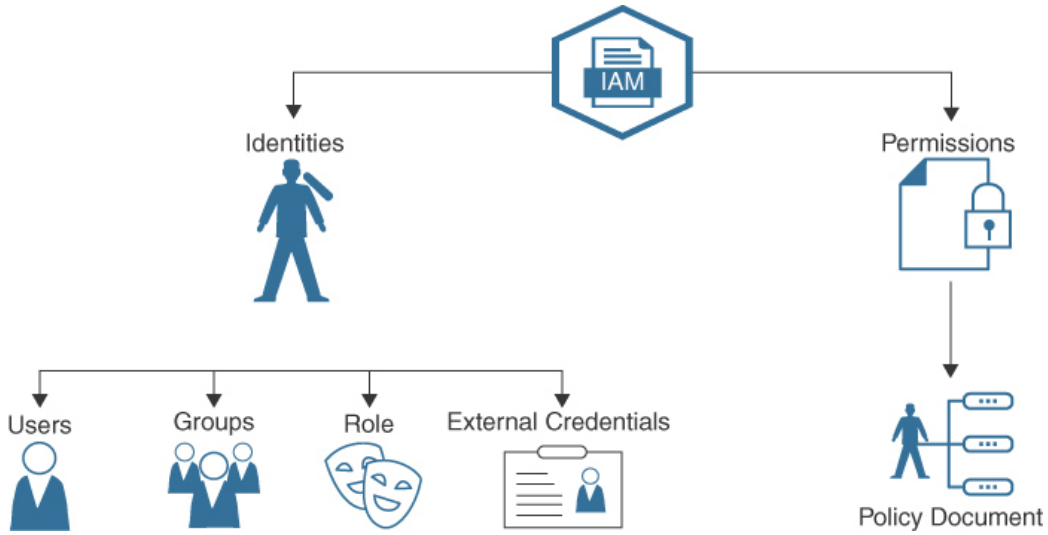


**Figure 3-5** IAM Authorization

When a principal makes a request to access AWS resources, the IAM service confirms the principal is authenticated, signed in, authorized, and

has the necessary permissions.

---

**Note**

You probably experienced an IAM-like authorization process as a teenager. It might have sounded like this: “Hey, Dad, can I go to the movies?” “Nope. All requests are denied.” So, you wait until Mom gets home. “Hey, Mom, can I go to the movies?” “I think so, but let me see whether you cleaned your room. Nope. You can’t go to the movies because you didn’t clean your room.” Mom was sneaky and also used a condition; IAM policies can also use conditions for additional control.

---

The IAM security system reviews the policies assigned and approves or denies the request. As mentioned earlier, IAM implicitly denies everything by default. Requests are authorized only if the specific request is allowed. The following request logic maps to **Figure 3-6**:

1. The evaluation logic follows exact rules that can’t be bent for anyone, not even Jeff Bezos; implicit deny by default. All requests are implicitly denied by default for IAM users.
2. Attached policies are evaluated.
3. An explicit deny denies any request. A default deny can only be overruled and allowed by an explicit allow permission.
4. Each explicit allow permission in the attached policies is allowed.
5. An explicit deny in any policy results in no access.

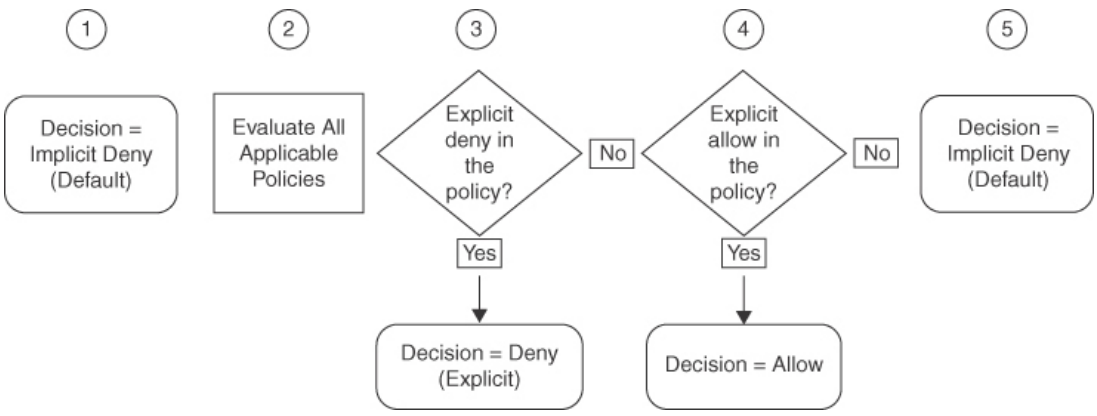


**Figure 3-6** Policy Evaluation Logic

## Actions

Actions are the tasks you can perform on an AWS resource, such as creating, editing, and deleting a resource. There can be many actions for each resource; for example, the EC2 service has more than 400 different actions that can be allowed or denied (see **Figure 3-7**). Once specific actions have been approved, only those actions, which are allowed in the policy, can be performed on the defined resource.

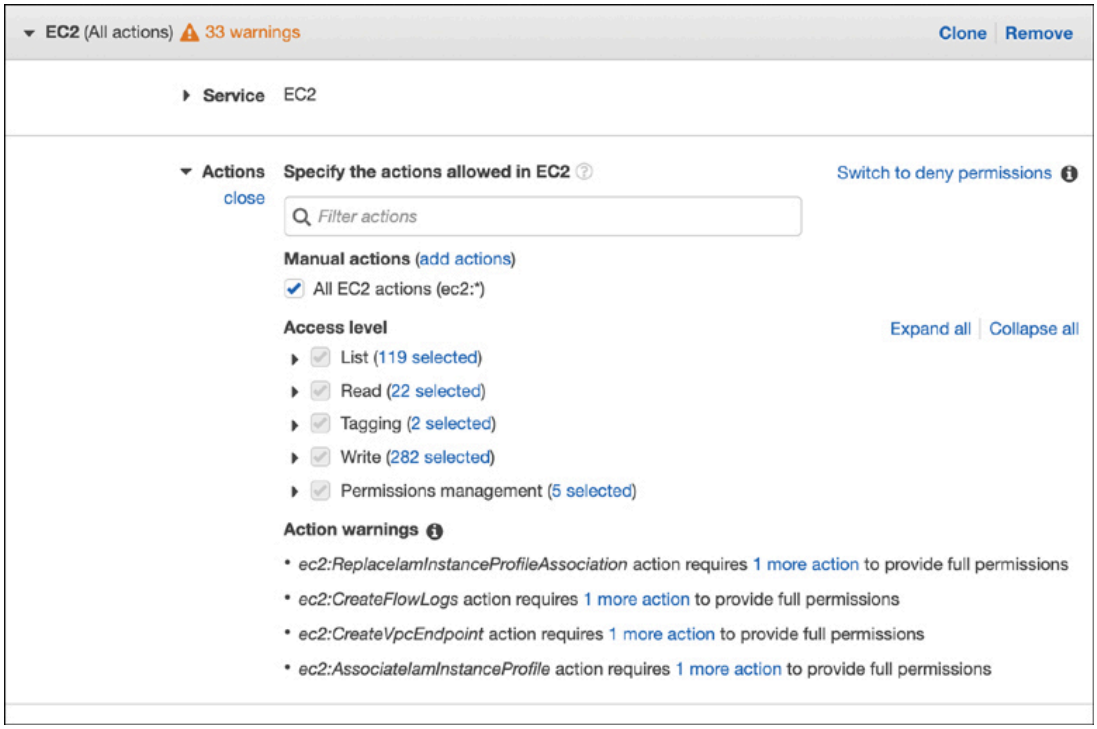


**Figure 3-7** Actions Approved by IAM

Each AWS resource has several actions that can be carried out; the initial access level choices are typically **List**, **Read** (typically **Get** or **List**), and **Write** (typically **Create**, **Put**, **Delete**, **Update**). The type of AWS resource determines the action choices that are available.

## IAM Users and Groups

When you think of the word *account*, you might think specifically of a user account or a group account in the context of an operating system. At AWS, the account that you initially signed up for was designed for organization-wide use, but each AWS account can be used by a single individual, or an organization. It can seem confusing at first.

It might help to think of your AWS account as a complete hosted cloud operating system with security features comparable to the Red Hat Enterprise Linux operating system or Microsoft Active Directory Domain Services.

Many organizations use many AWS accounts—perhaps one or more per developer. At AWS, all cloud services are available per AWS account—subject to the permissions and policies assigned to the authenticating IAM user accounts and roles.

Within each AWS account, IAM user identities or IAM roles are created for these requirements:

- An administrator who needs access to the AWS Management Console.
- An administrator or a developer who needs access to the AWS APIs using the AWS Management Console, and using the AWS CLI command-line interface typing single commands or running scripts, or development of applications using AWS SDKs, such as JavaScript or .NET.

---

**Note**

All companies need to consider the number of administrator and developer user accounts that need to be created and the number of AWS accounts that need to be managed. The best practice is to create roles for external access (access to other AWS accounts or federated access) as much as possible. Roles are explained later in this chapter.

---

**The Root User**

Every AWS account has an initial root user per AWS account created when each AWS account is first provisioned. The root user is the owner of the AWS account, and root credentials are the email address and password provided during the initial creation of each AWS account.

The first time you logged in to a new AWS account, you used the root account credentials to authenticate; after all, there weren't any other administrator accounts available. Perhaps you're still using your root account credentials as your daily administrative account; however, doing so is not recommended, as the root user is not an IAM user controlled by IAM security. The root user has unrestricted access to all resources in the AWS account. Each root account has specific tasks that only the root account can perform. Think of the root account as a special administrator account that should only be used to perform specific tasks, such as billing, changing the AWS support plan, or reviewing tax invoices. The root user is not meant for daily administrative duties. If the root account is the only admin account available in your AWS account, you need to create several

IAM users as administrators to properly safeguard your AWS account resources.

Here's a quick way to check if you're using an AWS account root account: What are the security credentials you use to log in to the AWS account? If you use an email address (see **Figure 3-8**), you are accessing this AWS account as the root user. Now think about how many other administrators could potentially be using the same root account login; each of these administrators could potentially delete everything in the associated AWS account when using the root user account logon. There's no way to disable root account actions because there are no IAM controls on the root account. And no controls can be added.

Key Topic

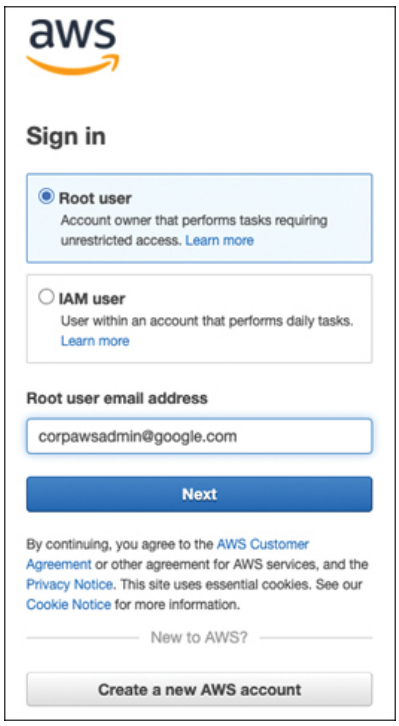


**Figure 3-8** Root User Logon

Why would AWS create an initial user account that has unlimited power? The first account in any operating system must have the most power; think of an Active Directory Domain Services enterprise administrator or the root user for the Linux operating system. As in any other operating system, the root credentials need to be protected. AWS will alert you to create additional IAM users to protect your AWS account.

The following tasks can only be carried out in each AWS account when authenticated as the AWS root user:

- Modifying the root user account details, including changing the password of the root account
- Closing an AWS account
- Changing your AWS support plan from free to Developer, Small Business, or Enterprise
- Enabling billing for the account or changing your payment options or billing information
- Creating a CloudFront key pair
- Enabling MFA on an S3 bucket in your AWS account
- Requesting permission to perform a penetration test
- Restoring IAM user permissions that have been revoked

---

> **Note**
>
> After you sign in for the first time using the root user for your AWS account, the best practice is to create an IAM user for administration duties, add the required administrative policies and privileges to your new IAM user account, and stop using the root account unless it is necessary to carry out a root administrative task.

---

**The IAM User**

An IAM user is an IAM security principal that can be used to access the following interfaces:

- Every IAM user with assigned username and password credentials can access AWS resources using the AWS Management Console.
- An IAM user with assigned username and password credentials and an active access key (access key ID and secret access key) is allowed both AWS Management Console access *and* programmatic access from the AWS CLI. Script or CLI commands will not execute until the IAM user account's access ID and secret access keys are validated.

There are two ways to identify an IAM user:

- The most common way is the name of the user account listed in the IAM dashboard. This username also shows up in each IAM group's list of associated IAM users.
- An Amazon Resource Name (ARN) uniquely identifies each IAM user across all AWS user accounts. Every resource that is created at AWS

also has a unique ARN. For example, if you create a resource policy to control access to an S3 bucket, you will need to specify the user's account ARN that can access the bucket in the following format: *arn:aws:iam::account ID:user/mark*.

### Creating an IAM User

The easiest way to start creating IAM users is to use the IAM dashboard and click Add Users, which opens the dialog box shown in **Figure 3-9**.

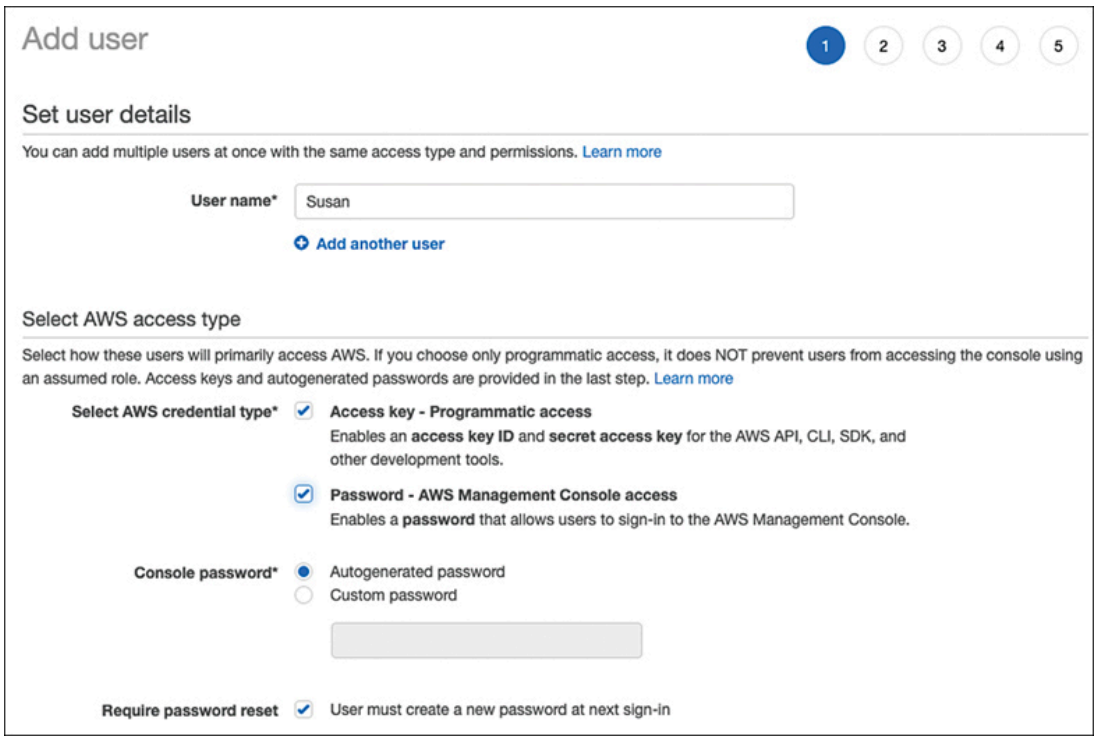


**Figure 3-9** Creating an IAM User

The first decision you must make is the type of access you want to allow your new IAM user to have:

- **Password – AWS Management Console access:** With this type of access, users enter a username and password to authenticate. If console access is all that is required, access keys (an access key ID and secret access key) are not required.
- **Access key – Programmatic access:** This type of access also allows working from the command prompt using the AWS CLI or AWS SDK. Checking both boxes allows both types of access.

---

**Note**

If you're taking over an existing AWS environment, you might find that IAM users have access keys assigned to their accounts, but they don't actually carry out programmatic tasks. If this is

the case, the current access keys can be deleted. In the future, if you decide that access keys are required, they can be added. It is a best practice to remove the root account access keys to make it impossible to run scripts and automation when logged in as a root user.

---

### IAM User Access Keys

Each user account can have two access keys: the access key ID and a secret access key. As discussed earlier in this chapter, access keys are also required when using the AWS CLI (see **Figure 3-10**), when running scripts, when running PowerShell scripts, or when calling AWS APIs directly or through an application.

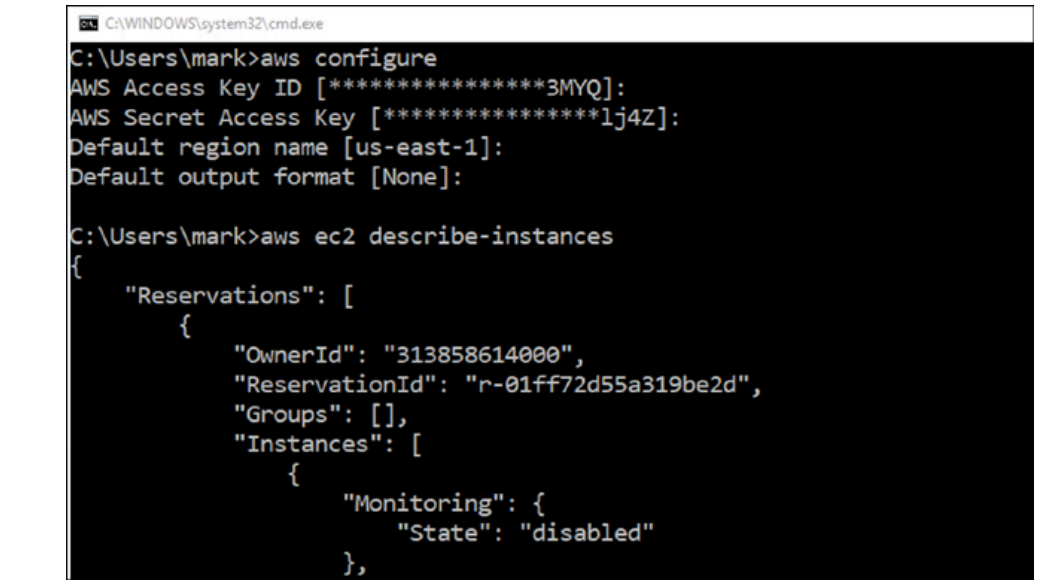


**Figure 3-10** Access Keys Required for CLI Operation

Once an IAM user account has been created successfully, you can download a copy of the access keys (access key ID and secret access key). This option is a one-shot deal: If you don't download a copy of the secret access key at the completion of the user account creation process, you cannot view the assigned secret access key again. However, a new set of access keys for an already created IAM user (access ID and secret access key) can be requested.

There are three options available when creating a new IAM user using the IAM dashboard, as shown in **Figure 3-11**:

- Add the user to an existing IAM group
- Copy permissions from existing IAM users to the user being created
- Attach existing policies directly to the new IAM user

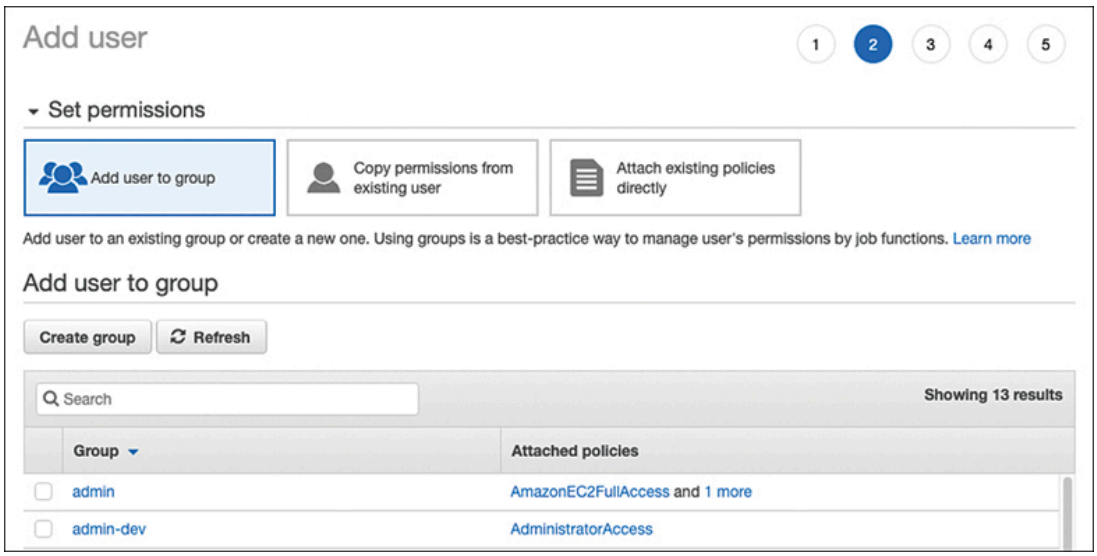


**Figure 3-11** IAM User Account Creation Options

Creating a new IAM user without adding additional permissions or groups creates an extremely limited IAM user as there are no security policies assigned by default. One best practice to consider is to add the new IAM user to an existing IAM group that has the permissions needed for the IAM user account's required duties. Even if you are creating an IAM user for a specific AWS task that just this IAM user will carry out, you might want to think about adding this person to an IAM group if there's a possibility of multiple IAM users carrying out the same task in the future.

---

**Note**

What each IAM user can and can't do at AWS is defined either by an explicit allow permission to carry out a task against AWS resources, or by explicit deny permissions that prohibit the user from being able to carry out a task. Note that an explicit deny for a specific task in any policy assigned to an IAM user account overrides any allow permissions defined in any other attached IAM policies.

---

### IAM Groups

An IAM group is a collection of IAM users. IAM groups are useful for delegating security policies to a specific group of IAM users. Attaching IAM groups to IAM user accounts makes assigning permissions much easier than having to modify each individual IAM user account. Each IAM user

listed in an IAM group has their own authentication credentials and possible memberships in additional IAM groups. Each IAM group that IAM users are members of are assigned their IAM group permissions only after they have successfully authenticated to AWS. The characteristics of IAM groups are as follows:

- Each IAM group can contain multiple IAM users from the same AWS account.
- IAM users can belong to multiple IAM groups in the same AWS account.
- IAM groups can't be nested.
- IAM groups can only contain IAM users and not any additional IAM groups.
- There are initial quotas on the number of IAM groups you can have in each AWS account, and there is a quota that defines how many IAM groups an IAM user can be in. An IAM user can be a member of 10 IAM groups, and the maximum number of IAM users that can be created in a single AWS account is 5000.

### Signing In as an IAM User

After IAM users have been created, to make it easier for your IAM users to sign in to the IAM console, you can create a custom URL that contains your AWS account ID, as shown in **Figure 3-12**.

---

**Note**

When creating and maintaining users, groups, and roles, you can manage IAM by using a third-party identity management product such as ForgeRock (**https://www.forgerock.com**), Okta (**https://www.okta.com**), or OneLogin (**https://www.onelogin.com**).

---

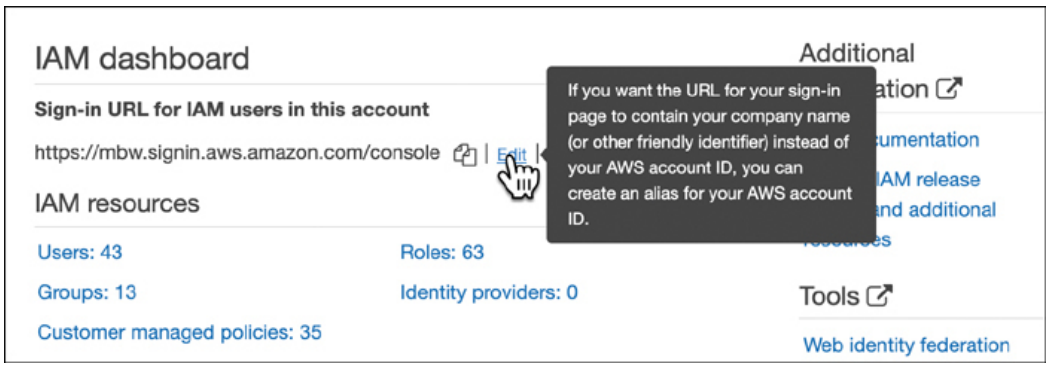


**Figure 3-12** Using Custom URL for IAM Users

### IAM Account Details

Each IAM user account displayed in the IAM console shows some useful information, including the IAM groups that the IAM user belongs to, the age of the access keys assigned to the IAM user account, the age of the current password, the last activity of the IAM account, and whether MFA has been enabled. Selecting an IAM user account in the console, you can see several additional account options, including the ARN of the account and information on the following tabs (see **Figure 3-13**):

- **Permissions:** This tab lists the applied permissions policies and the policy types.
- **Groups:** This tab lists the policies attached due to group membership.
- **Tags:** This tab lists key/value pairs (up to 50) that can be added for additional information.
- **Security Credentials:** This tab enables you to manage credential-related parameters such as the following:
  - The console password of the IAM user
  - The assigned MFA device (which can be a virtual or hardware device)
  - Signing certificates
  - Access keys
- **Access Advisor:** This tab lists the service permissions that have been granted to the IAM user and when the AWS services were last accessed within the calendar year.

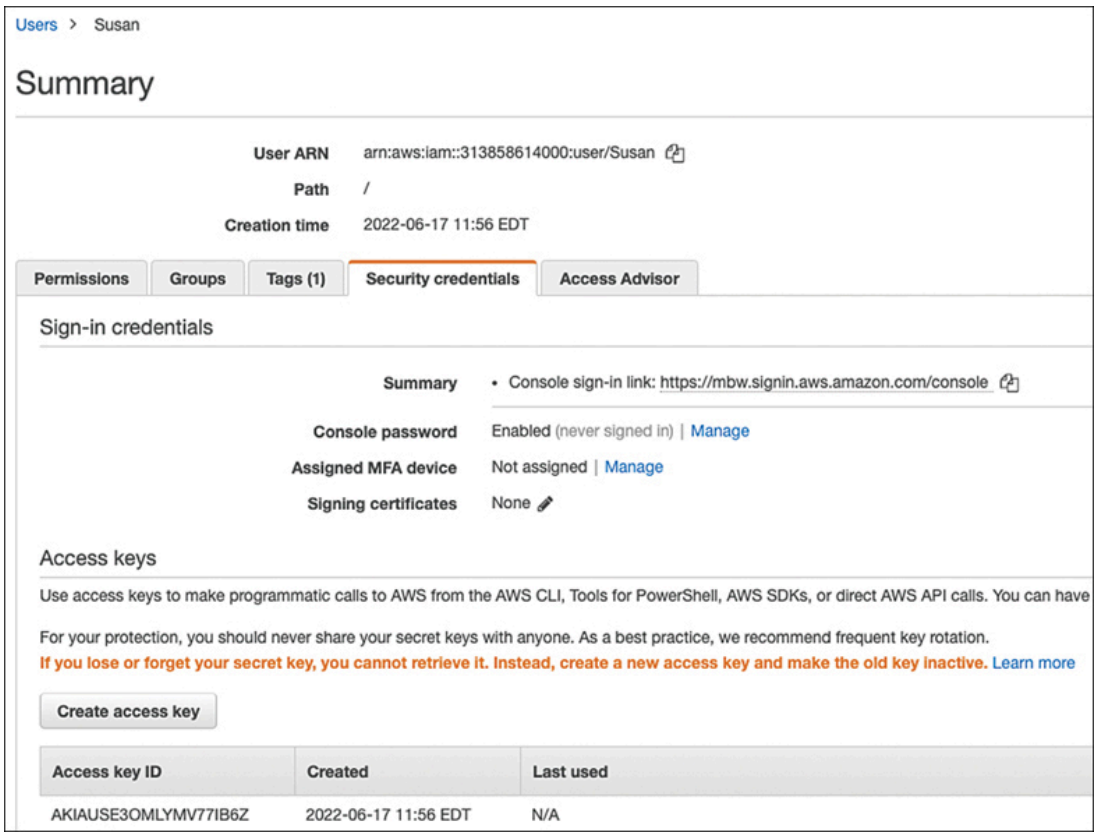


**Figure 3-13** User Account Summary Information

### Creating a Password Policy

**Password policies** can be defined by selecting Account Settings in the IAM console. After password policies are defined, they control all IAM user accounts created in the AWS account. Password options include password complexity, password expiration, password reuse, and whether IAM users can change their own passwords (see **Figure 3-14**).

Best practice is to review your corporate policy for passwords and consider whether more stringent rules need to be followed for working in the AWS cloud. If rules around password policy need to be tightened, the rules for the current on-premises password policy and the password policy defined in the AWS cloud should be analyzed and unified.

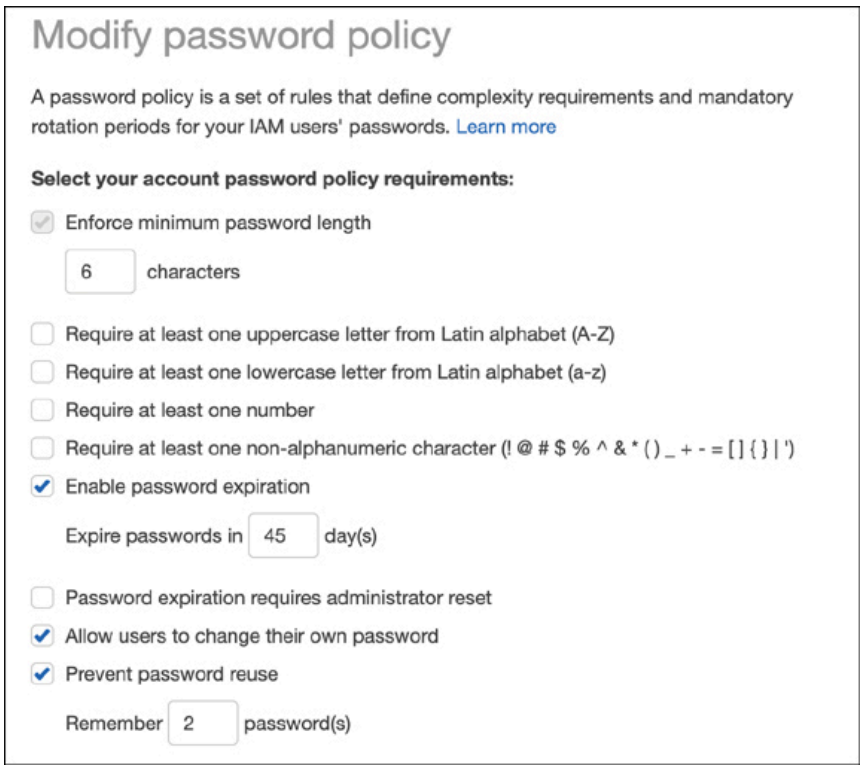


**Figure 3-14** Password Policy Options

### Rotating Access Keys

After an IAM user account has been created with access keys, the access keys are not changed unless they are manually rotated, or an automated process to perform the key rotation process is used such as a script or a custom Lambda function. Best practice is to rotate a user's access keys, preferably at the same time the IAM user password is changed, to maintain a higher level of security and avoid issues that can arise from compromised access keys. The access keys currently assigned to the IAM user can be viewed in the properties of the IAM user account on the Security Credentials tab. When a request is received to create a new access key, an associated secret access key is created along with the new access key ID, as shown in **Figure 3-15**.

The important task of rotating access keys, shown in **Figure 3-16**, should be assigned to a trusted IAM administrator account that will be carrying out the task of key rotation. Note that multiple **Get**, **Create**, **List**, **Update**, and **Delete** actions must be assigned to the selected IAM user in order to rotate access keys successfully.

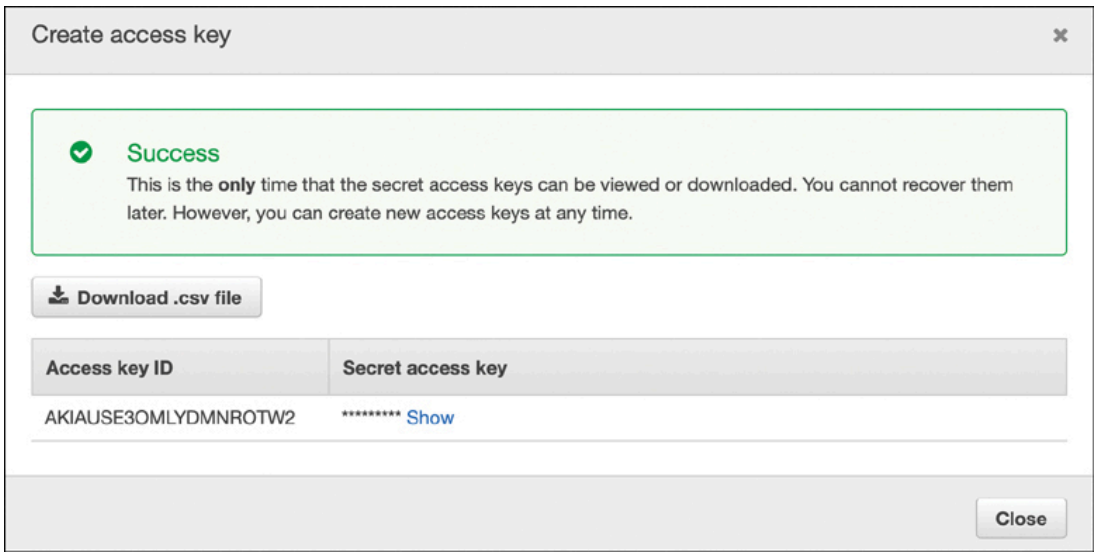


**Figure 3-15** Creating an Additional Access Key Manually

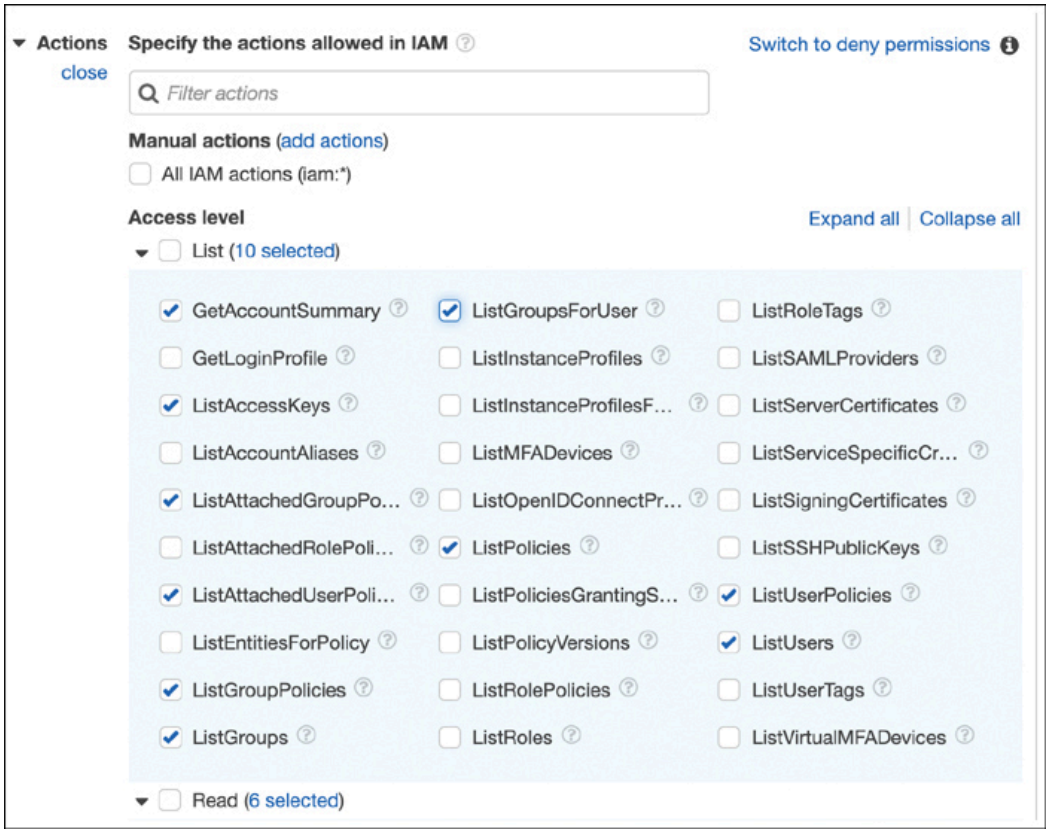


**Figure 3-16** Policy Actions for Rotating Access Keys

## Using Multi-Factor Authentication


Every AWS user account—including the root account and the IAM user account—supports MFA. With MFA enabled, during the process of authenticating to AWS, a user must provide a security code in addition to the username and password credentials provided to access AWS resources, as shown in Figure 3-17.

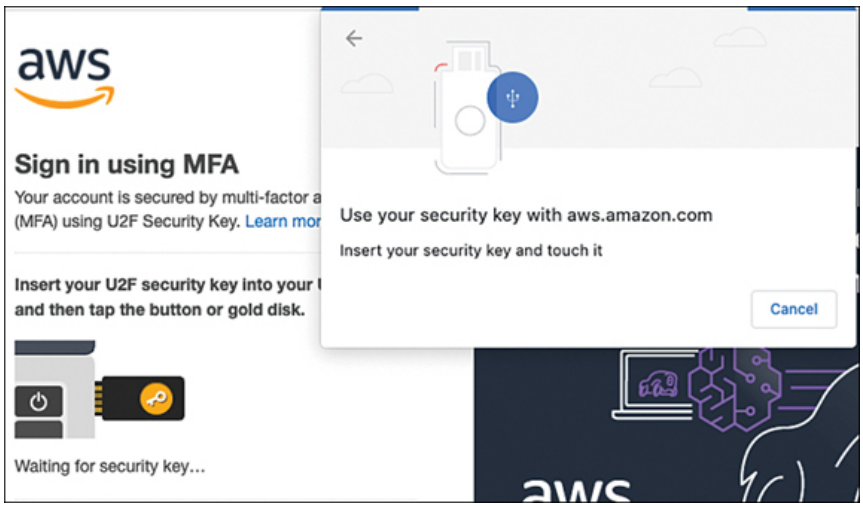


**Figure 3-17** Authenticating with MFA Enabled

There are several options available at AWS for deploying MFA:

- **Virtual MFA device:** A software app such as Google Authenticator or Authy, that typically is installed on the user's phone, can generate the six-digit code to be entered during authentication.
- **U2F security key:** A U2F security key is generated by a USB device that generates a security code when tapped. These types of devices are approved by the Fast Identity Online (FIDO) Alliance. These keys are supported by many industry leaders, including Microsoft, Google, AWS, VMware, and Intel.
- **Hardware MFA device:** A hardware device such as a Thales SafeNet security appliance can also generate an MFA security code. Thales devices can provide end-to-end management of the entire encryption process.

## Creating IAM Policies

Identity and Access Management enables you to use or create a large number of security policies. Identity-based policies use the Identity and Access Service to apply security policies to an identified IAM user, group, or role.

---

**Note**

The other type of security policy is called a resource-based policy. It is assigned to protect storage resources such as S3 buckets. Resource policies were available before the Identity and Access Management security service was introduced.

---

### IAM Policy Types

The actions for controlling AWS services with IAM policies are forever increasing as new features are added frequently to existing and new AWS cloud services. Make sure to check the documentation for each AWS service for the up-to-date choices. This section looks at the policy types that can be attached to IAM identities (users, groups, or roles).

### Identity-Based Policies

Identity-based policies are categorized as permission policies. Each identity-based policy contains permissions for specific actions an IAM user, group, or role can carry out. Polices can allow or deny access, and, optionally, indicate one or more mandatory conditions, must be met before access is allowed to the listed AWS cloud service or services defined in each policy.

There are three identity-based policy types:

- **Managed policies:** Managed policies, which are created and maintained by AWS, are read-only stand-alone identity-based policies that you can select and attach to IAM users, IAM groups, or roles created within each AWS account (see **Figure 3-18**). Listed are some concepts you need to understand when working with managed policies:
  - Managed policies can be attached to and detached from any identity (that is, user, group, or role).
  - A managed policy can be copied and saved as a custom policy.
  - Managed policies cannot be deleted. When you detach a managed policy, it is removed from the selected identity, user, group, or role; however, the managed policy is still available in the library of managed AWS policies for reuse.
  - Custom policies can be attached, detached, and deleted.

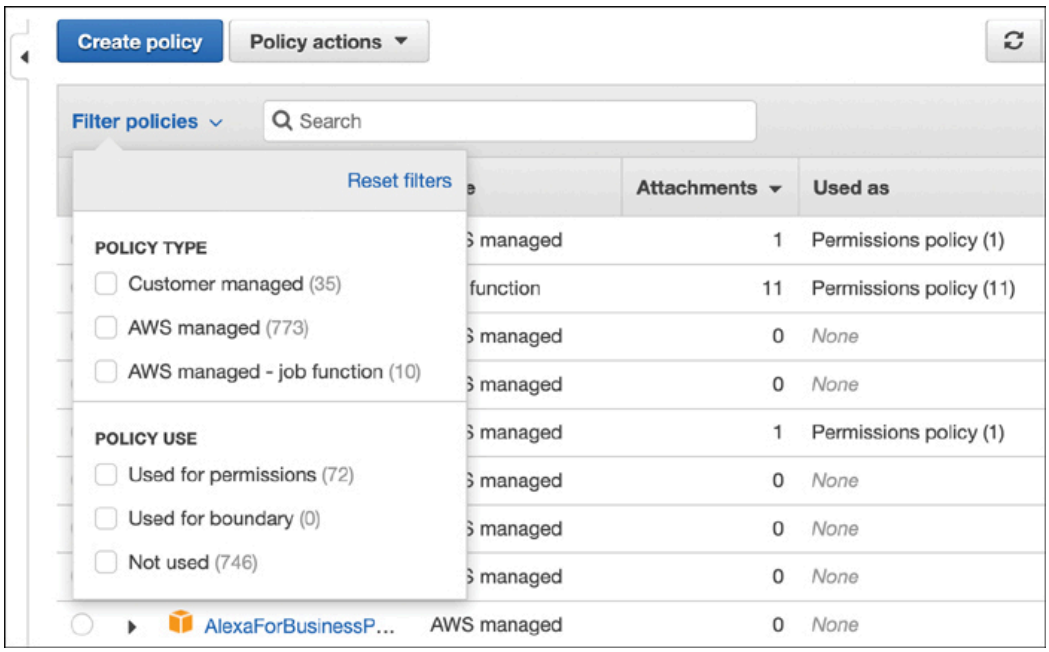


**Figure 3-18** Managed Policies

- **Managed Policies for Job function:** Job function policies, which are also created and managed by AWS, are specialized managed policies based on generic job descriptions (see **Figure 3-19**). Job function policies might at first seem like an excellent idea. However, you need to be careful when assigning job function policies because a job function policy may assign more permissions than you need or wish to assign. For example, the SystemAdministrator job function policy allows the creation and maintenance of resources across many AWS services, including AWS CloudTrail, AWS CloudWatch, AWS CodeCommit, AWS CodeDeploy, AWS Config, AWS Directory Service, Amazon EC2, AWS IAM, AWS Lambda, Amazon Relational Database Service (RDS), Amazon Route 53, AWS Trusted Advisor, and Amazon Virtual Private Cloud (VPC). However, a job function policy can be useful as a starting policy template that once imported as a custom policy enables you to make further modifications to suit your organization's needs. The job function policies that can be selected are Administrator, Billing, Database Administrator, Data Scientist, Developer Power User, Network Administrator, Security Auditor, Support User, System Administrator, and View-Only User.

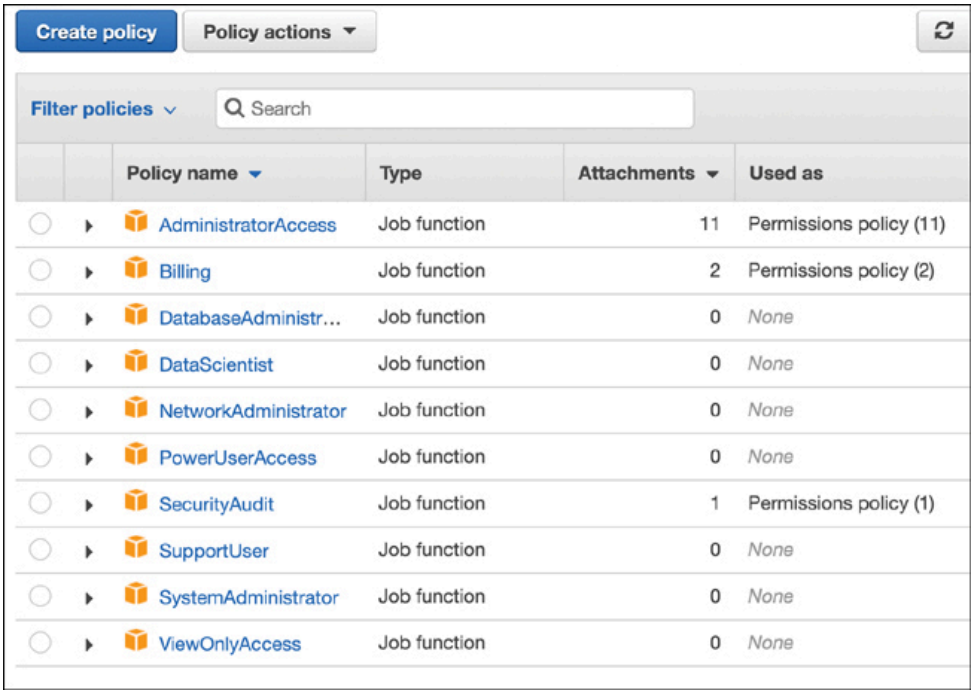


**Figure 3-19** Job Function Policies

- **Custom policies:** You can select any managed policy as a starting template, modify it for your requirements, saving it as a custom policy in your AWS account. You can also elect to start with a blank page when creating a custom policy document and create the entire policy from scratch using the IAM dashboard. Each custom policy created is managed and maintained by each organization.

### Resource-Based Policies


As previously discussed, identity-based policies are attached to an IAM user, group, or role defining what actions each attached identity is allowed, or not allowed to do. Resource-based policies are a little different in functionality because they are attached directly to AWS resources and are not created using the AWS Identity Access Management service. Resource-based policies are supported by several AWS storage services; the most common example is an Amazon S3 bucket, but there are other older AWS cloud services that support resource-based policies, including Amazon S3 Glacier vaults, Amazon Simple Notification Service (SNS), Amazon Simple Queue Service (SQS), and AWS Lambda functions. Because resource policies are attached directly to the AWS resource, each policy needs to define the access rules for the AWS resource and the IAM user, group, or AWS account that will access the resource. Resource-based policies are similar in functionality to IAM *inline policies* due to the direct attaching of the resource policy to the AWS resource; if a resource is

deleted, the resource policy is unattached and discarded. Resource-based policies are always a custom creation; AWS does not create any managed resource-based policies. Inline policies are discussed later in this chapter.

An IAM user can be assigned both a managed IAM identity policy and a resource-based policy for accessing the same AWS resource (see **Figure 3-20**):

- IAM User Mark has an identity-based policy that allows him to list and read from S3 Bucket A.
- The resource—in this case, the S3 bucket—has an attached resource-based policy that identifies that Mark can list and write on S3 Bucket A.
- S3 Bucket C has an attached resource-based policy that denies access to Mark. IAM User Julian also has a combination of identity- and resource-based policies.
- IAM User Jan has no managed policies assigned.
- Jan has access to S3 Bucket C because she is specifically listed in the resource policy using her IAM User ARN.

Key Topic

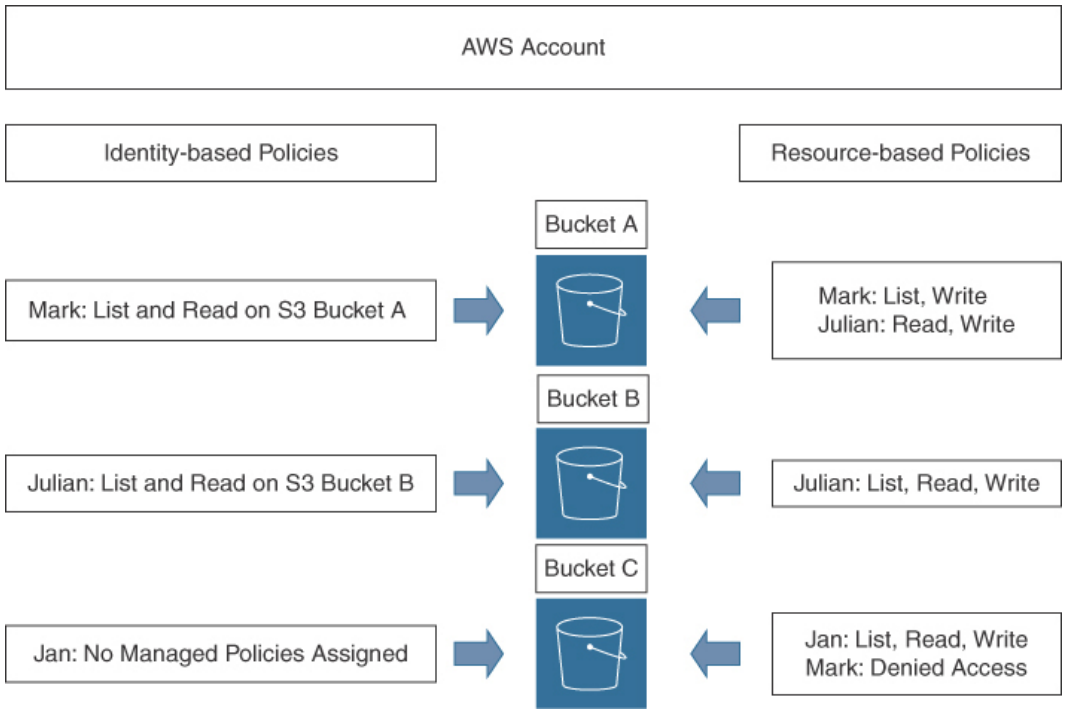


**Figure 3-20** Identity and Resource Policies Working Together

---

**Note**

Amazon S3 bucket policies are resource policies.

---

### Inline Policies

Another method of attaching IAM policies is through the process of what is called an inline or directly attached policy, as shown in Figure 3-21. An IAM policy that is attached inline helps you maintain a strict one-to-one relationship between the attached policy and the entity the policy is attached to. When the entity is deleted, the attached policies are discarded. In comparison, using a managed policy allows you to apply the policy to multiple IAM users and groups.

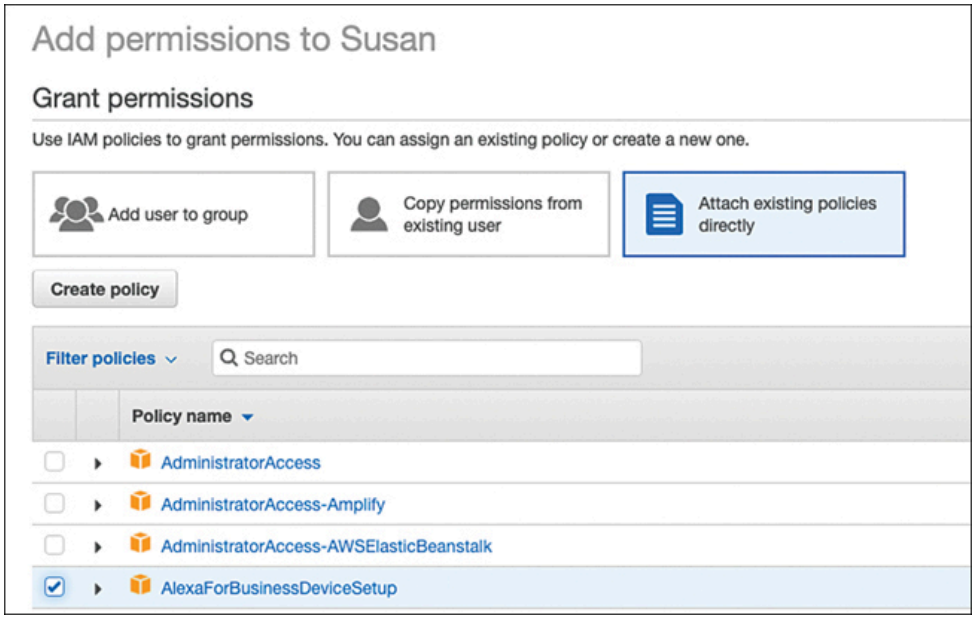


**Figure 3-21** Attaching Existing Policies Directly

For example, a specific user with high security clearance within your organization has been assigned the task of managing AWS CloudHSM, a security service that uses single-tenant hardware security modules (HSMs) for storing your organization's symmetric and asymmetric keys. You've decided to manage the security for this service by using inline policies that are attached to just one trusted administrator to ensure that only this person can carry out the specific tasks. Perhaps you have two security administrators, and you use inline policies to ensure that the policies are only assigned to these two individuals. You could use an IAM group but you don't want to make a mistake and accidentally add an additional IAM user to the existing group and weaken your security. If the administrator's IAM user accounts are deleted, the inline policies are discarded as well.

---

**Note**

IAM roles (which are discussed later in this chapter in the section "IAM Roles") are also attached directly to the IAM user or

federated user.

---

### IAM Policy Creation

Each IAM policy is crafted in what is called a lightweight data interchange format, JavaScript Object Notation (JSON) format. You can create and view any existing IAM policies by using the IAM dashboard or by using the AWS CLI and using the commands **create-policy** or **list-policies**. If you are just starting with AWS, it's probably best to start with the IAM dashboard, where you can easily view the IAM users and groups, policies, and roles. For crafting IAM policies using the AWS CLI, the AWS CLI command reference for Identity and Access management can be found here: **https://awscli.amazonaws.com/v2/documentation/api/latest/reference/iam/index.h**

Each IAM policy can define a single permission statement or multiple permission statements. When you create custom policies, it is important to keep them as simple as possible to start with; don't mix AWS resource types in a single policy just because you can. It's a good idea to separate custom policies by AWS resource type for easier deployment and troubleshooting. You can create IAM policies by using several methods:

- Create IAM policies by using the visual editor in the IAM console.
- Create IAM policies by using the JSON editor in the IAM console (see **Figure 3-22**).
- Create and add IAM policies by using standard copy and paste techniques to import policy settings in JSON format into your JSON editor.
- Create IAM policies by using a third-party IAM tool that has been installed and properly configured. After authenticating to AWS using a recognized IAM user with valid access keys and appropriate administrative permissions, you can create IAM users, groups, and roles with third-party tools, such as OneLogin or Ping Identity, instead of using the IAM console.

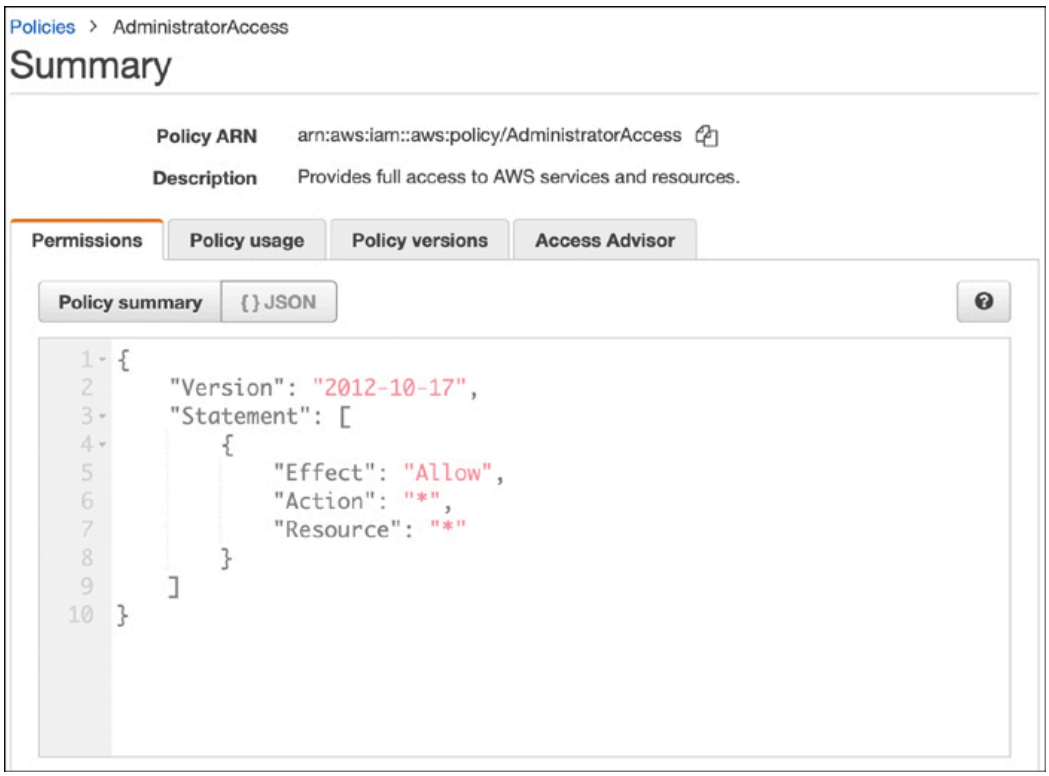


**Figure 3-22** The JSON Editor


### Policy Elements

Each IAM policy contains mandatory and optional elements that you need to understand and be familiar with:

- **Version:** (Mandatory) This element is the version of the policy language that the policy is using (see **Figure 3-23**). The latest policy language version is 2012-10-17; the date/time version number is added automatically to each policy document when you are manually creating a policy document using the IAM console. Add the latest version to all custom policies if they are created outside the IAM console to ensure that any new AWS features you are referencing in the custom policy are supported. If no version number is listed, the oldest IAM version number is used, which can potentially cause problems. For example, if you were using tags to determine access, or permission boundaries in a custom policy with no listed version number, these newer features would not work without the latest version number present at the top of the policy document.

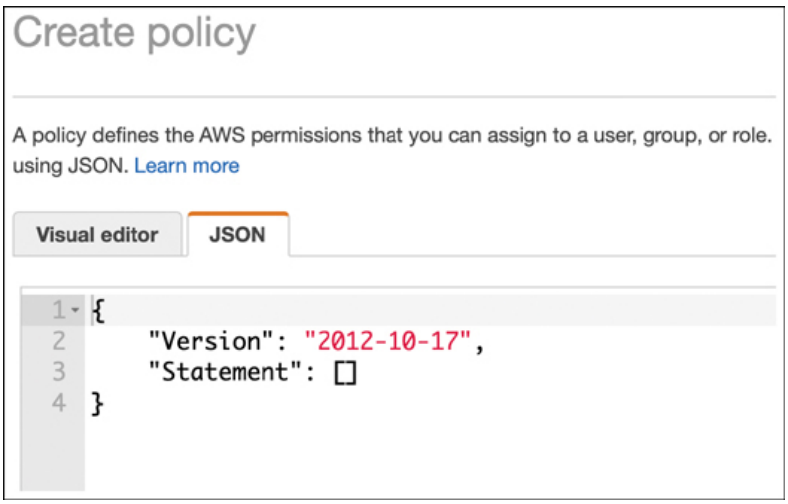


**Figure 3-23** Version Information

- **Statement:** (Mandatory) Each IAM policy has at least a single statement; multiple statements are allowed in a policy. When beginning to craft custom policies, it might be cleanest or simplest to limit each policy document to a single policy statement.
- **Sid:** (Optional) This element is a unique ID statement for additional identification purposes.
- **Effect:** (Mandatory) The effect of any listed action is Allow or Deny.
- **Action:** (Mandatory) Each action lists the API call(s) that is allowed or denied.
- **Principal:** (Optional) The account, user, role, or federated user of the policy allows or denies access to a resource.
- **Resource:** (Mandatory) This element identifies the AWS resource that the actions in the statement apply to.
- **Condition:** (Optional) This element defines the absolute circumstances that must be met for the policy to be applied.

### Reading a Simple JSON Policy

You need to follow a number of syntax and grammatical rules when creating custom IAM policies. One missing brace ({) or missed comma or colon can cause lots of pain when you're troubleshooting an IAM policy. These are the rules:

- Text values—that is, string values—are always encased in double quotes.
- A string value is followed by a colon.
- The data parts in a policy are defined as name/value pairs.
- The name and the value are separated with a colon (for example, **"Effect": "Allow"**).

- When data in a policy has multiple name/value pairs, the name/value pairs are separated using commas.
- Braces { } contain objects.
- Each object can hold multiple name/value pairs.
- If square brackets are used, there are multiple name/value pairs, separated by commas.

Let's look at a simple IAM policy example in **Example 3-1** and explore its construction. Note that the numbers shown are just for identification purposes.

**Example 3-1** IAM Policy

Click here to view code image

```
1.{
2. "Version": "2012-10-17",
3. "Statement": {
4. "Effect": "Allow",
5. "Action": "s3:ListBucket",
6. "Resource": "arn:aws:s3:::graphic_bucket"
7. }
8. }
```

Each policy starts with a left brace that defines the start of the policy statement block. A curly right brace denotes the end of the policy statement block. In **Example 3-1**, line 1 and line 8 start and end the policy statement block.

Line 2 shows the current version of IAM policies; both **Version** and the version number are in quotation marks because the values within the quotes are string values. You can treat the version line in an IAM policy as a mandatory policy element. The version number is a name/value pair, and the name and the value are separated by a colon. Because there are multiple name/value pairs in this policy, there is a comma at the end of each line that contains a name/value pair (that is, lines 2, 4, and 5).

The first statement in the policy, line 3, is defined by **"Statement"** (note the quotation marks) followed by a colon (**:**) and another inner left brace (**{**) that denotes the start of the statement block, which includes **Effect**, **Action**, and **Resource**:

- Line 4, **“Effect”** (note the quotation marks), followed by a colon (**:**), is set to the value **“Allow”** (also in quotation marks). **“Effect”** can be set to either **Allow** or **Deny**.
- Line 5, **“Action”** in this policy, is set to allow the listing of an S3 bucket.
- Line 6, **“Resource”**, specifies that the resource being controlled by this policy is the S3 bucket **graphic_bucket**. The resource references the ARN—the unique Amazon name that is assigned to each resource at creation. Resource lines in policies don’t have commas because a resource is a name/resource listing, not a name/value pair.

Line 7, the right curly brace (**}**), indicates that the statement block is complete. The final right curly bracket that starts line 8 indicates that the policy statement block is complete.

**Policy Actions**

When creating custom policies, you will typically have to provide several actions for the user to be able to carry out the required tasks. Take, for example, creating a policy for an administrator to be able to create, change, or remove their IAM user account password. The actions that need to be listed in the policy must include the following:

- **CreateLoginProfile:** The user needs to be able to create a login profile.
- **DeleteLoginProfile:** The user must be able to delete their login profile if they want to make changes.
- **GetLoginProfile:** The user has to be able to access the login profile.
- **UpdateLoginProfile:** After making changes, the user has to be able to update their login information.

For an IAM user to be able to perform administration tasks for a group of IAM users, the additional actions required include creating users, deleting users, listing users and groups, removing policies, and renaming or changing information. To be able to make changes to an AWS resource, you must be able to modify and delete. The statement in **Example 3-2** provides the details for this policy.


**Example 3-2** IAM Policy for Performing Administrative Tasks

Click here to view code image

```
"Statement": [
{
"Sid": "AllowUsersToPerformUserActions",
"Effect": "Allow",
"Action": [
"iam:ListPolicies",
"iam:GetPolicy",
"iam:UpdateUser",
"iam:AttachUserPolicy",
"iam:ListEntitiesForPolicy",
"iam:DeleteUserPolicy",
"iam:DeleteUser",
"iam:ListUserPolicies",
"iam:CreateUser",
"iam:RemoveUserFromGroup",
"iam:AddUserToGroup",
"iam:GetUserPolicy",
"iam:ListGroupsForUser",

"iam:PutUserPolicy",
"iam:ListAttachedUserPolicies",
"iam:ListUsers",
"iam:GetUser",
"iam:DetachUserPolicy"
}
],
```


### Additional Policy Control Options

Several policy options give you great power in how you manage security options for IAM users and groups, including permission boundaries, service control policies, access control lists, and session policies.

### Permission Boundaries

Permission boundaries are used to mandate the security policies that can be applied to an IAM user or role.

You can apply a permission boundary policy for both the IAM user and IAM role within a single AWS account. Without a permission boundary

being defined, the applied managed or custom policy defines the maximum permissions that are granted to each particular IAM user or role. Adding a permission boundary provides a level of control by filtering the permissions that can be applied. The IAM user or role can only carry out the actions that are allowed by *both* the assigned identity-based policy and the permission boundary policy. Therefore, the permission settings defined are controlled by a permission boundary policy that establishes the specific listing of permissions that can be applied.

For example, suppose you want administrator Mark to be able to manage Amazon S3 buckets and EC2 instances—and that's all. In this case, you need to create the custom policy shown in **Example 3-3**, which defines the permissions boundary for Mark—namely, that he can fully administrate Amazon S3 buckets and EC2 instances.

**Example 3-3** Mark's Permission Boundary

Click here to view code image

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

Once a permission boundary has been added to Mark's IAM account, as shown in **Figure 3-24**, the only two AWS services that Mark will have full administrative control over are Amazon S3 Buckets and EC2 instances. In the future, an IAM policy is added to Mark's account to enable him to work with AWS CloudTrail and create alerts and alarms. However, when Mark goes to carry out actions using AWS CloudTrail, if the current permission boundary has not been updated, listing that Mark can also use the CloudTrail service, then this action will be denied. The permission

boundary policy settings must match up with the IAM policy settings that are applied to Mark's IAM user account.

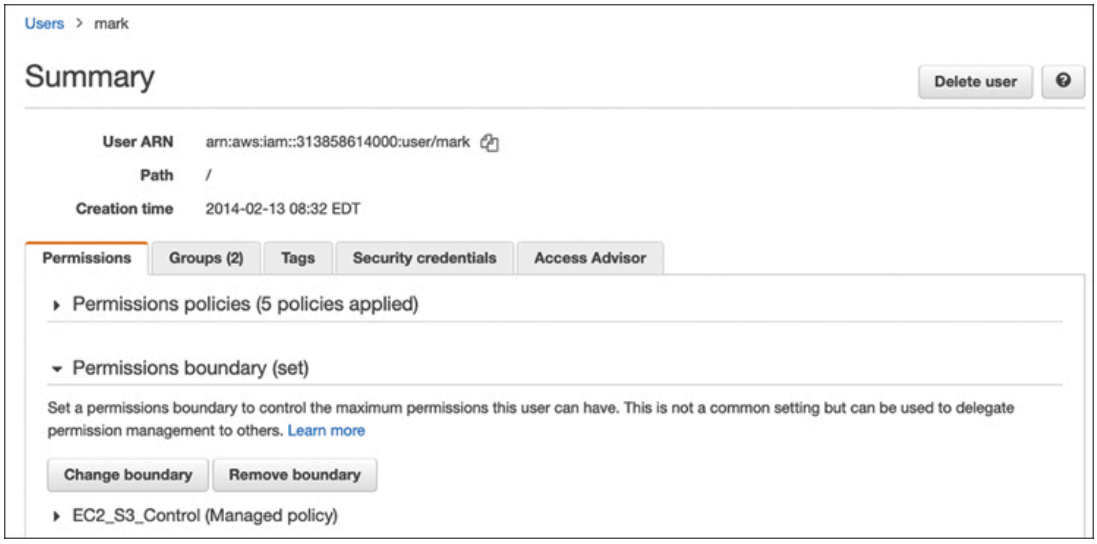


**Figure 3-24** Adding a Permission Boundary to a User Account

The permission boundary shown in **Figure 3-24** could be much more stringent; instead of listing full control, a permission boundary could mandate a specific listing of tasks that Mark could carry out for both Amazon S3 buckets and EC2 instances.


### AWS Organizations Service Control Policies

AWS Organizations enables organizations to manage security settings and services across AWS accounts that are grouped together in a tree formation. (More details on AWS Organizations are discussed later in this chapter.) One of the security features of AWS Organizations is a service control policy (SCP), which provides a permission boundary located at the root of the tree controlling all AWS account members, or to specific OUs containing AWS accounts in the AWS Organization tree. The SCP and the entity being controlled must have matching permissions for the desired permissions to be allowed (see **Figure 3-25**). Once an SCP has been enabled, permissions are allowed only if the IAM policy and the SCP list the identical permissions in both policies. The types of permission policies that can be controlled by an SCP are identity-based policies for IAM users, roles, the root user in any AWS account, and resource-based policies.

---

**Note**

Service control policies do not affect service-linked roles that delegate the permissions assigned to each AWS service to carry out their assigned tasks.

---

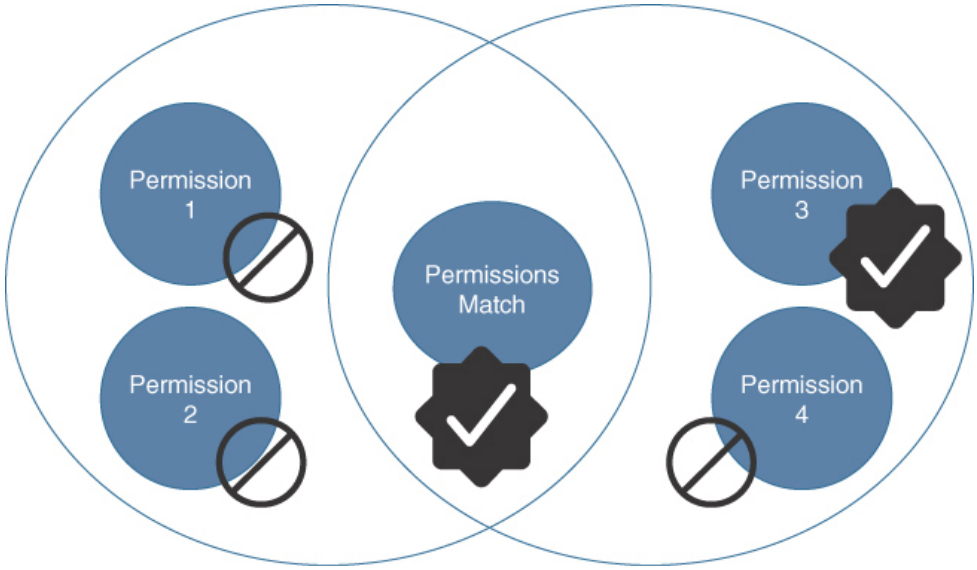


**Figure 3-25** Effective Permissions Using a Service Control Policy

### Access Control Lists

Access control lists (ACLs) are present for defining simple permission controls on objects in Amazon S3 buckets for cross-account permission access only between separate AWS accounts. ACLs cannot be used to grant permissions to entities in the same AWS account. ACLs are only present because of backward compatibility; it's a much better idea to use IAM roles to control cross-account access. Amazon recommends that ACLs not be used for applying security to S3 bucket contents. Instead use the Amazon S3 Object Ownership setting **Bucket owner enforced** to disable all of the ACLs associated with a bucket. When this bucket-level setting is applied, all of the objects in the bucket become owned by the AWS account that created the bucket and ACLs can no longer be used to grant access.

### Session Policies

Session policies are another version of a permission boundary to help limit what permissions can be assigned to federated users or IAM users assigned roles (see **Figure 3-26**). Developers can create session policies when IAM roles are used to access an application. When session policies are deployed, the effective permissions for the session are either the ones that are granted by the resource-based policy settings or the identity-based policy settings that match the session policy permission settings.

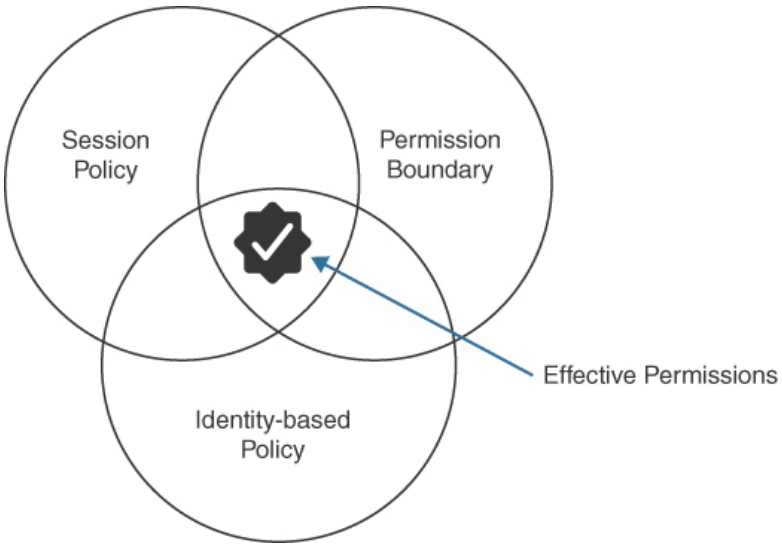


**Figure 3-26** Session Policies

### Reviewing Policy Permissions

For troubleshooting purposes, it may be necessary to review the assigned access levels, the required AWS resources, and any additional conditions that have been allowed or denied within each IAM policy. Thankfully, AWS provides these details in graphic *policy summary tables*, as shown in Figure 3-27, which make it easier to troubleshoot or analyze what an IAM user, group, or role combined with a select IAM policy can do. There are policy summaries on both IAM users and roles for all attached policies. View a policy summary by selecting the individual policy; on its summary page click Policy Summary. Information is displayed for the different types of policies: custom and AWS-managed policies and AWS-managed job function policies.

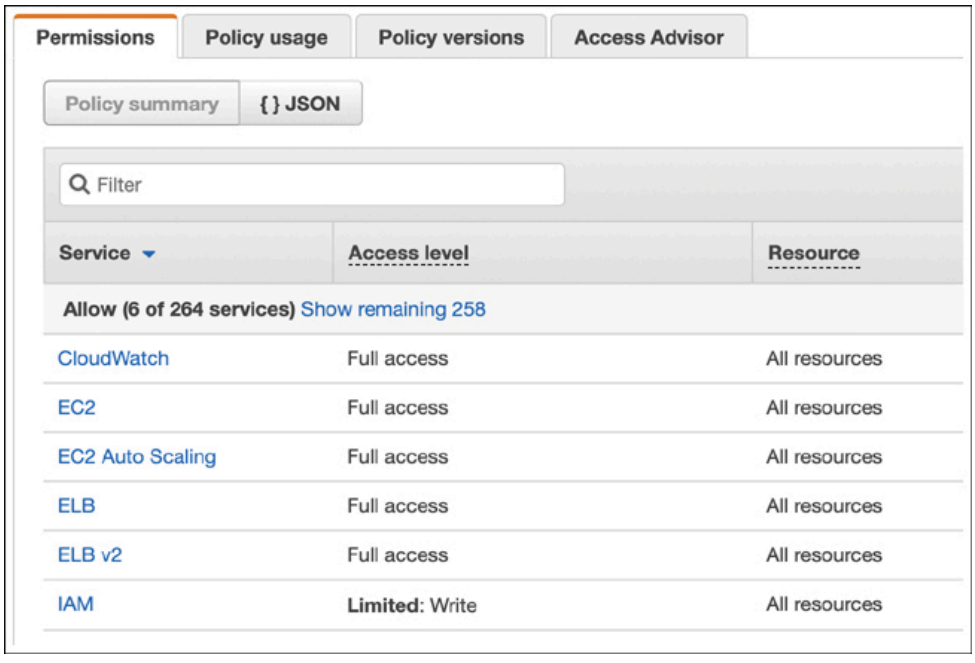


**Figure 3-27** Policy Summary Tables

Policy permissions information is contained in three tables:

- **Policy Summary (Services):** Information is grouped into explicit deny, allow, and uncategorized services when IAM can't figure out the service name due to a typo or when a custom third-party service is in use that has not been defined properly. Recognized services are listed based on whether the policy allows or explicitly denies the use of the service.
- **Service Summary (Actions):** Information displayed includes a list of the actions and permissions (for example, list, read, write) that have been defined in the policy for a particular service.
- **Action Summary (Resources):** Information includes a list of resources and the conditions that control each action. Details include the resources, the region where the resources have been defined, and what IAM accounts the actions are associated with.

### IAM Policy Versions


After you've created an IAM policy, in the future you may want to make additions or deletions. Regardless of whether a policy is a custom policy that you have created or an AWS-managed policy, every time an IAM policy is updated, a new version of the policy is created.

AWS stores up to five versions of each IAM policy. To define the default version of an IAM policy to be used, after selecting the policy, select the Policy versions tab, and from the displayed versions, select the version of the policy that you want to define as the current version to be used. From this point forward, the selected version of the policy becomes version enforced, as shown in **Figure 3-28**. If you want to make changes later, you can change the current version of the policy to another version of the policy.

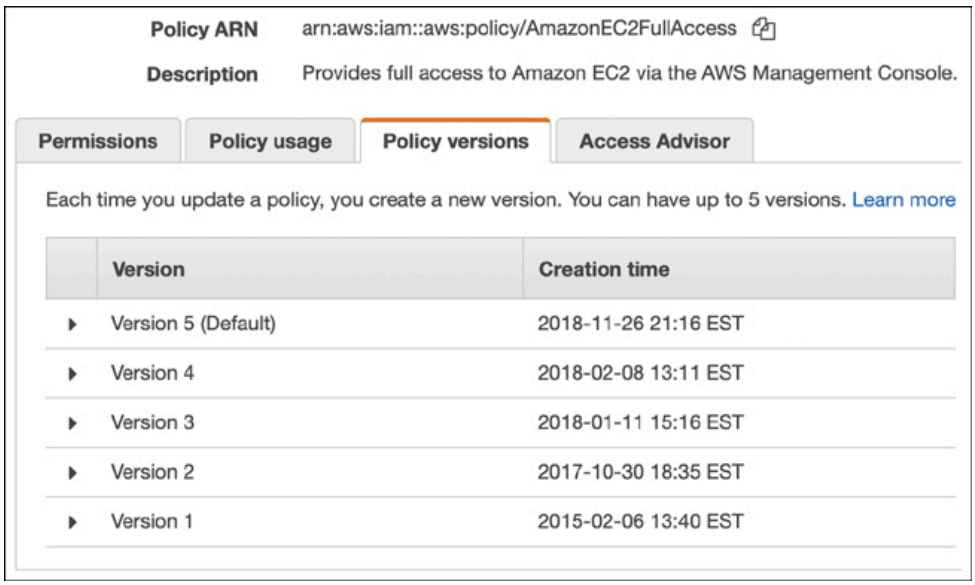


**Figure 3-28** Viewing Versions of IAM Policies

### Using Conditional Elements

Conditional elements of a JSON policy allow you to dictate optional parameters that must be met before the policy action is approved. Conditional elements are global or service-specific, as shown in the examples in Table 3-2. An organization could use the **aws:SourceIP** element, for example, to control the range of IP addresses from which administrators can log on to AWS.


**Table 3-2** Conditional Elements

| Element | Description |
|---|---|
| *Global Elements* | |
| **aws:CurrentTime** | This element checks for date/time conditions. |
| **aws:SecureTransport** | The request must use Secure Sockets Layer (SSL/TLS). |
| **aws:UserAgent** | This element allows certain client applications to make requests. |
| **aws:MultiFactorAuthPresent** | With this element, you can use the **BoolIfExists** operator to deny re- |

| | |
|---|---|
| | quests that do not include MFA. |
| **Bool** | The value of this element must be true. |
| **StringEquals** | The request must contain a specific value. |
| *Service-Specific Elements* | |
| **aws:PrincipalOrgID** | With this element, the user must be a member of a specific AWS organization. |
| **aws:PrincipalTag**/*tag-key* | This element checks for specific tags. |
| **aws:RequestTag/tag-key** | This element checks for a tag and a specific value. |
| **aws:PrincipalType** | This element checks for a specific user or role. |
| **aws:SourceVpce** | This element restricts access to a specific endpoint. |
| **aws:RequestedRegion** | This element allows you to control the regions to which API calls can be made. |
| **aws:SourceIp** | This element specifies an IPv4 or IPv6 address or range of addresses. |
| **aws:userid** | This element checks the user's ID. |

### Using Tags with IAM Identities

Most AWS resources allow you to define a number of tags for the resource you are creating or using. You can add custom attributes using

tags to both the IAM user and roles; for example, you can define a tag for an EC2 instance with the key **location** and the tag value **Toronto**.

Once you have tagged your resources, tags can be used to control IAM users and roles and their access to AWS resources. Tags can be added as a conditional element of each policy, mandating what tags need to be attached to the resource before the request is allowed. The following logic can be controlled using conditional tags:

- **Resources:** Tags can be used for IAM users and roles to determine whether access is allowed or denied to the requested resource based on the attached tags.
- **Principals:** Tags with Boolean logic can be used to control what the IAM user is allowed to do.

In **Example 3-4**, administrators can only delete users who have the **ResourceTag** set to **temp_user=can_terminate** tag and **PrincipalTag** attached to **useradmin=*true***. The tags in the example have been bolded for ease of reading.

**Example 3-4** Using Tags to Control Deletions

Click here to view code image

```
{
 "Version": "2012-10-17",
 "Statement": [{
 "Effect": "Allow",
 "Action": "iam:DeleteUser",
 "Resource": "*",
 "Condition": {"StringLike": {"iam:ResourceTag/temp_user":
    "can_terminate"}}
 }]
}
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "iam:* ",
 "Resource": "*",
 "Condition": {"StringEquals": {"aws:PrincipalTag/useradmin": "true"}}
 }
```

```
  ]
}
```

### IAM Roles


An **IAM role** is an IAM identity with specific permissions that define what the identity can and can't do at AWS. IAM roles provide temporary access to AWS resources once a role is associated with the following identities:

- An IAM user in the same AWS account as the role
- An IAM user in a different AWS account than the role
- An AWS web service such as Amazon EC2
- An external user authenticated by an external identity provider (IdP) service compatible with SAML 2.0 or OpenID Connect

When IAM roles are assumed by an identity, there is an additional linked policy called a *trust policy*. The use of an IAM role establishes a trust relationship between your *trusting* account and other AWS *trusted* accounts. The trusting account owns the AWS resource to be accessed. The trusted account contains the IAM identity that needs access to the resource. The trust policy and the security policy are assigned to the identity who will assume the role, as shown in **Example 3-5**. Roles do not have attached credentials. Temporary authentication credentials and a session token are assigned to an IAM user or federated user only after verification that the identity can assume the role. Trust policies are created for roles as follows:

- When a role is set up using the IAM console, the trust policy document is created and applied automatically.
- When a role is assigned to a user in the same AWS account, no trust policy is required, as the IAM user is already known to the AWS account.
- When a role is assigned to an IAM user residing in another AWS account, a trust policy must be assigned to the IAM user to be able to gain access.
- When the AWS CLI is used to create a role, both the trust policy and the permissions policy must be created.

**Example 3-5** IAM Role Trust Policy

Click here to view code image

```
{
 "Version":"2012-10-17",
 "Statement": {
 "Effect":"Allow",
 "Principal": {"AWS: "arn:iam::123456789:root" },
 "Action":"sts:AssumeRole",
 }
}
```

### When to Use IAM Roles


IAM roles are used for these authentication scenarios:

- Access to AWS resources using service-linked roles
- EC2 instances hosting applications needing access to AWS resources
- Third-party access required to AWS Accounts resources
- Web identity federation authentication by an external identity provider requiring access to AWS resources
- SAML 2.0 federation authentication requiring access to AWS resources
- Cross-account access—AWS account identities requiring access to resources in another AWS account

The following sections describe these scenarios.

#### AWS Services Perform Actions on Your Behalf

*Service-linked roles* assign the required permissions that allow each AWS service to carry out its job. AWS Config, Amazon Inspector, Amazon CloudWatch logs, and Amazon Elastic File System (EFS) are examples of AWS services using service-linked roles with the required permissions attached and temporary credentials granting access to carry out the requested tasks as required.

### EC2 Instances Hosting Applications Need Access to AWS Resources

AWS roles are useful for EC2 instances hosting applications that need access to AWS resources. For a workload to function properly, it needs valid AWS credentials to make its API requests to AWS resources. You could (but this a bad idea!) store a set of IAM users' credentials on the local hard disk of the application server or web server and allow the application to use those credentials.

Instead, implement the recommended best practice and create an IAM role that provides the required permission for the application hosted on the EC2 instance. The addition of a role to an EC2 instance creates an *instance profile* that is attached to the instance either during creation, as shown in **Figure 3-29**, or after creation.

Key Topic

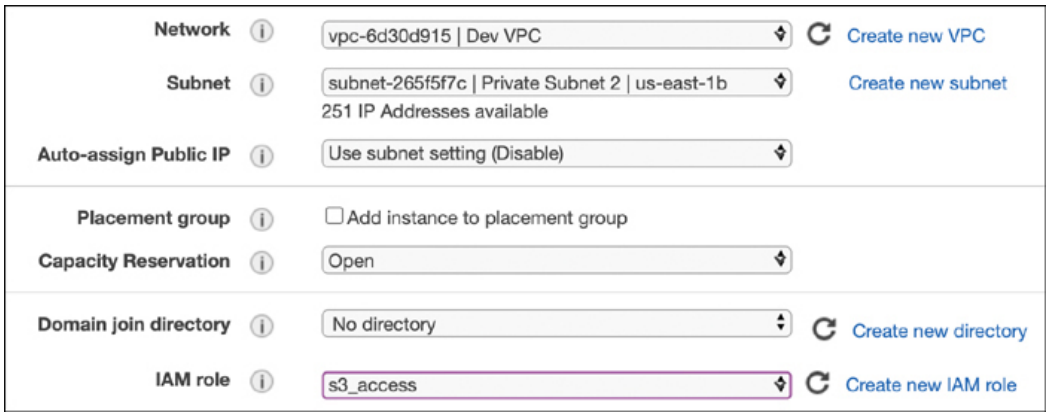


**Figure 3-29** Attaching an IAM Role to an EC2 Instance

When the IAM role is used, temporary credentials are supplied, and the application can access the required AWS resources. Each EC2 instance can have a single role assigned; however, the single role can be assigned to multiple instances. Any future changes made to the role are propagated to all instances that are currently using that role.

Using IAM roles means that you don't have to manage credentials. Instead, the AWS Security Token Service (STS) handles the authentication and authorization management. Each role assigned to an EC2 instance contains a permissions policy that lists the permissions to be used, plus a trust policy that allows the EC2 instance to be able to assume the assigned role and access the required AWS service. Each approved role is allowed access for a defined period of time; 1 hour is the default, as shown in **Figure 3-30**. The temporary credentials are stored in the memory of the

running instance and are part of the instance's metadata store under iam/security-credentials/role-name.

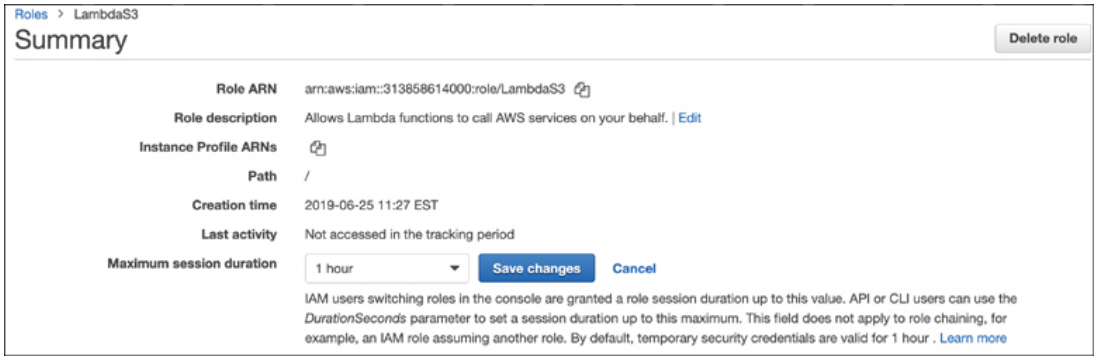


**Figure 3-30** Changing the Validity Time Frame for Temporary Credentials

Using temporary security credentials for an EC2 instance provides an additional advantage: The security credentials are automatically rotated just before their temporary session expires, ensuring that a valid set of credentials is always available for the application. IAM roles that control web/application server access to AWS cloud services is a concept that the AWS Certified Solutions Architect – Associate (SAA-C03) exam will expect you to know and understand.

### Access to AWS Accounts by Third Parties

Roles can be used to delegate access to third parties that require access to an organization's AWS resources. Perhaps the third party is managing some of your AWS resources. Granting access with a role and temporary security credentials allows you to grant access without sharing existing IAM security credentials. The role for the third party requires the following information:

- The third party's AWS account ID. The permissions policy specifies that identities from this AWS account number can assume the role.
- A secret identifier specified in the trust policy. The secret identifier is known to both the secure token service (AWS STS) and the third party.
- The permissions required by the third party to carry out their tasks.

### Web Identity Federation


Mobile applications can be designed to request temporary AWS security credentials using a process called *web identity federation*. Temporary cre-

dentials can map to a role with the required permissions to allow the mobile application to carry out its required tasks. Amazon Cognito is designed for scalable and secure mobile web-based federation to provide authentication for hundreds of thousands of users using social media providers such as Google, Facebook, Amazon, or any third-party identity provider that supports the OpenID Connect protocol. Amazon Cognito also provides support for enterprise federation using Microsoft Active Directory and any external IdP that supports SAML 2.0. Amazon Cognito uses user pools and federated identities to manage sign-up and authentication to mobile applications:

- **Amazon Cognito user pools:** Cognito enables you to create user pools of email addresses or phone numbers that can be linked to the desired application along with the type of authentication needed: through the user pool or by federating through a third-party IdP.
- **Amazon Cognito federated identities:** Cognito manages multiple IdPs—both identity federation and web-based federation options that mobile applications use for authentication and controlling access to your backend AWS resources and APIs, ensuring users get only the requested access to AWS services such as Amazon S3, Amazon DynamoDB, Amazon API Gateway, and AWS Lambda (see **Figure 3-31**).

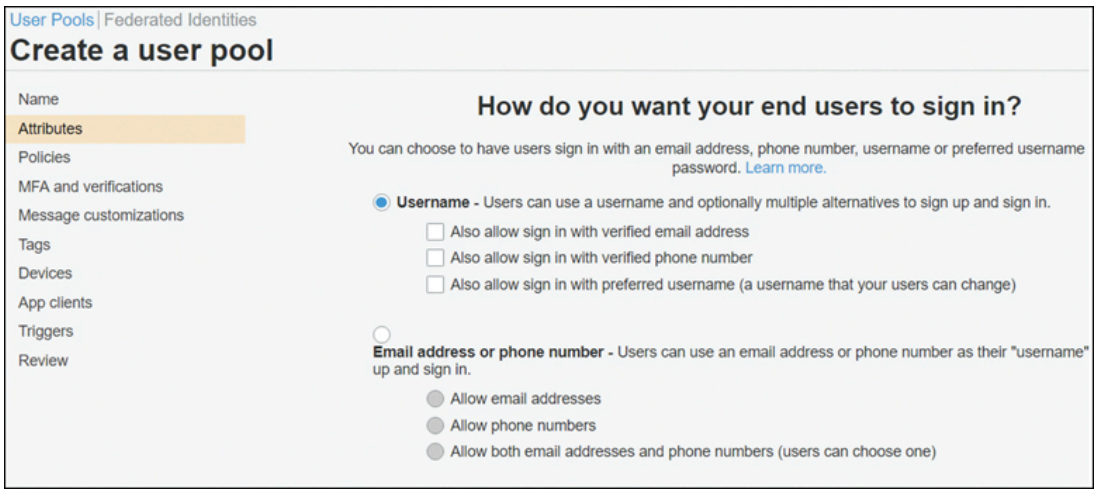


**Figure 3-31** Using Cognito for Mobile User Authentication

### SAML 2.0 Federation

The changes in authentication over the past 20 years have led to a number of options for single sign-on, including Cognito, AWS STS, Web Identity Federation, SAML 2.0, and Open ID Connect. Many companies use

Active Directory Domain Services, which has supported SAML for many years. SAML is supported by all public cloud providers in order to support most major corporations' ability to authenticate to the cloud using SSO.

Before the rise of mobile phones, corporate computer/user accounts were linked to applications hosted in the cloud. Mobile applications on devices are now commonplace requiring a unique type of authentication linking phones and devices running an application hosted by the cloud provider.

If an organization's end users already authenticate to a corporate network using a security service such as AD DS, you don't have to create separate IAM users for access to AWS services and resources. Instead, your users' corporate Active Directory user identities can be *federated* and synchronized to AWS with access to AWS resources using IAM roles. If your corporate network is compatible with SAML 2.0, it can be configured to provide an SSO process for gaining access to the AWS Management Console or other AWS services as required.

AWS provides several services to handle the different levels of federation used today. Amazon Cognito allows you to manage the variety of authentication providers in the industry, including Facebook, Google, Twitter, OpenID, and SAML, and even custom authentication providers that can be created from scratch. The odds are that you will use several of these prebuilt third-party authentication providers for controlling authentication and access to applications hosted at AWS. AD DS deployments with Active Directory Federated Services installed can take advantage of AWS Directory Service to build a trust relationship between your corporate Active Directory network, your corporate users, and resources hosted in an AWS account.

Here are big-picture steps for linking your on-premises Active Directory environment with AWS. Registration of the organization identity provider with AWS is necessary before you can create IAM roles that define the tasks that your corporate users can carry out at AWS.

**Step 1.** Register your organization's identity provider, such as Active Directory, with AWS. To do so, you must create and provide a metadata XML file, as shown in **Figure 3-32**, that lists your IdP and authentication keys used by AWS to validate the authentication requests from your organization.

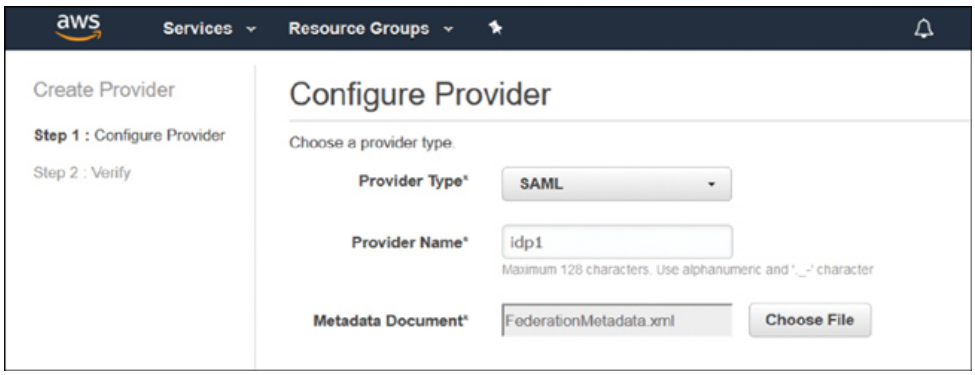


**Figure 3-32** Adding a Metadata XML

**Step 2.** Create IAM roles that provide access to the AWS resources. For the trust policy of the role, list your IdP as the principal. This ensures that users from your organization will be allowed to access AWS resources.

**Step 3.** Define which users or groups to map to the IAM roles, as shown in Figure 3-33, to provide access to the required AWS resources.

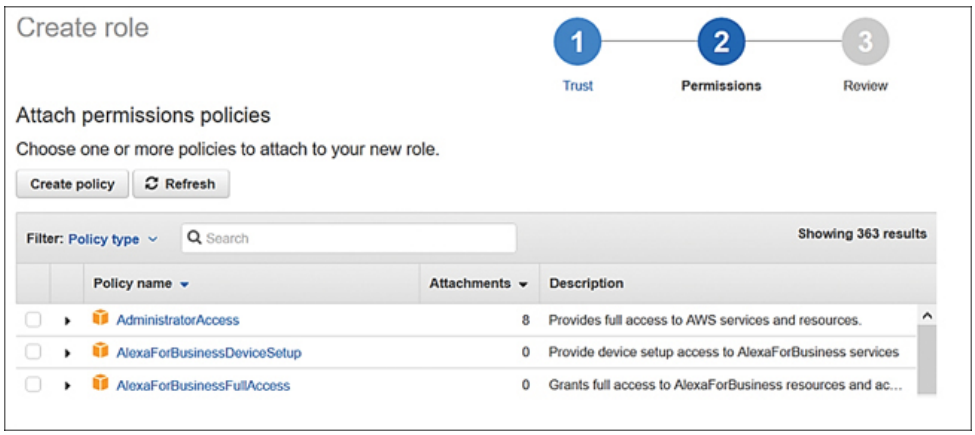


**Figure 3-33** Selecting Policies for an IAM Role

### Cross-Account Access

To allow access to resources in your AWS account from users in other AWS accounts rather than create an IAM user account within multiple AWS accounts for access, you can instead provide temporary *cross-account access* by using an IAM role. For example, a developer's IAM account located in the dev AWS account needs access to the S3 bucket **corp-docs** in the production AWS account. User identities in the dev AWS account use IAM roles to assume access to AWS resources in the production AWS account, using defined IAM roles and policies that authenticate using AWS STS. The following steps allow access to specific AWS services hosted in the production account from the dev AWS account:

**Step 1.** Create an IAM policy called **access-s3** in the production account that controls access to the S3 resource. The policy created is a custom policy that allows access to a specific S3 resource, as shown here:

Click here to view code image

```
Statement": [
{
"Effect": "Allow",
"Action": "s3:ListAllMyBuckets",
"Resource": "*"
},
{
"Effect": "Allow",
"Action": [
"s3:ListBucket",
"s3:GetBucketLocation"
],
"Resource": "arn:aws:s3:::corpdocs"
},
{
"Effect": " Allow",
"Action": [
"s3:GetObject",
"s3:PutObject",
"s3:DeleteObject"
],
" Resource": "arn:aws:s3:::corpdocs/*"
}
]
}
```

**Step 2.** Create an IAM role called **get-access**, which is assigned to the developer's IAM account that is linked to the IAM role policy **access-s3**.

**Step 3.** Get the ARN of the **get-access** role. The ARN is required to populate the custom IAM policy that allows the developer's IAM group to successfully switch accounts and access the **get-access** role.

**Step 4.** Grant access to the role in the developer's IAM user account by creating a custom policy that allows the developer to access the **get-access** role, as shown here:

Click here to view code image

```
{
"Version": "2012-10-17",
"Statement": {
"Effect": "Allow",
```

```
"Action": "sts:AssumeRole",
"Resource": "arn:aws:iam:::PRODUCTION-AWS-ACCT-ID:role/get-access"
}
}
```

**Step 5.** The developer can now switch roles by using the AWS Management Console and clicking Switch Role below the username, as shown in **Figure 3-34**, to gain access to the desired AWS resource.

---

**Note**

All Amazon services except for AWS RoboMaker, Amazon QuickSight, AWS Amplify, and Amazon Rekognition allow the use of roles. The action in the policy **AssumeRole** triggers communication with STS for verification.

---

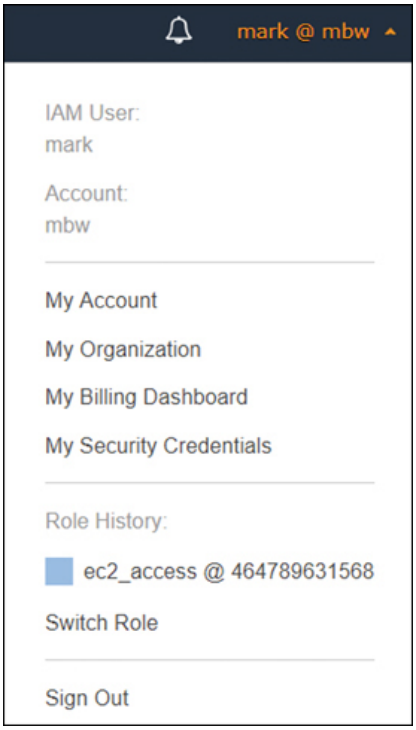


**Figure 3-34** Using the Switch Role Option for Cross-Account Access

### AWS Security Token Service

External authentication using identity federation is possible at AWS, including SSO federation with SAML 2.0, web-based federation (via

Amazon, Google, or Facebook), and federation using OpenID Connect. To support the various types of identity federation, running in the background at AWS is a global security service that provides temporary credentials upon request for external and internal access to AWS services using an attached IAM role. AWS STS uses a default global endpoint located in the US-East Northern Virginia region at **https://sts.amazonaws.com**; you can also choose to make STS API calls to other AWS regions using a regional endpoint if faster responses are needed. Temporary security credentials are, indeed, temporary, whereas access credentials linked to an IAM user are permanent. Temporary credentials are provided only when access to AWS resources is requested using a role.

The action in the policy can be defined as **AssumeRole**, **AssumeRoleWithSAML**, or **AssumeRoleWithWebIdentity**, as shown in **Figure 3-35**.

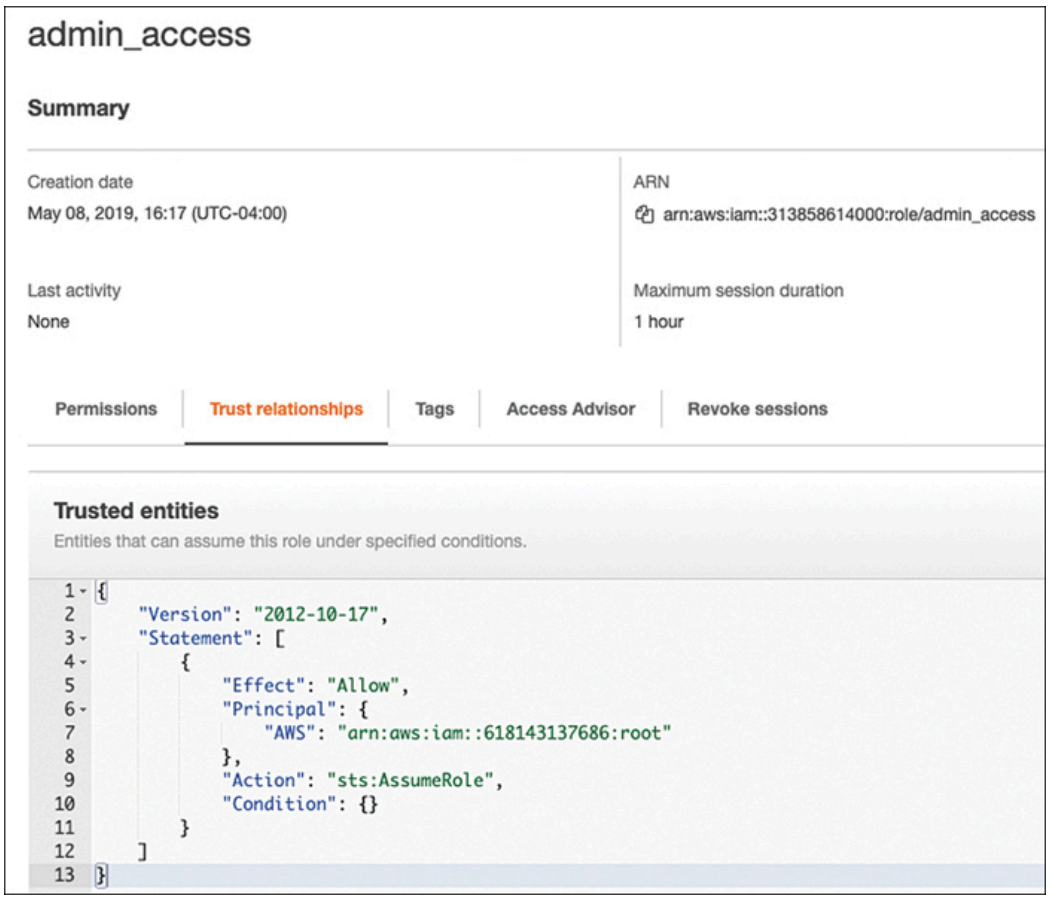


**Figure 3-35** Trusted Entities in Trust Policy

For either of these actions, STS is called. After verification, STS returns temporary credentials (access key, secret access key, and security token), which are valid for 1 hour by default. You can edit the maximum role session duration to control the exact length of time the assigned security credentials are valid (1 to 36 hours), or a custom length of time can be de-

fined. The advantages of using STS to provide temporary credentials for accessing AWS services are as follows:

- There's no need to rotate security credentials; STS performs credential rotation when temporary credentials are renewed.
- Applications use temporary credentials when they're hosted on EC2 instances with assigned roles, so there is no need for IAM user account credentials and passwords to be embedded in the application.
- STS manages and secures temporary credentials.
- Access to AWS resources can be defined without requiring a full IAM user account.
- Active sessions can be revoked at any time using the IAM dashboard, as shown in **Figure 3-36**.

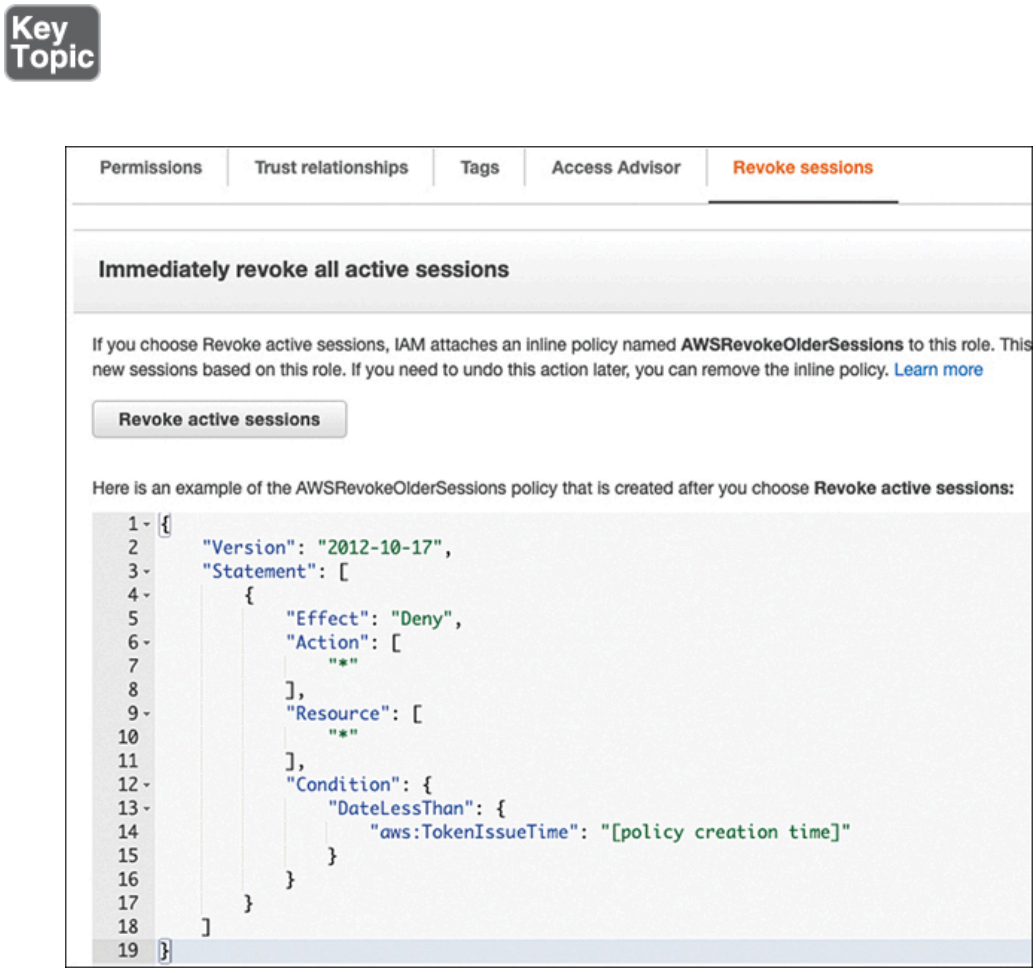


**Figure 3-36** Revoke Active Sessions

### IAM Best Practices

There are several best practices you should consider following when managing user security with IAM:

- **Root account:** Be careful with the AWS root account password. Don't create such a complicated password that you can't remember it and have to write it down. When you need access to the root account, reset the password. In addition, always enable MFA on a root account. Make sure your access keys for the root account have been deleted. You can check whether you have active access keys for your root account by logging on as the root user, opening the IAM console, and making sure the root account access keys have been removed (see **Figure 3-37**).

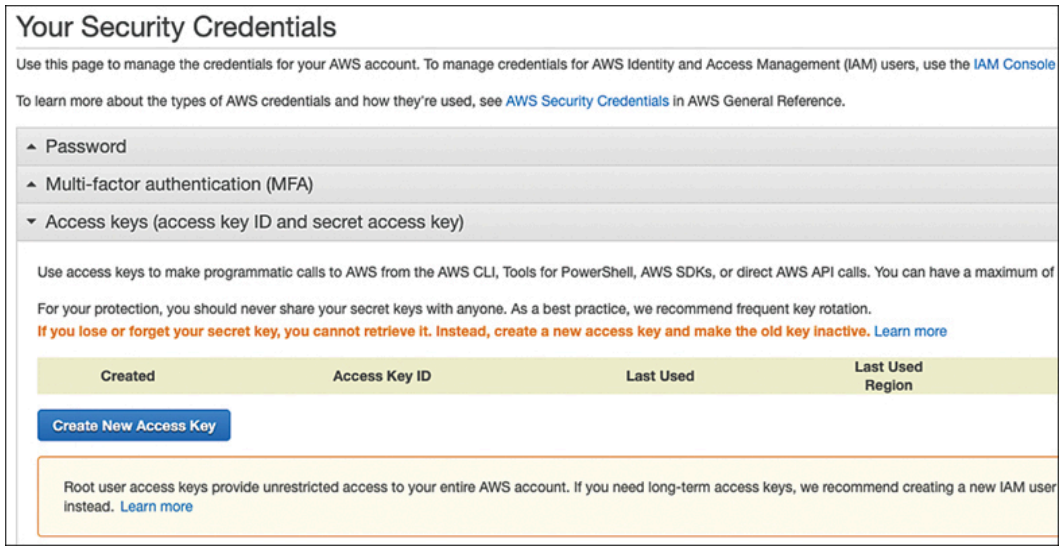


**Figure 3-37** Properly Set Up Root Account

- **Individual IAM users and groups for administration:** Even when creating single IAM users, consider placing them in an IAM group. At some point, each single user's duties may need to be assumed by someone else due to holidays or illness. It's much easier to add a new IAM user to an existing IAM group than to manage separate individual IAM users.
- **Permissions:** Grant least privileges when assigning IAM permissions. Take the time to get proficient at deploying IAM management policies. If necessary, create custom IAM policies for specific administrative access. Remember that most IAM accounts are administrator accounts. The goal should be to use IAM roles wherever possible because roles use controlled access with temporary credentials that are assigned and completely managed by STS.
- **Groups:** If possible, don't manage by individual IAM users; instead, manage by delegating access using IAM groups.
- **Conditions:** Consider restricting access with additional policy conditions. Consider adding a mandatory IP address range for administrators who need to perform administrative tasks and force authentication and access from a specific range of IP addresses.

- **CloudTrail logs:** Create a custom CloudTrail trail that saves all API calls and authentications from all AWS regions to a defined S3 bucket forever.
- **Passwords:** Make sure to create a strong password policy that matches corporate requirements.
- **Security credential rotation:** Consider rotating the security credentials on a timeline that matches the password change for each account. Even better, redesign your IAM security to use IAM roles for administrative tasks to ensure temporary credentials are used and managed by STS/AWS.
- **MFA:** Enable MFA for IAM users, including the root user of each AWS account. At the very least, use a software-based security code generator such as Google Authenticator or Authy.
- **Use IAM roles for application servers:** Use IAM roles to share temporary access to AWS resources for applications hosted on EC2 instances. Let AWS and STS manage application credentials.


### IAM Security Tools

Various security utilities and tools are available to make your job of managing IAM security easier. The following are tools to know for the SAA-C03 exam:

- **Credential Report:** From the IAM dashboard or using the AWS CLI, request and download a comma-separated values (CSV) report that lists the current status of IAM users in an AWS account (see **Figure 3-38**). Details include the status of the access keys (for example, usage, last used service, key rotation, passwords enabled/disabled, last time used, last changed, and MFA status. The information provided by the report is within the most recent 4-hour time status).

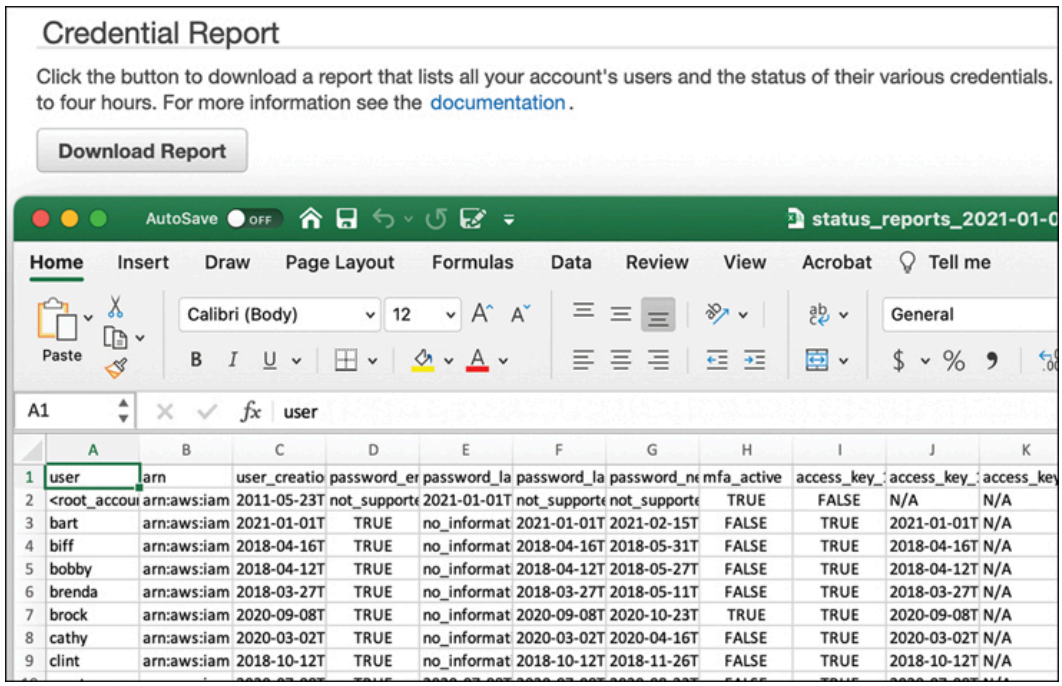


**Figure 3-38** Credential Report

- **Access Advisor:** Reports can be generated to display the last time an IAM user or role accessed an AWS service. View reports for each IAM entity by first selecting the IAM user, group, or role, selecting the Access Advisor tab, and then viewing the contents of the Access Advisor tab, as shown in **Figure 3-39**.

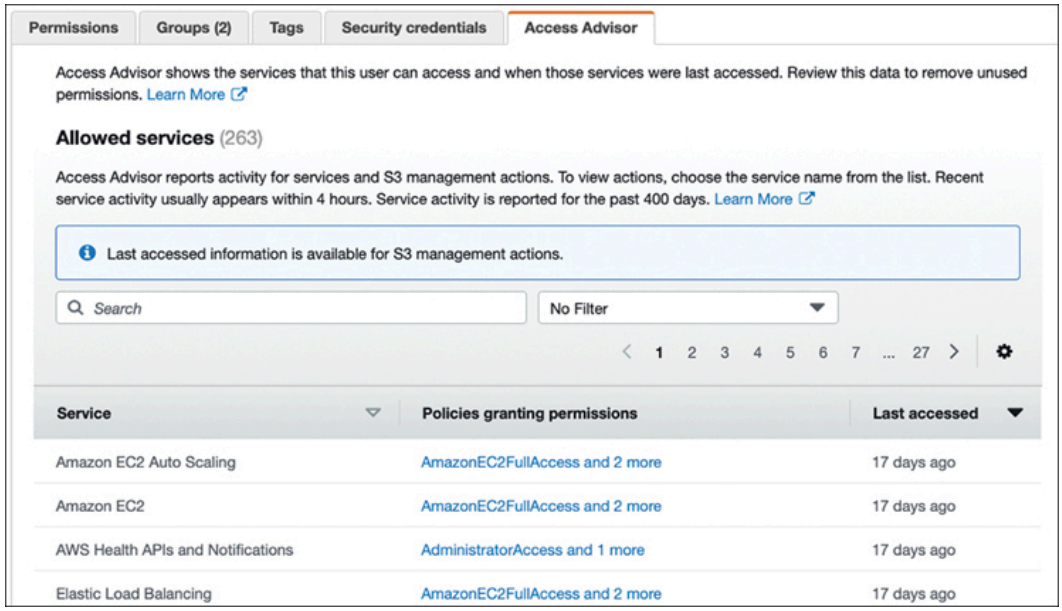


**Figure 3-39** Access Advisor Details

- **Policy Simulator:** After you've created your first policy, you might get lucky and have it work right away. If you are using the pre-created managed policies provided by AWS, they will usually work. However, a custom policy might not work as expected. Fortunately, Amazon has a simulator called the IAM Policy Simulator that you can use to test your policies (see **Figure 3-40**). The simulator evaluates your policy using

the same policy evaluation engine that would be used if real IAM policy requests were being carried out.

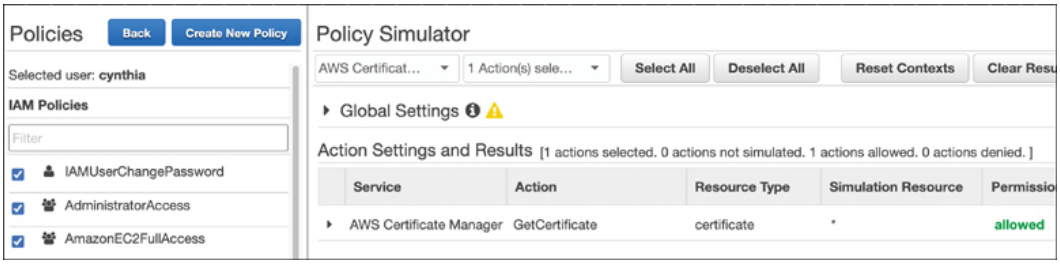


**Figure 3-40** IAM Policy Simulator

With the Policy Simulator, you can test IAM policies that are attached to IAM users, groups, or roles within an AWS account. You can select one or all security policies that are attached, and you can test all actions for what is being allowed or denied by the selected IAM policies. You can even include conditions such as the IP address that the request must come from. Both identity-based and resource-based policies can be tested with the Policy Simulator.


### IAM Cheat Sheet

For the AWS Certified Solutions Architect – Associate (SAA-C03) exam, you need to understand the following critical aspects of IAM:

- New IAM users are created with no access to any AWS services.
- IAM users can be assigned access keys, passwords, and multi-factor authentication.
- By using identity federation in conjunction with IAM roles, you can enable secure access to resources without needing to create an IAM user account.
- IAM is a global service that is not restricted to a single AWS region.
- IAM roles use security credentials provided by AWS STS that provide temporary access to AWS services and resources.
- Temporary security credentials include an AWS access key, a secret access key, and a security token.
- Each AWS account root account has full administrative permissions that cannot be restricted.
- IAM roles define a set of permissions for allowing or denying actions to AWS services.

- IAM roles are assumed by trusted identities.
- IAM roles allow you to delegate access permissions to AWS resources without requiring permanent credentials.
- There are no credentials assigned to an IAM role.
- IAM roles have two policies:
  - The permissions policy defines the permissions required for the role.
  - The trust policy defines the trusted accounts that are allowed to assume the role.


**AWS Identity Center**

AWS Identity Center, the successor to AWS Single Sign-On, is a cloud-based SSO service that manages access and permissions to third-party cloud applications and applications that support SAML 2.0 for AWS accounts contained in AWS Organizations. AWS Identity Center integrates with AWS Organizations and enumerated AWS accounts supporting the following features:

- Provides SSO access to cloud applications for AWS accounts
- Provides SSO access to AWS applications such as SageMaker
- Provides SSO access to EC2 Windows desktops
- Provides SSO access to IAM users and groups, AWS-Managed Microsoft AD directory users, and external identity providers
- Provides SSO access to many popular cloud-hosted applications (Salesforce, Box, Office 365)

To get started with AWS Identity Center, complete the following steps:

**Step 1.** AWS Organizations must first be deployed.

**Step 2.** Sign in using the AWS Organizations Management account credentials, which are required to enable AWS Identity Center.

**Step 3.** Choose the identity store that will have access to the AWS Identity Center user portal.

**Step 4.** After opening the AWS Identity Center console for the first time, enable the AWS Identity Center service.

**Step 5.** Add and configure applications that are to be integrated with AWS Identity Center, as shown in **Figure 3-41**.

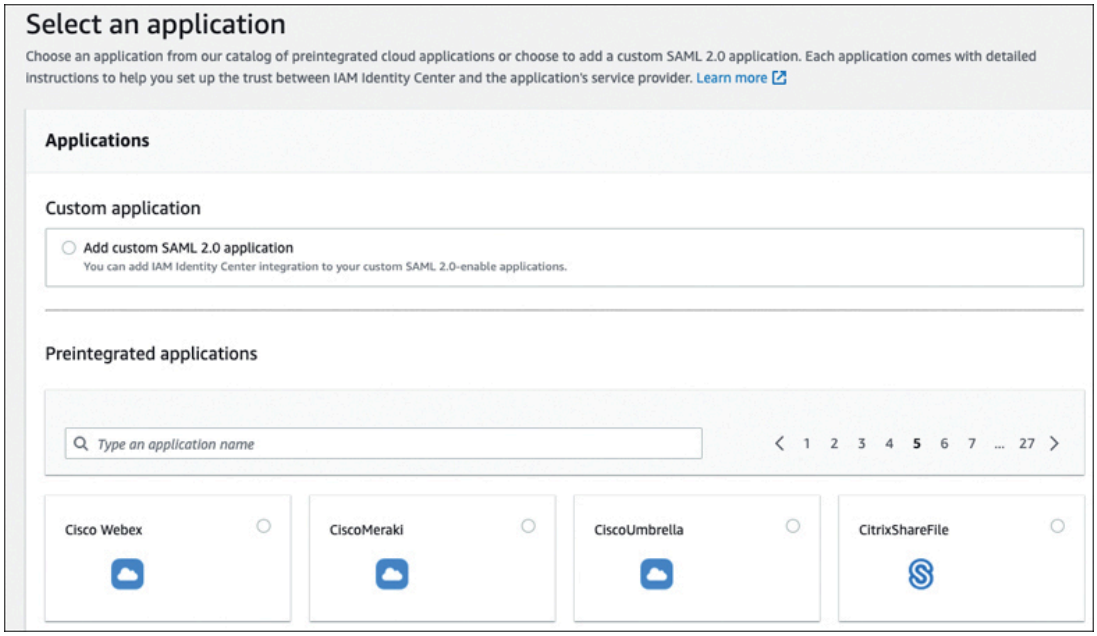


**Figure 3-41** AWS Identity Center Cloud Applications

## AWS Organizations

AWS Organizations enables centralized policy-based management for multiple AWS accounts that are grouped together in a tree structure. If you're lucky enough to not yet have multiple AWS accounts, you can look at AWS Organizations as a great starting point, especially if you know you're eventually going to have multiple AWS accounts to manage.

The first step to carry out with AWS Organizations is to create your initial organization with a specific AWS account; this account will henceforth be known as the *management account* (see **Figure 3-42**). The management account sits at the root of your AWS organization's tree.

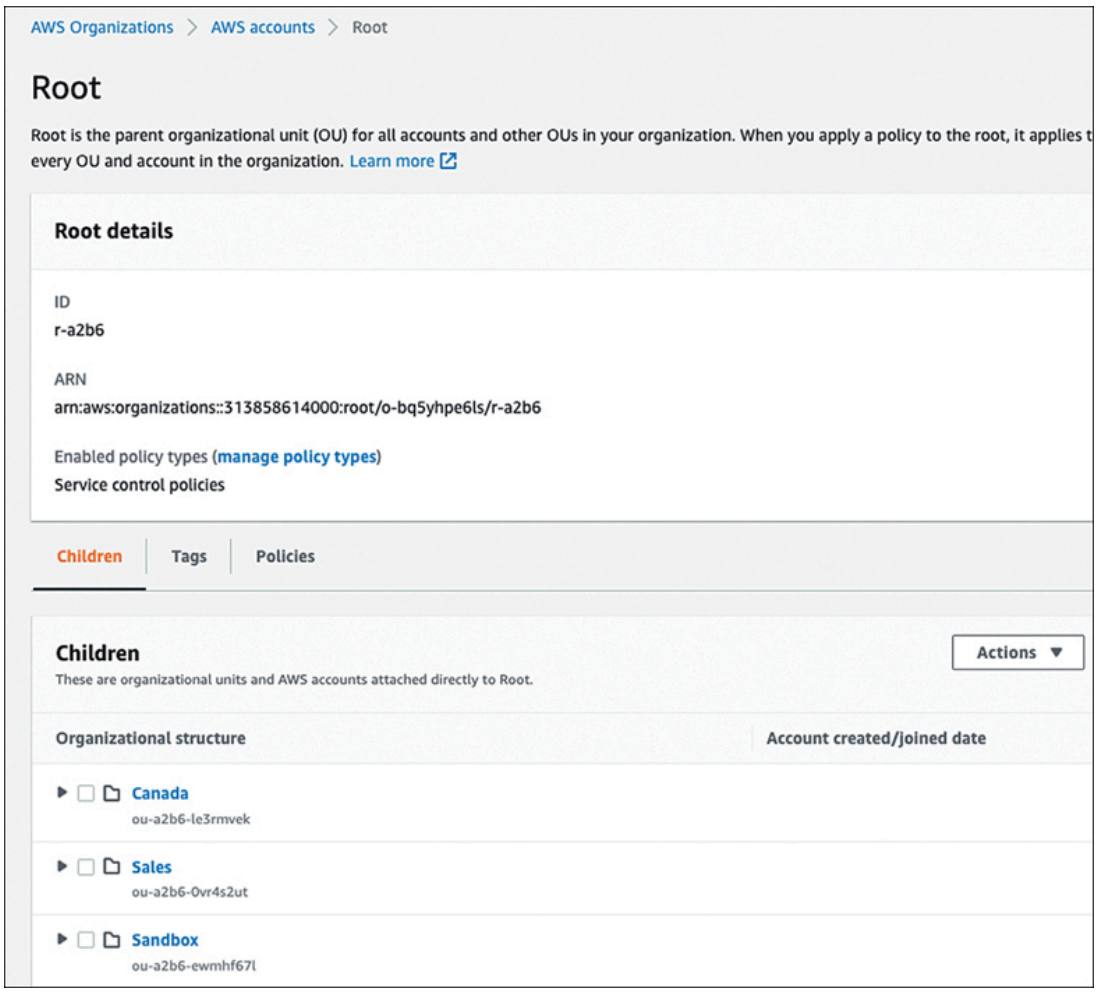


**Figure 3-42** AWS Organizations

---

### Note

The management account is also called the *payer account* because it is responsible for all the charges carried out by all the AWS accounts that are nested within AWS Organizations. AWS Organizations includes, by default, consolidated billing.

---

Using AWS Organizations, at the root, you can create new AWS accounts or add existing AWS accounts. All additional AWS accounts added to AWS Organizations are defined as member accounts. After grouping your AWS accounts, you can then apply security control policies to them. As introduced earlier in this chapter, the policies that can be applied to AWS Organizations are called service control policies (SCPs); these are permission boundaries that help define the effective permissions of applied IAM policies. If an SCP and an IAM policy assigned to a specific AWS account IAM user allow the same AWS service actions in both policy documents—that is, if the settings match—then the actions are allowed.

Within AWS Organizations, the AWS accounts can be organized into groupings called *organizational units (OUs)*, as shown in **Figure 3-43**. OUs

can be nested to create a tree-like hierarchy that meets your organization's needs and requirements. Nested OUs inherit SCPs from the parent OU and specific policy controls that can be applied directly to any OU. SCPs can be defined for an entire AWS organization, for specific OUs, or for specific AWS accounts located within an OU.

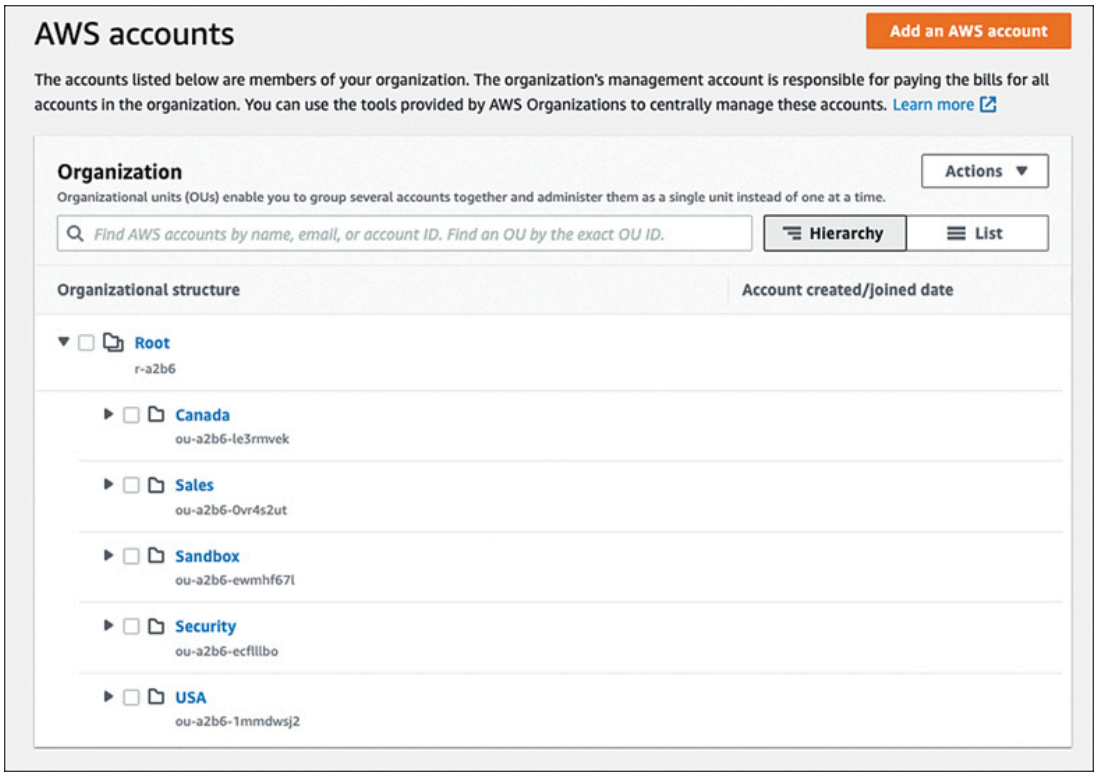


**Figure 3-43** OUs in AWS Organizations

### AWS Organizations Cheat Sheet


For the AWS Certified Solutions Architect – Associate (SAA-C03) exam, you need to understand the following critical aspects of AWS Organizations:

- An AWS organization is a collection of AWS accounts organized into a hierarchy that can be managed centrally.
- Each AWS account in an organization is designated as a member account located in a container. There is no technical difference between the master account and a member account other than its location.
- AWS Organizations supports consolidated billing.
- AWS Resource Access Manager can be used to share resources within the organization tree.
- Service control policies (SCPs) can be applied to AWS accounts or OUs contained within the AWS organization controlling access to AWS resources and services.

- AWS CloudTrail can be activated across all AWS accounts in the organization and cannot be turned off by member accounts.
- An organizational unit contains one or more AWS accounts within the AWS organizational tree.
- Security tools (AWS IAM, AWS Config, AWS Control Tower) can manage the needs and requirements of the AWS accounts that are members of the same AWS organization.
- AWS Cost Explorer can be used to track costs across accounts.

### AWS Resource Access Manager


AWS Resource Access Manager (RAM) allows you to centrally manage resources across AWS accounts and AWS Organizations.

AWS RAM allows you to share selected AWS resources hosted within a single AWS account with other AWS accounts. If you are using AWS Organizations, AWS RAM can also be used to share AWS resources between AWS accounts that are members of the same AWS Organization.

AWS RAM can share application or database servers between different AWS accounts instead of having to create duplicate resources.

To share resources using AWS RAM, first create a resource share (see **Figure 3-44**), configure the permissions to use for the resource, and select the principals that will have access to the resource.

With AWS RAM, the first task is to decide which resources that you own that you want to share. Next, you need to decide which principals to share the resource with; resource principals can be AWS accounts, OUs, IAM users, or the entire AWS organization.

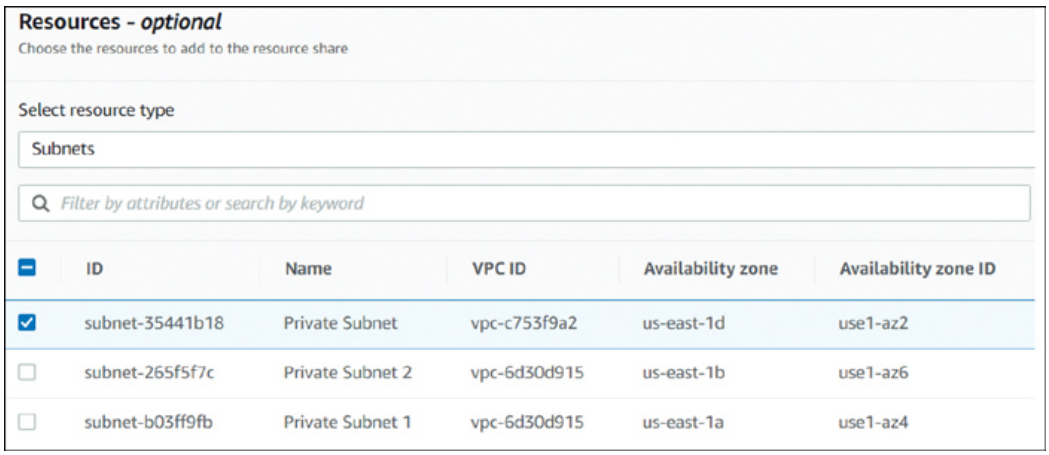


**Figure 3-44** Sharing Subnets with AWS RAM

If your AWS account is a member of an AWS organization, once sharing is enabled, any selected resource principal will be granted access to any resources shared by a resource share, as shown in **Figure 3-45**. If AWS Organizations is not deployed, the separate AWS account will receive an invitation from the AWS owner account that has created the resource share to join the resource share—after accepting the invitation, the AWS account will have access to the shared resource.

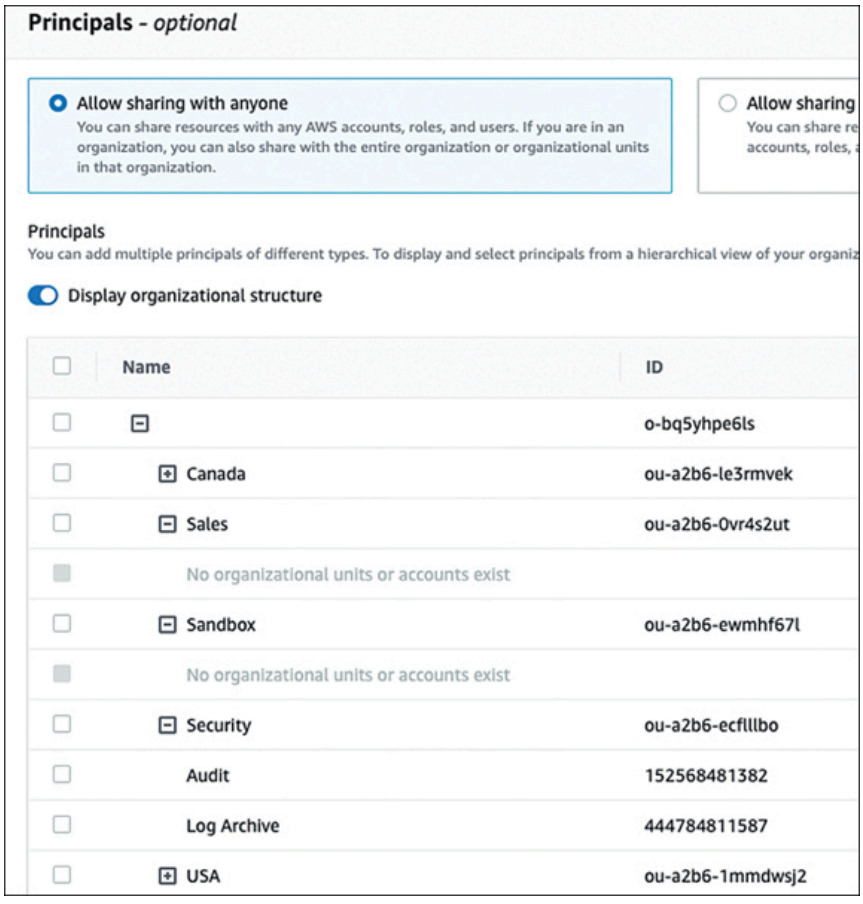


**Figure 3-45** Selecting Principals to Grant Access

The tasks that can be performed with the shared resource depends on the type of resource shared and the IAM security policies and service control policies that may have been applied. AWS resources that can be shared

with the AWS Resource Access Manager include Aurora DB clusters, Capacity reservations, Dedicated hosts, Glue catalogs, Image Builder images, AWS Outposts, and Transit Gateways.


### AWS Control Tower

AWS Control Tower automates the creation and governance of a secure, multi-account AWS environment using AWS Organizations, as shown in **Figure 3-46**. AWS Control Tower also automates the creation of a landing zone for onboarding AWS accounts using prebuilt blueprints that follow suggested best practices for configuring a default identity, federated access, and account structure. Deploying AWS Control Tower deploys and configures and integrates the following AWS services:

- AWS Organizations is deployed creating a multi-AWS account structure.

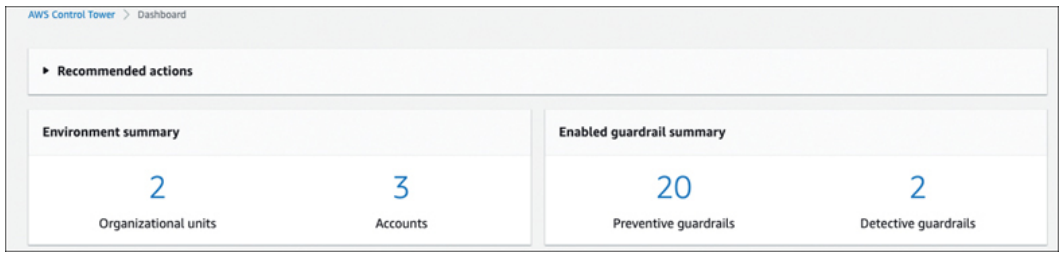


**Figure 3-46** AWS Control Tower Landing Zone

- AWS Organization's SCPs are deployed to prevent unwanted configuration changes.
- Federated access is enabled using AWS Identity Center.
- Central log archiving using AWS CloudTrail and AWS Config is stored in Amazon S3.
- Security audits are enabled across all AWS accounts using AWS Identity Center and Identity and Access Management.
- **Account Factory:** The account factory automates the provisioning of new AWS accounts in the deployed AWS Organization tree. Preapproved network configurations and AWS region selections can also be defined as the mandatory network baseline for all new AWS accounts, as shown in **Figure 3-47**.

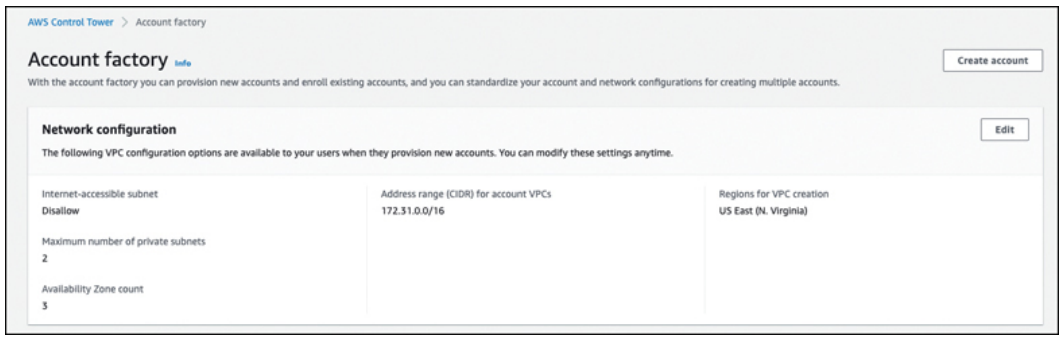


**Figure 3-47** Account Factory Preapproved Network Configurations

- **Guardrails:** Guardrails are used to provide ongoing governance of the AWS environment by preventing deployment of resources that don't follow prescribed policies. Guardrails prevent deployment of resources that don't match your rules. Guardrails can also be detective in nature by continually monitoring the resources that are deployed for noncon-formance. Guardrails are deployed using an AWS CloudFormation script that establishes the configuration baseline. SCPs create preven-tive guardrails that prevent unwanted infrastructure changes. Detective guardrails are created and enforced using AWS Config rules. There are also mandatory and optional guardrails that can be lever-aged, as shown in **Figure 3-48**. For example, organizations can man-date that any changes to logging configuration for Amazon S3 bucket policies can be set to disallowed. An example of an optional guardrail that can be enabled is detecting whether MFA is enabled for the root user.


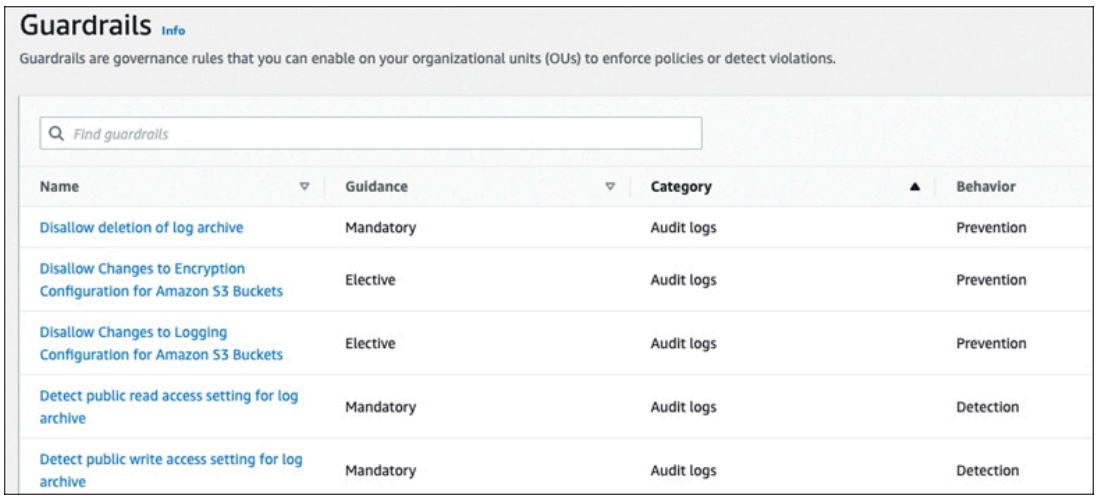


**Figure 3-48** AWS Control Tower Guardrails

## Exam Preparation Tasks

As mentioned in the section "How to Use This Book" in the Introduction, you have a couple of choices for exam preparation: the exercises here, Chapter 16, "Final Preparation," and the exam simulation questions in the Pearson Test Prep Software Online.

## Review All Key Topics

Review the most important topics in the chapter, noted with the Key Topic icon in the margin of the page. Table 3-3 lists these key topics and the page number on which each is found.


**Table 3-3** Chapter 3 Key Topics

| Key Topic Element | Description | Page Number |
|---|---|---|
| Figure 3-2 | AWS Config Service-Linked Roles Defined by IAM | 81 |
| Section | IAM Policy Definitions | 81 |
| Paragraph | IAM authentication | 82 |
| Figure 3-3 | IAM User Account Access Keys | 83 |
| Figure 3-4 | An Amazon AWS Resource Name (ARN) | 85 |
| Figure 3-6 | Policy Evaluation Logic | 87 |
| Figure 3-8 | Root User Logon | 89 |
| Figure 3-10 | Access Keys Required for CLI Operation | 92 |
| Figure 3-12 | Using Custom URL for IAM Users | 95 |
| Figure 3-15 | Creating an Additional Access Key Manually | 98 |
| Section | Using Multi-Factor Authentication | 99 |

| Key Topic Element | Description | Page Number |
|---|---|---|
| Figure 3-18 | Managed Policies | 101 |
| Section | Resource-Based Policies | 102 |
| Figure 3-20 | Identity and Resource Policies Working Together | 103 |
| Section | Policy Elements | 106 |
| Example 3-2 | IAM Policy for Performing Administrative Tasks | 110 |
| Section | Additional Policy Control Options | 110 |
| Section | AWS Organizations Service Control Policies | 112 |
| Section | IAM Policy Versions | 115 |
| Table 3-2 | Conditional Elements | 116 |
| Section | IAM Roles | 118 |
| Section | When to Use IAM Roles | 119 |
| Figure 3-29 | Attaching an IAM Role to an EC2 Instance | 120 |
| Section | Web Identity Federation | 121 |
| Figure 3-31 | Using Cognito for Mobile User Authentication | 122 |
| Figure 3-34 | Using the Switch Role Option for Cross-Account Access | 126 |
| Section | AWS Security Token Service | 126 |
| Figure 3-36 | Revoke Active Sessions | 128 |
| Section | IAM Best Practices | 128 |

| Key Topic Element | Description | Page Number |
|---|---|---|
| Section | IAM Security Tools | 130 |
| Section | IAM Cheat Sheet | 132 |
| Section | AWS Identity Center | 132 |
| Section | AWS Organizations Cheat Sheet | 136 |
| Section | AWS Resource Access Manager | 136 |
| Section | AWS Control Tower | 138 |
| Figure 3-48 | AWS Control Tower Guardrails | 139 |

## Define Key Terms

Define the following key terms from this chapter and check your answers in the glossary:

**Identity and Access Management (IAM)**

**multi-factor authentication (MFA)**

**externally authenticated user**

**condition**

**access key**

**IAM group**

**password policies**

**IAM role**

## Q&A

The answers to these questions appear in **Appendix A**. For more practice with exam format questions, use the Pearson Test Prep Software Online.

1. How can you tell when you're using the root account?

2. What is the best way to give an application secure access to AWS services?

3. What is the advantage of using a resource-based policy instead of an identity-based policy to protect an S3 bucket?

4. What is the best method for controlling access to AWS resources?

5. Why should inline policies be discouraged in most cases?

6. Which tool can you use to check your policies for proper functionality?

7. How can AWS Organizations help you manage multiple AWS accounts?

8. What security components are required to run a script from the command-line interface?