- Control who is authenticated (signed in) and authorized (has permissions) to use resources.
- AWS account **root user** is a single sign-in identity that has complete access to all AWS services and resources in the account.

Features

- You can grant other people permission to administer and use resources in your AWS account without having to share your password or access key.
- You can grant different permissions to different people for different resources.
- You can use IAM features to securely provide credentials for applications that run on EC2 instances which provide permissions for your applications to access other AWS resources.
- You can add two-factor authentication to your account and to individual users for extra security.
- You can allow users to use **identity federation** to get temporary access to your AWS account.
- You receive AWS CloudTrail log records that include information about **IAM identities** who made requests for resources in your account.
- You use an **access key** (an access key ID and secret access key) to make programmatic requests to AWS. An Access Key ID and Secret Access Key can only be uniquely generated once and must be regenerated if lost.
- IAM has been validated as being compliant with Payment Card Industry (PCI) Data Security Standard (DSS).
- IAM is *eventually consistent*. IAM achieves high availability by replicating data across multiple servers within Amazon's data centers around the world.
- IAM and AWS Security Token Service (STS) are offered at no additional charge.
- Your unique account sign-in page URL:
  https://*My_AWS_Account_ID*.signin.aws.amazon.com/console/

- You can use IAM tags to add custom attributes to an IAM user or role using a tag key–value pair.
- You can generate and download a credential report that lists all users on your AWS account. The report also shows the status of passwords, access keys, and MFA devices.

Infrastructure Elements

- **Principal**
  - An entity that can make a request for an action or operation on an AWS resource. Users, roles, federated users, and applications are all AWS principals.
  - Your AWS account root user is your *first principal*.
- **Request**

    - When a principal tries to use the AWS Management Console, the AWS API, or the AWS CLI, that principal sends a *request* to AWS.
    - Requests includes the following information:
        - **Actions or operations** – the actions or operations that the principal wants to perform.
        - **Resources** – the AWS resource object upon which the actions or operations are performed.
        - **Principal** – the user, role, federated user, or application that sent the request. Information about the principal includes the policies that are associated with that principal.
        - **Environment data** – information about the IP address, user agent, SSL enabled status, or the time of day.
        - **Resource data** – data related to the resource that is being requested.

- **Authentication**

    - To authenticate from the console as a user, you must sign in with your user name and password.
    - To authenticate from the API or AWS CLI, you must provide your access key and secret key.

- **Authorization**

    - AWS uses values from the *request context* to check for policies that apply to the request. It then uses the policies to determine whether to allow or deny the request.
    - Policies types can be categorized as *permissions policies* or *permissions boundaries*.
        - *Permissions policies* define the permissions for the object to which they're attached. These include identity-based policies, resource-based policies, and ACLs.
        - *Permissions boundary* is an advanced feature that allows you to use policies to limit the maximum permissions that a principal can have.
    - To provide your users with permissions to access the AWS resources in their own account, you need **identity-based policies**.
    - **Resource-based policies** are for granting cross-account access.
    - Evaluation logic rules for policies:
        - By default, **all requests are denied**.
        - An *explicit allow* in a permissions policy overrides this default.
        - A *permissions boundary* overrides the allow. If there is a permissions boundary that applies, that boundary must allow the request. Otherwise, it is implicitly denied.
        - An explicit deny in any policy overrides any allows.

- **Actions or Operations**

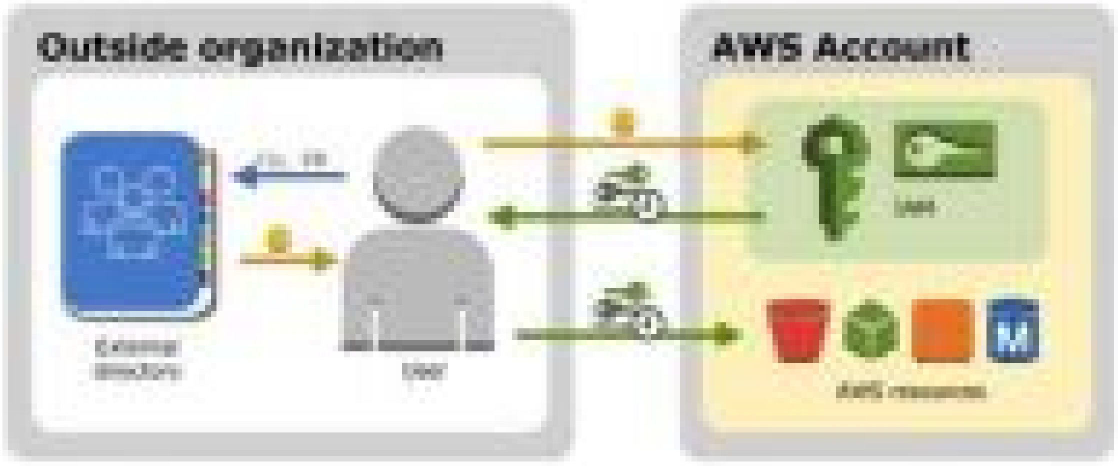
    - Operations are defined by a service, and include things that you can do to a resource, such as viewing, creating, editing, and deleting that resource.

- **Resource**

    - An object that exists within a service. The service defines a set of actions that can be performed on each resource.

Users

- **IAM Users**
    - Instead of sharing your root user credentials with others, you can create individual **IAM users** within your account that correspond to users in your organization. IAM users are not separate accounts; they are users within your account.
    - Each user can have its own password for access to the AWS Management Console. You can also create an individual access key for each user so that the user can make programmatic requests to work with resources in your account.
    - By default, a brand new IAM user has **NO permissions** to do anything.
    - Users are global entities.
- **Federated Users**
    - If the users in your organization already have a way to be authenticated, you can federate those user identities into AWS.



- **IAM Groups**
    - An IAM group is a collection of IAM users.
    - You can organize IAM users into IAM groups and attach access control policies to a group.
    - A user can belong to multiple groups.
    - Groups cannot belong to other groups.
    - Groups do not have security credentials, and cannot access web services directly.
- **IAM Role**
    - A role does not have any credentials associated with it.
    - An IAM user can assume a role to temporarily take on different permissions for a specific task. A role can be assigned to a federated user who signs in by using an external identity provider instead of IAM.
    - **AWS service role** is a role that a service assumes to perform actions in your account on your behalf. This service role must include all the permissions required for the service to access the AWS resources that it needs.

        - **AWS service role for an EC2 instance** is a special type of service role that a service assumes to launch an EC2 instance that runs your application. This role is assigned to the EC2 instance when it is launched.
        - **AWS service-linked role** is a unique type of service role that is linked directly to an AWS service. Service-linked roles are predefined by the service and include all the permissions that the service requires to call other AWS services on your behalf.
    - An instance profile is a container for an IAM role that you can use to pass role information to an EC2 instance when the instance starts.
- Users or groups can have multiple policies attached to them that grant different permissions.

When to Create IAM User	When to Create an IAM Role
You created an AWS account and you're the only person who works in your account.	You're creating an application that runs on an Amazon EC2 instance and that application makes requests to AWS.
Other people in your group need to work in your AWS account, and your group is using no other identity mechanism.	You're creating an app that runs on a mobile phone and that makes requests to AWS.
You want to use the command-line interface to work with AWS.	Users in your company are authenticated in your corporate network and want to be able to use AWS without having no sign in again (federate into AWS)

Policies

- Most permission policies are JSON policy documents.
- The IAM console includes *policy summary tables* that describe the access level, resources, and conditions that are allowed or denied for each service in a policy.
- The *policy summary table* includes a list of services. Choose a service there to see the *service summary*.
- This *summary table* includes a list of the actions and associated permissions for the chosen service. You can choose an action from that table to view the action *summary*.
- To assign permissions to federated users, you can create an entity referred to as a **role** and define permissions for the **role**.
- **Identity-Based Policies**
    - Permissions policies that you attach to a principal or identity.
    - **Managed policies** are standalone policies that you can attach to multiple users, groups, and roles in your AWS account.
    - **Inline policies** are policies that you create and manage and that are embedded directly into a single user, group, or role.

- **Resource-based Policies**
    - Permissions policies that you attach to a resource such as an Amazon S3 bucket.
    - Resource-based policies are only inline policies.
    - **Trust policies** – resource-based policies that are attached to a role and define which principals can assume the role.

AWS Security Token Service (STS)

- Create and provide trusted users with temporary security credentials that can control access to your AWS resources.
- Temporary security credentials are short-term and are not stored with the user but are generated dynamically and provided to the user when requested.
- By default, AWS STS is a global service with a single endpoint at https://sts.amazonaws.com.

Assume Role Options

- AssumeRole – Returns a set of temporary security credentials that you can use to access AWS resources that you might not normally have access to. These temporary credentials consist of an access key ID, a secret access key, and a security token. Typically, you use *AssumeRole*within your account or for cross-account access.
    - You can include multi-factor authentication (MFA) information when you call *AssumeRole*. This is useful for cross-account scenarios to ensure that the user that assumes the role has been authenticated with an AWS MFA device.
- AssumeRoleWithSAML – Returns a set of temporary security credentials for users who have been authenticated via a SAML authentication response. This allows you to link your enterprise identity store or directory to role-based AWS access without user-specific credentials or configuration.
- AssumeRoleWithWebIdentity – Returns a set of temporary security credentials for users who have been authenticated in a mobile or web application with a web identity provider. Example providers include Amazon Cognito, Login with Amazon, Facebook, Google, or any OpenID Connect-compatible identity provider.

STS Get Tokens

- GetFederationToken – Returns a set of temporary security credentials (consisting of an access key ID, a secret access key, and a security token) for a federated user. You must call the GetFederationToken operation using the long-term security credentials of an IAM user. A typical use is in a proxy application that gets temporary security credentials on behalf of distributed applications inside a corporate network.
- GetSessionToken – Returns a set of temporary credentials for an AWS account or IAM user. The credentials consist of an access key ID, a secret access key, and a security token. You must call the GetSessionToken operation using the long-term security credentials of an IAM user. Typically, you use GetSessionToken if you want to use MFA to protect programmatic calls to specific AWS API operations.

IAM Access Analyzer

- Provides policy checks that help you proactively validate policies when creating them. These checks analyze your policy and report errors, warnings, and suggestions with actionable recommendations that help you set secure and functional permissions.

- IAM Access Analyzer continuously monitors for new or updated resource policies and permissions granted for S3 buckets, KMS keys, SQS queues, IAM roles, Lambda functions, and Secrets Manager secrets.

Best Practices

- Lock Away Your AWS Account Root User Access Keys
- Create Individual IAM Users
- Use Groups to Assign Permissions to IAM Users
- Use AWS Defined Policies to Assign Permissions Whenever Possible
- Grant Least Privilege
- Use Access Levels to Review IAM Permissions
- Configure a Strong Password Policy for Your Users
- Enable MFA for Privileged Users
- Use Roles for Applications That Run on Amazon EC2 Instances
- Use Roles to Delegate Permissions
- Do Not Share Access Keys
- Rotate Credentials Regularly
- Remove Unnecessary Credentials
- Use Policy Conditions for Extra Security
- Monitor Activity in Your AWS Account

Become an IAM Policy Master in 60 Minutes or Less:

AWS IAM-related Cheat Sheets:

- Service Control Policies (SCP) vs IAM Policies

Note: If you are studying for the **AWS Certified Security Specialty exam**, we highly recommend that you take our **AWS Certified Security – Specialty Practice Exams** and read our **Security Specialty exam study guide**.

Validate Your Knowledge

Question 1

A company has 100 AWS accounts that are consolidated using AWS Organizations. The accountants from the finance department log in as IAM users in the `TD-Finance` AWS account. The finance team members need to read the consolidated billing information in the `TD-Master` AWS master account that pays the charges of all the member (linked) accounts. The required IAM access to the AWS billing services has already been provisioned in the master account. The Security Officer should ensure that the finance team must not be able to view any other resources in the master account.

Which of the following grants the finance team the necessary permissions for the above requirement?

1. Set up an IAM group for the finance users in the `TD-Finance` account then attach a `ViewBilling` permission and AWS managed `ReadOnlyAccess` IAM policy to the group.
2. Set up individual IAM users for the finance users in the `TD-Master` account then attach the AWS managed `ReadOnlyAccess` IAM policy to the group with cross-account access.
3. Set up an AWS IAM role in the `TD-Finance` account with the `ViewBilling` permission then grant the finance users in the `TD-Master` account the permission to assume that role.
4. Set up an IAM role in the `TD-Master` account with the `ViewBilling permission` then grant the finance users in the `TD-Finance` account the permission to assume the role.

Show me the answer!

Correct Answer: 4

You can use the consolidated billing feature in AWS Organizations to consolidate billing and payment for multiple AWS accounts or multiple Amazon Internet Services Pvt. Ltd (AISPL) accounts. Every organization in AWS Organizations has a *master (payer) account* that pays the charges of all the *member (linked) accounts*.

AWS Billing (service prefix: `aws-portal`) provides the service-specific resources, actions, and condition context keys for use in IAM permission policies. You can specify the following actions in the `Action` element of an IAM policy statement:

- **ModifyAccount** – Allow or deny IAM users permission to modify Account Settings.
- **ModifyAccount** – Allow or deny IAM users permission to modify Account Settings.
- **ModifyBilling** – Allow or deny IAM users permission to modify billing settings.
- **ModifyPaymentMethods** – Allow or deny IAM users permission to modify payment methods.
- **ViewAccount** – Allow or deny IAM users permission to view account settings.
- **ViewBilling** – Allow or deny IAM users permission to view billing pages in the console.
- **ViewPaymentMethods** – Allow or deny IAM users permission to view payment methods.
- **ViewUsage** – Allow or deny IAM users permission to view AWS usage reports.

Use policies to grant permissions to perform an operation in AWS. When you use an action in a policy, you usually allow or deny access to the API operation or CLI command with the same name. However, in some cases, a single action controls access to more than one operation.

Hence, the correct answer is: **Set up an IAM role in the `TD-Master` account with the `ViewBilling` permission then grant the finance users in the `TD-Finance` account the permission to assume the role.**

The option that says: **Set up an IAM group for the finance users in the `TD-Finance` account then attach a `ViewBilling` permission and AWS managed `ReadOnlyAccess` IAM policy to the group** is incorrect because the AWS managed policy called `ReadOnlyAccess` provides read-only access to all AWS services and resources. This violates the requirement that the finance people should not be able to view any other resources in the `TD-Master` account. In addition, you should use an IAM role in the `TD-Master` account and not an IAM Group in the `TD-Finance` account.

The option that says: **Set up individual IAM users for the finance users in the `TD-Master` account then attach the AWS managed `ReadOnlyAccess` IAM policy to the**

group with cross-account access is incorrect for the same reason as above. You should use an IAM role in the `TD-Master` account and not provide individual cross-account access to the IAM users.

The option that says: **Set up an AWS IAM role in the `TD-Finance` account with the `ViewBilling`permission, then grant the finance users in the `TD-Master` account the permission to assume that role** is incorrect because although it is correct to create an IAM Role with the `ViewBilling` permission, this should be done in the `TD-Master` account and not on the `TD-Finance` account.

References:

https://docs.aws.amazon.com/IAM/latest/UserGuide/list_awsbilling.html

https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/billing-permissions-ref.html

https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/billing-example-policies.html

Note: This question was extracted from our **AWS Certified Security Specialty Practice Exams**.

Question 2

You are working as a Solutions Architect for a leading insurance firm where you are instructed to provision access to certain IAM users who perform application development tasks in your VPC. The access should allow the users to create and configure various AWS resources such as deploying Windows EC2 servers. In addition, the users should be able to see the permissions in AWS Organizations to view information about the user's organization, including the master account email and organization limitations.

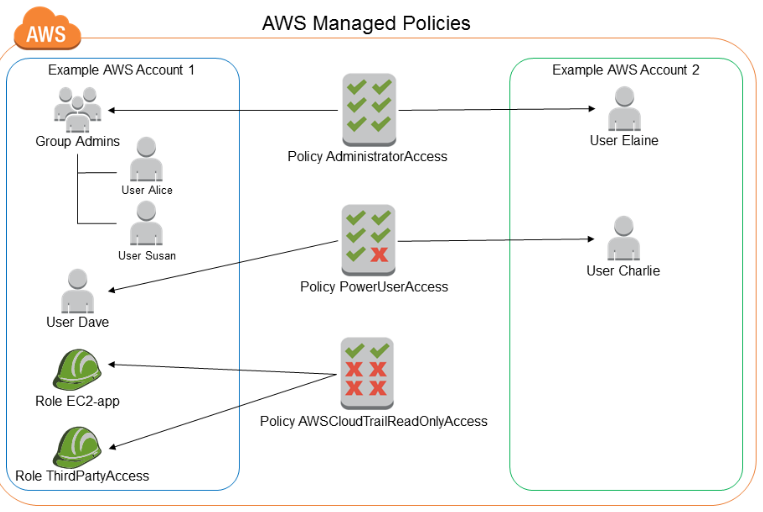
Which of the following should you implement to follow the standard security advice of granting the least privilege?

1. Attach the `PowerUserAccess` AWS managed policy to the IAM users.
2. Attach the `AdministratorAccess` AWS managed policy to the IAM users.
3. Create a new IAM role and attach the `SystemAdministrator` AWS managed policy to it. Assign the IAM Role to the IAM users.
4. Create a new IAM role and attach the `AdministratorAccess` AWS managed policy to it. Assign the IAM Role to the IAM users.

Show me the answer!

Correct Answer: 1

AWS managed policies for job functions are designed to closely align to common job functions in the IT industry. You can use these policies to easily grant the permissions needed to carry out the tasks expected of someone in a specific job function. These policies consolidate permissions for many services into a single policy that's easier to work with than having permissions scattered across many policies.



There are a lot of available AWS Managed Policies that you can directly attach to your IAM Users, such as Administrator, Billing, Database Administrator, Data Scientist, Developer Power User, Network Administrator, Security Auditor, System Administrator and many others.

For Administrators, you can use the AWS managed policy name: **AdministratorAccess** if you want to provision full access to a specific IAM User. This will enable the user to delegate permissions to every service and resource in AWS as this policy grants all actions for all AWS services and for all resources in the account.

For Developer Power Users, you can use the AWS managed policy name: **PowerUserAccess** if you have users who perform application development tasks. This policy will enable them to create and configure resources and services that support AWS aware application development. The first statement of this policy uses the *NotAction* element to allow all actions for all AWS services and for all resources except AWS Identity and Access Management and AWS Organizations. The second statement grants IAM permissions to create a service-linked role. This is required by some services that must access resources in another service, such as an Amazon S3 bucket. It also grants Organizations permissions to view information about the user's organization, including the master account email and organization limitations.

Therefore, the correct answer is: **Attach the `PowerUserAccess` AWS managed policy to the IAM users.**

The options that say: **Attach the AdministratorAccess AWS managed policy to the IAM users** and **Create a new IAM role and attach the `AdministratorAccess`AWS managed policy to it. Assign the IAM Role to the IAM users** are incorrect. Although an AdministratorAccess policy can meet the requirement, it is more suitable to attach a PowerUserAccess to the IAM users since this policy can provide the required access. Take note that you have to follow the standard security best practice of granting the least privilege. In addition, a managed policy can be directly attached to your IAM Users, which is one of the reasons why the latter option is incorrect.

The option that says: **Create a new IAM role and attach the `SystemAdministrator` AWS managed policy to it. Assign the IAM Role to the IAM users** is incorrect because the SystemAdministrator managed policy does not have AWS Organizations permissions to view information about the user's organization such as the master account email or the organization limitations. In this scenario, you have to use PowerUserAccess instead.

References:

https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_managed-vs-inline.html
https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_job-functions.html?shortFooter=true#access_policies_job-functions_create-policies

Note: This question was extracted from our **AWS Certified Solutions Architect Professional Practice Exams**.

Additional Training Materials: AWS IAM Video Courses on Udemy

1. AWS Identity Access Management (IAM) Practical Applications

References:
https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html
https://aws.amazon.com/iam/faqs/