Fundamental Concepts

1. **What is AWS IAM?**
   - **Answer:** AWS IAM (Identity and Access Management) is a web service that helps you securely control access to AWS resources. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources. It allows you to manage users, groups, roles, and their corresponding permissions centrally.
2. **Why is IAM essential in AWS? What problem does it solve?**
   - **Answer:** IAM is essential because it provides granular control over who can do what within your AWS account. It solves the problem of securely managing access for multiple users and applications without sharing passwords or access keys. Key benefits include:
     - **Enhanced Security:** Prevents unauthorized access by enforcing the principle of least privilege.
     - **Granular Control:** Allows specific permissions for specific resources.
     - **Centralized Management:** Manages users, groups, and permissions from one place.
     - **Temporary Credentials:** Enables secure access for applications running on EC2 or other services via Roles.
     - **Federated Access:** Allows users from external directories (like Active Directory) or identity providers to access AWS.
3. **What is the difference between Authentication and Authorization in the context of IAM?**
   - **Answer:**
     - **Authentication:** This is the process of verifying *who* a user or service is. In IAM, this is typically done using a username/password combination for console access, or access keys (Access Key ID and Secret Access Key) for programmatic access (API/CLI).
     - **Authorization:** This is the process of determining *what* an authenticated user or service is allowed to do. This is controlled by IAM policies attached to the user, group, or role, which specify allowed or denied actions on specific AWS resources.

Core IAM Components

1. **What is an IAM User?**
   - **Answer:** An IAM User represents a person or application that interacts with AWS. It has long-term credentials associated with it, such as a console password and/or access keys. It's generally recommended to create individual IAM users for each person who needs access to your AWS account, rather than sharing the root user credentials.
2. **What is an IAM Group?**
   - **Answer:** An IAM Group is a collection of IAM Users. You can attach IAM policies to a group, and all users within that group inherit those permissions. Groups are a way to

simplify permission management – instead of attaching policies to individual users, you manage permissions at the group level (e.g., a 'Developers' group, a 'Testers' group, an 'Admins' group). A user can belong to multiple groups.

3. **What is an IAM Role?**
   - **Answer:** An IAM Role is an IAM identity with specific permissions, similar to a user, but it does *not* have standard long-term credentials like a password or access keys associated with it. Roles are designed to be *assumed* by trusted entities (like IAM users, applications running on EC2 instances, or other AWS services like Lambda). When a role is assumed, it provides temporary security credentials for that session. Roles are the primary way to grant permissions to AWS services or enable cross-account access.
4. **When should you use an IAM Role instead of an IAM User?**
   - **Answer:** You should use an IAM Role in situations where you need to grant temporary access or delegate permissions without sharing long-term credentials. Key use cases include:
     - **Granting Permissions to AWS Services:** Allowing an EC2 instance, Lambda function, or other service to access other AWS resources (e.g., S3, DynamoDB). This is the most common and recommended use case.
     - **Cross-Account Access:** Allowing users or services from one AWS account to access resources in another account.
     - **Identity Federation:** Granting access to users authenticated through an external identity provider (like SAML 2.0 or OpenID Connect) or corporate directory.
     - **Temporary User Access:** Granting temporary elevated privileges to an IAM user who normally has lower permissions.
5. **What is an IAM Policy?**
   - **Answer:** An IAM Policy is a JSON document that formally defines permissions. It specifies what actions are allowed or denied, on which AWS resources, and under what conditions (optional). Policies are attached to IAM identities (users, groups, or roles) or sometimes directly to resources (resource-based policies).
6. **What are the key components of an IAM policy JSON structure?**
   - **Answer:** The main components are:
     - Version: Specifies the policy language version (usually "2012-10-17").
     - Statement: Contains one or more individual statements (rules). Each statement includes:
       - Sid (Optional): An identifier for the statement.
       - Effect: Specifies whether the statement results in an "Allow" or "Deny".
       - Principal (Required in resource-based policies, not used in identity-based): Specifies the user, account, service, or other entity allowed or denied access.

        - Action: Lists the API actions that are allowed or denied (e.g., "s3:GetObject", "ec2:DescribeInstances").
        - Resource: Specifies the object(s) that the statement applies to, using ARNs (Amazon Resource Names) (e.g., "arn:aws:s3:::my-bucket/*").
        - Condition (Optional): Specifies conditions under which the policy is in effect (e.g., based on time, source IP address, tags).

7. **What is the difference between AWS Managed Policies and Customer Managed Policies?**
    - **Answer:**
        - **AWS Managed Policies:** Created and managed by AWS. They cover common use cases (e.g., "ReadOnlyAccess", "AdministratorAccess", "AmazonS3FullAccess"). You cannot modify the permissions within these policies, but AWS may update them. They are convenient for getting started.
        - **Customer Managed Policies:** Created and managed by you within your AWS account. You have full control over the permissions defined in these policies. They allow you to create more granular permission sets tailored exactly to your needs, following the principle of least privilege. These can be attached to multiple users, groups, and roles.
8. **What is an Inline Policy?**
    - **Answer:** An Inline Policy is an IAM policy that is embedded directly into a single user, group, or role. It has a strict one-to-one relationship – if you delete the user/group/role, the inline policy is also deleted. Unlike managed policies, inline policies cannot be attached to multiple entities. They are useful for applying permissions that should not be shared or reused. Generally, customer-managed policies are preferred for better manageability unless a strict one-to-one relationship is explicitly required.

Security Best Practices

1. **What is the Principle of Least Privilege? Why is it important in IAM?**
    - **Answer:** The principle of least privilege means granting only the minimum permissions necessary for a user or service to perform its required tasks, and nothing more. It's critically important in IAM because it significantly reduces the potential "blast radius" if credentials are compromised or misused. By limiting permissions, you minimize the risk of accidental or intentional damage to your resources.
2. **What is the AWS Account Root User? What are the best practices for securing it?**
    - **Answer:** The Root User is the identity created when you first open an AWS account. It has complete, unrestricted access to all resources and billing information in the account.
    - **Best Practices:**
        - **Do Not Use for Everyday Tasks:** Avoid using the root user for routine administrative or application tasks. Create IAM users/roles instead.
        - **Enable MFA:** Secure the root user with Multi-Factor Authentication (MFA) immediately.

        - **Strong, Unique Password:** Use a complex password that is not used elsewhere.
        - **Do Not Create Access Keys:** Avoid creating access keys for the root user if possible. If they exist, delete them unless absolutely necessary for specific legacy tasks (which should be avoided).
        - **Secure Credentials:** Store the root user password and MFA device securely.

3. **What is MFA (Multi-Factor Authentication)? Why should you use it?**
    - **Answer:** MFA adds an extra layer of security to user sign-ins. It requires users to provide not only their password (something they know) but also a code from a secondary authentication device (something they have), such as a virtual MFA app (like Google Authenticator or Authy) on a smartphone, a U2F security key, or a hardware MFA token. You should use MFA because it significantly increases security by making it much harder for an unauthorized person to gain access, even if they somehow obtain the user's password. It's highly recommended for the root user and privileged IAM users.
4. **What is an IAM Password Policy?**
    - **Answer:** An IAM Password Policy allows you to define complexity requirements and mandatory rotation periods for IAM user passwords within your account. You can set rules like minimum password length, requiring specific character types (uppercase, lowercase, numbers, symbols), preventing password reuse, and setting password expiration periods. Enforcing a strong password policy enhances the security of user sign-in credentials.

Scenario Questions

1. **How would you grant an application running on an EC2 instance permission to read objects from an S3 bucket securely?**
    - **Answer:** The best practice is to use an IAM Role:
        1. **Create an IAM Policy:** Define a policy that allows the necessary S3 actions (e.g., s3:GetObject, s3:ListBucket) on the specific S3 bucket resource.
        2. **Create an IAM Role:** Create a role specifically for the EC2 service (select 'EC2' as the trusted entity).
        3. **Attach the Policy:** Attach the policy created in step 1 to this role.
        4. **Launch/Associate EC2 Instance:** Launch the EC2 instance, or modify an existing one, and associate the IAM role created in step 2 with the instance.

        - The EC2 instance will then automatically retrieve temporary credentials associated with the role via its instance metadata. The application code (using AWS SDKs) can use these credentials to make secure requests to S3 without needing hardcoded access keys.
2. **A new intern is joining your team and needs read-only access to EC2 and S3 services for learning purposes. How would you set this up using IAM best practices?**
    - **Answer:**

1. **Create an IAM User:** Create a new IAM user account for the intern. Ensure they only get console access initially (no access keys unless specifically needed and approved).
2. **Use Groups (Optional but recommended):** If there might be other interns or similar roles, create an 'Interns' or 'ReadOnlyUsers' IAM Group. Add the new user to this group.
3. **Attach Policies:** Attach the AWS managed policies AmazonEC2ReadOnlyAccess and AmazonS3ReadOnlyAccess to the *group* (or directly to the user if not using groups). This follows the principle of least privilege.
4. **Enforce MFA:** Instruct the intern to set up MFA on their IAM user account upon first login.
5. **Password Policy:** Ensure a strong password policy is in place for the account.
6. **Provide Credentials:** Securely provide the intern with their console login URL, username, and initial password (they should be forced to change it on first login).

Tips for Answering:

- **Emphasize Security:** IAM is all about security. Frame your answers around security best practices like least privilege and MFA.
- **Know the Core Components:** Be very clear on the definitions and use cases for Users, Groups, Roles, and Policies. The User vs. Role distinction is particularly important.
- **Understand Roles:** Roles are fundamental for service-to-service communication and are a common interview topic. Make sure you understand *why* they are used.
- **Clarity is Key:** Explain concepts simply and accurately.

Good luck with the interview! Understanding IAM well shows you have a good foundation in AWS security.