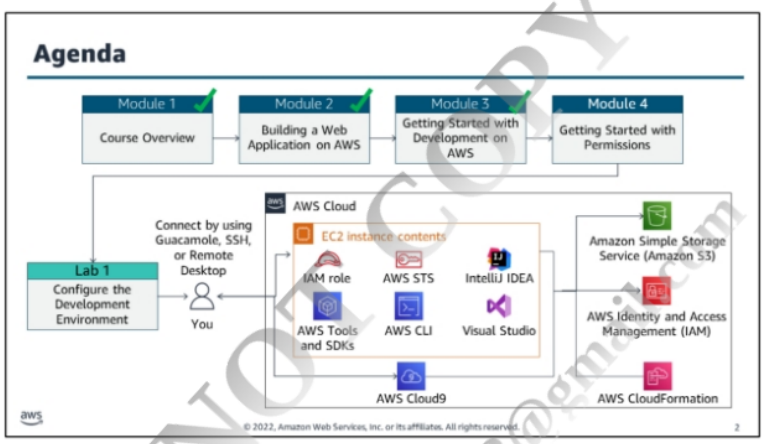
Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.



Module 1: Introduction

Module 2: Examine the details of the AWS architecture you can use to develop an application in its entirety.

Module 3: Review the benefits of AWS software development kits (AWS SDKs) when building an application.

Module 4: Configure a development environment with the support of AWS Identity and Access Management (IAM) permissions.

Lab 1: Use AWS Cloud9 to configure and test IAM permissions in a development environment.

Module objectives

By the end of this module, you will be able to:

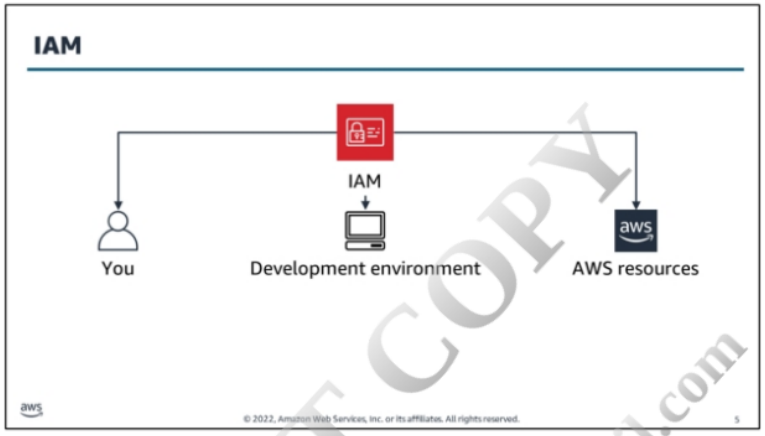
- Identify AWS Identity and Access Management (IAM) features and components
- Configure permissions to support a development environment
- Demonstrate how to test IAM permissions
- Configure your IDEs and SDKs to support a development environment
- Demonstrate accessing AWS services using SDKs and AWS Cloud9

 3

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

IAM recap

Module 4: Getting Started with Permissions



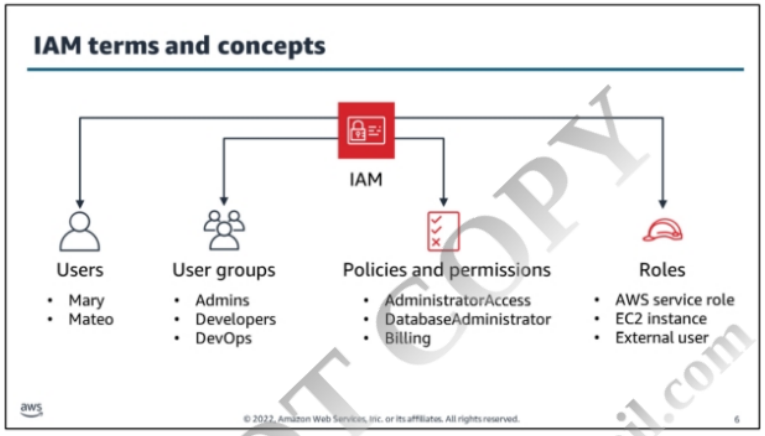
AWS Identity and Access Management (IAM) is a web service that helps you to control access to AWS resources for your users securely. You use IAM to control who can use your AWS resources (authentication). You also use IAM to control how users can use them (authorization).

Your application requires permissions for you, your development environment, and the AWS resources. Granting yourself access includes you and any others team members who require development or management access to your application.

With IAM, you can configure your development environment for efficient access to multiple AWS accounts.

IAM credentials also provide your resources access permissions to other resources in your AWS account or in other AWS accounts.

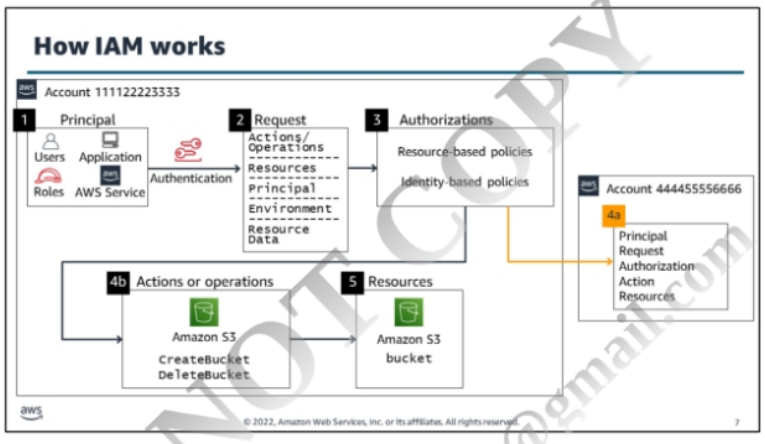
Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.



Use your knowledge of access control concepts to determine how to configure IAM access for your application. Think about who and what requires access to your application:

- **Users** – You, your team members, or entity that you create in AWS to represent the person or an application that uses it to interact with AWS.
- **Groups** – A collection of IAM users, often organized by job function. For example, developers or system administrators. Using groups, you can specify permissions for multiple users.
- **Policies** – Define permissions for an action regardless of the method that you use to perform the operation. You can attach an IAM customer managed policy or an AWS managed policy, such as *AdministratorAccess* or *DatabaseAdministrator*.
- **Roles** – Trusted entities, similar to an IAM user, with permissions policies that determine what the identity can and cannot do in AWS. However, the identity does not have any long-term credentials associated with it. An IAM user can assume a role with temporary security credentials for the role session.

For more information, see "IAM Identities (users, user groups, and roles)" in the *AWS Identity and Access Management User Guide* (https://docs.aws.amazon.com/IAM/latest/UserGuide/id.html).



IAM provides the infrastructure necessary to control authentication and authorization to your services. After becoming familiar with all IAM terms and the flow, you can build a more secure application and development environment.

1. **Principal**
   A principal is a person or application that uses the AWS account root user, an IAM user, role, or a federated user to sign in and make requests to AWS. To authenticate from the API, you must provide your access key and secret key.
2. **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. The request includes:
   - **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 person or application that used an entity (user or role) to send the request.
   - **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.

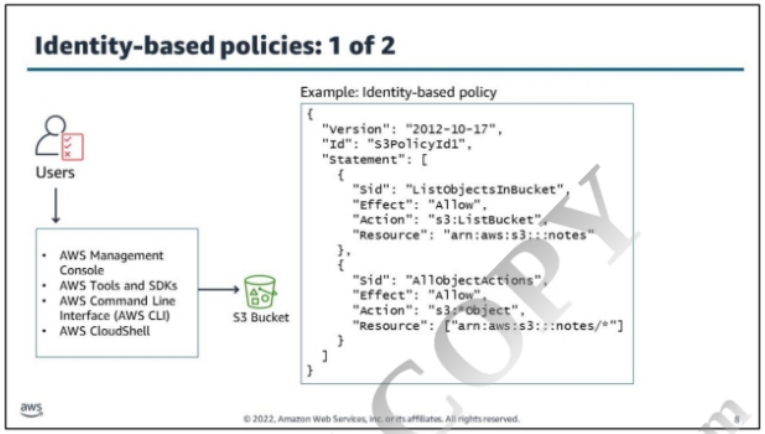
   AWS gathers the request information into a *request context*, which is used to evaluate and authorize the request.

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 4: Getting Started with Permissions

3. **Authorization**
   You must be authorized (allowed) to complete your request. During 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. Two of the most common policy types are *identity-based policies* and *resource-based policies*.
   - Identity-based policies are attached to an IAM user, group, or role. These policies let you specify what that identity can do (its permissions).
   - Resource-based policies are attached to a resource. For example, you can attach resource-based policies to Amazon S3 buckets. Resource-based policies are popular for granting cross-account access (**3a**).
4. **Actions (console) or operations (API, CLI)**
   After your request has been authenticated and authorized, AWS approves the actions or operations in your request. Operations are defined by a service and include things that you can do to a resource, such as creating or deleting that resource.
5. **Resources**
   After AWS approves the operations in your request, the operations can be performed on the related resources within your account. A *resource* is an object that exists within a service.

© 2022 Amazon Web Services, Inc. or its affiliates. All rights reserved.

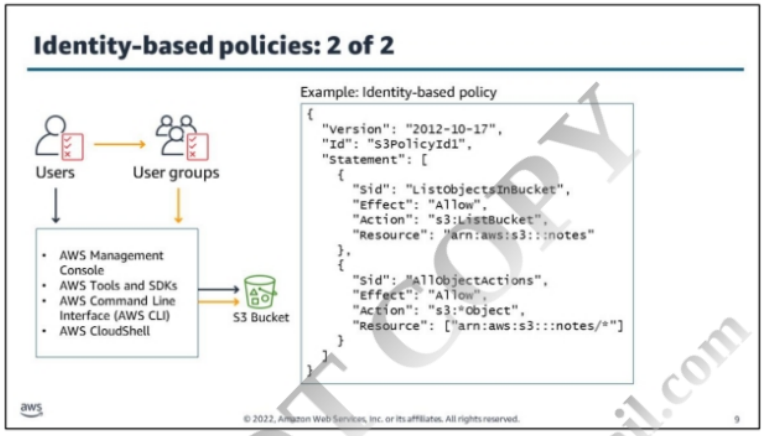


You can grant IAM users access to AWS services through the AWS Management Console or by using the AWS CLI or AWS SDKs.

When you create an IAM user, you grant it permissions by attaching an identity-based policy to the user directly.

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

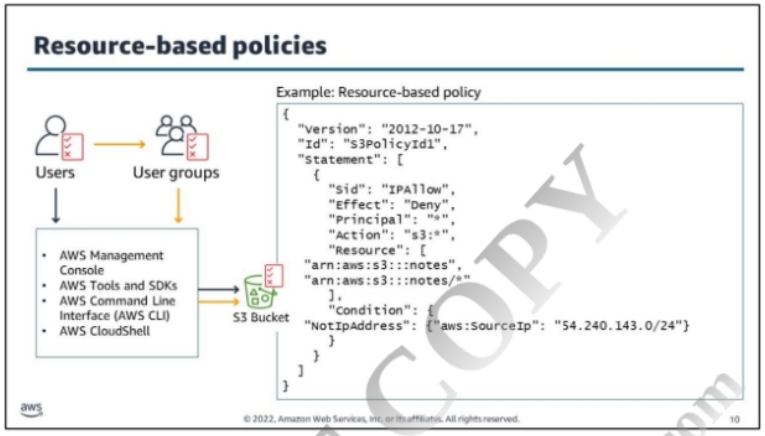
AWS Training and Certification Module 4: Getting Started with Permissions



Alternatively, you can make the IAM user a member of a user group that has appropriate permissions policies attached (recommended). This example of an identity-based policy allows read and write access to objects in a specific S3 bucket. Using IAM, you can add this policy to a user or a group.

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

- "Creating your first IAM admin user and user group" (https://docs.aws.amazon.com/IAM/latest/UserGuide/getting-started_create-admin-group.html)
- "Identity-based policies and resource-based policies" (https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_identity-vs-resource.html)



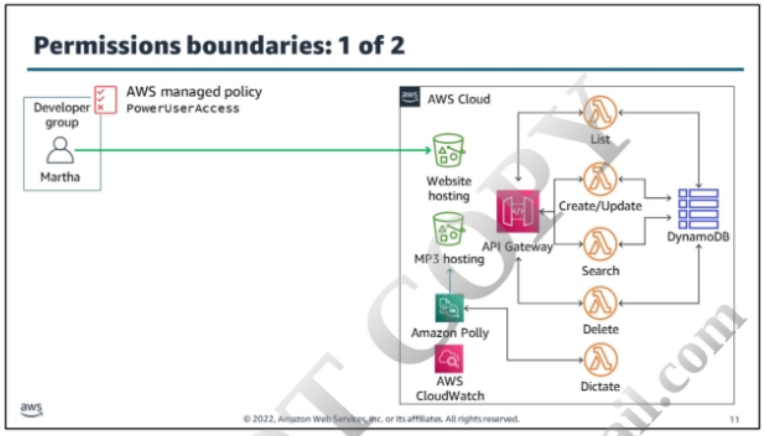
Resource-based policies attach to an AWS resource, such as an Amazon S3 bucket. These policies grant the specified principal permission to perform specific actions on that resource and defines under what conditions this applies.

The example resource-based policy denies permission for any user for any Amazon S3 operations on objects in the specified S3 bucket. Permissions are denied unless the request originates from the range of IP addresses.

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

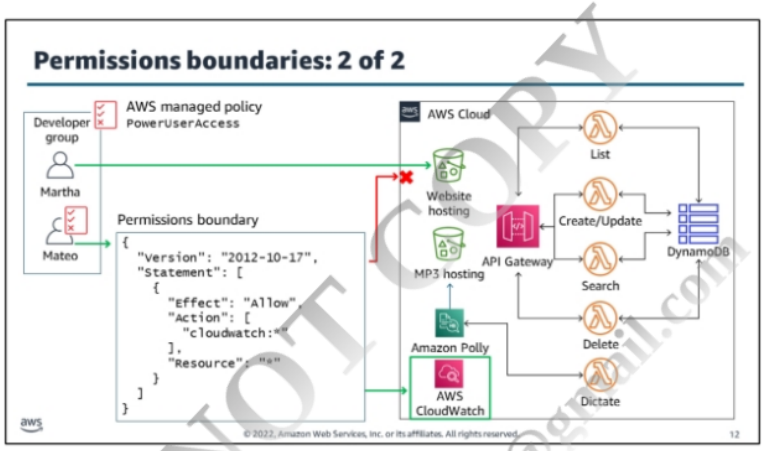
- "Creating your first IAM admin user and user group" (https://docs.aws.amazon.com/IAM/latest/UserGuide/getting-started_create-admin-group.html)
- "Identity-based policies and resource-based policies" (https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_identity-vs-resource.html).

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.



Permissions boundaries are an advanced IAM feature. They are used to set maximum permissions that an identity-based policy can grant to an IAM entity, such as users or roles. The entity's permissions boundary allows it to perform only the actions that are allowed by both its identity-based policies and its permissions boundaries. Permissions boundaries act as a barrier to prevent an entity from performing an action on a service that it should not perform.

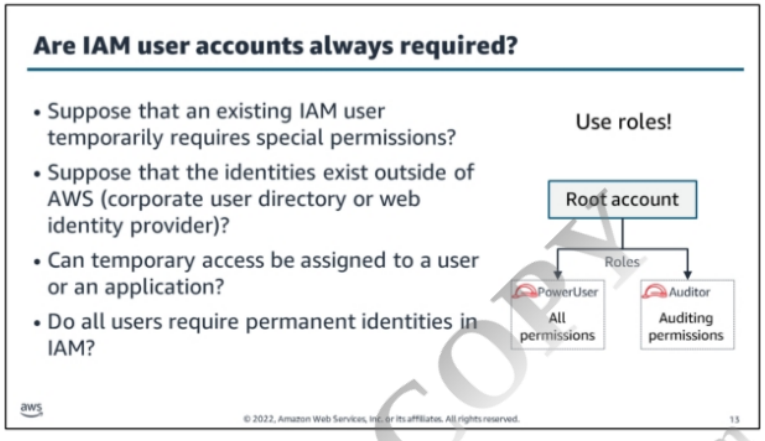
In this example, Martha belongs to the Developer group. An AWS managed policy, `PowerUserAccess` (Developer power user job function), is attached to the Developer group. This policy uses the `NotAction` element to allow all actions for all AWS services and for all resources except IAM and AWS Organizations. Only IAM permissions are granted to create a service-linked role.



Mateo is a new member of this group, which has an attached permissions boundary. This permissions boundary allows Mateo maximum access to a subset of services (Amazon CloudWatch). Mateo is a member of the Developer group, which allows access to Amazon S3. However, the bounded permissions prevent Mateo from performing actions on Amazon S3 because the actions are outside of Mateo's permissions boundary. For more information, see "AWS managed policies for job functions" in the *AWS Identity and Access Management User Guide* (https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_job-functions.html).

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 4: Getting Started with Permissions



Sometimes you need to delegate access temporarily to users or services that normally don't have access to your AWS resources. For example, a user in one AWS account might require access to resources in another account. Or a mobile app might use AWS resources. However, you don't want to store AWS keys with the application, where they can be difficult to rotate and where users can potentially extract them.

Roles: Example 1 of 3

Assuming a role

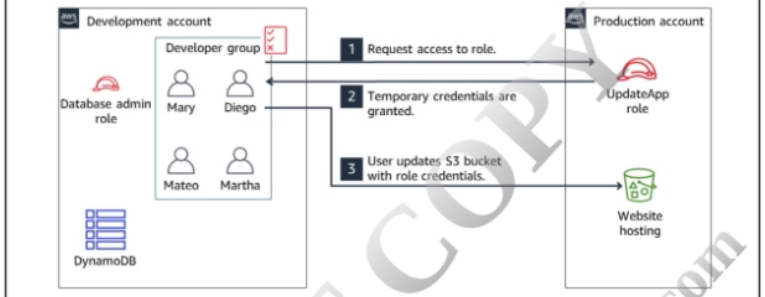
In addition to user and user-group management, IAM roles are also available. A user can assume a role and temporarily take on different permissions associated with the role. However, a role does not have any credentials (password or access keys) associated with it. Instead of being uniquely associated with one person, a role is intended to be assumable by anyone who requires it. For instance, a user of a development group can assume a Database admin role to obtain DynamoDB table write permissions not associated with the group policies.

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

- "IAM roles" (https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html)
- "AWS service role" (https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_terms-and-concepts.html#iam-term-service-role)
- "Creating a role to delegate permissions to an AWS service" (https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create_for-service.html)

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

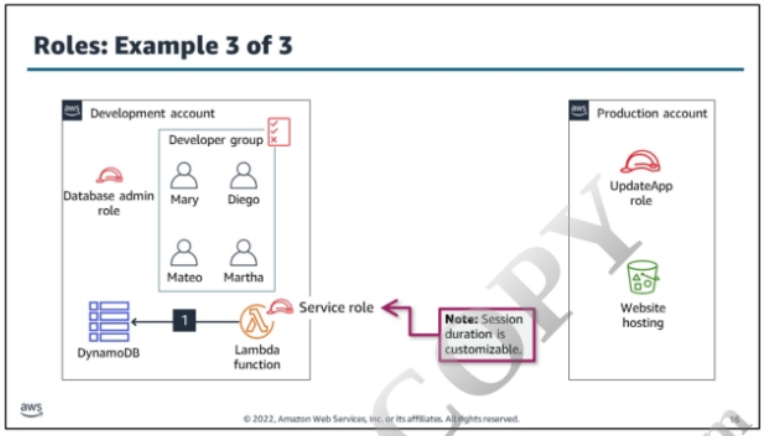
Roles: Example 2 of 3



Assuming a role

IAM roles also allow for cross-account access. For example, your organization may have multiple AWS accounts to isolate development and production environments. Users associated with one group are granted permissions to switch roles within your development account to request access to a role associated with the production account. The user switches (console) or assumes (AWS CLI) the role and obtains the temporary credentials to make changes to your production environment.

For more information, see "Providing access to an IAM user in another AWS account that you own" in the *AWS Identity and Access Management User Guide* (https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_common-scenarios_aws-accounts.html).

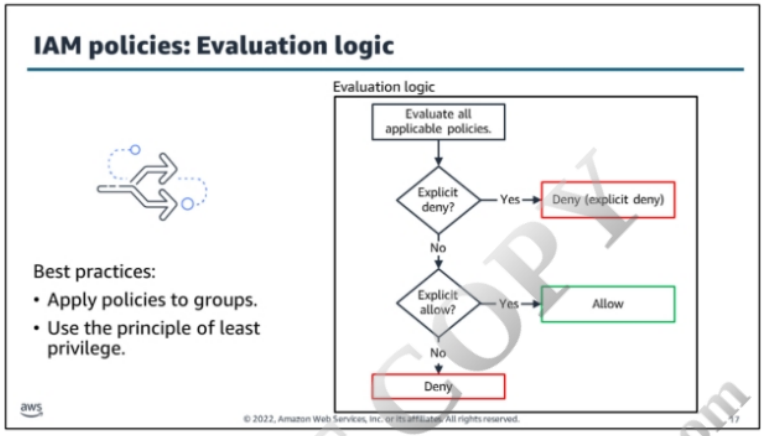


Assuming a role
Alternatively, a service, such as an AWS Lambda function, can assume a role to perform actions on a DynamoDB table on your behalf. This is called a *service role*.

For more information on IAM roles, see the following in the *AWS Identity and Access Management User Guide*:
- "IAM roles" (https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html)
- "AWS service role" (https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_terms-and-concepts.html#iam-term-service-role)
- "Creating a role to delegate permissions to an AWS service" (https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create_for-service.html)

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 4: Getting Started with Permissions



Use policies to fine-tune permissions granted to IAM users, groups, and roles. Because policies are in JSON format, you can use them with a version-control system. It's good practice to define least privilege access to each user, group, or role. Then, you can customize access to specific resources by using an authorization policy.

To determine whether a request for access should be allowed or denied, in general, IAM evaluates the policy as follows:

- By default, all requests are denied. (In general, requests made by using the account/root credentials for resources in the account are always allowed.)
- An explicit allow overrides this default.
- An explicit deny overrides any allows.

Evaluation logic
This diagram shows the logic of evaluating IAM policies. All policies that have been applied to the IAM entity are evaluated. If a conflict exists, the most restrictive policy is applied. For example, when one policy allows an action and another policy denies an action, the policy that denies the action is applied. The order in which the policies are evaluated has no effect on the outcome of the evaluation.

All policies are evaluated, and the result is always that the request is either allowed or denied. If

© 2022 Amazon Web Services, Inc. or its affiliates. All rights reserved.

there is an explicit deny statement, the final decision is to deny the action. Next, if there is an allow statement, the final decision is to allow the action. However, if there isn't an allow statement, the final decision is to deny it.

Any actions that you didn't explicitly allow are denied. Any actions that you explicitly deny are *always* denied.

For more information, see "Policy evaluation logic" in the *AWS Identity and Access Management User Guide* (http://docs.aws.amazon.com/IAM/latest/UserGuide/AccessPolicyLanguage_EvaluationLogic.html).

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 4: Getting Started with Permissions

In this demo we will create an S3 bucket using AWS CLI commands. `userwithpermissionboundary` is a member of an AWS managed policy group (`developers`). By default, this group allows members to create S3 buckets. However, `userwithpermissionboundary` has an attached permissions boundary whose maximum access does not support Amazon S3 access. We will then update `userwithpermissionboundary` to grant Amazon S3 access.

Assumptions:
AWS CLI is setup with multiple named profiles.
(https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-profiles.html)

Users:
- `userwithpermissionboundary`
- `userwithiamaccess`

Policies
- **PowerUserAccess** (https://console.aws.amazon.com/iam/home#/policies/arn:aws:iam::aws:policy/PowerUserAccess)
- **AdministratorAccess** (https://console.aws.amazon.com/iam/home#/policies/arn:aws:iam::aws:policy/AdministratorAccess)

s3elevated

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

S3restricted

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

Groups

Developers:
Permissions Policy = PowerUserAccess
userwithpermissionboundary

Admins
Permissions Policy = AdministratorAccess
Members = userwithiamaccess

User with permissions boundary attached:
userwithpermissionboundary = S3restricted

Demo start

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

1. Create a bucket with the user userwithpermissionboundary profile.

   ```
   aws s3 mb s3://bucketfordevonawsdemo05122021 --profile
   userwithpermissionboundary
   ```

2. Our attempt will fail to create the bucket. Let's investigate.

   ```
   aws iam get-user --user-name userwithpermissionboundary --profile
   userwithiamaccess
   ```

3. View the users' (userwithpermissionboundary) permissions boundary.

   ```
   arn:aws:iam::111122223333:policy/S3restricted.
   ```

4. Let's check the maximum level of access the permissions boundary allows.

   ```
   aws iam get-policy-version --policy-arn
   arn:aws:iam::111122223333:policy/S3restricted--version-id v1
   ```

5. We will then update the users' permissions boundary to a new permissions boundary that allows for S3 access.

   ```
   arn:aws:iam::1234567891011:policy/S3elevated
   ```

6. To accomplish the update, we will use the elevated permissions of another user (userwithiamaccess) to issue the put-user-permissions-boundary command. This command updates the user (userwithpermissionboundary) to a new permissions boundary that includes S3 access.

   ```
   aws iam put-user-permissions-boundary --permissions-boundary
   arn:aws:iam::111722413196:policy/S3elevated --user-name
   userwithpermissionboundary --profile userwithiamaccess
   ```

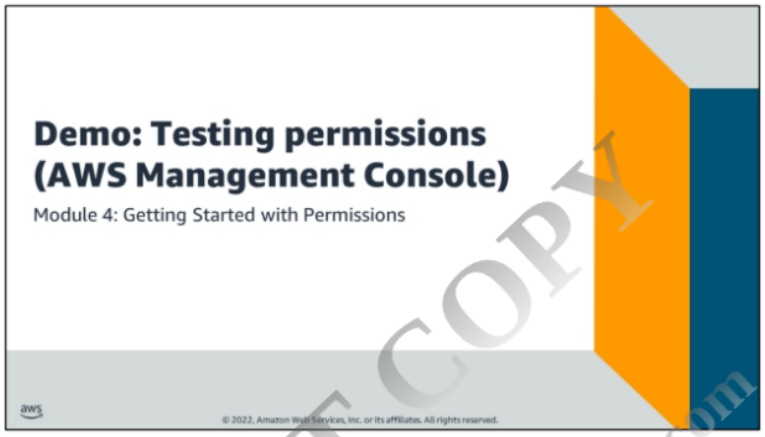
7. Let's determine whether the user has been updated to the new permissions boundary.

   ```
   aws iam get-user --user-name userwithpermissionboundary --profile
   userwithiamaccess
   ```

8. We will then again attempt to create an S3 bucket using AWS CLI commands with the same user.

   ```
   aws s3 mb s3://bucketfordevonawsdemo05122021 --profile
   userwithpermissionboundary
   ```

Success!



In this demo, we will sign in to the AWS Management Console as a user (contractor) who is granted access only to AWS CloudShell. We will attempt to create an S3 bucket using AWS CLI commands through an AWS CloudShell session, which will fail. To gain access to the Amazon S3 service and successfully create an S3 bucket, we will assume a role (`Contractors3access`), which allows for S3 bucket creation.

This demo illustrates the use of roles within the same organization.

Assumptions
The following assumptions are considered as we work through the demo:
Users
Contractor
Attached policy = none

Group
Contractor
Attached policy = `ContractorAccess`

Role
ContractorAccess
Attached policy = `ContractorS3access`

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

Policies
ContractorAccess

```
{
     "Version": "2012-10-17",
     "Statement": [
     {
          "Effect": "Allow",
          "Action": "cloudshell:*",
          "Resource": "*"
     },
     {
          "Effect": "Allow",
          "Action": "sts:AssumeRole",
          "Resource": "arn:aws:iam::112233445566:role/S3access"
     }
     ]
}
```

ContractorS3access

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

Demo start

1. Sign in to the AWS Management Console with the user "Contractor" and search for CloudShell.
2. Try to create an S3 bucket.

   ```
   aws s3 mb s3://DevonAWStest_bucket
   ```

3. Who am I logged in as?

   ```
   aws sts get-caller-identity
   ```

   Our attempt will fail to create the bucket.

4. Let's assume the assigned role **`ContractorAccess`** by first retrieving the temporary access information for the role.

```
aws sts assume-role --role-arn
"arn:aws:iam::111722413196:role/S3access" --role-session-name
DevOnAWS
```

The response includes access information required to assume the ContractorAccess: `AccessKeyId`, `SecretAccessKey`, and `SessionToken`

5. To assume the role, run the following three commands with the supplied `AccessKeyId`, `SecretAccessKey`, and `SessionToken` attached.

```
export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
export
AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
export AWS_SESSION_TOKEN=SessionToken
```

6. Let's determine whether we have assumed the role.

```
aws sts get-caller-identity
```

7. Try to create an S3 bucket again as the assumed role.

```
aws s3 mb s3://DevonAWStest_bucket
```

Success!

This example shows how you can manage access to your application by using roles.

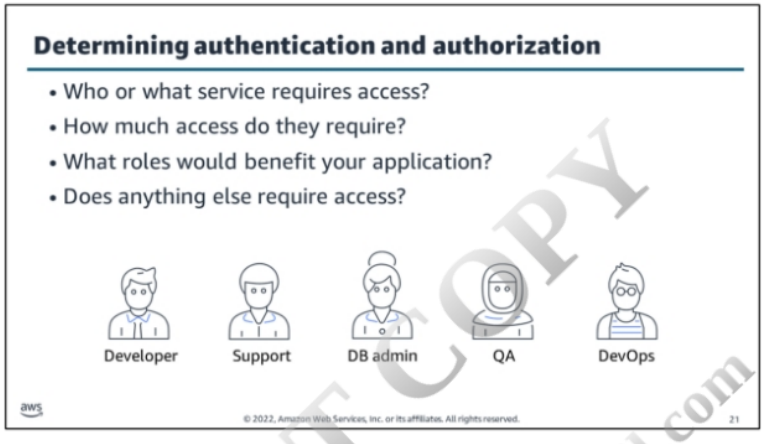
Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 4: Getting Started with Permissions

Product demonstration

- Testing IAM permissions

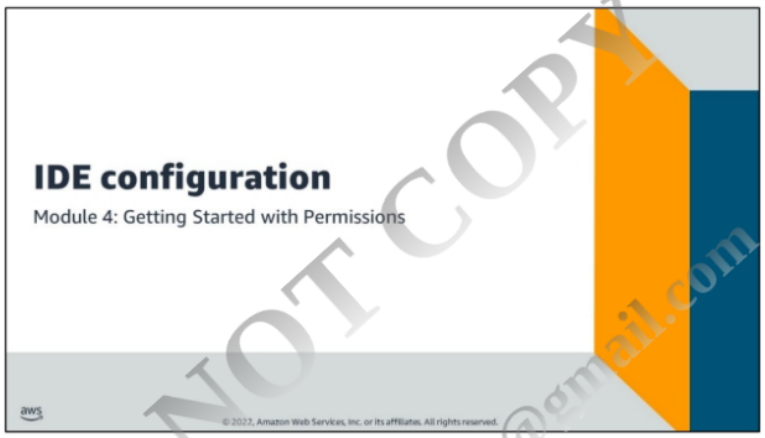
 20

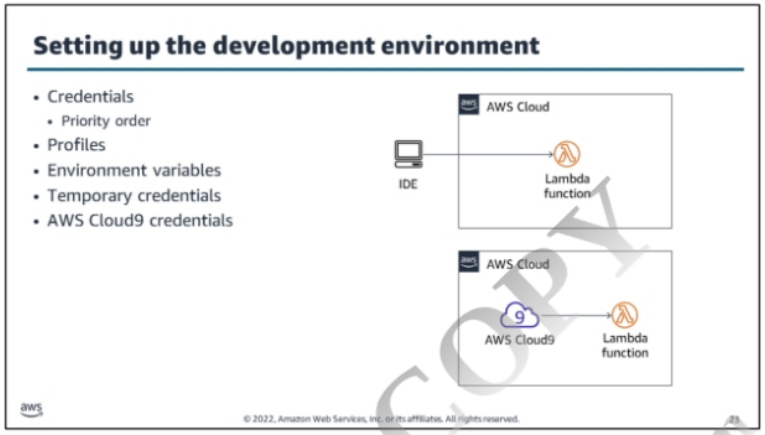


Access management begins with setting up users and groups to protect your resources. It also means connecting to other identity services to grant external users access to AWS resources. To design secure access to your applications, you must ask the following questions:

- Who or what service requires access to build, manage, or interact with your application?
- What level of access (policies and permissions) do they require to your application environment?
- Does everyone require full access to all services all the time? How would you manage your IAM principals?

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.





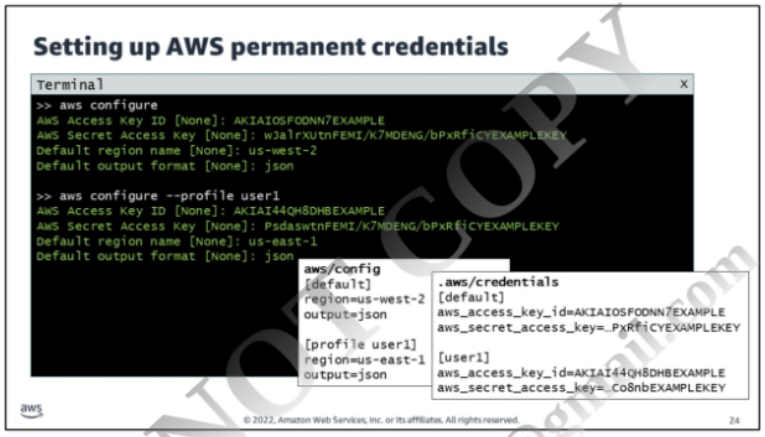
Performing application development tasks requires developers to configure their tools to suit their needs. Whether you are using a third-party IDE or AWS Cloud9 as your development tool, ensure that your settings are configured to enhance your development process.

You will begin by configuring credentials. Next, we examine priority order for credentials and named profiles so that you understand how they work with the AWS CLI and common IDEs. A *named profile* is a collection of settings and credentials that you can apply to an AWS CLI command. When you specify a profile to run a command, the settings and credentials are used to run that command.

Finally, temporary credentials and AWS Cloud9 credential options are discussed.

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 4: Getting Started with Permissions



To set up your AWS credentials:

1. On a terminal window, run the following command:
   `aws configure`
2. Paste in your user's AWS access key ID, then the AWS secret access key.
3. Enter a default region name.
4. Choose a default output format. Options include `json`, `yaml`, and `text`.

Resources

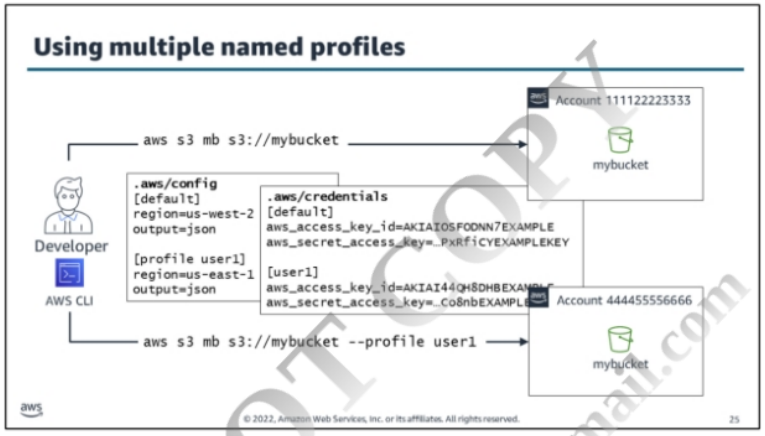
Locations for credentials and config files:

Linux, macOS, or Unix:

Windows:

`C:\Users\USERNAME\.aws\credentials`
`C:\Users\USERNAME\.aws\config`

For more information, see "Output format" in the *AWS Command Line Interface User Guide* (https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html#cli-configure-quickstart-format).



Using AWS CLI, you can configure multiple *named profiles* as a collection of settings and credentials used to interact with your AWS services. With named profiles, you can switch between accounts, users, roles, and regions. IDEs, such as VS Code, Eclipse, Visual Studio, and JetBrains, support named profiles. With this capability, developers can switch between environments.

For example, a developer is tasked with creating an S3 bucket in two accounts. Using the AWS CLI, they accomplish the task by issuing two commands. The first AWS CLI command, `aws s3 mb s3://mybucket`, is a standard S3 command that creates a bucket using the default profile. The second AWS CLI second, `aws s3 mb s3://mybucket --profile user1`, includes an additional `--profile` option, which specifies the user1 profile.

The AWS CLI and your IDE will use the default user unless the profile is changed to another user or role.

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

To add additional profiles to your config and credentials files, run `$ aws configure --profile`.

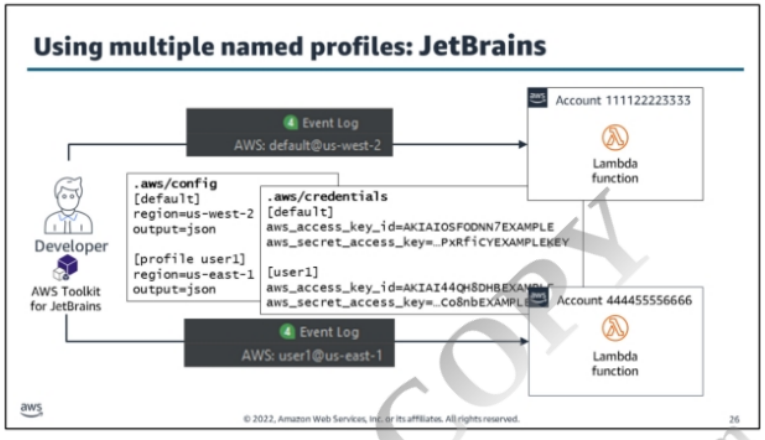
Credentials

- Linux and macOS
  `~/.aws/credentials`
- Windows
  `%USERPROFILE%\.aws\credentials [default]`

Config

- Linux and macOS
  `~/.aws/config`
- Windows
  `%USERPROFILE%\.aws\config`

For more information, see "Named profiles" in the *AWS Command Line Interface User Guide* (https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-profiles.html).



Switching profiles is also supported through your IDE when paired with AWS Toolkits. In this example, the developer deploys Lambda functions in multiple accounts by switching profiles from within their IDE. The AWS Toolkit for JetBrains streamlines the process through the Eclipse preferences pane.

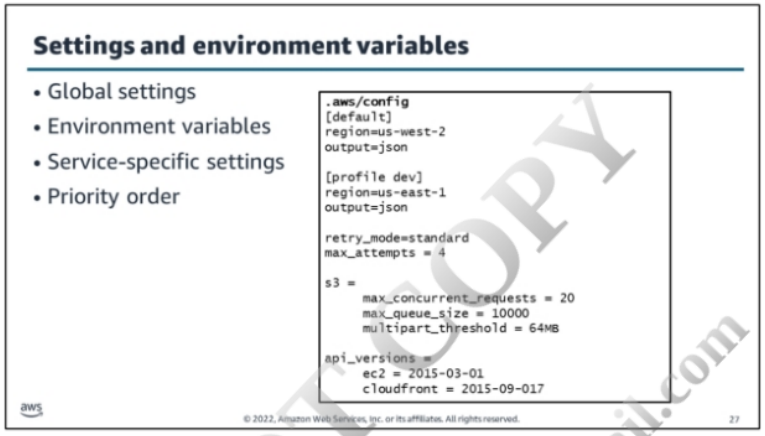
AWS offers toolkits for many commonly used IDEs.

For more information, see the following:

- (JetBrains) "Setting AWS credentials for the AWS Toolkit for JetBrains" in the *AWS Toolkit for JetBrains User Guide* (https://docs.aws.amazon.com/toolkit-for-jetbrains/latest/userguide/setup-credentials.html)
- (Eclipse) "Set up AWS Credentials" in the *AWS Toolkit for Eclipse User Guide* (https://docs.aws.amazon.com/toolkit-for-eclipse/v1/user-guide/setup-credentials.html)
- (Visual Studio) "Creating profiles for your AWS credentials" in the *AWS Toolkit for Visual Studio User Guide* (https://docs.aws.amazon.com/toolkit-for-visual-studio/latest/user-guide/keys-profiles-credentials.html)
- (Visual Studio) "Setting up your AWS credentials" in the *AWS Toolkit for Visual Studio Code User Guide* (https://docs.aws.amazon.com/toolkit-for-vscode/latest/userguide/setup-credentials.html)

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 4: Getting Started with Permissions



The config and credentials files contain additional settings that can be stored in the environment variables of your operating system. Although you can have only one set of environment variables in effect at a time, they are modified dynamically as your program runs and your requirements change.

In the example config file, the `profile user1` has additional environment variables to control the retry mode and maximum attempts on a request. Additional environment variables are also configured to optimize Amazon S3 operations.

Global settings affect all services. By contrast, environment variables affect only AWS SDKs and tools. You can also store settings specific to Amazon S3 in the config file.

Credential order

When an AWS SDK or AWS tool looks for credentials or a configuration setting, it invokes each credential provider in a certain order. The SDK or tool stops when it finds a value that it can use. Most AWS SDKs and tools check the credential providers in the following order:

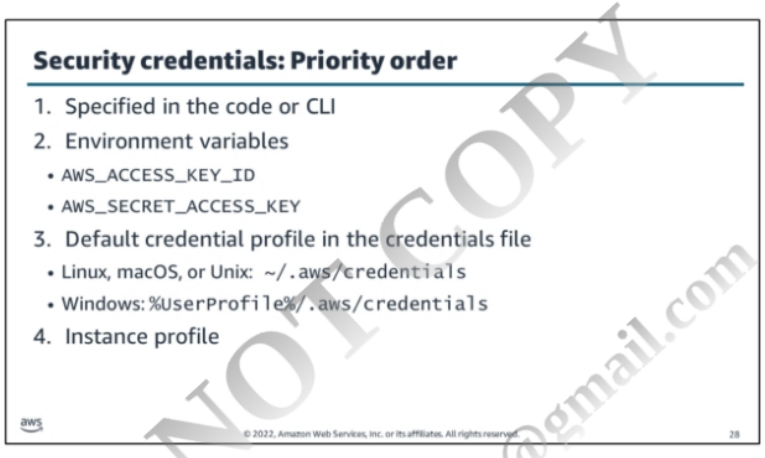
1. Per-operation parameter
2. Environment variable
3. Shared credentials file
4. Shared config file
5. Instance profile

For more information on settings and environment variables, see the following in the *AWS Reference Guide:*

- "Global settings for config and credentials files" (https://docs.aws.amazon.com/sdkref/latest/guide/settings-global.html)
- "Supported environment variables" (https://docs.aws.amazon.com/sdkref/latest/guide/environment-variables.html)

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 4: Getting Started with Permissions



To make requests to AWS, you must supply AWS credentials. Your code will find credentials to initialize a new service by following this priority order:

1. Credentials are specified in your code or CLI command.
   Providing credentials within code or a CLI command will override settings in any other location. You are enabled to also specify `--region`, `--output`, and `--profile` as parameters on the command line.

2. Environment variables
   You can store values in your system's environment variables.

3. Default credentials file
   The credentials and config files are updated when you run the command `aws configure`.

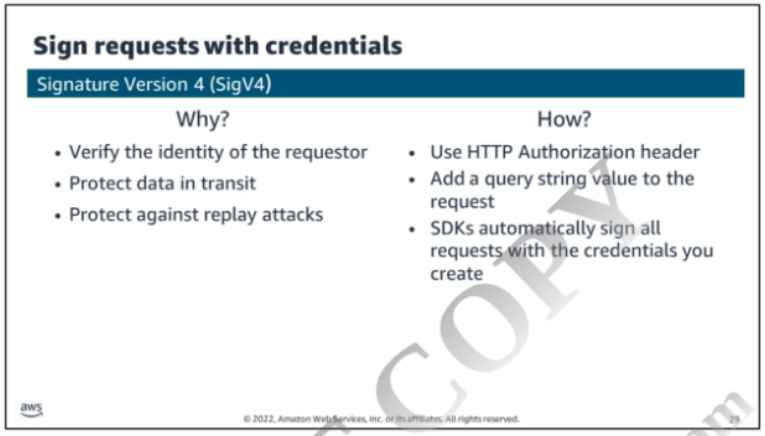
4. Instance profile
   You can associate an IAM role with each of your Amazon Elastic Compute Cloud (Amazon EC2) instances. Temporary credentials for that role are then available to code running in the instance.

For more information on credentials, see the following:

- (Java) "Working with AWS Credentials" in the *AWS SDK for Java Developer Guide*

(https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/credentials.html)
- (.NET) "Configuring AWS Credentials" in the *AWS SDK for .NET Developer Guide* (https://docs.aws.amazon.com/sdk-for-net/v3/developer-guide/net-dg-config-creds.html)
- (Boto3) "Credentials" in the Boto3 Documentation (https://boto3.amazonaws.com/v1/documentation/api/latest/guide/credentials.html)

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.



Signature Version 4 (SigV4) is the process to add authentication information to AWS API requests sent by HTTP. For security, most requests to AWS must be signed with an access key. The access key consists of an access key ID and secret access key, which are commonly referred to as your security credentials.

For more information, see "Signature Version 4 signing process" in the *AWS General Reference Guide* (https://docs.aws.amazon.com/general/latest/gr/signature-version-4.html).

Requests to AWS services must be signed. This means that they must contain information required to authenticate the requester. Requests are signed by using the access key ID and secret access key of an AWS account or of an IAM user.

Signing secures the request by doing the following:

- **Verifying the identity of the requester** – Signing ensures that the request has been issued by someone who has a valid access key ID and secret access key.
- **Protecting data in transit** – To prevent tampering with a request while it is in transit. Some request elements are used to calculate a hash (digest) of the request. The resulting hash value is included in the request. When AWS receives the request, it recalculates the hash based on the same information and matches it against the hash value that was included in the request. If the two hash values do not match, AWS denies the request.
- **Protecting against replay attacks** – To protect against replay attacks, all requests have a request expiration period (n minutes after the timestamp on the request). The exact duration of the request expiration period varies by service. If a request does not reach AWS within the expiration period, AWS denies the request.

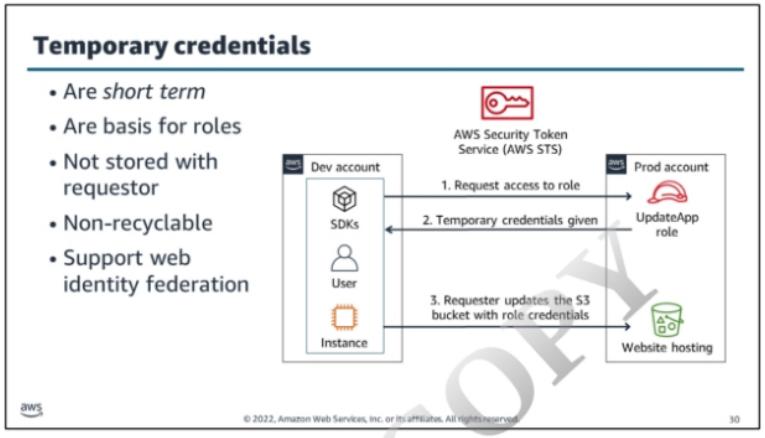
For additional security, transmit your requests using SSL by using HTTPS. SSL encrypts the transmission. This encryption protects your request or response from being viewed in transit.

You do not have to explicitly sign the requests when using one of the AWS SDKs, AWS CLI, or a service-specific CLI.

For more information, see "Signing AWS API requests" in the *AWS Reference Guide* (http://docs.aws.amazon.com/general/latest/gr/signing_aws_api_requests.html).

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 4: Getting Started with Permissions



Temporary credentials
Use AWS Security Token Service (AWS STS) to request limited permissions, temporary credentials for IAM users, or for users that you authenticate through identity federation.

You use security credentials to make programmatic requests for AWS resources using the AWS CLI or AWS SDKs or a service, such as an EC2 instance. The credentials are the same as a long-term credential. However, you must account for some considerations:

- **Temporary credentials are the basis for roles.** With AWS STS, you can specify the temporary credentials expiry interval. However, any service requests you make with expired credentials will fail, so you must request a new set of temporary credentials.
- **Temporary security credentials are not stored with the user.** Temporary security credentials are generated dynamically and provided to the user only when requested.

Web identity federation – You can allow users to sign in using a well-known third-party identity provider. Examples are: log in with Amazon, Facebook, Google, or any OpenID Connect (OIDC) 2.0 compatible provider. AWS STS web identity federation supports log in with Amazon, Facebook, Google, and any OpenID Connect (OIDC)-compatible identity provider.

Benefits

Some benefits of temporary credentials:

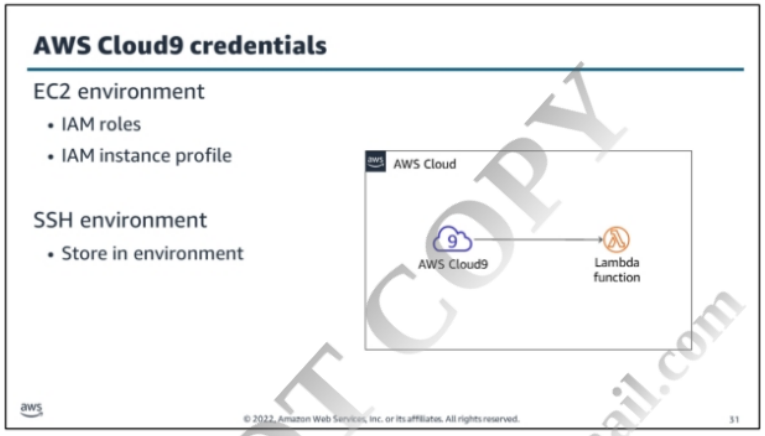
- No need to distribute or embed long-term AWS security credentials with an application.
- You can provide access to your AWS resources to users without having to define an AWS identity for them.
- After temporary security credentials expire, they cannot be reused. So, you do not have to rotate them or explicitly revoke them when they're no longer required.

For more information, see the following:

- "Temporary security credentials in IAM" in the *AWS Identity and Access Management User Guide* (https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_temp.html)
- "Actions" in the *AWS Security Token Service API Reference* (https://docs.aws.amazon.com/STS/latest/APIReference/API_Operations.html)
- "Using temporary security credentials with the AWS SDKs" in the *AWS Identity and Access Management User Guide* (https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_temp_use-resources.html#using-temp-creds-sdk)

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 4: Getting Started with Permissions



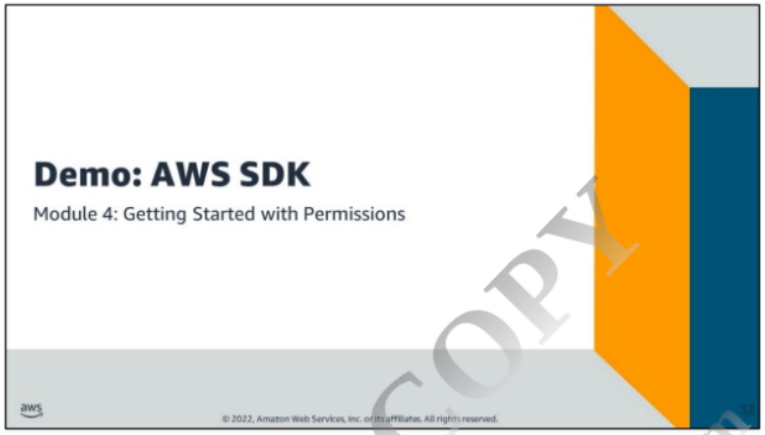
When using an AWS Cloud9 EC2 development environment instance or SSH environment, you provide credentials to your environment in a few ways. In an EC2 environment, we recommend creating IAM roles that provide temporary credentials and follow AWS best practices. Another option is to attach an IAM instance profile to the AWS Cloud9 instance. Although useful, this approach requires additional effort to create, manage, and attach the instance profile to the EC2 instance yourself.

For more information, see the following in the *AWS Cloud9 User Guide*

- "Create and use an instance profile to manage temporary credentials" (https://docs.aws.amazon.com/cloud9/latest/user-guide/credentials.html#credentials-temporary)
- "Calling AWS services from an environment in AWS Cloud9" (https://docs.aws.amazon.com/cloud9/latest/user-guide/credentials.html)
- "Create and store permanent access credentials in an Environment" (https://docs.aws.amazon.com/cloud9/latest/user-guide/credentials.html#credentials-permanent-create)

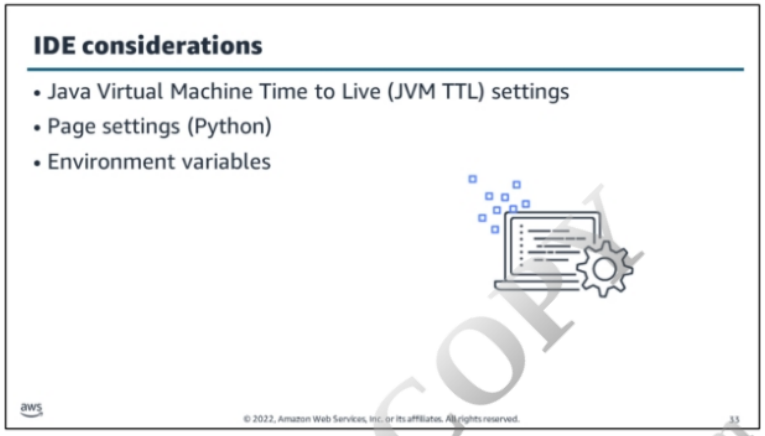
SSH environment

When using an SSH environment, you can store your access credentials within the environment. However, this method is considered less secure.



Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 4: Getting Started with Permissions



Performing application development tasks requires developers to configure their tools to suit their needs. Whether you are using a third-party IDE or AWS Cloud9 as your development tool, make sure that your settings are configured to enhance your development process.

JVM TTL settings (Java)
The Java virtual machine (JVM) caches DNS name lookups. Because AWS resources use DNS name entries that occasionally change, we recommend that you configure your JVM with a TTL value of no more than 60 seconds.

For more information, see "Setting the JVM TTL for DNS Name Lookups" in the *AWS SDK for Java Developer Guide* (https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/java-dg-jvm-ttl.html).

Environment variables
Environment variables provide another way to specify configuration options and credentials. They can be useful for scripting or temporarily setting a named profile as the default.

For additional information, see "Environment variables to configure the AWS CLI" in the *AWS Command Line Interface User Guide* (https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-envvars.html).

Error retries and exponential backoff

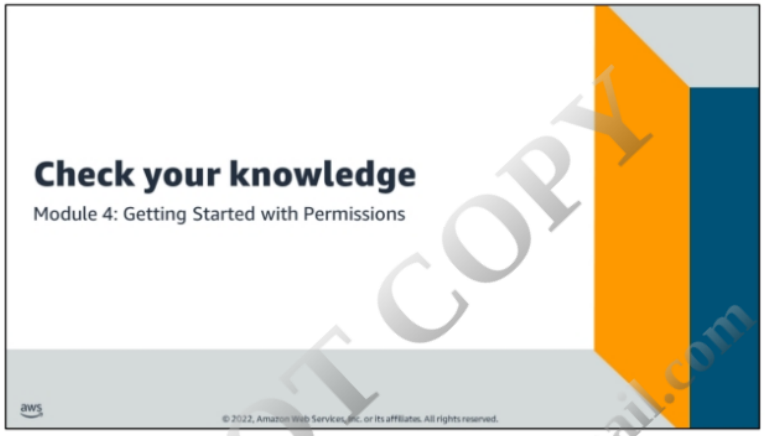
Numerous components on a network, such as DNS servers, switches, load balancers, and others, can generate errors anywhere in the life of a given request. The usual technique for dealing with these error responses in a networked environment is to implement retries in the client application.

In addition to retries, each AWS SDK implements an exponential backoff algorithm for better flow control. The idea behind exponential backoff is to use progressively longer waits between retries for consecutive error responses.

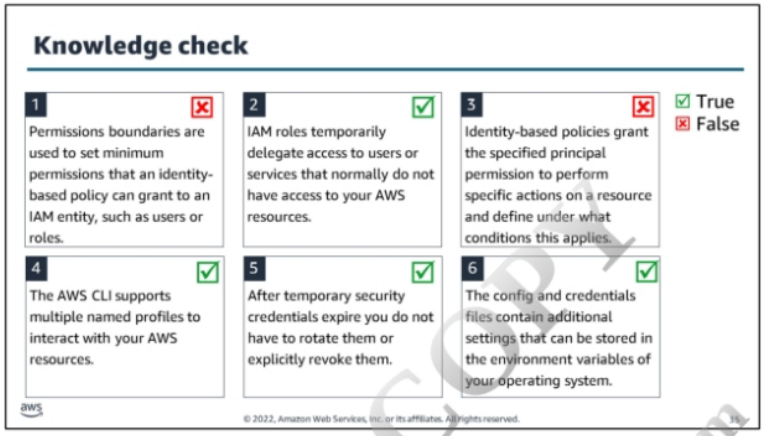
For more information, see "Error retries and exponential backoff in AWS" in the *AWS Reference Guide* (https://docs.aws.amazon.com/general/latest/gr/api-retries.html).

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 4: Getting Started with Permissions



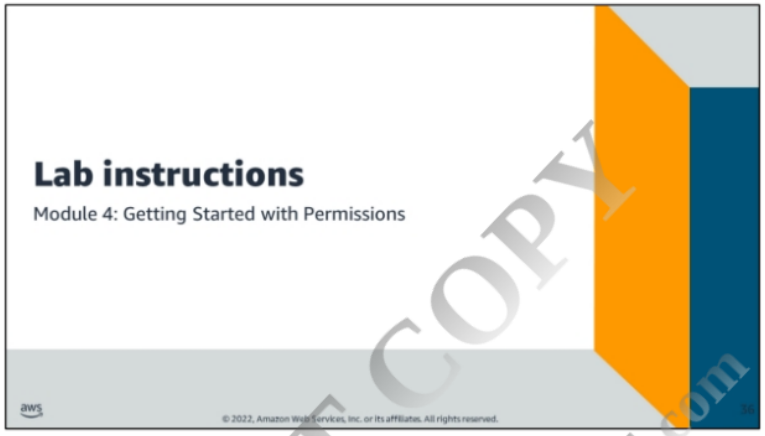
DO NOT COPY sameersheik68@gmail.com



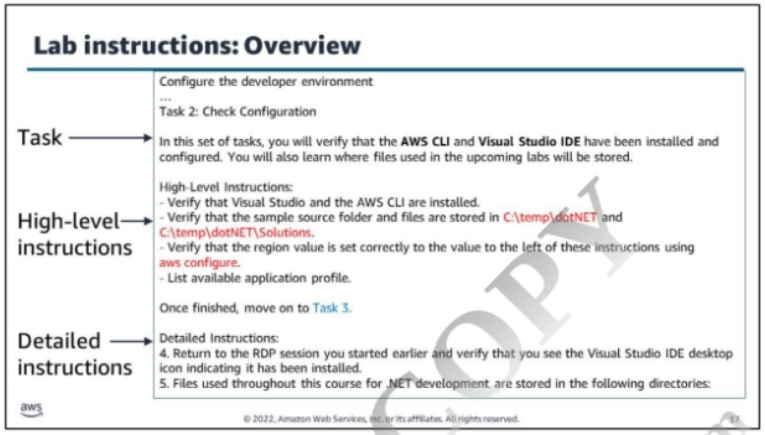
1. (False) Permissions boundaries are used to set **maximum** permissions that an identity-based policy can grant to an IAM entity, such as users or roles.
2. (True)
3. (False) **Resource-based** policies grant the specified principal permission to perform specific actions on that resource and defines under what conditions this applies.
4. (True)
5. (True)
6. (True)

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 4: Getting Started with Permissions



DO NOT COPY
sameersheik68@gmail.com

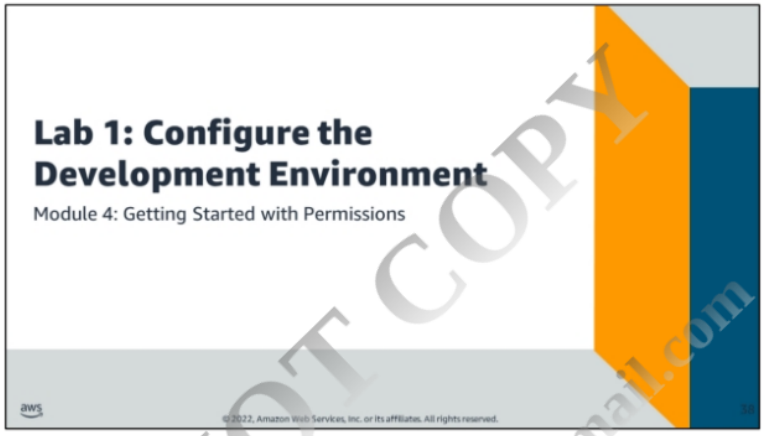


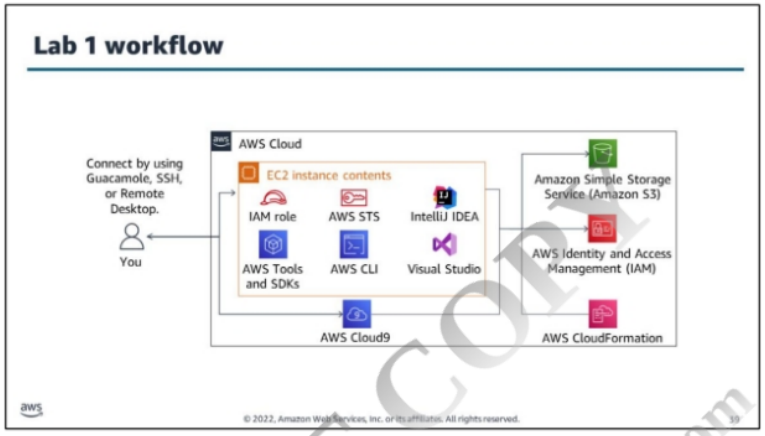
Each lab contains a series of **tasks** that make up an overall process. Each task describes the actions required to complete the procedure within the lab. Tasks may include multiple-choice options that are designed to test your knowledge.

Tasks are broken down into two levels. If you want to challenge yourself, you can follow the **high-level instructions.** These instructions provide hints to complete the task, but they do not include the detailed steps. If you successfully complete these high-level steps, you can skip the **Detailed Instructions** section, and proceed to the next task. The Detailed Instructions sections provide detailed steps to complete a task.

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 4: Getting Started with Permissions

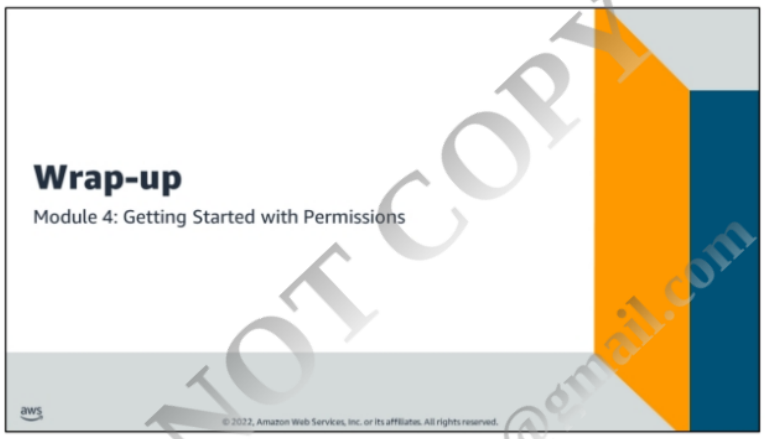




In this lab, you are going to connect to your sandbox environment that you will use to build out your end-to-end application for this course. You will verify that the appropriate development tools are installed and configured to access AWS services. You will review your specific IDE, learn how the AWS Toolkit works, and you will use AWS Identity and Access Management (IAM) to understand how permissions work.

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

AWS Training and Certification Module 4: Getting Started with Permissions



DO NOT COPY
sameersheik68@gmail.com

© 2022 Amazon Web Services, Inc. or its affiliates. All rights reserved.

Module summary

You are now able to:

- Review AWS Identity and Access Management (IAM) features and components
- Configure permissions to support a development environment
- Demonstrate how to test IAM permissions
- Configure your IDEs and SDKs to support a development environment
- Demonstrate accessing AWS services using SDKs and AWS Cloud9

 41

Printed by: sameersheik68@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

Thank you

Corrections, feedback, or other questions?
Contact us at https://support.aws.amazon.com/#/contacts/aws-training.
All trademarks are the property of their owners.

 42