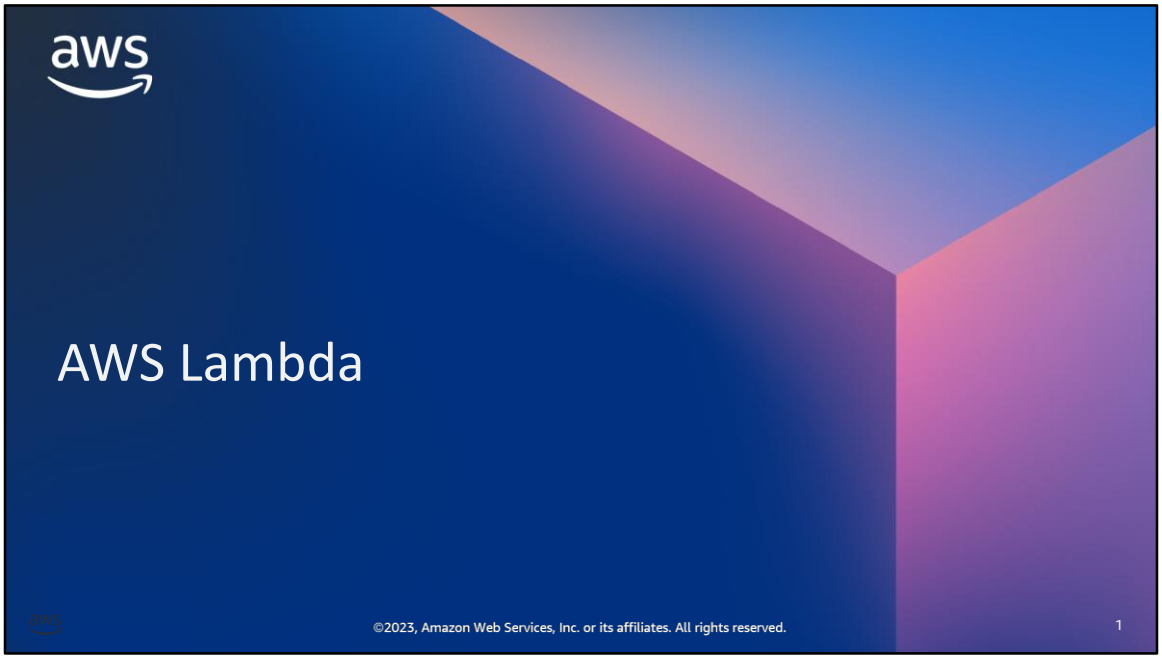


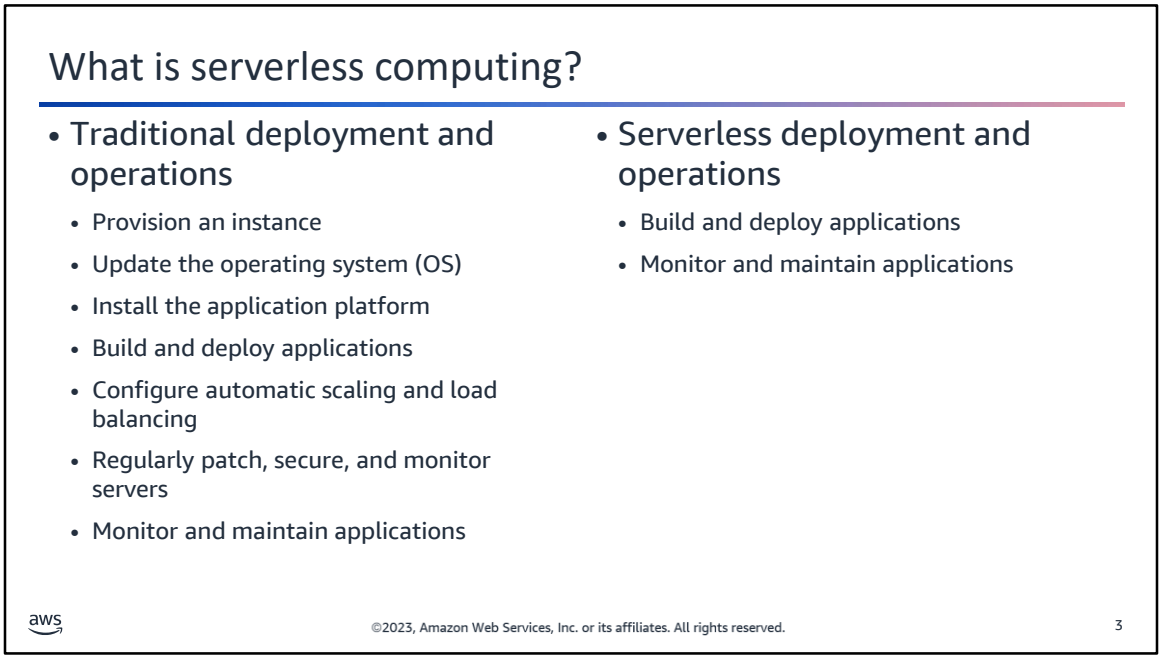
©2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

At the core of the lesson

You will learn how to do the following:

Define serverless computing and identify the purpose and functions of AWS Lambda.

aws
©2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Amazon Web Services (AWS) offers many compute options. One option is Amazon Elastic Compute Cloud (Amazon EC2), which provides virtual machines. A second option is a container solution, such as Amazon Elastic Container Service (Amazon ECS). However, a third approach to compute is available and does not require you to provision or manage servers. This third approach is referred to as serverless computing.
With serverless computing, you can build and run applications and services without thinking about servers. Serverless applications do not require you to provision, scale, or manage any servers.

What is AWS Lambda?

AWS Lambda

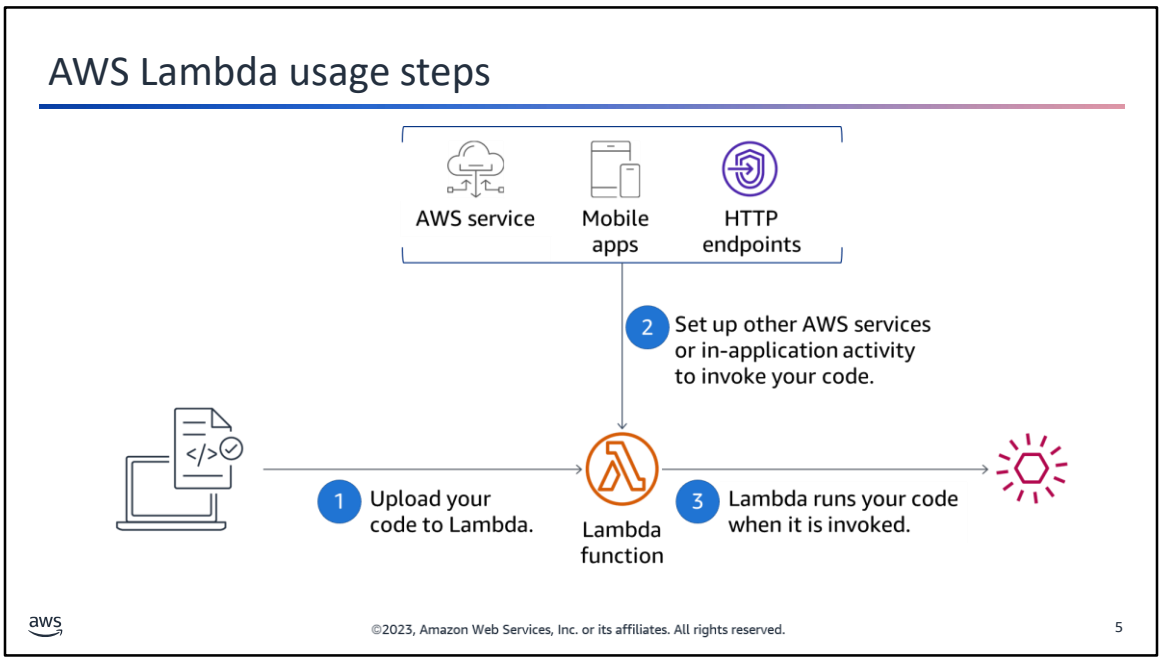
- Is a fully managed service for serverless compute
- Provides event-driven invocation
- Offers subsecond metering
- Limits the runtime of a function to a maximum of 15 minutes
- Supports multiple programming languages

aws

©2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

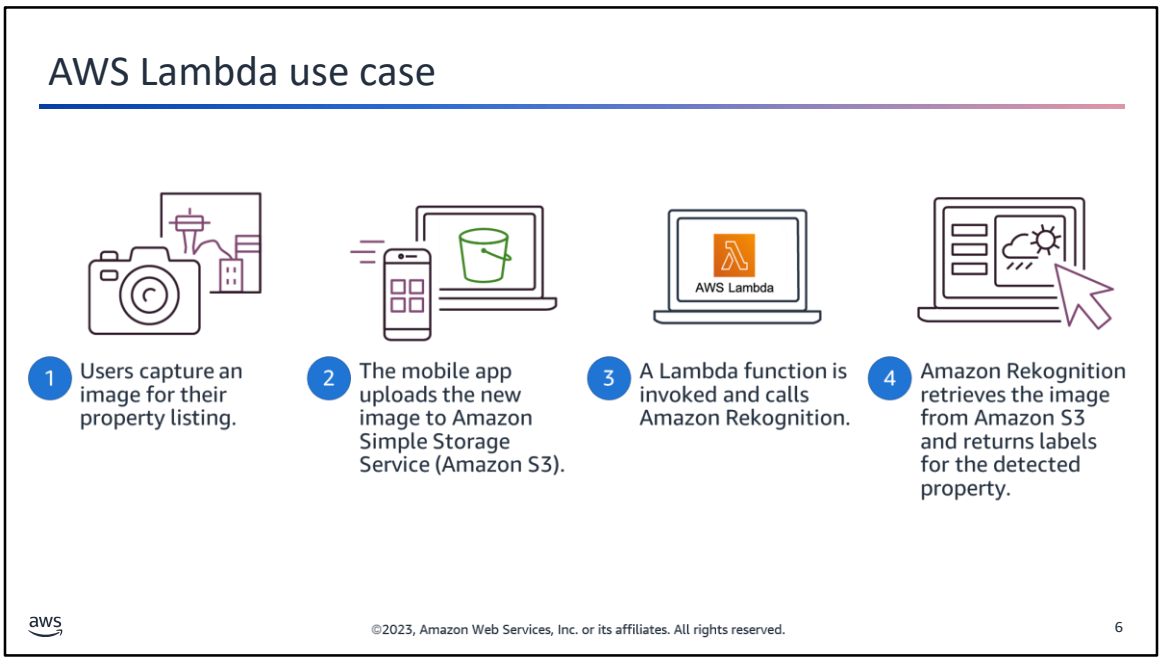
AWS provides serverless computing through the AWS Lambda service. You can create Lambda functions for nearly any type of application or backend service—all with minimal administration. You only need to upload your code, and Lambda handles everything that is required to run and scale your code with high availability. Lambda is a fully managed service for serverless compute. With Lambda, you run your code only when needed, and the service scales automatically to thousands of requests per second.

You can set up your code to automatically invoke from other AWS services, or you can call Lambda directly from any web or mobile application. The runtime of a Lambda function is limited to a maximum of 15 minutes. With Lambda, you do not need to learn any new languages, tools, or frameworks. You can use any third-party library, even native ones. Lambda includes support for Java, Node.js, C#, Python, Ruby, Go, and Powershell.



With Lambda, you can run code without provisioning or managing servers. The steps to use Lambda are as follows:

1. Upload your code to Lambda, and Lambda takes care of everything that is required to run and scale your code with high availability.
2. Set up your code to invoke from other AWS services, or invoke your code directly from any web or mobile application, or HTTP endpoint.
3. AWS Lambda runs your code only when invoked. You pay only for the compute time that you consume. You pay nothing when your code is not running.



Consider a use case where a Lambda function is invoked to process images that users upload to an application. First, users capture an image for their property by using an app on their mobile phone. The mobile app then uploads the new image to Amazon Simple Storage Service (Amazon S3). Adding this image to Amazon S3 invokes a Lambda function that calls Amazon Rekognition.

Amazon Rekognition can identify objects, people, text, scenes, and activities from images. The service provides facial analysis and recognition. In this example, Amazon Rekognition retrieves the image from Amazon S3 and returns labels for the property and its amenities.

Other Lambda use cases include the following:

- Automated backups
- Processing objects that are uploaded to Amazon S3
- Event-driven analysis of logs
- Event-driven transformations of data
- Internet of Things (IoT)
- Operating serverless websites

Steps to develop and deploy a Lambda function

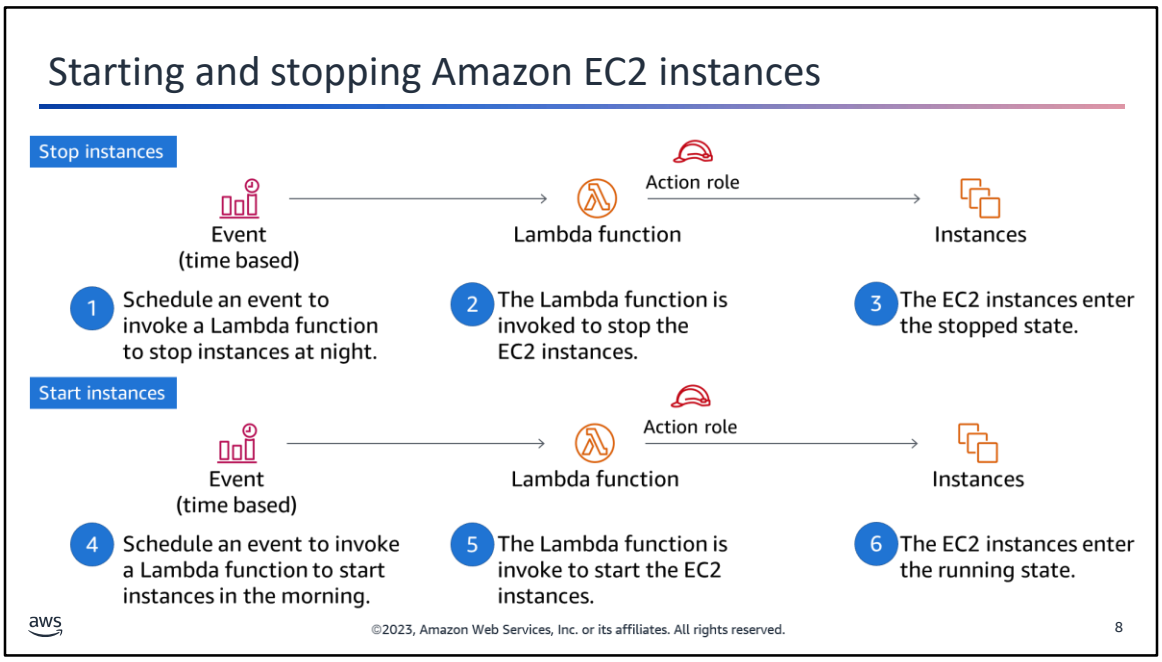
1. Define a handler class in the code for the function.
2. Create the Lambda function by using the AWS Management Console or the AWS Command Line Interface (AWS CLI).
3. Create and assign an AWS Identity and Access Management (IAM) role to the function. Include permissions to access the AWS services that are required.
4. Upload the code for the function.
5. Invoke the function to test it.
6. After you deploy the function to production, monitor it by using Amazon CloudWatch.

aws

©2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The process to develop and deploy a Lambda function consists of the following steps:

1. Define a Lambda handler class in your code. With the handler, you can specify where Lambda can begin running your code. For more information, see “Lambda Function Handler in Python” at https://docs.aws.amazon.com/lambda/latest/dg/python-handler.html.
2. Create the Lambda function. Think of the function as the code that you want to run. The function includes your code, associated configuration information, and resource requirements.
3. Configure access to resources by using AWS Identity and Access Management (IAM) roles and permissions. You can use security groups and network access control lists (ACLs) to provide your functions with access to your resources.
4. Upload your code.
5. Test the function, verify the results, and review your logs.
6. Monitor your Lambda functions through Amazon CloudWatch. Examples of metrics that you can track include the number of requests, latency, and the number of requests that result in errors.



Consider an example situation where you want to reduce usage of the Amazon Elastic Compute Cloud (Amazon EC2) service. You decide that you want to stop instances at a predefined time. For example, you want to stop EC2 instances at night (when less capacity is needed). Then, you want to start the instances back up in the morning (before the workday starts).

In this situation, you could configure Lambda and Amazon EventBridge to help you accomplish these actions automatically. The illustration on the slide shows what happens at each step:

1. You schedule an EventBridge event to run a Lambda function that stops your EC2 instances at night—for example, at 02:00 AM Coordinated Universal Time (UTC).
2. The Lambda function is invoked, and runs with the IAM role that grants it permissions to stop the EC2 instances. The EC2 instances enter the stopped state.
3. Later, at 05:00 AM UTC, an EventBridge event is scheduled to run a Lambda function to start the EC2 instances.
4. The Lambda function is invoked and runs with the IAM role that grants it permissions to start the EC2 instances.
5. The EC2 instances enter the running state.

For more information about creating stop and start functions with Lambda and EventBridge, see How do I stop and start Amazon EC2 instances at regular intervals using Lambda, https://aws.amazon.com/premiumsupport/knowledge-center/start-stop-lambda-eventbridge/.

AWS Lambda layers

By using layers, developers can do the following:

- Configure a Lambda function to use libraries that are not included in the deployment package.
- Keep the deployment package small.
- Avoid errors in code for package dependencies.
- Share libraries with other developers.

aws ©2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

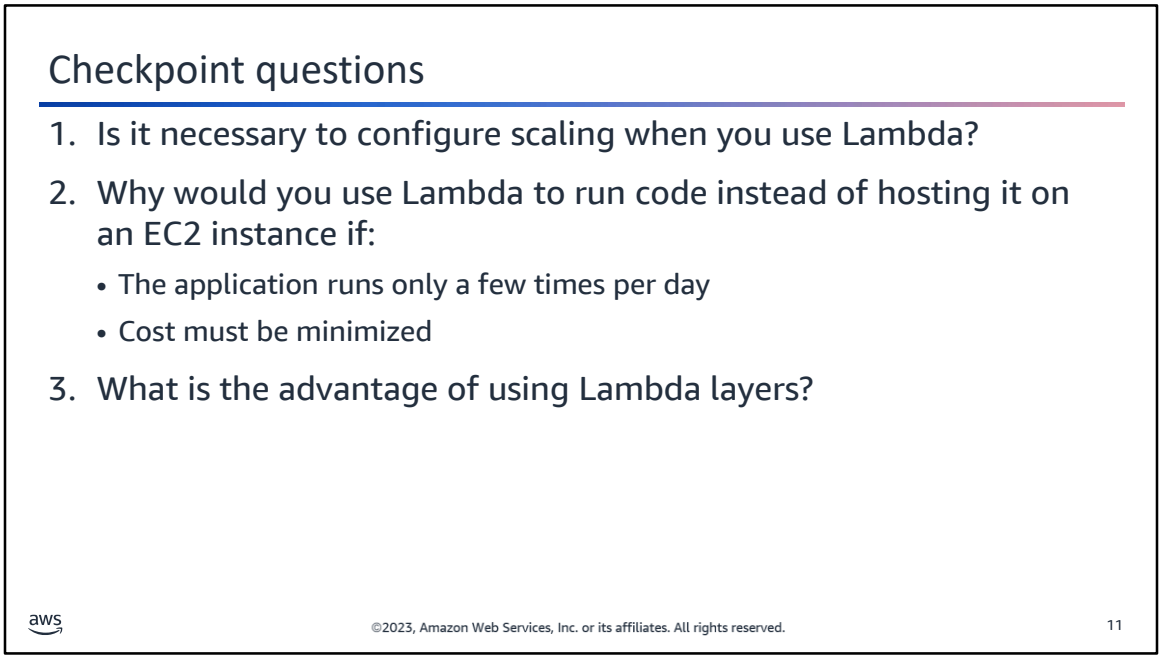
You can configure your Lambda function to pull in additional code and content in the form of *layers*. A Lambda layer is a .zip archive that contains libraries, data, configuration files, or a custom runtime.
By using layers, you can keep your deployment package small, which helps make development easier. You can avoid errors that might occur, such as package dependencies with your function code.
A function can use up to five layers at a time. The total unzipped size of the function and all its layers cannot exceed the unzipped deployment package size limit of 250 MB. Size limits are discussed on the next slide.
To add layers to a function, you would use the update-function-configuration command. For more information, see “Creating and Sharing Lambda Layers” at https://docs.aws.amazon.com/lambda/latest/dg/configuration-layers.html.

AWS Lambda quotas

- Quotas include the following:
  - Compute and storage resources
  - Function configuration, deployment, and execution
  - Lambda API requests
- A function that exceeds any of the limits will fail with an exceeded limits exception.

©2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Lambda sets quotas for the amount of compute and storage resources that you can use to run and store functions. For example, as of this writing, the maximum memory allocation for a single Lambda function is 10 GB. Lambda also sets quotas on concurrency and package size (250 MB as of this writing). If you are troubleshooting a Lambda deployment, keep these in mind.
By default, Lambda can handle up to 1,000 concurrent invocations in a single Region. For the current quota information, see “Lambda Quotas” at https://docs.aws.amazon.com/lambda/latest/dg/gettingstarted-limits.html.



The answers to the questions are as follows:

1. Is it necessary to configure scaling when you use Lambda?

   It is not necessary to configure Lambda for scaling. The service also automatically handles load balancing.

2. Why would you use Lambda to run code instead of hosting it on an EC2 instance if:
   - The application runs only a few times per day
   - Cost must be minimized

   You are charged only when Lambda runs. If your application runs only occasionally, it would be less expensive to host on Lambda instead of an EC2 instance that runs continuously.

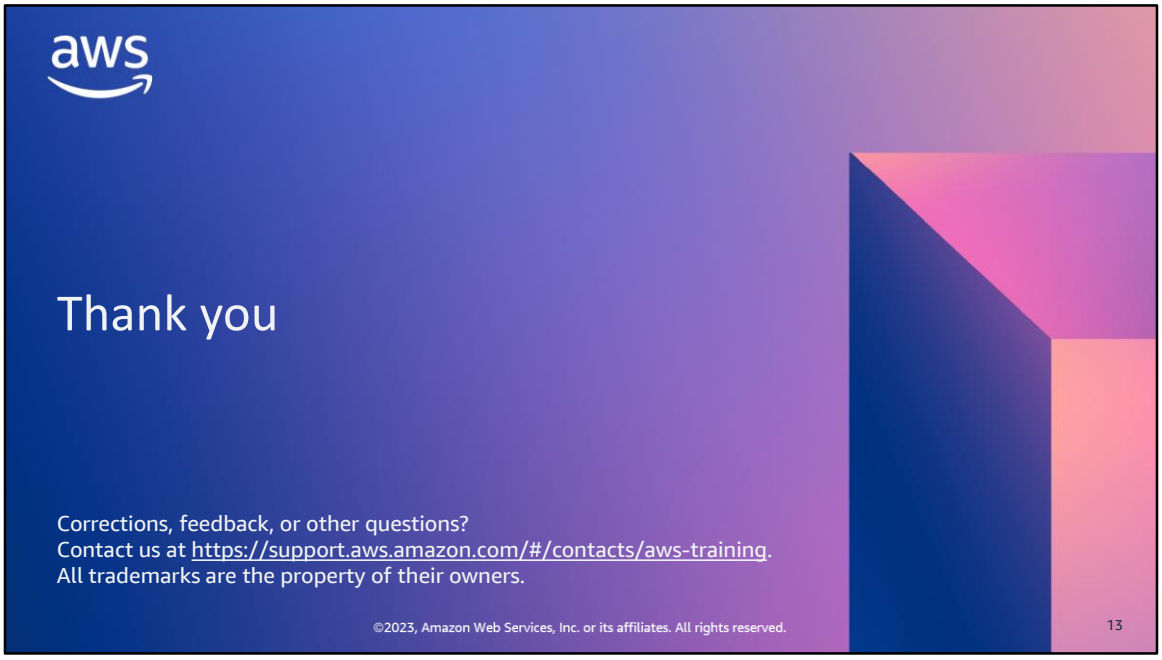
3. What is the advantage of using Lambda layers?

   By using layers, you can keep your deployment package small, which can make deployments easier.

Key ideas

- With serverless computing, you can build and run applications and services without provisioning or managing servers.
- AWS provides serverless computing through the AWS Lambda service.
- The maximum memory allocation for a single Lambda function is 10 GB.
- The maximum runtime for a Lambda function is 15 minutes.

aws
©2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.



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.

©2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.