AWS Lambda – FAQs

aws.amazon.com (https://aws.amazon.com/lambda/faqs/)

General

Q: What is AWS Lambda?

AWS Lambda lets you run code without provisioning or managing servers. You pay only for the compute time you consume - there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service - all with zero administration. Just upload your code and Lambda takes care of everything required to run and scale your code with high availability. You can set up your code to automatically trigger from other AWS services or call it directly from any web or mobile app.

Q: What is serverless computing?

Serverless computing allows you to build and run applications and services without thinking about servers. With serverless computing, your application still runs on servers, but all the server management is done by AWS. At the core of serverless computing is AWS Lambda, which lets you run your code without provisioning or managing servers. Learn more about serverless computing by visiting here.

Q: What events can trigger an AWS Lambda function?

Please see our documentation (http://docs.aws.amazon.com/lambda/latest/dg/intro-core-components.html#intro-core-components-event-sources) for a complete list of event sources.

Q: When should I use AWS Lambda versus Amazon EC2?

Amazon Web Services offers a set of compute services to meet a range of needs.

Amazon EC2 offers flexibility, with a wide range of instance types and the option to customize the operating system, network and security settings, and the entire software stack, allowing you to easily move existing applications to the cloud. With Amazon EC2 you are responsible for provisioning capacity, monitoring fleet health and performance, and designing for fault tolerance and scalability. AWS Elastic Beanstalk offers an easy-to-use service for deploying and scaling web applications in which you retain ownership and full control over the underlying EC2 instances. Amazon EC2 Container Service is a scalable management service that supports Docker containers and allows you to easily run distributed applications on a managed cluster of Amazon EC2 instances.

AWS Lambda makes it easy to execute code in response to events, such as changes to Amazon S3 buckets, updates to an Amazon DynamoDB table, or custom events generated by your applications or devices. With Lambda you do not have to provision your own instances; Lambda performs all the operational and administrative activities on your behalf, including capacity provisioning, monitoring fleet health, applying security patches to the underlying compute resources, deploying your code, running a web service front end, and monitoring and logging your code. AWS Lambda provides easy scaling and high availability to your code without additional effort on your part.

Q: What kind of code can run on AWS Lambda?

AWS Lambda offers an easy way to accomplish many activities in the cloud. For example, you can use AWS Lambda to build mobile back-ends that retrieve and transform data from Amazon DynamoDB, handlers that compress or transform objects as they are uploaded to Amazon S3, auditing and reporting of API calls made to any Amazon Web Service, and server-less processing of streaming data using Amazon Kinesis.

Q: What languages does AWS Lambda support?

AWS Lambda natively supports Java, Go, PowerShell, Node.js, C#, Python, and Ruby code, and provides a Runtime API which allows you to use any additional programming languages to author your functions. Please read our documentation on using Node.js (http://docs.aws.amazon.com/lambda/latest/dg/authoring-function-in-nodejs.html), Python (http://docs.aws.amazon.com/lambda/latest/dg/python-lambda.html), Java (http://docs.aws.amazon.com/lambda/latest/dg/java-lambda.html), Ruby (https://docs.aws.amazon.com/lambda/latest/dg/lambda-runtimes.html), C# (http://docs.aws.amazon.com/lambda/latest/dg/current-supported-versions.html), Go (https://docs.aws.amazon.com/lambda/latest/dg/go-programming-model.html) and PowerShell (https://docs.aws.amazon.com/lambda/latest/dg/powershell-programming-model.html).

Q: Can I access the infrastructure that AWS Lambda runs on?

No. AWS Lambda operates the compute infrastructure on your behalf, allowing it to perform health checks, apply security patches, and do other routine maintenance.

Q: How does AWS Lambda isolate my code?

Each AWS Lambda function runs in its own isolated environment, with its own resources and file system view. AWS Lambda uses the same techniques as Amazon EC2 to provide security and separation at the infrastructure and execution levels.

Q: How does AWS Lambda secure my code?

AWS Lambda stores code in Amazon S3 and encrypts it at rest. AWS Lambda performs additional integrity checks while your code is in use.

Q: What AWS regions are available for AWS Lambda?

Please refer to the AWS Global Infrastructure Region Table.

AWS Lambda functions

Q: What is an AWS Lambda function?

The code you run on AWS Lambda is uploaded as a “Lambda function”. Each function has associated configuration information, such as its name, description, entry point, and resource requirements. The code must be written in a “stateless” style i.e. it should assume there is no affinity to the underlying compute infrastructure. Local file system access, child processes, and similar artifacts may not extend beyond the lifetime of the request, and any persistent state should be stored in Amazon S3, Amazon DynamoDB, or another Internet-available storage service. Lambda functions can include libraries, even native ones.

Q: Will AWS Lambda reuse function instances?

To improve performance, AWS Lambda may choose to retain an instance of your function and reuse it to serve a subsequent request, rather than creating a new copy. To learn more about how Lambda reuses function instances, visit our documentation (http://docs.aws.amazon.com/lambda/latest/dg/lambda-introduction.html). Your code should not assume that this will always happen.

Q: What if I need scratch space on disk for my AWS Lambda function?

Each Lambda function receives 500MB of non-persistent disk space in its own /tmp directory.

Q: Why must AWS Lambda functions be stateless?

Keeping functions stateless enables AWS Lambda to rapidly launch as many copies of the function as needed to scale to the rate of incoming events. While AWS Lambda’s programming model is stateless, your code can access stateful data by calling other web services, such as Amazon S3 or Amazon DynamoDB.

Q: Can I use threads and processes in my AWS Lambda function code?

Yes. AWS Lambda allows you to use normal language and operating system features, such as creating additional threads and processes. Resources allocated to the Lambda function, including memory, execution time, disk, and network use, must be shared among all the threads/processes it uses. You can launch processes using any language supported by Amazon Linux.

Q: What restrictions apply to AWS Lambda function code?

Lambda attempts to impose as few restrictions as possible on normal language and operating system activities, but there are a few activities that are disabled: Inbound network connections are blocked by AWS Lambda, and for outbound connections only TCP/IP and UDP/IP sockets are supported, and ptrace (debugging) system calls are blocked. TCP port 25 traffic is also blocked as an anti-spam measure.

Q: How do I create an AWS Lambda function using the Lambda console?

If you are using Node.js or Python, you can author the code for your function using code editor in the AWS Lambda console which lets you author and test your functions, and view the results of function executions in a robust, IDE-like environment. Go to the console to get started (https://console.aws.amazon.com/lambda/home?region=us-east-1).

You can also package the code (and any dependent libraries) as a ZIP and upload it using the AWS Lambda console from your local environment or specify an Amazon S3 location where the ZIP file is located. Uploads must be no larger than 50MB (compressed). You can use the AWS Eclipse plugin to author and deploy Lambda functions in Java. You can use the Visual Studio plugin to author and deploy Lambda functions in C#, and Node.js.

Q: How do I create an AWS Lambda function using the Lambda CLI?

You can package the code (and any dependent libraries) as a ZIP and upload it using the AWS CLI from your local environment, or specify an Amazon S3 location where the ZIP file is located. Uploads must be no larger than 50MB (compressed). Visit the Lambda Getting Started guide (http://docs.aws.amazon.com/lambda/latest/dg/getting-started.html) to get started.

Q: Does AWS Lambda support environment variables?

Yes. You can easily create and modify environment variables from the AWS Lambda Console, CLI or SDKs. To learn more about environment variables, see the documentation (http://docs.aws.amazon.com/lambda/latest /dg/env_variables.html).

Q: Can I store sensitive information in environment variables?

For sensitive information, such as database passwords, we recommend you use client-side encryption using AWS Key Management Service (http://docs.aws.amazon.com/kms/latest/developerguide/overview.html) and store the resulting values as ciphertext in your environment variable. You will need to include logic in your AWS Lambda function code to decrypt these values.

Q: How can I manage my AWS Lambda functions?

You can easily list, delete, update, and monitor your Lambda functions using the dashboard in the AWS Lambda console. You can also use the AWS CLI and AWS SDK to manage your Lambda functions. Visit the Lambda Developer Guide (http://docs.aws.amazon.com/lambda/latest/dg/welcome.html) to learn more.

Q: Can I share code across functions?

Yes, you can package any code (frameworks, SDKs, libraries, and more) as a Lambda Layer (https://docs.aws.amazon.com/lambda/latest/dg/configuration-layers.html) and manage and share them easily across multiple functions.

Q: How do I monitor an AWS Lambda function?

AWS Lambda automatically monitors Lambda functions on your behalf, reporting real-time metrics through Amazon CloudWatch, including total requests, account-level and function-level concurrency usage, latency, error rates, and throttled requests. You can view statistics for each of your Lambda functions via the Amazon CloudWatch console or through the AWS Lambda console. You can also call third-party monitoring APIs in your Lambda function.

Visit Troubleshooting CloudWatch metrics (http://docs.aws.amazon.com/lambda/latest/dg/monitoring-functions.html) to learn more. Standard charges for AWS Lambda apply to use Lambda's built-in metrics.

Q: How do I troubleshoot failures in an AWS Lambda function?

AWS Lambda automatically integrates with Amazon CloudWatch logs, creating a log group for each Lambda function and providing basic application lifecycle event log entries, including logging the resources consumed for each use of that function. You can easily insert additional logging statements into your code. You can also call third-party logging APIs in your Lambda function. Visit Troubleshooting Lambda functions (http://docs.aws.amazon.com /lambda/latest/dg/monitoring-functions.html) to learn more. Amazon CloudWatch Logs rates will apply.

Q: How do I scale an AWS Lambda function?

You do not have to scale your Lambda functions – AWS Lambda scales them automatically on your behalf. Every time an event notification is received for your function, AWS Lambda quickly locates free capacity within its compute fleet and runs your code. Since your code is stateless, AWS Lambda can start as many copies of your function as needed without lengthy deployment and configuration delays. There are no fundamental limits to scaling a function. AWS Lambda will dynamically allocate capacity to match the rate of incoming events.

Q: How are compute resources assigned to an AWS Lambda function?

In the AWS Lambda resource model, you choose the amount of memory you want for your function, and are allocated proportional CPU power and other resources. For example, choosing 256MB of memory allocates approximately twice as much CPU power to your Lambda function as requesting 128MB of memory and half as much CPU power as choosing 512MB of memory. To learn more, see our Function Configuration documentation (https://docs.aws.amazon.com/lambda/latest/dg/resource-model.html).

You can set your memory in 64MB increments from 128MB to 3GB.

Q: How long can an AWS Lambda function execute?

AWS Lambda functions can be configured to run up to 15 minutes per execution. You can set the timeout to any value between 1 second and 15 minutes.

Q: How will I be charged for using AWS Lambda functions?

AWS Lambda is priced on a pay per use basis. Please see the AWS Lambda pricing page for details.

Q: Does AWS Lambda support versioning?

Yes. By default, each AWS Lambda function has a single, current version of the code. Clients of your Lambda function can call a specific version or get the latest implementation. Please read out documentation on versioning Lambda functions (http://docs.aws.amazon.com/lambda/latest/dg/versioning-aliases.html).

Q: How long after uploading my code will my AWS Lambda function be ready to call?

Deployment times may vary with the size of your code, but AWS Lambda functions are typically ready to call within seconds of upload.

Q: Can I use my own version of a supported library?

Yes. you can include your own copy of a library (including the AWS SDK) in order to use a different version than the default one provided by AWS Lambda.

Using AWS Lambda to process AWS events

Q: What is an event source?

An event source is an AWS service or developer-created application that produces events that trigger an AWS Lambda function to run. Some services publish these events to Lambda by invoking the cloud function directly (for example, Amazon S3). Lambda can also poll resources in other services that do not publish events to Lambda. For example, Lambda can pull records from an Amazon Kinesis stream or an Amazon SQS queue and execute a Lambda function for each fetched message.

Many other services, such as AWS CloudTrail, can act as event sources simply by logging to Amazon S3 and using S3 bucket notifications to trigger AWS Lambda functions.

Q: What event sources can be used with AWS Lambda?

Please see our documentation (http://docs.aws.amazon.com/lambda/latest/dg/intro-core-components.html#intro-core-components-event-sources) for a complete list of event sources.

Q: How are events represented in AWS Lambda?

Events are passed to a Lambda function as an event input parameter. For event sources where events arrive in batches, such as Amazon SQS, Amazon Kinesis, and Amazon DynamoDB Streams, the event parameter may contain multiple events in a single call, based on the batch size you request.To learn more about Amazon S3 event notifications visit Configuring Notifications for Amazon S3 Events (http://docs.aws.amazon.com/AmazonS3/latest/dev/NotificationHowTo.html). To learn more about Amazon DynamoDB Streams visit the DynamoDB Stream

Developers Guide (http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.html). To learn more about invoking Lambda functions using Amazon SNS, visit the Amazon SNS Developers Guide (http://docs.aws.amazon.com/sns/latest/dg/sns-lambda.html). For more information on Amazon Cognito events, visit Amazon Cognito. For more information on AWS CloudTrail logs and auditing API calls across AWS services, see AWS CloudTrail.

Q: How do I make an AWS Lambda function respond to changes in an Amazon S3 bucket?

From the AWS Lambda console, you can select a function and associate it with notifications from an Amazon S3 bucket. Alternatively, you can use the Amazon S3 console and configure the bucket's notifications to send to your AWS Lambda function. This same functionality is also available through the AWS SDK and CLI.

Q: How do I make an AWS Lambda function respond to updates in an Amazon DynamoDB table?

You can trigger a Lambda function on DynamoDB table updates by subscribing your Lambda function to the DynamoDB Stream associated with the table. You can associate a DynamoDB Stream with a Lambda function using the Amazon DynamoDB console, the AWS Lambda console or Lambda's registerEventSource API.

Q: How do I use an AWS Lambda function to process records in an Amazon Kinesis stream?

From the AWS Lambda console, you can select a Lambda function and associate it with an Amazon Kinesis stream owned by the same account. This same functionality is also available through the AWS SDK and CLI.

Q: How does AWS Lambda process data from Amazon Kinesis streams and Amazon DynamoDB Streams?

The Amazon Kinesis and DynamoDB Streams records sent to your AWS Lambda function are strictly serialized, per shard. This means that if you put two records in the same shard, Lambda guarantees that your Lambda function will be successfully invoked with the first record before it is invoked with the second record. If the invocation for one record times out, is throttled, or encounters any other error, Lambda will retry until it succeeds (or the record reaches its 24-hour expiration) before moving on to the next record. The ordering of records across different shards is not guaranteed, and processing of each shard happens in parallel.

Q: How do I use an AWS Lambda function to respond to notifications sent by Amazon Simple Notification Service (SNS)?

From the AWS Lambda console, you can select a Lambda function and associate it with an Amazon SNS topic. This same functionality is also available through the AWS SDK and CLI.

Q: How do I use an AWS Lambda function to respond to emails sent by Amazon Simple Email Service (SES)?

From the Amazon SES Console, you can set up your receipt rule to have Amazon SES deliver your messages to an AWS Lambda function. The same functionality is available through the AWS SDK and CLI.

Q: How do I use an AWS Lambda function to respond to Amazon CloudWatch alarms?
First, configure the alarm to send Amazon SNS notifications. Then from the AWS Lambda console, select a Lambda function and associate it with that Amazon SNS topic. See the Amazon CloudWatch Developer Guide (http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/WhatIsCloudWatch.html) for more on setting up Amazon CloudWatch alarms.

Q: How do I use an AWS Lambda function to respond to changes in user or device data managed by Amazon Cognito?

From the AWS Lambda console, you can select a function to trigger when any datasets associated with an Amazon Cognito identity pool are synchronized. This same functionality is also available through the AWS SDK and CLI. Visit Amazon Cognito for more information on using Amazon Cognito to share and synchronize data across a user's devices.

Q: How can my application trigger an AWS Lambda function directly?

You can invoke a Lambda function using a custom event through AWS Lambda's invoke API. Only the function's owner or another AWS account that the owner has granted permission can invoke the function. Visit the Lambda Developers Guide (http://docs.aws.amazon.com/lambda/latest/dg/welcome.html) to learn more.

Q: What is the latency of invoking an AWS Lambda function in response to an event?

AWS Lambda is designed to process events within milliseconds. Latency will be higher immediately after a Lambda function is created, updated, or if it has not been used recently.

Q: How do I create a mobile back-end using AWS Lambda?

You upload the code you want AWS Lambda to execute and then invoke it from your mobile app using the AWS Lambda SDK included in the AWS Mobile SDK. You can make both direct (synchronous) calls to retrieve or check data in real time as well as asynchronous calls. You can also define a custom API using Amazon API Gateway and invoke your Lambda functions through any REST compatible client. To learn more about the AWS Mobile SDK, visit the AWS Mobile SDK page. To learn more about Amazon API Gateway, visit the Amazon API Gateway page.

Q: How do I invoke an AWS Lambda function over HTTPS?

You can invoke a Lambda function over HTTPS by defining a custom RESTful API using Amazon API Gateway. This gives you an endpoint for your function which can respond to REST calls like GET, PUT and POST. Read more about using AWS Lambda with Amazon API Gateway.

Q: How can my AWS Lambda function customize its behavior to the device and app making the request?

When called through the AWS Mobile SDK, AWS Lambda functions automatically gain insight into the device and application that made the call through the 'context' object.

Q: How can my AWS Lambda function personalize their behavior based on the identity of the end user of an application?

When your app uses the Amazon Cognito identity, end users can authenticate themselves using a variety of public login providers such as Amazon, Facebook, Google, and other OpenID Connect-compatible services. User identity is then automatically and secured presented to your Lambda function in the form of an Amazon Cognito id, allowing it to access user data from Amazon Cognito, or as a key to store and retrieve data in Amazon DynamoDB or other web services.

Q: How do I create an Alexa skill using AWS Lambda?

AWS Lambda is integrated with the Alexa Skills Kit, a collection of self-service APIs, tools, documentation and code samples that make it easy for you to create voice-driven capabilities (or “skills”) for Alexa. You simply upload the Lambda function code for the new Alexa skill you are creating, and AWS Lambda does the rest, executing the code in response to Alexa voice interactions and automatically managing the compute resources on your behalf. Read the Alexa Skills Kit documentation for more details.

Q: What happens if my function fails while processing an event?

For Amazon S3 bucket notifications and custom events, AWS Lambda will attempt execution of your function three times in the event of an error condition in your code or if you exceed a service or resource limit.

For ordered event sources that AWS Lambda polls on your behalf, such as Amazon DynamoDB Streams and Amazon Kinesis streams, Lambda will continue attempting execution in the event of a developer code error until the data expires. You can monitor progress through the Amazon Kinesis and Amazon DynamoDB consoles and through the Amazon CloudWatch metrics that AWS Lambda generates for your function. You can also set Amazon CloudWatch alarms based on error or execution throttling rates.

Using AWS Lambda to build applications

Q: What is a serverless application?

Lambda-based applications (also referred to as serverless applications) are composed of functions triggered by events. A typical serverless application consists of one or more functions triggered by events such as object uploads to Amazon S3, Amazon SNS notifications, or API actions. These functions can stand alone or leverage other resources such as DynamoDB tables or Amazon S3 buckets. The most basic serverless application is simply a function.

Q: How do I deploy and manage a serverless application?

You can deploy and manage your serverless applications using the AWS Serverless Application Model (AWS SAM). AWS SAM is a specification that prescribes the rules for expressing serverless applications on AWS. This

specification aligns with the syntax used by AWS CloudFormation today and is supported natively within AWS CloudFormation as a set of resource types (referred to as "serverless resources"). These resources make it easier for AWS customers to use CloudFormation to configure and deploy serverless applications, using existing CloudFormation APIs.

Q: How can I discover existing serverless applications developed by the AWS community?

You can choose from a collection of serverless applications published by developers, companies, and partners in the AWS community with the AWS Serverless Application Repository. After finding an application, you can configure and deploy it straight from the Lambda console (https://console.aws.amazon.com/lambda/home?region=us-east-1).

Q: How do I automate deployment for a serverless application?

You can automate your serverless application's release process using AWS CodePipeline and AWS CodeDeploy. CodePipeline is a continuous delivery service that enables you to model, visualize and automate the steps required to release your serverless application. CodeDeploy provides a deployment automation engine for your Lambda-based applications. CodeDeploy lets you orchestrate deployments according to established best-practice methodologies such as canary and linear deployments, and helps you establish the necessary guardrails to verify that newly-deployed code is safe, stable, and ready to be fully released to production.

To learn more about serverless CI/CD, visit our documentation (http://docs.aws.amazon.com/lambda/latest /dg/automating-deployment.html).

Q: How do I get started on building a serverless application?

To get started, visit the AWS Lambda console and download one of our blueprints. The file you download will contain an AWS SAM file (which defines the AWS resources in your application), and a .ZIP file (which includes your function's code). You can then use AWS CloudFormation commands to package and deploy the serverless application that you just downloaded. For more details, visit our documentation (http://docs.aws.amazon.com /lambda/latest/dg/deploying-lambda-apps.html).

Q: How do I coordinate calls between multiple AWS Lambda functions?

You can use AWS Step Functions to coordinate a series of AWS Lambda functions in a specific order. You can invoke multiple Lambda functions sequentially, passing the output of one to the other, and/or in parallel, and Step Functions will maintain state during executions for you.

Q: How do I troubleshoot a serverless application?

You can enable your Lambda function for tracing with AWS X-Ray by adding X-Ray permissions to your Lambda function's execution role and changing your function's "tracing mode" to "active. " When X-Ray is enabled for your Lambda function, AWS Lambda will emit tracing information to X-Ray regarding the Lambda service overhead

incurred when invoking your function. This will provide you with insights such as Lambda service overhead, function init time, and function execution time. In addition, you can include the X-Ray SDK in your Lambda deployment package to create your own trace segments, annotate your traces, or view trace segments for downstream calls made from your Lambda function. X-Ray SDKs are currently available for Node.js and Java. Visit Troubleshooting Lambda-based applications (http://docs.aws.amazon.com/lambda/latest/dg/lambda-x-ray.html) to learn more. AWS X-Ray rates will apply.

Q: How is AWS SAM licensed?

The specification is open sourced under Apache 2.0, which allows you and others to adopt and incorporate AWS SAM into build, deployment, monitoring and management tools with a commercial-friendly license. You can access the AWS SAM repository on GitHub here (https://github.com/awslabs/serverless-application-specification).

Lambda@Edge

Q: What is Lambda@Edge?

Lambda@Edge allows you to run code across AWS locations globally without provisioning or managing servers, responding to end users at the lowest network latency. You just upload your Node.js code to AWS Lambda and configure your function to be triggered in response to Amazon CloudFront requests (i.e., when a viewer request lands, when a request is forwarded to or received back from the origin, and right before responding back to the end user). The code is then ready to execute across AWS locations globally when a request for content is received, and scales with the volume of CloudFront requests globally. Learn more in our documentation (http://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/lambda-at-the-edge.html).

Q: How do I use Lambda@Edge?

To use Lambda@Edge, you just upload your code to AWS Lambda and associate a function version to be triggered in response to Amazon CloudFront requests. Your code must satisfy the Lambda@Edge service limits. Lambda@Edge only supports Node.js for global invocation by CloudFront events at this time. Learn more in our documentation (http://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/lambda-at-the-edge.html).

Q: When should I use Lambda@Edge?

Lambda@Edge is optimized for latency sensitive use cases where your end viewers are distributed globally. All the information you need to make a decision should be available at the CloudFront edge, within the function and the request. This means that use cases where you are looking to make decisions on how to serve content based on user characteristics (e.g., location, client device, etc) can now be executed and served close to your users without having to be routed back to a centralized server.

Q: Can I deploy my existing Lambda functions for global invocation?

You can associate existing Lambda functions with CloudFront events for global invocation if the function satisfies the Lambda@Edge service requirements and limits. Read more here (http://docs.aws.amazon.com/lambda/latest/dg/API_UpdateFunctionConfiguration.html) on how to update your function properties.

Q: What Amazon CloudFront events can be used to trigger my functions?

Your functions will automatically trigger in response to the following Amazon CloudFront events:

- Viewer Request - This event occurs when an end user or a device on the Internet makes an HTTP(S) request to CloudFront, and the request arrives at the edge location closest to that user.
- Viewer Response - This event occurs when the CloudFront server at the edge is ready to respond to the end user or the device that made the request.
- Origin Request - This event occurs when the CloudFront edge server does not already have the requested object in its cache, and the viewer request is ready to be sent to your backend origin webserver (e.g. Amazon EC2, or Application Load Balancer, or Amazon S3).
- Origin Response - This event occurs when the CloudFront server at the edge receives a response from your backend origin webserver.

Q: How is AWS Lambda@Edge different from using AWS Lambda behind Amazon API Gateway?

The difference is that API Gateway and Lambda are regional services. Using Lambda@Edge (https://aws.amazon.com/lambda/edge/) and Amazon CloudFront (https://aws.amazon.com/cloudfront/) allows you to execute logic across multiple AWS locations based on where your end viewers are located.

Scalability and availability

Q: How available are AWS Lambda functions?

AWS Lambda is designed to use replication and redundancy to provide high availability for both the service itself and for the Lambda functions it operates. There are no maintenance windows or scheduled downtimes for either.

Q: Do my AWS Lambda functions remain available when I change my code or its configuration?

Yes. When you update a Lambda function, there will be a brief window of time, typically less than a minute, when requests could be served by either the old or the new version of your function.

Q: Is there a limit to the number of AWS Lambda functions I can execute at once?

No. AWS Lambda is designed to run many instances of your functions in parallel. However, AWS Lambda has a default safety throttle for number of concurrent executions per account per region (visit here (http://docs.aws.amazon.com/lambda/latest/dg/concurrent-executions.html#concurrent-execution-safety-limit) for info on default safety throttle limits). You can also control the maximum concurrent executions for individual AWS Lambda functions which you can use to reserve a subset of your account concurrency limit for critical functions, or

cap traffic rates to downstream resources.

If you wish to submit a request to increase the throttle limit you can visit our Support Center (https://aws.amazon.com/support), click "Open a new case," and file a service limit increase request.

Q: What happens if my account exceeds the default throttle limit on concurrent executions?

On exceeding the throttle limit, AWS Lambda functions being invoked synchronously will return a throttling error (429 error code). Lambda functions being invoked asynchronously can absorb reasonable bursts of traffic for approximately 15-30 minutes, after which incoming events will be rejected as throttled. In case the Lambda function is being invoked in response to Amazon S3 events, events rejected by AWS Lambda may be retained and retried by S3 for 24 hours. Events from Amazon Kinesis streams and Amazon DynamoDB streams are retried until the Lambda function succeeds or the data expires. Amazon Kinesis and Amazon DynamoDB Streams retain data for 24 hours.

Q: Is the default limit applied on a per function level?

No, the default limit only applies at an account level.

Q: What happens if my Lambda function fails during processing an event?

On failure, Lambda functions being invoked synchronously will respond with an exception. Lambda functions being invoked asynchronously are retried at least 3 times. Events from Amazon Kinesis streams and Amazon DynamoDB streams are retried until the Lambda function succeeds or the data expires. Kinesis and DynamoDB Streams retain data for a minimum of 24 hours.

Q: What happens if my Lambda function invocations exhaust the available policy?

On exceeding the retry policy for asynchronous invocations, you can configure a “dead letter queue” (DLQ) into which the event will be placed; in the absence of a configured DLQ the event may be rejected. On exceeding the retry policy for stream based invocations, the data would have already expired and therefore rejected.

Q: What resources can I configure as a dead letter queue for a Lambda function?

You can configure an Amazon SQS queue or an Amazon SNS topic as your dead letter queue.

Security and access control

Q: How do I allow my AWS Lambda function access to other AWS resources?

You grant permissions to your Lambda function to access other resources using an IAM role. AWS Lambda assumes the role while executing your Lambda function, so you always retain full, secure control of exactly which AWS resources it can use. Visit Setting up AWS Lambda (http://docs.aws.amazon.com/lambda/latest/dg/setting-

up.html) to learn more about roles.

Q: How do I control which Amazon S3 buckets can call which AWS Lambda functions?

When you configure an Amazon S3 bucket to send messages to an AWS Lambda function a resource policy rule will a be created that grants access. Visit the Lambda Developer's Guide (http://docs.aws.amazon.com/lambda/latest/dg/welcome.html) to learn more about resource policies and access controls for Lambda functions.

Q: How do I control which Amazon DynamoDB table or Amazon Kinesis stream an AWS Lambda function can poll?

Access controls are managed through the Lambda function's role. The role you assign to your Lambda function also determines which resource(s) AWS Lambda can poll on its behalf. Visit the Lambda Developer's Guide (http://docs.aws.amazon.com/lambda/latest/dg/welcome.html) to learn more.

Q: How do I control which Amazon SQS queue an AWS Lambda function can poll?

Access controls can be managed by the Lambda function's role or a resource policy setting on the queue itself. If both policies are present, the more restrictive of the two permissions will be applied.

Q: Can I access resources behind Amazon VPC with my AWS Lambda function?

Yes. You can access resources behind Amazon VPC.

Q: How do I enable and disable the VPC support for my Lambda function?

To enable VPC support, you need to specify one or more subnets in a single VPC and a security group as part of your function configuration. To disable VPC support, you need to update the function configuration and specify an empty list for the subnet and security group. You can change these settings using the AWS APIs, CLI, or AWS Lambda Management Console.

Q: Can a single Lambda function have access to multiple VPCs?

No. Lambda functions provide access only to a single VPC. If multiple subnets are specified, they must all be in the same VPC. You can connect to other VPCs by peering your VPCs.

Q: Can Lambda functions in a VPC also be able to access the internet and AWS Service endpoints?

Lambda functions configured to access resources in a particular VPC will not have access to the internet as a default configuration. If you need access to external endpoints, you will need to create a NAT (http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/vpc-nat-gateway.html) in your VPC to forward this traffic and configure your security group to allow this outbound traffic.

AWS Lambda functions in Java

Q: How do I compile my AWS Lambda function Java code?

You can use standard tools like Maven or Gradle to compile your Lambda function. Your build process should mimic the same build process you would use to compile any Java code that depends on the AWS SDK. Run your Java compiler tool on your source files and include the AWS SDK 1.9 or later with transitive dependencies on your classpath. For more details, see our documentation (http://docs.aws.amazon.com/lambda/latest/dg/java-lambda.html).

Q: What is the JVM environment Lambda uses for execution of my function?

Lambda provides the Amazon Linux build of openjdk 1.8.

AWS Lambda functions in Node.js

Q: Can I use packages with AWS Lambda?

Yes. You can use NPM packages as well as custom packages. Learn more here (https://aws.amazon.com/blogs/compute/nodejs-packages-in-lambda/).

Q: Can I execute other programs from within my AWS Lambda function written in Node.js?

Yes. Lambda's built-in sandbox lets you run batch ("shell") scripts, other language runtimes, utility routines, and executables. Learn more here (https://aws.amazon.com/blogs/compute/running-executables-in-aws-lambda/).

Q: Is it possible to use native modules with AWS Lambda functions written in Node.js?

Yes. Any statically linked native module can be included in the ZIP file you upload, as well as dynamically linked modules compiled with an rpath pointing to your Lambda function root directory. Learn more here (https://aws.amazon.com/blogs/compute/nodejs-packages-in-lambda/).

Q: Can I execute binaries with AWS Lambda written in Node.js?

Yes. You can use Node.js' child_process command to execute a binary that you've included in your function or any executable from Amazon Linux that is visible to your function. Alternatively several NPM packages exist that wrap command line binaries such as node-ffmpeg. Learn more here (https://aws.amazon.com/blogs/compute/running-executables-in-aws-lambda/).

Q: How do I deploy AWS Lambda function code written in Node.js?

To deploy a Lambda function written in Node.js, simply package your Javascript code and dependent libraries as a ZIP. You can upload the ZIP from your local environment, or specify an Amazon S3 location where the ZIP file is located. For more details, see our documentation (http://docs.aws.amazon.com/lambda/latest/dg/authoring-function-in-nodejs.html).

AWS Lambda functions in Python

Q: Can I use Python packages with AWS Lambda?

Yes. You can use pip to install any Python packages needed.

AWS Lambda functions in C#

Q: How do I package and deploy an AWS Lambda function in C#?

You can create a C# Lambda function using the Visual Studio IDE by selecting "Publish to AWS Lambda" in the Solution Explorer. Alternatively, you can directly run the "dotnet lambda publish" command from the dotnet CLI which has the [# Lambda CLI tools patch] installed, which creates a ZIP of your C# source code along with all NuGet dependencies as well as your own published DLL assemblies, and automatically uploads it to AWS Lambda using the runtime parameter “dotnetcore1.0”

AWS Lambda functions in PowerShell

Q: How do I deploy AWS Lambda function code written in PowerShell?

A PowerShell Lambda deployment package is a ZIP file that contains your PowerShell script, PowerShell modules that are required for your PowerShell script, and the assemblies needed to host PowerShell Core. You then use the *AWSLambdaPSCore* PowerShell module that you can install from the PowerShell Gallery (https://www.powershellgallery.com/packages/AWSLambdaPSCore/1.1.0.0) to create your PowerShell Lambda deployment package.

Q: How do I deploy AWS Lambda function code written in PowerShell? A PowerShell Lambda deployment package is a ZIP file that contains your PowerShell script, PowerShell modules that are required for your PowerShell script, and the assemblies needed to host PowerShell Core. You then use the *AWSLambdaPSCore* PowerShell module that you can install from the PowerShell Gallery to create your PowerShell Lambda deployment package.
Q: How do I deploy AWS Lambda function code written in PowerShell? A PowerShell Lambda deployment package is a ZIP file that contains your PowerShell script, PowerShell modules that are required for your PowerShell script, and the assemblies needed to host PowerShell Core. You then use the *AWSLambdaPSCore* PowerShell module that you can install from the PowerShell Gallery to create your PowerShell Lambda deployment package.

Q: How do I package and deploy an AWS Lambda function in Go?

Upload your Go executable artifact as a ZIP file through the AWS CLI or Lambda console and select the go1.x runtime. With Lambda, you can use Go's native tools to build and package your code. For more details, read our documentation (https://docs.aws.amazon.com/lambda/latest/dg/go-programming-model.html).

Q: How do I deploy AWS Lambda function code written in Ruby?

To deploy a Lambda function written in Ruby, package your Ruby code and gems as a ZIP. You can upload the ZIP from your local environment, or specify an Amazon S3 location where the ZIP file is located.

Other topics

Q: Which versions of Amazon Linux, Node.js, Python, JDK, .NET Core, SDKs, and additional libraries does AWS Lambda support?

You can view the list of supported versions here (http://docs.aws.amazon.com/lambda/latest/dg/current-supported-versions.html).

Q: Can I change the version of Amazon Linux or any language runtime?

No. AWS Lambda offers a single version of the operating system and managed language runtime to all users of the service. You can bring your own language runtime (https://docs.amazon.com/lambda/latest/dg/runtimes-custom.html) to use in Lambda.

Q: How can I record and audit calls made to the AWS Lambda API?

AWS Lambda is integrated with AWS CloudTrail. AWS CloudTrail can record and deliver log files to your Amazon S3 bucket describing the API usage of your account.

Q: How do I coordinate calls between multiple Lambda functions?

You can use Amazon Step Functions to coordinate multiple invoking Lambda functions. You can invoke multiple Lambda functions serially, passing the output of one to the other, or in parallel. See our documentation (http://docs.aws.amazon.com/step-functions/latest/dg/hello-lambda.html) for more details.

Learn more about AWS Lambda pricing

Visit the pricing page
Ready to get started?
Sign up (https://portal.aws.amazon.com/gp/aws/developer/registration/index.html)
Have more questions?
Contact us (https://aws.amazon.com/contact-us/)
FAQs
General AWS Lambda Functions Using AWS Lambda to process AWS events Using AWS Lambda to build applications Lambda@Edge Scalability and availability Security and access control AWS Lambda functions in Java AWS Lambda functions in Node.js AWS Lambda functions in Python AWS Lambda functions in C# AWS Lambda functions in PowerShell AWS Lambda functions in Go AWS Lambda functions in Ruby Other topics

aws.amazon.com (https://aws.amazon.com/lambda/faqs/)