AWS Lambda

AWS Lambda Cheat Sheet for the AWS Certified Solutions Architect Associate (SAA-C02) exam. This AWS cheat sheet contains detailed exam-specific facts to help you pass your AWS Certified Solutions Architect exam.

AWS Lambda Concepts

AWS Lambda lets you run code as functions without provisioning or managing servers.

Lambda-based applications (also referred to as serverless applications) are composed of functions triggered by events.

With serverless computing, your application still runs on servers, but all the server management is done by AWS.

You cannot log in to the compute instances that run Lambda functions or customize the operating system or language runtime.

Lambda functions:

* Consist of code and any associated dependencies.
* Configuration information is associated with the function.
* You specify the configuration information when you create the function.
* API provided for updating configuration data.

You specify the amount of memory you need allocated to your Lambda functions.

AWS Lambda allocates CPU power proportional to the memory you specify using the same ratio as a general purpose EC2 instance type.

Functions can access:

* AWS services or non-AWS services.
* AWS services running in VPCs (e.g. RedShift, Elasticache, RDS instances).
* Non-AWS services running on EC2 instances in an AWS VPC.

To enable your Lambda function to access resources inside your private VPC, you must provide additional VPC-specific configuration information that includes VPC subnet IDs and security group IDs.

AWS Lambda uses this information to set up elastic network interfaces (ENIs) that enable your function.

Compute resources:

* You can request additional memory in 64MB increments from 128MB to 3008MB.
* Functions larger than 1536MB are allocated multiple CPU threads, and multi-threaded or multi-process code is needed to take advantage.

There is a maximum execution timeout.

* Max is 15 minutes (900 seconds), default is 3 seconds.
* You pay for the time it runs.
* Lambda terminates the function at the timeout.

Code is invoked using API calls made using AWS SDKs.

Lambda assumes an IAM role when it executes the function.

The handler name refers to the method in your code where Lambda begins execution.

The components of AWS Lambda are:

* A Lambda function which is comprised of your custom code and any dependent libraries.
* Event sources such as SNS or a custom service that triggers your function and executes its logic.
* Downstream resources such as DynamoDB or Amazon S3 buckets that your Lambda function calls once it is triggered.
* Log streams are custom logging statements that allow you to analyze the execution flow and performance of your Lambda function.

Prefer to learn through doing? In the video tutorial below, you’ll learn what AWS Lambda is, how it works, and how to create a simple Hello World function that writes an event to Amazon CloudWatch Logs.

Lambda is an event-driven compute service where AWS Lambda runs code in response to events such as a changes to data in an S3 bucket or a DynamoDB table.

An event source is an AWS service or developer-created application that produces events that trigger an AWS Lambda function to run.

Event sources are mapped to Lambda functions.

Event sources maintain the mapping configuration except for stream-based services (e.g. DynamoDB, Kinesis) for which the configuration is made on the Lambda side and Lambda performs the polling.

Supported AWS event sources include:

* [**Amazon S3**](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-s3).
* [**Amazon DynamoDB**](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-dynamo-db).
* [**Amazon Kinesis Data Streams**](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-kinesis-streams).
* [**Amazon Simple Notification Service**](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-sns).
* [**Amazon Simple Email Service**](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-ses).
* [**Amazon Simple Queue Service**](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-sqs).
* [**Amazon Cognito**](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-cognito).
* [**AWS CloudFormation**](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-cloudformation).
* [**Amazon CloudWatch Logs**](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-cloudwatch-logs).
* [**Amazon CloudWatch Events**](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-cloudwatch-events).
* [**AWS CodeCommit**](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-codecommit).
* [**Scheduled Events (powered by Amazon CloudWatch Events)**](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-scheduled-events).
* [**AWS Config**](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-config).
* [**Amazon Alexa**](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-echo).
* [**Amazon Lex**](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-lex).
* [**Amazon API Gateway**](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-api-gateway).
* [**AWS IoT Button**](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-iot-button).
* [**Amazon CloudFront**](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-cloudfront).
* [**Amazon Kinesis Data Firehose**](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#supported-event-source-kinesis-firehose).
* [**Other Event Sources: Invoking a Lambda Function On Demand**](https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html#api-gateway-with-lambda).

Other event sources can invoke Lambda functions on-demand (application needs permissions to invoke the Lambda function).

Lambda can run code in response to HTTP requests using Amazon API gateway or API calls made using the AWS SDKs.

AWS Lambda supports code written in Node.js (JavaScript), Python, Java (Java 8 compatible), C# (.NET Core), Ruby, Go and PowerShell.

AWS Lambda stores code in Amazon S3 and encrypts it at rest.

Continuous scaling – scales out not up.

Lambda scales concurrently executing functions up to your default limit (1000).

Lambda functions are serverless and independent, 1 event = 1 function.

Functions can trigger other functions so 1 event can trigger multiple functions.

For non stream-based event sources each published event is a unit of work, run in parallel up to your account limit (one Lambda function per event).

For stream-based event sources the number of shards indicates the unit of concurrency (one function per shard).

Lambda works globally.

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.

Lambda functions provide access only to a single VPC. If multiple subnets are specified, they must all be in the same VPC.

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 in your VPC to forward this traffic and configure your security group to allow this outbound traffic.

Versioning can be used to run different versions of your code.

Each Lambda function has a unique Amazon Resource Name (ARN) which cannot be changed after publishing.

Use cases fall within the following categories:

* Using Lambda functions with AWS services as event sources.
* On-demand Lambda function invocation over HTTPS using Amazon API Gateway (custom REST API and endpoint).
* On-demand Lambda function invocation using custom applications (mobile, web apps, clients) and AWS SDKs, AWS Mobile SDKs, and the AWS Mobile SDK for Android.
* Scheduled events can be configured to run code on a scheduled basis through the AWS Lambda Console.

Building Lambda Apps

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”).

You can automate your serverless application’s release process using AWS CodePipeline and AWS CodeDeploy.

You can enable your Lambda function for tracing with AWS X-Ray.

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.

Lambda@Edge lets you run Node.js and Python Lambda functions to customize content that CloudFront delivers, executing the functions in AWS locations closer to the viewer.

The functions run in response to CloudFront events, without provisioning or managing servers. You can use Lambda functions to change CloudFront requests and responses at the following points:

* After CloudFront receives a request from a viewer (viewer request).
* Before CloudFront forwards the request to the origin (origin request).
* After CloudFront receives the response from the origin (origin response).
* Before CloudFront forwards the response to the viewer (viewer response).

You just upload your Node.js code to AWS Lambda and configure your function to be triggered in response to an Amazon CloudFront request.

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.

Limits

Memory – minimum 128MB, maximum 3008MB in 64MB increments.

Ephemeral disk capacity (/tmp space) per invocation – 512 MB.

Number of file descriptors – 1024.

Number of processes and threads (combined) – 1024.

Maximum execution duration per request – 900 seconds.

Concurrent executions per account – 1000 (soft limit).

Operations and Monitoring

Lambda automatically monitors Lambda functions and reports metrics through CloudWatch.

Lambda tracks the number of requests, the latency per request, and the number of requests resulting in an error.

You can view the request rates and error rates using the AWS Lambda Console, the CloudWatch console, and other AWS resources.

X-Ray is an AWS service that can be used to detect, analyse and optimise performance issues with Lambda applications.

X-Ray collects metadata from the Lambda service and any upstream and downstream services that make up your application.

Lambda is integrated with CloudTrail for capturing API calls and can deliver log files to your S3 bucket.

Charges

Priced based on:

* Number of requests. First 1 million are free then $0.20 per 1 million.
* Duration. Calculated from the time your code begins execution until it returns or terminates. Depends on the amount of memory allocated to a function.