Security Overview of AWS Lambda

An In-Depth Look at Lambda Security

March 2019

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents AWS's current product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS's products or services are provided "as is" without warranties, representations, or conditions of any kind, whether express or implied. AWS's responsibilities and liabilities to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Contents

Abstract

This whitepaper presents a deep dive of the AWS Lambda service through a security lens. It provides a well-rounded picture of the service, which can be useful for new adopters, as well as deepening understanding of AWS Lambda for current users.

The intended audience for this whitepaper is Chief Information Security Officers (CISOs), information security groups, security analysts, enterprise architects, compliance teams, and any others interested in understanding the underpinnings of AWS Lambda.

Introduction

Today, more workloads are using AWS Lambda to achieve scalability, performance, and cost efficiency, without managing the underlying infrastructure. These workloads scale to thousands of concurrent requests per second, making AWS Lambda one of the many important services that is offered by AWS today. AWS Lambda is used by hundreds of thousands of AWS customers to serve trillions of requests every month.

A broad variety of customers, from media and entertainment to financial services and other regulated industries, are taking notice of AWS Lambda. These customers are more interested in decreasing time to market, optimizing costs, and improving agility by focusing on what they do best: running their business. Lambda has become the obvious choice for mission critical applications in many industries.

The *managed runtime environment* model of AWS Lambda intentionally hides many implementation details from the user, making some of the existing best practices for cloud security irrelevant, and creating the need for new best practices. This paper presents those best practices, including information on the underpinnings of Lambda, providing a detailed view to developers, security analysts, compliance teams, and other stakeholders.

About AWS Lambda

AWS Lambda is an event-driven, serverless compute service that extends other AWS services with custom logic, or creates other backend services that operate with scale, performance, and security. Lambda can automatically run code in response to multiple events, such as HTTP requests through Amazon API Gateway, modifications to objects in Amazon S3 buckets, table updates in Amazon DynamoDB, and state transitions in AWS Step Functions. You can also run code directly from any web or mobile app. Lambda runs code on a highly available compute infrastructure, and performs all of the administration of the underlying platform, including server and operating system maintenance, capacity provisioning and automatic scaling, patching, code monitoring, and logging.

With Lambda, you can just upload your code and configure when to invoke it; Lambda takes care of everything else required to run your code with high availability. Lambda integrates with many other AWS services and enables you to create serverless applications or backend services, ranging from periodically triggered, simple automation tasks to full-fledged microservices applications.

Lambda can also be configured to access resources within your Amazon Virtual Private Cloud, and by extension, your on-premises resources.

You can easily wrap up Lambda with a strong security posture using AWS Identity and Access Management (IAM), and other techniques discussed in this whitepaper, to maintain a high level of security and auditing, and to meet your compliance needs.

Benefits of Lambda

Customers who want to unleash the creativity and speed of their development organizations, without compromising their IT team's ability to provide a scalable, cost-effective, and manageable infrastructure, find that AWS Lambda lets them trade operational complexity for agility and better pricing, without compromising on scale or reliability.

Lambda offers many benefits, including the following.

No Servers to Manage

Lambda runs your code on highly available, fault-tolerant infrastructure spread across multiple Availability Zones in a single region, seamlessly deploying code, and providing all the administration, maintenance, and patches of the infrastructure. Lambda also provides built-in logging and monitoring, including integration with Amazon CloudWatch, CloudWatch Logs, and AWS CloudTrail.

Continuous Scaling

Lambda precisely manages scaling of your functions (or application) by running event triggered code in parallel, and processing each event individually.

Subsecond Metering

With AWS Lambda, you are charged for every 100 ms your code executes and the number of times your code is triggered. You pay for consistent throughput or execution duration, instead of by server unit.

Increases Innovation

Lambda frees up your programming resources by taking over the infrastructure management, allowing them to focus more on innovation and development of business logic.

Modernize your Applications

Lambda enables you to use functions with pre-trained machine learning models to inject artificial intelligence into applications easily. A single API request can classify images, analyze videos, convert speech to text, perform natural language processing, and more.

Rich Ecosystem

Lambda provides a rich ecosystem, supporting developers through AWS Serverless Application Repository for discovering, deploying and publishing serverless applications, AWS Serverless Application Model for building serverless applications and integrations with various IDEs like AWS Cloud9, AWS Toolkit for Visual Studio, AWS Tools for Visual Studio Team Services, and several others. Along with this, Lambda is integrated with additional AWS services to provide you a rich ecosystem for building serverless applications.

Cost for Running Lambda-Based Applications

Lambda offers a granular, pay-as-you-go pricing model. With this model, you are charged based on the number of function invocations and their duration (the time it takes for the code to execute). In addition to this flexible pricing model, Lambda also offers 1 million perpetually free requests per month, allowing many customers to automate their process without any costs.

The Shared Responsibility Model

Security and Compliance is a shared responsibility between AWS and the customer. This shared responsibility model can help relieve your operational burden as AWS operates, manages, and controls the components from the host operating system and virtualization layer, down to the physical security of the facilities in which the service operates. You assume responsibility and management of the guest operating system (including updates and security patches) and other associated application software, as well as the configuration of the AWS-provided security group firewall.

For AWS Lambda, AWS manages the underlying infrastructure and foundation services, the operating system, and the application platform. You are responsible for the security of your code, the storage and accessibility of sensitive data, and identity and access management (IAM) to the Lambda service and within your function.

Figure 1 shows the shared responsibility model for AWS Lambda. AWS responsibilities appear in orange and customer responsibilities appear in blue. AWS assumes more responsibility for applications deployed to Lambda.

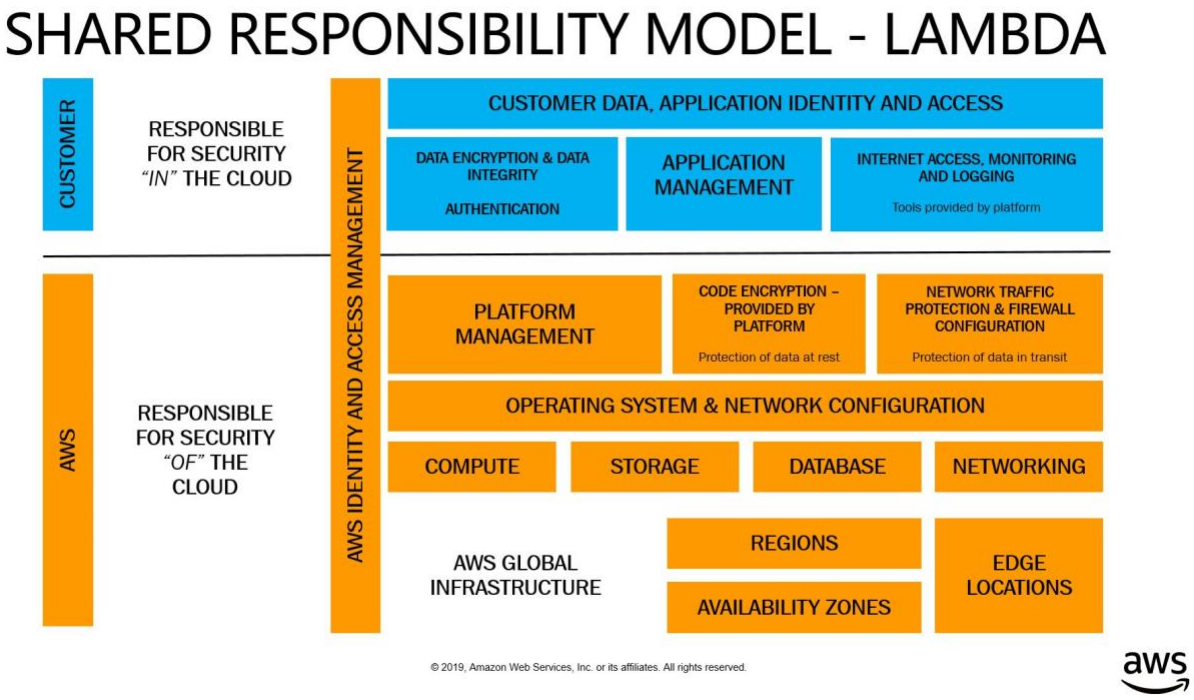


Figure 1 – Shared Responsibility Model for AWS Lambda

Lambda Runtime Environment

When Lambda executes a function on your behalf, it manages both provisioning and the resources necessary to run your code. This enables your developers to focus on business logic and writing code, not administering systems.

The Lambda service is split into the *control plane* and the *data plane*. Each plane serves a distinct purpose in the service. The control plane provides the function management APIs (*CreateFunction*, *UpdateFunctionCode*), and manages integrations with all AWS services. The data plane controls the *Invoke* API that runs Lambda functions. When a Lambda function is invoked, the data plane allocates an *execution environment* to that function, or chooses an existing execution environment that has already been set up for that function, then runs the function code in that environment.

Each function runs in one or more dedicated execution environments that are used for the lifetime of the function and then destroyed. Each execution environment hosts one concurrent invocation, but is reused in place across multiple serial invocations of the same function. Execution environments run on hardware virtualized virtual machines (microVMs). A microVM is dedicated to an AWS account, but can be reused by execution environments across functions within an account. MicroVMs are packed onto an AWS owned and managed hardware platform (Lambda Workers). Execution environments are never shared across functions, and microVMs are never shared across AWS accounts.

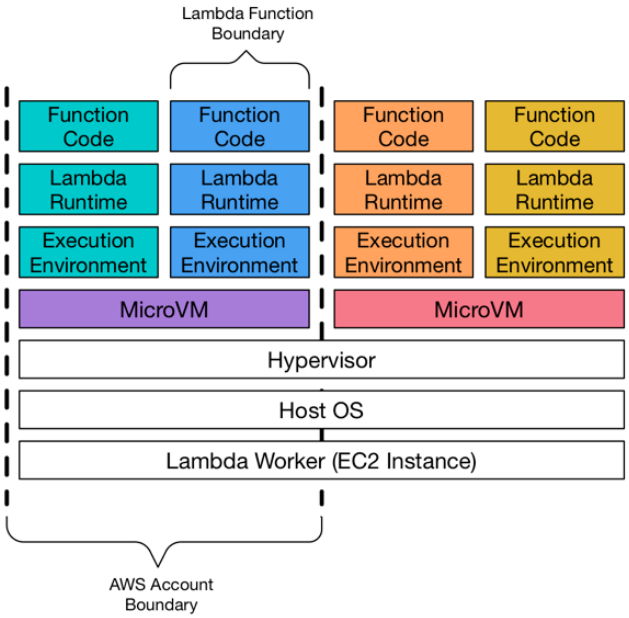


Figure 2 – Isolation model for AWS Lambda

Isolation Between Functions and Between MicroVMs

Each execution environment contains a dedicated copy of the following items:

- The function code
- Any Lambda layers selected for your function
- The function runtime, either built-in (Java 8, NodeJS 8, Python 3.7, etc.) or custom runtime
- A minimal Linux userland based on Amazon Linux

Execution environments are isolated from one another using several container technologies built in to the Linux kernel. These technologies include:

- **cgroups** – Constrain resource access to limiting CPU, memory, disk throughput, and network throughput, per execution environment.
- **namespaces** – Group process IDs, user IDs, network interfaces, and other resources managed by the Linux kernel. Each execution environment runs in a dedicated namespace.
- **seccomp-bpf** – Limit the syscalls that can be used from within the execution environment.
- **iptables and routing tables** – Isolate execution environments from each other.
- **chroot** – Provide scoped access to the underlying filesystem.

Along with AWS proprietary isolation technologies, these mechanisms provide strong isolation between execution environments. This isolation ensures that environments are not able to access or modify data that belongs to other environments.

Although multiple execution environments from a single AWS account can run on a single microVM, microVMs are never shared or reused between AWS accounts. At this time, AWS Lambda uses two different mechanisms for isolating microVMs: EC2 instances and Firecracker. EC2 instances have been used for Lambda guest isolation since 2015. Firecracker is a new open source hypervisor developed by AWS especially for serverless workloads, and was introduced in 2018. The underlying physical hardware running microVMs will be shared by workloads from multiple accounts. For more information, see *Appendix – Lambda EC2 & Firecracker Models*.

Storage and State

Though Lambda execution environments are never reused across functions, a single execution environment can be reused for invoking the same function, potentially existing for hours before it is destroyed. Functions can take advantage of this behavior to improve efficiency by keeping local caches, reusing long-lived connections between invocations, and pre-computing common results. Inside an execution environment, these multiple invocations are handled by a single process, so any process-wide state (such as a *static* state in Java) could be available for future invocations to reuse, if the invocation occurs on a reused execution environment.

Each Lambda execution environment also includes a writeable file system, available at `/tmp`. This storage is not accessible to other execution environments. As with the process state, files written to `/tmp` remain for the lifetime of the execution environment. This allows expensive transfer operations—such as downloading machine learning (ML) models—to be amortized across multiple invocations. Functions that do not want to persist data between invocations should either not write to `/tmp`, or delete their files from `/tmp` after each invocation. The `/tmp` storage is implemented with either Amazon Elastic Block Store (Amazon EBS) or local storage on the Lambda worker instance.

Also, prior to a function's first invocation, Lambda scrubs the memory before assigning it to an execution environment, which effectively guards against memory sharing between functions that belong to the same account and different customer accounts. To facilitate execution environment reuse, Lambda does not scrub memory between subsequent invocations on the same execution environment for the same function. You can implement your own memory encryption and wiping process before function termination.

Invoke Data Path

The Invoke API can be called in two modes: *event* mode and *request-response* mode. *Event* mode queues the invocation for later execution. *Request-response* mode immediately invokes the function with the provided payload, and returns the response. In both cases, the actual function execution is done in a Lambda execution environment, but the payload takes different paths. For more information, see the *Lambda Runtime Environment* section.

For *request-response* invocations, the payload passes from the API caller—such as AWS API Gateway or the AWS SDK—to a load balancer, and then to the *Lambda invoke service*. This service identifies an execution environment for the function, and passes the payload to that execution environment to complete the invocation. Traffic to the load balancer passes over the internet, and is secured with TLS. Traffic within the Lambda service (from the load balancer down) passes through a Lambda internal VPC within a single AWS region.

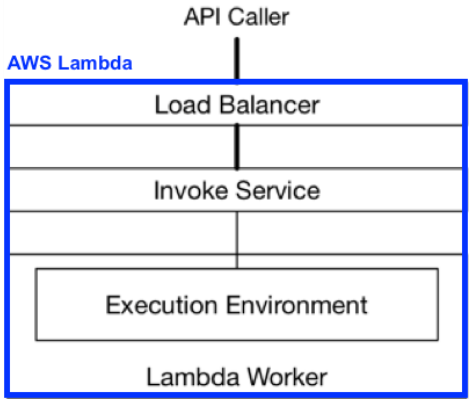


Figure 3 – Invocation model for AWS Lambda: request-response

Event invocations can be executed immediately or queued for processing. In some cases, the queue is implemented with Amazon Simple Queue Service (Amazon SQS), and passed back to the *Lambda invoke service* by an internal poller process. Traffic on this path is secured with TLS, but no additional encryption is provided for data stored in Amazon SQS. The Amazon SQS queues used by Lambda are managed by the Lambda service, and not visible to you as a customer. For event invokes, no response is returned, and any response data is discarded by the worker. Invocations from Amazon S3, Amazon SNS, CloudWatch events, and other event sources follow the *event* invoke path in the Lambda service. Invocations from Amazon Kinesis and DynamoDB streams, SQS queues, Application Load Balancer, and API Gateway follow the *request-response* path.

Runtime Maintenance in Lambda

AWS Lambda provides support for multiple programming languages through the use of runtimes, including Java 8, Python 3.7, Go, NodeJS 8, .NET core 2, and others. For a complete list, see AWS Lambda Runtimes. Lambda provides support for these runtimes, including updates, security patches, and other maintenance. Typically, no action is required to pick up the latest patches for Lambda runtimes, but sometimes action might be required to test patches before they are deployed. AWS will contact you (for example, through the Personal Health Dashboard) if action is required from you.

You can use other languages in Lambda by implementing a custom runtime. For custom runtimes, maintenance becomes your responsibility, including making sure that the runtime you provide includes the latest security patches. For more information, see Custom AWS Lambda Runtimes in the *AWS Lambda Developer Guide*.

Sometimes, Lambda deprecates runtime versions, such as when they marked as End-of-life (EOL) by their upstream maintainers. Versions that are marked as deprecated stop supporting creation of new functions and updates to existing functions that were authored in the deprecated runtime. AWS Lambda does not provide security updates, technical support, or hotfixes for deprecated runtimes, and reserves the right to disable invocations of functions configured to run on a deprecated runtime at any time. For details on when runtimes are deprecated, see the AWS Lambda Runtime Support Policy.

Monitoring and Auditing Lambda Functions

You can monitor and audit Lambda functions with many AWS methods and services, including the following services.

Amazon CloudWatch

AWS Lambda automatically monitors Lambda functions on your behalf. Through Amazon CloudWatch, it reports metrics such as the number of requests, the execution duration per request, and the number of requests resulting in an error. These metrics are exposed at the function level, which you can then leverage to set CloudWatch alarms. For a list of metrics exposed by Lambda, see AWS Lambda Metrics.

Amazon CloudTrail

Using Amazon CloudTrail, you can implement governance, compliance, operational auditing, and risk auditing of your entire AWS account, including Lambda. CloudTrail enables you to log, continuously monitor, and retain account activity related to actions across your AWS infrastructure, providing a complete event history of actions taken through the AWS Management Console, AWS SDKs, command line tools, and other AWS services. Using CloudTrail, you can optionally encrypt the log files using Amazon Key Management Service (KMS) and also leverage the CloudTrail log file integrity validation for positive assertion.

AWS X-Ray

Using AWS X-Ray, you can analyze and debug production and distributed Lambda-based applications, which enables you to understand the performance of your application and its underlying services, so you can eventually identify and troubleshoot the root cause of performance issues and errors. X-Ray's end-to-end view of requests as they travel through your application shows a map of the application's underlying components, so you can analyze applications during development and in production.

AWS Config

With AWS Config, you can track configuration changes to the Lambda functions (including deleted functions), runtime environments, tags, handler name, code size, memory allocation, timeout settings, and concurrency settings, along with Lambda IAM execution role, subnet, and security group associations. This gives you a holistic view of the Lambda function's lifecycle and enables you to surface that data for potential audit and compliance requirements.

Architecting and Operating Lambda Functions

Now that we have discussed the foundations of the Lambda service, we move on to architecture and operations. For information about standard best practices for serverless applications, see the Serverless Application Lens whitepaper, which defines and explores the pillars of the AWS Well Architected Framework in a Serverless context.

- **Operational Excellence Pillar** – The ability to run and monitor systems to deliver business value and to continually improve supporting processes and procedures.
- **Security Pillar** – The ability to protect information, systems, and assets while delivering business value through risk assessment and mitigation strategies.
- **Reliability Pillar** – The ability of a system to recover from infrastructure or service disruptions, dynamically acquire computing resources to meet demand, and mitigate disruptions such as misconfigurations or transient network issues.
- **Performance Efficiency Pillar** – The efficient use of computing resources to meet requirements and the maintenance of that efficiency as demand changes and technologies evolve.

The Serverless Application Lens whitepaper includes topics such as logging metrics and alarming, throttling and limits, assigning permissions to Lambda functions, and making sensitive data available to Lambda functions.

Lambda and Compliance

As mentioned in *The Shared Responsibility Model* section, you are responsible for determining which compliance regime applies to your data. After you have determined your compliance regime needs, you can use the various Lambda features to match those controls. You can contact AWS experts (such as Solution Architects, Domain experts, Technical Account Managers and other human resources) for assistance. However, AWS cannot advise customers on whether (or which) compliance regimes are applicable to a particular use case.

As of March 2019, Lambda is compliant with SOC 1, SOC 2, SOC 3, PCI DSS, U.S. Health Insurance Portability and Accountability Act (HIPAA), etc. For a list of compliance information, see the AWS Services in Scope by Compliance Program page.

Because of the sensitive nature of some compliance reports, they cannot be shared publicly. For access to these reports, you can sign in to your AWS console and use AWS Artifact—a no cost, self-service portal—for on-demand access to AWS compliance reports.

Conclusion

AWS Lambda offers a powerful toolkit for building secure and scalable applications. Many of the best practices for security and compliance in AWS Lambda are the same as in all AWS services, but some are particular to Lambda. This paper describes the benefits of Lambda, its suitability for applications, and the Lambda managed runtime environment. It also includes information about monitoring and auditing, and security and compliance best practices. As you think about your next implementation, consider what you learned about AWS Lambda and how it might improve your next workload solution.

Contributors

Contributors to this document include:

- Mayank Thakkar, Global Life Sciences Solutions Architect
- Brian McNamara, Specialist Technical Account Manager (Serverless)
- Marc Brooker, Senior Principal Engineer (Serverless)
- Osman Surkatty, Senior Security Engineer (Serverless)

Further Reading

For additional information, see:

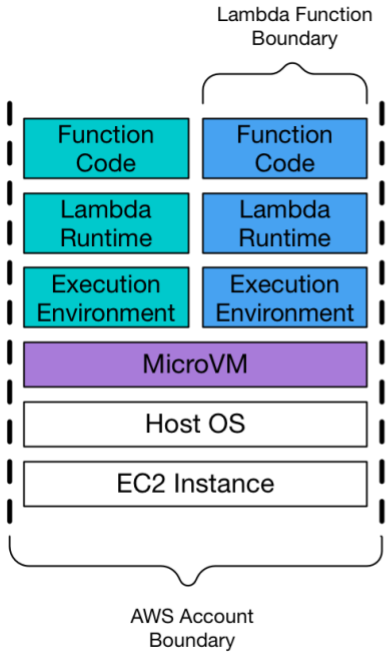
- AWS Security Best Practices covers best practices for AWS security in general, and many AWS services. It also introduces the Shared Responsibility model, and best practices for managing IAM roles and accounts.
- Serverless Application Lens covers the AWS well-architected framework identifies key elements to ensure your workloads are architected according to best practices
- Introduction to AWS Security provides a broad introduction to thinking about security in AWS.
- AWS Risk and Compliance provides an overview of compliance in AWS.

Document Revisions

Date	Description
April 2019	Replaced Lambda EC2 & Firecracker Models diagrams.
March 2019	First publication

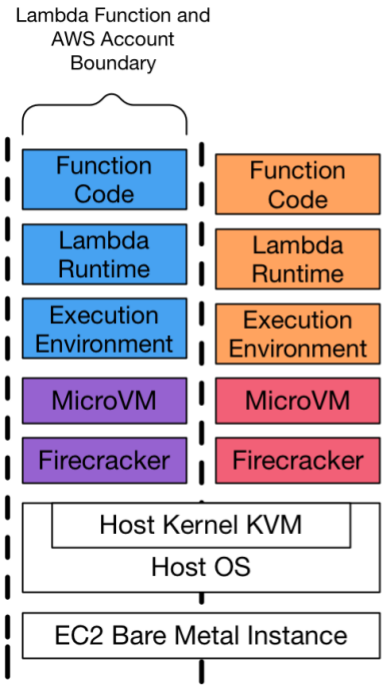
Appendix – Lambda EC2 & Firecracker Models

The following is a comparison of EC2 and Firecracker models for AWS Lambda.



Lambda's EC2 Model

Note 1:1 mapping between MicroVM and EC2 Instance.



Lambda's Firecracker Model

Note 1:1 mapping between execution environment and MicroVM.