AWS 백서

AWS Lambda 보안 개요

AWS Lambda 보안 개요: AWS 백서

Table of Contents

요약 i
요약 1
소개 2
AWS Lambda 정보 3
Lambda의 이점 3
관리할 서버가 없음 3
지속적인 크기 조정 4
밀리초 측정 4
혁신 증대 4
애플리케이션 현대화 4
풍부한 에코시스템 4
Lambda 기반 애플리케이션의 실행 비용 4
공동 책임 모델 5
Lambda 함수 6
Lambda 호출 모드 7
Lambda 실행 9
Lambda 실행 환경 9
실행 역할 10
Lambda MicroVM 및 작업자 11
Lambda 격리 기술 12
스토리지 및 상태 12
Lambda의 런타임 유지 관리 14
Lambda 함수 모니터링 및 감사 15
Amazon CloudWatch 15
Amazon CloudTrail 15
AWS X-Ray 15
AWS Config 15
Lambda 함수 설계 및 운영 16
Lambda와 규정 준수 17
Lambda 이벤트 소스 18
결론 19
기여자 20
참고 문헌 21
문서 개정 22

고지 사항 23

AWS Lambda 보안 개요

게시 날짜: 2021년 2월 12일(문서 개정)

요약

이 백서에서는 보안 측면에서 AWS Lambda 서비스를 심층 분석합니다. 서비스를 새로 사용하는 사용자를 위한 유용한 서비스의 종합 설명과 기존 사용자를 위한 심도 있는 Lambda 관련 정보를 제공합니다.

이 백서는 CISO(최고 정보 보안 책임자), 정보 보안 엔지니어, 엔트프라이즈 아키텍트, 규정 준수 팀 및 기타 AWS Lambda의 기초를 이해하는 데 관심이 있는 사용자를 대상으로 작성되었습니다.

소개

최근에는 기본 인프라를 관리하지 않으면서 확장성, 성능 및 비용 효율성을 실현하기 위해 AWS Lambda를 이용하는 워크로드가 늘어나고 있습니다. 이러한 워크로드는 초당 수천 개의 동시 요청으로 확장됩니다. Lambda는 현재 AWS에서 제공하는 여러 중요한 서비스 중 하나입니다. 수십만 개의 Amazon Web Services(AWS) 고객사에서 Lambda를 사용하여 매달 수조 건의 요청을 처리하고 있습니다.

Lambda는 많은 산업 분야의 중요 업무용 애플리케이션에 적합합니다. 미디어 및 엔터테인먼트부터 금융 서비스 및 기타 규제를 받는 산업 분야에 이르기까지, 다양한 분야의 고객들이 Lambda에 주목하고 있습니다. 이 고객들은 자신이 가장 잘하는 것, 즉 비즈니스 운영에 집중함으로써 시장 출시 시간을 단축하고 비용을 최적화하며 민첩성을 향상합니다.

관리형런타임 환경 모델을 통해 Lambda는 서버리스 워크로드 실행에 대한 구현 세부 정보의 대부분을 관리할 수 있습니다. 이 모델은 클라우드 보안을 단순화하면서 공격 표면을 더욱 줄입니다. 이 백서는 개발자, 보안 분석가, 보안 및 규정 준수 팀 및 기타 이해 관계자에게 모범 사례와 함께 해당 모델의 토대를 제시합니다.

AWS Lambda 정보

AWS Lambda는 이벤트 중심의 서버리스 컴퓨팅 서비스로, 사용자 지정 로직으로 다른 AWS 서비스를 확장하거나 확장성, 성능 및 보안을 보장하면서 운영되는 다른 백엔드 서비스를 생성합니다. Lambda는 Amazon API Gateway를 통한 HTTP 요청, Amazon S3 버킷의 객체 수정, Amazon DynamoDB의 테이블 업데이트, AWS Step Functions의 상태 전환과 같은 여러 이벤트에 응답으로 코드를 자동으로 실행할 수 있습니다. 또한 웹 또는 모바일 앱에서 코드를 직접 실행할 수도 있습니다. Lambda는 고가용성 컴퓨팅 인프라에서 코드를 실행하고 서버 및 운영 체제 유지 보수, 용량 프로비저닝 및 자동 크기 조정, 패치 적용, 코드 모니터링 및 로깅을 비롯하여 기본 플랫폼의 모든 관리 작업을 수행합니다.

Lambda를 사용하면 코드를 업로드하고 호출 시점을 구성할 수 있습니다. Lambda는 고가용성을 유지하며 코드를 실행하는 데 필요한 다른 모든 작업을 처리합니다. Lambda는 다른 많은 AWS 서비스와 통합되며, 주기적으로 작동되는 단순한 자동화 작업부터 완전한 마이크로서비스 애플리케이션에 이르기까지 다양한 서버리스 애플리케이션 또는 백엔드 서비스를 만들도록 지원합니다.

또한 Amazon Virtual Private Cloud 내의 리소스와 더 나아가 온프레미스 리소스에 액세스하도록 Lambda를 구성할 수도 있습니다.

이 백서에서 설명하는 AWS Identity and Access Management(IAM) 및 기타 기법을 사용하여 강력한 보안 태세를 Lambda에 손쉽게 적용함으로써 높은 수준의 보안과 감사를 유지하고 규정 준수 요구 사항을 충족할 수 있습니다.

주제

- Lambda의 이점
- Lambda 기반 애플리케이션의 실행 비용

Lambda의 이점

확장 가능하고 비용 효율적이며 관리가 용이한 인프라를 제공하는 IT 팀의 능력을 저해하지 않으면서 개발 조직의 창의성과 속도를 증진하려는 고객은 AWS Lambda를 활용하여 규모 또는 안정성을 저해하지 않으면서 운영 복잡성을 없애고 대응력을 확보하고 더 유리한 요금을 적용받을 수 있습니다.

Lambda는 다음을 비롯하여 다양한 이점을 제공합니다.

관리할 서버가 없음

Lambda는 단일 리전의 여러 가용 영역(AZ)에 걸쳐 분산되어 있는 고가용성과 내결함성을 갖춘 인프라에서 코드를 실행하며, 원활하게 코드를 배포하고 인프라에 대한 모든 관리, 유지 관리 및 패치 기능을

제공합니다. 또한 Lambda는 Amazon CloudWatch, CloudWatch Logs 및 AWS CloudTrail과의 통합을 비롯하여 로깅 및 모니터링 기능을 기본 제공합니다.

지속적인 크기 조정

Lambda는 트리거된 이벤트 코드를 병렬로 실행하고 각 이벤트를 개별적으로 처리하여 함수(또는 애플리케이션)의 크기 조정을 정밀하게 관리합니다.

밀리초 측정

AWS Lambda를 사용하면 코드가 실행되는 1밀리초(ms)마다 요금이 부과되며 코드가 트리거된 횟수에 대한 요금이 부과됩니다. 서버 단위가 아니라 일정한 처리량 또는 실행 시간에 대해 요금을 지불합니다.

혁신 증대

Lambda는 인프라 관리 작업을 자동으로 수행함으로써 프로그래밍 리소스의 여유 공간을 늘려 비즈니스 로직의 혁신과 개발에 좀 더 집중할 수 있습니다.

애플리케이션 현대화

Lambda는 사전 학습된 기계 학습 모델이 적용된 함수를 사용하여 애플리케이션에 손쉽게 인공 지능을 추가합니다. 단일 애플리케이션 프로그래밍 인터페이스(API) 요청을 통해 이미지를 분류하고, 동영상을 분석하고, 음성을 텍스트로 변환하고, 자연어 처리를 수행하는 등 다양한 작업을 수행할 수 있습니다.

풍부한 에코시스템

Lambda는 서버리스 애플리케이션의 검색, 배포 및 게시에 사용되는 AWS Serverless Application Repository을 통해, 또한 서버리스 애플리케이션의 구축과 AWS Cloud9, AWS Toolkit for Visual Studio, AWS Tools for Visual Studio Team Services 및 몇 가지 기타 서비스를 비롯하여 다양한 통합 개발 환경(IDE)과의 통합을 위한 AWS Serverless Application Model을 통해 개발자를 지원합니다. Lambda는 추가 AWS 서비스와 통합되어 서버리스 애플리케이션 구축을 지원하는 풍부한 에코시스템을 제공합니다.

Lambda 기반 애플리케이션의 실행 비용

Lambda는 세분화된 종량 요금제 모델을 제공합니다. 이 모델에서는 함수 호출 횟수와 함수 실행 시간(코드 실행에 걸리는 시간)을 기준으로 요금이 부과됩니다. 이 유연한 요금 모델 외에도 Lambda는 많은 고객이 추가 비용 없이 프로세스를 자동화하도록 매달 1백만 건의 영구 무료 요청을 제공합니다.

공동 책임 모델

보안 및 규정 준수는 AWS와 고객의 공동 책임입니다. 이 공동 책임 모델에서는 AWS가 호스트 운영 체제 및 가상화 계층부터 서비스가 운영되는 시설의 물리적 보안에 이르기까지 모든 구성 요소를 운영, 관리 및 제어하므로 고객의 운영 부담을 더는 데 도움이 됩니다.

AWS Lambda의 경우, AWS는 기본 인프라 및 서비스, 운영 체제 및 애플리케이션 플랫폼을 관리합니다. 사용자는 함수 내의 코드 보안과 Lambda 서비스의 자격 증명 및 액세스 관리(IAM)에 대한 책임이 있습니다.

그림 1은 AWS Lambda의 공통된 고유 구성 요소에 적용되는 공동 책임 모델을 보여줍니다. AWS 책임은 주황색으로 점선 아래에, 고객 책임은 파란색으로 점선 위에 표시됩니다.

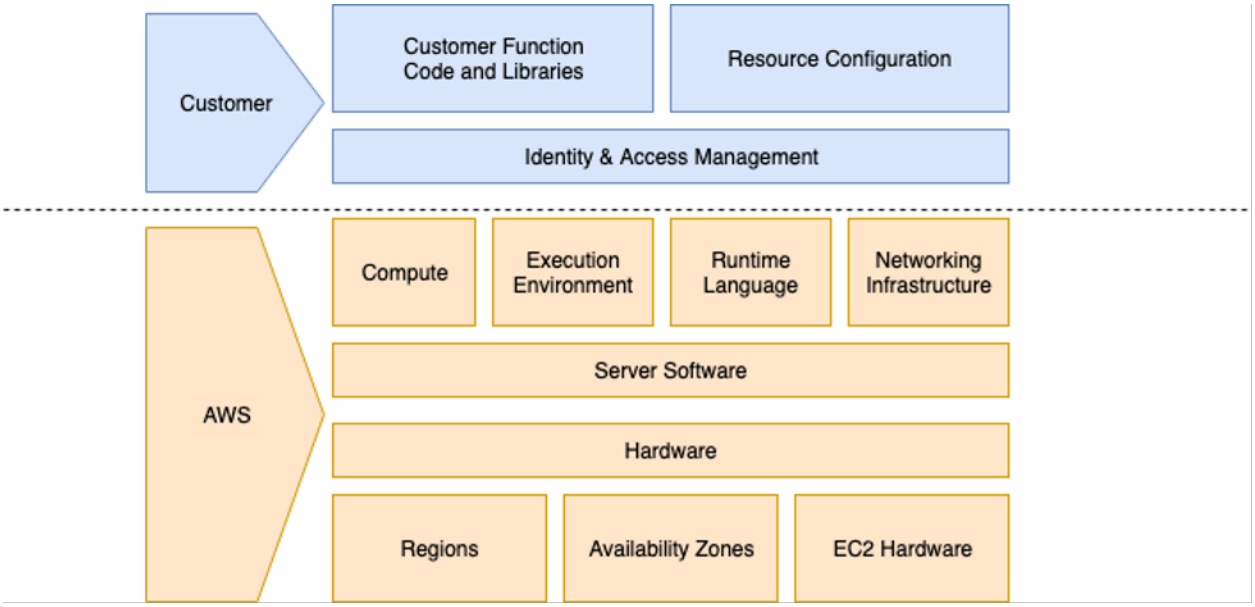

그림 1 - AWS Lambda의 공동 책임 모델

Lambda 함수 및 계층

Lambda를 사용하면 기본 인프라를 관리할 필요 없이 가상으로 코드를 실행할 수 있습니다. Lambda를 제공하는 코드와 Lambda가 사용자를 대신하여 해당 코드를 실행하는 방법에 대한 구성에만 책임이 있습니다. 현재 Lambda는 함수와 계층이라는 두 가지 유형의 코드 리소스를 지원합니다.

함수는 Lambda에서 코드를 실행하기 위해 호출할 수 있는 리소스이며, 함수에는 계층이라는 공통 또는 공유 리소스가 포함될 수 있습니다. 계층을 사용하여 여러 함수 또는 AWS 계정 간에 공통 코드 또는 데이터를 공유할 수 있습니다. 함수 또는 계층 내에 포함된 모든 코드를 관리할 책임은 사용자에게 있습니다. Lambda가 고객으로부터 함수 또는 계층 코드를 받으면 Lambda는 AWS Key Management Service(AWS KMS)를 사용하여 저장된 상태로 암호화하고 TLS 1.2+로 전송 중에 이를 암호화하여 액세스를 보호합니다.

AWS Lambda 정책이나 리소스 기반 권한을 통해 함수와 계층에 대한 액세스를 관리할 수 있습니다. IAM에서 지원되는 IAM 기능의 전체 목록은 IAM과 함께 작동하는 AWS 서비스를 참조하세요.

또한 Lambda의 제어 영역 API를 통해 함수와 계층의 전체 수명 주기를 제어할 수 있습니다. 예를 들어 `DeleteFunction`을 호출하여 함수를 삭제하거나 `RemovePermission`을 호출하여 다른 계정의 권한을 취소하도록 선택할 수 있습니다.

Lambda 호출 모드

호출 API는 이벤트 모드와 요청-응답 모드의 두 가지 모드로 호출할 수 있습니다.

- 이벤트 모드는 비동기식 호출에 대한 페이로드를 대기열에 넣습니다.
- 요청-응답 모드에서는 제공된 페이로드와 동시에 함수를 호출하고 즉시 응답을 반환합니다.

두 경우 모두 함수 실행은 항상 Lambda 실행 환경에서 이루어지지만 페이로드의 경로는 다릅니다. 자세한 내용은 이 문서의 “Lambda 실행 환경”을 참조하세요.

사용자를 대신하여 호출을 수행하는 다른 AWS 서비스를 사용할 수도 있습니다. 사용되는 호출 모드는 사용 중인 AWS 서비스 및 해당 서비스의 구성 방법에 따라 다릅니다. 다른 AWS 서비스가 Lambda와 통합되는 방법에 대한 자세한 내용은 다른 서비스와 함께 AWS Lambda 사용 단원을 참조하세요.

Lambda가 요청-응답 호출을 수신하면 호출 서비스로 직접 전달됩니다. 호출 서비스를 사용할 수 없는 경우, 호출자는 페이로드 클라이언트 측을 일시적으로 대기시켜 지정된 횟수만큼 호출을 재시도할 수 있습니다. 호출 서비스가 페이로드를 수신하면 서비스는 요청에 대해 사용 가능한 실행 환경을 식별하려고 시도하고 페이로드를 해당 실행 환경에 전달하여 호출을 완료합니다. 기존 또는 적절한 실행 환경이 없는 경우, 요청에 따라 실행 환경이 동적으로 생성됩니다. 전송 중에 호출 서비스로 전송된 호출 페이로드는 TLS 1.2+로 보호됩니다. Lambda 서비스 내 트래픽(로드 밸런서에서 하향)은 요청이 전송된 AWS 리전 내에서 Lambda 서비스 소유의 격리된 내부 Virtual Private Cloud(VPC)를 통과합니다.

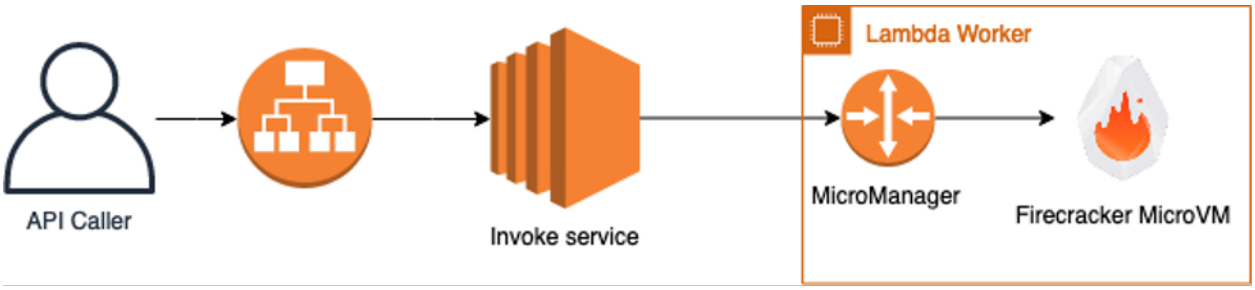

그림 2 - AWS Lambda 요청-응답의 호출 모델

이벤트 호출 모드 페이로드는 호출 전에 처리를 위해 항상 대기열에 있습니다. 모든 페이로드는 Amazon Simple Queue Service(Amazon SQS) 대기열에서 처리하도록 배치됩니다. 대기 중인 이벤트는 항상 TLS 1.2+를 통해 전송 중에 보호되지만, 현재 저장된 상태에서는 암호화되지 않습니다. Lambda에 사용되는 Amazon SQS 대기열은 Lambda 서비스에 의해 관리되며 고객에게 표시되지 않습니다. 대기 중인 이벤트는 공유 대기열에 저장할 수 있지만 고객이 직접 제어할 수 없는 여러 요소(예: 호출 속도, 이벤트 크기 등)에 따라 전용 대기열에 마이그레이션되거나 할당될 수 있습니다.

대기 중인 이벤트는 Lambda의 폴러 플릿에 의해 일괄적으로 검색됩니다. 폴러 플릿은 아직 처리되지 않은 대기 중인 이벤트 호출을 처리하는 것을 목적으로 하는 EC2 인스턴스 그룹입니다. 폴러 플릿에서 처리해야 하는 대기 중인 이벤트를 검색할 때 고객이 요청-응답 모드에서 호출하는 것처럼 호출 서비스에 전달하여 처리합니다.

호출을 수행할 수 없는 경우, 폴러 플릿은 실행을 완료할 수 있거나 실행 재시도 횟수가 초과될 때까지 이벤트를 일시적으로 메모리 내 호스트에 저장합니다. 폴러 플릿 자체의 디스크에 페이로드 데이터가 기록되지 않습니다. AWS 고객 전체에서 폴링 플릿을 처리할 수 있으므로 호출 시간이 가장 짧습니다. 이벤트 호출 모드를 사용할 수 있는 서비스에 대한 자세한 내용은 다른 서비스 함께 AWS Lambda 사용을 참조하세요.

Lambda 실행

Lambda가 사용자를 대신하여 함수를 실행할 때 코드 실행에 필요한 기본 시스템의 프로비저닝과 구성 모두를 관리합니다. 이를 통해 개발자는 기본 시스템의 관리 작업이 아니라 비즈니스 로직과 코드 작성에 집중할 수 있습니다.

Lambda 서비스는 제어 영역 및 데이터 영역으로 나뉩니다. 각 영역의 용도는 서비스마다 다릅니다. 제어 영역은 관리 API(예:`CreateFunction`, `UpdateFunctionCode`, `PublishLayerVersion` 등)를 제공하고 모든 AWS 서비스와의 통합을 관리합니다. Lambda의 제어 영역에 대한 통신은 TLS에 의해 전송 중에 보호됩니다. Lambda의 제어 영역 내에 저장된 모든 고객 데이터는 AWS KMS를 사용하여 미사용 시 암호화되며, 이는 무단 공개 또는 변조로부터 데이터를 보호하도록 설계되었습니다.

데이터 영역은 Lambda 함수의 호출을 트리거하는 Lambda의 호출 API입니다. Lambda 함수가 호출되면 데이터 영역은 AWS Lambda 작업자(또는 단순히 Amazon EC2 인스턴스의 한 유형인 작업자)의 실행 환경을 해당 함수 버전에 할당하거나 해당 함수 버전에 대해 이미 설정된 기존 실행 환경을 선택한 다음 호출을 완료하는 데 사용합니다. 자세한 내용은 이 문서의 "AWS Lambda MicroVM 및 작업자” 단원을 참조하세요.

Lambda 실행 환경

각 호출은 Lambda의 호출 서비스에 의해 요청을 서비스할 수 있는 작업자의 실행 환경으로 라우팅됩니다. 고객과 다른 사용자는 데이터 영역을 통해서만 실행 환경과의 인바운드/수신 네트워크 통신을 직접 시작할 수 있습니다. 이렇게 하면 실행 환경과의 통신을 인증하고 승인할 수 있습니다.

실행 환경은 특정 함수 버전용으로 예약되어 있으며 함수 버전, 함수 또는 AWS 계정에서 재사용할 수 없습니다. 즉, 두 가지 버전이 있을 수 있는 단일 함수로 최소 두 개의 고유한 실행 환경을 만들 수 있습니다.

각 실행 환경은 한 번에 하나의 동시 호출에만 사용할 수 있으며, 성능상의 이유로 동일한 함수 버전의 여러 호출에서 재사용할 수 있습니다. 여러 요소(예: 호출 속도, 함수 구성 등)에 따라 지정된 함수 버전에 대해 하나 이상의 실행 환경이 존재할 수 있습니다. 이러한 접근 방식을 통해 Lambda는 고객에게 함수 버전 수준의 격리를 제공할 수 있습니다.

Lambda는 현재 함수 버전의 실행 환경 내에서 호출을 격리하지 않습니다. 즉, 한 번의 호출이 다음 호출에 영향을 줄 수 있는 상태(예: /tmp에 기록된 파일 또는 메모리 내 데이터)를 남길 수 있습니다. Lambda는 한 호출이 다른 호출에 영향을 주지 않도록 별도의 함수를 추가로 생성하는 것을 권장합니다. 예를 들어 오류가 발생하기 쉬운 복잡한 구문 분석 작업에 대해 고유한 함수를 만들고 보안에 중요

한 작업을 수행하지 않는 함수를 재사용할 수 있습니다. Lambda는 현재 고객이 생성할 수 있는 함수의 수를 제한하지 않습니다. 한도에 대한 자세한 내용은 Lambda 할당량 페이지를 참조하십시오.

실행 환경은 Lambda에 의해 지속적으로 모니터링 및 관리되며, 다음을 포함하되 이에 국한되지 않는 여러 이유로 생성 또는 제거될 수 있습니다.

- 새 호출이 도착했으며 적절한 실행 환경이 없음
- 내부 런타임 또는 작업자 소프트웨어 배포가 발생함
- 새 프로비저닝된 동시성 구성이 게시됨
- 실행 환경 또는 작업자의 임대 시간이 최대 수명에 가까워지거나 초과되었습니다.
- 기타 내부 워크로드 재분배 프로세스

고객은 함수 구성에 프로비저닝된 동시성을 구성하여 함수 버전에 대해 존재하는 사전 프로비저닝된 실행 환경의 수를 관리할 수 있습니다. 이렇게 구성되면 Lambda는 구성된 실행 환경 수가 항상 존재하도록 생성, 관리 및 보장합니다. 이를 통해 고객은 규모에 관계없이 서버리스 애플리케이션의 시작 성능을 보다 효과적으로 제어할 수 있습니다.

프로비저닝된 동시성 구성을 통해서만 고객은 호출에 응답하여 Lambda에서 생성하거나 관리하는 실행 환경의 수를 결정적으로 제어할 수 있습니다.

실행 역할

또한 각 Lambda 함수는 함수와 관련된 제어 영역 및 데이터 영역 작업을 수행할 때 Lambda 서비스에서 위임하는 IAM 역할인 실행 역할로 구성해야 합니다. Lambda 서비스는 이 역할을 위임하여 함수를 호출하는 동안 환경 변수로 사용할 수 있는 임시 보안 자격 증명을 가져옵니다. 성능상의 이유로 Lambda 서비스는 이러한 자격 증명을 캐시하여 동일한 실행 역할을 사용하는 여러 실행 환경에서 다시 사용할 수 있습니다.

최소 권한 원칙을 준수하기 위해 Lambda는 각 함수에 고유한 역할이 있으며 필요한 최소 권한 집합으로 구성할 것을 권장합니다.

또한 Lambda 서비스는 VPC 기능에 대한 탄력적 네트워크 인터페이스(ENI) 생성 및 구성, Amazon CloudWatch Application Insights에 로그 보내기, AWS X-Ray에 추적 보내기 또는 기타 호출되지 않은 관련 작업과 같은 특정 제어 영역 작업을 수행하는 실행 역할을 위임할 수 있습니다. 고객은 AWS CloudTrail의 감사 로그를 검토하여 이러한 사용 사례를 언제든지 검토하고 감사할 수 있습니다.

이 주제에 대한 자세한 내용은 AWS Lambda 실행 역할 설명서 페이지를 참조하세요.

Lambda MicroVM 및 작업자

Lambda는 AWS Lambda 작업자라는 Amazon EC2 인스턴스 플릿에서 실행 환경을 생성합니다. 작업자는 고객에게 표시되지 않는 별도의 격리된 AWS 계정에서 Lambda에 의해 시작되고 관리되는 베어 메탈 EC2 Nitro 인스턴스입니다. 작업자는 Firecracker에서 만든 하나 이상의 하드웨어 가상화 마이크로 가상 머신(MVM)을 보유하고 있습니다. Firecracker는 Linux의 KVM(커널 기반 가상 머신)을 사용하여 MVM을 만들고 관리하는 오픈 소스 VMM(가상 머신 모니터)입니다. 서버리스 운영 모델을 제공하는 안전한 다중 테넌트 컨테이너 및 함수 기반 서비스를 만들고 관리하기 위해 특별히 구축되었습니다. Firecracker의 보안 모델에 대한 자세한 내용은 Firecracker 프로젝트 웹 사이트를 참조하십시오.

공동 책임 모델의 일부인 Lambda는 작업자의 보안 구성, 제어 및 패치 수준을 유지 관리할 책임이 있습니다. Lambda 팀은 Amazon Inspector를 사용하여 알려진 잠재적 보안 문제는 물론 기타 사용자 지정 보안 문제 알림 메커니즘 및 사전 공개 목록을 검색하므로 고객은 실행 환경의 기본 보안 태세를 관리할 필요가 없습니다.

그림 3 – AWS Lambda 작업자의 격리 모델

작업자의 임대 수명은 최대 14시간입니다. 작업자가 최대 임대 시간에 가까워지면 더 이상 호출이 라우팅되지 않고, MVM이 정상적으로 종료되며, 기본 Worker 인스턴스가 종료됩니다. Lambda는 플릿 수명의 수명 주기 활동을 지속적으로 모니터링하고 경보를 보냅니다.

작업자와의 모든 데이터 영역 통신은 Galois/Counter Mode(AES-GCM)의 고급 암호화 표준을 사용하여 암호화됩니다. Lambda의 서비스 계정에서 Lambda가 관리하는 네트워크 격리 Amazon VPC에 호스팅되므로 고객은 데이터 영역 작업을 통해서만 작업자와 직접 상호 작용할 수 있습니다.

작업자가 새 실행 환경을 만들어야 하는 경우, 고객 함수 아티팩트에 액세스할 수 있는 시간 제한 권한이 부여됩니다. 이 아티팩트는 Lambda의 실행 환경과 작업자를 위해 특별히 최적화되었습니다. ZIP 형식을 사용하여 업로드한 함수 코드는 한 번 최적화된 후 AWS 관리형 키와 AES-GCM을 사용하여 암호화된 형식으로 저장됩니다.

컨테이너 이미지 형식을 사용하여 Lambda에 업로드된 함수도 최적화됩니다. 컨테이너 이미지는 먼저 원본 소스에서 다운로드되어 고유한 청크로 최적화된 다음, AES-CTR, AES-GCM 및SHA-256 MAC의 조합을 사용하는 인증된 수렴 암호화 방법을 사용하여 암호화된 청크로 저장됩니다. 수렴 암호화 방법을 사용하면 Lambda로 암호화된 청크의 중복을 안전하게 제거할 수 있습니다. 고객 데이터의 암호화를 해제하는 데 필요한 모든 키는 고객 관리형 AWS KMS 고객 마스터 키(CMK)를 사용하여 보호됩니다. 고객은 Lambda 서비스의 CMK 사용량을 통해 AWS CloudTrail 로그에서 추적 및 감사를 수행할 수 있습니다.

Lambda 격리 기술

Lambda는 다양한 오픈 소스 및 독점 격리 기술을 사용하여 작업자와 실행 환경을 보호합니다. 각 실행 환경에는 다음 항목의 전용 복사본이 포함되어 있습니다.

- 특정 함수 버전의 코드
- 함수 버전에 대해 선택된 모든 AWS Lambda 계층
- 선택한 함수 런타임(예: Java 11, NodeJS 12, Python 3.8 등) 또는 함수의 사용자 지정 런타임
- 쓰기 가능한 /tmp 디렉터리
- Amazon Linux 2를 기반으로 한 최소한의 Linux 사용자 공간

실행 환경은 AWS 독점 격리 기술과 함께 Linux 커널에 내장된 여러 컨테이너 유사 기술을 사용하여 서로 격리됩니다. 이러한 기술은 다음과 같습니다.

- cgroups– CPU 및 메모리에 대한 함수의 액세스를 제한하는 데 사용됩니다.
- namespaces – 각 실행 환경은 전용 네임스페이스에서 실행됩니다. 고유한 그룹 프로세스 ID, 사용자 ID, 네트워크 인터페이스 및 기타 Linux 커널에 의해 관리되는 리소스를 사용하여 이를 수행합니다.
- seccomp-bpf – 실행 환경 내에서 사용할 수 있는 시스템 호출(syscalls)을 제한합니다.
- iptables 및 라우팅 테이블 – 수신 네트워크 통신을 방지하고 MVM 간의 네트워크 연결을 분리합니다.
- chroot – 기반 파일 시스템에 대한 범위가 한정된 액세스를 제공합니다.
- Firecracker 구성 – 블록 디바이스 및 네트워크 디바이스 처리량을 제한하는 데 사용됩니다.
- Firecracker 보안 기능 – Firecracker의 현재 보안 설계에 대한 자세한 내용은 Firecracker의 최신 설계 문서를 참조하세요.

이러한 메커니즘은 AWS의 고유한 격리 기술과 함께 실행 환경 간의 강력한 격리 기능을 제공합니다.

스토리지 및 상태

실행 환경은 서로 다른 함수 버전이나 고객 간에 재사용되지 않지만 동일한 함수 버전을 호출하는 사이에 단일 환경을 재사용할 수 있습니다. 즉, 호출 사이에 데이터와 상태가 지속될 수 있습니다. 데이터 및/또는 상태는 정상적인 실행 환경 수명 주기 관리의 일부로 제거되기 전에 몇 시간 동안 계속 유지될 수 있습니다. 성능상의 이유로 함수는 호출 간에 로컬 캐시 또는 오래 지속되는 연결을 유지하고 재사

용하여 효율성을 향상할 수 있습니다. 실행 환경 내에서 이러한 여러 호출은 단일 프로세스에 의해 처리되므로, 재사용된 실행 환경에서 호출이 발생하는 경우 프로세스 차원의 상태(예: Java의 정적 상태)를 향후 재사용할 호출에 사용할 수 있습니다.

각 Lambda 실행 환경에는 `/tmp`에서 사용할 수 있는 쓰기 가능한 파일 시스템도 포함되어 있습니다. 이 스토리지는 실행 환경 간에 액세스하거나 공유할 수 없습니다. 프로세스 상태와 마찬가지로 `/tmp`에 기록된 파일은 실행 환경의 수명 기간 동안 유지됩니다. 따라서 ML(기계 학습) 모델을 다운로드하는 것과 같은 값비싼 전송 작업을 여러 호출에 걸쳐 분할할 수 있습니다. 호출 간에 데이터를 유지하지 않을 함수의 경우, 호출 사이에 /tmp에 쓰거나 /tmp에서 파일을 삭제하지 않아야 합니다. `/tmp` 디렉터리는 Amazon EC2 인스턴스 스토어에서 지원되며 저장 시 암호화됩니다.

실행 환경 외부에서 파일 시스템에 데이터를 유지하려는 고객은 Lambda와 Amazon Elastic File System(Amazon EFS)을 통합하는 것을 고려해야 합니다. 자세한 내용은 AWS Lambda와 함께 Amazon EFS 사용을 참조하세요.

고객이 호출 간에 데이터 또는 상태를 유지하지 않으려는 경우, 실행 컨텍스트 또는 실행 환경을 사용하여 데이터나 상태를 저장하지 않는 것이 좋습니다. 고객이 호출 전반에 걸쳐 데이터 또는 상태 유출을 적극적으로 방지하려는 경우, Lambda는 각 상태에 대해 고유한 함수를 생성할 것을 권장합니다. Lambda는 보안에 중요한 상태가 호출 간에 변경될 수 있으므로 고객이 실행 환경에 사용하거나 저장하는 것을 권장하지 않습니다. 대신 각 호출 상태를 다시 계산하는 것이 좋습니다.

Lambda의 런타임 유지 관리

Lambda는 호환되는 업데이트와 보안 패치를 지속적으로 스캔 및 배포하고 다른 런타임 유지 관리 활동을 수행하여 이러한 런타임을 지원합니다. 이를 통해 고객은 함수 및 계층에 포함된 코드의 유지 관리 및 보안에만 집중할 수 있습니다. Lambda 팀은 Amazon Inspector로 알려진 보안 문제뿐만 아니라 기타 사용자 지정 보안 문제, 알림 메커니즘 및 사전 공개 목록을 검색하여 런타임 언어와 실행 환경에 계속 패치가 적용되도록 합니다. 새로운 패치 또는 업데이트가 확인되면 Lambda는 고객의 개입 없이 런타임 업데이트를 테스트하고 배포합니다. Lambda의 규정 준수 프로그램에 대한 자세한 내용은 이 문서의 “Lambda 및 규정 준수” 단원을 참조하십시오.

일반적으로 지원되는 Lambda 런타임에 대한 최신 패치를 적용하기 위해 별도의 조치를 취할 필요가 없지만, 배포 전 패치를 테스트하는 작업(예: 호환되지 않은 런타임 패치)이 필요할 수도 있습니다. 고객이 어떤 조치를 취해야 하는 경우, Lambda는 Personal Health Dashboard, AWS 계정의 이메일 또는 기타 수단을 통해 고객에게 연락하고 필요한 특정 조치를 취합니다.

고객은 사용자 지정 런타임을 구현하여 Lambda에서 다른 프로그래밍 언어를 사용할 수 있습니다. 사용자 지정 런타임의 경우, 사용자 지정 런타임에 최신 보안 패치가 포함되는지 확인하는 것을 포함하여 런타임의 유지 관리는 고객의 책임입니다. 자세한 내용은 AWS Lambda 개발자 안내서의 사용자 지정 AWS Lambda 런타임을 참조하십시오.

업스트림 런타임 언어 관리자가 해당 언어를 EOL(End of Life)로 표시하면 Lambda는 런타임 언어 버전을 더 이상 지원하지 않음으로써 이를 적용합니다. Lambda에 런타임 버전이 더 이상 사용되지 않는 것으로 표시되면 Lambda는 더 이상 사용되지 않는 런타임에 작성된 기존 함수에 대한 업데이트 및 새 함수 생성을 더 이상 지원하지 않습니다. 다가오는 런타임 사용 중단에 대해 고객에게 알리기 위해 Lambda는 다가오는 사용 중단 날짜와 예상 가능한 사항에 대한 알림을 고객에게 보냅니다. Lambda는 사용 중지된 런타임에 대해 보안 업데이트, 기술 지원 또는 핫픽스를 제공하지 않으며, 사용 중지된 런타임에서 실행되도록 구성된 함수의 호출을 언제든 비활성화할 권리가 있습니다. 고객이 더 이상 사용하지 않거나 지원되지 않는 런타임 버전을 계속 실행하려는 경우, 자체 사용자 지정 AWS Lambda 런타임을 생성할 수 있습니다. 런타임이 사용 중지되는 시점에 대한 자세한 내용은 AWS Lambda 런타임 지원 정책을 참조하십시오.

Lambda 함수 모니터링 및 감사

다음 서비스를 비롯한 여러 AWS 서비스 및 메서드를 사용하여 Lambda 함수를 모니터링하고 감사할 수 있습니다.

Amazon CloudWatch

AWS Lambda는 사용자 대신 Lambda 함수를 자동으로 모니터링합니다. Amazon CloudWatch를 통해 요청 수, 요청당 실행 시간, 오류가 발생한 요청 수 등의 지표를 보고합니다. 이 같은 지표는 함수 수준에서 노출되어, 이를 활용하여 CloudWatch 경보를 설정할 수 있습니다. Lambda에서 제공하는 지표 목록은 AWS Lambda 지표를 참조하세요.

Amazon CloudTrail

AWS CloudTrail을 사용하여 Lambda를 포함한 전체 AWS 계정에 대한 거버넌스, 규정 준수, 운영 감사 및 위험 감사 기능을 구현할 수 있습니다. CloudTrail을 사용하면 AWS 인프라 전반의 작업과 관련된 계정 활동을 로깅하고 지속적으로 모니터링하고 보존하여 AWS Management Console, AWS SDK, 명령줄 도구 및 기타 AWS 서비스를 통해 수행된 작업의 전체 이벤트 기록을 제공할 수 있습니다. CloudTrail을 사용하면 AWS KMS를 통해 선택적으로 로그 파일을 암호화할 수 있으며, 긍정적 주장에 대한 CloudTrail 로그 파일 무결성 검증 기능을 활용할 수 있습니다.

AWS X-Ray

AWS X-Ray를 사용하면 생산 및 분산 Lambda 기반 애플리케이션을 분석하고 디버깅할 수 있으며, 이를 통해 애플리케이션과 기반 서비스의 성능을 파악하여 궁극적으로 성능 문제와 오류의 근본 원인을 찾아 해결할 수 있습니다. 애플리케이션을 통해 전달되는 요청에 대한 X-Ray의 포괄적 보기는 애플리케이션 기반 구성 요소에 대한 맵을 제공하므로 개발 중에 또는 프로덕션 환경에서 애플리케이션을 분석할 수 있습니다.

AWS Config

AWS Config를 사용하면 Lambda 함수(삭제된 함수 포함), 런타임 환경, 태그, 핸들러 이름, 코드 크기, 메모리 할당, 제한 시간 설정, 동시성 설정의 구성 변경 사항을 Lambda IAM 실행 역할, 서브넷 및 보안 그룹 연결과 함께 추적할 수 있습니다. 따라서 Lambda 함수의 수명주기를 종합적으로 파악하고 잠재적인 감사 및 규정 준수 요건을 준수하기 위해 해당 데이터를 표면화할 수 있습니다.

Lambda 함수 설계 및 운영

이 단원에서는 Lambda 설계 및 운영에 대해 설명합니다. 서버리스 애플리케이션에 대한 표준 모범 사례는 서버리스 애플리케이션 렌즈 백서를 참조하세요. 이 백서에서는 서버리스 컨텍스트에서 AWS Well Architected Framework의 핵심 요소를 정의하고 살펴봅니다.

- 운영 우수성 요소 - 비즈니스 가치를 실현하고 지원 프로세스 및 절차를 지속적으로 개선하기 위해 시스템을 실행하고 모니터링하는 기능입니다.
- 보안 요소 - 정보, 시스템, 자산을 보호하는 동시에 위험 진단과 문제 해결 전략을 통해 비즈니스 가치를 실현하는 기능입니다.
- 안정성 요소 - 시스템이 인프라 또는 서비스 가동 중단으로부터 복구되고, 수요를 충족할 컴퓨팅 리소스를 동적으로 확보하며, 잘못된 구성 또는 일시적 네트워크 문제와 같은 가동 중단 문제를 해결하는 기능입니다.
- 성능 효율성 요소 - 컴퓨팅 리소스를 효율적으로 사용하여 요구 사항을 충족하고 수요 변화와 기술 발전에 따른 효율성을 유지합니다.
- 비용 최적화 요소 - 수요 변화 및 기술 발전에 따라 비용을 최소화하면서 비즈니스 성과의 달성을 보장하기 위한 지속적인 개선 프로세스입니다.

서버리스 애플리케이션 렌즈 백서에는 로깅 지표 및 경보, 제한 및 한도, Lambda 함수에 대한 권한 할당, 민감한 데이터를 Lambda 함수에 사용하는 방법 등의 주제에 대한 내용이 포함되어 있습니다.

Lambda와 규정 준수

“공동 책임 모델” 단원에서 언급했듯이 데이터에 적용할 규정 준수 체계를 결정하는 것은 고객의 몫입니다. 규정 준수 체계 요구 사항을 결정한 후에는 해당 제어 체계에 맞추어 다양한 Lambda 기능을 사용할 수 있습니다. 솔루션 아키텍트, 도메인 전문가, 기술 계정 관리자 및 기타 담당자 등의 AWS 전문가에게 문의하여 도움을 받을 수 있습니다. 단, AWS는 규정 준수 체계가 특정 사용 사례에 적용 가능한지 또는 어떤 규정 준수 체계가 적합한지에 대해 고객에게 자문을 제공할 수 없습니다.

2020년 11월부터 Lambda는 SOC 1, SOC 2 및 SOC 3 보고서의 범위에 포함되며, 이 보고서는 AWS가 주요 규정 준수 제어 항목 및 통제 대상을 어떻게 준수하고 있는지를 입증하는 독립적인 서드 파티 심사 보고서입니다. 규정 준수 정보의 최신 목록은 규정 준수 프로그램 제공 AWS 범위 내 서비스 페이지를 참조하세요.

일부 규정 준수 보고서는 기밀을 유지해야 하는 특성상 공개적으로 공유할 수 없습니다. 그러한 보고서에 액세스하려면 AWS Management Console에 로그인한 후 무료로 제공되는 셀프 서비스 포털인 AWS Artifact를 사용하여 AWS 규정 준수 보고서에 온디맨드로 접속하세요.

Lambda 이벤트 소스

Lambda는 직접 통합 및 Amazon EventBridge 이벤트 버스를 통해 140여 개의 AWS 서비스와 통합됩니다. 일반적으로 사용되는 Lambda 이벤트 소스는 다음과 같습니다.

- Amazon API Gateway
- Amazon CloudWatch Events
- Amazon CloudWatch Logs
- Amazon DynamoDB Streams
- Amazon EventBridge
- Amazon Kinesis Data Streams
- Amazon S3
- Amazon SNS
- Amazon SQS
- AWS Step Functions

이러한 이벤트 소스를 사용하여 다음을 수행할 수 있습니다.

- AWS Identity and Access Management를 사용하여 서비스 및 리소스에 대한 액세스를 안전하게 관리합니다.

- 저장된 데이터를 암호화합니다.* 모든 서비스는 전송 중인 데이터를 암호화합니다.
- VPC 엔드포인트를 사용하여 Amazon Virtual Private Cloud에서 액세스(제공: AWS PrivateLink
- Amazon CloudWatch Application Insights를 사용하여 지표를 수집, 보고 및 경보합니다.
- AWS CloudTrail을 사용하면 AWS 인프라 전반의 작업과 관련된 계정 활동을 로깅하고 지속적으로 모니터링하고 보존하여 AWS Management Console >AWS SDK, 명령줄 도구 및 기타 AWS 서비스를 통해 수행된 작업의 전체 이벤트 기록을 제공할 수 있습니다.

*게시 당시에는 Amazon EventBridge에서 저장된 데이터를 암호화할 수 없었습니다. 계속해서 서비스 홈페이지를 모니터링하여 이 기능에 대한 업데이트를 확인하세요.

결론

AWS Lambda는 보안과 확장성이 뛰어난 애플리케이션을 구축할 수 있는 강력한 도구 키트를 제공합니다. Lambda의 보안 및 규정 준수에 대한 모범 사례는 대부분 다른 모든 AWS 서비스와 동일하지만, Lambda에만 적용되는 특정한 모범 사례도 있습니다. 이 백서에서는 Lambda의 이점, 애플리케이션에 대한 Lambda 적합성, Lambda 관리형 런타임 환경에 대해 설명합니다. 또한 모니터링 및 감사에 대한 정보와 보안 및 규정 준수 모범 사례도 수록되어 있습니다. 다음 구현을 계획할 때에는 AWS Lambda에 대해 배운 내용을 적용하여 다음으로 개발할 워크로드 솔루션을 향상시키는 것을 고려해 보시기 바랍니다.

기여자

이 문서를 작성하는 데 도움을 주신 분들입니다.

- Mayank Thakkar, 글로벌 생명과학 솔루션스 아키텍트
- Marc Brooker, 선임 수석 엔지니어(서버리스)
- Osman Surkatty, 선입 보안 엔지니어(서버리스)

참고 문헌

다음에서 추가 정보를 참조하세요.

- 공동 책임 모델에서는 AWS가 보안에 대해 일반적으로 어떻게 생각하는지를 설명합니다.
- AWS 보안 모범 사례에서는 AWS Identity and Access Management(IAM) 서비스에 대한 권장 사항을 다룹니다.
- 서버리스 애플리케이션 렌즈에서는 AWS Well-Architected Framework에 대해 다루며, 모범 사례에 따라 워크로드의 설계를 보장하는 주요 요소를 살펴봅니다.
- AWS 보안 소개에서는 AWS의 보안과 관련한 고려 사항을 전반적으로 소개합니다.
- AWS 위험 및 규정 준수에서는 AWS의 규정 준수에 대한 개요를 제공합니다.

문서 개정

이 백서의 업데이트에 대한 알림을 받으려면 RSS 피드를 구독하세요.

업데이트 기록-변경	update-history-description	update-history-date
업데이트됨	중요한 업데이트	2021년 2월 15일
최초 게시	처음 게시된 백서	2019년 1월 3일

고지 사항

고객은 본 문서에 포함된 정보를 독자적으로 평가할 책임이 있습니다. 본 문서는 (a) 정보 제공만을 위한 것이며, (b) 사전 고지 없이 변경될 수 있는 현재의 AWS 제품 제공 서비스 및 사례를 보여 주며, (c) AWS 및 자회사, 공급업체 또는 라이선스 제공자로부터 어떠한 약정 또는 보증도 하지 않습니다. AWS 제품 또는 서비스는 명시적이든 묵시적이든 어떠한 종류의 보증, 진술 또는 조건 없이 '있는 그대로' 제공됩니다. 고객에 대한 AWS의 책임과 법적 책임은 AWS 계약서에 준하며 본 문서는 AWS와 고객 간의 계약에 포함되지 않고 계약을 변경하지도 않습니다.