AWS ホワイトペーパー

AWS Lambda のセキュリティの概要

AWS Lambda のセキュリティの概要: AWS ホワイトペーパー

Table of Contents

AWS Lambda のセキュリティの概要

公開日: 2021 年 2 月 12 日 (改訂履歴)

要約

このホワイトペーパーでは、セキュリティに焦点を合わせて AWS Lambda サービスについて詳しく説明します。AWS Lambda について包括的に理解できるため、このサービスを初めて導入する方に役立つ内容となっています。また、現在お使いの方がこのサービスについての理解を深めるのにも役立ちます。

このホワイトペーパーの対象読者は、CISO、情報セキュリティグループ、セキュリティアナリスト、エンタープライズアーキテクト、コンプライアンス部門、AWS Lambda の基礎を理解することに関心をお持ちの方です。

はじめに

現在、ますます多くのお客様が AWS Lambda を採用し、基盤インフラストラクチャを管理する必要なく、ワークロードのスケーラビリティ、パフォーマンス、コストの最適化を達成しています。これらのワークロードは、1 秒あたり数千の同時リクエストまでスケーリングできます。Lambda は、現在 AWS が提供している数多くの重要なサービスの 1 つです。アマゾン ウェブ サービス (AWS) をご利用になる数十万社のお客様による AWS Lambda のリクエスト処理数は、毎月数兆件にも達します。

Lambda は、多くの業界におけるミッションクリティカルなアプリケーションに適しています。メディアやエンターテインメントから、規制の厳しい金融サービスまで、多くの業界で Lambda が活用されています。Lambda を採用した企業では、事業運営に集中することができ、市場投入までの時間の短縮、コストの最適化、俊敏性の向上の実現につながっています。

マネージドランタイム環境モデルにより、Lambda では実行中のサーバーレスワークロードの運用詳細のほとんどを管理できます。このモデルを採用することにより、クラウドセキュリティがシンプルになり、攻撃対象領域が減少します。このホワイトペーパーでは、そのモデルの基礎に関する情報をベストプラクティスとともに、デベロッパー、セキュリティアナリスト、セキュリティとコンプライアンス部門などのステークホルダー向けにご提供しています。

AWS Lambda について

AWS Lambda はイベント駆動型のサーバーレスコンピューティングサービスです。ユーザーはカスタムロジックで AWS の他のサービスを拡張したり、スケール、パフォーマンス、セキュリティを備えたバックエンドサービスを構築したりできます。Lambda では、Amazon API Gateway を介した HTTP リクエスト、Amazon S3 バケットでのオブジェクトの変更、Amazon DynamoDB でのテーブルの更新、AWS Step Functions での状態遷移など、複数のイベントに応答してコードを自動的に実行できます。また、任意のウェブアプリケーションやモバイルアプリケーションから直接コードを実行できます。Lambda では、可用性の高いコンピューティングインフラストラクチャでコードを実行し、サーバーとオペレーティングシステムのメンテナンス、キャパシティーのプロビジョニング、自動スケーリング、パッチ適用、コードのモニタリング、ログ記録など、基盤となるプラットフォームのあらゆる管理タスクを実行します。

ユーザーはコードをアップロードし、コードを呼び出すタイミングを設定するだけです。あとは Lambda が高い可用性でコードを実行するために必要となるすべてのことを行います。Lambda は AWS の他の多数のサービスと統合されています。定期的にトリガーされる単純なオートメーションタスクから本格的なマイクロサービスアプリケーションまで、多様なレベルでサーバーレスアプリケーションやバックエンドサービスを構築できます。

また、Amazon Virtual Private Cloud 内のリソース、さらにオンプレミスのリソースにアクセスするよう Lambda を設定できます。

AWS Identity and Access Management (IAM) や、このホワイトペーパーで説明するその他の手法で、高レベルなセキュリティと監査体制の維持やコンプライアンス準拠に必要となる、強力なセキュリティ体制を Lambda に簡単に整えることができます。

トピック

Lambda の利点

多くの企業が、スケーラブルでコスト効率が高く管理しやすいインフラストラクチャを提供する IT チームの能力と、開発組織の創造力とスピード向上の両立を模索しています。AWS Lambda を採用することで、スケールや信頼性を損なうことなく、俊敏性と優れた料金体系で、煩雑な運用が不要になります。

Lambda には、以下のようなさまざまな利点があります。

サーバー管理が不要

Lambda では、1 つのリージョン内で複数のアベイラビリティーゾーン (AZ) に分散された、高可用性と耐障害性を備えたインフラストラクチャでコードを実行します。コードをシームレスにデプロイし、インフラストラクチャの管理、メンテナンス、パッチ適用などのあらゆるタスクを遂行します。また、Lambda には、Amazon CloudWatch、CloudWatch Logs、AWS CloudTrail との統合など、組み込みのログ記録とモニタリング機能も用意されています。

継続的スケーリング

Lambda では、イベントによってトリガーされたコードを同時に実行し、各イベントを個別に処理することで、関数 (またはアプリケーション) のスケーリングを正確に管理します。

ミリ秒単位の課金

AWS Lambda では、コードが実行されるミリ秒 (ms) ごと、およびコードがトリガーされた回数に対して課金されます。サーバー単位ではなく、安定したスループットまたは実行時間に対して料金が発生します。

イノベーションを加速

Lambda がインフラストラクチャの管理を引き継ぎ、プログラミング担当者を煩雑な管理から解放し、担当者はこれまで以上にイノベーションやビジネスロジックの開発に注力できるようになります。

アプリケーションのモダナイゼーション

Lambda では、関数と事前にトレーニングされた機械学習モデルを使用して、AI をアプリケーションに簡単に組み込むことができます。1 回のアプリケーションプログラムインターフェイス (API) リクエストで、画像の分類、動画の分析、音声からテキストへの変換、自然言語処理などを実行できます。

充実したエコシステム

Lambda では、サーバーレスアプリケーションの検索、デプロイ、公開を行うための AWS Serverless Application Repository、サーバーレスアプリケーションを構築するための AWS サーバーレスアプリケーションモデル、AWS Cloud9、AWS Toolkit for Visual Studio、AWS Tools for Visual

Studio Team Services、その他のツールキットなどのさまざまな統合開発環境 (IDE) との統合により、デベロッパーをサポートしています。Lambda には AWS のその他のサービスが統合されているため、サーバーレスアプリケーションを構築するための充実したエコシステムが利用できます。

Lambda ベースアプリケーションの運用コスト

Lambda では、きめ細かい従量制料金モデルを導入しています。このモデルでは、関数を呼び出した回数とその実行時間 (コードを実行するのにかかった時間) に基づいて課金されます。この柔軟な料金モデルに加えて、Lambda では、無期限の無料利用枠として 1 か月あたり 100 万件のリクエストを提供しています。これにより、多くのお客様がコストを発生させることなく、プロセスを自動化しています。

責任共有モデル

セキュリティとコンプライアンスは、AWS とお客様の共有責任です。この責任共有モデルでは、ホストオペレーティングシステムや仮想レイヤーから、サービスが運用されている施設の物理的なセキュリティまで、さまざまなコンポーネントを AWS が運用、管理、コントロールするため、お客様の運用上の負担が軽減します。

AWS Lambda については、AWS が基盤となるインフラストラクチャ、基盤サービス、オペレーティングシステム、アプリケーションプラットフォームを管理します。お客様は、Lambda サービスおよび関数内のコードおよび Identity and Access Management (IAM) のセキュリティに責任を負います。

図 1 は、AWS Lambda の共通および個別のコンポーネントに適用される責任共有モデルを説明しています。AWS の責任はオレンジ色の点線の下に表示され、お客様の責任は青色の点線の上に表示されています。

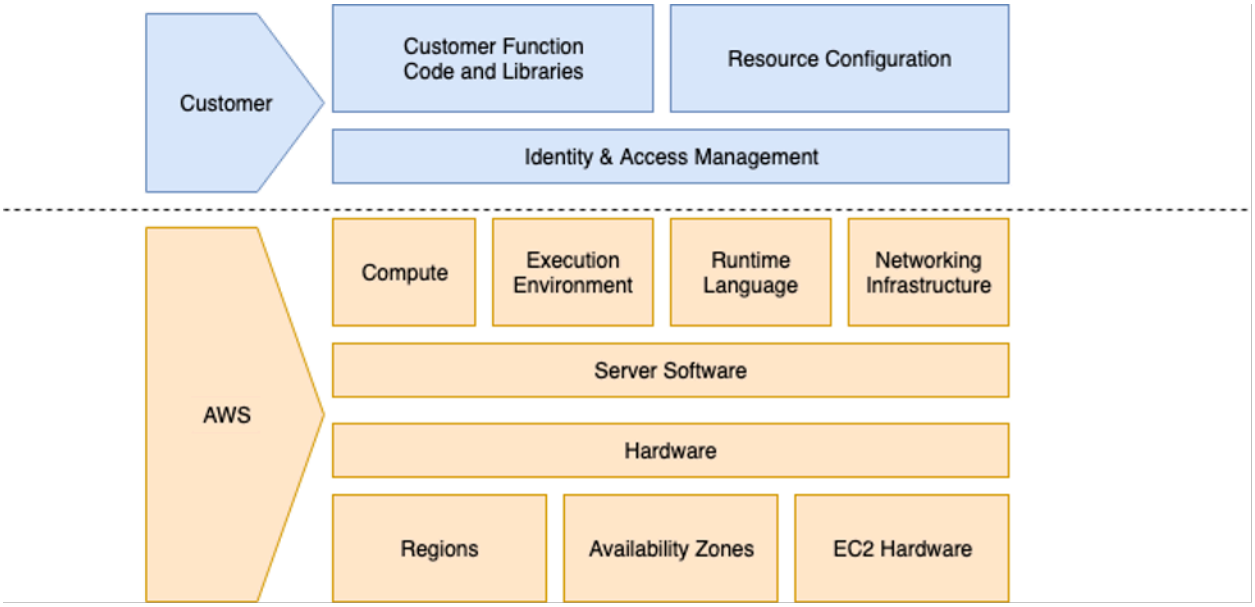


図 1 – AWS Lambda の責任共有モデル

Lambda 関数とレイヤー

Lambda を使用すると、基盤となるインフラストラクチャを管理する必要なく、バーチャルにコードを実行できます。お客様が責任を負うのは、Lambda に提供するコードと、Lambda がお客様に代わってそのコードを実行する方法の設定についてのみです。現在、Lambda は、関数とレイヤーの 2 種類のコードリソースをサポートしています。

関数は、Lambda でコードを実行するために呼び出されるリソースです。関数には、レイヤーと呼ばれる共通 (共有) リソースを含めることができます。レイヤーを使用すると、異なる関数や AWS アカウント間で共通のコードやデータを共有できます。関数またはレイヤーに含まれるすべてのコードの管理は、ユーザーの責任となります。Lambda がお客様から関数またはレイヤーのコードを受け取ると、保管時には AWS Key Management Service (AWS KMS) を、転送中には TLS 1.2 以上を使用して暗号化し、コードをアクセスから保護します。

AWS Lambda ポリシーまたはリソースベースのアクセス権限を介して、関数とレイヤーへのアクセスを管理できます。IAM でサポートされている IAM 機能の完全なリストについては、「IAM と連携する AWS のサービス」をご覧ください。

また、Lambda のコントロールプレーン API を使用して、関数とレイヤーのライフサイクル全体を制御することもできます。たとえば、`DeleteFunction` を呼び出して関数を削除したり、`RemovePermission` を呼び出して別のアカウントのアクセス許可を取り消したりすることができます。

Lambda 呼び出しモード

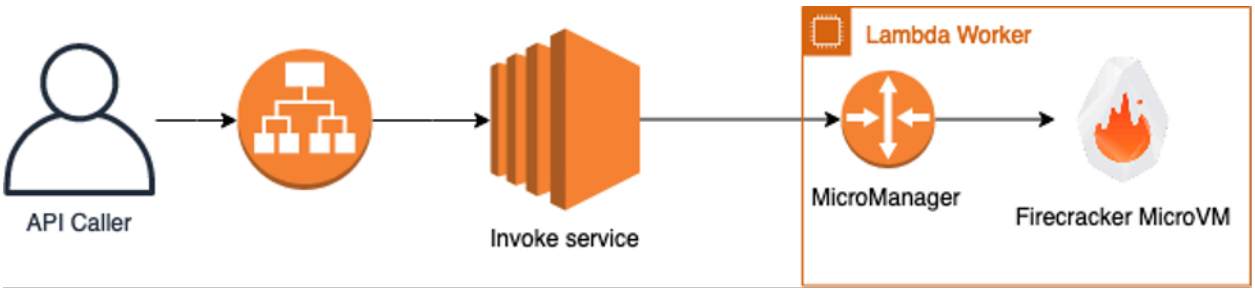
Invoke API は event モードと request-response モードの 2 つのモードで呼び出すことができます。

- event モードでは、ペイロードが非同期呼び出しのキューに入れられます。
- request-response モードでは、指定されたペイロードで関数が同期的に呼び出され、ただちに応答が返されます。

どちらの場合も、関数の実行は常に Lambda 実行環境で行われますが、ペイロードは異なるパスを通ります。詳細については、このドキュメントの「Lambda 実行環境」を参照してください。

また、ユーザーに代わって呼び出しを実行する AWS の他のサービスを使用することもできます。どの呼び出しモードが使用されるかは、利用する AWS のサービスと設定方法により異なります。その他の AWS のサービスと Lambda の統合方法の詳細については、「AWS Lambda を他のサービスで使用する」を参照してください。

Lambda がリクエスト/レスポンスの呼び出しを受け取ると、この呼出は直接呼び出しサービスに渡されます。呼び出しサービスが利用できない場合、呼び出し元はクライアント側でペイロードを一時的にキューに入れて、設定された回数だけ呼び出しを再試行できます。呼び出しサービスがペイロードを受け取ると、リクエストに対して使用可能な実行環境を特定し、ペイロードをその実行環境に渡して呼び出しを完了します。既存の実行環境または適切な実行環境が存在しない場合は、リクエストに応じて動的に作成されます。転送中、呼び出しサービスに送信される呼び出しペイロードは TLS 1.2 以上を使用して保護されます。Lambda サービス内のトラフィック (ロードバランサーから下) は、リクエストが送信された AWS リージョン内の Lambda サービスが所有する隔離された内部 Virtual Private Cloud (VPC) を通過します。



図表 2 – AWS Lambdaのリクエスト/レスポンスの呼び出しモデル

event 呼び出しモードのペイロードは、呼び出し前に常にキューに入れられて処理されます。すべてのペイロードは Amazon Simple Queue Service (Amazon SQS) キューに入れられて処理されます。キューに入れられたイベントは、転送中は常に TLS 1.2 以上で保護されます。ただし現在、保存時

には暗号化されません。Lambda によって使用される Amazon SQS キューは Lambda サービスで管理され、ユーザーには表示されません。キューに入れられたイベントは共有キューに保存できますが、ユーザーが直接制御できないさまざまな要因 (呼び出し速度、イベントのサイズなど) によっては、移行または専用キューに割り当てられる場合があります。

キューに入れられたイベントは、Lambda のポーラーフリートにより、バッチで取得されます。ポーラーフリートは、まだ処理されていないキューに入れられたイベント呼び出しを処理することを目的とした EC2 インスタンスのグループです。ポーラーフリートは、処理する必要がある、キューに入れられたイベントを取得すると、リクエスト/レスポンスモードの呼び出しと同様に、そのイベントを呼び出しサービスに渡します。

呼び出しを実行できない場合、実行を正常に完了できるか、実行の再試行回数を超えるまで、ポーラーフリートはホストのメモリ内にイベントを一時的に保存します。ポーラーフリート自体では、ペイロードデータがディスクに書き込まれることは決してありません。ポーラーフリートは AWS のユーザー間でタスクを実行できるため、呼び出し時間が最短になります。どのサービスがイベント呼び出しモードを取る可能性があるかの詳細については、「他のサービスで AWS Lambda を使用する」を参照してください。

Lambda の実行

Lambda がユーザーに代わって関数を実行する際、コードの実行に必要となる基盤のシステムのプロビジョニングと設定の両方が Lambda によって管理されます。これにより、デベロッパーは基盤となるシステムの管理ではなく、ビジネスロジックやコードの記述に集中できます。

Lambda サービスはコントロールプレーンとデータプレーンに分割されています。各プレーンは、サービスで別々の役割を果たします。コントロールプレーンは、管理 API (`CreateFunction`、`UpdateFunctionCode`、`PublishLayerVersion` など) を提供し、すべての AWS のサービスとの統合を管理します。Lambda のコントロールプレーンへの通信は、転送中に TLS により保護されます。Lambda のコントロールプレーンに保存されるすべての顧客データは、不正な開示や改ざんから保護するために設計された AWS KMS を使用して、保存時に暗号化されます。

データプレーンは、Lambda 関数の呼び出しをトリガーする Lambda の Invoke API です。Lambda 関数が呼び出されると、データプレーンは AWS Lambda ワーカー (または単に Amazon EC2 インスタンスの一種であるワーカー) 上の実行環境をその関数バージョンに割り当てるか、すでにセットアップされている既存の実行環境を選択します。この環境が呼び出しを完了するために使用されます。詳細については、このドキュメントの「AWS Lambda MicroVM とワーカー」セクションを参照してください。

Lambda 実行環境

各呼び出しは、Lambda の Invoke サービスによって、リクエストを処理できるワーカー上の実行環境にルーティングされます。お客様やその他のユーザーは、実行環境とのインバウンドネットワーク通信またはイングレスネットワーク通信をデータプレーン経由以外で直接開始することはできません。これにより、実行環境への通信が確実に認証および承認されます。

実行環境は特定の関数バージョン向けに予約されており、関数バージョン、関数、または AWS アカウント間で再利用することはできません。つまり、2 つの異なるバージョンを持つ関数には、少なくとも 2 つの一意の実行環境が生成されることになります。

各実行環境は、一度に 1 つの同時呼び出しに対してのみ使用できます。パフォーマンス上の理由から、同じ関数バージョンの複数の呼び出し間で再利用できます。さまざまな要因 (呼び出し速度、関数設定など) により、特定の関数バージョンに対して複数の実行環境が存在する場合もあります。このアプローチを採用すると、Lambda は関数のバージョンレベルを分離することができます。

Lambda は現在、関数バージョンの実行環境内での呼び出しを分離していません。つまり、ある呼び出しにより、次の呼び出し (/tmp に書き込まれたファイルやメモリ内のデータなど) に影響を与える可能性のある状態が残される場合があります。ある呼び出しにより、別の呼び出しに影響を与えることを避けるには、Lambda では追加の個別の関数を作成することをお勧めします。たとえば、エラーが発生する可能性が高い複雑な解析操作向けに個別の関数を作成したり、セキュリティに関する操作を実行しない関数を再利用したりすることができます。Lambda では現在、お客様が作成できる関数の数に制限はありません。制限の詳細については、「Lambda のクォータ」のページを参照してください。

実行環境は Lambda によって継続的にモニタリングおよび管理されていますが、次のようなさまざまな理由で作成または破棄される可能性があります。

- 新しい呼び出しが到着しても、適切な実行環境が存在しない場合
- 内部ランタイムまたはワーカーソフトウェアのデプロイがあった場合
- プロビジョニングされた同時実行数の設定が新しく公開された場合
- 実行環境またはワーカーのリース時間がライフタイムの上限に近づいているか、ライフタイムの上限を超えている場合
- その他の内部ワークロード再分散プロセス

関数設定にプロビジョニングされた同時実行数を設定すると、関数バージョンに存在する事前プロビジョニングされた実行環境の数を管理できます。この設定を行うと、Lambda は設定された数の実行環境を作成、管理し、その存在を常に確認します。これにより、規模を問わず、サーバーレスアプリケーションスタートアップ時のパフォーマンスを確実により詳細に制御できます。

呼び出しに応じて Lambda によって作成または管理される実行環境の数を確実に制御する方法は、プロビジョニングされた同時実行数を設定する以外にはありません。

実行ロール

それぞれの Lambda 関数には実行ロールを設定する必要があります。実行ロールとは、関数に関するコントロールプレーンおよびデータプレーンのオペレーションを実行する際に Lambda サービスが引き受ける IAM ロールです。Lambda サービスはこのロールを引き受け、一時的なセキュリティ認証情報をフェッチします。この認証情報は、関数の呼び出し中に環境変数として使用できます。Lambda サービスは、パフォーマンス上の理由で、この認証情報をキャッシュし、同じ実行ロールを使用する異なる実行環境で再利用する場合があります。

最小特権の原則を確実に遵守するため、Lambda では、各関数が固有のロールを持ち、必要最小限のアクセス権限のセットで設定を行うことをお勧めします。

Lambda サービスは、VPC 関数用の Elastic Network Interface (ENI) の作成と設定、Amazon CloudWatch Application Insights へのログの送信、AWS X-Ray へのトレースの送信に関連する操作や、呼び出しに関係のないその他のオペレーションなど、特定のコントロールプレーンオペレーションを実行するための実行ロールを引き受けることもあります。これらのユースケースは、AWS CloudTrail の監査ログでいつでも確認および監査できます。

このトピックの詳細については、「AWS Lambda 実行ロール」のドキュメントページを参照してください。

Lambda MicroVM とワーカー

Lambda は、AWS Lambda ワーカーと呼ばれる Amazon EC2 インスタンスのフリートに実行環境を作成します。ワーカーとはベアメタル EC2 Nitro インスタンスであり、Lambda によって、分離された AWS アカウントで、ユーザーには見えないところで起動および管理されます。ワーカーには、Firecracker により作成された 1 つまたは複数のハードウェア仮想化マイクロ仮想マシン (MVM) があります。Firecracker は、Linux のカーネルベースの仮想マシン (KVM) を使用して MVM を作成および管理する、オープンソースの仮想マシンモニター (VMM) です。サーバーレス運用モデルを提供する、安全性の高いマルチテナントコンテナおよび機能ベースのサービスの作成と管理を目的として構築されています。Firecracker のセキュリティモデルの詳細については、Firecracker プロジェクトの Web サイトを参照してください。

責任共有モデルの一環として、ワーカーのセキュリティ設定、コントロール、パッチ適用レベルを維持する責任は Lambda が担います。Lambda チームは、Amazon Inspector に加え、その他のカスタムセキュリティ問題通知メカニズムと事前開示リスト使用して、既知の潜在的なセキュリティ問題を検出するため、ユーザーが実行環境の基盤となるセキュリティ体制を管理する必要はなくなります。

図 3 – AWS Lambda ワーカーの分離モデル

ワーカーの最大リース有効期間は 14 時間です。ワーカーのリース有効期間の終わりに近づくと、それ以降は呼び出しのルーティングが行われず、MVM は正常終了され、基盤となるワーカーインスタンスは終了します。Lambda は、フリートのライフサイクルアクティビティを継続的にモニタリングし、アラームを発します。

ワーカーへのすべてのデータプレーン通信は、ガロア/カウンターモードによる高度暗号化標準 (AES-GCM) を使用して暗号化されます。ワーカーは、Lambda のサービスアカウントで Lambda が

管理する、ネットワーク分離された Amazon VPC でホストされているため、データプレーンオペレーション以外の手段ではワーカーを直接操作することはできません。

ワーカーが新しい実行環境を作成する必要がある場合、ワーカーにはお客様の関数アーティファクトにアクセスするための期限付きの権限が付与されます。これらのアーティファクトは、Lambda の実行環境とワーカー向けに最適化されています。ZIP 形式を使用してアップロードされた関数コードは一度最適化され、AWS が管理するキーと AES-GCM を使用して暗号化された形式で保存されます。

コンテナイメージ形式を使用して Lambda にアップロードされた関数も最適化されます。コンテナイメージは、まず元のソースからダウンロードされ、個別のチャンクに最適化された後、AES-CTR、AES-GCM、SHA-256 MAC の組み合わせを使用する認証済みコンバージェント暗号化方式を使用して暗号化されたチャンクとして保存されます。コンバージェント暗号化方式を採用することにより、Lambda は暗号化されたチャンクを安全に重複除去できます。顧客データの復号化に必要なすべてのキーは、カスタマー管理の AWS KMS カスタマーマスターキー (CMK) を使用して保護されます。Lambda サービスによる CMK の使用状況の追跡と監査は、AWS CloudTrail ログで提供されます。

Lambda 分離テクノロジー

Lambda では、ワーカーと実行環境を保護するために、さまざまなオープンソースおよび独自の分離テクノロジーが使用されています。各実行環境は、以下の項目の専用コピーで構成されます。

- 特定の関数バージョンのコード
- 関数バージョン用に選択された任意の AWS Lambda レイヤー
- 選択された関数ランタイム (Java 11、NodeJS 12、Python 3.8 など) または関数のカスタムランタイム
- 書き込み可能なディレクトリまたは tmp ディレクトリ
- Amazon Linux 2 を基盤とする最小限の Linux ユーザースペース

実行環境は、Linux カーネルに組み込まれている、コンテナに類似した複数のテクノロジーと AWS 独自の分離テクノロジーを使用して相互に分離されます。この技術には、次のようなものがあります。

- cgroups – 関数の CPU とメモリーへのアクセスを制限するために使用します。
- namespaces – 各実行環境は、専用の名前空間で実行されます。これは、Linux カーネルが管理する一意のグループプロセス ID、ユーザー ID、ネットワークインターフェイスなどのリソースを用いることで実現します。
- seccomp-bpf – 実行環境内から使用できるシステムコール (syscalls) を制限します。
- iptablesとルーティングテーブル – イングレスネットワーク通信を防止し、MVM 間のネットワーク接続を分離します。
- chroot – 基盤となるファイルシステムにアクセスできる範囲を指定します。
- Firecracker 設定 – ブロックデバイスとネットワークデバイスのスループットをレート制限するために使用します。
- Firecracker のセキュリティ機能 – Firecracker の現在のセキュリティ設計の詳細については、Firecracker の最新の設計ドキュメントを参照してください。

AWS 独自の分離技術とこれらのメカニズムを組み合わせることで、実行環境間を強力に分離できます。

ストレージと状態

実行環境は、異なる関数バージョンやお客様の間で再利用されることはありません。ただし、同じ関数バージョンの呼び出し間では 1 つの環境を再利用できます。つまり、呼び出し間でデータと状態が保持される場合があります。データや状態は、通常の実行環境のライフサイクル管理の一環として、破棄されるまで数時間保持される場合があります。パフォーマンス上の理由から、関数ではこの動作を利用して、呼び出し間でローカルキャッシュまたは長期間有効な保持して再利用し、共通の結果をあらかじめ演算処理しておくことで、効率を向上させることができます。実行環境内では、これらの複数の呼び出しは 1 つのプロセスによって処理されるため、呼び出しが再利用の実行環境で行われた場合、プロセス全体の状態 (Java の静的状態など) をその後の呼び出しで再利用できる可能性があります。

また、各 Lambda 実行環境には、書き込み可能なファイルシステムが `/tmp` にあります。このストレージはアクセスできず、実行環境間での共有もできません。プロセスの状態と同様、`/tmp` に書き込まれたファイルは、実行環境の存続期間中維持されます。これにより、機械学習 (ML) モデルのダウンロードなど、コストの高い転送オペレーションを複数の呼び出しで利用できます。呼び出し間でデータを維持する必要がない関数の場合、/tmp に書き込まないか、各呼び出し間に /tmp からファイルを削除する必要があります。`/tmp` ディレクトリは Amazon EC2 インスタンスストアによって保護され、保管時には暗号化されます。

実行環境外のファイルシステムにデータを保持する場合は、Lambda の Amazon Elastic File System (Amazon EFS) との統合を検討する必要があります。詳細については、「AWS Lambda で Amazon EFS を使用する」を参照してください。

呼び出し間でのデータや状態の保持を望まない場合は、Lambda で実行コンテキストや実行環境を使用してデータや状態を保存しないことをお勧めします。呼び出し間でのデータや状態の漏洩を積極的に防止するには、Lambda では状態ごとに異なる関数を作成することをお勧めします。呼び出し間で変更される可能性があるため、Lambda では、セキュリティに関する状態を使用したり、実行環境に保存したりすることはお勧めしません。代わりに、呼び出しのたびに状態を再計算することをお勧めします。

Lambda でのランタイムのメンテナンス

Lambda は、互換性のある更新やセキュリティパッチを継続的にスキャンしてデプロイしたり、その他のランタイムメンテナンスアクティビティを実行したりして、これらのランタイムをサポートします。これにより、使用する関数およびレイヤーに含まれるコードの保守とセキュリティだけに集中できます。Lambda チームは、Amazon Inspector に加え、その他のカスタムセキュリティ問題通知メカニズムと事前開示リスト使用して、既知のセキュリティ問題を検出し、ランタイム言語と実行環境にパッチが適用されていることを確認します。新しいパッチまたは更新が特定された場合、Lambda はユーザーの関与なしにランタイム更新をテストしてデプロイします。Lambda のコンプライアンスプログラムの詳細については、このドキュメントの「Lambda とコンプライアンス」セクションを参照してください。

通常、サポートされている Lambda ランタイムの最新パッチを取得するために操作を行う必要はありません。ただしパッチをデプロイする前にテストするための操作が必要になる場合があります (互換性のないことがわかっているランタイムパッチなど)。何らかの操作が必要な場合、Lambda は Personal Health Dashboard、AWS アカウントの E メール、またはその他の手段を通じて、必要な操作についてユーザーに通知します。

カスタムランタイムを実装すると、Lambda で他のプログラミング言語を使用できます。カスタムランタイムについては、カスタムランタイムに最新のセキュリティパッチが適用されていることを確認するなど、ランタイムに関するメンテナンスはお客様の責任となります。詳細については、AWS Lambda 開発者ガイドの「カスタム AWS Lambda ランタイム」を参照してください。

アップストリームのランタイム言語メンテナンス担当者が担当言語に End Of-Life (EOL) とマークすると、Lambda はこれを認識し、そのランタイム言語バージョンをサポートしなくなります。ランタイムバージョンが Lambda で廃止とマークされると、Lambda は、新しい関数の作成や、廃止されたランタイムで作成された既存の関数の更新のサポートを停止します。Lambda は、今後のランタイムの廃止について警告するため、今後の廃止予定日と予想される内容の通知をお客様に送信します。Lambda では、廃止となったランタイムにセキュリティ更新、テクニカルサポート、修正プログラムを提供しません。また、AWS は、廃止となったランタイムで実行するように設定された関数の呼び出しを任意の時点で無効にする権利を留保します。廃止となったランタイムのバージョンまたはサポートされていないランタイムバージョンを引き続き実行する場合は、独自のカスタム AWS Lambda ランタイムを作成できます。ランタイムの廃止予定の詳細については、「AWS Lambda ランタイムサポートポリシー」を参照してください。

Lambda 関数のモニタリングと監査

Lambda 関数は、AWS が提供する多数のサービスと方法によってモニタリングおよび監査することができます。例えば、次のようなサービスがあります。

Amazon CloudWatch

AWS Lambda では、お客様の代わりに Lambda 関数を自動的にモニタリングします。Amazon CloudWatch を使用して、リクエストの数、リクエストごとの実行時間、エラーとなったリクエストの数といったメトリクスを報告します。これらのメトリクスは関数レベルで公開されるため、これらのメトリクスを利用して CloudWatch アラームを設定できます。Lambda によって公開されるメトリクスのリストについては、「AWS Lambda のメトリクス」を参照してください。

Amazon CloudTrail

AWS CloudTrail を使用すると、Lambda を含む AWS アカウント全体のガバナンス、コンプライアンス、運用の監査、リスクの監査を実装できます。CloudTrail では、AWS インフラストラクチャ全体のアクションに関連するアカウントアクティビティのログ記録、継続的なモニタリング、保持を行うことができるため、AWS マネジメントコンソール、AWS SDK、コマンドラインツール、AWS のその他のサービスから実行されたアクションの包括的なイベント履歴を得ることができます。CloudTrail では、AWS KMS を使用してログファイルを必要に応じて暗号化できます。また、CloudTrail ログファイルの整合性の検証を利用してポジティブアサーションを行うこともできます。

AWS X-Ray

AWS X-Ray を使用すると、本番環境の Lambda ベースの分散型アプリケーションを分析およびデバッグできます。これにより、アプリケーションと基盤となるサービスのパフォーマンスを理解できるため、パフォーマンスの問題とエラーの根本原因を特定し、トラブルシューティングを行うことができるようになります。X-Ray では、リクエストがアプリケーション内を通過する際にリクエストのエンドツーエンドのビューが提供され、アプリケーションの基盤となるコンポーネントのマップも表示されるため、開発中でも本番環境でもアプリケーションを分析できます。

AWS Config

AWS Config を使用すると、Lambda 関数 (削除済みの関数を含む)、ランタイム環境、タグ、ハンドラー名、コードのサイズ、メモリの割り当て、タイムアウトの設定、同時実行の設定、Lambda IAM 実行ロール、サブネット、セキュリティグループの関連付けに対する設定変更を追跡できます。これにより、Lambda 関数のライフサイクルを包括的に表示できるため、考えられる監査とコンプライアンスの要件に合わせてそのデータを表示できます。

Lambda 関数のアーキテクチャ設計と運用

このセクションでは、Lambda のアーキテクチャと運用について説明します。サーバーレスアプリケーションの標準ベストプラクティスの詳細については、サーバーレスアプリケーションレンズホワイトペーパーを参照してください。このホワイトペーパーでは、サーバーレスでの AWS Well Architected Framework の柱を定義し、説明しています。

- 運用上の優秀性の柱 – ビジネス価値を提供し、継続的にプロセスと手順のサポートを改善するために、システムを運用およびモニタリングする能力。
- セキュリティの柱 – リスク評価とリスク軽減の戦略を通してビジネス価値を提供しながら、情報、システム、資産を保護する能力。
- 信頼性の柱 – インフラストラクチャやサービスの障害からの復旧、必要に応じた動的なコンピューティングリソースの獲得、設定ミスや一時的なネットワークの問題などによる障害の軽減といったシステムの能力。
- パフォーマンス効率の柱 – コンピューティングリソースを効率的に使って要件を満たし、需要が変化し技術が発展するのに合わせて効率性を維持する能力。
- コスト最適化の柱 — 需要の変化やテクノロジーの進化に応じてコストを最小限に抑えると同時に、ビジネス上の成果を確実に達成するための改良と改善の継続的なプロセス。

サーバーレスアプリケーションレンズホワイトペーパーには、メトリクスのログ記録とアラーム、スロットリングと制限、Lambda 関数へのアクセス許可の割り当て、Lambda 関数で機密データを利用できるようにする方法などのトピックが含まれています。

Lambda とコンプライアンス

「責任共有モデル」の項で説明したとおり、データに適用されるコンプライアンス制度を確認する責任はお客様にあります。コンプライアンス制度の要件を確認したら、Lambda のさまざまな機能を使用してそれらの規制に対応できます。お客様は、AWS エキスパート (ソリューションアーキテクト、ドメインエキスパート、テクニカルアカウントマネージャー、その他の担当者) に支援を求めることができます。ただし、AWS は、特定のユースケースにコンプライアンス制度が適用されるかどうか、またはどのコンプライアンス制度が適用されるかについてお客様にアドバイスを提供することはできません。

2020 年 11 月現在、Lambda は SOC 1、SOC 2、SOC 3 報告書の対象となります。SOC 1、SOC 2、および SOC 3 報告書は、重要なコンプライアンス管理および目標を AWS がどのように達成したかを実証する、独立したサードパーティーによる審査レポートです。コンプライアンス情報の最新情報のリストについては、「コンプライアンスプログラムの対象範囲となる AWS のサービス」ページを参照してください。

一部のコンプライアンスレポートは機密であるため、公開して共有することができません。これらのレポートにアクセスするには、AWS マネジメントコンソールにサインインして AWS Artifact を使用します。これは、無料のセルフサービスポータルで、AWS のコンプライアンスレポートにオンデマンドでアクセスできます。

Lambda イベントソース

Lambda は、ダイレクトなインテグレーションと Amazon EventBridge のイベントバスを介して、140 種類以上の AWS のサービスと統合しています。よく利用される Lambda イベントソースは次のとおりです。

- 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

これらのイベントソースを使用すると、次のことが実現できます。

- AWS Identity and Access Management を使用して、サービスおよびリソースへのアクセスを安全に管理することができます。

- 保存中のデータを暗号化できます。* すべてのサービスで、転送中のデータは暗号化されます。
- AWS PrivateLink 提供の VPC エンドポイントを使用して Amazon Virtual Private Cloudからアクセスできます。
- Amazon CloudWatch Application Insights を使用して、メトリクスを収集し、メトリクスを基にした報告、警告を生成できます。
- AWS CloudTrail を使用して、AWS インフラストラクチャ全体のアクションに関連するアカウントアクティビティの口グ記録、継続的なモニタリング、保持を行うことができるため、AWS マネジメントコンソール、AWS SDK、コマンドラインツールなど、AWS のサービスから実行されたアクションの包括的なイベント履歴を取得できます。

*この記事の公開時点では、Amazon EventBridge では保管中のデータの暗号化は提供されていませんでした。機能更新の確認のために、サービスのホームページを継続的にモニタリングできます。

まとめ

AWS Lambda には、安全でスケーラブルなアプリケーションを構築するための強力なツールキットが用意されています。Lambda のセキュリティとコンプライアンスに関するベストプラクティスの多くは、AWS のすべてのサービスと同じですが、一部のベストプラクティスは Lambda に固有のものです。このホワイトペーパーでは、Lambda の利点、アプリケーションに対する適性、Lambda マネージドランタイム環境について説明しました。また、モニタリングと監査、セキュリティとコンプライアンスのベストプラクティスについても説明しました。次回の実装を検討する際は、AWS Lambda について学んだことを活かし、どうすれば次のワークロードソリューションをより良いものにすることができるのか考えてみてください。

寄稿者

この文書の寄稿者は次のとおりです。

- Mayank Thakkar、グローバルライフサイエンスソリューションアーキテクト
- Marc Brooker、シニアプリンシパルエンジニア (サーバーレス)
- Osman Surkatty、シニアセキュリティエンジニア (サーバーレス)

その他の資料

詳細については、次の資料を参照してください。

- 「責任共有モデル」では、セキュリティ全般についての AWS の考え方を説明しています。
- 「AWS におけるセキュリティのベストプラクティス」では、AWS Identity and Access Management (IAM) サービスについての推奨事項が説明されています。
- 「サーバーレスアプリケーションレンズ」では、AWS Well-Architected Framework について説明し、ベストプラクティスに従ってワークロードのアーキテクチャを確実に設計するための主な要素を取り上げています。
- 「AWS セキュリティ入門」では、AWS のセキュリティについての考えを幅広く紹介しています。
- 「AWS のリスクとコンプライアンス」では、AWS のコンプライアンスの概要を説明しています。

改訂履歴

このホワイトペーパーの更新に関する通知を受け取るには、RSS フィードをサブスクライブしてください。

update-history-change	update-history-description	update-history-date
更新	重要な更新	2021 年 2 月 15 日
初版発行	ホワイトペーパーの初版公開	2019 年 1 月 3 日

注意

お客様は、この文書に記載されている情報を独自に評価する責任を負うものとします。本書は、(a) 情報提供のみを目的とし、(b) AWS の現行製品と慣行について説明しており、これらは予告なしに変更されることがあり、(c) AWS およびその関連会社、サプライヤーまたはライセンサーからの契約上の義務や保証をもたらすものではありません。AWS の製品やサービスは、明示または暗示を問わず、一切の保証、表明、条件なしに「現状のまま」提供されます。お客様に対する AWS の責任は、AWS 契約により規定されます。本書は、AWS とお客様の間で締結されるいかなる契約の一部でもなく、その内容を修正するものでもありません。