AWS Lambda のセキュリティの概要

Lambda のセキュリティを詳しく理解する

2019 年 3 月

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。この文書は、(a) 情報提供のみを目的としており、(b) AWS の現行製品と慣行を表したものであるため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義務や確約を意味する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、いかなる保証、表明、条件を伴うことなく「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 とお客様の間で行われるいかなる契約の一部でもなく、そのような契約の内容を変更するものでもありません。

目次

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、最高情報セキュリティ責任者 (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、コンプライアンスチーム、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

AWS Lambda では基盤インフラストラクチャ管理の必要がありません。現在、ますます多くのお客様が AWS Lambda で、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。AWS Lambda ならば、1 秒あたり数千件の同時リクエストレベルまでワークロードをスケール可能です。AWS の多彩に展開する根幹サービスのうちの 1 つです。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。

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 AWS Lambda に注目が集まっています。特に、事業の運営に集中し、市場投入までの時間の短縮、コストの最適化、俊敏性の向上を実現したいとお考えの企業で大きな反響を呼んでいます。Lambda は、多くの業界でミッションクリティカルなアプリケーションに採用されることが多くなってきています。

AWS Lambda の**マネージドランタイム環境**モデルでは、実装のさまざまな細部を意図的にユーザーから見えなくしているため、クラウドセキュリティの既存のベストプラクティスに代わる、新しいベストプラクティスが必要になります。このホワイトペーパーでは、開発者、セキュリティアナリスト、コンプライアンスチーム、その他の関係者のみなさまにより一層ご理解いただけるよう、そうしたベストプラクティスや、Lambda の基礎に関する情報をご提供します。

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ご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簡単に整えることができます。

Lambda の利点

多くの企業で、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 ならば、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で、煩雑な運用を不要にします。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。

秒以下の単位での計測

AWS Lambda では、コードが実行される 100 ミリ秒ごと、およびコードがトリガーされた回数による課金となります。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を計測します。

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のみなさまを煩雑な管理から解放します。担当者のみなさまに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していただけます。

アプリケーションを最新化

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、人工知能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 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。

豊富なエコシステム

Lambda には豊富なエコシステムがあります。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 SAM、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 IDE との統合により、開発者をサポートしています。さらに、Lambda には AWS のその他のサービスが統合されているので、サーバーレスアプリケーションを構築するめの豊富なエコシステムが利用可能です。

Lambda ベースアプリケーションの実行料金

Lambda では、きめ細かい従量課金制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。お客様は、ゲストオペレーティングシステム (更新やセキュリティパッチなど)、その他の関連アプリケーションソフトウェア、ならびに AWS より提供されるセキュリティグループファイアウォールの設定に対する責任とその管理を引き受けます。

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、コードのセキュリティ、機密データの保管とアクセス、Lambda サービスに対するアイデンティティとアクセスの管理 (IAM)、関数内のアイデンティティとアクセスの管理について責任を負います。

図 1 に AWS Lambda の責任共有モデルを示します。 AWS の責任はオレンジ色で、お客様の責任は青色で示しています。AWS は、Lambda にデプロイされたアプリケーションについてより多くの責任を負います。

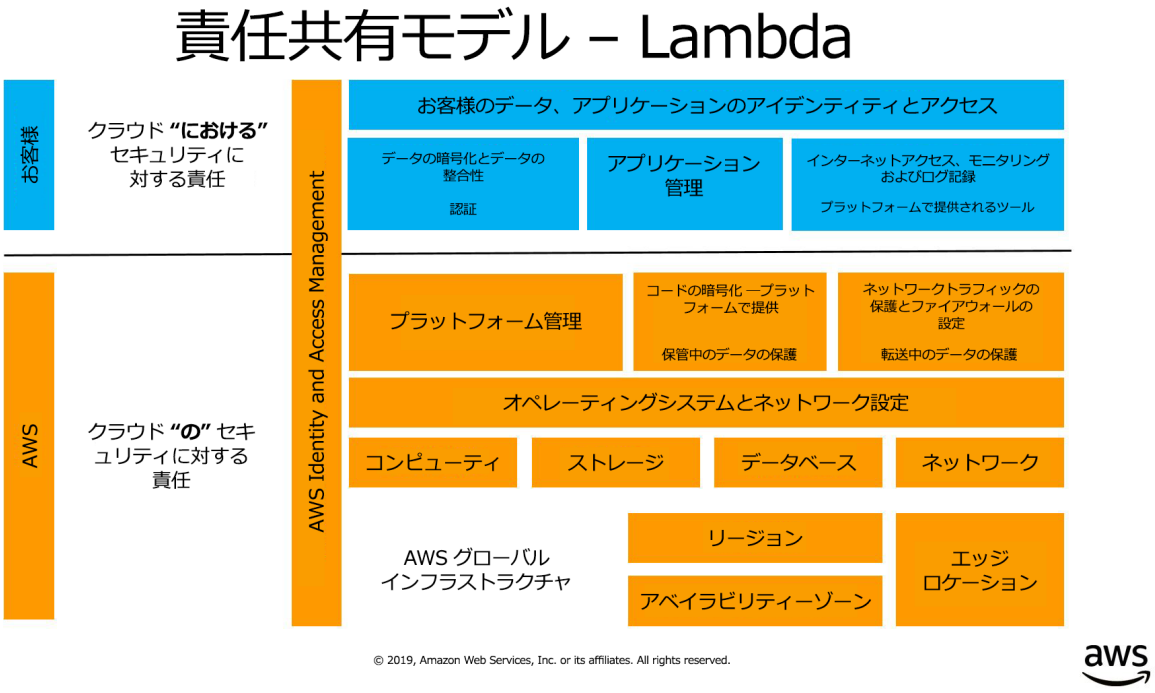


図 1 – AWS Lambda の責任共有モデル

Lambda ランタイム環境

Lambda では、お客様の代わりに関数を実行するときに、コードの実行に必要なプロビジョニングとリソースの両方を管理します。これにより、開発者は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では、関数管理 API (**CreateFunction**、**UpdateFunctionCode**) を提供したり、AWS のすべてのサービスとの統合を管理します。データプレーンでは、Lambda 関数を実行する **Invoke** API を制御します。Lambda 関数が呼び出されると、データプレーンでは**実行環境**をその関数に割り当てるか、その関数用に用意されている既存の実行環境を選択し、その環境で関数コードを実行します。

各関数は 1 つ以上の専用の実行環境で実行され、実行環境は、関数の存続期間中使用され、その後破棄されます。各実行環境では 1 つの同時呼び出しをホストしますが、実行環境は同じ関数の複数の順次呼び出しで再利用されます。実行環境は、ハードウェア仮想化仮想マシン (microVM) で実行されます。microVM は 1 つの AWS アカウント専用ですが、同じアカウント内の関数全体で実行環境によって再利用できます。microVM は、AWS が所有および管理するハードウェアプラットフォーム (Lambda ワーカー) に配置されます。実行環境が関数間で共有されることはありません。また、microVM が AWS アカウント間で共有されることもありません。

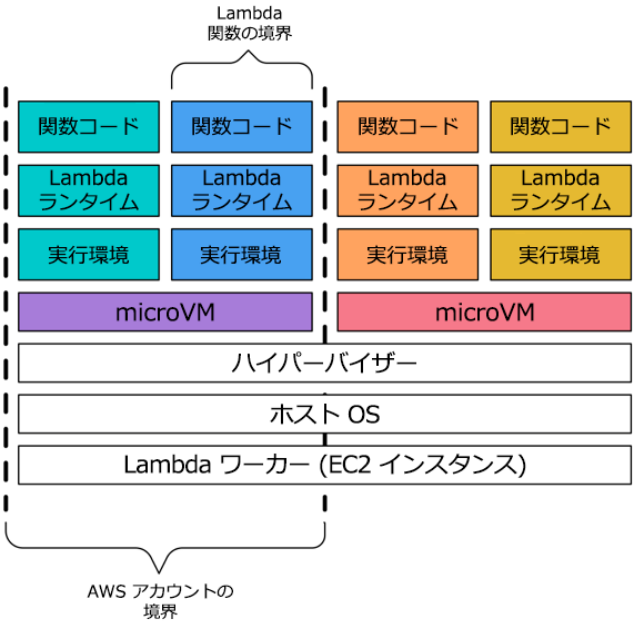


図 2 – AWS Lambda の分離モデル

関数の間および microVM 間でのセキュリティ分離

各実行環境は、以下の項目の専用コピーで構成されます。

- 関数コード
- 関数用に選択された任意の Lambda レイヤー
- 関数のランタイム (組み込みランライム [Java、NodeJS、Python など] またはカスタムランタイム)
- Amazon Linux ベースの最小限の Linux ユーザーランド

実行環境は、Linux カーネルに組み込まれている複数のコンテナ技術を使用して相互に分離されます。この技術には、以下のようなものがあります。

- **cgroups** – 実行環境ごとに制限 CPU、メモリ、ディスクスループット、ネットワークスループットへのリソースのアクセスを制限します。
- **namespaces** – Linux カーネルによって管理されるプロセス ID、ユーザー ID、ネットワークインターフェイス、その他のリソースをグループ化します。各実行環境は、専用の名前空間で実行されます。
- **seccomp-bpf** – 実行環境内から使用できるシステムコールを制限します。
- **iptables とルーティングテーブル** – 実行環境を互いに分離します。
- **chroot** – 基盤となるファイルシステムでアクセスできる範囲を指定します。

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。この分離により、別の環境に配置されたデータにアクセスしたり、変更したりすることができなくなります。
1 つの AWS アカウントから実行される複数の実行環境は 1 つの microVM で実行できますが、microVM が AWS アカウント間で共有または再利用されることはありません。現時点では、AWS Lambda では、EC2 インスタンスと Firecracker の 2 つのメカニズムを使用して microVM を分離しています。EC2 インスタンスは、2015 年から Lambda のゲスト分離に使用されています。Firecracker は、特にサーバーレスワークロード向けに AWS が開発した新しいオープンソースのハイパーバイザーで、2018 年に導入されました。microVM を実行する基盤となる物理ハードウェアは、複数のアカウントからのワークロードで共有されます。詳細については、**付録 – Lambda EC2 モデルと Firecracker モデル**を参照してください。

ストレージと状態

Lambda 実行環境は関数をまたがって再利用されることはありませんが、1 つの実行環境は同じ関数を呼び出すために再利用できます。このため、実行環境は破棄されるまで数時間存続する可能性があります。関数ではこの動作を利用して、ローカルキャッシュを維持し、呼び出し間の存続期間の長い接続を再利用し、共通の結果をあらかじめ演算処理することで、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**スタティックな**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、他の実行環境からはアクセスできません。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 に書き込まないようにするか、各呼び出し後に /tmp からファイルを削除するようにしてください。/tmp ストレージは、Amazon Elastic Block Store (Amazon EBS) または Lambda ワーカーインスタンスのローカルストレージを使用して実装されています。
また、関数の最初の呼び出し前に、Lambda では、メモリを実行環境に割り当てる前にメモリを消去します。こうすることで、同じアカウントに属する関数と、異なるお客様アカウントの間でメモリが共有されるのを効果的に防ぐことができます。Lambda では、実行環境の再利用を容易にするために、同じ実行環境における同じ関数に対する

初回以降の呼び出しの前にはメモリを消去しません。ユーザーは関数の終了前にメモリを暗号化し、消去するプロセスを自分で実装できます。

呼び出しデータパス

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、後で実行するために呼び出しがキューに配置されます。**request-response** モードでは、指定されたペイロードで関数をすぐに呼び出し、応答を返します。どちらの場合も、実際の関数の実行は 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、「Lambda ランタイム環境」セクションを参照してください。

request-response 呼び出しの場合、ペイロードは、Amazon API Gateway、AWS SDK などの API 呼び出し元からロードバランサーに向かい、**Lambda 呼び出しサービス**に到達します。このサービスでは、関数の実行環境を識別し、呼び出しを完了するためにその実行環境にペイロードを渡します。ロードバランサーへのトラフィックはインターネットを経由し、TLS で保護されます。Lambda サービス内のトラフィック (ロードバランサー以降のトラフィック) は、1 つの AWS リージョン内の Lambda 内部 VPC を通ります。

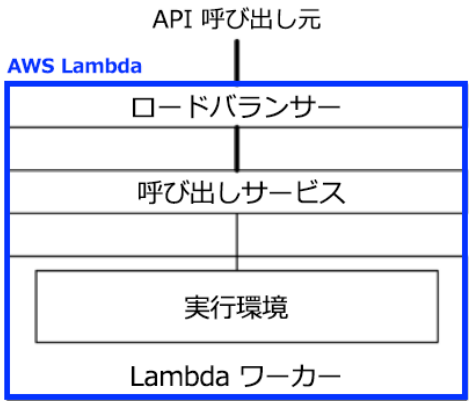


図 3 – AWS Lambda の呼び出しモデル: request-response

event 呼び出しは、すぐに実行するか、処理のためにキューに配置することができます。場合によっては、キューは Amazon Simple Queue Service (Amazon SQS) を使用して実装され、内部の Poller プロセスによって **Lambda 呼び出しサービス**に戻されます。このパスのトラフィックは TLS で暗号化されますが、Amazon SQS に保存されるデー

タに対してその他の暗号化は行われません。ただし 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、お客様に表示されることはありません。イベント呼び出しの場合、応答は返されず、応答データはワーカーによって破棄されます。Amazon S3、Amazon SNS、CloudWatch Events、その他のイベントソースからの呼び出しは、Lambda サービスの **event** 呼び出しパスを通ります。Amazon Kinesis と DynamoDB のストリーム、SQS キュー、Application Load Balancer、API Gateway からの呼び出しは、**request-response** パスを通ります。

Lambda でのランタイムのメンテナンス

AWS Lambda では、Java、Python、Go、NodeJS、.NET core などのランタイムを使用することで、複数のプログラミング言語をサポートしています。すべてのリストについては、「AWS Lambda ランタイム」を参照してください。Lambda では、更新、セキュリティパッチ、その他のメンテナンスについて、これらのランタイムにサポートを提供しています。通常、Lambda ランタイムの最新のパッチを入手するために操作を行う必要はありませんが、場合によってはパッチをデプロイする前にパッチをテストするための操作が必要になるかもしれません。そのような場合は、(AWS Personal Health Dashboard などを経由して) AWS からご連絡いたします。

Lambda ではカスタムランタイムを実装することで、他の言語を使用できます。カスタムランタイムについては、最新のセキュリティパッチを必ず適用するなどの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。

Lambda では、ランタイムのバージョンが提供側の保守方針としてサポート終了 (EOL) に指定された場合、このバージョンを廃止する場合があります。廃止とマークされたバージョンに対しては、リスク回避のために、関数の新規作成や廃止ランタイムで作成された既存の関数のコードの更新を停止させていただきます。AWS Lambda では、廃止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。ランタイムの廃止予定の詳細については、「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 が提供する多数の方法とサービス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、以下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

Amazon 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、Amazon Key Management Service (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、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 (削除された関数の情報を含む)。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

これまで、Lambda サービスの基礎について説明しました。ここからは、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、サーバーレスにおける AWS Well Architected フレームワークの柱を定義し、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。

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」セクションで説明したように、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、領域専門家、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。

2019/03 時点で、Lambda は SOC 1、SOC 2、SOC 3、PCI DSS、米国医療保険の相互運用性と説明責任に関する法令 (HIPAA) などに準拠しています。コンプライアンス情報の詳細については、「コンプライアンスプログラムによる AWS 対象範囲内のサービス」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 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。

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。AWS 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。次回の実装について考えるときに、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
- Brian McNamara、スペシャリストテクニカルアカウントマネージャー (サーバーレス)
- 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、以下を参照してください。

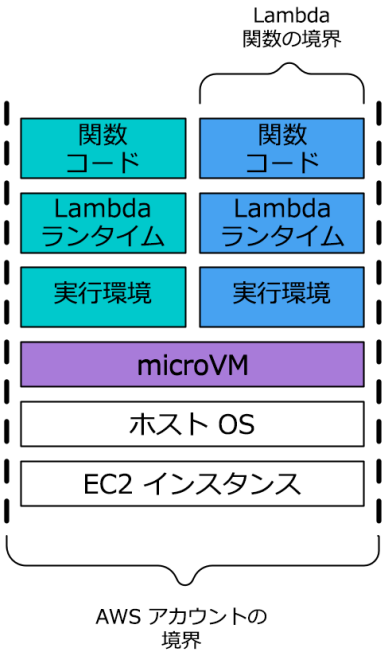
- AWS セキュリティのベストプラクティスでは、全般的な AWS セキュリティと AWS のさまざまなサービスでの AWS セキュリティのベストプラクティスについて説明しています。また、責任共有モデル、IAM ロールとアカウントを管理するためのベストプラクティスについても説明しています。
- サーバーレスアプリケーションレンズでは、AWS Well-Architected フレームワークについて説明し、ベストプラクティスに従ってワークロードのアーキテクチャを確実に設計するための主な要素を取り上げています。
- AWS セキュリティ入門では、AWS のセキュリティについての考えを幅広く紹介しています。
- 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について説明しています。

ドキュメントの改訂

日付	説明
2019 年 3 月	初版発行

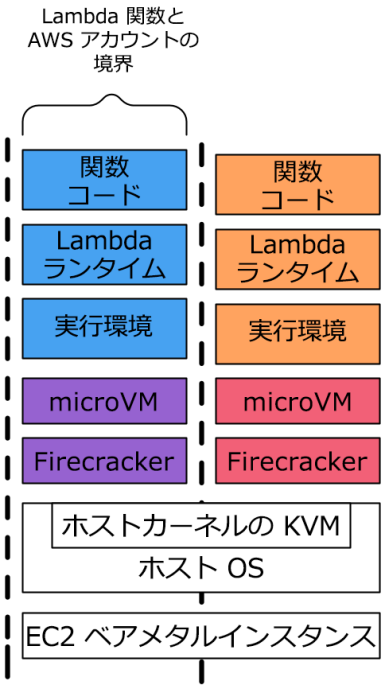
付録
– Lambda EC2 モデルと Firecracker モデル

以下に、AWS Lambda の EC2 モデルと Firecracker モデルの比較を示します。



Lambda の EC2 モデル

microVM と EC2 インスタンス間は
1 対 1 でマッピングされています。



Lambda の Firecracker モデル

実行環境と microVM 間は
1 対 1 で
マッピングされています。