aws

ユーザーガイド

Elastic Load Balancing

Elastic Load Balancing: ユーザーガイド

Table of Contents

Elastic Load Balancing とは?

Elastic Load Balancing は、受信したトラフィックを複数のアベイラビリティーゾーンの複数のターゲット (EC2 インスタンス、コンテナ、IP アドレスなど) に自動的に分散させます。登録されているターゲットの状態をモニタリングし、正常なターゲットにのみトラフィックをルーティングします。Elastic Load Balancing は、着信トラフィックの変化に応じて、自動的にロードバランサーの容量を拡張します。

ロードバランサーの利点

ロードバランサーは、ワークロードを仮想サーバーなど複数のコンピューティングリソース間に分散させます。ロードバランサーを使用すると、アプリケーションの可用性と耐障害性が向上します。

アプリケーションへのリクエストの流れを中断することなく、ニーズの変化に応じてロードバランサーに対してコンピューティングリソースの追加と削除を行うことができます。

ロードバランサーが正常なものにのみリクエストを送信するように、コンピューティングリソースのヘルス状態をモニタリングするヘルスチェックを設定できます。コンピューティングリソースがメインワークに集中できるように、暗号化および復号の作業をロードバランサーに任せることもできます。

Elastic Load Balancing の特徴

Elastic Load Balancing は、複数のロードバランサータイプをサポートしています。ニーズに最適なタイプのロードバランサーを選択できます。詳細については、「Elastic Load Balancing の機能」の「」を参照してください。

現行世代のロードバランサーの詳細については、次のドキュメントを参照してください。

- Application Load Balancer のユーザーガイド
- Network Load Balancer のユーザーガイド
- Gateway Load Balancer のユーザーガイド

Classic Load Balancer は、Elastic Load Balancing の前世代のロードバランサーです。現行世代のロードバランサーに移行することをお勧めします。詳細については、「Classic Load Balancer の移行」を参照してください。

Elastic Load Balancing へのアクセス

次のインターフェイスのいずれかを使用して、ロードバランサーの作成、アクセス、管理を行うことができます。

- AWS マネジメントコンソール — Elastic Load Balancing へのアクセスに使用できるウェブインターフェイスを提供します。
- AWS コマンドラインインターフェイス (AWS CLI) — Elastic Load Balancing を含む幅広い AWS サービスのコマンドを提供します。 AWS CLI は Windows、macOS、Linux でサポートされています。詳細については、「AWS Command Line Interface」を参照してください。
- AWS SDKs — 言語固有の APIsを提供し、署名の計算、リクエストの再試行の処理、エラー処理など、接続の詳細の多くを処理します。詳細については、AWS SDK をご参照ください。
- クエリ API— HTTPS リクエストを使用して呼び出す低レベル API アクションを提供します。クエリ API の使用は、Elastic Load Balancing にアクセスする最も直接的な方法です。ただし、クエリ API では、リクエストに署名するハッシュの生成やエラー処理など、低レベルの詳細な作業をアプリケーションで処理する必要があります。詳細については次を参照してください:
  - Application Load Balancer、Network Load Balancer、Gateway Load Balancer — API バージョン 2015-12-01
  - Classic Load Balancer — API バージョン 2012-06-01

関連する のサービス

Elastic Load Balancing は、アプリケーションの可用性とスケーラビリティを高める以下のサービスを使用します。

- Amazon EC2 — クラウドでアプリケーションを実行する仮想サーバーです。EC2 インスタンスへのトラフィックをルーティングするように、ロードバランサーを設定できます。詳細については、「Amazon EC2 ユーザーガイド」を参照してください。
- Amazon EC2 Auto Scaling — インスタンスに障害が発生した場合でも、必要な数のインスタンスを実行していることを確認します。Amazon EC2 Auto Scaling を使用すると、インスタンスに対する需要の変化に応じて、インスタンスの数を自動的に増減することもできます。Elastic Load Balancing で Auto Scaling を有効にすると、Auto Scaling によって起動されるインスタンスは自動的にロードバランサーに登録されます。同様に、Auto Scaling によって終了されたインスタンスは、ロードバランサーから自動的に登録解除されます。詳細については、「Amazon EC2 Auto Scaling ユーザーガイド」を参照してください。

- AWS Certificate Manager — HTTPS リスナーを作成するには、ACM で提供された証明書を指定できます。ロードバランサーは、証明書を使用して接続を終了し、クライアントからのリクエストを復号します。
- Amazon CloudWatch — ロードバランサーを監視し、必要に応じてアクションを実行することができます。詳細については、Amazon CloudWatch ユーザーガイドを参照してください。
- Amazon ECS — EC2 インスタンスのクラスター上で Docker コンテナを実行、停止、管理することができます。コンテナにトラフィックをルーティングするように、ロードバランサーを設定できます。詳細については、Amazon Elastic Container Service デベロッパーガイドを参照してください。
- AWS Global Accelerator — アプリケーションの可用性とパフォーマンスが向上します。アクセラレーターを使用して、1 つ以上の AWS リージョンの複数のロードバランサーにトラフィックを分散します。詳細については、「AWS Global Accelerator デベロッパーガイド」を参照してください。
- Route 53 — ドメイン名を、コンピュータが相互の接続に使用する数字の IP アドレスに変換することで、閲覧者をウェブサイトにルーティングするための信頼性が高く、コスト効率のよい方法を提供します。たとえば、数値 IP アドレス `www.example.com` に変換されます`192.0.2.1`。 は、ロードバランサーなどのリソースに URLsを AWS 割り当てます。ただし、ユーザーが覚えやすい URL を使用することもできます。たとえば、ドメイン名をお客様のロードバランサーにマッピングすることができます。詳細については、Amazon Route 53 デベロッパーガイドを参照してください。
- AWS WAF — Application Load Balancer AWS WAF で を使用して、ウェブアクセスコントロールリスト (ウェブ ACL) のルールに基づいてリクエストを許可またはブロックできます。詳細については、AWS WAF デベロッパーガイドを参照してください。

料金

ロードバランサーについては、お客様が利用された分のみのお支払いとなります。詳細については、Elastic Load Balancing の料金表を参照してください。

Elastic Load Balancing の仕組み

ロードバランサーは、クライアントからの受信トラフィックを受け入れ、リクエストを 1 つ以上のアベイラビリティーゾーンにある登録済みのターゲット (EC2 インスタンスなど) にルーティングします。また、ロードバランサーは登録されているターゲットの状態を監視して、トラフィックが正常なターゲットにのみルーティングされるようにします。ロードバランサーは、異常なターゲットを検出すると、そのターゲットへのトラフィックのルーティングを中止します。その後、ターゲットが再び正常になったことを検出すると、そのターゲットへのトラフィックのルーティングを再開します。

1 つ以上のリスナーを指定することで、受信トラフィックを受け入れるようにロードバランサーを設定します。リスナーとは接続リクエストをチェックするプロセスです。これは、クライアントからロードバランサーへの接続用のプロトコルとポート番号を使用して設定します。同様に、ロードバランサーからターゲットへの接続用のプロトコルとポート番号を使用して設定します。

内容

アベイラビリティーゾーンとロードバランサーノード

ロードバランサー用のアベイラビリティーゾーンを有効にすると、Elastic Load Balancing はアベイラビリティーゾーンにロードバランサーノードを作成します。ターゲットをアベイラビリティーゾーンに登録したが、アベイラビリティーゾーンを有効にしていない場合、登録したターゲットはトラフィックを受信しません。有効な各アベイラビリティーゾーンに 1 つ以上の登録済みターゲットが含まれるようにすると、ロードバランサーは最も効果的に機能します。

すべてのロードバランサーに対して複数のアベイラビリティーゾーンを有効にすることをお勧めします。ただし、Application Load Balancer では、少なくとも 2 つ以上のアベイラビリティーゾーンを有効にする必要があります。この設定により、ロードバランサーが引き続きトラフィックをルーティングできるようになります。1 つのアベイラビリティーゾーンが利用できなくなるか、正常なターゲットがなくなった場合、ロードバランサーは別のアベイラビリティーゾーンの正常なターゲットにトラフィックをルーティングできます。

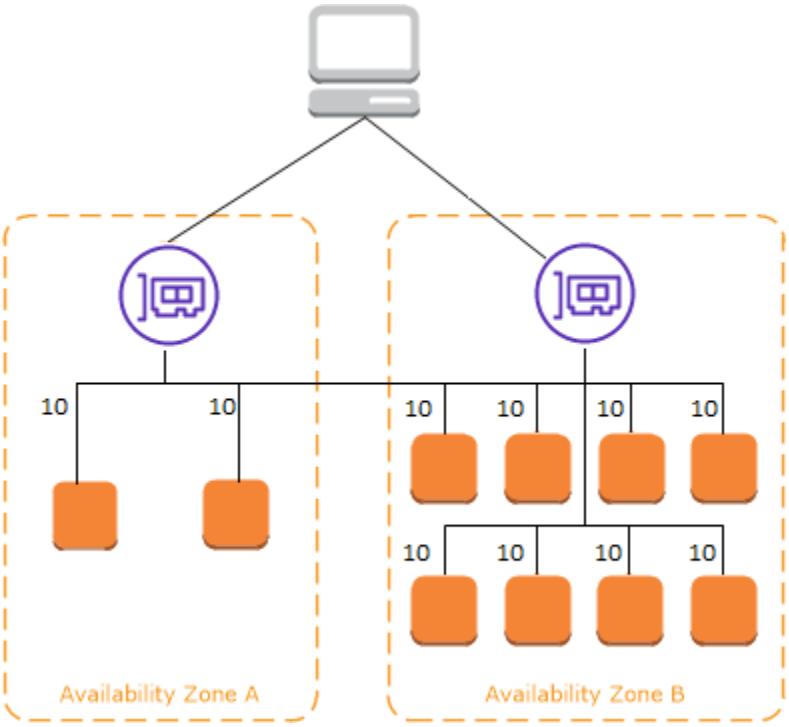
アベイラビリティーゾーンを無効にしても、そのアベイラビリティーゾーンのターゲットはロードバランサーに登録されたままになります。ただし、登録されたままであっても、ロードバランサーはトラフィックをルーティングしません。

クロスゾーン負荷分散

ロードバランサーのノードは、クライアントからのリクエストを登録済みターゲットに分散させます。クロスゾーン負荷分散が有効な場合、各ロードバランサーノードは、有効なすべてのアベイラビリティーゾーンの登録済みターゲットにトラフィックを分散します。クロスゾーン負荷分散が無効な場合、各ロードバランサーノードは、そのアベイラビリティーゾーンの登録済みターゲットにのみトラフィックを分散します。

次の図はラウンドロビンをデフォルトのルーティングアルゴリズムとして使用したクロスゾーンロードバランシングの効果について示しています。有効なアベイラビリティーゾーンが 2 つがあり、アベイラビリティーゾーン A には 2 つのターゲット、アベイラビリティーゾーン B には 8 つのターゲットがあります。クライアントがリクエストを送信すると、Amazon Route 53 はロードバランサーノードのいずれか 1 つの IP アドレスを使用して各リクエストに応答します。ラウンドロビンルーティングアルゴリズムに基づいて、各ロードバランサーノードがクライアントからのトラフィックの 50% を受け取るようにトラフィックが分散されます。各ロードバランサーノードは、範囲内の登録済みターゲット間で配分されたトラフィックを分散します。

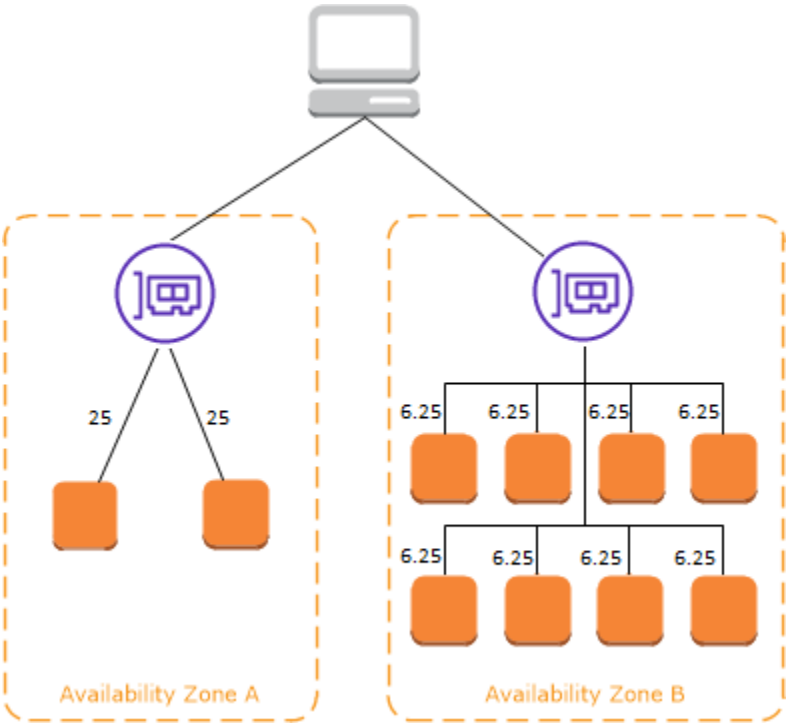
クロスゾーン負荷分散が有効な場合、10 個のターゲットのそれぞれがトラフィックの 10% を受け取ります。これは、各ロードバランサーノードが、そのクライアントトラフィックの 50% を 10 個のターゲットすべてにルーティングできるためです。



クロスゾーン負荷分散が無効な場合:

- アベイラビリティーゾーン A の 2 つのターゲットはそれぞれ、トラフィックの 25% を受け取ります。
- アベイラビリティーゾーン B の 8 つのターゲットはそれぞれ、トラフィックの 6.25% を受け取ります。

これは、各ロードバランサーノードが、そのクライアントトラフィックの 50% を自身のアベイラビリティーゾーンのターゲットにのみルーティングできるためです。



Application Load Balancer では、クロスゾーン負荷分散がロードバランサーレベルで常に有効になっています。ターゲットグループレベルでは、クロスゾーン負荷分散を無効化できます。詳細については、「Application Load Balancer ユーザーガイド」の「クロスゾーン負荷分散をオフにする」を参照してください。

Network Load Balancer と Gateway Load Balancer では、クロスゾーン負荷分散はデフォルトで無効になっています。ロードバランサーの作成後は、いつでもクロスゾーン負荷分散を有効または無効にできます。詳細は「Network Load Balancers ユーザーガイド」の「Cross-zone load balancing」を参照してください。

Classic Load Balancer の作成時における、クロスゾーン負荷分散のデフォルト状態は、ロードバランサーの作成方法により異なります。API または CLI を使用する場合、クロスゾーン負荷分散はデフォルトで無効化されます。では AWS マネジメントコンソール、クロスゾーン負荷分散を有効にするオプションがデフォルトで選択されています。クロスゾーン負荷分散は、Classic Load Balancer の作成後に、いつでも有効化または無効化できます。詳細については、Classic Load Balancer ユーザーガイドの Enable cross-zone load balancing を参照してください。

ゾーンシフト

ゾーンシフトは Amazon Application Recovery Controller (ARC) の機能です。ゾーンシフトを使用すると、1 回のアクションでロードバランサーのリソースを障害のあるアベイラビリティーゾーンから

移動できます。このようにして、 AWS リージョンの他の正常なアベイラビリティーゾーンから操作を継続できます。

ゾーンシフトを開始すると、ロードバランサーは、影響を受けるアベイラビリティーゾーンへのリソースのトラフィックの送信を停止します。ARC はこのゾーンシフトをすぐに作成します。ただし、影響を受けるアベイラビリティーゾーンで進行中の既存の接続が完了するまでには、通常は数分程度の短い時間がかかる場合があります。詳細は「Amazon Application Recovery Controller (ARC) デベロッパーガイド」の「How a zonal shift works: health checks and zonal IP addresses」を参照してください。

ゾーンシフトを使用する前に、以下を確認してください。

- ゾーンシフトは、クロスゾーン負荷分散を無効にした Network Load Balancer を使用している場合はサポートされます。
- 1 つのアベイラビリティーゾーンに対してのみ、特定のロードバランサーのゾーンシフトを開始できます。複数のアベイラビリティーゾーンに対してゾーンシフトを開始することはできません。
- AWS は、複数のインフラストラクチャの問題がサービスに影響を与える場合、DNS からゾーンロードバランサーの IP アドレスをプロアクティブに削除します。ゾーンシフトを開始する前に、現在のアベイラビリティーゾーンの容量を必ず確認してください。ロードバランサーのクロスゾーンロードバランシングがオフになっていて、ゾーンシフトを使用してゾーンロードバランサーの IP アドレスを削除すると、ゾーンシフトの影響を受けるアベイラビリティーゾーンもターゲット容量を失います。

詳細は「Amazon Application Recovery Controller (ARC) デベロッパーガイド」の「ARC のゾーンシフトのベストプラクティス」を参照してください。

リクエストルーティング

クライアントがリクエストをロードバランサーに送信する前に、ドメインネームシステム (DNS) サーバーを使用してロードバランサーのドメイン名を解決します。ロードバランサーは `amazonaws.com` ドメインにあるため、DNS エントリは Amazon によって制御されます。Amazon DNS サーバーがクライアントに 1 つ以上の IP アドレスを返します。これらは、ロードバランサーのロードバランサーノードの IP アドレスです。Network Load Balancer を使用すると、Elastic Load Balancing は、有効にする各アベイラビリティーゾーンのネットワークインターフェイスを作成し、それを使用して静的 IP アドレスを取得します。Network Load Balancer の作成時に、必要に応じて 1 つの Elastic IP アドレスを各ネットワークインターフェイスに関連付けることができます。

アプリケーションへのトラフィックが時間の経過とともに変化すると、Elastic Load Balancing はロードバランサーをスケーリングして DNS エントリを更新します。DNS エントリは、60 秒の有効期限 (TTL) も指定します。これにより、トラフィックの変化に応じて IP アドレスを迅速に再マッピングできます。

クライアントは、ロードバランサーにリクエストを送信するために使用する IP アドレスを決定します。リクエストを受信したロードバランサーノードは、正常な登録済みターゲットを選択し、そのプライベート IP アドレスを使用してターゲットにリクエストを送信します。

詳細については、Amazon Route 53 デベロッパーガイドの ELB ロードバランサーへのトラフィックのルーティングを参照してください。

ルーティングアルゴリズム

Application Load Balancer では、リクエストを受信するロードバランサーノードは、次のプロセスを使用します。

1. リスナールールを優先度順に評価して、適用するルールを決定します。
2. ターゲットグループに設定されたルーティングアルゴリズムを使用して、ルールアクションのターゲットグループからターゲットを選択します。デフォルトのルーティングアルゴリズムはラウンドロビンです。それぞれのターゲットグループでルーティングは個別に実行され、複数のターゲットグループに登録されているターゲットの場合も同じです。

Network Load Balancer では、接続を受信するロードバランサーノードは、次のプロセスを使用します。

1. フローハッシュアルゴリズムを使用して、デフォルトルールのターゲットグループからターゲットを選択します。アルゴリズムは以下に基づきます。
   - プロトコル
   - 送信元 IP アドレスと送信元ポート
   - 送信先 IP アドレスと送信先ポート
   - TCP シーケンス番号
2. 接続中、各 TCP 接続を単一のターゲットにルーティングします。クライアントからの TCP 接続のソースポートとシーケンス番号は異なり、別のターゲットにルーティングできます。

Gateway Load Balancer では、接続を受信するロードバランサーノードは 5 タプルフローハッシュアルゴリズムを使用してターゲットアプライアンスを選択します。フローが確立されると、同じフ

ローのすべてのパケットが同じターゲットアプライアンスに一貫してルーティングされます。ロードバランサーとターゲットアプライアンスは、ポート 6081 の GENEVE プロトコルを使用してトラフィックを交換します。

Classic Load Balancer では、リクエストを受信するロードバランサーノードは、次のように登録済みインスタンスを選択します。

- TCP リスナーのラウンドロビンルーティングアルゴリズムを使用する
- HTTP リスナーと HTTPS リスナーの最小未処理リクエストルーティングアルゴリズムを使用する

HTTP 接続

Classic Load Balancer は事前に開かれた接続を使用しますが、Application Load Balancer は使用しません。Classic Load Balancer と Application Load Balancer はどちらも接続の多重化を使用します。つまり、複数のフロントエンド接続の複数のクライアントからのリクエストは、1 つのバックエンド接続を介して指定のターゲットにルーティングできます。接続の多重化により、レイテンシーが改善され、アプリケーションの負荷が低下します。接続の多重化を回避するには、HTTP レスポンスの `Connection: close` ヘッダーを設定して、HTTP `keep-alive` ヘッダーを無効にします。

Application Load Balancer と Classic Load Balancer は、フロントエンド接続でパイプライン化された HTTP をサポートします。バックエンド接続ではパイプライン化された HTTP をサポートしていません。

Application Load Balancer は、GET、HEAD、POST、PUT、DELETE、OPTIONS、PATCH の HTTP リクエストメソッドをサポートしています。

Application Load Balancer はフロントエンド接続の次のプロトコールをサポートします: HTTP/0.9、HTTP/1.0、HTTP/1.1、HTTP/2。HTTPS/2 は HTTPS リスナーにのみ使用でき、HTTP/2 接続を使用して最大 128 のリクエストを並行して送信できます。Application Load Balancer は、HTTP から WebSockets への接続アップグレードもサポートします。ただし、接続のアップグレードがある場合、Application Load Balancer リスナーのルーティングルールと AWS WAF 統合は適用されません。

デフォルトでは、Application Load Balancer はバックエンド接続 (登録されたターゲットへのロードバランサー) で HTTP/1.1 を使用します。ただし、プロトコルバージョンを使用して、HTTP/2 または gRPC を使用するターゲットに要求を送信することができます。詳細については、「プロトコルバージョン」を参照してください。`keep-alive` ヘッダーは、デフォルトでは、バックエンド接続でサポートされています。ホストヘッダーを満たさないクライアントからの HTTP/1.0 リクエストの

場合、ロードバランサーによりバックエンド接続で送信されたリクエストに対してHTTP/1.1ホストヘッダーを生成します。ホストヘッダーにはロードバランサーの DNS 名が含まれます。

Classic Load Balancer はフロントエンド接続 (ロードバランサーのクライアント) の次のプロトコールをサポートします: HTTP/0.9、HTTP/1.0、HTTP/1.1。デフォルトでは、 バックエンド接続 (登録されたターゲットへのロードバランサー) で HTTP/1.1 を使用します。`keep-alive` ヘッダーは、デフォルトでは、バックエンド接続でサポートされています。ホストヘッダーを満たさないクライアントからの HTTP/1.0 リクエストの場合、ロードバランサーによりバックエンド接続で送信されたリクエストに対してHTTP/1.1ホストヘッダーを生成します。ホストヘッダーにはロードバランサーノードの IP アドレスが含まれています。

HTTP ヘッダー

Application Load Balancer と Classic Load Balancer は、X-Forwarded-For、X-Forwarded-Proto、および X-Forwarded-Port ヘッダーを自動的にリクエストに追加します。

アプリケーションロードバランサは、HTTP ホストヘッダーのホスト名を小文字に変換してから、ターゲットに送信します。

HTTP/2 を使用するフロントエンド接続の場合は、ヘッダー名は小文字です。リクエストが HTTP/1.1 を使用してターゲットに送信される前に、以下のヘッダー名は、大小混合文字に変換されます: X-Forwarded-For、X-Forwarded-Proto、X-Forwarded-Port、Host、X-Amzn-Trace-Id、Upgrade、および Connection。そのほかのヘッダー名はすべて小文字です。

Application Load Balancer および Classic Load Balancer は、応答をクライアントにプロキシした後、着信クライアント要求からの接続ヘッダーを尊重します。

HTTP/1.1 を使用している Application Load Balancer と Classic Load Balancer は、Expect: 100-Continue ヘッダーを受け取ると、コンテンツ長ヘッダーをテストすることなく、直ちに HTTP/1.1 100 Continue で応答します。Expect: 100-Continue リクエストヘッダーはターゲットに転送されません。

HTTP/2 を使用する場合、Application Load Balancer はクライアントリクエストの Expect: 100-Continue ヘッダーをサポートしません。Application Load Balancer は HTTP/2 100 Continue で応答したり、このヘッダーをターゲットに転送したりしません。

HTTP ヘッダーの制限

Application Load Balancer の以下のサイズ制限は、変更できないハードリミットです。

- リクエストライン: 16 K

- 単一ヘッダー: 16 K
- レスポンスのヘッダー全体: 32 K
- リクエストのヘッダー全体: 64 K

ロードバランサーのスキーム

ロードバランサーを作成するとき、ロードバランサーを内部向けにするかインターネット向けにするか選択する必要があります。

インターネット向けロードバランサーのノードにはパブリック IP アドレスが必要です。インターネット向けロードバランサーの DNS 名は、ノードのパブリック IP アドレスにパブリックに解決可能です。したがって、インターネット向けロードバランサーは、クライアントからインターネット経由でリクエストをルーティングできます。

内部ロードバランサーのノードはプライベート IP アドレスのみを持ちます。内部ロードバランサーの DNS 名は、ノードのプライベート IP アドレスにパブリックに解決可能です。そのため、内部向けロードバランサーは、ロードバランサー用に VPC へのアクセス権を持つクライアントからのみ、リクエストをルーティングできます。

インターネット向けロードバランサーと内部向けロードバランサーは、どちらもプライベート IP アドレスを使用してリクエストをターゲットにルーティングします。したがって、ターゲットは、内部またはインターネット向けロードバランサーからリクエストを受信するためのパブリック IP アドレスを必要としません。

アプリケーションに複数の層がある場合は、内部向けロードバランサーとインターネット向けロードバランサーを併用するアーキテクチャを設計できます。これはたとえば、アプリケーションで、インターネットに接続する必要があるウェブサーバーと、ウェブサーバーにのみ接続するアプリケーションサーバーを使用する場合に該当します。インターネット接続ロードバランサーを作成し、そこにウェブサーバーを登録します。内部ロードバランサーを作成し、そこにアプリケーションサーバーを登録します。ウェブサーバーは、インターネット接続ロードバランサーからリクエストを受け取り、アプリケーションサーバー用のリクエストを内部ロードバランサーに送信します。アプリケーションサーバーは、内部ロードバランサーからリクエストを受け取ります。

IP アドレスのタイプ

ユーザーがロードバランサーに指定した IP アドレスのタイプによって、クライアントがロードバランサーと通信する方法が決まります。

- IPv4 のみ – クライアントはパブリックおよびプライベートの IPv4 アドレスを使用して通信します。ロードバランサー用に選択するサブネットには IPv4 アドレス範囲が必要です。
- デュアルスタック – クライアントはパブリックおよびプライベートの IPv4 および IPv6 アドレスを使用して通信します。ロードバランサー用に選択するサブネットには IPv4 および IPv6 のアドレス範囲が必要です。
- パブリック IPv4 を使用しないデュアルスタック – クライアントはパブリックおよびプライベートの IPv6 アドレスと、プライベートの IPv4 アドレスを使用して通信します。ロードバランサー用に選択するサブネットには IPv4 および IPv6 のアドレス範囲が必要です。この方法は、`internal` ロードバランサースキームではサポートされていません。

次の表は、各ロードバランサータイプでサポートされている IP アドレスタイプをまとめたものです。

ロードバランサーのタイプ	IPv4 のみ	デュアルスタック	パブリック IPv4 を使用しないデュアルスタック
Application Load Balancer	はい	あり	はい
Network Load Balancer	はい	はい	いいえ
Gateway Load Balancer	はい	はい	いいえ
Classic Load Balancer	はい	いいえ	いいえ

ユーザーがターゲットグループに指定した IP アドレスタイプによって、ロードバランサーがターゲットと通信する方法が決まります。

- IPv4 のみ – ロードバランサーはプライベートの IPv4 アドレスを使用して通信します。IPv4 アドレスを指定したターゲットは IPv4 ターゲットグループに登録する必要があります。

- IPv6 のみ – ロードバランサーは IPv6 アドレスを使用して通信します。IPv6 アドレスを指定したターゲットは IPv6 ターゲットグループに登録する必要があります。ターゲットグループは、デュアルスタックのロードバランサーで使用する必要があります。

次の表は、各ターゲットグループプロトコルでサポートされている IP アドレスタイプを示しています。

ターゲットグループプロトコル	IPv4 のみ	IPv6 のみ	
HTTP および HTTPS	はい	はい	
TCP	はい	はい	
TLS	はい	はい	
UDP と TCP_UDP	はい	はい	
GENEVE	-	-	

ロードバランサーのネットワーク MTU

最大送信単位 (MTU) は、ネットワーク上で送信できる最大のパケットサイズ (バイト単位) を決定します。接続の MTU が大きいほど、より多くのデータを単一のパケットで渡すことができます。イーサネットフレームはパケット (送信している実際のデータ) とそれを囲むネットワークオーバーヘッド情報で構成されています。インターネットゲートウェイを介して送信されるトラフィックの MTU は 1500 です。つまり、パケットが 1500 バイトを超えている場合は、断片化されて複数のフレームを使用して送信されるか、`Don't Fragment` が IP ヘッダーに設定されていればドロップされます。

ロードバランサーノードの MTU サイズは設定できません。Application Load Balancer、Network Load Balancer、Classic Load Balancer のロードバランサーノード全体で、ジャンボフレーム (9001 MTU) は、標準になっています。Gateway Load Balancer は 8500 MTU をサポートします。詳細につ

いては、「Gateway Load Balancer のユーザーガイド」の「最大送信単位 (MTU)」を参照してください。

パス MTU は、送信側ホストと受信側ホスト間のパスでサポートされている最大のパケットサイズです。2 つのデバイス間のパス MTU を判断するために、パス MTU 検出 (PMTUD) が使用されます。パス MTU 検出は、クライアントまたはターゲットがジャンボフレームをサポートしていない場合に特に重要です。

ホストがパスに沿って送信するパケットが、受信側ホストの MTU、あるいはデバイスの MTU よりも大きな場合、受信側ホストまたはデバイスはそのパケットを削除し、次のような ICMP メッセージ `Destination Unreachable: Fragmentation Needed and Don't Fragment was Set (Type 3, Code 4)` を返します。このメッセージは送信側ホストに対し、ペイロードを複数の小さなパケットに分割し再送信することを指示します。

クライアントまたはターゲットインターフェイスの MTU サイズより大きいパケットが引き続き削除される場合、Path MTU 検出 (PMTUD) が機能していない可能性があります。これを回避するには、Path MTU 検出がエンドツーエンドで動作していること、およびクライアントおよびターゲットでジャンボフレームを有効にしていることを確認します。パス MTU 検出とジャンボフレームの有効化の詳細については、Amazon EC2 ユーザーガイド の [パス MTU 検出]を参照してください。

Elastic Load Balancing の使用開始

Elastic Load Balancing は、複数のロードバランサータイプをサポートしています。ニーズに最適なタイプのロードバランサーを選択できます。詳細については、「Elastic Load Balancing の機能」の「」を参照してください。

ロードバランサー

- Application Load Balancer の作成
- Network Load Balancer を作成する
- ゲートウェイロードバランサーを作成

一般的なロードバランサー設定のデモについては、Elastic Load Balancing のデモを参照してください。

既存の Classic Load Balancer がある場合は、Application Load Balancer または Network Load Balancer に移行できます。詳細については、「Classic Load Balancer の移行」を参照してください。

Elastic Load Balancing のセキュリティ

のクラウドセキュリティが最優先事項 AWS です。お客様は AWS 、セキュリティを最も重視する組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャからメリットを得られます。

セキュリティは、 AWS お客様とお客様の間の責任共有です。責任共有モデルではこれをクラウドのセキュリティおよびクラウド内のセキュリティと説明しています。

- クラウドのセキュリティ – AWS は、 で AWS サービスを実行するインフラストラクチャを保護する責任を担います AWS クラウド。 は、お客様が安全に使用できるサービス AWS も提供します。サードパーティーの監査者は、AWS コンプライアンスプログラムコンプライアンスプログラムの一環として、当社のセキュリティの有効性を定期的にテストおよび検証。Elastic Load Balancing に適用されるコンプライアンスプログラムの詳細については、「コンプライアンスプログラムAWS による対象範囲内のサービスコンプライアンスプログラム」を参照してください。
- クラウド内のセキュリティ – お客様の責任は、使用する AWS サービスによって決まります。また、お客様は、お客様のデータの機密性、企業の要件、および適用可能な法律および規制などの他の要因についても責任を担います。

このドキュメントは、Elastic Load Balancing を使用する際の責任共有モデルの適用方法を理解するのに役立ちます。ここでは、セキュリティとコンプライアンスの目標を満たすように Elastic Load Balancing を設定する方法について説明します。また、Elastic Load Balancing リソースのモニタリングや保護に役立つ他の AWS サービスの使用方法についても説明します。

ゲートウェイロードバランサー では、アプライアンスベンダーのソフトウェアを選択して適格化する必要があります。ロードバランサーからのトラフィックを検査または変更するには、アプライアンスソフトウェアを信頼する必要があります。ロードバランサーは、OSI (Open Systems Interconnection) モデルのレイヤー 3 (ネットワークレイヤー) で動作します。Elastic Load Balancing パートナーとしてリストされているアプライアンスベンダーは、アプライアンスソフトウェアを と統合し、認定しています AWS。このリストのベンダーのアプライアンスソフトウェアには、より高いレベルの信頼を置くことができます。ただし、 AWS は、これらのベンダーのソフトウェアのセキュリティや信頼性を保証しません。

内容

- Elastic Load Balancing のコンプライアンス検証
- Elastic Load Balancing の復元性
- Elastic Load Balancing のインフラストラクチャセキュリティ
- インターフェイスエンドポイントを使用して Elastic Load Balancing にアクセスする (AWS PrivateLink)

Elastic Load Balancing のデータ保護

責任 AWS 共有モデル、Elastic Load Balancing でのデータ保護に適用されます。このモデルで説明されているように、 AWS はすべての を実行するグローバルインフラストラクチャを保護する責任があります AWS クラウド。ユーザーは、このインフラストラクチャでホストされるコンテンツに対する管理を維持する責任があります。また、使用する「 AWS のサービス 」のセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、データプライバシーに関するよくある質問を参照してください。欧州でのデータ保護の詳細については、AWS セキュリティブログに投稿された「AWS 責任共有モデルおよび GDPR」のブログ記事を参照してください。

データ保護の目的で、認証情報を保護し AWS アカウント 、 AWS IAM アイデンティティセンターまたは AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定することをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 は必須ですが、TLS 1.3 を推奨します。
- で API とユーザーアクティビティの口グ記録を設定します AWS CloudTrail。CloudTrail 証跡を使用して AWS アクティビティをキャプチャする方法については、「 AWS CloudTrail ユーザーガイド」のCloudTrail 証跡の使用」を参照してください。
- AWS 暗号化ソリューションと、その中のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。
- Amazon Macie などの高度な管理されたセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API AWS を介して にアクセスするときに FIPS 140-3 検証済み暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-3」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報を、タグ、または [名前] フィールドなどの自由形式のテキストフィールドに含めないことを強くお勧めします。これは、コンソール、API、または SDK を使用して Elastic Load Balancing AWS CLIまたは他の AWS のサービス を使用する場合も同様です。 AWS SDKs タグ、または名前に使用される自由記述のテキストフィールドに入力したデータは、請求または診断ログに使用される場合があります。外部サーバーに URL を提供する場合、そのサーバーへのリクエストを検証できるように、認証情報を URL に含めないことを強くお勧めします。

保管中の暗号化

Elastic Load Balancing アクセスログの S3 バケットに対して、Amazon S3 管理の暗号化キーによるサーバー側の暗号化 (SSE-S3) を有効にした場合、Elastic Load Balancing は、S3 バケットに保存される前にアクセスログファイルを自動的に暗号化し、また、Elastic Load Balancing は、アクセスログファイルにアクセスするときに復号化を行います。各ログファイルは一意のキーで暗号化されます。この一意のキー自体が、定期的に更新される KMS キーで更新されます。

転送中の暗号化

Elastic Load Balancing は、ロードバランサーでクライアントからの HTTPS および TLS トラフィックを終了することで、セキュアなウェブアプリケーションの構築プロセスを簡素化します。ロードバランサーは、各 EC2 インスタンスに TLS ターミネーションの処理を要求する代わりに、トラフィックの暗号化と復号化の作業を実行します。セキュアリスナーを設定するときは、アプリケーションでサポートされている暗号スイートとプロトコルバージョン、およびロードバランサーにインストールするサーバー証明書を指定します。 AWS Certificate Manager (ACM) または AWS Identity and Access Management (IAM) を使用してサーバー証明書を管理できます。Application Load Balancer は HTTPS リスナーをサポートします。Network Load Balancer は TLS リスナーをサポートします。Classic Load Balancer は、HTTPS リスナーと TLS リスナーの両方をサポートします。

Elastic Load Balancing の Identity and Access Management

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御 AWS のサービス するのに役立つ です。IAM 管理者は、Elastic Load Balancing リソースの使用をユーザーに許可するために、認証 (サインイン) および承認 (アクセス許可を付与する) の制御を行います。IAM は、追加料金なしで使用できる AWS のサービス です。

内容

- ポリシーを使用したアクセスの管理
- Elastic Load Balancing で利用できる IAM 機能
- リソース作成時にタグ付けするElastic Load Balancing API のアクセス許可
- Elastic Load Balancing のサービスにリンクされたロール
- AWS Elastic Load Balancing の マネージドポリシー

対象者

AWS Identity and Access Management (IAM) の使用方法は、Elastic Load Balancing で行う作業によって異なります。

Service user (サービスユーザー) – Elastic Load Balancing サービスを使用してジョブを実行する場合は、必要な認証情報とアクセス許可を管理者が用意します。さらに多くの Elastic Load Balancing 機能を使用して作業を行う場合は、追加のアクセス許可が必要になることがあります。アクセスの管理方法を理解すると、管理者に適切なアクセス許可をリクエストするのに役に立ちます。

Service administrator (サービス管理者) – 社内の Elastic Load Balancing リソースを担当している場合は、通常、Elastic Load Balancing へのフルアクセスがあります。サービスのユーザーがどの Elastic Load Balancing 機能やリソースにアクセスするかを決めるのは管理者の仕事です。その後、IAM 管理者にリクエストを送信して、サービスユーザーの権限を変更する必要があります。このページの情報を点検して、IAM の基本概念を理解してください。

IAM administrator (IAM 管理者) – IAM 管理者は、Elastic Load Balancing へのアクセスを管理するポリシーの作成方法の詳細について確認する場合があります。

アイデンティティを使用した認証

認証とは、ID 認証情報 AWS を使用して にサインインする方法です。、IAM ユーザー AWS アカウントのルートユーザー、または IAM ロールを引き受けることで認証される必要があります。

AWS IAM アイデンティティセンター (IAM Identity Center)、シングルサインオン認証、Google/Facebook 認証情報などの ID ソースからの認証情報を使用して、フェデレーティッド ID としてサインインできます。サインインの詳細については、「AWS サインイン ユーザーガイド」の「AWS アカウントにサインインする方法」を参照してください。

プログラムによるアクセスの場合、 は SDK と CLI AWS を提供してリクエストを暗号化して署名します。詳細については、「IAM ユーザーガイド」の「API リクエストに対するAWS 署名バージョン 4」を参照してください。

AWS アカウント ルートユーザー

を作成するときは AWS アカウント、すべての AWS のサービス および リソースへの完全なアクセス権を持つ AWS アカウント root ユーザーと呼ばれる 1 つのサインインアイデンティティから始めます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザー認証情報を必要とするタスクについては、「IAM ユーザーガイド」の「ルートユーザー認証情報が必要なタスク」を参照してください。

フェデレーテッドアイデンティティ

ベストプラクティスとして、人間のユーザーが一時的な認証情報 AWS のサービス を使用して にアクセスするには、ID プロバイダーとのフェデレーションを使用する必要があります。

フェデレーティッド ID は、エンタープライズディレクトリ、ウェブ ID プロバイダー、または ID ソースからの認証情報 AWS のサービス を使用して Directory Service にアクセスするユーザーです。フェデレーテッドアイデンティティは、一時的な認証情報を提供するロールを引き受けます。

アクセスを一元管理する場合は、 AWS IAM アイデンティティセンターをお勧めします。詳細については、「AWS IAM アイデンティティセンター ユーザーガイド」の「IAM アイデンティティセンターとは」を参照してください。

IAM ユーザーとグループ

IAM ユーザーは、特定の個人やアプリケーションに対する特定のアクセス許可を持つアイデンティティです。長期認証情報を持つ IAM ユーザーの代わりに一時的な認証情報を使用することをお勧めします。詳細については、IAM ユーザーガイドの「ID プロバイダーとのフェデレーションを使用して にアクセスすることを人間のユーザーに要求する AWS」を参照してください。

IAM グループは、IAM ユーザーの集合を指定し、大量のユーザーに対するアクセス許可の管理を容易にします。詳細については、「IAM ユーザーガイド」の「IAM ユーザーに関するユースケース」を参照してください。

IAM ロール

IAM ロールは、特定のアクセス許可を持つアイデンティであり、一時的な認証情報を提供します。ユーザーから IAM ロール (コンソール) に切り替えるか、 または API オペレーションを呼び出すことで、ロールを引き受けることができます。 AWS CLI AWS 詳細については、「IAM ユーザーガイド」の「ロールを引き受けるための各種方法」を参照してください。

IAM ロールは、フェデレーションユーザーアクセス、一時的な IAM ユーザーのアクセス許可、クロスアカウントアクセス、クロスサービスアクセス、および Amazon EC2 で実行するアプリケーショ

ンに役立ちます。詳細については、IAM ユーザーガイド の IAM でのクロスアカウントリソースアクセス を参照してください。

ポリシーを使用したアクセスの管理

でアクセスを制御する AWS には、ポリシーを作成し、ID AWS またはリソースにアタッチします。ポリシーは、アイデンティティまたはリソースに関連付けられたときにアクセス許可を定義します。は、プリンシパルがリクエストを行うときにこれらのポリシー AWS を評価します。ほとんどのポリシーは JSON ドキュメント AWS として に保存されます。JSON ポリシードキュメントの詳細については、「IAM ユーザーガイド」の「JSON ポリシー概要」を参照してください。

管理者は、ポリシーを使用して、どのプリンシパルがどのリソースに対して、どのような条件でアクションを実行できるかを定義することで、誰が何にアクセスできるかを指定します。

デフォルトでは、ユーザーやロールにアクセス許可はありません。IAM 管理者は IAM ポリシーを作成してロールに追加し、このロールをユーザーが引き受けられるようにします。IAM ポリシーは、オペレーションの実行方法を問わず、アクセス許可を定義します。

アイデンティティベースのポリシー

アイデンティティベースのポリシーは、アイデンティティ (ユーザー、グループ、またはロール) にアタッチできる JSON アクセス許可ポリシードキュメントです。これらのポリシーは、アイデンティティがどのリソースに対してどのような条件下でどのようなアクションを実行できるかを制御します。アイデンティティベースポリシーの作成方法については、IAM ユーザーガイド の カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する を参照してください。

アイデンティティベースのポリシーは、インラインポリシー (単一の ID に直接埋め込む) または管理ポリシー (複数の ID にアタッチされたスタンドアロンポリシー) にすることができます。管理ポリシーとインラインポリシーのいずれかを選択する方法については、「IAM ユーザーガイド」の「管理ポリシーとインラインポリシーのいずれかを選択する」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。例としては、IAM ロール信頼ポリシーや Amazon S3 バケットポリシーなどがあります。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーでは、IAM の AWS マネージドポリシーを使用できません。

その他のポリシータイプ

AWS は、より一般的なポリシータイプによって付与されるアクセス許可の最大数を設定できる追加のポリシータイプをサポートしています。

- アクセス許可の境界 – アイデンティティベースのポリシーで IAM エンティティに付与することのできるアクセス許可の数の上限を設定します。詳細については、「IAM ユーザーガイド」の「IAM エンティティのアクセス許可境界」を参照してください。
- サービスコントロールポリシー (SCP) - AWS Organizations内の組織または組織単位の最大のアクセス許可を指定します。詳細については、「AWS Organizations ユーザーガイド」の「サービスコントロールポリシー」を参照してください。
- リソースコントロールポリシー (RCP) – は、アカウント内のリソースで利用できる最大数のアクセス許可を定義します。詳細については、「AWS Organizations ユーザーガイド」の「リソースコントロールポリシー (RCP)」を参照してください。
- セッションポリシー – ロールまたはフェデレーションユーザーの一時セッションを作成する際にパラメータとして渡される高度なポリシーです。詳細については、「IAM ユーザーガイド」の「セッションポリシー」を参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成されるアクセス許可を理解するのがさらに難しくなります。が複数のポリシータイプが関与する場合にリクエストを許可するかどうか AWS を決定する方法については、「IAM ユーザーガイド」の「ポリシー評価ロジック」を参照してください。

Elastic Load Balancing で利用できる IAM 機能

IAM を使用して Elastic Load Balancing へのアクセスを管理する前に、Elastic Load Balancing で使用できる IAM 機能について学びます。

Elastic Load Balancing で使用できる IAM 機能

IAM 機能	Elastic Load Balancing サポート
アイデンティティベースのポリシー	あり
リソースベースのポリシー	なし

IAM 機能	Elastic Load Balancing サポート
ポリシーアクション	あり
ポリシーリソース	はい
ポリシー条件キー (サービス固有)	はい
ACL	なし
ABAC (ポリシー内のタグ)	あり
一時的な認証情報	あり
プリンシパルアクセス権限	あり
サービスロール	いいえ
サービスリンクロール	はい

Elastic Load Balancing のアイデンティティベースのポリシー

アイデンティティベースのポリシーのサポート: あり

アイデンティティベースポリシーは、IAM ユーザー、ユーザーグループ、ロールなど、アイデンティティにアタッチできる JSON 許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティベースポリシーの作成方法については、「IAM ユーザーガイド」の「カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する」を参照してください。

IAM アイデンティティベースのポリシーでは、許可または拒否するアクションとリソース、およびアクションを許可または拒否する条件を指定できます。JSON ポリシーで使用できるすべての要素について学ぶには、「IAM ユーザーガイド」の「IAM JSON ポリシーの要素のリファレンス」を参照してください。

Elastic Load Balancing 内のリソースベースのポリシー

リソースベースのポリシーのサポート: なし

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあげ

られます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスをコントロールできます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーで、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

クロスアカウントアクセスを有効にするには、全体のアカウント、または別のアカウントの IAM エンティティを、リソースベースのポリシーのプリンシパルとして指定します。詳細については、IAM ユーザーガイドのIAM でのクロスアカウントリソースアクセスを参照してください。

Elastic Load Balancing のポリシーアクション

ポリシーアクションのサポート: あり

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

JSON ポリシーの `Action` 要素にはポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。このアクションは関連付けられたオペレーションを実行するためのアクセス許可を付与するポリシーで使用されます。

Elastic Load Balancing アクションのリストを表示するには、「サービス認可リファレンス」の「Elastic Load Balancing V2 で定義されるアクション」および「Elastic Load Balancing V1 で定義されるアクション」を参照してください。

Elastic Load Balancing のポリシーアクションは、アクションの前に以下のプレフィックス を使用します。

```
elasticloadbalancing
```

単一のステートメントで複数のアクションを指定するには、アクションをカンマで区切ります。

```
"Action": [
    "elasticloadbalancing:action1",
    "elasticloadbalancing:action2"
]
```

ワイルドカード (*) を使用して複数アクションを指定できます。例えば、`Describe` という単語で始まるすべてのアクションを指定するには次のアクションを含めます。

```
"Action": "elasticloadbalancing:Describe*"
```

Elastic Load Balancing の API アクションの詳細なリストについては、次のドキュメントを参照してください。

- Application Load Balancer、Network Load Balancer、および　Gateway Load Balancer — API リファレンスバージョン 2015-12-01
- Classic Load Balancer — API リファレンスバージョン 2012-06-01

Elastic Load Balancing のポリシーリソース

ポリシーリソースのサポート: あり

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

`Resource` JSON ポリシー要素はアクションが適用されるオブジェクトを指定します。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。リソースレベルのアクセス許可をサポートしないアクションの場合は、ステートメントがすべてのリソースに適用されることを示すために、ワイルドカード (*) を使用します。

```
"Resource": "*"
```

複数のリソースをサポートする Elastic Load Balancing API アクションがあります。複数リソースを単一ステートメントで指定するには、ARN をカンマで区切ります。

```
"Resource": [
    "resource1",
    "resource2"
]
```

Elastic Load Balancing リソースタイプとその ARN のリストを確認するには、「サービス認可リファレンス」の「Elastic Load Balancing V2 で定義されるリソース」および「Elastic Load Balancing V1 で定義されるリソース」を参照してください。　各リソースの ARN を指定できるアクションについては、「Elastic Load Balancing V2 で定義されるアクション」および「Elastic Load Balancing V1 で定義されるアクション」を参照してください。

Elastic Load Balancing のポリシー条件キー

サービス固有のポリシー条件キーのサポート: あり

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

`Condition` 要素は、定義された基準に基づいてステートメントが実行される時期を指定します。イコールや未満などの条件演算子を使用して条件式を作成して、ポリシーの条件とリクエスト内の値を一致させることができます。すべての AWS グローバル条件キーを確認するには、「IAM ユーザーガイド」のAWS 「グローバル条件コンテキストキー」を参照してください。

Elastic Load Balancing の条件キーのリストについては、「サービス認可リファレンス」の「Elastic Load Balancing V2 の条件キー」および「Elastic Load Balancing V1 の条件キー」を参照してください。どのアクションおよびリソースと条件キーを使用できるかについては、「Elastic Load Balancing V2 で定義されるアクション」および「Elastic Load Balancing V1 で定義されるアクション」を参照してください。

条件キー

- elasticloadbalancing:ListenerProtocol 条件キー
- elasticloadbalancing:SecurityPolicy 条件キー
- elasticloadbalancing:Scheme 条件キー
- elasticloadbalancing:SecurityGroup 条件キー
- elasticloadbalancing:Subnet 条件キー
- elasticloadbalancing:ResourceTag 条件キー

elasticloadbalancing:ListenerProtocol 条件キー

`elasticloadbalancing:ListenerProtocol` 条件キーは、作成と使用が可能なリスナーのタイプを定義する条件として使用できます。このポリシーは、Application Load Balancer、Network Load Balancer、Classic Load Balancer で使用できます。以下のアクションでこの条件キーがサポートされています。

API バージョン 2015-12-01

- `CreateListener`
- `ModifyListener`

API バージョン 2012-06-01

- `CreateLoadBalancer`
- `CreateLoadBalancerListeners`

次のポリシー例では、ユーザーは Application Load Balancer のリスナーの HTTPS プロトコルと、Network Load Balancer のリスナーの TLS プロトコルを選択する必要があります。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "elasticloadbalancing:CreateListener",
            "elasticloadbalancing:ModifyListener"
        ],
        "Resource": "*",
        "Condition": {
            "ForAnyValue:StringEquals": {
                "elasticloadbalancing:ListenerProtocol": [
                    "HTTPS",
                    "TLS"
                ]
            }
        }
    }
}
```

Classic Load Balancer を使用すると、1 回の呼び出しで複数のリスナーを指定できます。したがって、次の例に示すように、ポリシーは複数値のコンテキストキーを使用する必要があります。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:CreateLoadBalancer",
                "elasticloadbalancing:CreateLoadBalancerListeners"
            ],
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "elasticloadbalancing:ListenerProtocol": [
                        "TCP",
                        "HTTP",
                        "HTTPS"
                    ]
                }
            }
        }
    ]
}
```

elasticloadbalancing:SecurityPolicy 条件キー

elasticloadbalancing:SecurityPolicy 条件キーは、ロードバランサーで特定のセキュリティポリシーを定義して適用する条件として使用できます。このポリシーは、Application Load Balancer、Network Load Balancer、Classic Load Balancer で使用できます。以下のアクションでこの条件キーがサポートされています。

API バージョン 2015-12-01

- CreateListener
- ModifyListener

API バージョン 2012-06-01

- CreateLoadBalancerPolicy
- SetLoadBalancerPoliciesOfListener

次のポリシー例では、ユーザーは Application Load Balancer と Network Load Balancer のに対して、指定されたセキュリティポリシーの 1 つを選択する必要があります。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "elasticloadbalancing:CreateListener",
            "elasticloadbalancing:ModifyListener"
        ],
        "Resource": "*",
        "Condition": {
            "ForAnyValue:StringEquals": {
                "elasticloadbalancing:SecurityPolicy": [
                    "ELBSecurityPolicy-TLS13-1-2-2021-06",
                    "ELBSecurityPolicy-TLS13-1-2-Res-2021-06",
                    "ELBSEcurityPolicy-TLS13-1-1-2021-06"
                ]
            }
        }
    }
}
```

elasticloadbalancing:Scheme 条件キー

`elasticloadbalancing:Scheme` 条件キーは、ロードバランサーの作成時に選択できるスキームを、定義する条件として使用できます。このポリシーは、Application Load Balancer、Network Load Balancer、Classic Load Balancer で使用できます。以下のアクションでこの条件キーがサポートされています。

API バージョン 2015-12-01

- `CreateLoadBalancer`

API バージョン 2012-06-01

- `CreateLoadBalancer`

以下のポリシーの例では、ユーザーは、指定されたスキームをロードバランサー用に選択する必要があります。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "elasticloadbalancing:CreateLoadBalancer",
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "elasticloadbalancing:Scheme": "internal"
            }
        }
    }
}
```

`elasticloadbalancing:SecurityGroup` 条件キー

> **Important**
>
> Elastic Load Balancing はすべて大文字のセキュリティグループ ID を受け入れます。ただし、`StringEqualsIgnoreCase` のように大文字と小文字を区別しない適切な条件演算子を使用してください。

`elasticloadbalancing:SecurityGroup` 条件キーは、ロードバランサーに適用できるセキュリティグループを、定義する条件として使用できます。このポリシーは、Application Load Balancer、Network Load Balancer、Classic Load Balancer で使用できます。以下のアクションでこの条件キーがサポートされています。

API バージョン 2015-12-01

- `CreateLoadBalancer`
- `SetSecurityGroups`

API バージョン 2012-06-01

- `CreateLoadBalancer`
- `ApplySecurityGroupsToLoadBalancer`

以下のポリシーの例では、ユーザーは、指定されたセキュリティグループの 1 つをロードバランサー用に選択する必要があります。

```
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "elasticloadbalancing:CreateLoadBalancer",
            "elasticloadbalancing:SetSecurityGroup"
        ],
        "Resource": "*",
        "Condition": {
            "ForAnyValue:StringEqualsIgnoreCase":{
                "elasticloadbalancing:SecurityGroup": [
                    "sg-51530134",
                    "sg-51530144",
                    "sg-51530139"
                ]
            },
        }
    }
}
```

elasticloadbalancing:Subnet 条件キー

> **Important**
>
> Elastic Load Balancing はすべて大文字のサブネット ID を受け入れます。ただし、`StringEqualsIgnoreCase` のように大文字と小文字を区別しない適切な条件演算子を使用してください。

`elasticloadbalancing:Subnet` 条件キーは、作成した後、ロードバランサーにアタッチできるサブネットを、定義する条件として使用できます。このポリシーは、Application Load

Balancer、Network Load Balancer、Gateway Load Balancer、Classic Load Balancer で使用できます。以下のアクションでこの条件キーがサポートされています。

API バージョン 2015-12-01

- `CreateLoadBalancer`
- `SetSubnets`

API バージョン 2012-06-01

- `CreateLoadBalancer`
- `AttachLoadBalancerToSubnets`

以下のポリシーの例では、ユーザーは、指定されたサブネットの 1 つをロードバランサー用に選択する必要があります。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "elasticloadbalancing:CreateLoadBalancer",
            "elasticloadbalancing:SetSubnets"
        ],
        "Resource": "*",
        "Condition": {
            "ForAnyValue:StringEqualsIgnoreCase": {
                "elasticloadbalancing:Subnet": [
                    "subnet-01234567890abcdef",
                    "subnet-01234567890abcdeg "
                ]
            }
        }
    }
}
```

elasticloadbalancing:ResourceTag 条件キー

`elasticloadbalancing:ResourceTag/##` 条件キーは Elastic Load Balancing 固有です。すべての変異アクションは、この条件キーをサポートします。

Elastic Load Balancing の ACL

ACL のサポート: なし

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするためのアクセス許可を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Elastic Load Balancing での ABAC

ABAC (ポリシー内のタグ) のサポート: あり

属性ベースのアクセス制御 (ABAC) は、タグと呼ばれる属性に基づいてアクセス許可を定義する認可戦略です。IAM エンティティと AWS リソースにタグをアタッチし、プリンシパルのタグがリソースのタグと一致するときにオペレーションを許可するように ABAC ポリシーを設計できます。

タグに基づいてアクセスを管理するには、`aws:ResourceTag/key-name`、`aws:RequestTag/key-name`、または `aws:TagKeys` の条件キーを使用して、ポリシーの条件要素でタグ情報を提供します。

サービスがすべてのリソースタイプに対して 3 つの条件キーすべてをサポートする場合、そのサービスの値はありです。サービスが一部のリソースタイプに対してのみ 3 つの条件キーのすべてをサポートする場合、値は「部分的」になります。

ABAC の詳細については、「IAM ユーザーガイド」の「ABAC 認可でアクセス許可を定義する」を参照してください。ABAC をセットアップする手順を説明するチュートリアルについては、「IAM ユーザーガイド」の「属性ベースのアクセスコントロール (ABAC) を使用する」を参照してください。

Elastic Load Balancing での一時的な認証情報の使用

一時的な認証情報のサポート: あり

一時的な認証情報は、 AWS リソースへの短期的なアクセスを提供し、フェデレーションまたは切り替えロールを使用する場合に自動的に作成されます。長期的なアクセスキーを使用する代わりに、一時的な認証情報を動的に生成 AWS することをお勧めします。詳細については、「IAM ユーザーガイ

ド」の「IAM の一時的な認証情報」および「AWS のサービス と IAM との連携」を参照してください。

Elastic Load Balancing のクロスサービスプリンシパル許可

転送アクセスセッション (FAS) のサポート: あり

転送アクセスセッション (FAS) は、 を呼び出すプリンシパルのアクセス許可と AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストをリクエストする を使用します。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。

Elastic Load Balancing のサービスロール

サービスロールのサポート: なし

サービスロールとは、サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、IAM ユーザーガイドの AWS のサービスに許可を委任するロールを作成するを参照してください。

Elastic Load Balancing のサービスリンクロール

サービスリンクロールのサポート: あり

サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。IAM 管理者は、サービスリンクロールのアクセス許可を表示できますが、編集することはできません。

Elastic Load Balancing サービスにリンクされたロールの作成または管理の詳細については、「Elastic Load Balancing のサービスにリンクされたロール」を参照してください。

リソース作成時にタグ付けするElastic Load Balancing API のアクセス許可

ユーザーが作成時にタグを付けるには、`elasticloadbalancing:CreateLoadBalancer` または `elasticloadbalancing:CreateTargetGroup` などのリソースを作成するアクションを使用するためのアクセス許可が必要です。リソース作成アクションでタグが指定されている場合、作成されたリソースにタグを適用するための認可をユーザーが持っているかどうか確認するため、`elasticloadbalancing:AddTags` アクションで追加の承認が必要です そのため、ユーザー

にはelasticloadbalancing:AddTags アクションを使用する明示的なアクセス権限が必要です。

elasticloadbalancing:AddTags アクションの IAM ポリシー定義で、Condition 要素をelasticloadbalancing:CreateAction 条件キーと使用して、リソース作成アクションにタグ付けのアクセス許可を付与します。

次の例では、ユーザーがターゲットグループを作成し、作成中に任意のタグを適用できるポリシーを示しています。ユーザーには、既存のリソースへのタグ付けが許可されません (elasticloadbalancing:AddTags アクションを直接呼び出すことはできません)。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
         "elasticloadbalancing:CreateTargetGroup"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
         "elasticloadbalancing:AddTags"
      ],
      "Resource": "*",
      "Condition": {
         "StringEquals": {
             "elasticloadbalancing:CreateAction" : "CreateTargetGroup"
         }
      }
    }
  ]
}
```

同様に、次のポリシーでは、ユーザーはロードバランサーの作成と作成時のタグの適用が可能になります。ユーザーには、既存のリソースへのタグ付けが許可されません (elasticloadbalancing:AddTags アクションを直接呼び出すことはできません)。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
         "elasticloadbalancing:CreateLoadBalancer"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
         "elasticloadbalancing:AddTags"
      ],
      "Resource": "*",
      "Condition": {
         "StringEquals": {
             "elasticloadbalancing:CreateAction" : "CreateLoadBalancer"
         }
      }
    }
  ]
}
```

elasticloadbalancing:AddTags アクションはタグがリソース作成アクション時に適用された場合のみ評価されます。したがって、リクエストでタグが指定されていない場合、リソースを作成するアクセス許可を持っているユーザー (タグ付け条件がないと仮定) にはelasticloadbalancing:AddTags アクションを実行するアクセス許可は必要ありません。ただし、ユーザーがタグを使用してリソースを作成しようとした場合、ユーザーが elasticloadbalancing:AddTags アクションを使用するアクセス許可を持っていない場合はリクエストに失敗します。

Elastic Load Balancing のサービスにリンクされたロール

Elastic Load Balancing は、Elastic Load Balancing がユーザーに代わって他の AWS サービスを呼び出すために必要なアクセス許可を持つサービスにリンクされたロールを使用します。詳細については、「IAM ユーザーガイド」の「Service-linked roles」を参照してください。

サービスにリンクされたロールによって付与されるアクセス許可

Elastic Load Balancing は、AWSServiceRoleForElasticLoadBalancing と名付けられたサービスにリンクされたロールを使用して、お客様に代わってその他の AWS サービスを呼び出します。

AWSServiceRoleForElasticLoadBalancing は、このロールを引き受けるために `elasticloadbalancing.amazonaws.com` サービスを信頼します。

ロールのアクセス許可ポリシーは AWSElasticLoadBalancingServiceRolePolicy です。このポリシーのアクセス許可を確認するには、「AWS マネージドポリシーリファレンス」の「AWSElasticLoadBalancingServiceRolePolicy」を参照してください。

サービスにリンクされたロールの作成

AWSServiceRoleForElasticLoadBalancing ロールを手動で作成する必要はありません。このロールは、ロードバランサーまたはターゲットグループの作成時に Elastic Load Balancing によって作成されます。

Elastic Load Balancing がユーザーに代わってサービスにリンクされたロールを作成するには、必要なアクセス許可がユーザーに付与されていなければなりません。詳細については、IAM ユーザーガイド の「サービスにリンクされたロールのアクセス許可」を参照してください。

サービスにリンクされたロールを編集する

IAM を使用して AWSServiceRoleForElasticLoadBalancing の説明を編集することができます。詳細については、「IAM ユーザーガイド」の「サービスにリンクされたロールの編集」を参照してください。

サービスにリンクされたロールを削除する

Elastic Load Balancing を使用する必要がなくなった場合は、AWSServiceRoleForElasticLoadBalancing を削除することをお勧めします。

このサービスにリンクされたロールは、 AWS アカウント内のすべてのロードバランサーを削除した後にのみ削除できます。これにより、ロードバランサーへのアクセス許可を誤って削除することがな

くなります。詳細については、Application Load Balancer の削除、Network Load Balancer の削除、およびClassic Load Balancer の削除を参照してください。

サービスにリンクされたロールは、IAM コンソール、IAM CLI、または IAM API を使用して削除することができます。詳細については、「IAM ユーザーガイド」の「サービスにリンクされたロールの削除」を参照してください

AWSServiceRoleForElasticLoadBalancing を削除した後、ロードバランサーを作成すると、Elastic Load Balancing によって再度このロールが作成されます。

AWS Elastic Load Balancing の マネージドポリシー

AWS 管理ポリシーは、 によって作成および管理されるスタンドアロンポリシーです AWS。 AWS 管理ポリシーは、ユーザー、グループ、ロールにアクセス許可の割り当てを開始できるように、多くの一般的なユースケースにアクセス許可を提供するように設計されています。

AWS 管理ポリシーは、すべての AWS お客様が使用できるため、特定のユースケースに対して最小特権のアクセス許可を付与しない場合があることに注意してください。ユースケースに固有のカスタマー管理ポリシーを定義して、アクセス許可を絞り込むことをお勧めします。

AWS 管理ポリシーで定義されているアクセス許可は変更できません。が AWS マネージドポリシーで定義されたアクセス許可 AWS を更新すると、ポリシーがアタッチされているすべてのプリンシパル ID (ユーザー、グループ、ロール) に影響します。 AWS は、新しい が起動されるか、新しい API オペレーション AWS のサービス が既存のサービスで使用できるようになったときに、 AWS マネージドポリシーを更新する可能性が高くなります。

詳細については、「IAM ユーザーガイド」の「AWS マネージドポリシー」を参照してください。

AWS マネージドポリシー: AWSElasticLoadBalancingClassicServiceRolePolicy

このポリシーには、Elastic Load Balancing (Classic Load Balancer) がユーザーに代わって他の AWS サービスを呼び出すために必要なすべてのアクセス許可が含まれています。サービスリンクロールは事前定義されています。事前定義されたロールを使用すると、Elastic Load Balancing がユーザーに代わってアクションを完了するために必要とする許可を手動で追加する必要がなくなります。このポリシーをアタッチ、デタッチ、変更、または削除することはできません。

このポリシーのアクセス許可を確認するには、「AWS マネージドポリシーリファレンス」の「AWSElasticLoadBalancingClassicServiceRolePolicy」を参照してください。

AWS 管理ポリシー: AWSElasticLoadBalancingServiceRolePolicy

このポリシーには、Elastic Load Balancing がユーザーに代わって AWS のその他サービスを呼び出すために必要とするすべての許可が含まれています。サービスリンクロールは事前定義されています。事前定義されたロールを使用すると、Elastic Load Balancing がユーザーに代わってアクションを完了するために必要とする許可を手動で追加する必要がなくなります。このポリシーをアタッチ、デタッチ、変更、または削除することはできません。

このポリシーのアクセス許可を確認するには、「AWS マネージドポリシーリファレンス」の「AWSElasticLoadBalancingServiceRolePolicy」を参照してください。

AWS マネージドポリシー: ElasticLoadBalancingFullAccess

このポリシーは、Elastic Load Balancing サービスへのフルアクセスと、 AWS マネジメントコンソールを介した他のサービスへの制限付きアクセスを許可します。

このポリシーのアクセス許可を確認するには、「AWS マネージドポリシーリファレンス」の「ElasticLoadBalancingFullAccess」を参照してください。

AWS マネージドポリシー: ElasticLoadBalancingReadOnly

このポリシーは、Elastic Load Balancing および依存サービスに対する読み取り専用アクセス権を付与します。

このポリシーのアクセス許可を確認するには、「AWS マネージドポリシーリファレンス」の「ElasticLoadBalancingReadOnly」を参照してください。

AWS 管理ポリシーに対する Elastic Load Balancing の更新

このサービスがこれらの変更の追跡を開始してからの Elastic Load Balancing の AWS マネージドポリシーの更新に関する詳細を表示します。

変更	説明	日付
ElasticLoadBalancingFullAccess – 既存ポリシーへの更新	入力の検証中にアベイラビリティーゾーンを記述するアクセス許可を付与する`ec2:DescribeAvailityZones` アクションを追加しました。	2026 年 2 月 23 日

変更	説明	日付
AWSElasticLoadBalancingServiceRolePolicy – 既存ポリシーへの更新	入力の検証中にアベイラビリティーゾーンを記述するアクセス許可を付与するec2:DescribeAvailityZones アクションを追加しました。	2025 年 11 月 21 日
AWSElasticLoadBalancingServiceRolePolicy – 既存ポリシーへの更新	IPAM プールから CIDR ブロックを割り当てるアクセス許可を付与する ec2:AllocateIpamPoolCidr アクションを追加しました。	2025 年 2 月 17 日
ElasticLoadBalancingFullAccess – 既存ポリシーへの更新	ゾーンシフトに必要なアクセス許可を付与する arc-zonal-shift:* アクションを追加しました。	2023 年 11 月 28 日
ElasticLoadBalancingReadOnly – 既存ポリシーへの更新	ゾーンシフトに必要なアクセス許可を付与するアクション arc-zonal-shift:GetManagedResource 、arc-zonal-shift:ListManagedResources 、および arc-zonal-shift:ListZonalShifts を追加しました。	2023 年 11 月 28 日
AWSElasticLoadBalancingServiceRolePolicy – 既存ポリシーへの更新	ピア接続に必要なアクセス許可を付与する ec2:DescribeVpcPeeringConnections アクションを追加しました。	2021 年 10 月 11 日
ElasticLoadBalancingFullAccess – 既存ポリシーへの更新	ピア接続に必要なアクセス許可を付与する ec2:DescribeVpcPeeringConnections アクションを追加しました。	2021 年 10 月 11 日
ElasticLoadBalancingFullAccess – 新しいポリシー	Elastic Load Balancing および依存サービスへのフルアクセス権を付与。	2018 年 9 月 20 日
ElasticLoadBalancingReadOnly – 新しいポリシー	Elastic Load Balancing および依存サービスへの読み取り専用アクセス権を付与。	2018 年 9 月 20 日

変更	説明	日付
Elastic Load Balancing が変更の追跡を開始	Elastic Load Balancing は AWS 、管理ポリシーの変更の追跡を開始しました。	2018 年 9 月 20 日

Elastic Load Balancing のコンプライアンス検証

AWS のサービス が特定のコンプライアンスプログラムの範囲内にあるかどうかを確認するには、AWS のサービス「コンプライアンスプログラムによる対象範囲内」の「コンプライアンス」を参照して、関心のあるコンプライアンスプログラムを選択します。一般的な情報については、AWS「コンプライアンスプログラム」を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、「Downloading Reports in AWS Artifact」を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS のサービス は、お客様のデータの機密性、貴社のコンプライアンス目的、適用可能な法律および規制によって決まります。を使用する際のコンプライアンス責任の詳細については AWS のサービス、AWS 「 セキュリティドキュメント」を参照してください。

Elastic Load Balancing の復元性

AWS グローバルインフラストラクチャは、 AWS リージョンとアベイラビリティーゾーンを中心に構築されています。リージョンには、低レイテンシー、高いスループット、そして高度の冗長ネットワークで接続されている複数の物理的に独立および隔離されたアベイラビリティーゾーンがあります。アベイラビリティーゾーンでは、ゾーン間で中断することなく自動的にフェイルオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性、フォールトトレランス、および拡張性が優れています。

AWS リージョンとアベイラビリティーゾーンの詳細については、AWS 「 グローバルインフラストラクチャ」を参照してください。

AWS グローバルインフラストラクチャに加えて、Elastic Load Balancing には、データの耐障害性をサポートするために以下の機能が用意されています。

- 1 つまたは複数のアベイラビリティーゾーンにある複数のインスタンスの間で受信トラフィックを分散する機能

- Application Load Balancer AWS Global Accelerator で を使用すると、受信トラフィックを 1 つ以上の AWS リージョンの複数のロードバランサーに分散できます。詳細については、「AWS Global Accelerator デベロッパーガイド」を参照してください。
- Amazon ECS により、EC2 インスタンスのクラスターで Docker コンテナを実行、停止、管理できます。ロードバランサーを使用して、クラスター内のサービス全体に受信トラフィックを分散させるように、Amazon ECS サービスを設定できます。詳細については、Amazon Elastic Container Service デベロッパーガイドを参照してください。

Elastic Load Balancing のインフラストラクチャセキュリティ

マネージドサービスである Elastic Load Balancing は、 AWS グローバルネットワークセキュリティで保護されています。 AWS セキュリティサービスと がインフラストラクチャ AWS を保護する方法については、AWS 「 クラウドセキュリティ」を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して環境を AWS 設計するには、「Security Pillar AWS Well-Architected Framework」の「Infrastructure Protection」を参照してください。

AWS 公開された API コールを使用して、ネットワーク経由で Elastic Load Balancing にアクセスします。クライアントは以下をサポートする必要があります。

- Transport Layer Security (TLS)。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- DHE (楕円ディフィー・ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。これらのモードは Java 7 以降など、ほとんどの最新システムでサポートされています。

ネットワークの隔離

Virtual Private Cloud (VPC) は、 AWS クラウド内の独自の論理的に隔離されたエリアにある仮想ネットワークです。サブネットは、ある範囲の IP アドレスが示す VPC 内の領域です。ロードバランサーを作成するときに、ロードバランサーノードに 1 つ以上のサブネットを指定できます。VPC のサブネットに EC2 インスタンスをデプロイし、ロードバランサーに登録できます。VPC およびサブネットの詳細については、Amazon VPC ユーザーガイドを参照してください。

VPC 内にロードバランサーを作成する場合、インターネット向けまたは内部のいずれかを選択できます。内部向けロードバランサーは、ロードバランサー用に VPC へのアクセス権を持つクライアントからのリクエストのみをルーティングできます。

ロードバランサーは、プライベート IP アドレスを使用して、登録されたターゲットにリクエストを送信します。したがって、ロードバランサーからのリクエストを受信するために、ターゲットにパブリック IP アドレスは必要ありません。

プライベート IP アドレスを使用して VPC から Elastic Load Balancing API を呼び出すには、 AWS PrivateLinkを使用します。詳細については、「インターフェイスエンドポイントを使用して Elastic Load Balancing にアクセスする (AWS PrivateLink)」を参照してください。

ネットワークトラフィックの制御

ロードバランサーを使用する場合、ネットワークトラフィックをセキュリティで保護するには、次のオプションを検討してください。

- セキュアなリスナーを使用して、クライアントとロードバランサーの間で暗号化された通信をサポートします。Application Load Balancer は HTTPS リスナーをサポートします。Network Load Balancer は TLS リスナーをサポートします。Classic Load Balancer は、HTTPS リスナーと TLS リスナーの両方をサポートします。ロードバランサーの事前定義されたセキュリティポリシーから選択して、アプリケーションでサポートされている暗号スイートとプロトコルバージョンを指定できます。 AWS Certificate Manager (ACM) または AWS Identity and Access Management (IAM) を使用して、ロードバランサーにインストールされているサーバー証明書を管理できます。Server Name Indication (SNI) プロトコルを使用して、単一のセキュアなリスナーを使用して複数の安全なウェブサイトを提供できます。複数のサーバー証明書をセキュアリスナーに関連付けると、ロードバランサーで SNI が自動的に有効になります。
- 特定のクライアントからのトラフィックのみを受け入れるように、Application Load Balancer と Classic Load Balancer のセキュリティグループを設定します。これらのセキュリティグループは、リスナーポート上のクライアントからのインバウンドトラフィックと、クライアントへのアウトバウンドトラフィックを許可する必要があります。
- ロードバランサーからのトラフィックのみを受け入れるように、Amazon EC2 インスタンスのセキュリティグループを設定します。これらのセキュリティグループは、リスナーポートとヘルスチェックポートでロードバランサーからのインバウンドトラフィックを許可する必要があります。
- ID プロバイダーまたは社内認証を使用してユーザーを安全に認証するように Application Load Balancer を設定します。詳細については、Application Load Balancer を使用してユーザーを認証するを参照してください。
- Application Load Balancer で AWS WAF を使用して、ウェブアクセスコントロールリスト (ウェブ ACL) のルールに基づいてリクエストを許可またはブロックできます。

インターフェイスエンドポイントを使用して Elastic Load Balancing にアクセスする (AWS PrivateLink)

インターフェイス VPC エンドポイントを作成することで、Virtual Private Cloud (VPC) と Elastic Load Balancing API の間にプライベート接続を確立できます。この接続を使用すると、インターネットゲートウェイ、NAT インスタンス、または VPN 接続を VPC にアタッチする必要なく、VPC から Elastic Load Balancing API を呼び出すことができます。エンドポイントは、ロードバランサーの作成と管理に使用する Elastic Load Balancing API バージョン 2015-12-01 および 2012-06-01 への信頼性の高いスケーラブルな接続を提供します。

インターフェイス VPC エンドポイントは AWS PrivateLink、アプリケーションとプライベート IP アドレス AWS のサービス 間の通信を可能にする機能である を利用しています。詳細については、「AWS PrivateLink」を参照してください。

制限

AWS PrivateLink は、50 を超えるリスナーを持つ Network Load Balancer をサポートしていません。

Elastic Load Balancing のインターフェイスエンドポイントを作成する

以下のサービス名を使用して、Elastic Load Balancing のエンドポイントを作成します。

```
com.amazonaws.region.elasticloadbalancing
```

詳細については、「AWS PrivateLink ガイド」の「インターフェイスエンドポイントを作成」を参照してください。

Elastic Load Balancing 用の VPC エンドポイントポリシーを作成する

Elastic Load Balancing API へのアクセスをコントロールするために VPC エンドポイントにポリシーをアタッチすることができます。このポリシーでは以下の内容を指定します。

- アクションを実行できるプリンシパル。
- 実行可能なアクション。
- このアクションを実行できるリソース。

次の例は、エンドポイントを介してロードバランサーを作成するアクセス許可を全員に対して拒否する VPC エンドポイントポリシーを示しています。このポリシー例では、他のすべてのアクションを実行するアクセス許可も全員に付与しています。

```
{
   "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        },
        {
            "Action": "elasticloadbalancing:CreateLoadBalancer",
            "Effect": "Deny",
            "Resource": "*",
            "Principal": "*"
        }
    ]
}
```

詳細については、「AWS PrivateLink ガイド」の「Control access to services using endpoint policies」を参照してください。

Elastic Load Balancing API のリクエストスロットリング

Elastic Load Balancing は、リージョンごとに各 AWS アカウントの API リクエストを調整します。これは、サービスのパフォーマンスと可用性を支援するために行われます。スロットリングにより、Elastic Load Balancing API へのリクエストが API リクエストの最大許容上限を超えないことが保証されます。API リクエストは、リクエストを呼び出すか、ユーザーに代わって呼び出されるか (AWS マネジメントコンソール やサードパーティーアプリケーションなど) にかかわらず、リクエスト制限の対象となります。

Elastic Load Balancing API スロットリング制限を超えると、`ThrottlingException` エラーコードと `Rate exceeded` エラーメッセージが表示されます。

スロットリングを適切に処理する準備をすることをお勧めします。詳細については、「タイムアウト、リトライ、ジッターによるバックオフ」を参照してください。高いレベルのスロットリングが発生した場合は、 AWS サポート に連絡して API の使用状況と潜在的なソリューションの評価に役立ててください。各ケースは個別に評価されます。高可用性と予測可能なパフォーマンスを維持するために、システムの安全制限内で制限を引き上げる サポート 可能性があります。

スロットリングの適用方法

Elastic Load Balancing はトークンバケットアルゴリズムを使用して、API スロットリングを実装します。このアルゴリズムでは、アカウントには、特定の数のトークンを保持するバケットがあります。バケット内のトークンの数は、特定の秒におけるスロットリング制限を表します。

Elastic Load Balancing には、2 セットの API アクションが用意されています。ELB API バージョン 2ではロードバランサーとして、Application Load Balancer、Network Load Balancer および Gateway Load Balancer をサポートしています。ELB API バージョン 1 は Classic Load Balancer をサポートしています。各 ELB API バージョンには、独自のバケットとトークンがあります。

Amazon EC2、Amazon ECS、Amazon EC2 Auto Scaling など、ユーザーに代わって Elastic Load Balancing API を呼び出し、独自のアカウントレベルのバケット AWS CloudFormation を持つサービス。これらのサービスは、バケットのトークンを消費しません。

リクエストレート制限

リクエストレートを制限すると、実行する API リクエストの数がスロットリングされます。各リクエストは、バケットから 1 つのトークンが削除されます。たとえば、非変更 API アクションのトークンバケットサイズは 40 トークンです。1 秒あたり最大 40 個の `Describe*` リクエストを行うこ

とができます。1 秒で 40 個の `Describe*` リクエストを超えると、スロットルされ、その秒以内の残りのリクエストは失敗します。

バケットは設定されたレートで自動的に補充されます。バケットが最大容量に達していない場合、最大容量に達するまで、設定した数のトークンが毎秒追加されます。リフィルトークンが到着したときにバケットが満杯である場合、リフィルトークンは破棄されます。バケットは最大数を超えてトークンを保持することはできません。例えば、 (非変化 API アクション) のバケットサイズが 40 トークンで、リフィルレートが 1 秒あたり 10 トークンであるとします。1 秒間に 40 個の `DescribeLoadBalancers` リクエストを行うと、バケットはゼロ (0) トークンまで減らされます。バケットは、最大容量の 40 トークンに達するまで、毎秒 10 トークンが補充されます。つまり、その間にリクエストが行われなかった場合、空のバケットが最大容量に達するまでに 4 秒かかります。

API リクエストを行う前に、バケットが完全にいっぱいになるまで待つ必要はありません。トークンはバケットに追加されるとすぐに使用できます。リフィルトークンをすぐに使用しても、バケットが最大容量に達することはありません。

すべての Elastic Load Balancing API アクションで共有されるアカウントレベルのスロットリング制限があります。アカウントレベルのバケットの容量は 40 トークンで、補充レートは 1 秒あたり 10 リクエストトークンです。

リクエストトークンバケットのサイズとリフィルレート

リクエストレート制限の目的で、API アクションはカテゴリにグループ化されます。各カテゴリには独自の制限があります。

カテゴリ

- 変異アクション – リソースを作成、変更、または削除する API アクション。このカテゴリには、通常、非変異アクションとして分類されていないすべての API アクションが含まれます。これらのアクションは、非変異 API アクションよりもスロットリング上限が低くなります。
- 非変異アクション — リソースに関するデータを取得する API アクション。これらの API アクションは通常、API スロットリング上限が最も高くなります。
- リソース集約型アクション — 完了するのに最も時間がかかり、最も多くのリソースを消費する API アクション。これらのアクションのスロットリング上限は、変異アクションよりもさらに低くなります。これらのアクションは、他の変異アクションとは別にスロットリングされます。
- 登録アクション — ターゲットを登録または登録解除する API アクション。これらの API アクションは、他の変異アクションとは別にスロットリングされます。

- 未分類アクション — これらの API アクションは、他のカテゴリのいずれかに該当していても、独自のトークンバケットサイズと補充レートを受け取ります。

次の表は、分類されたリクエストトークンバケットのデフォルトの容量と補充レートを示しています。

Category	ELBv2 アクション	ELBv1 アクション	バケットキャパシティ	補充レート/秒
リソース集約型	CreateLoadBalancer , SetSubnets	CreateLoadBalancer , AttachLoadBalancerToSubnets , DetachLoadBalancerFromSubnets , EnableAvailabilityZonesForLoadBalancer , DisableAvailabilityZonesForLoadBalancer	10	0.2 †
Registration (登録)	RegisterTargets , DeregisterTargets	RegisterInstancesWithLoadBalancer , DeregisterInstancesFromLoadBalancer	20	4
非変更	DescribeAccountLimits , DescribeCapacityReservation , DescribeListenerAttributes , DescribeListenerCertificates , DescribeListeners , DescribeLoadBalancerAttributes , DescribeLoadBalanc	Describe*	40	10

Category	ELBv2 アクション	ELBv1 アクション	バケットキャパシティ	補充レート/秒
	ers , DescribeRules , DescribeSSLPolicies , DescribeTags , DescribeTargetGroupAttributes , DescribeTargetGroups , DescribeTargetHealth			

Category	ELBv2 アクション	ELBv1 アクション	バケットキャパシティ	補充レート/秒
ミューテーション	AddListenerCertificates , AddTags, CreateListener , CreateRule , CreateTargetGroup , DeleteListener , DeleteLoadBalancer , DeleteRule , DeleteTargetGroup , ModifyCapacityReservation , ModifyIpPools , ModifyListener , ModifyListenerAttributes , ModifyLoadBalancerAttributes , ModifyRule , ModifyTargetGroup , ModifyTargetGroupAttributes , RemoveListenerCertificates , RemoveTags , SetIpAddressType , SetRulePriorities , SetSecurityGroups	AddTags, ApplySecurityGroupsToLoadBalancer , ConfigureHealthCheck , CreateAppCookieStickinessPolicy , CreateLbCookieStickinessPolicy , CreateLoadBalancerListener , CreateLoadBalancerPolicy , Delete*, ModifyLoadBalancerAttributes , RemoveTags , SetLoadBalancer*	20	3

次の表は、ELBv2 の未分類リクエストトークンバケットのデフォルトの容量と補充レートを示しています。

ELBv2 アクション	バケットキャパシティ	補充レート/秒
CreateTrustStore	10	0.2 †
AddTrustStoreRevocations , DeleteSharedTrustStoreAssociation , DeleteTrustStore , ModifyTrustStore , RemoveTrustStoreRevocations	10	0.2 †
GetResourcePolicy , GetTrustStoreCaCertificatesBundle , GetTrustStoreRevocationContent	20	4
DescribeTrustStoreAssociations , DescribeTrustStoreRevocations , DescribeTrustStores	40	10

† 小数補充レートは、1 つの完全なトークンを生成するのに数秒かかります。

API リクエストのモニタリング

AWS CloudTrail を使用して、Elastic Load Balancing API リクエストをモニタリングできます。詳細については、「を使用して Elastic Load Balancing の API コールを口グに記録する AWS CloudTrail」を参照してください。

請求および使用状況レポートの Elastic Load Balancing のコードを理解する

Elastic Load Balancing を使用すると、 AWS 請求レポートと使用状況レポートに関連コードが含まれます。これらのコードを確認すると、ロードバランサーのコストと使用パターンを把握できます。コストを最適化するには、費用の追跡と管理が不可欠です。

詳細については、Elastic Load Balancing の料金表を参照してください。

以下の表では、請求および使用状況レポートに表示される Elastic Load Balancing のコードについて説明します。単位は時間またはロードバランサーキャパシティユニット (LCU) です。各ロードバランサータイプには、LCU の特定の定義があります。各ロードバランサータイプの LCU の詳細については、「Elastic Load Balancing 料金表」を参照してください。請求および使用状況レポートで使用されるリージョンコードのリストについては、「AWS Region billing codes」を参照してください。

アプリケーション ロード バランサー

[コード]	説明	単位
region-LoadBalancerUsage	実行時間。	時間
region-LCUUsage	使用されている LCU。	LCU
region-IdleProvisionedLBCapacity	予約されているが使用されていない LCU。	LCU
region-TS-LoadBalancerUsage	トラストストアが相互 TLS によって使用される時間。	時間
region-Outposts-LoadBalancerUsage	Outposts の実行時間。	時間
region-Outposts-LCUUsage	Outposts で使用されている LCU。	LCU

[コード]	説明	単位
region-ReservedLCUUsage	予約されている LCU。	LCU

Network Load Balancer

[コード]	説明	単位
region-LoadBalancerUsage	実行時間。	時間
region-LCUUsage	使用されている LCU。	LCU

Gateway Load Balancer

[コード]	説明	単位
region-LoadBalancerUsage	実行時間。	時間
region-LCUUsage	使用されている LCU。	LCU

Classic Load Balancer

[コード]	説明	単位
region-LoadBalancerUsage	実行時間。	時間
region-DataProcessing-Bytes	処理されたデータ。	GB

[コード]	説明	単位
region-IdleProvisionedLBCapacity	予約されているが使用されていない LCU。	LCU

を使用して Elastic Load Balancing の API コールをログに記録する AWS CloudTrail

Elastic Load Balancing は、ユーザー AWS CloudTrail、ロール、または のサービスによって実行されたアクションを記録する AWS サービスである と統合されています。CloudTrail は、Elastic Load Balancing の API 呼び出しをイベントとしてキャプチャします。キャプチャされた呼び出しには、からの呼び出し AWS マネジメントコンソール と Elastic Load Balancing API オペレーションへのコード呼び出しが含まれます。CloudTrail で収集された情報を使用して、Elastic Load Balancing に対するリクエスト、リクエスト元の IP アドレス、リクエストの作成日時、その他の詳細を確認できます。

各イベントまたはログエントリには、誰がリクエストを生成したかという情報が含まれます。アイデンティティ情報は、以下を判別するのに役立ちます。

- ルートユーザーまたはユーザー認証情報のどちらを使用してリクエストが送信されたか。
- リクエストが IAM Identity Center ユーザーに代わって行われたかどうか。
- リクエストがロールまたはフェデレーションユーザーのテンポラリなセキュリティ認証情報を使用して行われたかどうか。
- リクエストが、別の AWS のサービスによって送信されたかどうか。

CloudTrail は、アカウントを作成する AWS アカウント と でアクティブになり、CloudTrail イベント履歴に自動的にアクセスできます。CloudTrail の [イベント履歴] では、 AWS リージョンで過去 90 日間に記録された管理イベントの表示、検索、およびダウンロードが可能で、変更不可能な記録を確認できます。詳細については、「AWS CloudTrail ユーザーガイド」の「CloudTrail イベント履歴の使用」を参照してください。[イベント履歴] の閲覧には CloudTrail の料金はかかりません。

AWS アカウント 過去 90 日間のイベントの継続的な記録については、証跡または CloudTrail Lake イベントデータストアを作成します。

CloudTrail 証跡

証跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。を使用して作成されたすべての証跡 AWS マネジメントコンソール はマルチリージョンです。 AWS CLIを使用する際は、単一リージョンまたは複数リージョンの証跡を作成できます。アカウント AWS リージョン 内のすべての でアクティビティをキャプチャするため、マルチリージョン証跡を作成することをお勧めします。単一リージョンの証跡を作成する場合、証跡の AWS リージョンに記録

されたイベントのみを表示できます。証跡の詳細については、「AWS CloudTrail ユーザーガイド」の「AWS アカウントの証跡の作成」および「組織の証跡の作成」を参照してください。

証跡を作成すると、進行中の管理イベントのコピーを 1 つ無料で CloudTrail から Amazon S3 バケットに配信できますが、Amazon S3 ストレージには料金がかかります。CloudTrail の料金の詳細については、「AWS CloudTrail の料金」を参照してください。Amazon S3 の料金に関する詳細については、「Amazon S3 の料金」を参照してください。

CloudTrail Lake イベントデータストア

[CloudTrail Lake] を使用すると、イベントに対して SQL ベースのクエリを実行できます。CloudTrail Lake は、行ベースの JSON 形式の既存のイベントを Apache ORC 形式に変換します。ORC は、データを高速に取得するために最適化された単票ストレージ形式です。イベントは、イベントデータストアに集約されます。イベントデータストアは、高度なイベントセレクタを適用することによって選択する条件に基づいた、イベントのイミュータブルなコレクションです。どのイベントが存続し、クエリに使用できるかは、イベントデータストアに適用するセレクタが制御します。CloudTrail Lake の詳細については、 AWS CloudTrail ユーザーガイドのAWS CloudTrail 「Lake の使用」を参照してください。

CloudTrail Lake のイベントデータストアとクエリにはコストがかかります。イベントデータストアを作成する際に、イベントデータストアに使用する料金オプションを選択します。料金オプションによって、イベントの取り込みと保存にかかる料金、および、そのイベントデータストアのデフォルトと最長の保持期間が決まります。CloudTrail の料金の詳細については、「AWS CloudTrail の料金」を参照してください。

CloudTrail での Elastic Load Balancing 管理イベント

管理イベントは、 のリソースで実行される管理オペレーションに関する情報を提供します AWS アカウント。これらのイベントは、コントロールプレーンオペレーションとも呼ばれます。CloudTrail は、デフォルトで管理イベントをログ記録します。

Elastic Load Balancing は、コントロールプレーンオペレーションを管理イベントとしてログに記録します。コントロールプレーンオペレーションのリストについては以下を参照してください。

- Application Load Balancers — Elastic Load Balancing API Reference version 2015-12-01
- Network Load Balancers — Elastic Load Balancing API Reference version 2015-12-01
- Gateway Load Balancers — Elastic Load Balancing API Reference version 2015-12-01
- Classic Load Balancers — Elastic Load Balancing API Reference version 2012-06-01

Elastic Load Balancing のイベントの例

各イベントは任意の送信元からの単一のリクエストを表し、リクエストされた API オペレーション、オペレーションの日時、リクエストパラメータなどに関する情報を含みます。CloudTrail ログファイルは、パブリック API コールの順序付けられたスタックトレースではないため、イベントは特定の順序で表示されません。

次の例は、ロードバランサーを作成した後、 AWS CLIを使用してこれを削除したユーザーの CloudTrail イベントを示しています。`userAgent` 要素を使用して CLI を特定できます。`eventName` 要素を使用して、リクエストされた API コールを特定できます。ユーザーに関する情報 (`Alice`) は `userIdentity` 要素で確認できます。

Example例 1: ELBv2 API の CreateLoadBalancer


```
{
    "eventVersion": "1.03",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "123456789012",
        "arn": "arn:aws:iam::123456789012:user/Alice",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "Alice"
    },
    "eventTime": "2016-04-01T15:31:48Z",
    "eventSource": "elasticloadbalancing.amazonaws.com",
    "eventName": "CreateLoadBalancer",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "198.51.100.1",
    "userAgent": "aws-cli/1.10.10 Python/2.7.9 Windows/7 botocore/1.4.1",
    "requestParameters": {
        "subnets": ["subnet-8360a9e7","subnet-b7d581c0"],
        "securityGroups": ["sg-5943793c"],
        "name": "my-load-balancer",
        "scheme": "internet-facing"
    },
    "responseElements": {
        "loadBalancers":[{
            "type": "application",
            "loadBalancerName": "my-load-balancer",
            "vpcId": "vpc-3ac0fb5f",
            "securityGroups": ["sg-5943793c"],
```

```
            "state": {"code":"provisioning"},
            "availabilityZones": [
               {"subnetId":"subnet-8360a9e7","zoneName":"us-west-2a"},
               {"subnetId":"subnet-b7d581c0","zoneName":"us-west-2b"}
            ],
            "dNSName": "my-load-balancer-1836718677.us-west-2.elb.amazonaws.com",
            "canonicalHostedZoneId": "Z2P70J7HTTTPLU",
            "createdTime": "Apr 11, 2016 5:23:50 PM",
            "loadBalancerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/ffcddace1759e1d0",
            "scheme": "internet-facing"
        }]
    },
    "requestID": "b9960276-b9b2-11e3-8a13-f1ef1EXAMPLE",
    "eventID": "6f4ab5bd-2daa-4d00-be14-d92efEXAMPLE",
    "eventType": "AwsApiCall",
    "apiVersion": "2015-12-01",
    "recipientAccountId": "123456789012"
}
```

Example例 2: ELBv2 API の DeleteLoadBalancer

```
{
    "eventVersion": "1.03",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "123456789012",
        "arn": "arn:aws:iam::123456789012:user/Alice",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "Alice"
    },
    "eventTime": "2016-04-01T15:31:48Z",
    "eventSource": "elasticloadbalancing.amazonaws.com",
    "eventName": "DeleteLoadBalancer",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "198.51.100.1",
    "userAgent": "aws-cli/1.10.10 Python/2.7.9 Windows/7 botocore/1.4.1",
    "requestParameters": {
        "loadBalancerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/ffcddace1759e1d0"
    },
    "responseElements": null,
```

```
    "requestID": "349598b3-000e-11e6-a82b-298133eEXAMPLE",
    "eventID": "75e81c95-4012-421f-a0cf-babdaEXAMPLE",
    "eventType": "AwsApiCall",
    "apiVersion": "2015-12-01",
    "recipientAccountId": "123456789012"
}
```

Example例 3: ELB API の CreateLoadBalancer

```
{
    "eventVersion": "1.03",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAJDPLRKLG7UEXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/Alice",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "Alice"
    },
    "eventTime": "2016-04-01T15:31:48Z",
    "eventSource": "elasticloadbalancing.amazonaws.com",
    "eventName": "CreateLoadBalancer",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "198.51.100.1",
    "userAgent": "aws-cli/1.10.10 Python/2.7.9 Windows/7 botocore/1.4.1",
    "requestParameters": {
        "subnets": ["subnet-12345678","subnet-76543210"],
        "loadBalancerName": "my-load-balancer",
        "listeners": [{
           "protocol": "HTTP",
           "loadBalancerPort": 80,
           "instanceProtocol": "HTTP",
           "instancePort": 80
        }]
    },
    "responseElements": {
        "dNSName": "my-loadbalancer-1234567890.elb.amazonaws.com"
    },
    "requestID": "b9960276-b9b2-11e3-8a13-f1ef1EXAMPLE",
    "eventID": "6f4ab5bd-2daa-4d00-be14-d92efEXAMPLE",
    "eventType": "AwsApiCall",
    "apiVersion": "2012-06-01",
    "recipientAccountId": "123456789012"
```

```
}
```

Example例 4: ELB API の DeleteLoadBalancer

```
{
    "eventVersion": "1.03",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAJDPLRKLG7UEXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/Alice",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "Alice"
    },
    "eventTime": "2016-04-08T12:39:25Z",
    "eventSource": "elasticloadbalancing.amazonaws.com",
    "eventName": "DeleteLoadBalancer",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "198.51.100.1",
    "userAgent": "aws-cli/1.10.10 Python/2.7.9 Windows/7 botocore/1.4.1",
    "requestParameters": {
        "loadBalancerName": "my-load-balancer"
    },
    "responseElements": null,
    "requestID": "f0f17bb6-b9ba-11e3-9b20-999fdEXAMPLE",
    "eventID": "4f99f0e8-5cf8-4c30-b6da-3b69fEXAMPLE"
    "eventType": "AwsApiCall",
    "apiVersion": "2012-06-01",
    "recipientAccountId": "123456789012"
}
```

CloudTrail レコードの内容については、「AWS CloudTrail ユーザーガイド」の「CloudTrail record contents」を参照してください。

Classic Load Balancer の移行

Elastic Load Balancing は、Application Load Balancer、Network Load Balancer、Gateway Load Balancer、Classic Load Balancer などのロードバランサーをサポートします。各ロードバランサータイプの機能の違いの詳細については、「Elastic Load Balancing の機能」を参照してください。

また、VPC にある既存の Classic Load Balancer を、Application Load Balancer または Network Load Balancer に移行させることもできます。

Classic Load Balancer からの移行のメリット

ロードバランサーの各タイプにはそれぞれ独自の機能、関数、設定があります。各ロードバランサーの利点を確認し、最適なタイプを選定します。

Application Load Balancer

Classic Load Balancer の代わりに Application Load Balancer を使用すると以下の利点があります。

以下をサポート:

- パスの条件、ホストの条件、HTTP ヘッダーの条件。
- 特定の URL から別の URL にリクエストをリダイレクトし、1つの EC2 インスタンス上の複数のアプリケーションにリクエストをルーティングする。
- カスタムの HTTP レスポンスを返す。
- IP アドレス別にターゲットを登録し、Lambda 関数をターゲットとして登録する。ロードバランサーの VPC 外のターゲットを含める。
- 企業 ID またはソーシャル ID によりユーザーを認証する。
- Amazon Elastic Container Service (Amazon ECS) コンテナ化アプリケーション。
- 各サービスのヘルスを個別にモニタリングする。

アクセスログには追加情報が含まれ、圧縮形式で保存されます。

ロードバランサーのパフォーマンスが全体的に向上します。

Network Load Balancer

Classic Load Balancer の代わりに Network Load Balancer を使用すると、次の利点があります。

以下をサポート:

- 静的 IP アドレス。ロードバランサーに対して有効になっているサブネットごとに 1 つの Elastic IP アドレスを割り当てることができます。
- ロードバランサーの VPC 外のターゲットを含め、IP アドレス別にターゲットを登録する。
- 1 つの EC2 インスタンスで複数のアプリケーションへのリクエストをルーティングする。
- Amazon Elastic Container Service (Amazon ECS) コンテナ化アプリケーション。
- 各サービスのヘルスを個別にモニタリングする。

揮発性のワークロードを処理し、毎秒数百万のリクエストに対応できる能力。

移行ウィザードを使用した移行

移行ウィザードは、Classic Load Balancer の設定を使って同等の Application Load Balancer または Network Load Balancer を作成します。他の方法に比べ、Classic Load Balancer の移行に要する時間と労力を削減できます。

> **Note**
>
> 移行ウィザードは新しいロードバランサーを作成します。このウィザードは、既存の Classic Load Balancer を Application Load Balancer または Network Load Balancer に変換しません。トラフィックを新たに作成されたロードバランサーにリダイレクトするには手動で行う必要があります。

制限事項

- 新しいロードバランサーの名前は、同じリージョン内にある同じタイプの既存のロードバランサーと同じにすることはできません。
- Classic Load Balancer のキーに `aws:` プレフィックスを含むタグがあると、そのタグは移行されません。

Application Load Balancer に移行する場合

- Classic Load Balancer にサブネットが 1 つしかない場合、2 つ目のサブネットを指定する必要があります。

- Classic Load Balancer に TCP ヘルスチェックを使用する HTTP/HTTPS リスナーがある場合、ヘルスチェックのプロトコルは HTTP に更新され、パスは「/」に設定されます。
- Classic Load Balancer に、カスタムのまたはサポートされていないセキュリティポリシーを使用する HTTPS リスナーがある場合、移行ウィザードは新しいロードバランサータイプのデフォルトのセキュリティポリシーを使用します。

Network Load Balancer に移行する場合

- 次のインスタンスタイプは、新しいターゲットグループには登録されません。C1、CC1、CC2、CG1、CG2、CR1、CS1、G1、G2、HI1、HS1、M1、M2、M3、T1。
- Classic Load Balancer の一部のヘルスチェック設定は、新しいターゲットグループに移行できない場合があります。このような状況は、移行ウィザードの概要セクションに変更として表示されます。
- Classic Load Balancer に SSL リスナーがある場合、移行ウィザードは SSL リスナーの証明書とセキュリティポリシーを使用して TLS リスナーを作成します。

移行ウィザードのプロセス

移行ウィザードを使用して Classic Load Balancer を移行するには

1. Amazon EC2 コンソールの https://console.aws.amazon.com/ec2/ を開いてください。
2. ナビゲーションペインの [ロードバランシング] で [ロードバランサー] を選択します。
3. 移行させる Classic Load Balancer を選択します。
4. ロードバランサーの [詳細] セクションで [移行ウィザードを起動]を選択します。
5. [Application Load Balancer に移行] または [Network Load Balancer に移行] を選択して移行ウィザードを開きます。
6. [新しいロードバランサーに名前を付ける] の [ロードバランサー名] に新しいロードバランサーの名前を入力します。
7. [新しいターゲットグループに名前を付けて、ターゲットを確認] の [ターゲットグループ名] に新しいターゲットグループの名前を入力します。
8. (オプション) [ターゲット] では、新しいターゲットグループに登録されるターゲットインスタンスを確認できます。
9. (オプション) [タグを確認] では、新しいロードバランサーに適用されるタグを確認できます。

10. [Application Load Balancer の概要] または [Network Load Balancer の概要] で、移行ウィザードによって割り当てられた設定オプションを確認し、確定します。
11. 設定に問題がなければ、[Application Load Balancer を作成] または [Network Load Balancer を作成] を選択して移行を開始します。

ロードバランサーコピーユーティリティを使用した移行

ロードバランサーのコピーユーティリティは、 AWS GitHub ページの Elastic Load Balancing Tools リポジトリで使用できます。

リソース

- Elastic Load Balancing Tools
- Classic Load Balancer to Application Load Balancer copy utility
- Classic Load Balancer to Network Load Balancer copy utility

ロードバランサーを手動で移行する

以下の情報は、VPC 内の既存の Classic Load Balancer に基づいて、新しい Application Load Balancer または Network Load Balancer を手動で作成するための一般的な手順を示しています。 AWS マネジメントコンソール、、 AWS CLIまたは AWS SDK を使用して移行できます。詳細については、「Elastic Load Balancing の使用開始」を参照してください。

移行プロセスを完了すると、新しいロードバランサーの機能を利用できます。

手動による移行のプロセス

ステップ 1: 新しいロードバランサーを作成する

移行する Classic Load Balancer と同等の設定でロードバランサーを作成します。

1. 新しいロードバランサーを、Classic Load Balancer と同じスキーム (インターネット向けまたは内部向け)、サブネット、セキュリティグループを設定して作成します。
2. ロードバランサーの 1 つのターゲットグループを、Classic Load Balancerと同じヘルスチェック設定で作成します。
3. 次のいずれかを行ってください。

- Classic Load Balancer が Auto Scaling グループに接続されている場合は、ターゲットグループを Auto Scaling グループに接続します。これにより、Auto Scaling インスタンスがターゲットグループに登録されます。
- EC2 インスタンスをターゲットグループに登録します。

4. 1 つ以上のリスナーを作成し、各リスナーに、リクエストをターゲットグループに転送するデフォルトのルールを設定します。HTTPS リスナーを作成する場合は、Classic Load Balancer 用に指定したのと同じ証明書を指定できます。デフォルトのセキュリティポリシーを使用することをお勧めします。
5. Classic Load Balancer がタグ付けされている場合は、それらを見直して、関連性のあるタグを新しいロードバランサーに追加します。

ステップ 2: トラフィックを新しいロードバランサーに段階的にリダイレクトする

インスタンスを新しいロードバランサーに登録したら、古いロードバランサーから新しいロードバランサーにトラフィックをリダイレクトするプロセスを開始できます。これにより、アプリケーションの可用性に与えるリスクを最小限に抑えながら、新しいロードバランサーをテストできます。

トラフィックを新しいロードバランサーに段階的にリダイレクトするには

1. インターネットに接続したウェブブラウザのアドレスフィールドに、新しいロードバランサーの DNS 名を貼り付けます。すべて適切な場合は、ブラウザにアプリケーションのデフォルトページが表示されます。
2. ドメイン名を新しいロードバランサーに関連付ける新しい DNS レコードを作成します。DNS サービスが重み付けをサポートしている場合は、新しい DNS レコードに重み 1 を、古いロードバランサーの既存の DNS レコードに重み 9 を指定します。これで、トラフィックの 10% が新しいロードバランサーに、90% が古いロードバランサーにリダイレクトされます。
3. 新しいロードバランサーをモニタリングして、トラフィックが受信され、リクエストがインスタンスにルーティングされていることを確認します。

> **Important**
>
> DNS レコードの有効期限 (TTL) は 60 秒です。つまり、ドメイン名を解決する DNS サーバーは、変更が反映される間、レコード情報を 60 秒間キャッシュに保持します。したがって、これらの DNS サーバーは、前のステップを完了してから最大 60 秒間、トラフィックを引き続き古いロードバランサーにルーティングできます。伝達の実行中、トラフィックは両方のロードバランサーにリダイレクトされる可能性があります。

4. すべてのトラフィックが新しいロードバランサーにリダイレクトされるまで、DNS レコードの重みの更新を繰り返します。完了したら、古いロードバランサーの DNS レコードを削除できます。

ステップ 3: ポリシー、スクリプト、およびコードを更新する

Classic Load Balancer を Application Load Balancer または Network Load Balancer に移行した場合は、必ず以下のことを行ってください。

- API バージョン 2012-06-01 を使用する IAM ポリシーを更新して、バージョン 2015-12-01 を使用します。
- `AWS/ELB` 名前空間の CloudWatch メトリクスを使用するプロセスを更新して、`AWS/ApplicationELB` または `AWS/NetworkELB` 名前空間のメトリクスを使用します。
- aws elbv2 AWS CLI コマンドを使用してaws elb AWS CLI コマンドを使用するスクリプトを更新します。
- `AWS::ElasticLoadBalancing::LoadBalancer` リソースを使用する CloudFormation テンプレートを更新して、 `AWS::ElasticLoadBalancingV2`リソースを使用します。
- Elastic Load Balancing API バージョン 2012-06-01 を使用するコードを更新して、バージョン 2015-12-01 を使用します。

リソース

- AWS CLI コマンドリファレンス の elbv2
- Elastic Load Balancing API リファレンスバージョン 2015-12-01
- Elastic Load Balancing の Identity and Access Management
- Application Load Balancer ユーザーガイドの Application Load Balancer metrics
- Network Load Balancer ユーザーガイドの Network Load Balancer metrics
- AWS CloudFormation ユーザーガイド の AWS::ElasticLoadBalancingV2::LoadBalancer

ステップ 4: 古いロードバランサーを削除する

古い Classic Load Balancer は、以下を行った後に削除できます。

- すべてのトラフィックを古いロードバランサーから新しいロードバランサーにリダイレクトしました。

- 古いロードバランサーにルーティングされたすべての既存のリクエストが完了しました。

ユーザーが Classic Load Balancer を作成できないようにする

ユーザーがアカウントに Classic Load Balancer を作成できないようにする IAM ポリシーを作成できます。

Elastic Load Balancing V2 API と Elastic Load Balancing V1 API の両方が `CreateLoadBalancer` API アクションを提供します。Classic Load Balancer を作成するときは、ロードバランサーとリスナーの両方を作成する V1 API アクションを使用します。Application Load Balancer、Network Load Balancer、または Gateway Load Balancer を作成するときは、ロードバランサーのみを作成する V2 API アクションを使用します。V2 API は、ロードバランサーの作成後にリスナーを作成するために使用する `CreateListener` アクションを提供します。

次のポリシーは、リスナープロトコルが指定されている場合、ロードバランサーを作成するアクセス許可をユーザーに拒否します。Classic Load Balancer の作成時に少なくとも 1 つのリスナーを設定する必要があるため、このポリシーはユーザーが Classic Load Balancer を作成できないようにします。これらのロードバランサーとそのリスナーを作成するための個別の API アクションがあるため、ユーザーが他のタイプのロードバランサーを作成するのを防ぐことはできません。

```
{
    "Version": "2012-10-17",
    "Effect": "Deny",
    "Action": "elasticloadbalancing:CreateLoadBalancer",
    "Resource": [
        "arn:aws:elasticloadbalancing:*:*:loadbalancer/*"
    ],
    "Condition": {
        "Null": {
            "elasticloadbalancing:ListenerProtocol": false
        }
    }
}
```

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。