aws

アプリケーション ロード バランサー

Elastic Load Balancing

Elastic Load Balancing: アプリケーション ロード バランサー

Table of Contents

Application Load Balancer 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、受信トラフィックの時間的な変化に応じて、ロードバランサーをスケーリングします。また、大半のワークロードに合わせて自動的にスケーリングできます。

Elastic Load Balancing は、Application Load Balancer、Network Load Balancer、Gateway Load Balancer、Classic Load Balancer といったロードバランサーをサポートします。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。このガイドでは、Application Load Balancers について説明します。その他のロードバランサーの詳細については、Network Load Balancer ユーザーガイド、Gateway Load Balancer ユーザーガイド、およびClassic Load Balancer ユーザーガイドを参照してください。

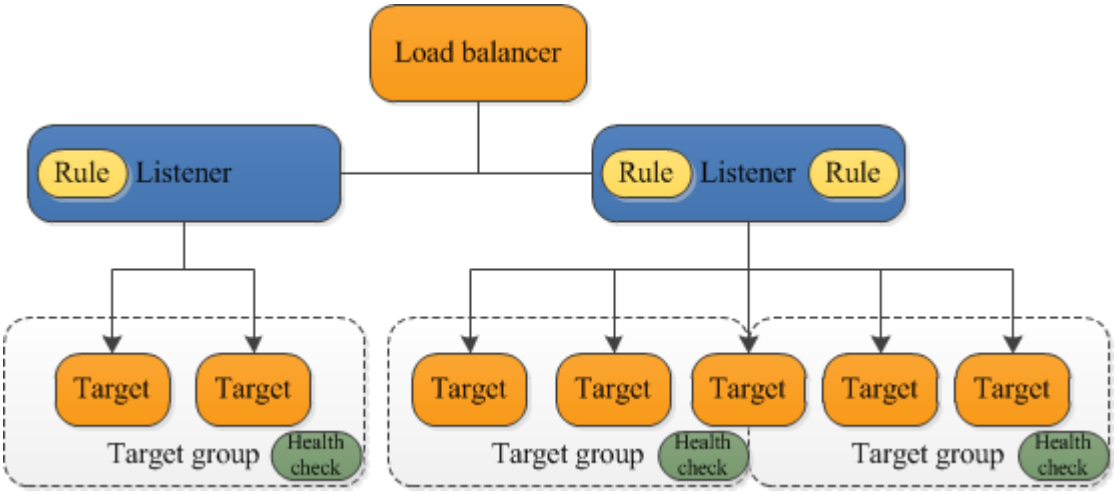
Application Load Balancer のコンポーネント

ロードバランサーは、クライアントにとって単一の通信先として機能します。このロードバランサーは、受信アプリケーション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 インスタンスなど) に分散します。これにより、アプリケーションの可用性が向上します。ロードバランサーに 1 つ以上のリスナーを追加できます。

リスナーは、設定したプロトコルとポートを使用して、クライアントからの接続リクエストをチェックします。リスナーに対して定義したルールにより、ロードバランサーが登録済みターゲットにリクエストをルーティングする方法が決まります。各ルールは優先度、1 つ以上のアクション、および 1 つ以上の条件で構成されています。ルールの条件が満たされると、アクションが実行されます。リスナーごとにデフォルトのルールを定義する必要があり、オプションで追加のルールを定義できます。

各ターゲットグループは、指定されたプロトコルとポート番号を使用して、1 つ以上の登録済みのターゲット (EC2 インスタンスなど) にリクエストをルーティングできます。1 つのターゲットを複数のターゲットグループに登録できます。ターゲットグループ単位でヘルスチェックを設定できます。ヘルスチェックは、ロードバランサーのリスナールールに指定されたターゲットグループに登録されたすべてのターゲットで実行されます。

次の図に、基本コンポーネントを示します。各リスナーにデフォルトのルールがあり、1 つのリスナーにリクエストを別のターゲットグループにルーティングする別のルールが含まれていることに注意してください。1 つのターゲットが 2 つのターゲット グループに登録されています。



詳細については、次のドキュメントを参照してください。

- ロードバランサー
- リスナー
- ターゲットグループ

Application Load Balancer の概要

Application Load Balancer は、開放型システム間相互接続 (OSI) モデルの第 7 層であるアプリケーションレイヤーで機能します。ロードバランサーはリクエストを受信すると、優先度順にリスナールールを評価して適用するルールを決定し、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。リスナールールを構成し、アプリケーショントラフィックのコンテンツに基づいて異なるターゲットグループにリクエストをルーティングできます。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。ターゲットグループレベルで使用するルーティングアルゴリズムを設定できます。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。代わりに最小の未処理のリクエストを指定することもできます。

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ターゲットの追加と削除を行うことができます。Elastic Load Balancing は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のに応じてロードバランサーをスケーリングします。Elastic Load Balancing では、大半のワークロードに合わせた自動的なスケーリングが可能です。

登録済みのインスタンスのヘルス状態をモニタリングするために使用されるヘルスチェックを設定することで、ロードバランサーは正常なターゲットにのみリクエストを送信できます。

詳細については、Elastic Load Balancing ユーザーガイドの How Elastic Load Balancing works を参照してください。

Classic Load Balancer からの移行のメリット

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。

- パスの条件 のサポート。リクエスト内の URL に基づいてリクエストを転送するリスナーのルールを設定できます。これにより、アプリケーションをより小さなサービスとして構成し、URL の内容に基づいて適切なサービスにリクエストをルーティングできます。
- ホストの条件 のサポート。HTTP ヘッダー内のホストフィールドに基づいてリクエストを転送するリスナーのルールを設定できます。これにより、1 つのロードバランサーを使用して複数のドメインにリクエストをルーティングできます。
- HTTP ヘッダー条件 とメソッド、クエリパラメータ、送信元 IP アドレスなど、リクエスト内のフィールドに基づくルーティングのサポート。
- 1 つの EC2 インスタンス上での複数のアプリケーションへのルーティングリクエストのサポート。インスタンスまたは IP アドレスは、それぞれ異なるポート上の複数のターゲットグループに登録できます。
- 1 つの URL から別の URL へのリクエストのリダイレクトのサポート。
- カスタム HTTP レスポンスの出力のサポート。
- ロードバランサーの VPC 外のターゲットを含め、IP アドレスによるターゲットの登録をサポート。
- ターゲットとしての Lambda 関数の登録のサポート。
- リクエストをルーティングする前に企業 ID またはソーシャル ID を通じてアプリケーションのユーザーを認証するロードバランサーのサポート。
- コンテナ化されたアプリケーションのサポート。Amazon Elastic Container Service (Amazon ECS) は、タスクをスケジュールするときに未使用のポートを選択し、そのポートを使用するターゲットグループにタスクを登録できます。これにより、クラスターを効率的に使用することができます。
- ターゲットグループレベルでヘルスチェックが定義され、ターゲットグループレベルで多くの CloudWatch メトリクスがレポートされるため、各サービスの個別のヘルスモニタリングのサポート。ターゲットグループを Auto Scaling グループにアタッチすることで、各サービスをオンデマンドで動的にスケールすることができます。
- アクセスログへの情報の追加と圧縮形式での保存。

- ロードバランサーのパフォーマンスの向上。

各ロードバランサータイプでサポートされている機能の詳細については、Elastic Load Balancing の製品比較を参照してください。

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。
- 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インスタンスの実行数を保証し、需要の変化に応じて自動的にインスタンス数を増減できるようにします。Elastic Load Balancing を使用して Auto Scaling を有効にした場合、Auto Scaling によって起動されたインスタンスは自動的にターゲットグループに登録され、Auto Scaling によって終了されたインスタンスは自動的にターゲットグループから登録解除されます。
- 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。詳細については、「SSL 証明書」を参照してください。
- Amazon CloudWatch — ロードバランサーをモニタリングし、必要に応じてアクションを実行できます。詳細については、「CloudWatch Application Load Balancer の メトリクス」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 の Service load balancing 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 (`www.example.com` など) 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 (`192.0.2.1` など) 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。AWS は、ロードバランサーなどの URL をリソースに割り当てます。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。

- AWS WAF — 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。詳細については、「Application Load Balancer と AWS WAF」を参照してください。

ロードバランサーに統合されたサービスに関する情報を表示するには、AWS Management Console でロードバランサーを選択し、[統合されたサービス] タブを選択します。

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。

Application Load Balancer の開始方法

このチュートリアルでは、ウェブベースのインターフェイスである を通じて AWS Management Console Application Load Balancer を実際に紹介します。最初の Application Load Balancer を作成するには、次のステップを完了します。

タスク

- 開始する前に
- ステップ 1: ターゲットグループの設定
- ステップ 1: ロードバランサーの種類の選択
- ステップ 2: ロードバランサーとリスナーの設定
- ステップ 4: ロードバランサーのテスト
- ステップ 5: ロードバランサーを削除 (オプション)

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。

開始する前に

- EC2 インスタンスに使用する 2 つのアベイラビリティーゾーンを決定します。これらの各アベイラビリティーゾーンに少なくとも 1 つのパブリックサブネットがある Virtual Private Cloud (VPC) を設定します。これらのパブリックサブネットは、ロードバランサーを設定するために使用されます。その代わりに、これらのアベイラビリティーゾーンの他のサブネットで EC2 インスタンスを起動することができます。
- 各アベイラビリティーゾーンで少なくとも 1 つの EC2 インスタンスを起動します。必ず、Apache や Internet Information Services (IIS) などのウェブサーバーを各 EC2 インスタンスにインストールします。これらのインスタンスのセキュリティグループでは、ポート 80 の HTTP アクセスを許可していることを確認してください。

ステップ 1: ターゲットグループの設定

リクエストルーティングで使用されるターゲットグループを作成します。リスナーのデフォルトルールは、このターゲットグループ内の登録済みターゲットにリクエストをルーティングします。ロード

バランサーは、ターゲットグループに定義されたヘルスチェック設定を使用してこのターゲットグループ内のターゲットの状態を確認します。

コンソールを使用してターゲットグループを設定するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインの [ロードバランシング] で [ターゲットグループ] を選択します。
3. [Create target group] を選択します。
4. [基本的な設定] で、[ターゲットタイプ] をインスタンスのままにします。
5. [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。
6. デフォルトのプロトコル (HTTP) とポート (80) のままにします。
7. インスタンスを含んでいる VPC を選択します。プロトコルのバージョンを HTTP1 のままにします。
8. [ヘルスチェック] で、デフォルトの設定を保持します。
9. [次へ] をクリックします。
10. [ターゲットの登録] ページで、次の手順を完了します。これは、ロードバランサーを作成するオプションのステップです。ただし、ロードバランサーをテストし、このターゲットにトラフィックをルーティングしていることを確認する場合は、ターゲットを登録する必要があります。
    a. [使用可能なインスタンス] で、1 つ以上のインスタンスを選択します。
    b. デフォルトのポート 80 のままにして、[保留中として以下を含める] を選択します。
11. [ターゲットグループの作成] を選択します。

ステップ 1: ロードバランサーの種類の選択

Elastic Load Balancing では、異なる種類のロードバランサーがサポートされています。このチュートリアルでは、Application Load Balancer を作成します。

コンソールを使用して Application Load Balancer を作成するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションバーで、ロードバランサーのリージョンを選択します。EC2 インスタンス用に使用したリージョンと同じリージョンを必ず選択してください。
3. ナビゲーションペインの [Load Balancing] で、[Load Balancers] を選択します。

4. [Create Load Balancer] を選択します。
5. [Application Load Balancer] で [作成] を選択します。

ステップ 2: ロードバランサーとリスナーの設定

Application Load Balancer を作成するには、まず、名前、スキーム、IP アドレスのタイプなど、ロードバランサーの基本設定情報を指定する必要があります。次に、ネットワークに関する情報と 1 つ以上のリスナーを指定します。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を使用して設定します。サポートされるプロトコルとポートの詳細については、「リスナーの設定」を参照してください。

ロードバランサーとリスナーを設定するには

1. [ロードバランサー名] に、ロードバランサーの名前を入力します。たとえば、`my-alb` と指定します。
2. [スキーム] および [IP アドレスタイプ] については、デフォルト値のままにします。
3. [ネットワークマッピング] で、EC2 インスタンスに使用する VPC を選択します。アベイラビリティーゾーンを少なくとも 2 つ選択し、ゾーンごとに 1 つのサブネットを選択します。EC2 インスタンスの起動に使用した各アベイラビリティーゾーンについて、アベイラビリティーゾーンを選択し、そのアベイラビリティーゾーンのパブリックサブネットを 1 つ選択します。
4. [セキュリティグループ] で、前のステップで選択した VPC のデフォルトのセキュリティグループを選択します。代わりに、別のセキュリティグループを選択できます。セキュリティグループには、ロードバランサーがリスナーポートとヘルスチェックポートの両方で登録済みのターゲットと通信することを許可するルールが含まれている必要があります。詳細については、「セキュリティグループのルール」を参照してください。
5. [リスナーとルーティング] で、デフォルトのプロトコルとポートを保持し、リストからターゲットグループを選択します。これにより、ポート 80 で HTTP トラフィックを受け付けるリスナーが設定され、選択されたターゲットグループへトラフィックをデフォルトで転送します。このチュートリアルでは、HTTPS リスナーを作成しません。
6. [デフォルトのアクション] で、ステップ 1: [ターゲットグループを設定] で作成して登録したターゲットグループを選択します。
7. (オプション) タグを追加して、ロードバランサーを分類します。タグキーは、各ロードバランサーで一意である必要があります。使用できる文字は、文字、スペース、数字 (UTF-8)、および特殊文字 (+-=. _:/@) です。ただし、先頭または末尾にはスペースを使用しないでください。タグ値は大文字と小文字が区別されます。

8. 設定を確認し、[ロードバランサーの作成] を選択します。作成時に、ロードバランサーにいくつかのデフォルト属性が適用されます。ロードバランサーの作成後に、それらを表示および編集できます。詳細については、「ロードバランサーの属性」を参照してください。

ステップ 4: ロードバランサーのテスト

ロードバランサーを作成した後で、EC2 インスタンスにトラフィックを送信するかどうかを検証します。

ロードバランサーをテストするには

1. ロードバランサーが正常に作成されたことが通知されたら、[閉じる] を選択します。
2. ナビゲーションペインの [ロードバランシング] で [ターゲットグループ] を選択します。
3. 新しく作成したターゲットグループを選択します。
4. [Targets] を選択して、インスタンスの準備ができていることを確認します。インスタンスのステータスが `initial` の場合、インスタンスがまだ登録の途中であるか、正常と見なされるのに必要なヘルスチェックの最小数に合格しなかったと考えられます。少なくとも 1 つのインスタンスのステータスが `healthy` であれば、ロードバランサーをテストできます。
5. ナビゲーションペインの [Load Balancing] で、[Load Balancers] を選択します。
6. 新しく作成したロードバランサーを選択します。
7. 説明 を選択し、ロードバランサーの DNS 名をコピーします (例: my-load-balancer-1234567890abcdef.elb.us-east-2.amazonaws.com)。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 DNS 名を貼り付けます。すべて適切な場合は、ブラウザにサーバーのデフォルトページが表示されます。
8. (省略可能) 追加のリスナールールを定義するには、「ルールの追加」を参照してください。

ステップ 5: ロードバランサーを削除 (オプション)

ロードバランサーが利用可能になると、ロードバランサーの実行時間に応じて 1 時間ごと、または 1 時間未満の時間について課金されます。不要になったロードバランサーは削除できます。ロードバランサーが削除されると、ロードバランサーの課金も停止されます。ロードバランサーを削除しても、ロードバランサーに登録されたターゲットには影響を与えません。例えば、このガイドで作成したロードバランサーを削除した後も、EC2 インスタンスは実行され続けます。

コンソールを使用してロードバランサーを削除するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインの [Load Balancing] で、[Load Balancers] を選択します。
3. ロードバランサーのチェックボックスを選択し、[アクション]、[削除] の順に選択します。
4. 確認を求めるメッセージが表示されたら、[Yes、Delete] を選択します。

チュートリアル: AWS CLIを使用した Application Load Balancer の作成

このチュートリアルでは、 を通じて Application Load Balancer を実際に紹介します AWS CLI。

開始する前に

- 次のコマンドを使用して、Application Load Balancer をサポートするバージョンの AWS CLI を実行していることを確認します。

```
aws elbv2 help
```

  elbv2 が有効な選択肢ではないことを示すエラーメッセージが発生した場合は、 AWS CLIを更新します。詳細については、AWS Command Line Interface ユーザーガイドの「AWS Command Line Interfaceのインストール」を参照してください。
- Virtual Private Cloud (VPC) で EC2 インスタンスを起動します。これらのインスタンスのセキュリティグループがリスナーポートとヘルスチェックポートでアクセスを許可することを確認します。詳細については、「ターゲットセキュリティグループ」を参照してください。
- IPv4 ロードバランサーとデュアルスタックロードバランサーのどちらを作成するかを決定します。クライアントが IPv4 アドレスだけを使用してロードバランサーと通信する場合、IPv4 を使用します。クライアントが IPv4 および IPv6 アドレスを使用してロードバランサーと通信する場合、デュアルスタックを使用します。また、IPv6 を使用して IPv6 アプリケーションやデュアルスタックサブネットなどのバックエンドターゲットと通信するために、デュアルスタックを使用することもできます。
- 必ず、Apache や Internet Information Services (IIS) などのウェブサーバーを各 EC2 インスタンスにインストールします。これらのインスタンスのセキュリティグループでは、ポート 80 の HTTP アクセスを許可していることを確認してください。

ロードバランサーを作成する

最初のロードバランサーを作成するには、次のステップを完了します。

ロードバランサーを作成するには

1. create-load-balancer コマンドを使用してロードバランサーを作成します。同じアベイラビリティーゾーンにない 2 つのサブネットを指定する必要があります。

```
aws elbv2 create-load-balancer --name my-load-balancer  \
--subnets subnet-0e3f5cac72EXAMPLE subnet-081ec835f3EXAMPLE --security-groups
 sg-07e8ffd50fEXAMPLE
```

   create-load-balancer コマンドを使用してロード**dualstack**バランサーを作成します。

```
aws elbv2 create-load-balancer --name my-load-balancer  \
--subnets subnet-0e3f5cac72EXAMPLE subnet-081ec835f3EXAMPLE --security-groups
 sg-07e8ffd50fEXAMPLE --ip-address-type dualstack
```

   出力には、次の形式でロードバランサーの Amazon リソースネーム (ARN) が含まれます。

```
arn:aws:elasticloadbalancing:us-east-2:123456789012:loadbalancer/app/my-load-
balancer/1234567890123456
```

2. create-target-group コマンドを使用してターゲットグループを作成し、EC2 インスタンスに使用したのと同じ VPC を指定します。

   IPv4 ターゲットグループおよび IPv6 ターゲットグループを作成して、デュアルスタックロードバランサーに関連付けることができます。ターゲットグループの IP アドレスタイプによって、ロードバランサーがバックエンドターゲットと通信したり、バックエンドターゲットの状態をチェックしたりするのに使用する IP バージョンが決定されます。

```
aws elbv2 create-target-group --name my-targets --protocol HTTP --port 80 \
--vpc-id vpc-0598c7d356EXAMPLE --ip-address-type [ipv4 or ipv6]
```

   出力には、次の形式のターゲットグループの ARN が含まれます。

```
arn:aws:elasticloadbalancing:us-east-2:123456789012:targetgroup/my-
targets/1234567890123456
```

3. インスタンスをターゲットグループに登録するには、register-targets コマンドを使用します。

```
aws elbv2 register-targets --target-group-arn targetgroup-arn  \
```

```
--targets Id=i-0abcdef1234567890 Id=i-1234567890abcdef0
```

4. ターゲットグループにリクエストを転送するデフォルトルールを持つロードバランサーのリスナーを作成するには、create-listener コマンドを使用します。

```
aws elbv2 create-listener --load-balancer-arn loadbalancer-arn \
--protocol HTTP --port 80  \
--default-actions Type=forward,TargetGroupArn=targetgroup-arn
```

出力には、次の形式のリスナーの ARN が含まれます。

```
arn:aws:elasticloadbalancing:us-east-2:123456789012:listener/app/my-load-
balancer/1234567890123456/1234567890123456
```

5. (オプション) 次のdescribe-target-healthコマンドを使用して、ターゲットグループに登録されたターゲットの状態を確認できます。

```
aws elbv2 describe-target-health --target-group-arn targetgroup-arn
```

HTTPS リスナーの追加

HTTP リスナーを持つロードバランサーがある場合、次のように HTTPS リスナーを追加できます。

ロードバランサーに HTTPS リスナーを追加するには

1. 次のいずれかの方法を使用して、ロードバランサーで使用する SSL 証明書を作成します。
   - AWS Certificate Manager (ACM) を使用して証明書を作成またはインポートします。詳細については、AWS Certificate Manager ユーザーガイド の 証明書のリクエスト または 証明書のインポート を参照してください。
   - AWS Identity and Access Management (IAM) を使用して証明書をアップロードします。詳細については、IAM ユーザーガイド の Working with server certificates を参照してください。
2. ターゲットグループにリクエストを転送するデフォルトルールを持つリスナーを作成するには、create-listener コマンドを使用します。HTTPS リスナーを作成するときは、SSL 証明書を指定する必要があります。--ssl-policy オプションを使用してデフォルト以外の SSL ポリシーを指定できます。

```
aws elbv2 create-listener --load-balancer-arn loadbalancer-arn \
```

```
--protocol HTTPS --port 443  \
--certificates CertificateArn=certificate-arn \
--default-actions Type=forward,TargetGroupArn=targetgroup-arn
```

パスベースのルーティングの追加

1 つのターゲットグループにリクエストを転送するデフォルトルールを持つリスナーがある場合、URL に基づいて別のターゲットグループにリクエストを転送するルールを追加できます。たとえば、1 つのターゲットグループに全般的なリクエストをルーティングし、イメージを表示するリクエストを別のターゲットグループにルーティングできます。

パスパターンを持つリスナーにルールを追加するには

1. create-target-group コマンドを使用してターゲットグループを作成します。

   ```
   aws elbv2 create-target-group --name my-targets --protocol HTTP --port 80 \
   --vpc-id vpc-0598c7d356EXAMPLE
   ```

2. インスタンスをターゲットグループに登録するには、register-targets コマンドを使用します。

   ```
   aws elbv2 register-targets --target-group-arn targetgroup-arn  \
   --targets Id=i-0abcdef1234567890 Id=i-1234567890abcdef0
   ```

3. URL に指定されたパターンが含まれている場合に、ターゲットグループにリクエストを転送するルールをリスナーに追加するには、create-rule コマンドを使用します。

   ```
   aws elbv2 create-rule --listener-arn listener-arn --priority 10 \
   --conditions Field=path-pattern,Values='/img/*' \
   --actions Type=forward,TargetGroupArn=targetgroup-arn
   ```

ロードバランサーの削除

ロードバランサーとターゲットグループが必要なくなった場合は、次のように削除することができます。

```
aws elbv2 delete-load-balancer --load-balancer-arn loadbalancer-arn
aws elbv2 delete-target-group --target-group-arn targetgroup-arn
```

Application Load Balancer

ロードバランサーは、クライアントにとって単一の通信先として機能します。クライアントはロードバランサーにリクエストを送信し、ロードバランサーはターゲット (EC2 インスタンスなど) にそれらのリクエストを送信します。ロードバランサーを設定するには、ターゲットグループを作成し、ターゲットグループにターゲットを登録します。さらに、リスナーを作成してクライアントからの接続リクエストがないかチェックし、リスナールールを作成してリクエストをクライアントから 1 つ以上のターゲットグループ内のターゲットにルーティングします。

詳細については、Elastic Load Balancing ユーザーガイドの How Elastic Load Balancing works を参照してください。

目次

- ゾーンシフト

ロードバランサーのサブネット

Application Load Balancer を作成するときは、ターゲットを含むゾーンを有効にする必要があります。ゾーンを有効にするには、ゾーン内のサブネットを指定します。Elastic Load Balancing は、指定した各ゾーンにロードバランサーノードを作成します。

考慮事項

- 有効な各ゾーンに 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。
- ターゲットをアベイラビリティーゾーンに登録したが、ゾーンを有効にしていない場合、登録したターゲットはロードバランサーからのトラフィックを受信しません。
- ロードバランサーで複数のゾーンを有効にする場合、ゾーンは同じタイプでなければなりません。例えば、アベイラビリティーゾーンとローカルゾーンの両方を有効にすることはできません。
- 自分と共有しているサブネットを指定できます。

Application Load Balancer では、次の タイプのサブネットがサポートされます。

サブネットタイプ

- アベイラビリティーゾーンサブネット
- ローカルゾーンサブネット
- Outpost サブネット

アベイラビリティーゾーンサブネット

少なくとも 2 つのアベイラビリティーゾーンサブネットを選択する必要があります。以下の制限が適用されます。

- それぞれのサブネットは、異なるアベイラビリティーゾーンに属している必要があります。
- ロードバランサーが正しくスケールできるように、ロードバランサーのアベイラビリティーゾーンサブネットごとに CIDR ブロックを最低でも /27 ビットマスク (例: `10.0.0.0/27`) にし、少なくとも各サブネットにつき 8 個の空き IP アドレスを用意してください。これらの 8 個の IP アドレスは、ロードバランサーが必要に応じてスケールアウトできるようにするために必要です。ロード

バランサーはこれらの IP アドレスを使用して、ターゲットとの接続を確立します。それらがないと、Application Load Balancer ではノード交換の試行で問題が発生し、失敗状態になる可能性があります。

注: スケールの試行中に Application Load Balancer サブネットが使用可能な IP アドレスを使い果たした場合、Application Load Balancer は不十分なキャパシティで実行されます。この間、古いノードは引き続きトラフィックを処理しますが、スケーリングの試行が停止すると、接続の確立を試行する際に 5xx エラーまたはタイムアウトが発生する可能性があります。

ローカルゾーンサブネット

1 つ以上のローカルゾーンサブネットを指定できます。以下の制限が適用されます。

- ロードバランサー AWS WAF で を使用することはできません。
- Lambda 関数をターゲットとして使用することはできません。
- スティッキーセッションやアプリケーションの維持を使用することはできません。

Outpost サブネット

1 つの Outpost サブネットを指定できます。以下の制限が適用されます。

- オンプレミスのデータセンターに Outpost をインストールして設定しておく必要があります。Outpost と AWS リージョンの間に信頼できるネットワーク接続が必要です。詳細については、AWS Outposts ユーザーガイドを参照してください。
- ロードバランサーでは、ロードバランサーノードの Outpost に 2 つの `large` インスタンスが必要です。サポートされているインスタンスタイプを以下の表に示します。ロードバランサーは必要に応じてスケールし、一度に 1 サイズずつノードのサイズ変更を行います (`large` から `xlarge` に変更、`xlarge` から `2xlarge` に変更、あるいは `2xlarge` から`4xlarge` に変更)。ノードを最大のインスタンスサイズにスケールした後、追加の容量が必要な場合は、`4xlarge` インスタンスをロードバランサーノードとして追加します。ロードバランサーをスケールするのに十分なインスタンス容量または使用可能な IP アドレスがない場合、ロードバランサーは AWS Health Dashboard にイベントを報告し、ロードバランサーの状態は `active_impaired` になります。
- インスタンス ID または IP アドレスでターゲットを登録できます。Outpost の AWS リージョンにターゲットを登録する場合、ターゲットは使用されません。
- ターゲットとしての Lambda 機能、 AWS WAF 統合、スティッキーセッション、認証サポート、および AWS Global Acceleratorとの統合は使用できません。

Application Load Balancer は、Outpost の c5/c5d、m5/m5d、または r5/r5d インスタンスにデプロイできます。次の表は、ロードバランサーが Outpost で使用できるインスタンスタイプごとのサイズと EBS ボリュームを示しています。

インスタンスタイプとサイズ	EBS ボリューム (GB)
c5/c5d	
large	50
xlarge	50
2xlarge	50
4xlarge	100
m5/m5d	
large	50
xlarge	50
2xlarge	100
4xlarge	100
r5/r5d	
large	50
xlarge	100
2xlarge	100
4xlarge	100

ロードバランサーのセキュリティグループ

セキュリティグループは、ロードバランサーとの間で許可されているトラフィックを制御するファイアウォールとして機能します。インバウンドトラフィックとアウトバウンドトラフィックの両方を許可するポートとプロトコルを選択できます。

ロードバランサーに関連付けられたセキュリティグループのルールは、リスナーポートとヘルスチェックポートの両方における両方向のトラフィックを許可する必要があります。リスナーをロードバランサーに追加するとき、またはターゲットグループのヘルスチェックポートを更新するときは必ず、セキュリティグループルールを見直し、新しいポートで両方向のトラフィックが許可されていることを確認する必要があります。詳細については、「推奨ルール」を参照してください。

ロードバランサーの状態

ロードバランサーの状態は次のいずれかです。

`provisioning`

ロードバランサーはセットアップ中です。

`active`

ロードバランサーは完全にセットアップされており、トラフィックをルーティングする準備ができています。

`active_impaired`

ロードバランサーはトラフィックをルーティングしていますが、スケールするのに必要なリソースがありません。

`failed`

ロードバランサークラウドをセットアップできませんでした。

ロードバランサーの属性

ロードバランサーの属性は以下のとおりです。

`access_logs.s3.enabled`

Amazon S3 に保存されたアクセスログが有効かどうかを示します。デフォルト: `false`。

`access_logs.s3.bucket`

アクセスログの Amazon S3 バケットの名前。この属性は、アクセスログが有効になっている場合は必須です。詳細については、「アクセスログの有効化」を参照してください。

`access_logs.s3.prefix`

Amazon S3 バケットの場所のプレフィックス。

`client_keep_alive.seconds`

クライアントのキープアライブ値を秒単位で表します。デフォルトは 3600 秒です。

`deletion_protection.enabled`

削除保護が有効化されているかどうかを示します。デフォルト: `false`。

`idle_timeout.timeout_seconds`

アイドルタイムアウト値 (秒単位)。デフォルト値は 60 秒です。

`ipv6.deny_all_igw_traffic`

ロードバランサーへのインターネットゲートウェイ (IGW) アクセスをブロックし、インターネットゲートウェイ経由の内部ロードバランサーへの意図しないアクセスを防止します。インターネット向けロードバランサーでは `false`、内部ロードバランサーでは `true` に設定されます。この属性は、IGW 以外のインターネットアクセス (ピアリング、Transit Gateway、、 AWS Direct Connectなど) を妨げません AWS VPN。

`routing.http.desync_mitigation_mode`

アプリケーションにセキュリティ上のリスクをもたらす可能性があるリクエストをロードバランサーで処理する方法を指定します。指定できる値は、`monitor`、`defensive`、および `strictest` です。デフォルト: `defensive`。

`routing.http.drop_invalid_header_fields.enabled`

有効ではないヘッダーフィールドを持つ HTTP ヘッダーがロードバランサーによって削除されるか (`true`)、ターゲットにルーティングされるか (`false`) を示します。デフォルト: `false`。Elastic Load Balancing では、HTTP フィールド名レジストリに記載されているとおり、有効な HTTP ヘッダー名が正規表現 `[-A-Za-z0-9]+` に準拠している必要があります。それぞれの名前は英数字またはハイフンで構成されます。このパターンに適合しない HTTP ヘッダーをリクエストから削除したい場合は、`true` を選択します。

routing.http.preserve_host_header.enabled

Application Load Balancer が HTTP リクエストに Host ヘッダーを保持し、変更を加えずターゲットに送信するかどうかを示します。指定できる値は true および false です。デフォルトは false です。

routing.http.x_amzn_tls_version_and_cipher_suite.enabled

ネゴシエートされた TLS バージョンと暗号スイートに関する情報を含む 2 つのヘッダー (x-amzn-tls-version および x-amzn-tls-cipher-suite) が、ターゲットに送信される前にクライアントのリクエストに追加されるかどうかを指定します x-amzn-tls-version ヘッダーには、クライアントとネゴシエートされた TLS プロトコルのバージョンに関する情報があり、x-amzn-tls-cipher-suite ヘッダーには、クライアントとネゴシエートされた暗号スイートに関する情報があります。どちらのヘッダーも OpenSSL 形式です。この属性に指定できる値は true と false です。デフォルト: false。

routing.http.xff_client_port.enabled

X-Forwarded-For ヘッダーが、クライアントがロードバランサーへの接続に使用したソースポートを保持するかどうかを指定します。指定できる値は true および false です。デフォルトは false です。

routing.http.xff_header_processing.mode

Application Load Balancer がターゲットにリクエストを送信する前に、HTTP リクエストの X-Forward-For ヘッダーを変更、保持、または削除できるようにします。指定できる値は、append、preserve、および remove です。デフォルト: append。

- 値が append の場合、Application Load Balancer は、(ラストホップの) クライアント IP アドレスを HTTP リクエストの X-Forward-For ヘッダーに追加してからターゲットに送信します。
- 値が preserve の場合、Application Load Balancer は、HTTP リクエストの X-Forward-For ヘッダーを保持し、変更を加えずにターゲットに送信します。
- 値が remove の場合、Application Load Balancer は、HTTP リクエストの X-Forward-For ヘッダーを削除してからターゲットに送信します。

routing.http2.enabled

HTTP/2 が有効化されているかどうかを示します。デフォルト: true。

`waf.fail_open.enabled`

リクエストを に転送できない場合に、 AWS WAFが有効なロードバランサーがターゲットにリクエストをルーティングできるようにするかどうかを示します AWS WAF。指定できる値は `true` および `false` です。デフォルトは `false` です。

> **Note**
>
> `routing.http.drop_invalid_header_fields.enabled` 属性は HTTP 非同期保護を提供するために導入されました。`routing.http.desync_mitigation_mode` 属性は、アプリケーションの HTTP 非同期からのより包括的な保護を提供するために追加されました。両方の属性を使用する必要はなく、アプリケーションの要件に応じてどちらかを選択できます。

IP アドレスタイプ

インターネット向けや内部のロードバランサーへのアクセスにクライアントが使用できる IP アドレスのタイプは、ユーザーが設定できます。

Application Load Balancer は、次の IP アドレスタイプをサポートしています。

`ipv4`

クライアントは IPv4 アドレス (192.0.2.1 など) を使用してロードバランサーに接続する必要があります。

`dualstack`

クライアントは、IPv4 アドレス (192.0.2.1 など) と IPv6 アドレス (たとえば、2001:0db8:85a3:0:0:8a2e:0370:7334) の両方を使用してロードバランサーに接続できます。

考慮事項

- ロードバランサーは、ターゲットグループの IP アドレスのタイプに基づいてターゲットと通信します。
- ロードバランサーのデュアルスタックモードを有効にすると、Elastic Load Balancing がロードバランサーの AAAA DNS レコードを提供します。IPv4 アドレスを使用してロードバランサーと通信するクライアントは、A DNS レコードを解決します。IPv6 アドレスを使用してロードバランサーと通信するクライアントは、AAAA DNS レコードを解決します。

- インターネットゲートウェイを経由する内部デュアルスタックロードバランサーへのアクセスがブロックされ、意図しないインターネットアクセスを防止します。ただし、IGW 以外のインターネットアクセス (ピアリング、Transit Gateway、 など) AWS Direct Connectは妨げられません AWS VPN。

`dualstack-without-public-ipv4`

クライアントは、IPv6 アドレス (2001:0db8:85a3:0:0:8a2e:0370:7334 など) を使用してロードバランサーに接続する必要があります。

考慮事項

- Application Load Balancer 認証は、ID プロバイダー (IdP) または Amazon Cognito エンドポイントに接続する場合にのみ IPv4 をサポートします。パブリック IPv4 アドレスがないと、ロードバランサーは認証プロセスを完了できず、HTTP 500 エラーが発生します。

IP アドレスタイプの詳細については、「」を参照してくださいApplication Load Balancer の IP アドレスタイプ。

Application Load Balancer リソースマップ

Application Load Balancer リソースマップは、関連するリスナー、ルール、ターゲットグループ、ターゲットなど、ロードバランサーのアーキテクチャをインタラクティブに表示します。リソースマップでは、すべてのリソース間の関係とルーティングパスも強調表示され、ロードバランサーの設定が視覚的に表示されます。

コンソールを使用して Application Load Balancer のリソースマップを表示するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[ロードバランサー] を選択します。
3. ロードバランサーを選択します。
4. リソースマップタブを選択して、ロードバランサーのリソースマップを表示します。

リソースマップコンポーネント

マップビュー

Application Load Balancer リソースマップには、概要 と異常なターゲットマップ の 2 つのビューがあります。 概要はデフォルトで選択され、ロードバランサーのすべてのリソースが表示されま

す。異常なターゲットマップビューを選択すると、異常なターゲットとそれらに関連付けられたリソースのみが表示されます。

異常なターゲットマップビューは、ヘルスチェックに失敗したターゲットのトラブルシューティングに使用できます。詳細については、「リソースマップを使用した異常なターゲットのトラブルシューティング」を参照してください。

Resource Groups

Application Load Balancer リソースマップには、リソースタイプごとに 1 つずつ、4 つのリソースグループが含まれています。リソースグループは、リスナー 、ルール 、ターゲットグループ 、ターゲット です。

リソースタイル

グループ内の各リソースには独自のタイルがあり、その特定のリソースの詳細が表示されます。

- リソースタイルにカーソルを合わせると、そのタイルと他のリソースとの関係が強調表示されます。
- リソースタイルを選択すると、そのタイルと他のリソースとの関係が強調表示され、そのリソースに関する追加の詳細が表示されます。
  - ルール条件： 各ルールの条件。
  - ターゲットグループのヘルスサマリー： 各ヘルスステータスの登録済みターゲットの数。
  - ターゲットヘルスステータス ターゲットの現在のヘルスステータスと説明。

> **Note**
> リソース詳細の表示をオフにすると、リソースマップ内の追加の詳細を非表示にできます。

- 各リソースタイルには、選択したときにそのリソースの詳細ページに移動するリンクが含まれています。
  - リスナー - リスナープロトコル：ポートを選択します。例えば、次のようになります: `HTTP:80`
  - ルール - ルールアクションを選択します。例えば、次のようになります: `Forward to target group`
  - ターゲットグループ - ターゲットグループ名を選択します。例えば、次のようになります: `my-target-group`

- ターゲット - ターゲット ID を選択します。例えば、次のようになります: `i-1234567890abcdef0`

リソースマップをエクスポートする

Export を選択すると、Application Load Balancer のリソースマップの現在のビューを PDF として エクスポートできます。

ロードバランサー接続

リクエストを処理するとき、ロードバランサーは 2 つの接続を維持します。1 つはクライアントとの 接続、もう 1 つはターゲットとの接続です。ロードバランサーとクライアント間の接続は、フロン トエンド接続とも呼ばれます。ロードバランサーとターゲット間の接続は、バックエンド接続とも呼 ばれます。

接続のアイドルタイムアウト

接続アイドルタイムアウトは、ロードバランサーが接続を閉じる前に、既存のクライアントまたは ターゲット接続を非アクティブのままにして、データの送受信を行わない期間です。

ファイルのアップロードなどの長い操作を完了させるには、各アイドルタイムアウト期間が経過する 前に少なくとも 1 バイトのデータを送信し、必要に応じてアイドルタイムアウト期間の長さを増や します。また、アプリケーションのアイドルタイムアウトは、ロードバランサーに設定されたアイド ルタイムアウトよりも大きな値に設定することをお勧めします。そうしないと、アプリケーションが ロードバランサーへの TCP 接続を正常に閉じない場合、ロードバランサーは、接続が閉じられたこ とを示すパケットを受信する前に、アプリケーションにリクエストを送信することがあります。この 場合、ロードバランサーは HTTP 502 Bad Gateway エラーをクライアントに送信します。

デフォルトでは、Elastic Load Balancing はロードバランサーのアイドルタイムアウト値を 60 秒、 つまり 1 分に設定します。別のアイドルタイムアウト値を設定するには、以下の手順を使用しま す。

コンソールを使用して接続アイドルタイムアウト値を更新するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[ロードバランサー] を選択します。
3. ロードバランサーを選択します。
4. [属性] タブで、[編集] を選択します。

5. トラフィック設定 で、接続アイドルタイムアウト の値を入力します。有効範囲は 1～4000 秒です。
6. [変更の保存] をクリックします。

を使用してアイドルタイムアウト値を更新するには AWS CLI

`idle_timeout.timeout_seconds` 属性を指定して modify-load-balancer-attributes コマンドを使用します。

HTTP クライアントのキープアライブ期間

HTTP クライアントのキープアライブ期間は、Application Load Balancer がクライアントへの永続的な HTTP 接続を維持する最大時間です。設定された HTTP クライアントのキープアライブ期間が経過すると、Application Load Balancer は 1 つのリクエストを受け入れ、接続を正常に閉じるレスポンスを返します。

ロードバランサーによって送信されるレスポンスのタイプは、クライアント接続で使用される HTTP バージョンによって異なります。HTTP 1.x を使用して接続されたクライアントの場合、ロードバランサーはフィールド を含む HTTP ヘッダーを送信しますConnection: close。HTTP/2 を使用して接続されたクライアントの場合、ロードバランサーはGOAWAYフレームを送信します。

デフォルトでは、Application Load Balancer は HTTP クライアントのキープアライブ期間値を 3600 秒、つまり 1 時間に設定します。HTTP クライアントのキープアライブ期間をオフにしたり、最小の 60 秒未満に設定したりすることはできませんが、HTTP クライアントのキープアライブ期間を最大 604,800 秒、つまり 7 日間に増やすことができます。Application Load Balancer は、クライアントへの HTTP 接続が最初に確立されたときに、HTTP クライアントのキープアライブ期間を開始します。期間は、トラフィックがない場合も実行され続け、新しい接続が確立されるまでリセットされません。

> **Note**
>
> Application Load Balancer の IP アドレスタイプをロードバランサーに切り替えるdualstack-without-public-ipv4と、すべてのアクティブな接続が完了するまで待機します。Application Load Balancer の IP アドレスタイプの切り替えにかかる時間を短縮するには、HTTP クライアントのキープアライブ期間を短くすることを検討してください。

Application Load Balancer は、最初の接続中に HTTP クライアントのキープアライブ期間を 1 回割り当てます。HTTP クライアントのキープアライブ期間を更新すると、HTTP クライアントのキープ

アライブ期間の値が異なる同時接続が発生する可能性があります。既存の接続では、最初の接続時に適用された HTTP クライアントのキープアライブ継続時間値が保持されますが、新しい接続では更新された HTTP クライアントのキープアライブ継続時間値が受信されます。

コンソールを使用してクライアントキープアライブ期間の値を更新するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[ロードバランサー] を選択します。
3. ロードバランサーを選択します。
4. [属性] タブで、[編集] を選択します。
5. トラフィック設定 で、HTTP クライアントのキープアライブ期間 の値を入力します。有効範囲は 60 ~ 604800 秒です。
6. [変更の保存] をクリックします。

を使用してクライアントキープアライブ期間の値を更新するには AWS CLI

`client_keep_alive.seconds` 属性を指定して modify-load-balancer-attributes コマンドを使用します。

クロスゾーンロードバランサー

クロスゾーン負荷分散は、Application Load Balancer のデフォルト状態でオンになっており、ロードバランサーレベルでは変更できません。詳細については、「Elastic Load Balancing ユーザーガイド」の「クロスゾーン負荷分散」を参照してください。

クロスゾーン負荷分散は、ターゲットグループレベルでオフにすることができます。詳細については、「the section called “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。

削除保護

ロードバランサーが誤って削除されるのを防ぐため、削除保護を有効にできます。デフォルトでは、ロードバランサーで削除保護が無効になっています。

ロードバランサーの削除保護を有効にした場合、ロードバランサーを削除する前に無効にする必要があります。

コンソールを使用して削除保護を有効にするには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[ロードバランサー] を選択します。
3. ロードバランサーを選択します。
4. [属性] タブで、[編集] を選択します。
5. [構成] で、[削除保護] をオンにします。
6. [変更の保存] をクリックします。

コンソールを使用して削除保護を無効にするには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[ロードバランサー] を選択します。
3. ロードバランサーを選択します。
4. [属性] タブで、[編集] を選択します。
5. [構成] ページで、[削除保護] をオンにします。
6. [変更の保存] をクリックします。

を使用して削除保護を有効または無効にするには AWS CLI

`deletion_protection.enabled` 属性を指定して modify-load-balancer-attributes コマンドを使用します。

Desync 軽減モード

Desync 軽減モードは、HTTP Desync に伴う問題からアプリケーションを保護します。ロードバランサーは、脅威レベルに基づいて各リクエストを分類します。安全なリクエストは許可し、指定した軽減モードで指定されたリスクに対しては軽減処理を行います。Desync 軽減モードの種類は、モニタリングモード、防御モード、厳密モードです。デフォルトは防御モードで、アプリケーションの可用性を維持しながら、HTTP Desync に対する永続的な軽減を提供します。厳密モードに切り替えると、アプリケーションで RFC 7230 に準拠するリクエストだけを受信できます。

http_desync_guardian ライブラリは、HTTP リクエストを分析して、HTTP Desync 攻撃を防ぎます。詳細については、「」の「HTTP Desync Guardian」を参照してください GitHub。

分類

分類は次のとおりです。

- Compliant — リクエストは RFC 7230 に準拠しており、セキュリティ上の既知の脅威はありません。
- Acceptable — リクエストは RFC 7230 に準拠していませんが、セキュリティ上の既知の脅威はありません。
- Ambiguous — リクエストは RFC 7230 に準拠しておらず、ウェブサーバーやプロキシごとに処理方法が異なる場合、リスクが生じます。
- Severe — リクエストは高いセキュリティリスクをもたらしています。ロードバランサーはリクエストをブロックし、クライアントに 400 レスポンスを提供し、クライアント接続を閉じます。

リクエストが RFC 7230 に準拠していない場合、ロードバランサーは `DesyncMitigationMode_NonCompliant_Request_Count` メトリクスを増分します。詳細については、「Application Load Balancer のメトリック」を参照してください。

各リクエストの分類は、ロードバランサーのアクセスログに含まれます。リクエストが準拠しない場合、アクセスログには分類理由コードが含まれます。詳細については、「分類の理由」を参照してください。

モード

次の表は、Application Load Balancer で、リクエストがモードと分類に基づきどのように処理されるかを示しています。

分類	モニタリングモード	防御モード	厳密モード
準拠	許可	許可	許可
Acceptable	許可	許可	ブロック
Ambiguous	許可	許可[1]	ブロック
Severe	許可	ブロック	ブロック

[1] リクエストはルーティングされますが、クライアントとターゲットの接続は閉じられます。ロードバランサーが防御モードで多数のあいまいなリクエストを受信すると、追加料金が発生する可能性

があります。これは、1 秒あたりの新しい接続数の増加が、1 時間あたりに使用されるロードバランサーキャパシティーユニット (LCU) に影響するためです。NewConnectionCount メトリクスを使用すると、モニタリングモードと防御モードで、ロードバランサーによってどのように新しい接続が確立されているかを比較できます。

コンソールを使用して Desync 軽減モードを更新するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[ロードバランサー] を選択します。
3. ロードバランサーを選択します。
4. [属性] タブで、[編集] を選択します。
5. [パケット処理] の [非同期緩和モード] で、[防御]、[厳密]、または [モニタリング] を選択します。
6. [変更の保存] をクリックします。

を使用して非同期緩和モードを更新するには AWS CLI

modify-load-balancer-attributes コマンドを、`routing.http.desync_mitigation_mode` 属性に `monitor`、`defensive`、`strictest` のいずれかを設定しながら使用します。

ホストヘッダーの保持

Preserve host header (ホストヘッダーの保持) 属性を有効にした場合、Application Load Balancer は HTTP リクエストの `Host` ヘッダーを保持し、変更を加えずにヘッダーをターゲットに送信します。Application Load Balancer が複数の `Host` ヘッダーを受信した場合、すべてが保持されます。リスナールールは最初に受信した `Host` ヘッダーにのみ適用されます。

デフォルトで、Preserve host header (ホストヘッダーの保持) 属性が有効になっていない場合、Application Load Balancer は `Host` ヘッダーを次のように変更します。

ホストヘッダーの保持が有効になっておらず、リスナーポートがデフォルト以外のポートである場合: デフォルトポート (ポート 80 または 443) を使用しない場合、クライアントによってポート番号が追加されなければ、ホストヘッダーにポート番号を追加します。例えば、リスナーポートが `8080` などのデフォルト以外のポートである場合、`Host: www.example.com` を含む HTTP リクエストの `Host` ヘッダが `Host: www.example.com:8080` に変更されます。

ホストヘッダーの保持が有効になっておらず、リスナーポートがデフォルトポート (ポート 80 または 443) の場合: デフォルトのリスナーポート (ポート 80 または 443) の場合、送信されるホストヘッ

ダーにポート番号を追加しません。受信したホストヘッダーに含まれていたポート番号はすべて削除されます。

次の表では、Application Load Balancers がリスナーポートに基づいて HTTP リクエストのホストヘッダーを処理する方法の例をさらに示します。

リスナーポート	リクエストの例	リクエストのホストヘッダー	ホストヘッダーの保持が無効です (デフォルト動作)	ホストヘッダーの保持が有効です
リクエストはデフォルトの HTTP/HTTPS リスナーで送信されます。	`GET / index.html HTTP/1.1 Host: example.com`	example.com	example.com	example.com
リクエストはデフォルトの HTTP リスナーで送信され、ホストヘッダーにはポート (80 や 443 など) があります。	`GET / index.html HTTP/1.1 Host: example.com:80`	example.com:80	example.com	example.com:80
リクエストには絶対パスが含まれます。	`GET https://dns_name/index.html HTTP/1.1 Host: example.com`	example.com	dns_name	example.com
リクエストがデフォルト以外のリスナーポート (8080 など) で送信される	`GET / index.html HTTP/1.1 Host: example.com`	example.com	example.com:8080	example.com

リスナーポート	リクエストの例	リクエストのホストヘッダー	ホストヘッダーの保持が無効です (デフォルト動作)	ホストヘッダーの保持が有効です
リクエストはデフォルト以外のリスナーポートで送信され、ホストヘッダーにはポート (例えば、8080) が含まれます。	`GET / index.html HTTP/1.1 Host: example.com:8080`	example.com:8080	example.com:8080	example.com:8080

コンソールを使用してホストヘッダーの保存を有効にするには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[ロードバランサー] を選択します。
3. ロードバランサーを選択します。
4. [属性] タブで、[編集] を選択します。
5. [パケット処理] で [ホストヘッダーを保存] をオンにします。
6. [変更の保存] をクリックします。

を使用してホストヘッダーの保存を有効にするには AWS CLI

modify-load-balancer-attributes コマンドを使用します。この場合、`routing.http.preserve_host_header.enabled` 属性は `true` に設定します。

Application Load Balancer と AWS WAF

Application Load Balancer AWS WAF で を使用すると、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の Working with web ACLs を参照してください。

デフォルトでは、ロードバランサーが からレスポンスを取得できない場合 AWS WAF、HTTP 500 エラーが返され、リクエストは転送されません。に接続できない場合でも、ロードバランサーがター

ゲットにリクエストを転送する必要がある場合は AWS WAF、 AWS WAF 統合を有効にできます。ロードバランサーが と統合されているかどうかを確認するには AWS WAF、 でロードバランサー AWS Management Console を選択し、統合サービスタブを選択します。

事前定義されたウェブ ACLs

AWS WAF 統合を有効にするときに、事前定義されたルールを使用して新しいウェブ ACL を自動的に作成することを選択できます。事前定義されたウェブ ACL には、最も一般的なセキュリティ脅威に対する保護を提供する 3 つの AWS マネージドルールが含まれています。

- `AWSManagedRulesAmazonIpReputationList` - Amazon IP 評価リストのルールグループは、ボットやその他の脅威に通常関連付けられている IP アドレスをブロックします。詳細については、「 AWS WAF デベロッパーガイド」の「Amazon IP 評価リストマネージドルールグループ」を参照してください。
- `AWSManagedRulesCommonRuleSet` - コアルールセット (DCR) ルールグループは、OWTAK Top 10 で説明されている高リスクで一般的に発生する脆弱性の一部を含む、さまざまな脆弱性の悪用に対する保護を提供します。詳細については、「 AWS WAF デベロッパーガイド」の「コアルールセット (CRS) マネージドルールグループ」を参照してください。
- `AWSManagedRulesKnownBadInputsRuleSet` - 既知の不正な入力ルールグループは、無効であることがわかっており、脆弱性の悪用または検出に関連するリクエストパターンをブロックします。詳細については、「 AWS WAF デベロッパーガイド」の「既知の不正な入力マネージドルールグループ」を参照してください。

コンソール AWS WAF を使用して を有効にするには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[ロードバランサー] を選択します。
3. ロードバランサーを選択します。
4. 統合 タブで、AWS ウェブアプリケーションファイアウォール (WAF) を展開し、WAF ウェブ ACL の関連付け を選択します。
5. ウェブ ACL で、事前定義されたウェブ ACL の自動作成 を選択するか、既存のウェブ ACL を選択します。
6. ルールアクション で、ブロック またはカウント を選択します。
7. [確認] を選択します。

を使用して AWS WAF フェイルオープンを有効にするには AWS CLI

modify-load-balancer-attributes コマンドを使用します。この場合、`waf.fail_open.enabled` 属性は `true` に設定します。

Application Load Balancer の作成

ロードバランサーはクライアントからリクエストを受け取り、ターゲットグループのターゲット間でリクエストを分散します。

開始する前に、ターゲットが使用する各ゾーンで少なくとも 1 つのパブリックサブネットを持つ仮想プライベートクラウド (VPC) があることを確認します。詳細については、「the section called "ロードバランサーのサブネット"」を参照してください。

を使用してロードバランサーを作成するには、 AWS CLI「」を参照してくださいチュートリアル: AWS CLIを使用した Application Load Balancer の作成。

を使用してロードバランサーを作成するには AWS Management Console、次のタスクを実行します。

タスク

- ステップ 1: ターゲットグループの設定
- ステップ 2: ターゲットの登録
- ステップ 3: ロードバランサーとリスナーの設定
- ステップ 4: ロードバランサーのテスト

ステップ 1: ターゲットグループの設定

ターゲットグループを設定すると、EC2 インスタンスなどのターゲットを登録できます。このステップで設定するターゲットグループは、ロードバランサーを設定するときに、リスナールールでターゲットグループとして使用されます。詳細については、「Application Load Balancer のターゲットグループ」を参照してください。

コンソールを使用してターゲットグループを設定するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[ターゲットグループ] を選択します。
3. [ターゲットグループの作成] を選択します。
4. [基本的な設定] セクションで、以下のパラメータを設定します。

a. [ターゲットタイプの選択] で、[インスタンス] を選択してインスタンス ID でターゲットを指定するか、[IP アドレス] を選択して IP アドレスのみでターゲットを指定します。ターゲットタイプが [Lambda 関数] の場合は、[ヘルスチェック] セクションの [有効] を選択してヘルスチェックを有効にできます。

b. [ターゲットグループ名] に、ターゲットグループの名前を入力します。

c. [ポート] と [プロトコル] を必要に応じて設定します。

d. ターゲットタイプが [インスタンス] または [IP アドレス] の場合は、[IPv4] または [IPv6] を [IP アドレスタイプ] として選択します。そうでない場合は、次のステップに進みます。

このターゲットグループに含めることができるのは、選択した IP アドレスタイプを持つターゲットのみであることに注意してください。ターゲットグループの作成後に IP アドレスタイプを変更することはできません。

e. [VPC] では、ターゲットグループに含めるターゲットがある Virtual Private Cloud (VPC) を選択します。

f. [Protocol version] (プロトコルバージョン) で、リクエストプロトコルが HTTP/1.1 または HTTP/2 の場合は [HTTP1] を選択し、リクエストプロトコルが HTTP/2 または gRPC の場合は [HTTP2] を選択し、リクエストプロトコルが gRPC の場合は [gRPC] を選択します。

5. [ヘルスチェック] セクションで、必要に応じてデフォルト設定を変更します。[ヘルスチェックの詳細設定] で、ヘルスチェックポート、カウント、タイムアウト、インターバルを選択し、成功コードを指定します。ヘルスチェックが [異常なしきい値] のカウントを連続して超えると、ロードバランサーはターゲットを停止中の状態にします。ヘルスチェックが [正常なしきい値] のカウントを連続して超えると、ロードバランサーはターゲットを稼働状態に戻します。詳細については、「ターゲットグループのヘルスチェック」を参照してください。

6. (オプション) 次のように 1 つ以上のタグを追加します。

a. [Tags (タグ)] セクションを展開します。

b. [Add tag] を選択します。

c. タグキーとタグ値を入力します。使用できる文字は、文字、スペース、数字 (UTF-8)、および特殊文字 (+-=. _:/@) です。ただし、先頭または末尾にはスペースを使用しないでください。タグ値は大文字と小文字が区別されます。

7. [Next] を選択します。

ステップ 2: ターゲットの登録

EC2 インスタンス、IP アドレス、または Lambda 関数をターゲットグループのターゲットとして登録できます。これは、ロードバランサーを作成するためのオプションのステップです。ただし、ターゲットを登録して、ロードバランサーがトラフィックをターゲットにルーティングするようにする必要があります。

1. [ターゲットの登録] ページで、次のように 1 つ以上のターゲットを追加します。
   - ターゲットタイプがインスタンスである場合は、1 つ以上のインスタンスを選択し、1 つ以上のポートを入力して、[保留中として以下を含める] を選択します。
   - ターゲットタイプが [IP addresses] (IP アドレス) の場合は、以下を実行してください。
     a. リストからネットワーク [VPC] を選択、または [Other private IP addresses] (その他のプライベート IP アドレス) を選択します。
     b. IP アドレスを手動で入力する、またはインスタンスの詳細を使用して IP アドレスを検索します。IP アドレスは、一度に 5 個まで入力できます。
     c. 指定された IP アドレスにトラフィックをルーティングするためのポートを入力します。
     d. [Include as pending below] (保留中として以下を含める) をクリックします。
   - ターゲットタイプが Lambda なら、Lambda 関数を選択するか、Lambda 関数 ARN を入力して、以下で保留として含めます。
2. [ターゲットグループの作成] を選択します。

ステップ 3: ロードバランサーとリスナーの設定

Application Load Balancer を作成するには、まず、名前、スキーム、IP アドレスのタイプなど、ロードバランサーの基本設定情報を指定する必要があります。次に、ネットワークに関する情報と 1 つ以上のリスナーを指定します。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を使用して設定します。サポートされるプロトコルとポートの詳細については、「リスナーの設定」を参照してください。

コンソールを使用してロードバランサーとリスナーを設定するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[ロードバランサー] を選択します。
3. [Create Load Balancer] を選択します。

4. [Application Load Balancer] で [作成] を選択します。
5. 基本的な設定

   a. [ロードバランサー名] に、ロードバランサーの名前を入力します。たとえば、**my-alb** と指定します。Application Load Balancer の名前は、リージョンの Application Load Balancer と Network Load Balancer のセット内で一意である必要があります。名前は最大 32 文字で、英数字とハイフンのみを使用できます。先頭および末尾にハイフンまたは `internal-` を使用することはできません。Application Load Balancer の名前は、作成後に変更することはできません。

   b. [スキーム] で、[インターネット向け] または [内部] を選択します。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ターゲットにルーティングします。内部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ターゲットにリクエストをルーティングします。

   c. IP アドレスタイプ で、IPv4、デュアルスタック 、またはパブリック IPv4 のないデュアルスタックを選択します。クライアントが IPv4 アドレスを使用してロードバランサーと通信する場合は、IPv4 を選択します。クライアントが IPv4 および IPv6 アドレスの両方を使用してロードバランサーと通信する場合、デュアルスタック を使用します。クライアントが IPv6 アドレスのみを使用してロードバランサーと通信する場合は、パブリック IPv4 を使用しないデュアルスタックを選択します。 IPv6

6. ネットワークマッピング

   a. [VPC] では、EC2 インスタンスで使用したのと同じ VPC を選択します。[スキーム] で [インターネット向け] を選択した場合は、インターネットゲートウェイを持つ VPC だけを選択できます。

   b. マッピングでは、次のようにサブネットを選択してロードバランサーのゾーンを有効にします。

      - 2 つ以上のアベイラビリティーゾーンからのサブネット
      - 1 つ以上の Local Zones からのサブネット
      - 1 つの Outpost サブネット

      詳細については、「the section called “ロードバランサーのサブネット”」を参照してください。

      内部ロードバランサーの場合、IPv4 アドレスおよび IPv6 アドレスは、サブネット CIDR から割り当てられます。

ロードバランサーのデュアルスタックモードを有効にした場合は、IPv4 および IPv6 CIDR ブロックが関連付けられているサブネットを選択します。

7. [セキュリティグループ] で、既存のセキュリティグループを選択することも、新しいセキュリティグループを作成することもできます。

   ロードバランサーのセキュリティグループは、リスナーポート ポートとヘルスチェックポートの両方で登録済みターゲットとの通信を許可する必要があります。コンソールは、この通信を許可するルールにより、お客様に代わってロードバランサー用のセキュリティグループを作成できます。代わりにセキュリティグループを作成して選択することもできます。詳細については、「推奨ルール」を参照してください。

   (オプション) ロードバランサー用に新しいセキュリティグループを作成するには、[新しいセキュリティグループの作成] を選択します。

8. [Listeners and routing] のデフォルトは、ポート 80 で HTTP トラフィックを受け付けるリスナーです。デフォルトのプロトコルとポートを維持するか、別のプロトコルとポートを選択できます。[デフォルトアクション] で、先ほど作成したターゲットグループを選択します。[リスナーを追加] を選択して別のリスナー (HTTPS リスナーなど) を追加できます。

9. (オプション) HTTPS リスナーを使用する場合

   [セキュリティポリシー] では、常に最新の事前定義されたセキュリティポリシーを使用することをお勧めします。

   a. [デフォルトの SSL/TLS 証明書] では、以下のオプションがあります。

      - を使用して証明書を作成またはインポートした場合は AWS Certificate Manager、「ACM から」を選択し、「証明書の選択」から証明書を選択します。
      - IAM を使用して証明書をすでにインポートしている場合は、[IAM から] を選択し、[証明書の選択] から対象の証明書を選択します。
      - インポートする証明書はあるが、リージョンで ACM を利用できない場合は、[インポート] を選択し、次に [IAM へ] を選択します。[証明書名] フィールドに証明書の名前を入力します。[証明書のプライベートキー] に、PEM エンコードされたプライベートキーファイルの内容をコピーして貼り付けます。[証明書本文] に、PEM エンコードされたパブリックキー証明書ファイルの内容をコピーして貼り付けます。自己署名証明書を使用しておらず、ブラウザが暗黙的に証明書を受け入れることが重要である場合に限り、[Certificate Chain] に、PEM エンコードされた証明書チェーンファイルの内容をコピーして貼り付けます。

b. （オプション) 相互認証を有効にするには、クライアント証明書処理で相互認証 (mTLS) を有効にします。

有効にすると、デフォルトの相互 TLS モードはパススルー になります。

Trust Store で検証 を選択した場合：

- デフォルトでは、有効期限切れのクライアント証明書を使用した接続は拒否されます。この動作を変更するには、アドバンスト mTLS 設定 を展開し、クライアント証明書の有効期限 で、期限切れのクライアント証明書を許可する を選択します。
- Trust Store で既存の信頼ストアを選択するか、新しい信頼ストア を選択します。
  - 新しい信頼ストア を選択した場合は、信頼ストア名 、S3 URI 認証局の場所 、およびオプションで S3 URI 証明書失効リストの場所 を指定します。

10. （オプション) 作成時に他の サービスをロードバランサーと統合するには、「サービス統合 で最適化」を参照してください。

- ロードバランサーAWS WAFのセキュリティ保護を、既存のウェブ ACL または自動的に作成されたウェブ ACL に含めることを選択できます。作成後、ウェブ ACLs はAWS WAF コンソール で管理できます。詳細については、「 デベロッパーガイド」の「ウェブ ACL と AWS リソースの関連付けまたは関連付け解除」を参照してください。 AWS WAF
- に アクセラレーターAWS Global Acceleratorを作成し、ロードバランサーを アクセラレーターに関連付けるように選択できます。アクセラレーター名には、a～z、A～Z、0～9、. (ピリオド）、- (ハイフン) の文字を使用できます。アクセラレーターを作成したら、AWS Global Accelerator コンソール で管理できます。詳細については、「 デベロッパーガイド」の「ロードバランサーの作成時にアクセラレーターを追加する」を参照してください。 AWS Global Accelerator

11. タグ付けして作成

a. (オプション) タグを追加して、ロードバランサーを分類します。タグキーは、各ロードバランサーで一意である必要があります。使用できる文字は、文字、スペース、数字 (UTF-8)、および特殊文字 (+-=. _:/@) です。ただし、先頭または末尾にはスペースを使用しないでください。タグ値は大文字と小文字が区別されます。

b. 設定を確認し、[ロードバランサーの作成] を選択します。作成時に、ロードバランサーにいくつかのデフォルト属性が適用されます。ロードバランサーの作成後に、それらを表示および編集できます。詳細については、「ロードバランサーの属性」を参照してください。

ステップ 4: ロードバランサーのテスト

ロードバランサーを作成したら、EC2 インスタンスが最初のヘルスチェックに合格することを検証できます。ロードバランサーが EC2 インスタンスにトラフィックを送信するかどうかを検証できます。ロードバランサーを削除するには、Application Load Balancer の削除 を参照してください。

ロードバランサーをテストするには

1. ロードバランサーが作成されたら、[Close] を選択します。
2. ナビゲーションペインで、[ターゲットグループ] を選択します。
3. 新しく作成したターゲットグループを選択します。
4. [Targets] を選択して、インスタンスの準備ができていることを確認します。インスタンスのステータスが `initial` である場合、通常、インスタンスがまだ登録の進行中であることが原因です。このステータスは、インスタンスが正常と見なされるのに必要なヘルスチェックの最小数に合格しなかったことを示すこともできます。少なくとも 1 つのインスタンスのステータスが正常であれば、ロードバランサーをテストできます。詳細については、「ターゲットヘルスステータス」を参照してください。
5. ナビゲーションペインで、[ロードバランサー] を選択します。
6. 新しく作成したロードバランサーを選択します。
7. 説明 を選択し、インターネット向けまたは内部ロードバランサーの DNS 名をコピーします (例: my-load-balancer-1234567890abcdef.elb.us-east-2.amazonaws.com)。

   - インターネット向けのロードバランサーの場合、インターネットに接続されたウェブブラウザのアドレスフィールドに DNS 名を貼り付けます。
   - 内部ロードバランサーの場合、VPC へのプライベート接続を持つウェブブラウザのアドレスフィールドに DNS 名を貼り付けます。

   すべてが正しく設定されている場合は、ブラウザにサーバーのデフォルトページが表示されます。
8. ウェブページが表示されない場合は、追加の設定ヘルプとトラブルシューティングステップについて、次のドキュメントを参照してください。

   - DNS 関連の問題については、「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。
   - Load Balancer 関連の問題については、「Application Load Balancer のトラブルシューティング」を参照してください。

Application Load Balancer のアベイラビリティーゾーン

アベイラビリティーゾーンは、ロードバランサーに対していつでも有効または無効にできます。アベイラビリティーゾーンを有効にしたら、ロードバランサーはこれらのアベイラビリティーゾーン内の登録済みターゲットにリクエストをルーティングするようになります。有効な各アベイラビリティーゾーンに少なくとも 1 つの登録済みターゲットがあるようにする場合、ロードバランサーが最も効果的です。

アベイラビリティーゾーンを無効にすると、そのアベイラビリティーゾーン内のターゲットはロードバランサーに登録されたままですが、ロードバランサーはリクエストをターゲットにルーティングしなくなります。

コンソールを使用してアベイラビリティーゾーンを更新するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[ロードバランサー] を選択します。
3. ロードバランサーを選択します。
4. [Network mapping] (ネットワークマッピング) タブで、[Edit subnets] (サブネットの編集) を選択します。
5. アベイラビリティーゾーンを有効にするには、そのチェックボックスを選択し、サブネットを 1 つ選択します。使用可能なサブネットが 1 つしかない場合は、それが選択されます。
6. 有効なアベイラビリティーゾーンのサブネットを変更するには、リストから他のサブネットのいずれかを選択します。
7. アベイラビリティゾーンを無効にするには、そのチェックボックスをオフにします。
8. [変更の保存] をクリックします。

を使用してアベイラビリティーゾーンを更新するには AWS CLI

set-subnets コマンドを使用します。

Application Load Balancer のセキュリティグループ

Application Load Balancer のセキュリティグループは、ロードバランサーへのインバウンド/アウトバウンドのトラフィックを制御します。ロードバランサーが、リスナーポートとヘルスチェックポートの両方で、登録済みターゲットと通信できることを確認する必要があります。ロードバランサーにリスナーを追加するか、リクエストをルーティングするためにロードバランサーにより使用される

ターゲットグループのヘルスチェックポートを更新する場合は必ず、ロードバランサーに関連付けられたセキュリティグループが新しいポートで両方向のトラフィックを許可することを確認する必要があります。許可しない場合、現在関連付けられているセキュリティグループのルールを編集するか、別のセキュリティグループをロードバランサーに関連付けることができます。許可するポートとプロトコルを選択することができます。たとえば、ロードバランサーが ping リクエストに応答できるよう、Internet Control Message Protocol (ICMP) 接続を開くことができます (ただし、ping リクエストは登録済みインスタンスに転送されません)。

推奨ルール

インターネット向けロードバランサーには、次のルールが推奨されます。

Inbound		
Source	Port Range	Comment
0.0.0.0/0	####	ロードバランサーのリスナーポートですべてのインバウンドトラフィックを許可する

Outbound		
Destination	Port Range	Comment
################	##########	インスタンスリスナーポートでインスタンスへのアウトバウンドトラフィックを許可する
################	#######	ヘルスチェックポートでインスタンスへのアウトバウンドトラフィックを許可する

内部ロードバランサーには、次のルールが推奨されます。

Inbound		
Source	Port Range	Comment

VPC CIDR	*####*	ロードバランサーのリスナーポートで VPC CIDR からのインバウンドトラフィックを許可する

Outbound

Destination	Port Range	Comment
################	*##########*	インスタンスリスナーポートでインスタンスへのアウトバウンドトラフィックを許可する
################	*#######*	ヘルスチェックポートでインスタンスへのアウトバウンドトラフィックを許可する

Network Load Balancer のターゲットとして使用される Application Load Balancer には、以下のルールが推奨されます。

Inbound

Source	Port Range	Comment
###### IP ####/CIDR	*alb ####*	ロードバランサーのリスナーポートでインバウンドクライアントトラフィックを許可する
VPC CIDR	*alb ####*	ロードバランサーリスナーポート AWS PrivateLink で 経由でインバウンドクライアントトラフィックを許可する
VPC CIDR	*alb ####*	Network Load Balancer からのインバウンドヘルスチェックトラフィックを許可する

Outbound

Destination	Port Range	Comment
################	##########	インスタンスリスナーポートでインスタンスへのアウトバウンドトラフィックを許可する
################	#######	ヘルスチェックポートでインスタンスへのアウトバウンドトラフィックを許可する

Application Load Balancer のセキュリティグループは、接続追跡を使用して Network Load Balancer からのトラフィックに関する情報を追跡することに注意してください。これは、Application Load Balancer に設定されたセキュリティグループルールを問わず実行されます。Amazon EC2 接続の追跡の詳細については、「Amazon EC2 ユーザーガイド」の「セキュリティグループ接続の追跡」を参照してください。 Amazon EC2

ターゲットがロードバランサーからのみトラフィックを受信できるようにするには、ターゲットに関連付けられたセキュリティグループがロードバランサーからのみトラフィックを受け入れるように制限します。これは、ターゲットのセキュリティグループの進入ルールでロードバランサーのセキュリティグループをソースとして設定することで実現できます。

また、パス MTU 検出をサポートするため、インバウンド ICMP トラフィックを許可することをお勧めします。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。 Amazon EC2

関連付けられたセキュリティグループの更新

ロードバランサーに関連付けられたセキュリティグループは、いつでも更新できます。

コンソールを使用してセキュリティグループの更新するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[ロードバランサー] を選択します。
3. ロードバランサーを選択します。
4. [セキュリティ] タブで、[編集] を選択します。

5. セキュリティグループをロードバランサーに関連付けるには、そのセキュリティグループを選択します。セキュリティグループの関連付けを削除するには、セキュリティグループの [X] アイコンを選択します。
6. [変更の保存] をクリックします。

を使用してセキュリティグループを更新するには AWS CLI

set-security-groups コマンドを使用します。

Application Load Balancer の IP アドレスタイプ

Application Load Balancer は、クライアントが IPv4 アドレスのみを使用してロードバランサーと通信できるように設定する、または IPv4 アドレスと IPv6 アドレスの両方 (デュアルスタック) を使用してロードバランサーと通信できるように設定することができます。ロードバランサーは、ターゲットグループの IP アドレスのタイプに基づいてターゲットと通信します。詳細については、「IP アドレスタイプ」を参照してください。

デュアルスタックの要件

- ロードバランサーの作成時に IP アドレスの種類を設定し、いつでも更新できます。
- ロードバランサーに指定する Virtual Private Cloud (VPC) とサブネットには、IPv6 CIDR ブロックが関連付けられている必要があります。詳細については、Amazon EC2 ユーザーガイドの IPv6 アドレスを参照してください。
- ロードバランサーサブネットのルートテーブルは、IPv6 トラフィックをルーティングする必要があります。
- ロードバランサーのセキュリティグループは、IPv6 トラフィックを許可する必要があります。
- ロードバランサーサブネットのネットワーク ACL は、IPv6 トラフィックを許可する必要があります。

作成時に IP アドレスの種類を設定するには

??? の説明に従って設定を行います。

IP アドレスを更新するには、コンソールを使用して入力します。

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[ロードバランサー] を選択します。

3. ロードバランサーを選択します。
4. [ネットワークマッピング] タブで、[IP アドレスタイプの編集] を選択します。
5. IP アドレスタイプ では、IPv4 アドレスのみをサポートするには IPv4、IPv4 アドレスと IPv6 アドレスの両方をサポートするにはデュアルスタック、IPv6 アドレスIPv6のみをサポートするにはパブリック IPv4 を使用しない場合はデュアルスタックを選択します。
6. [変更の保存] をクリックします。

を使用して IP アドレスタイプを更新するには AWS CLI

set-ip-address-type コマンドを使用します。

Application Load Balancer にタグを付ける

タグを使用すると、ロードバランサーを目的、所有者、環境などさまざまな方法で分類することができます。

各ロードバランサーに対して複数のタグを追加できます。すでにロードバランサーに関連付けられているキーを持つタグを追加すると、そのキーの値が更新されます。

タグが不要になったら、ロードバランサーからタグを削除できます。

制限事項

- リソースあたりのタグの最大数 – 50
- キーの最大長 – 127 文字 (Unicode)
- 値の最大長 – 255 文字 (Unicode)
- タグのキーと値は大文字と小文字が区別されます。使用できる文字は、UTF-8 で表現できる文字、スペース、および数字と、特殊文字 (+、-、=、.、_、:、/、@) です。ただし、先頭または末尾にはスペースを使用しないでください。
- タグ名または値に aws: プレフィックスを使用しないでください。このプレフィックスは AWS 使用のために予約されています。このプレフィックスが含まれるタグの名前または値は編集または削除できません。このプレフィックスを持つタグは、リソースあたりのタグ数の制限時には計算されません。

コンソールを使用してロードバランサーのタグを更新するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。

2. ナビゲーションペインで、[ロードバランサー] を選択します。
3. ロードバランサーを選択します。
4. [タグ] タブで、[タグの管理] を選択し、次の 1 つ以上の操作を行います。
   a. タグを更新するには、[キー] と [ボールト] の値を編集します。
   b. 新しいタグを追加するには、[タグの作成] を選択し、次に [キー] と [値] に値を入力します。
   c. タグを削除するには、タグの横にある [Remove] (削除) ボタンを選択します。
5. タグの更新を完了したら、[変更内容の保存] を選択します。

を使用してロードバランサーのタグを更新するには AWS CLI

add-tags コマンドと remove-tags コマンドを使用します。

Application Load Balancer の削除

ロードバランサーが利用可能になると、ロードバランサーの実行時間に応じて 1 時間ごと、または 1 時間未満の時間について課金されます。不要になったロードバランサーは削除できます。ロードバランサーが削除されると、ロードバランサーの課金も停止されます。

削除保護が有効になった場合、ロードバランサーを削除することはできません。詳細については、「削除保護」を参照してください。

ロードバランサーを削除しても、登録済みターゲットには影響を与えない点に注意してください。たとえば、EC2 インスタンスは実行を続け、ターゲットグループに登録されたままです。ターゲットグループを削除するには、「ターゲットグループの削除」を参照してください。

コンソールを使用してロードバランサーを削除するには

1. ロードバランサーをポイントするドメインの DNS レコードが存在する場合は、新しい場所にポイントして DNS の変更が有効になってから、ロードバランサーを削除します。

   例：

   - 有効期限 (TTL) が 300 秒の CNAME レコードの場合は、少なくとも 300 秒待ってから次のステップに進みます。
   - Route 53 エイリアス (A) レコードの場合は、少なくとも 60 秒間待機します。

- Route 53 を使用している場合、レコードに対する変更が世界中のすべての Route 53 ネームサーバーに反映されるまで 60 秒かかります。更新の対象となるレコードの TTL 値には、この時間を加算します。

2. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
3. ナビゲーションペインで、[ロードバランサー] を選択します。
4. ロードバランサーを選択して [アクション]、[ロードバランサーの削除] を選択します。
5. 確認を求められたら、**`confirm`**と入力し、[削除] を選択します。

を使用してロードバランサーを削除するには AWS CLI

delete-load-balancer コマンドを使用します。

ゾーンシフト

ゾーンシフトは Amazon Route 53 Application Recovery Controller (Route 53 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。Route 53 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。詳細については、「Amazon Route 53 Application Recovery Controller デベロッパーガイド」の「How a zonal shift works: health checks and zonal IP addresses」(ゾーンシフトの仕組み: ヘルスチェックとゾーン IP アドレス) を参照してください。

ゾーンシフトは、クロスゾーン負荷分散がオフになっている Application Load Balancer と Network Load Balancer でのみサポートされます。クロスゾーンロードバランサーをオンにすると、ゾーンシフトを開始できなくなります。詳細については、「Amazon Route 53 Application Recovery Controller Developer Guide」の「Resources supported for zonal shifts」(ゾーンシフトでサポートされるリソース) 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。

- ゾーンシフトでは、クロスゾーンロードバランサーはサポートされていません。この機能を使用するには、クロスゾーンロードバランシングをオフにする必要があります。

- Application Load Balancer を AWS Global Accelerator でアクセラレータエンドポイントとして使用する場合、ゾーンシフトはサポートされません。
- 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する場合、ゾーンロードバランサーの IP アドレスを DNS から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。
- Application Load Balancer が Network Load Balancer のターゲットである場合は、常に Network Load Balancer からゾーンシフトを開始します。Application Load Balancer からゾーンシフトを開始すると、Network Load Balancer はシフトを認識せず、引き続き Application Load Balancer にトラフィックを送信します。

詳細については、「Amazon Route 53 Application Recovery Controller Developer Guide」の「Best practices with Route 53 ARC zonal shifts」(Route 53 ARC 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。

ゾーンシフトを開始する

この手順のステップでは、Amazon EC2 コンソールでゾーンシフトを開始する方法について説明します。Route 53 ARC コンソールを使用してゾーンシフトを開始する手順については、「Amazon Route 53 Application Recovery Controller Developer Guide」の「Starting a zonal shift」(ゾーンシフトの開始) を参照してください。

コンソールを使用してゾーンシフトを開始するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
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. ロードバランサー名を選択します。
4. [Integrations] (統合) タブの [Route 53 Application Recovery Controller] (Route 53 Application Recovery Controller) で、[Start zonal shift] (ゾーンシフトの開始) を選択します。
5. トラフィックを移動させたいアベイラビリティーゾーンを選択します。
6. ゾーンシフトの有効期限を選択または入力します。ゾーンシフトは、最初は 1 分から最大 3 日 (72 時間) まで設定できます。

すべてのゾーンシフトは一時的なものです。有効期限を設定する必要がありますが、アクティブなシフトを後で更新して有効期限を設定できます。

7. コメントを入力します。必要に応じて、後でゾーンシフトを更新してコメントを編集できます。
8. このチェックボックスを選択して、ゾーンシフトを開始すると、トラフィックがアベイラビリティーゾーンからシフトされてアプリケーションの容量が減少することを確認します。
9. [Start] (開始) を選択します。

AWS CLI を使用してゾーンシフトを開始するには

プログラムによるゾーンシフトの操作については、「Zonal Shift API Reference Guide」(ゾーンシフト API リファレンスガイド) を参照してください。

ゾーンシフトの更新

この手順のステップでは、Amazon EC2 コンソールでゾーンシフトを更新する方法について説明します。Amazon Route 53 Application Recovery Controller コンソールを使用してゾーンシフトを更新する手順については、「Amazon Route 53 Application Recovery Controller Developer Guide」の「Update a zonal shift」(ゾーンシフトの更新) を参照してください。

コンソールを使用してゾーンシフトを更新するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
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. アクティブなゾーンシフトを持つロードバランサー名を選択します。
4. [Integrations] (統合) タブの [Route 53 Application Recovery Controller] で、[Update zonal shift] (ゾーンシフトの更新) を選択します。

   これにより、Route 53 ARC コンソールが開き、更新が続行されます。
5. [Set zonal shift expiration time] (ゾーンシフトの有効期限の設定) で、オプションで有効期限を選択または入力します。
6. [Comment] (コメント) には、必要に応じて既存のコメントを編集するか、新しいコメントを入力します。
7. [更新] を選択します。

AWS CLI を使用してゾーンシフトを更新するには

プログラムによるゾーンシフトの操作については、「Zonal Shift API Reference Guide」(ゾーンシフト API リファレンスガイド) を参照してください。

ゾーンシフトのキャンセル

この手順のステップでは、Amazon EC2 コンソールでゾーンシフトをキャンセルする方法について説明します。Amazon Route 53 Application Recovery Controller コンソールを使用してゾーンシフトをキャンセルする手順については、「Amazon Route 53 Application Recovery Controller Developer Guide」の「Cancel a zonal shift」(ゾーンシフトのキャンセル) を参照してください。

コンソールを使用してゾーンシフトをキャンセルするには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
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. アクティブなゾーンシフトを持つロードバランサー名を選択します。
4. [Integrations] (統合) タブの [Route 53 Application Recovery Controller] で、[Cancel zonal shift] (ゾーンシフトのキャンセル) を選択します。

   これにより、Route 53 ARC コンソールが開き、キャンセルが続行されます。
5. [Cancel zonal shift] (ゾーンシフトをキャンセル) を選択します。
6. ダイアログボックスで、[Confirm] (確認) を選択します。

AWS CLI を使用してゾーンシフトをキャンルするには

プログラムによるゾーンシフトの操作については、「Zonal Shift API Reference Guide」(ゾーンシフト API リファレンスガイド) を参照してください。

Application Load Balancer のリスナー

リスナーとは、設定したプロトコルとポートを使用して接続リクエストをチェックするプロセスです。Application Load Balancer の使用を開始する前に、最低 1 つのリスナーを追加する必要があります。ロードバランサーにリスナーがない場合、クライアントからのトラフィックを受信できません。リスナーに対して定義したルールにより、ロードバランサーが EC2 インスタンスなど登録したターゲットにリクエストをルーティングする方法が決まります。

内容

リスナーの設定

リスナーは次のポートとプロトコルをサポートします。

- プロトコル: HTTP、HTTPS
- ポート: 1 ~ 65535

アプリケーションがビジネスロジックに集中できるように、HTTPS リスナーを使用して、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ができます。リスナープロトコルが HTTPS の場合

は、リスナーに SSL サーバー証明書を少なくとも 1 つデプロイする必要があります。詳細については、「Application Load Balancer 用の HTTPS リスナーを作成する」を参照してください。

ターゲットがロードバランサーではなく HTTPS トラフィックを復号化する必要がある場合は、ポート 443 に TCP リスナーを使用してNetwork Load Balancer を作成できます。TCP リスナーを使用すると、ロードバランサーは暗号化されたトラフィックを復号化せずにターゲットに渡します。の詳細については、「 Network Load Balancer のユーザーガイド」を参照してください。

Application Load Balancer は、 のネイティブサポートを提供します WebSockets。HTTP 接続アップグレードを使用して、既存の HTTP/1.1 接続を WebSocket （ ws または wss) 接続にアップグレードできます。アップグレードすると、リクエストに使用される TCP 接続 (ロードバランサーとターゲット) が、ロードバランサーを介したクライアントとターゲット間の永続的な WebSocket 接続になります。HTTP WebSockets リスナーと HTTPS リスナーの両方で を使用できます。リスナーに選択したオプションは、 WebSocket 接続と HTTP トラフィックに適用されます。詳細については、「Amazon CloudFront デベロッパーガイド」の WebSocket 「プロトコルの仕組み」を参照してください。

Application Load Balancer は HTTPS リスナーで HTTP/2 のネイティブサポートを提供します。1 つの HTTP/2 コネクションで最大 128 のリクエストを並行して送信できます。プロトコルバージョンを使用して、HTTP/2 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。HTTP/2 ではフロントエンド接続を効率的に使用するため、クライアントとロードバランサー間の接続数が減少します。HTTP/2 のサーバープッシュ機能は使用できません。

詳細については、Elastic Load Balancing ユーザーガイドのルーティングのリクエストを参照してください。

リスナールール

リスナーには、必ずデフォルトアクションがあります。これはデフォルトルールとも言います。デフォルトルールは削除できず、必ず最後に実行されます。新しいルールの作成は可能で、優先度、1 つ以上のアクション、および 1 つ以上の条件を構成できます。ルールの追加や編集はいつでも行うことができます。詳細については、「ルールの編集」を参照してください。

デフォルトのルール

リスナーを作成するときは、デフォルトのルールのアクションを定義します。デフォルトのルールに条件を定義することはできません。リスナーのルールに設定された条件のいずれも満たされない場合は、デフォルトのルールのアクションが実行されます。

デフォルトのルールの例を、コンソールに表示されるとおりに次に示します。

Priority	Conditions (If)	Actions (Then)
Last (default)	*If no other rule applies*	**Forward to target group** • my-targets: 1 (100%) • Group-level stickiness: Off

ルールの優先順位

各ルールには優先順位があります。ルールは優先順位の低~高順によって評価されます。デフォルトのルールが最後に評価されます。デフォルト以外のルールは、優先順位をいつでも変更できます。デフォルトルールの優先順位は変更できません。詳細については、「ルールの優先度の更新」を参照してください。

ルールのアクション

ルールのアクションごとに、タイプ、優先度、およびアクションを実行するために必要な情報があります。詳細については、「ルールアクションタイプ」を参照してください。

ルールの条件

ルールのアクションごとにタイプと設定情報があります。ルールの条件が満たされると、アクションが実行されます。詳細については、「ルールの条件のタイプ」を参照してください。

ルールアクションタイプ

リスナールールでサポートされているアクションタイプを以下に示します。

`authenticate-cognito`

[HTTPS リスナー] Amazon Cognito を使用してユーザーを認証します。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。

`authenticate-oidc`

[HTTPS リスナー] OpenID Connect (OIDC) に準拠する ID プロバイダーを使用してユーザーを認証します。

`fixed-response`

カスタムの HTTP レスポンスを返します。詳細については、「固定レスポンスアクション」を参照してください。

`forward`

指定されたターゲットグループにリクエストを転送します。詳細については、「転送アクション」を参照してください。

`redirect`

1 つの URL から別の URL にリクエストをリダイレクトします。詳細については、「リダイレクトアクション」を参照してください。

最も優先度の低いアクションが最初に実行されます。各ルールには次のアクションのうち、厳密に 1 つを含む必要があります。`forward`、`redirect`、`fixed-response`。またはそれは最後に実行されるアクションである必要があります。

プロトコルバージョンが gRPC または HTTP/2 の場合、サポートされているアクションは `forward` アクションだけです。

固定レスポンスアクション

クライアントリクエストを破棄し、カスタムの HTTP レスポンスを返すには、`fixed-response` アクションを使用します。このアクションを使用して、2XX、4XX、または 5XX のレスポンスコードとオプションのメッセージを返すことができます。

`fixed-response` アクションが実行されると、そのアクションと、リダイレクトターゲットの URL がアクセスログに記録されます。詳細については、「アクセスログのエントリ」を参照してください。`fixed-response` アクションが正常に実行された数は、`HTTP_Fixed_Response_Count` メトリクスでレポートされます。詳細については、「Application Load Balancer のメトリック」を参照してください。

Example の固定レスポンスアクションの例 AWS CLI

ルールの作成時、または変更時にアクションを指定できます。詳細については、create-rule および modify-rule コマンドを参照してください。次のアクションは指定されたステータスコードとメッセージの本文を含む固定レスポンスを送信します。

```
[
```

```
  {
      "Type": "fixed-response",
      "FixedResponseConfig": {
          "StatusCode": "200",
          "ContentType": "text/plain",
          "MessageBody": "Hello world"
      }
  }
]
```

転送アクション

forward アクションを使用して、1 つ以上のターゲットグループにリクエストをルーティングできます。forward アクションに複数のターゲットグループを指定する場合は、ターゲットグループごとに重みを指定する必要があります。各ターゲットグループの重みは、0 ~ 999 の値です。加重ターゲットグループを持つリスナールールと一致するリクエストは、それらの重みに基づいてこれらのターゲットグループに分散されます。たとえば、ターゲットグループを 2 つ指定し、それぞれ重みが 10 の場合、各ターゲットグループはリクエストの半分を受け取ります。2 つのターゲットグループ (1 つは重みが 10 で、もう 1 つは重みが 20) を指定すると、重みが 20 のターゲットグループは他のターゲットグループの 2 倍の数のリクエストを受信します。

デフォルトでは、加重ターゲットグループ間でトラフィックを分散するようにルールを設定しても、スティッキーセッションが優先されるとは限りません。スティッキーセッションが確実に優先されるようにするには、ルールのターゲットグループの維持を有効にします。ロードバランサーは、最初に加重ターゲットグループにリクエストをルーティングするときに、選択したターゲットグループに関する情報をエンコード AWSALBTG し、Cookie を暗号化し、クライアントへのレスポンスに Cookie を含める という名前の Cookie を生成します。クライアントは、受信した Cookie を、ロードバランサーへの後続のリクエストに含める必要があります。ロードバランサーが、ターゲットグループの維持が有効になっているルールに一致するリクエストを受信して Cookie が含まれている場合、リクエストは Cookie で指定されたターゲットグループにルーティングされます。

Application Load Balancer は、URL エンコードされた Cookie 値をサポートしていません。

CORS (クロスオリジンリソース共有) リクエストの場合、一部のブラウザは維持を有効にするために SameSite=None; Secure を必要とします。この場合、Elastic Load Balancing は 2 番目の Cookie を生成します。これには AWSALBTGCORS、元の維持 Cookie と同じ情報とこのSameSite属性が含まれます。クライアントは両方の Cookie を受け取ります。

Example 1 つのターゲットグループを使用した転送アクションの例

ルールの作成時、または変更時にアクションを指定できます。詳細については、create-rule および modify-rule コマンドを参照してください。次のアクションは指定されたターゲットグループにリクエストを転送します。

```
[
  {
      "Type": "forward",
      "ForwardConfig": {
          "TargetGroups": [
              {
                  "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067"
              }
          ]
      }
  }
]
```

Example 2 つの加重ターゲットグループを使用した転送アクションの例

次のアクションは、各ターゲットグループの重みに基づいて、指定された 2 つのターゲットグループにリクエストを転送します。

```
[
  {
      "Type": "forward",
      "ForwardConfig": {
          "TargetGroups": [
              {
                  "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/blue-targets/73e2d6bc24d8a067",
                  "Weight": 10
              },
              {
                  "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/green-targets/09966783158cda59",
                  "Weight": 20
              }
          ]
      }
  }
```

```
]
```

Example 維持を有効にした転送アクションの例

複数のターゲットグループを含む転送アクションがあり、1 つ以上のターゲットグループでスティッキーセッションが有効になっている場合は、ターゲットグループの維持を有効にする必要があります。

次のアクションは、ターゲットグループの維持を有効にして、指定された 2 つのターゲットグループにリクエストを転送します。維持 Cookie を含まないリクエストは、各ターゲットグループの重みに基づいてルーティングされます。

```
[
  {
      "Type": "forward",
      "ForwardConfig": {
          "TargetGroups": [
              {
                  "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/blue-targets/73e2d6bc24d8a067",
                  "Weight": 10
              },
              {
                  "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/green-targets/09966783158cda59",
                  "Weight": 20
              }
          ],
          "TargetGroupStickinessConfig": {
              "Enabled": true,
              "DurationSeconds": 1000
          }
      }
  }
]
```

リダイレクトアクション

クライアントリクエストを別の URL にリダイレクトするには、`redirect` アクションを使用します。必要に応じて、一時的 (HTTP 302) または恒久的 (HTTP 301) としてリダイレクトを設定します。

URI のコンポーネントは次のとおりです。

```
protocol://hostname:port/path?query
```

リダイレクトのループを避けるために、次のコンポーネントを 1 つ以上変更する必要があります: protocol、hostname、port、または path。変更しないコンポーネントはすべて、元の値が保持されます。

プロトコル

プロトコル (HTTP または HTTPS)。HTTP から HTTP、HTTP から HTTPS、HTTPS から HTTPS にリダイレクトできます。HTTPS から HTTP にリダイレクトすることはできません。

hostname

ホスト名。ホスト名は大文字と小文字を区別せず、最大 128 文字の長さで、英数字、ワイルドカード (* と ?)、およびハイフン (-) で構成されます。

port

ポート (1 ~ 65535)。

パス

絶対パス (先頭は「/」で始まる)。パスは大文字と小文字が区別され、最大 128 文字の長さで、英数字、ワイルドカード (* と ?)、& (& を使用) および次の特殊文字 _-.$/~"'@:+ で構成されます。

query

クエリパラメータ 最大長は 128 文字です。

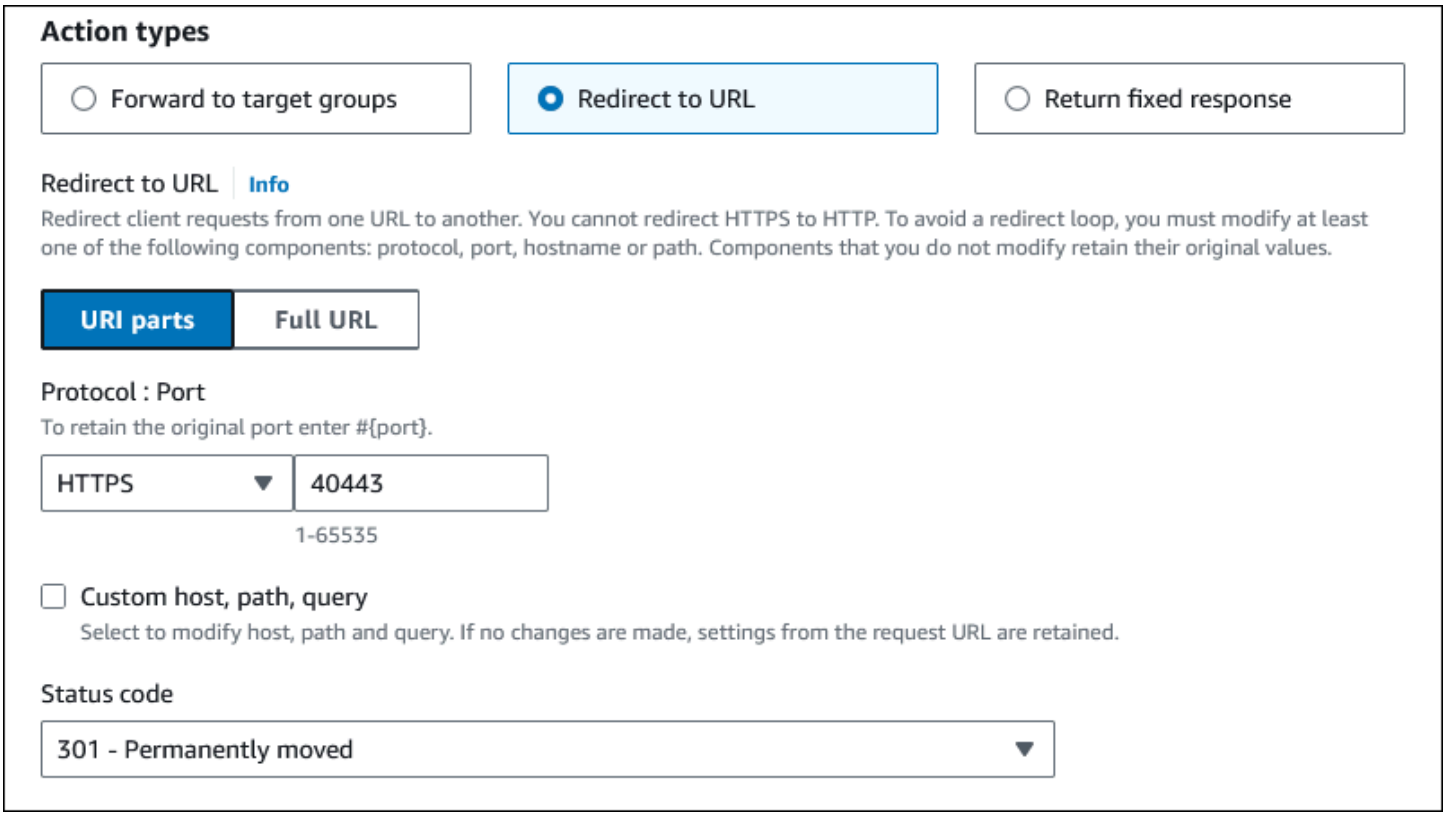
次の予約キーワードを使用して、元 URL の URI コンポーネントをターゲット URL で再利用できます。

- `#{protocol}` – プロトコルが保持されます。プロトコルとクエリコンポーネントで使用します。
- `#{host}` – ドメインが保持されます。ホスト名、パス、クエリコンポーネントで使用します。
- `#{port}` – ポートが保持されます。ポート、パス、クエリコンポーネントで使用します。
- `#{path}` – パスが保持されます。パスとクエリコンポーネントで使用します。
- `#{query}` – クエリパラメータが保持されます。クエリコンポーネントで使用します。

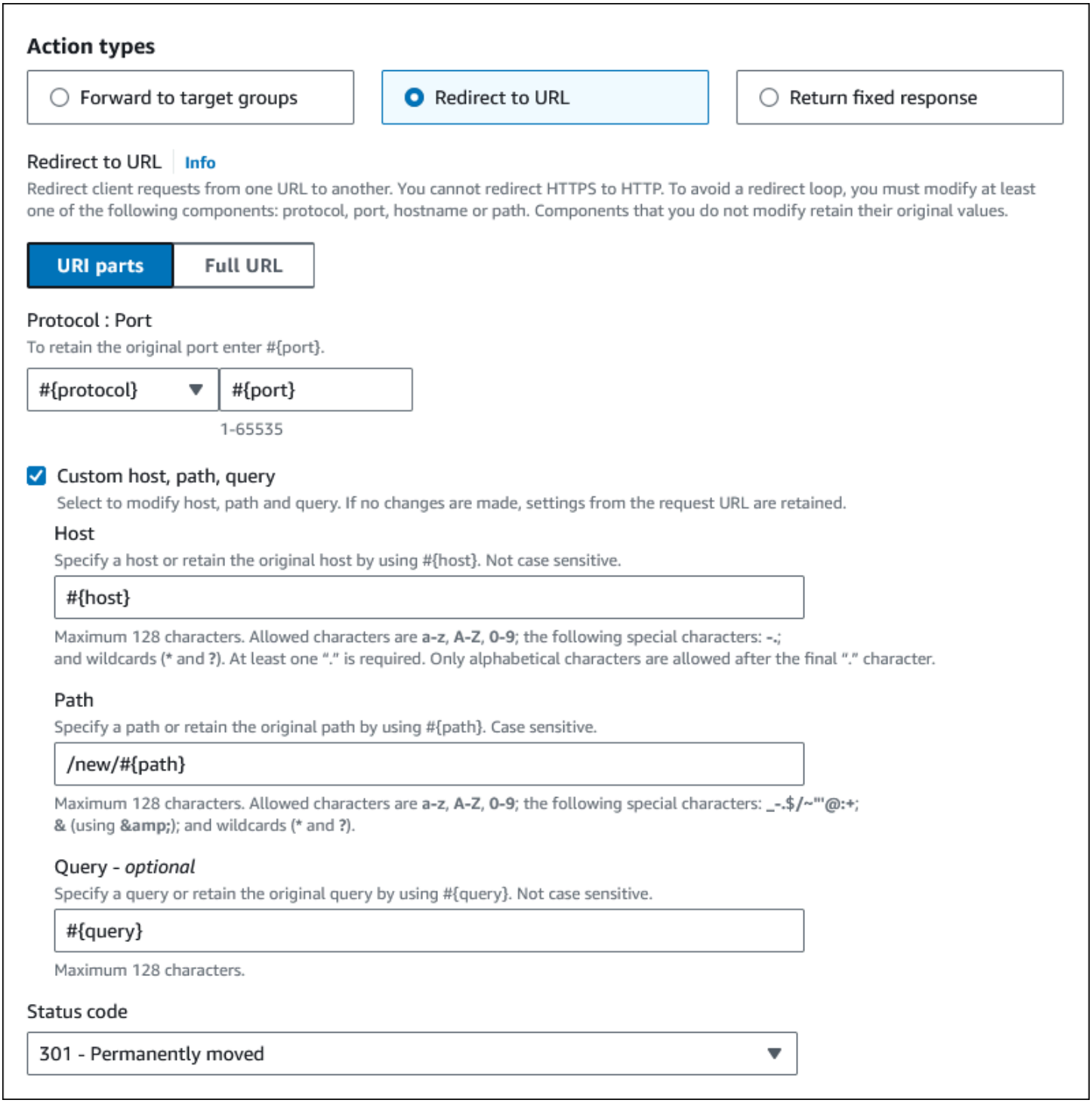
redirect アクションが実行されると、そのアクションはアクセスログに記録されます。詳細については、「アクセスログのエントリ」を参照してください。redirect アクションが正常に実行された数は、HTTP_Redirect_Count メトリクスでレポートされます。詳細については、「Application Load Balancer のメトリック」を参照してください。

Example コンソールを使用したリダイレクトアクションの例

次のルールでは、HTTPS プロトコルと指定されたポート (40443) を使用する URL にリダイレクトする永続的 URL が設定されますが、元のホスト名、パス、およびクエリパラメータは保持されます。この画面は「https://#{host}:40443/#{path}?#{query}」に相当します。



次のルールでは、元のプロトコル、ポート、ホスト名、クエリパラメータを保持する URL に恒久的ダイレクトをセットアップし、#{path} キーワードを使用して次の変更されたパスを作成します。この画面は「#{protocol}://#{host}:#{port}/new/#{path}?#{query}」に相当します。



Example のリダイレクトアクションの例 AWS CLI

ルールの作成時、または変更時にアクションを指定できます。詳細については、create-rule および modify-rule コマンドを参照してください。以下のアクションは、HTTP リクエストと同じホスト名、パス、およびクエリストリングを使用して、HTTP リクエストをポート 443 上の HTTPS リクエストにリダイレクトします。

```
[
  {
      "Type": "redirect",
      "RedirectConfig": {
          "Protocol": "HTTPS",
          "Port": "443",
          "Host": "#{host}",
          "Path": "/#{path}",
          "Query": "#{query}",
          "StatusCode": "HTTP_301"
      }
  }
]
```

ルールの条件のタイプ

サポートされているルールの条件のタイプを以下に示します。

`host-header`

各リクエストのホスト名に基づいたルーティング。詳細については、「ホストの条件」を参照してください。

`http-header`

各リクエストの HTTP ヘッダーに基づいたルーティング。詳細については、「HTTP ヘッダー条件」を参照してください。

`http-request-method`

各リクエストの HTTP リクエストメソッドに基づいたルーティング。詳細については、「HTTP リクエストメソッド条件」を参照してください。

`path-pattern`

リクエスト URL のパスパターンに基づいたルーティング。詳細については、「パスの条件」を参照してください。

`query-string`

キーと値のペアまたはクエリストリングの値に基づいたルーティング。詳細については、「クエリ文字列の条件」を参照してください。

`source-ip`

各リクエストの送信元 IP アドレスに基づいたルーティング。詳細については、「送信元 IP アドレス条件」を参照してください。

各ルールには、オプションで次の各条件を 1 つ含めることができます。host-header、http-request-method、path-pattern、および source-ip。各ルールには、オプションで次の各条件を 1 つ以上含めることもできます。http-header および query-string。

1 つの条件につき最大 3 つの一致評価を指定できます。たとえば、各 http-header 条件に対して、リクエスト内の HTTP ヘッダーの値と比較する最大 3 つの文字列を指定できます。いずれかのストリングが HTTP ヘッダーの値と一致すれば、条件は満たされます。すべての文字列が一致することを要求するには、一致評価ごとに 1 つの条件を作成します。

1 つのルールにつき最大 5 つの一致評価を指定できます。すべての文字列が一致であることを要求するには、一致評価ごとに1つの条件を作成します。

http-header、host-header、path-pattern、query-string 条件に対する一致評価にワイルドカード文字を含めることができます。1 ルールあたりのワイルドカード文字は 5 つまでです。

ルールは表示可能な ASCII 文字にのみ適用されます。制御文字 (0x00 ~ 0x1f および 0x7f) は除外されます。

デモについては、「Advanced Request Routing」を参照してください。

HTTP ヘッダー条件

HTTP ヘッダー条件を使用して、リクエストの HTTP ヘッダーに基づいてリクエストをルーティングするルールを設定できます。標準またはカスタムの HTTP ヘッダーフィールドの名前を指定できます。ヘッダー名と一致評価では大文字と小文字は区別されません。次のワイルドカード文字は比較文字列でサポートされています。* (0 個以上の文字と一致) および ? (厳密に 1 文字と一致) ワイルドカード文字はヘッダー名ではサポートされていません。

Example の HTTP ヘッダー条件の例 AWS CLI

ルールの作成時、または変更時に条件を指定できます。詳細については、create-rule および modify-rule コマンドを参照してください。以下の条件は、指定された文字列の 1 つに一致するユーザーエージェントヘッダーを使用するリクエストによって満たされます。

```
[
  {
```

```
      "Field": "http-header",
      "HttpHeaderConfig": {
          "HttpHeaderName": "User-Agent",
          "Values": ["*Chrome*", "*Safari*"]
      }
  }
]
```

HTTP リクエストメソッド条件

HTTP リクエストメソッド条件を使用して、リクエストの HTTP リクエストメソッドに基づいてリクエストをルーティングするルールを設定できます。標準またはカスタムの HTTP メソッドを指定できます。一致評価は大文字と小文字を区別します。ワイルドカード文字はサポートされていません。したがって、メソッド名は厳密な一致である必要があります。

HEAD リクエストに対する応答はキャッシュされる可能性があるため、GET リクエストと HEAD リクエストを同じ方法でルーティングすることをお勧めします。

Example の HTTP メソッド条件の例 AWS CLI

ルールの作成時、または変更時に条件を指定できます。詳細については、create-rule および modify-rule コマンドを参照してください。以下の条件は、指定されたメソッドを使用するリクエストによって満たされます。

```
[
  {
      "Field": "http-request-method",
      "HttpRequestMethodConfig": {
          "Values": ["CUSTOM-METHOD"]
      }
  }
]
```

ホストの条件

ホストの条件を使用して、ホストヘッダーのホスト名に基づいてリクエストをルーティングする (ホストベースのルーティングとも呼ばれます) ルールを定義できます。これにより、1 つのロードバランサーを使用して複数のサブドメインおよび異なるトップレベルドメインをサポートできます。

ホスト名では大文字と小文字が区別されず、最大 128 文字までの次の文字を含めることができます。

- A ~ Z、a ~ z、0 ~ 9
- - .
- * (0 個以上の文字と一致します)
- ? (厳密に 1 個の文字と一致します)

少なくとも 1 つの「.」文字を含める必要があります。最後の「.」の文字の後はアルファベット文字のみ含めることができます。

ホスト名の例

- **example.com**
- **test.example.com**
- ***.example.com**

ルール ***.example.com** は、**test.example.com** には一致しますが、**example.com** には一致しません。

Example のホストヘッダー条件の例 AWS CLI

ルールの作成時、または変更時に条件を指定できます。詳細については、create-rule および modify-rule コマンドを参照してください。以下の条件は、指定された文字列に一致するホストヘッダーを使用するリクエストによって満たされます。

```
[
  {
      "Field": "host-header",
      "HostHeaderConfig": {
          "Values": ["*.example.com"]
      }
  }
]
```

パスの条件

パスの条件を使用して、リクエスト内の URL に基づいてリクエストをルーティングするルールを定義できます (パスベースのルーティングとも呼ばれます)。

パスパターンは URL のパスにのみ適用され、クエリパラメータには適用されません。表示可能な ASCII 文字にのみ適用されます。制御文字 (0x00 ~ 0x1f および 0x7f) は除外されます。

ルール評価は、URI の正規化が発生した後にのみ実行されます。

パスパターンでは大文字と小文字が区別され、最大 128 文字までの次の文字を含めることができます。

- A ~ Z、a ~ z、0 ~ 9
- _ - . $ / ~ " ' @ : +
- & (& を使用)
- * (0 個以上の文字と一致します)
- ? (厳密に 1 個の文字と一致します)

プロトコルバージョンが grPC の場合は、パッケージ、サービス、またはメソッドに固有の条件を指定できます。

HTTP パスパターンの例

- `/img/*`
- `/img/*/pics`

grPC パスパターンの例

- /package
- /package.service
- /package.service/method

パスパターンはリクエストのルーティングに使用されますが、変更はしません。例えば、ルールにパスパターン `/img/*` がある場合、ルールは `/img/picture.jpg` のリクエストを `/img/picture.jpg` のリクエストとして、指定されたターゲットグループに転送します。

Example のパスパターン条件の例 AWS CLI

ルールの作成時、または変更時に条件を指定できます。詳細については、create-rule および modify-rule コマンドを参照してください。以下の条件は、指定された文字列を含む URL を使用するリクエストによって満たされます。

```
[
  {
```

```
      "Field": "path-pattern",
      "PathPatternConfig": {
          "Values": ["/img/*"]
      }
  }
]
```

クエリ文字列の条件

クエリ文字列の条件を使用して、キー/値のペアまたはクエリ文字列内の値に基づいてリクエストをルーティングするルールを設定できます。一致評価では大文字と小文字は区別されません。次のワイルドカード文字はサポートされています。* (0 個以上の文字と一致) および ? (厳密に 1 文字と一致)

Example のクエリ文字列条件の例 AWS CLI

ルールの作成時、または変更時に条件を指定できます。詳細については、create-rule および modify-rule コマンドを参照してください。次の条件は、「version = v1」のキー/値ペア、または「example」に設定されたキーのいずれかを含むクエリ文字列を使用するリクエストによって満たされます。

```
[
  {
      "Field": "query-string",
      "QueryStringConfig": {
          "Values": [
            {
                "Key": "version",
                "Value": "v1"
            },
            {
                "Value": "*example*"
            }
          ]
      }
  }
]
```

送信元 IP アドレス条件

送信元 IP アドレス条件を使用して、リクエストの送信元 IP アドレスに基づいてリクエストをルーティングするルールを設定できます。IP アドレスは CIDR 形式で指定する必要があります。IPv4 と

IPv6 の両方のアドレスを使用できます。ワイルドカード文字はサポートされていません。送信元 IP ルール条件に 255.255.255.255/32 CIDR を指定することはできません。

クライアントがプロキシの背後にある場合、これはクライアントの IP アドレスではなくプロキシの IP アドレスです。

この条件は、X-Forwarded-For ヘッダーのアドレスでは満たされません。X-Forwarded-For ヘッダー内のアドレスを検索するにhttp-header 条件を使用します。

Example のソース IP 条件の例 AWS CLI

ルールの作成時、または変更時に条件を指定できます。詳細については、create-rule および modify-rule コマンドを参照してください。次の条件は、指定された CIDR ブロックの 1 つの送信元 IP アドレスを使用するリクエストによって満たされます。

```
[
  {
      "Field": "source-ip",
      "SourceIpConfig": {
          "Values": ["192.0.2.0/24", "198.51.100.10/32"]
      }
  }
]
```

Application Load Balancer 用の HTTP リスナーを作成する

リスナーは、接続リクエストをチェックします。ロードバランサーを作成するときにリスナーを定義し、いつでもロードバランサーにリスナーを追加できます。

このページの情報は、ロードバランサー用の HTTP リスナーを作成するのに役立ちます。ロードバランサーに HTTPS リスナーを追加するには、Application Load Balancer 用の HTTPS リスナーを作成する を参照してください。

前提条件

- 転送アクションをデフォルトのリスナールールに追加するには、利用可能なターゲットグループを指定する必要があります。詳細については、「ターゲットグループの作成」を参照してください。
- 複数のリスナーで同じターゲットグループを指定できますが、これらのリスナーは同じロードバランサーに属している必要があります。ロードバランサーでターゲットグループを使用するには、

ターゲットグループが他のロードバランサーのリスナーによって使用されていないことを確認する必要があります。

HTTP リスナーを追加する

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、およびデフォルトのリスナールールのターゲットグループでリスナーを設定します。詳細については、「リスナーの設定」を参照してください。

コンソールを使用した HTTP リスナーを追加するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[ロードバランサー] を選択します。
3. ロードバランサーを選択します。
4. [リスナーとルール] タブから、[リスナーの追加] を選択します。
5. [プロトコル : ポート] で、[HTTP] を選択してデフォルトのポートのままにするか、別のポートを入力します。
6. [デフォルトアクション] で、以下のいずれかを選択します。
   - [ターゲットグループへ転送] — トラフィックを転送するターゲットグループを 1 つ以上選択します。ターゲットグループを追加するには、[ターゲットグループの追加] を選択します。複数のターゲットグループを使用している場合は、ターゲットグループごとに重みを選択し、それに関連付けられている割合を確認します。1 つ以上のターゲットグループに対して維持設定を有効にしている場合は、ルールのグループレベルの維持設定を有効にする必要があります。
   - [URL にリダイレクト] — クライアントリクエストのリダイレクト先の URL を指定します。これを行うには、部分ごとに分けて [URI 部分] タブに入力するか、完全なアドレスを [完全な URL] タブに入力します。必要に応じて、[ステータスコード] で、一時的 (HTTP 302) または恒久的 (HTTP 301) としてリダイレクトを設定します。
   - [固定レスポンスを返す] — ドロップされたクライアントリクエストに対して返される [レスポンスコード] を指定します。[コンテンツタイプ] と [レスポンス本文] の指定もできますが、これは必須ではありません。
7. [追加] を選択します。

を使用して HTTP リスナーを追加するには AWS CLI

create-listener コマンドを使用してリスナーとデフォルトのルールを作成し、create-rule コマンドを使用して追加のリスナールールを定義します。

Application Load Balancer 用の HTTPS リスナーを作成する

リスナーは、接続リクエストをチェックします。ロードバランサーを作成するときにリスナーを定義し、いつでもロードバランサーにリスナーを追加できます。

HTTPS リスナーを作成するには、ロードバランサーに SSL サーバー証明書を少なくとも 1 つデプロイする必要があります。ロードバランサーはサーバー証明書を使用してフロントエンド接続を終了してから、ターゲットにリクエストを送信する前に、クライアントからのリクエストを復号します。また、クライアントとロードバランサー間の安全な接続をネゴシエートするために使用されるセキュリティポリシーも指定する必要があります。

ロードバランサーが復号化せずに、暗号化されたトラフィックをターゲットに渡す必要がある場合は、ポート 443 で TCP リスナーを使用して Network Load Balancer または Classic Load Balancer を作成できます。TCP リスナーを使用すると、ロードバランサーは暗号化されたトラフィックを復号化せずにターゲットに渡します。

Application Load Balancer は、ED25519 キーをサポートしていません。

このページの情報は、ロードバランサー用の HTTPS リスナーを作成するのに役立ちます。ロードバランサーに HTTP リスナーを追加するには、Application Load Balancer 用の HTTP リスナーを作成する を参照してください。

目次

SSL 証明書

ロードバランサーには X.509 証明書 (SSL/TLS サーバー証明書) が必要です。証明書とは、認証機関 (CA) によって発行された識別用デジタル形式です。証明書には、認識用情報、有効期間、パブリックキー、シリアル番号と発行者のデジタル署名が含まれます。

ロードバランサーで使用する証明書を作成するときに、ドメイン名を指定する必要があります。TLS 接続を検証できるように、証明書のドメイン名は、カスタムドメイン名レコードと一致する必要があります。一致しない場合、トラフィックは暗号化されません。

`www.example.com` などの証明書の完全修飾ドメイン名 (FQDN) または `example.com` などの apex ドメイン名を指定する必要があります。また、同じドメインで複数のサイト名を保護するために、アスタリスク (*) をワイルドカードとして使用できます。ワイルドカード証明書をリクエストする場合、アスタリスク (*) はドメイン名の一番左の位置に付ける必要があり、1 つのサブドメインレベルのみを保護できます。例えば、`*.example.com` は `corp.example.com`、`images.example.com` を保護しますが、`test.login.example.com` を保護することはできません。また、`*.example.com` は、`example.com` のサブドメインのみを保護し、ネイキッドドメインまたは apex ドメイン (`example.com`) は保護しないことに注意してください。ワイルドカード名は、証明書の [サブジェクト] フィールドと [サブジェクト代替名] 拡張子に表示されます。公開証明書の詳細については、「AWS Certificate Manager ユーザーガイド」の「公開証明書」を参照してください。

AWS Certificate Manager (ACM) を使用して、ロードバランサーの証明書を作成することをお勧めします。ACM は、2048 ビット、3072 ビット、4096 ビットのキー長の RSA 証明書およびすべての ECDSA 証明書をサポートします。ACM は Elastic Load Balancing と統合して、ロードバランサーに証明書をデプロイできます。詳細については、AWS Certificate Manager ユーザーガイドを参照してください。

または、SSL/TLS ツールを使用して証明書署名リクエスト (CSR) を作成し、CA によって CSR 署名を取得して証明書を生成し、証明書を ACM にインポートするか、証明書を AWS Identity and Access Management (IAM) にアップロードすることもできます。ACM への証明書のインポートに関する詳細については、『AWS Certificate Manager ユーザーガイド』の「Importing Certificates」を参照してください。IAM への証明書のアップロードに関する詳細については、IAM ユーザーガイドの「IAM でのサーバー証明書の管理」を参照してください。

デフォルトの証明書

HTTPS リスナーを作成するには、厳密に 1 つの証明書を指定する必要があります。この証明書は、default certificate として知られています。HTTPS リスナーを作成した後、デフォルトの証明書

を置き換えることができます。詳細については、「デフォルトの証明書の置き換え」を参照してください。

証明書のリスト内の追加の証明書を指定する場合、クライアントがホスト名を指定するために Server Name Indication (SNI) プロトコルを使用せずに接続した場合、または証明書リストに一致する証明書がない場合にのみデフォルトの証明書が使用されます。

追加の証明書を指定せずに単一のロードバランサーを介して複数の安全なアプリケーションをホストする必要がある場合は、ワイルドカード証明書を使用するか、または追加ドメインごとにサブジェクト代替名 (SAN) を証明書に追加できます。

証明書リスト

HTTPS リスナーを作成すると、デフォルトの証明書と空の証明書リストが表示されます。リスナーの証明書リストに証明書を追加することもできます。証明書リストを使用すると、ロードバランサーは同じポートで複数のドメインをサポートし、ドメインごとに異なる証明書を提供できます。詳細については、「証明書リストに証明書を追加する」を参照してください。

ロードバランサーは、SNI をサポートするスマート証明書の選択アルゴリズムを使用します。クライアントから提供されたホスト名が証明書リスト内の単一の証明書と一致する場合、ロードバランサーはこの証明書を選択します。クライアントが提供するホスト名が証明書リストの複数の証明書と一致する場合、ロードバランサーはクライアントがサポートできる最適な証明書を選択します。証明書の選択は、次の条件と順序に基づいて行われます。

- パブリックキーアルゴリズム (RSA よりも ECDSA が優先)
- ハッシュアルゴリズム (MD5 よりも SHA が優先)
- キーの長さ (最大が優先)
- 有効期間

ロードバランサーアクセスログエントリは、クライアントが指定したホスト名とクライアントが提出する証明書を示します。詳細については、「アクセスログのエントリ」を参照してください。

証明書の更新

各証明書には有効期間が記載されています。有効期間が終了する前に、必ずロードバランサーの各証明書を更新するか、置き換える必要があります。これには、デフォルトの証明書と証明書リスト内の証明書が含まれます。証明書を更新または置き換えしても、ロードバランサーノードが受信し、正常なターゲットへのルーティングを保留中の未処理のリクエストには影響しません。証明書更新後、新

しいリクエストは更新された証明書を使用します。証明書置き換え後、新しいリクエストは新しい証明書を使用します。

証明書の更新と置き換えは次のとおりに管理できます。

- によって提供され AWS Certificate Manager 、ロードバランサーにデプロイされた証明書は、自動的に更新できます。ACM は、期限切れになる前に証明書の更新を試みます。詳細については、AWS Certificate Manager ユーザーガイドの 管理された更新 を参照してください。
- 証明書を ACM にインポートした場合は、証明書の有効期限をモニタリングし、期限切れ前に更新する必要があります。詳細については、AWS Certificate Manager ユーザーガイドの 証明書のインポート を参照してください。
- IAM に証明書をインポートする場合、新しい証明書を作成し、この新しい証明書を ACM あるいは IAM にインポートします。ロードバランサーにこの新しい証明書を追加し、期限切れの証明書をロードバランサーから削除します。

セキュリティポリシー

Elastic Load Balancing は、Secure Sockets Layer (SSL) ネゴシエーション設定 (セキュリティポリシーと呼ばれます) を使用して、クライアントとロードバランサー間の SSL 接続をネゴシエートします。セキュリティポリシーはプロトコルと暗号の組み合わせです。プロトコルは、クライアントとサーバーの間の安全な接続を確立し、クライアントとロードバランサーの間で受け渡しされるすべてのデータのプライバシーを保証します。暗号とは、暗号化キーを使用してコード化されたメッセージを作成する暗号化アルゴリズムです。プロトコルは、複数の暗号を使用し、インターネットを介してデータを暗号化します。接続ネゴシエーションのプロセスで、クライアントとロードバランサーでは、それぞれサポートされる暗号とプロトコルのリストが優先される順に表示されます。デフォルトでは、サーバーのリストで最初にクライアントの暗号と一致した暗号が安全な接続用に選択されます。

考慮事項:

- Application Load Balancer は、ターゲット接続のみの SSL 再ネゴシエーションをサポートしています。
- Application Load Balancer は、カスタムセキュリティポリシーをサポートしていません。
- `ELBSecurityPolicy-TLS13-1-2-2021-06` ポリシーは、 を使用して作成された HTTPS リスナーのデフォルトのセキュリティポリシーです AWS Management Console。
- `ELBSecurityPolicy-2016-08` ポリシーは、 を使用して作成された HTTPS リスナーのデフォルトのセキュリティポリシーです AWS CLI。

- HTTPS リスナーを作成するときは、セキュリティポリシーを選択する必要があります。
  - TLS 1.3 を含み、TLS 1.2 と下位互換性がある`ELBSecurityPolicy-TLS13-1-2-2021-06`セキュリティポリシーをお勧めします。
- フロントエンド接続に使用されるセキュリティポリシーを選択できますが、バックエンド接続には選択できません。
  - バックエンド接続では、HTTPS リスナーが TLS 1.3 セキュリティポリシーを使用している場合、`ELBSecurityPolicy-TLS13-1-0-2021-06` セキュリティポリシーが使用されます。それ以外の場合、バックエンド接続には `ELBSecurityPolicy-2016-08` セキュリティポリシーが使用されます。
- 特定の TLS プロトコルバージョンの無効化を必要とするコンプライアンスおよびセキュリティ標準を満たすため、または非推奨の暗号を必要とするレガシークライアントをサポートするには、いずれかの`ELBSecurityPolicy-TLS-`セキュリティポリシーを使用できます。Application Load Balancer へのリクエストの TLS プロトコルバージョンを表示するには、ロードバランサーのアクセスログ記録を有効にし、対応するアクセスログエントリを調べます。詳細については、Application Load Balancer のアクセスログ」を参照してください。
- IAM およびサービスコントロールポリシー (SCPs) でそれぞれ Elastic Load Balancing 条件キー AWS アカウント を使用することで、 および 全体の AWS Organizations ユーザーが利用できるセキュリティポリシーを制限できます。詳細については、「 AWS Organizations ユーザーガイド」のSCPs)」を参照してください。

TLS1.3 セキュリティポリシー

Elastic Load Balancing では、Application Load Balancer に次の TLS 1.3 セキュリティポリシーが用意されています。

- `ELBSecurityPolicy-TLS13-1-2-2021-06`（推奨）
- `ELBSecurityPolicy-TLS13-1-2-Res-2021-06`
- `ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06`
- `ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06`
- `ELBSecurityPolicy-TLS13-1-1-2021-06`
- `ELBSecurityPolicy-TLS13-1-0-2021-06`
- `ELBSecurityPolicy-TLS13-1-3-2021-06`

FIPS セキュリティポリシー

> **Important**
>
> Application Load Balancer にアタッチされたすべてのセキュアリスナーは、FIPS セキュリティポリシーまたは非 FIPS セキュリティポリシーのいずれかを使用する必要があります。これらを混在させることはできません。既存の Application Load Balancer に非 FIPS ポリシーを使用するリスナーが 2 つ以上あり、代わりにリスナーに FIPS セキュリティポリシーを使用させたい場合は、1 つしかないまですべてのリスナーを削除します。リスナーのセキュリティポリシーを FIPS に変更し、FIPS セキュリティポリシーを使用して追加のリスナーを作成します。または、FIPS セキュリティポリシーのみを使用して、新しいリスナーを持つ新しい Application Load Balancer を作成することもできます。

連邦情報処理規格 (FIPS) は、機密情報を保護する暗号化モジュールのセキュリティ要件を指定する米国およびカナダ政府の規格です。詳細については、 AWS クラウドセキュリティコンプライアンスページの「連邦情報処理規格 (FIPS) 140」を参照してください。

すべての FIPS ポリシーは、AWS-LC FIPS 検証済み暗号化モジュールを活用します。詳細については、NIST 暗号化モジュール検証プログラムサイトの AWS-LC 暗号化モジュールページを参照してください。

Elastic Load Balancing は、Application Load Balancer に次の FIPS セキュリティポリシーを提供します。

- `ELBSecurityPolicy-TLS13-1-3-FIPS-2023-04`
- `ELBSecurityPolicy-TLS13-1-2-Res-FIPS-2023-04`
- `ELBSecurityPolicy-TLS13-1-2-FIPS-2023-04` (推奨)
- `ELBSecurityPolicy-TLS13-1-2-Ext0-FIPS-2023-04`
- `ELBSecurityPolicy-TLS13-1-2-Ext1-FIPS-2023-04`
- `ELBSecurityPolicy-TLS13-1-2-Ext2-FIPS-2023-04`
- `ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04`
- `ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04`

FS がサポートするポリシー

Elastic Load Balancing では、Application Load Balancer に対して次の FS (フォワードシークレット) がサポートするセキュリティポリシーが用意されています。

- ELBSecurityPolicy-FS-1-2-Res-2020-10
- ELBSecurityPolicy-FS-1-2-Res-2019-08
- ELBSecurityPolicy-FS-1-2-2019-08
- ELBSecurityPolicy-FS-1-1-2019-08
- ELBSecurityPolicy-FS-2018-06

TLS 1.0 - 1.2 セキュリティポリシー

Elastic Load Balancing では、Application Load Balancer に次の TLS 1.0 ~ 1.2 セキュリティポリシーが用意されています。

- ELBSecurityPolicy-TLS-1-2-Ext-2018-06
- ELBSecurityPolicy-TLS-1-2-2017-01
- ELBSecurityPolicy-TLS-1-1-2017-01
- ELBSecurityPolicy-2016-08
- ELBSecurityPolicy-TLS-1-0-2015-04
- ELBSecurityPolicy-2015-05 (と同じELBSecurityPolicy-2016-08)

TLS プロトコルと暗号

TLS 1.3

次の表は、使用可能な TLS 1.3 セキュリティポリシーでサポートされている TLS プロトコルと暗号を示しています。

注： セキュリティポリシー行のポリシー名からELBSecurityPolicy-プレフィックスが削除されました。

例： セキュリティポリシーELBSecurityPolicy-TLS13-1-2-2021-06は として表示されますTLS13-1-2-2021-06。

セキュリティポリシー	TLS13-1-2-2021-06	TLS13-1-3-2021-06	TLS13-1-2-Res-2021-06	TLS13-1-2-Ext2-2021-06	TLS13-1-2-Ext1-2021-06	TLS13-1-1-2021-06	TLS13-1-0-2021-06
TLS Protocols							
Protocol-TLSv1							✓
Protocol-TLSv1.1						✓	✓
Protocol-TLSv1.2	✓		✓	✓	✓	✓	✓
Protocol-TLSv1.3	✓	✓	✓	✓	✓	✓	✓
TLS Ciphers							
TLS_AES_128_GCM_SHA256	✓	✓	✓	✓	✓	✓	✓
TLS_AES_256_GCM_SHA384	✓	✓	✓	✓	✓	✓	✓
TLS_CHACHA20_POLY1305_SHA256	✓	✓	✓	✓	✓	✓	✓
ECDHE-ECDSA-AES128	✓		✓	✓	✓	✓	✓

セキュリティポリシー	TLS13-1-2-2021-06	TLS13-1-3-2021-06	TLS13-1-2-Res-2021-06	TLS13-1-2-Ext2-2021-06	TLS13-1-2-Ext1-2021-06	TLS13-1-1-2021-06	TLS13-1-0-2021-06
-GCM-SHA256							
ECDHE-RSA-AES128-GCM-SHA256	✓		✓	✓	✓	✓	✓
ECDHE-ECDSA-AES128-SHA256	✓			✓	✓	✓	✓
ECDHE-RSA-AES128-SHA256	✓			✓	✓	✓	✓
ECDHE-ECDSA-AES128-SHA				✓		✓	✓
ECDHE-RSA-AES128-SHA				✓		✓	✓

セキュリティポリシー	TLS13-1-2-2021-06	TLS13-1-3-2021-06	TLS13-1-2-Res-2021-06	TLS13-1-2-Ext2-2021-06	TLS13-1-2-Ext1-2021-06	TLS13-1-1-2021-06	TLS13-1-0-2021-06
ECDHE-ECDSA-AES256-GCM-SHA384	✓		✓	✓	✓	✓	✓
ECDHE-RSA-AES256-GCM-SHA384	✓		✓	✓	✓	✓	✓
ECDHE-ECDSA-AES256-SHA384	✓			✓	✓	✓	✓
ECDHE-RSA-AES256-SHA384	✓			✓	✓	✓	✓
ECDHE-RSA-AES256-SHA				✓		✓	✓
ECDHE-ECDSA-AES256-SHA				✓		✓	✓

セキュリティポリシー	TLS13-1-2-2021-06	TLS13-1-3-2021-06	TLS13-1-2-Res-2021-06	TLS13-1-2-Ext2-2021-06	TLS13-1-2-Ext1-2021-06	TLS13-1-1-2021-06	TLS13-1-0-2021-06
AES128-GCM-SHA256				✓	✓	✓	✓
AES128-SHA256				✓	✓	✓	✓
AES128-SHA				✓		✓	✓
AES256-GCM-SHA384				✓	✓	✓	✓
AES256-SHA256				✓	✓	✓	✓
AES256-SHA				✓		✓	✓

CLI を使用して TLS 1.3 ポリシーを使用する HTTPS リスナーを作成するには

任意の TLS 1.3 セキュリティポリシー で create-listener コマンドを使用します。 ???

この例では、 ELBSecurityPolicy-TLS13-1-2-2021-06 セキュリティポリシーを使用しています。

```
aws elbv2 create-listener --name my-listener \
--protocol HTTPS --port 443 \
--ssl-policy ELBSecurityPolicy-TLS13-1-2-2021-06
```

CLI を使用して TLS 1.3 ポリシーを使用するように HTTPS リスナーを変更するには

TLS 1.3 セキュリティポリシー で modify-listener コマンドを使用します。 ???

この例では、 ELBSecurityPolicy-TLS13-1-2-2021-06 セキュリティポリシーを使用しています。

```
aws elbv2 modify-listener \
--listener-arn arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/my-load-balancer/abcdef01234567890/1234567890abcdef0 \
--ssl-policy ELBSecurityPolicy-TLS13-1-2-2021-06
```

CLI を使用してリスナーが使用するセキュリティポリシーを表示するには

リスナーの で describe-listeners コマンドarnを使用します。

```
aws elbv2 describe-listeners \
--listener-arn arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/my-load-balancer/abcdef01234567890/1234567890abcdef0
```

CLI を使用して TLS 1.3 セキュリティポリシーの設定を表示するには

describe-ssl-policies コマンドを任意の TLS 1.3 セキュリティポリシー で使用します。

この例では、 ELBSecurityPolicy-TLS13-1-2-2021-06 セキュリティポリシーを使用しています。

```
aws elbv2 describe-ssl-policies \
--names ELBSecurityPolicy-TLS13-1-2-2021-06
```

FIPS

> **Important**
>
> ポリシー ELBSecurityPolicy-TLS13-1-1-FIPS-2023-04 および ELBSecurityPolicy-TLS13-1-0-FIPS-2023-04は、レガシー互換性のためにのみ提供されています。FIPS140 モジュールを使用して FIPS 暗号化を使用していますが、TLS 設定に関する最新の NIST ガイダンスに準拠していない可能性があります。

次の表は、使用可能な FIPS セキュリティポリシーでサポートされている TLS プロトコルと暗号を示しています。

注： セキュリティポリシー行のポリシー名からELBSecurityPolicy-プレフィックスが削除されました。

例： セキュリティポリシーELBSecurityPolicy-TLS13-1-2-FIPS-2023-04は として表示されますTLS13-1-2-FIPS-2023-04。

セキュリティポリシー	TLS13-1-3-FIPS-2023-04	TLS13-1-2-Res-FIPS-2023-04	TLS13-1-2-FIPS-2023-04	TLS13-1-2-Ext0-FIPS-2023-04	TLS13-1-2-Ext1-FIPS-2023-04	TLS13-1-2-Ext2-FIPS-2023-04	TLS13-1-1-FIPS-2023-04	TLS13-1-0-FIPS-2023-04
TLS Protocols								
Protocol-TLSv1								✓
Protocol-TLSv1.1							✓	✓
Protocol-TLSv1.2		✓	✓	✓	✓	✓	✓	✓
Protocol-TLSv1.3	✓	✓	✓	✓	✓	✓	✓	✓
TLS Ciphers								
TLS_AES_128_GCM_SHA256	✓	✓	✓	✓	✓	✓	✓	✓
TLS_AES_256_GCM_SHA384	✓	✓	✓	✓	✓	✓	✓	✓

セキュリティポリシー	TLS13-1-3-FIPS-2023-04	TLS13-1-2-Res-FIPS-2023-04	TLS13-1-2-FIPS-2023-04	TLS13-1-2-Ext0-FIPS-2023-04	TLS13-1-2-Ext1-FIPS-2023-04	TLS13-1-2-Ext2-FIPS-2023-04	TLS13-1-1-FIPS-2023-04	TLS13-1-0-FIPS-2023-04
ECDHE-ECDSA-AES128-GCM-SHA256		✓	✓	✓	✓	✓	✓	✓
ECDHE-RSA-AES128-GCM-SHA256		✓	✓	✓	✓	✓	✓	✓
ECDHE-ECDSA-AES128-SHA256			✓	✓	✓	✓	✓	✓
ECDHE-RSA-AES128-SHA256			✓	✓	✓	✓	✓	✓

セキュリティポリシー	TLS13-1-3-FIPS-2023-04	TLS13-1-2-Res-FIPS-2023-04	TLS13-1-2-FIPS-2023-04	TLS13-1-2-Ext0-FIPS-2023-04	TLS13-1-2-Ext1-FIPS-2023-04	TLS13-1-2-Ext2-FIPS-2023-04	TLS13-1-1-FIPS-2023-04	TLS13-1-0-FIPS-2023-04
ECDHE-ECDSA-AES128-SHA				✓		✓	✓	✓
ECDHE-RSA-AES128-SHA				✓		✓	✓	✓
ECDHE-ECDSA-AES256-GCM-SHA384		✓	✓	✓	✓	✓	✓	✓
ECDHE-RSA-AES256-GCM-SHA384		✓	✓	✓	✓	✓	✓	✓

セキュリティポリシー	TLS13-1-3-FIPS-2023-04	TLS13-1-2-Res-FIPS-2023-04	TLS13-1-2-FIPS-2023-04	TLS13-1-2-Ext0-FIPS-2023-04	TLS13-1-2-Ext1-FIPS-2023-04	TLS13-1-2-Ext2-FIPS-2023-04	TLS13-1-1-FIPS-2023-04	TLS13-1-0-FIPS-2023-04
ECDHE-ECDSA-AES256-SHA384			✓	✓	✓	✓	✓	✓
ECDHE-RSA-AES256-SHA384			✓	✓	✓	✓	✓	✓
ECDHE-RSA-AES256-SHA				✓		✓	✓	✓
ECDHE-ECDSA-AES256-SHA				✓		✓	✓	✓
AES128-GCM-SHA256					✓	✓	✓	✓

セキュリティポリシー	TLS13-1-3-FIPS-2023-04	TLS13-1-2-Res-FIPS-2023-04	TLS13-1-2-FIPS-2023-04	TLS13-1-2-Ext0-FIPS-2023-04	TLS13-1-2-Ext1-FIPS-2023-04	TLS13-1-2-Ext2-FIPS-2023-04	TLS13-1-1-FIPS-2023-04	TLS13-1-0-FIPS-2023-04
AES128-SHA256					✓	✓	✓	✓
AES128-SHA						✓	✓	✓
AES256-GCM-SHA384					✓	✓	✓	✓
AES256-SHA256					✓	✓	✓	✓
AES256-SHA						✓	✓	✓

CLI を使用して FIPS ポリシーを使用する HTTPS リスナーを作成するには

任意の FIPS セキュリティポリシー で create-listener コマンドを使用します。 ???

この例では、 ELBSecurityPolicy-TLS13-1-2-FIPS-2023-04 セキュリティポリシーを使用しています。

```
aws elbv2 create-listener --name my-listener \
--protocol HTTPS --port 443 \
--ssl-policy ELBSecurityPolicy-TLS13-1-2-FIPS-2023-04
```

CLI を使用して FIPS ポリシーを使用するように HTTPS リスナーを変更するには

任意の FIPS セキュリティポリシー で modify-listener コマンドを使用します。 ???

この例では、 ELBSecurityPolicy-TLS13-1-2-FIPS-2023-04 セキュリティポリシーを使用しています。

```
aws elbv2 modify-listener \
--listener-arn arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/my-load-balancer/abcdef01234567890/1234567890abcdef0 \
--ssl-policy ELBSecurityPolicy-TLS13-1-2-FIPS-2023-04
```

CLI を使用してリスナーが使用するセキュリティポリシーを表示するには

リスナーの で describe-listeners コマンドarnを使用します。

```
aws elbv2 describe-listeners \
--listener-arn arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/my-load-balancer/abcdef01234567890/1234567890abcdef0
```

CLI を使用して FIPS セキュリティポリシーの設定を表示するには

describe-ssl-policies コマンドを任意の FIPS セキュリティポリシー で使用します。

この例では、 ELBSecurityPolicy-TLS13-1-2-FIPS-2023-04 セキュリティポリシーを使用しています。

```
aws elbv2 describe-ssl-policies \
--names ELBSecurityPolicy-TLS13-1-2-FIPS-2023-04
```

FS

次の表は、使用可能な FS がサポートするセキュリティポリシーでサポートされている TLS プロトコルと暗号を示しています。

注： セキュリティポリシー行のポリシー名からELBSecurityPolicy-プレフィックスが削除されました。

例： セキュリティポリシーELBSecurityPolicy-FS-2018-06は として表示されますFS-2018-06。

セキュリティポリシー	Default	FS-1-2-Res-2020-10	FS-1-2-Res-2019-08	FS-1-2-2019-08	FS-1-1-2019-08	FS-2018-06
TLS Protocols						
Protocol-TLSv1	✓					✓
Protocol-TLSv1.1	✓				✓	✓
Protocol-TLSv1.2	✓	✓	✓	✓	✓	✓
TLS Ciphers						
ECDHE-ECDSA-AES128-GCM-SHA256	✓	✓	✓	✓	✓	✓
ECDHE-RSA-AES128-GCM-SHA256	✓	✓	✓	✓	✓	✓
ECDHE-ECDSA-AES128-SHA256	✓		✓	✓	✓	✓

セキュリティポリシー	Default	FS-1-2-Res-2020-10	FS-1-2-Res-2019-08	FS-1-2-2019-08	FS-1-1-2019-08	FS-2018-06
ECDHE-RSA-AES128-SHA256	✓		✓	✓	✓	✓
ECDHE-ECDSA-AES128-SHA	✓			✓	✓	✓
ECDHE-RSA-AES128-SHA	✓			✓	✓	✓
ECDHE-ECDSA-AES256-GCM-SHA384	✓	✓	✓	✓	✓	✓
ECDHE-RSA-AES256-GCM-SHA384	✓	✓	✓	✓	✓	✓

セキュリティポリシー	Default	FS-1-2-Res-2020-10	FS-1-2-Res-2019-08	FS-1-2-2019-08	FS-1-1-2019-08	FS-2018-06
ECDHE-ECDSA-AES256-SHA384	✓		✓	✓	✓	✓
ECDHE-RSA-AES256-SHA384	✓		✓	✓	✓	✓
ECDHE-RSA-AES256-SHA	✓			✓	✓	✓
ECDHE-ECDSA-AES256-SHA	✓			✓	✓	✓
AES128-GCM-SHA256	✓					
AES128-SHA256	✓					
AES128-SHA	✓					

セキュリティポリシー	Default	FS-1-2-Res-2020-10	FS-1-2-Res-2019-08	FS-1-2-2019-08	FS-1-1-2019-08	FS-2018-06
AES256-GCM-SHA384	✓					
AES256-SHA256	✓					
AES256-SHA	✓					

CLI を使用して FS がサポートするポリシーを使用する HTTPS リスナーを作成するには

FS がサポートするセキュリティポリシー で create-listener コマンドを使用します。 ???

この例では、 ELBSecurityPolicy-FS-2018-06 セキュリティポリシーを使用しています。

```
aws elbv2 create-listener --name my-listener \
--protocol HTTPS --port 443 \
--ssl-policy ELBSecurityPolicy-FS-2018-06
```

CLI を使用して FS がサポートするポリシーを使用するように HTTPS リスナーを変更するには

FS がサポートするセキュリティポリシー で modify-listener コマンドを使用します。 ???

この例では、 ELBSecurityPolicy-FS-2018-06 セキュリティポリシーを使用しています。

```
aws elbv2 modify-listener \
--listener-arn arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/my-load-balancer/abcdef01234567890/1234567890abcdef0 \
--ssl-policy ELBSecurityPolicy-FS-2018-06
```

CLI を使用してリスナーが使用するセキュリティポリシーを表示するには

リスナーの で describe-listeners コマンドarnを使用します。

```
aws elbv2 describe-listeners \
--listener-arn arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/my-load-balancer/abcdef01234567890/1234567890abcdef0
```

CLI を使用して FS がサポートするセキュリティポリシーの設定を表示するには

describe-ssl-policies コマンドは、FS がサポートするセキュリティポリシー で使用します。

この例では、 ELBSecurityPolicy-FS-2018-06 セキュリティポリシーを使用しています。

```
aws elbv2 describe-ssl-policies \
--names ELBSecurityPolicy-FS-2018-06
```

TLS 1.0 - 1.2

次の表は、使用可能な TLS 1.0-1.2 セキュリティポリシーでサポートされている TLS プロトコルと暗号を示しています。

注： セキュリティポリシー行のポリシー名からELBSecurityPolicy-プレフィックスが削除されました。

例： セキュリティポリシーELBSecurityPolicy-TLS-1-2-Ext-2018-06は として表示されますTLS-1-2-Ext-2018-06。

セキュリティポリシー	Default	TLS-1-2-Ext-2018-06	TLS-1-2-2017-01	TLS-1-1-2017-01	TLS-1-0-2015-04*
TLS Protocols					
Protocol-TLSv1	✓				✓

セキュリティポリシー	Default	TLS-1-2-Ext-2018-06	TLS-1-2-2017-01	TLS-1-1-2017-01	TLS-1-0-2015-04*
Protocol-TLSv1.1	✓			✓	✓
Protocol-TLSv1.2	✓	✓	✓	✓	✓
TLS Ciphers					
ECDHE-ECDSA-AES128-GCM-SHA256	✓	✓	✓	✓	✓
ECDHE-RSA-AES128-GCM-SHA256	✓	✓	✓	✓	✓
ECDHE-ECDSA-AES128-SHA256	✓	✓	✓	✓	✓
ECDHE-RSA-AES128-SHA256	✓	✓	✓	✓	✓
ECDHE-ECDSA-AES128-SHA	✓	✓		✓	✓

セキュリティポリシー	Default	TLS-1-2-Ext-2018-06	TLS-1-2-2017-01	TLS-1-1-2017-01	TLS-1-0-2015-04*
ECDHE-RSA-AES128-SHA	✓	✓		✓	✓
ECDHE-ECDSA-AES256-GCM-SHA384	✓	✓	✓	✓	✓
ECDHE-RSA-AES256-GCM-SHA384	✓	✓	✓	✓	✓
ECDHE-ECDSA-AES256-SHA384	✓	✓	✓	✓	✓
ECDHE-RSA-AES256-SHA384	✓	✓	✓	✓	✓
ECDHE-RSA-AES256-SHA	✓	✓		✓	✓
ECDHE-ECDSA-AES256-SHA	✓	✓		✓	✓

セキュリティポリシー	Default	TLS-1-2-Ext-2018-06	TLS-1-2-2017-01	TLS-1-1-2017-01	TLS-1-0-2015-04*
AES128-GCM-SHA256	✓	✓	✓	✓	✓
AES128-SHA256	✓	✓	✓	✓	✓
AES128-SHA	✓	✓		✓	✓
AES256-GCM-SHA384	✓	✓	✓	✓	✓
AES256-SHA256	✓	✓	✓	✓	✓
AES256-SHA	✓	✓		✓	✓
DES-CBC3-SHA					✓

* DES-CBC3-SHA 暗号 (弱い暗号) を必要とするレガシークライアントをサポートする必要がない限り、このポリシーは使用しないでください。

CLI を使用して TLS 1.0-1.2 ポリシーを使用する HTTPS リスナーを作成するには

TLS 1.0-1.2 でサポートされているセキュリティポリシー で create-listener コマンドを使用します。

この例では、 ELBSecurityPolicy-2016-08 セキュリティポリシーを使用しています。

```
aws elbv2 create-listener --name my-listener \
--protocol HTTPS --port 443 \
--ssl-policy ELBSecurityPolicy-2016-08
```

CLI を使用して TLS 1.0-1.2 ポリシーを使用するように HTTPS リスナーを変更するには

TLS 1.0-1.2 でサポートされているセキュリティポリシー で modify-listener コマンドを使用します。

この例では、 ELBSecurityPolicy-2016-08 セキュリティポリシーを使用しています。

```
aws elbv2 modify-listener \
--listener-arn arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/my-load-balancer/abcdef01234567890/1234567890abcdef0 \
--ssl-policy ELBSecurityPolicy-2016-08
```

CLI を使用してリスナーが使用するセキュリティポリシーを表示するには

リスナーの で describe-listeners コマンドarnを使用します。

```
aws elbv2 describe-listeners \
--listener-arn arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/my-load-balancer/abcdef01234567890/1234567890abcdef0
```

CLI を使用して TLS 1.0-1.2 セキュリティポリシーの設定を表示するには

describe-ssl-policies コマンドは、TLS 1.0-1.2 でサポートされているセキュリティポリシー で使用します。

この例では、 ELBSecurityPolicy-2016-08 セキュリティポリシーを使用しています。

```
aws elbv2 describe-ssl-policies \
--names ELBSecurityPolicy-2016-08
```

HTTPS リスナーの追加

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、およびデフォルトのリスナールールのターゲットグループでリスナーを設定します。詳細については、「リスナーの設定」を参照してください。

前提条件

- HTTPS リスナーを作成するには、証明書とセキュリティポリシーを指定する必要があります。ターゲットにリクエストをルーティングする前に、ロードバランサーはこの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。ロードバランサーは、クライアントと SSL 接続のネゴシエーションを行うときにセキュリティポリシーを使用します。
- 転送アクションをデフォルトのリスナールールに追加するには、利用可能なターゲットグループを指定する必要があります。詳細については、「ターゲットグループの作成」を参照してください。
- 複数のリスナーで同じターゲットグループを指定できますが、これらのリスナーは同じロードバランサーに属している必要があります。ロードバランサーでターゲットグループを使用するには、ターゲットグループが他のロードバランサーのリスナーによって使用されていないことを確認する必要があります。

コンソールを使用した HTTPS リスナーを追加するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[ロードバランサー] を選択します。
3. ロードバランサーを選択します。
4. [リスナーとルール] タブから、[リスナーの追加] を選択します。
5. [プロトコル : ポート] で、[HTTPS] を選択してデフォルトのポートのままにするか、別のポートを入力します。
6. (オプション) 認証を有効にするには、[認証] で [OpenID または Amazon Cognito を使用する] を選択し、要求された情報を提供してください。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。
7. [Default actions (デフォルトアクション)] で、次のいずれかを実行します。
   - [ターゲットグループへ転送] — トラフィックを転送するターゲットグループを 1 つ以上選択します。ターゲットグループを追加するには、[ターゲットグループの追加] を選択します。複数のターゲットグループを使用している場合は、ターゲットグループごとに重みを選択し、それに関連付けられている割合を確認します。1 つ以上のターゲットグループに対して維持設定を有効にしている場合は、ルールのグループレベルの維持設定を有効にする必要があります。
   - [URL にリダイレクト] — クライアントリクエストのリダイレクト先の URL を指定します。これを行うには、部分ごとに分けて [URI 部分] タブに入力するか、完全なアドレスを [完全な URL] タブに入力します。必要に応じて、[ステータスコード] で、一時的 (HTTP 302) または恒久的 (HTTP 301) としてリダイレクトを設定します。

- [固定レスポンスを返す] — ドロップされたクライアントリクエストに対して返される [レスポンスコード] を指定します。[コンテンツタイプ] と [レスポンス本文] の指定もできますが、これは必須ではありません。

8. [セキュリティポリシー] では、常に最新の事前定義されたセキュリティポリシーを使用することをお勧めします。
9. [デフォルトの SSL/TLS 証明書] では、以下のオプションがあります。

   - を使用して証明書を作成またはインポートした場合は AWS Certificate Manager、「ACM から」を選択し、「証明書の選択」から証明書を選択します。
   - IAM を使用して証明書をすでにインポートしている場合は、[IAM から] を選択し、[証明書の選択] から対象の証明書を選択します。
   - インポートする証明書はあるが、リージョンで ACM を利用できない場合は、[インポート] を選択し、次に [IAM へ] を選択します。[証明書名] フィールドに証明書の名前を入力します。[証明書のプライベートキー] に、PEM エンコードされたプライベートキーファイルの内容をコピーして貼り付けます。[証明書本文] に、PEM エンコードされたパブリックキー証明書ファイルの内容をコピーして貼り付けます。自己署名証明書を使用しておらず、ブラウザが暗黙的に証明書を受け入れることが重要である場合に限り、[Certificate Chain] に、PEM エンコードされた証明書チェーンファイルの内容をコピーして貼り付けます。

10. （オプション) 相互認証を有効にするには、クライアント証明書処理で相互認証 (mTLS) を有効にします。

    有効にすると、デフォルトの相互 TLS モードはパススルー になります。

    Trust Store で検証 を選択した場合：

    - デフォルトでは、期限切れのクライアント証明書を持つ接続は拒否されます。この動作を変更するには、アドバンスト mTLS 設定 を展開し、クライアント証明書の有効期限 で、期限切れのクライアント証明書を許可する を選択します。
    - Trust Store で既存の信頼ストアを選択するか、新しい信頼ストア を選択します。
      - 新しい信頼ストア を選択した場合は、信頼ストア名 、S3 URI 認証局の場所 、およびオプションで S3 URI 証明書失効リストの場所 を指定します。

11. [保存] を選択します。

を使用して HTTPS リスナーを追加するには AWS CLI

create-listener コマンドを使用してリスナーとデフォルトのルールを作成し、create-rule コマンドを使用して追加のリスナールールを定義します。

Application Load Balancer のリスナールール

リスナーに対して定義したルールは、ロードバランサーが 1 つ以上のターゲットグループ内のターゲットにリクエストをルーティングする方法を決定します。

各ルールは優先度、1 つ以上のアクション、および 1 つ以上の条件で構成されています。詳細については、「リスナールール」を参照してください。

要件

- ルールはセキュアリスナーにのみアタッチできます。
- 各ルールには次のアクションのうち、厳密に 1 つを含む必要があります。`forward`、`redirect`、`fixed-response`。またはそれは最後に実行されるアクションである必要があります。
- 各ルールには、以下の条件の 0 個または 1 つを含めることができます。`host-header`、`http-request-method`、`path-pattern`、`source-ip`および 0 個以上の以下の条件 `http-header`、`query-string`を含めることができます。
- 条件ごとに最大 3 つの比較文字列、ルールごとに最大 5 つの比較文字列を指定できます。
- `forward` アクションはリクエストをそのターゲットグループにルーティングします。`forward` アクションを追加する前に、ターゲットグループを作成し、それにターゲットを追加します。詳細については、「ターゲットグループの作成」を参照してください。

ルールの追加

リスナーを作成するときはデフォルトのルールを定義し、デフォルト以外の追加のルールはいつでも定義できます。

コンソールを使用してルールを追加するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[ロードバランサー] を選択します。
3. ロードバランサーを選択すると、詳細が表示されます。
4. [リスナーとルール] タブで、次のいずれかを行います。

a. [プロトコル:ポート] 列のテキストを選択して、リスナーの詳細ページを開きます。

   [ルール] タブで、[ルールを追加する] を選択します。

b. ルールを追加するリスナーを選択します。

   [ルールを管理]、[ルールを追加する] の順に選択します。

5. [名前とタグ] でルール名を指定できますが、これは必須ではありません。

   新しいタグを追加するには、[さらにタグを追加] を選択します。

6. [次へ] をクリックします。

7. [条件を追加] を選択します。

8. 以下から条件を 1 つ以上追加します。

   - [ホストヘッダー] — ホストヘッダーを定義します。例: `*.example.com`。[確認] を選択し、内容を保存します。

     最大 128 文字 大文字と小文字は区別されません。使用できる文字は a ~ z、A ~ Z、0 ~ 9、特殊文字 (-_.)、ワイルドカード (* および ?) です。

   - [パス] — パスを定義します。例: `/item/*` 。[確認] を選択し、内容を保存します。

     最大 128 文字 大文字と小文字の区別があります。使用できる文字は a ~ z、A ~ Z、0 ~ 9、特殊文字 (_-.$/~'"@:+;&)、ワイルドカード (* および ?) です。

   - [HTTP リクエストメソッド] — HTTP リクエストメソッドを定義します。[確認] を選択し、内容を保存します。

     最大 40 文字 大文字と小文字の区別があります。使用できる文字は A ~ Z および特殊文字 (-_) です。ワイルドカードがサポートされていません。

   - [送信元 IP] — 送信元の IP アドレスを CIDR 形式で定義します。[確認] を選択し、内容を保存します。

     IPv4 CDIR と IPv6 CIDR のどちらも使用できます。ワイルドカードがサポートされていません。

   - [HTTP ヘッダー] – ヘッダーの名前を入力して、1 つ以上の比較文字列を追加します。[確認] を選択し、内容を保存します。

     - [HTTP ヘッダー名] — ルールによって、このヘッダーを含むリクエストが評価され、値が一致することが確認されます。

最大 40 文字 大文字と小文字は区別されません。使用できる文字は a ~ z、A ~ Z、0 ~ 9、特殊文字 (*?-!#$%&'+.^_`|~) です。ワイルドカードがサポートされていません。

- [HTTP ヘッダー値] — HTTP ヘッダー値と比較する文字列を入力します。

  最大 128 文字 大文字と小文字は区別されません。使用できる文字は a ~ z、A ~ Z、0 ~ 9、スペース、特殊文字 (!"#$%&'()+,./:;#=>@[]^_`{|}~-)、ワイルドカード (* および ?) です。

- [クエリ文字列] – キーと値のペアまたはクエリ文字列の値に基づいて、リクエストをルーティングします。[確認] を選択し、内容を保存します。

  最大 128 文字 大文字と小文字は区別されません。使用できる文字は a ~ z、A ~ Z、0 ~ 9、特殊文字 (_-.$/~"'@:+&()!,;=)、ワイルドカード (* および ?) です。

9. [次へ] をクリックします。
10. ルールに以下のアクションのいずれかを定義します。

    - [ターゲットグループへ転送] — トラフィックを転送するターゲットグループを 1 つ以上選択します。ターゲットグループを追加するには、[ターゲットグループの追加] を選択します。複数のターゲットグループを使用している場合は、ターゲットグループごとに重みを選択し、それに関連付けられている割合を確認します。1 つ以上のターゲットグループに対して維持設定を有効にしている場合は、ルールのグループレベルの維持設定を有効にする必要があります。
    - [URL にリダイレクト] — クライアントリクエストのリダイレクト先の URL を指定します。これを行うには、部分ごとに分けて [URI 部分] タブに入力するか、完全なアドレスを [完全な URL] タブに入力します。必要に応じて、[ステータスコード] で、一時的 (HTTP 302) または恒久的 (HTTP 301) としてリダイレクトを設定します。
    - [固定レスポンスを返す] — ドロップされたクライアントリクエストに対して返される [レスポンスコード] を指定します。[コンテンツタイプ] と [レスポンス本文] の指定もできますが、これは必須ではありません。
11. [次へ] をクリックします。
12. 1 ~ 50000 の値を入力して、ルールの Priority を指定します。
13. [次へ] をクリックします。
14. 新しいルールの詳細と現在構成されている設定を確認します。選択した内容でよければ、[作成] を選択します。

を使用してルールを追加するには AWS CLI

ルールを作成するには、create-rule コマンドを使用します。ルールに関する情報を確認するには、describe-rules コマンドを使用します。

ルールの編集

ルールのアクションおよび条件はいつでも編集できます。ルールの更新はすぐには反映されないため、ルールの更新後しばらくの間、リクエストは以前のルール設定を使用してルーティングされます。すべての未処理のリクエストが完了します。

コンソールを使用してルールを編集するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[ロードバランサー] を選択します。
3. ロードバランサーを選択します。
4. [リスナーとルール] タブで、次のいずれかを行います。
   - [プロトコル:ポート] 列のテキストを選択して、リスナーの詳細ページを開きます。
     i. [ルール] タブの [リスナールール] セクションで、編集するルールの [Name タグ] 列にあるテキストを選択します。

        [アクション]、[ルールの編集] の順に選択します。
     ii. [ルール] タブの [リスナールール] セクションで、編集するルールを選択します。

        [アクション]、[ルールの編集] の順に選択します。
5. 必要に応じて名前とタグを変更します。新しいタグを追加するには、[さらにタグを追加] を選択します。
6. [次へ] を選択します。
7. 必要に応じて条件を変更します。既存の条件を追加、編集、または削除できます。
8. [次へ] を選択します。
9. 必要に応じてアクションを変更します。
10. [次へ] を選択します。
11. 必要に応じてルールの優先度を変更します。1 ~ 50000 の値を入力できます。
12. [次へ] を選択します。
13. ルールに設定されたすべての詳細と更新された設定を確認します。選択内容に問題がなければ、変更を保存 を選択します。

を使用してルールを編集するには AWS CLI

modify-rule コマンドを使用します。

ルールの優先度の更新

ルールは優先順位の低~高順によって評価されます。デフォルトのルールが最後に評価されます。デフォルト以外のルールは、優先順位をいつでも変更できます。デフォルトルールの優先順位は変更できません。

コンソールを使用してルールの優先度を更新するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[ロードバランサー] を選択します。
3. ロードバランサーを選択します。
4. [リスナーとルール] タブで、次のいずれかを行います。

   a. [プロトコル:ポート] または [ルール] 列のテキストを選択して、リスナーの詳細ページを開きます。

      i. [アクション]、[ルールに優先順位を再設定] の順に選択します。

      ii. [ルール] タブの [リスナールール] セクションで、[アクション]、[ルールに優先順位を再設定] の順に選択します。

   b. リスナーを選択します。

      - [ルールを管理]、[ルールに優先順位を再設定] の順に選択します。

5. [リスナールール] セクションの [優先度] 列には、現在のルールの優先度が表示されます。ルールの優先度を更新するには、1 ~ 50000 の値を入力します。
6. 変更内容に問題がなければ、[変更内容の保存] を選択します。

を使用してルールの優先順位を更新するには AWS CLI

set-rule-priorities コマンドを使用します。

ルールの削除

リスナーのデフォルト以外のルールはいつでも削除できます。リスナーのデフォルトのルールは削除できません。リスナーを削除すると、そのルールはすべて削除されます。

コンソールを使用してルールを削除するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[ロードバランサー] を選択します。
3. ロードバランサーを選択します。
4. [リスナーとルール] タブで、次のいずれかを行います。
   a. [プロトコル:ポート] または [ルール] 列のテキストを選択して、リスナーの詳細ページを開きます。
      i. 削除するルールを選択します。
      ii. [アクション]、[ルールを削除] の順に選択します。
      iii. テキストフィールドに `confirm` と入力し、[削除] を選択します。
   b. [Name タグ] 列のテキストを選択し、ルールの詳細ページを開きます。
      i. [アクション]、[ルールを削除] の順に選択します。
      ii. テキストフィールドに `confirm` と入力し、[削除] を選択します。

を使用してルールを削除するには AWS CLI

delete-rule コマンドを使用します。

Application Load Balancer 用の HTTPS リスナーを更新する

HTTPS リスナーを作成すると、デフォルトの証明書の置き換え、証明書リストの更新、またはセキュリティポリシーの置き換えが可能になります。

タスク

- デフォルトの証明書の置き換え
- 証明書リストに証明書を追加する
- 証明書リストから証明書を削除する
- セキュリティポリシーの更新

デフォルトの証明書の置き換え

次の手順でリスナーのデフォルトの証明書を置き換えできます。詳細については、「SSL 証明書」を参照してください。

コンソールを使用してデフォルトの証明書を変更するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[ロードバランサー] を選択します。
3. ロードバランサーを選択します。
4. [リスナーとルール] タブで、[プロトコル:ポート] 列のテキストを選択して、リスナーの詳細ページを開きます。
5. [証明書] タブで、[デフォルトを変更] を選択します。
6. [ACM および IAM 証明書] 表内の新しいデフォルト証明書を選択します。
7. [デフォルトとして保存] を選択します。

を使用してデフォルトの証明書を変更するには AWS CLI

modify-listener コマンドを使用します。

証明書リストに証明書を追加する

次の手順でリスナーの証明書リストに証明書を追加できます。最初に HTTPS リスナーを作成したときは、証明書リストは空です。1 つ以上の証明書を追加できます。デフォルトの証明書として置き換えても、この証明書が SNI プロトコルで使用されるように、デフォルトの証明書をオプションで追加できます。詳細については、「SSL 証明書」を参照してください。

コンソールを使用してデフォルトの証明書を変更するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[ロードバランサー] を選択します。
3. ロードバランサーを選択します。
4. [リスナーとルール] タブで、[プロトコル:ポート] 列のテキストを選択して、リスナーの詳細ページを開きます。
5. [証明書] タブで、[証明書の追加] を選択します。

6. [ACM および IAM 証明書] 表から、追加する証明書を選択し、[保留中として以下を含める] を選択します。
7. ACM または IAM が管理していない証明書がある場合は、[証明書のインポート] を選択し、フォームに記入し、[インポート] を選択します。
8. [保留中の証明書を追加] を選択します。

を使用して証明書リストに証明書を追加するには AWS CLI

add-listener-certificates コマンドを使用します。

証明書リストから証明書を削除する

次の手順で HTTPS リスナーの証明書リストから証明書を削除できます。HTTPS リスナーのデフォルトの証明書は削除するには、デフォルトの証明書の置き換え を参照してください。

コンソールを使用して証明書リストから証明書を削除するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[ロードバランサー] を選択します。
3. ロードバランサーを選択します。
4. [リスナーとルール] タブで、[プロトコル:ポート] 列のテキストを選択して、リスナーの詳細ページを開きます。
5. [証明書] タブで、証明書のチェックボックスを選択し、[削除] を選択します。
6. 確認を求められたら、**confirm** と入力し、[削除] を選択します。

を使用して証明書リストから証明書を削除するには AWS CLI

remove-listener-certificates コマンドを使用します。

セキュリティポリシーの更新

HTTPS リスナーを作成するときに、ニーズを満たすセキュリティポリシーを選択できます。新しいセキュリティのポリシーを追加したら、HTTPS リスナーを更新して新しいセキュリティポリシーを使用できます。Application Load Balancer は、カスタムセキュリティポリシーをサポートしていません。詳細については、「セキュリティポリシー」を参照してください。

Application Load Balancer での FIPS ポリシーの使用：

Application Load Balancer にアタッチされたすべてのセキュアリスナーは、FIPS セキュリティポリシーまたは非 FIPS セキュリティポリシーのいずれかを使用する必要があります。これらを混在させることはできません。既存の Application Load Balancer に FIPS 以外のポリシーを使用するリスナーが 2 つ以上あり、代わりにリスナーに FIPS セキュリティポリシーを使用させたい場合は、1 つしかないまですべてのリスナーを削除します。リスナーのセキュリティポリシーを FIPS に変更し、FIPS セキュリティポリシーを使用して追加のリスナーを作成します。または、FIPS セキュリティポリシーのみを使用して、新しいリスナーで新しい Application Load Balancer を作成することもできます。

コンソールを使用してセキュリティポリシーを更新するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[ロードバランサー] を選択します。
3. ロードバランサーを選択します。
4. [リスナーとルール] タブで、[プロトコル:ポート] 列のテキストを選択して、リスナーの詳細ページを開きます。
5. [詳細] ページで、[アクション]、[リスナーを編集] の順に選択します。
6. 「セキュアリスナー設定」セクションのセキュリティポリシー で、新しいセキュリティポリシーを選択します。
7. [変更の保存] をクリックします。

を使用してセキュリティポリシーを更新するには AWS CLI

modify-listener コマンドを使用します。

Application Load Balancer での TLS による相互認証

相互 TLS 認証は、トランスポートレイヤーセキュリティ (TLS) のバリエーションです。従来の TLS は、サーバーとクライアント間の安全な通信を確立し、サーバーはクライアントにアイデンティティを提供する必要があります。相互 TLS では、ロードバランサーは TLS のネゴシエート中にクライアントとサーバー間の相互認証をネゴシエートします。Application Load Balancer で相互 TLS を使用すると、認証管理が簡素化され、アプリケーションの負荷が軽減されます。

Application Load Balancer で相互 TLS を使用することで、ロードバランサーはクライアント認証を管理して、信頼できるクライアントのみがバックエンドアプリケーションと通信できるようにします。この機能を使用すると、Application Load Balancer は、サードパーティー認証局 (CA) からの

証明書を使用するか、必要に応じて失効チェックで AWS Private Certificate Authority (PCA) を使用してクライアントを認証します。Application Load Balancer は、クライアント証明書情報をバックエンドに渡します。バックエンドは、アプリケーションが認証に使用できます。Application Load Balancer で相互 TLS を使用すると、確立されたライブラリを使用する証明書ベースのエンティティに対して、組み込みのスケーラブルなマネージド認証を取得できます。

Application Load Balancer の相互 TLS には、X.509v3 クライアント証明書を検証するための次の 2 つのオプションがあります。

注： X.509v1 クライアント証明書はサポートされていません。

- 相互 TLS パススルー： 相互 TLS パススルーモードを使用すると、Application Load Balancer は HTTP ヘッダーを使用してクライアント証明書チェーン全体をターゲットに送信します。次に、クライアント証明書チェーンを使用して、対応する認証および認可ロジックをアプリケーションに実装できます。
- 相互 TLS 検証： 相互 TLS 検証モードを使用する場合、Application Load Balancer はロードバランサーが TLS 接続をネゴシエートするときに、クライアントに対して X.509 クライアント証明書認証を実行します。

パススルーを使用して Application Load Balancer で相互 TLS の使用を開始するには、クライアントからの証明書を受け入れるようにリスナーを設定するだけで済みます。検証で相互 TLS を使用するには、次の操作を行う必要があります。

- 新しい信頼ストアリソースを作成します。
- 認証局 (CA) バンドルと、オプションで失効リストをアップロードします。
- クライアント証明書を検証するように設定されたリスナーに信頼ストアをアタッチします。

Application Load Balancer で相互 TLS 検証モードを設定する step-by-step 手順については、「」を参照してくださいApplication Load Balancer での相互 TLS の設定。 Application Load Balancer

Application Load Balancer で相互 TLS の設定を開始する前に

Application Load Balancer で相互 TLS の設定を開始する前に、次の点に注意してください。

クォータ

Application Load Balancer には、 AWS アカウント内で使用されている信頼ストア、CA 証明書、および証明書失効リストの量に関連する特定の制限が含まれています。

詳細については、「Application Load Balancer のクォータ」を参照してください。

証明書の要件

Application Load Balancer は、相互 TLS 認証で使用される証明書に対して以下をサポートします。

- サポートされている証明書: X.509v3
- サポートされているパブリックキー: RSA 2K – 8K または ECDSA secp256r1、secp384r1、secp521r1
- サポートされている署名アルゴリズム: SHA256、384、512 with RSA/SHA256、384、512 with EC/SHA256、384、512 hash with RSASSA-PSS with MGF1

CA 証明書バンドル

認証局 (CA) バンドルには、次のものが適用されます。

- Application Load Balancer は、各認証局 (CA) 証明書バンドルをバッチとしてアップロードします。Application Load Balancer は、個々の証明書のアップロードをサポートしていません。新しい証明書を追加する必要がある場合は、証明書バンドルファイルをアップロードする必要があります。
- CA 証明書バンドルを置き換えるには、 ModifyTrustStore API を使用します。

パススルーの証明書の順序

相互 TLS パススルーを使用すると、Application Load Balancer はヘッダーを挿入して、クライアント証明書チェーンをバックエンドターゲットに提示します。プレゼンテーションの順序は、リーフ証明書で始まり、ルート証明書で終わります。

セッションの再開

Application Load Balancer で相互 TLS パススルーモードまたは検証モードを使用している間は、セッションの再開はサポートされていません。

HTTP ヘッダー

Application Load Balancer は、相互 TLS を使用してクライアント接続をネゴシエートするときに、`X-Amzn-Mtls`ヘッダーを使用して証明書情報を送信します。ヘッダーの詳細と例については、「」を参照してくださいHTTP ヘッダーと相互 TLS。

CA 証明書ファイル

CA 証明書ファイルは、次の要件を満たしている必要があります。

- 証明書ファイルは PEM (Privacy Enhanced Mail) 形式を使用する必要があります。

- 証明書の内容は、 -----BEGIN CERTIFICATE-----および の-----END CERTIFICATE-----境界内に囲む必要があります。
- コメントの前に#文字を付ける必要があります。
- 空白行は使用できません。

受け入れられない証明書の例 (無効):

```
# comments

Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 01
    Signature Algorithm: ecdsa-with-SHA384
        Issuer: C=US, O=EXAMPLE, OU=EXAMPLE, CN=EXAMPLE
        Validity
            Not Before: Jan 11 23:57:57 2024 GMT
            Not After : Jan 10 00:57:57 2029 GMT
        Subject: C=US, O=EXAMPLE, OU=EXAMPLE, CN=EXAMPLE
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (384 bit)
                pub:
                    00:01:02:03:04:05:06:07:08
                ASN1 OID: secp384r1
                NIST CURVE: P-384
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment, Certificate Sign, CRL Sign
            X509v3 Basic Constraints: critical
                CA:TRUE
            X509v3 Subject Key Identifier:
                00:01:02:03:04:05:06:07:08
            X509v3 Subject Alternative Name:
                URI:EXAMPLE.COM
    Signature Algorithm: ecdsa-with-SHA384
         00:01:02:03:04:05:06:07:08
-----BEGIN CERTIFICATE-----
Base64-encoded certificate
-----END CERTIFICATE-----
```

受け入れられる証明書の例 (有効):

1. 単一証明書 (PEM エンコード):

```
# comments
-----BEGIN CERTIFICATE-----
Base64-encoded certificate
-----END CERTIFICATE-----
```

2. 複数の証明書 (PEM エンコード):

```
# comments
-----BEGIN CERTIFICATE-----
Base64-encoded certificate
-----END CERTIFICATE-----
# comments
-----BEGIN CERTIFICATE-----
Base64-encoded certificate
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
Base64-encoded certificate
-----END CERTIFICATE-----
```

HTTP ヘッダーと相互 TLS

このセクションでは、相互 TLS を使用してクライアントとの接続をネゴシエートするときに Application Load Balancer が証明書情報を送信するために使用する HTTP ヘッダーについて説明します。Application Load Balancer が使用する特定の`X-Amzn-Mtls`ヘッダーは、指定した相互 TLS モードによって異なります。パススルーモードまたは検証モードです。

Application Load Balancer でサポートされている他の HTTP ヘッダーについては、「」を参照してくださいHTTP ヘッダーと Application Load Balancer。

パススルーモードの HTTP ヘッダー

パススルーモードでの相互 TLS の場合、Application Load Balancer は次のヘッダーを使用します。

X-Amzn-Mtls-Clientcert

このヘッダーには、接続に表示されるクライアント証明書チェーン全体の URL エンコードされた PEM 形式が含まれており、安全文字+=/として が使用されています。

ヘッダーの内容の例：

```
X-Amzn-Mtls-Clientcert: -----BEGIN%20CERTIFICATE-----%0AMIID<...reduced...>do0g
%3D%3D%0A-----END%20CERTIFICATE-----%0A-----BEGIN%20CERTIFICATE-----
%0AMIID1<...reduced...>3eZlyKA%3D%3D%0A-----END%20CERTIFICATE-----%0A
```

検証モードの HTTP ヘッダー

検証モードでの相互 TLS の場合、Application Load Balancer は次のヘッダーを使用します。

X-Amzn-Mtls-Clientcert-Serial-Number

このヘッダーには、リーフ証明書のシリアル番号の 16 進数表現が含まれています。

ヘッダーの内容例：

```
X-Amzn-Mtls-Clientcert-Serial-Number: 03A5B1
```

X-Amzn-Mtls-Clientcert-Issuer

このヘッダーには、発行者の識別名 (DN) の RFC2253 文字列表現が含まれています。

ヘッダーの内容例：

```
X-Amzn-Mtls-Clientcert-Issuer:
 CN=rootcamtls.com,OU=rootCA,O=mTLS,L=Seattle,ST=Washington,C=US
```

X-Amzn-Mtls-Clientcert-Subject

このヘッダーには、サブジェクトの識別名 (DN) の RFC2253 文字列表現が含まれています。

ヘッダーの内容例：

```
X-Amzn-Mtls-Clientcert-Subject: CN=client_.com,OU=client-3,O=mTLS,ST=Washington,C=US
```

X-Amzn-Mtls-Clientcert-Validity

このヘッダーには、ISO8601 形式の `notBefore`および `notAfter`日付が含まれます。

ヘッダーの内容例：

```
X-Amzn-Mtls-Clientcert-Validity:
 NotBefore=2023-09-21T01:50:17Z;NotAfter=2024-09-20T01:50:17Z
```

X-Amzn-Mtls-Clientcert-Leaf

このヘッダーには、安全な文字+=/として、リーフ証明書の URL エンコードされた PEM 形式が含まれています。

ヘッダーコンテンツの例：

```
X-Amzn-Mtls-Clientcert-Leaf: -----BEGIN%20CERTIFICATE-----%0AMIIG<...reduced...>NmrUlw
%0A-----END%20CERTIFICATE-----%0A
```

Application Load Balancer での相互 TLS の設定

このセクションでは、Application Load Balancer での認証に相互 TLS 検証モードを設定する手順について説明します。

相互 TLS パススルーモードを使用するには、クライアントからの証明書を受け入れるようにリスナーを設定するだけで済みます。相互 TLS パススルーを使用すると、Application Load Balancer は HTTP ヘッダーを使用してクライアント証明書チェーン全体をターゲットに送信します。これにより、対応する認証および認可ロジックをアプリケーションに実装できます。詳細については、のCreate an HTTPS Listener for Your Application Load Balancerを参照してください。

検証モードで相互 TLS を使用すると、ロードバランサーが TLS 接続をネゴシエートするときに、Application Load Balancer はクライアントの X.509 クライアント証明書認証を実行します。

相互 TLS 検証モードを利用するには、以下を実行します。

- 新しい信頼ストアリソースを作成します。
- 認証局 (CA) バンドルをアップロードし、オプションで失効リストをアップロードします。
- クライアント証明書を検証するように設定されたリスナーに信頼ストアをアタッチします。

このセクションの手順に従って、 の Application Load Balancer で相互 TLS 検証モードを設定します AWS Management Console。コンソールの代わりに API オペレーションを使用して相互 TLS を設定するには、Application Load Balancer API リファレンスガイド」を参照してください。

タスク

- トラストストアを作成する
- トラストストアを関連付ける
- トラストストアの詳細を表示する

- トラストストアを変更する
- 信頼ストアを削除する

トラストストアを作成する

信頼ストアを作成する方法は 3 つあります。Application Load Balancer を作成する場合、セキュアなリスナーを作成する場合、および Trust Store コンソールを使用する方法です。ロードバランサーまたはリスナーの作成時に信頼ストアを追加すると、信頼ストアは自動的に新しいリスナーに関連付けられます。Trust Store コンソールを使用してトラストストアを作成する場合は、自分でリスナーに関連付ける必要があります。

このセクションでは、Trust Store コンソールを使用した信頼ストアの作成について説明しますが、Application Load Balancer またはリスナーの作成時に使用する手順は同じです。詳細については、「ロードバランサーとリスナーを設定する」および「HTTPS リスナーを追加する」を参照してください。

前提条件:

- トラストストアを作成するには、認証局 (CA) からの証明書バンドルが必要です。

コンソールを使用して信頼ストアを作成するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、トラストストア を選択します。
3. 信頼ストアの作成 を選択します。
4. トラストストアの設定
   a. 信頼ストア名 には、信頼ストアの名前を入力します。
   b. 認証機関バンドルには、トラストストアで使用する ca 証明書バンドルへの Amazon S3 パスを入力します。

      オプション： オブジェクトバージョンを使用して、ca 証明書バンドルの以前のバージョンを選択します。それ以外の場合は、現在のバージョンが使用されます。
5. 失効の場合、オプションで証明書失効リストを信頼ストアに追加できます。
   - 証明書失効リストに、信頼ストアで使用する証明書失効リストへの Amazon S3 パスを入力します。

オプション： オブジェクトバージョンを使用して、証明書失効リストの以前のバージョンを選択します。それ以外の場合は、現在のバージョンが使用されます。

6. トラストストアタグには、オプションで最大 50 個のタグを入力してトラストストアに適用できます。
7. 信頼ストアの作成 を選択します。

トラストストアを関連付ける

信頼ストアを作成したら、Application Load Balancer が信頼ストアの使用を開始する前に、そのストアをリスナーに関連付ける必要があります。各セキュアリスナーに関連付けることができる信頼ストアは 1 つだけですが、1 つの信頼ストアを複数のリスナーに関連付けることができます。

このセクションでは、信頼ストアを既存のリスナーに関連付ける方法について説明します。または、Application Load Balancer またはリスナーの作成時に信頼ストアを関連付けることもできます。詳細については、「ロードバランサーとリスナーを設定する」および「HTTPS リスナーを追加する」を参照してください。

コンソールを使用して信頼ストアを関連付けるには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[ロードバランサー] を選択します。
3. ロードバランサーを選択すると、その詳細ページが表示されます。
4. リスナーとルールタブで、プロトコル：ポート列のリンクを選択して、セキュアリスナーの詳細ページを開きます。
5. セキュリティタブで、セキュアリスナー設定の編集 を選択します。
6. (オプション) 相互 TLS が有効になっていない場合は、クライアント証明書処理で相互認証 (mTLS) を選択し、信頼ストア で検証を選択します。
7. 信頼ストア で、作成した信頼ストアを選択します。
8. [変更の保存] をクリックします。

トラストストアの詳細を表示する

CA 証明書バンドル

CA 証明書バンドルは、信頼ストアの必須コンポーネントです。これは、認証機関によって検証された信頼できるルート証明書と中間証明書のコレクションです。これらの検証済み証明書により、クライアントは提示される証明書がロードバランサーによって所有されていることを信頼できます。

信頼ストアの現在の CA 証明書バンドルの内容はいつでも表示できます。

CA 証明書バンドルを表示する

コンソールを使用して CA 証明書バンドルを表示するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、信頼ストア を選択します。
3. 信頼ストアを選択して、詳細ページを表示します。
4. アクション を選択し、CA バンドル を取得します。
5. リンクの共有、または のダウンロードを選択します。

証明書失効リスト

オプションで、トラストストアの証明書失効リストを作成できます。失効リストは認証機関によってリリースされ、取り消された証明書のデータが含まれます。Application Load Balancer は、PEM 形式の証明書失効リストのみをサポートします。

証明書失効リストが信頼ストアに追加されると、失効 ID が与えられます。失効 IDsトラストストアに追加された失効リストごとに増加し、変更することはできません。証明書失効リストが信頼ストアから削除された場合、その失効 ID も削除され、信頼ストアの存続期間中は再利用されません。

> **Note**
>
> Application Load Balancer は、証明書失効リスト内で負のシリアル番号を持つ証明書を取り消すことはできません。

証明書失効リストを表示する

コンソールを使用して失効リストを表示するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、信頼ストア を選択します。
3. 信頼ストアを選択して、詳細ページを表示します。
4. 証明書失効リスト タブで、アクション を選択し、失効リスト を取得します。
5. リンクの共有 または ダウンロード を選択します。

トラストストアを変更する

トラストストアに含めることができる CA 証明書バンドルは一度に 1 つだけですが、トラストストアの作成後はいつでも CA 証明書バンドルを置き換えることができます。

CA 証明書バンドルを置き換える

コンソールを使用して CA 証明書バンドルを置き換えるには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、信頼ストア を選択します。
3. 信頼ストアを選択して、詳細ページを表示します。
4. アクション を選択し、CA バンドル を置き換えます。
5. CA バンドルの置き換えページで、認証機関バンドルの下に、目的の CA バンドルの Amazon S3 の場所を入力します。
6. (オプション) オブジェクトバージョンを使用して、証明書失効リストの以前のバージョンを選択します。それ以外の場合は、現在のバージョンが使用されます。
7. CA バンドルの置き換え を選択します。

証明書失効リストを追加する

コンソールを使用して失効リストを追加するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、信頼ストア を選択します。
3. 信頼ストアを選択して、詳細ページを表示します。

4. 証明書失効リスト タブで、アクション を選択し、失効リスト を追加します。
5. 「失効リストの追加」ページの「証明書失効リスト」に、目的の証明書失効リストの Amazon S3 の場所を入力します。
6. (オプション) オブジェクトバージョンを使用して、証明書失効リストの以前のバージョンを選択します。それ以外の場合は、現在のバージョンが使用されます。
7. 失効リストの追加を選択します。

証明書失効リストを削除する

コンソールを使用して失効リストを削除するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、トラストストア を選択します。
3. 信頼ストアを選択して、詳細ページを表示します。
4. 証明書失効リストタブで、アクション を選択し、失効リスト を削除します。
5. を入力して削除を確認しますconfirm。
6. [削除] を選択します。

信頼ストアを削除する

トラストストアに を使用しなくなったら、削除できます。

注： 現在リスナーに関連付けられている信頼ストアは削除できません。

コンソールを使用して信頼ストアを削除するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、信頼ストア を選択します。
3. 信頼ストアを選択して、詳細ページを表示します。
4. アクション を選択し、信頼ストア を削除します。
5. を入力して削除を確認しますconfirm。
6. 削除を選択する

Application Load Balancer の接続ログ

Elastic Load Balancing は、Application Load Balancer に送信されたリクエストに関する属性をキャプチャする接続ログを提供します。接続ログには、クライアントの IP アドレスとポート、クライアント証明書情報、接続結果、使用されている TLS 暗号などの情報が含まれます。これらの接続ログを使用して、リクエストパターンやその他の傾向を確認できます。

接続ログの詳細については、「」を参照してください。 Application Load Balancer の接続ログ

Application Load Balancer を使用してユーザーを認証する

Application Load Balancer を設定して、ユーザーがアプリケーションにアクセスしたときに安全に認証できます。これにより、アプリケーションがビジネスロジックに集中できるように、ユーザーを認証する作業をロードバランサーに任せることができます。

次にユースケースがサポートされています。

- OpenID Connect (OIDC) に準拠する ID プロバイダー (IdP) を使用してユーザーを認証します。
- Amazon や Google IdPsなどのソーシャル を通じて FaceBook、Amazon Cognito でサポートされているユーザープールを通じてユーザーを認証します。
- 企業アイデンティティを介して、SAML、OpenID Connect (OIDC) や OAuth、さらには Amazon Cognito がサポートするユーザープールを経由して、ユーザーを認証します。

OIDC 準拠 IdP を使用する準備を整える

Application Load Balancer で OIDC 準拠 IdP を使用している場合は、以下を実行します。

- IdP に新しい OIDC アプリを作成します。IdP の DNS はパブリックに解決可能でなければなりません。
- クライアント ID およびクライアントシークレットを設定する必要があります。
- その IdP によって発行される次のエンドポイントを取得します。認証、トークン、およびユーザー情報。この情報は config で確認できます。
- IdP エンドポイント証明書は、信頼できる公開認証機関によって発行されている必要があります。
- エンドポイントのDNSエントリは、プライベートIPアドレスに解決される場合でも、パブリックに解決できる必要があります。

- IdP アプリで次のリダイレクト URL のいずれかを許可します。ユーザーが使用するものであればどれでも構いません。ここで DNS はロードバランサーのドメイン名であり、CNAME はアプリケーションの DNS エイリアスです。
  - https://*DNS*/oauth2/idpresponse
  - https://*CNAME*/oauth2/idpresponse

Amazon Cognito を使用する準備を行う

利用可能なリージョン

Application Load Balancer の Amazon Cognito 統合は、次のリージョンで利用できます。

- 米国東部 (バージニア北部)
- 米国東部 (オハイオ)
- 米国西部 (北カリフォルニア)
- 米国西部 (オレゴン)
- カナダ (中部)
- 欧州 (ストックホルム)
- 欧州 (ミラノ)
- 欧州 (フランクフルト)
- 欧州 (チューリッヒ)
- 欧州 (アイルランド)
- 欧州 (ロンドン)
- 欧州 (パリ)
- 南米 (サンパウロ)
- アジアパシフィック (東京)
- アジアパシフィック (ソウル)
- アジアパシフィック (大阪)
- アジアパシフィック (ムンバイ)
- アジアパシフィック (シンガポール)
- アジアパシフィック (シドニー)
- アジアパシフィック (ジャカルタ)

- 中東 (アラブ首長国連邦)
- 中東 (バーレーン)
- アフリカ (ケープタウン)
- イスラエル (テルアビブ)

Application Load Balancer でAmazon Cognito ユーザープールを使用している場合は、以下を実行します。

- ユーザープールを作成します。詳細については、Amazon Cognito デベロッパーガイドの Amazon Cognito user poolsを参照してください。
- ユーザープールのクライアントを作成します。クライアントシークレットを生成し、コード付与フローを使用し、ロードバランサーが使用するものと同じ OAuth スコープをサポートするように、クライアントを設定する必要があります。詳細については、Amazon Cognito デベロッパーガイドの Configuring a user pool app client を参照してください。
- ユーザープールのドメインを作成します。詳細については、Amazon Cognito デベロッパーガイドの Adding a Domain name for your user pool を参照してください。
- リクエストされたスコープで ID トークンが返されていることを確認します。たとえば、デフォルトのスコープ `openid` では ID トークンが返されますが、`aws.cognito.signin.user.admin` スコープでは返されません。

  注： Application Load Balancer は、Amazon Cognito によって発行されたカスタマイズされたアクセストークンをサポートしていません。詳細については、Amazon Cognito デベロッパーガイド」の「トークン生成前」を参照してください。
- ソーシャルまたは企業 IdP と連携するには、フェデレーションセクションで IdP を有効にします。詳細については、Amazon Cognito デベロッパーガイドの Add social sign-in to a user pool または Add sign-in with a SAML IdP to a user pool を参照してください。
- Amazon Cognito のコールバック URL フィールドで次のリダイレクト URL のいずれかを許可します。ここで DNS はロードバランサーのドメイン名であり、CNAME はアプリケーションの DNS エイリアス (使用している場合) です。
  - https://*DNS*/oauth2/idpresponse
  - https://*CNAME*/oauth2/idpresponse
- IdP アプリのコールバック URL でユーザープールドメインを許可します。IdP の形式を使用します。以下に例を示します。
  - https://*domain-prefix*.auth.*region*.amazoncognito.com/saml2/idpresponse

- https://*user-pool-domain*/oauth2/idpresponse

アプリクライアント設定のコールバック URL はすべて小文字を使用する必要があります。

特定ユーザーが Amazon Cognito を使用してユーザーを認証するようにロードバランサーを設定するには、`cognito-idp:DescribeUserPoolClient` アクションを呼び出すアクセス許可をユーザーに付与する必要があります。

Amazon を使用するための準備 CloudFront

Application Load Balancer の前に CloudFront ディストリビューションを使用している場合は、次の設定を有効にします。

- 転送リクエストヘッダー (すべて) — CloudFront が認証されたリクエストのレスポンスをキャッシュしないようにします。これにより、認証セッションが期限切れになった後にキャッシュから処理されるのを防ぎます。または、キャッシュが有効になっている間にこのリスクを軽減するために、 CloudFront ディストリビューションの所有者は、認証 Cookie の有効期限が切れる前に time-to-live (TTL) 値を有効期限に設定することもできます。
- クエリ文字列の転送とキャッシュ (すべて) — ロードバランサーが、IdP を使用してユーザーを認証するために必要なクエリ文字列パラメータにアクセスできるようにします。
- Cookie 転送 (すべて) — がすべての認証 Cookie をロードバランサー CloudFront に転送します。

ユーザー認証を設定する

1 つ以上のリスナールールの認証アクションを作成して、ユーザー認証を設定します。`authenticate-cognito` および `authenticate-oidc` アクションタイプは HTTPS リスナーでのみサポートされています。対応するフィールドの説明については、Elastic Load Balancing API リファレンスバージョン 2015-12-01 AuthenticateOidcActionConfigのAuthenticateCognitoActionConfig「」および「」を参照してください。

ロードバランサーは、認証ステータスを維持するためにセッション Cookie をクライアントに送信します。ユーザー認証には HTTPS リスナーが必要であるため、この Cookie には常に `secure` 属性が含まれます。この Cookie には、CORS (クロスオリジンリソース共有) リクエストを持つ `SameSite=None` 属性が含まれています。

独立したクライアント認証を必要とする複数のアプリケーションをサポートしているロードバランサーについては、認証アクションがある各リスナールールに一意の cookie 名が必要です。これは、

クライアントが常に IdP で認証されてから、ルールで指定されているターゲットグループにルーティングされることを確実にします。

Application Load Balancer は、URL エンコードされた Cookie 値をサポートしていません。

デフォルトでは、SessionTimeout フィールドは 7 日に設定されています。セッションをこれより短くするばあいは、セッションタイムアウトを最短 1 秒に設定できます。詳細については、「セッションタイムアウト」を参照してください。

アプリケーションの必要に応じて、OnUnauthenticatedRequest フィールドを設定します。以下に例を示します。

- ユーザーがソーシャル ID または企業 ID を使用してログインする必要があるアプリケーション — デフォルトのオプション authenticate でサポートされています。ユーザーがログインしていない場合は、ロードバランサーはリクエストを IdP 認証エンドポイントにリダイレクトし、IdP によってユーザーにユーザーインターフェイスを使用したログインを求めるプロンプトが表示されます。
- ログインしているユーザーにはパーソナライズされたビューを、ログインしていないユーザーには一般的なビューを提供するアプリケーション — このタイプのアプリケーションをサポートするには、allow オプションを使用します。ユーザーがログインしている場合、ロードバランサーがユーザークレームを提供するため、アプリケーションはパーソナライズされたビューを提供できます。ユーザーがログインしていない場合、ロードバランサーはリクエストをユーザークレームなしで転送するため、アプリケーションは一般的なビューを提供できます。
- 数秒ごとにロード JavaScript される単一ページアプリケーション — denyオプションを使用すると、ロードバランサーは認証情報を持たない AJAX 呼び出しに HTTP 401 Unauthorized エラーを返します。ただし、ユーザーに有効期限の切れた認証情報がある場合は、クライアントを IdP 認証エンドポイントにリダイレクトします。

ロードバランサーは、IdP トークンのエンドポイント (TokenEndpoint) および IdP ユーザー情報エンドポイント (UserInfoEndpoint) と通信できる必要があります。Application Load Balancer は、これらのエンドポイントと通信する場合にのみ IPv4 をサポートします。IdP がパブリックアドレスを使用している場合は、ロードバランサーのセキュリティグループと VPC ACLs がエンドポイントへのアクセスを許可していることを確認します。内部ロードバランサーまたは IP アドレスタイプ を使用する場合dualstack-without-public-ipv4、NAT ゲートウェイはロードバランサーがエンドポイントと通信できるようにします。詳細については、Amazon VPC ユーザーガイド の NAT ゲートウェイベーシック を参照してください。

次の create-rule コマンドを使用して、ユーザー認証を設定します。

```
aws elbv2 create-rule --listener-arn listener-arn --priority 10 \
--conditions Field=path-pattern,Values="/login" --actions file://actions.json
```

actions.json ファイルの例を次に示します。authenticate-oidc アクションと forward アクション AuthenticationRequestExtraParams を使用すると、認証中に IdP に追加のパラメーターを渡すことができます。ID プロバイダーから提供されたドキュメントに従って、サポートされているフィールドを確認してください。

```
[{
    "Type": "authenticate-oidc",
    "AuthenticateOidcConfig": {
        "Issuer": "https://idp-issuer.com",
        "AuthorizationEndpoint": "https://authorization-endpoint.com",
        "TokenEndpoint": "https://token-endpoint.com",
        "UserInfoEndpoint": "https://user-info-endpoint.com",
        "ClientId": "abcdefghijklmnopqrstuvwxyz123456789",
        "ClientSecret": "1234567890123456789012345678901234567890",
        "SessionCookieName": "my-cookie",
        "SessionTimeout": 3600,
        "Scope": "email",
        "AuthenticationRequestExtraParams": {
            "display": "page",
            "prompt": "login"
        },
        "OnUnauthenticatedRequest": "deny"
    },
    "Order": 1
},
{
    "Type": "forward",
    "TargetGroupArn": "arn:aws:elasticloadbalancing:region-code:account-
id:targetgroup/target-group-name/target-group-id",
    "Order": 2
}]
```

actions.json アクションと authenticate-cognito アクションを指定する forward ファイルの例を次に示します。

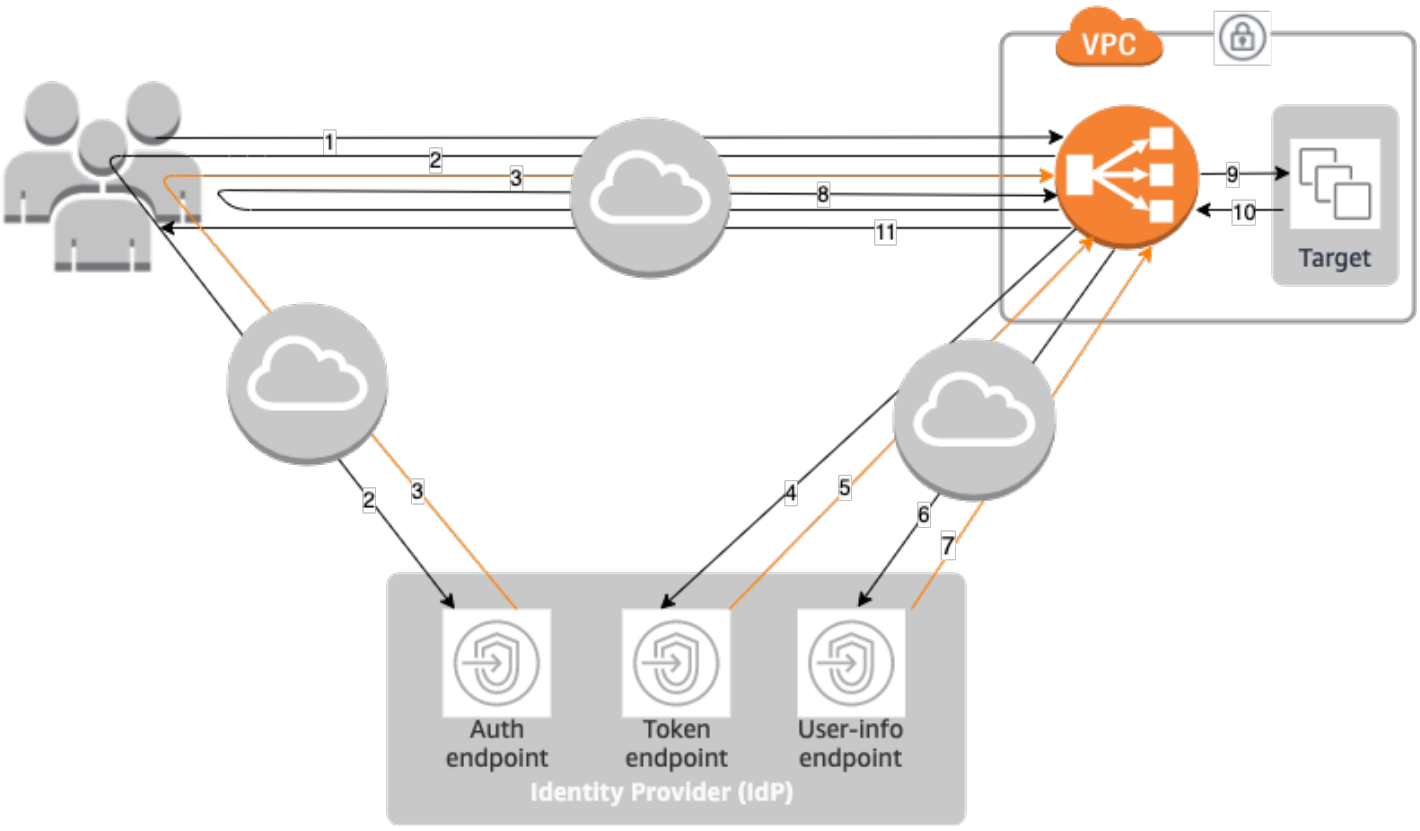
```
[{
    "Type": "authenticate-cognito",
    "AuthenticateCognitoConfig": {
```

```
        "UserPoolArn": "arn:aws:cognito-idp:region-code:account-id:userpool/user-pool-
id",
        "UserPoolClientId": "abcdefghijklmnopqrstuvwxyz123456789",
        "UserPoolDomain": "userPoolDomain1",
        "SessionCookieName": "my-cookie",
        "SessionTimeout": 3600,
        "Scope": "email",
        "AuthenticationRequestExtraParams": {
            "display": "page",
            "prompt": "login"
        },
        "OnUnauthenticatedRequest": "deny"
    },
    "Order": 1
},
{
    "Type": "forward",
    "TargetGroupArn": "arn:aws:elasticloadbalancing:region-code:account-
id:targetgroup/target-group-name/target-group-id",
    "Order": 2
}]
```

詳細については、「リスナールール」を参照してください。

認証のフロー

次のネットワーク図は、Application Load Balancer が OIDC を使用してユーザーを認証する方法を示しています。



下の番号付き項目は、前のネットワーク図に示した要素を強調表示し、説明しています。

1. ユーザーは、Application Load Balancer の背後にホストされているウェブサイトに HTTPS リクエストを送信します。認証アクションを持つルールの条件が満たされている場合、ロードバランサーはリクエストヘッダーに認証セッション Cookie があるかどうかを確認します。
2. Cookie が存在しない場合、ロードバランサーはユーザーを IdP 認証エンドポイントにリダイレクトし、IdP でユーザーを認証できるようにします。
3. ユーザーが認証されると、IdP は認可付与コードを伴ったユーザーをロードバランサーにリダイレクトして戻します。
4. ロードバランサーは認可付与コードを IdP トークンエンドポイントに示します。
5. 有効な認可付与コードを受け取ると、IdP は ID トークンとアクセストークンをApplication Load Balancer に提供します。
6. 次に、Application Load Balancer がアクセストークンをユーザー情報エンドポイントに送信します。
7. ユーザー情報エンドポイントは、ユーザーの要求に対してアクセストークンを交換します。
8. Application Load Balancer は、`AWSELB` 認証セッション cookie を持つユーザーを元の URI にリダイレクトします。ほとんどのブラウザでは Cookie のサイズを 4K に制限しているため、ロー

ドバランサーはサイズが 4K を超える Cookie を複数の Cookie に分割します。IdP から受け取ったユーザークレームとアクセストークンの合計サイズが 11K バイトを超える場合、ロードバランサーは HTTP 500 エラーをクライアントに返し、`ELBAuthUserClaimsSizeExceeded` メトリクスを増分します。

9. Application Load Balancer は Cookie を検証し、ユーザー情報を `X-AMZN-OIDC-*` HTTP ヘッダー設定のターゲットに転送します。詳細については、「ユーザークレームのエンコードと署名の検証」を参照してください。
10. ターゲットは応答をApplication Load Balancer に戻します。
11. Application Load Balancer は、最終応答をユーザーに送信します。

新しいリクエストはステップ 1 ~ 11 を通過し、後続のリクエストはステップ 9 ~ 11 を通過します。つまり、cookie の有効期限が切れていない限り、後続のすべてのリクエストはステップ 9 から始まります。

ユーザーが IdP で認証された後、`AWSALBAuthNonce` cookie がリクエストヘッダーに追加されます。これによって、Application Load Balancer が IdP からのリダイレクトリクエストを処理する方法が変わることはありません。

IdP によって ID トークンに有効な更新トークンが提供されている場合、ロードバランサーは更新トークンを保存し、アクセストークンの有効期限が切れるたびにこれを使用して、セッションがタイムアウトするか、IdP の更新に失敗するまで、ユーザークレームを更新します。ユーザーがログアウトすると、更新は失敗し、ロードバランサーはユーザーを IdP 認証エンドポイントにリダイレクトします。これにより、ロードバランサーはユーザーがログアウト後にセッションを削除できます。詳細については、「セッションタイムアウト」を参照してください。

> **Note**
>
> cookie の有効期限は、認証セッションの有効期限とは異なります。cookie の有効期限は cookie の属性であり、7 日に設定されています。認証セッションの実際の長さは、認証機能用に Application Load Balancer で設定されたセッションタイムアウトによって決まります。このセッションタイムアウトは、認証 cookie 値に含まれ、暗号化されます。

ユーザークレームのエンコードと署名の検証

ロードバランサーがユーザーの認証に成功すると、IdP から受け取ったユーザークレームがターゲットに送信されます。ロードバランサーはユーザークレームに署名するため、アプリケーションは署名

の検証およびそのクレームがロードバランサーから送信されたものであることの検証を行うことができます。

ロードバランサーは以下の HTTP ヘッダーを追加します。

`x-amzn-oidc-accesstoken`

トークンエンドポイントからのアクセストークン (プレーンテキスト)。

`x-amzn-oidc-identity`

ユーザー情報エンドポイントからの件名フィールド (sub) (プレーンテキスト)。

注: サブクレームは、特定のユーザーを識別する最良の方法です。

`x-amzn-oidc-data`

ユーザークレーム (JSON ウェブトークン (JWT) 形式)

アクセストークンとユーザーの要求が、ID トークンと異なります。アクセストークンとユーザーの要求は、サーバーリソースへのアクセスのみを許可しますが、ID トークンはユーザーを認証するための追加情報を保持しています。Application Load Balancer は、ユーザーを認証するときに新しいアクセストークンを作成し、アクセストークンとクレームのみをバックエンドに渡しますが、ID トークン情報は渡しません。

これらのトークンは JWT 形式に従いますが、ID トークンではありません。JWT 形式には、base64 URL エンコードされたヘッダー、ペイロード、および署名が含まれ、末尾にパディング文字が含まれます。Application Load Balancer は ES256 (P-256 と SHA256 を使用する ECDSA) を使用して JWT 署名を生成します。

この JWT ヘッダーは、次のフィールドを持つ JSON オブジェクトです。

```
{
   "alg": "algorithm",
   "kid": "12345678-1234-1234-1234-123456789012",
   "signer": "arn:aws:elasticloadbalancing:region-code:account-id:loadbalancer/
app/load-balancer-name/load-balancer-id",
   "iss": "url",
   "client": "client-id",
   "exp": "expiration"
}
```

この JWT ペイロードは、IdP ユーザー情報エンドポイントから受け取ったユーザークレームを含む JSON オブジェクトです。

```
{
   "sub": "1234567890",
   "name": "name",
   "email": "alias@example.com",
   ...
}
```

ロードバランサーではユーザークレームが暗号化されないため、HTTPS を使用するようにターゲットグループを設定することをお勧めします。HTTP を使用するようにターゲットグループを設定する場合は、ロードバランサーへのトラフィックをセキュリティグループを使用するトラフィックのみに制限してください。

セキュリティを確保するには、クレームに基づいて認証を行う前に署名を検証し、JWT ヘッダーの `signer`フィールドに予想される Application Load Balancer ARN が含まれていることを確認する必要があります。

パブリックキーを取得するには、JWT ヘッダーからキー ID を取得し、それを使用して次のエンドポイントからパブリックキーを検索します: それぞれの AWS リージョンの場合、エンドポイントは次のとおりです:

```
https://public-keys.auth.elb.region.amazonaws.com/key-id
```

の場合 AWS GovCloud (US)、エンドポイントは次のとおりです。

```
https://s3-us-gov-west-1.amazonaws.com/aws-elb-public-keys-prod-us-gov-west-1/key-id
https://s3-us-gov-east-1.amazonaws.com/aws-elb-public-keys-prod-us-gov-east-1/key-id
```

次の例は、Python 3.x でキー ID、公開キー、およびペイロードを取得する方法を示しています。

```
import jwt
import requests
import base64
import json

# Step 1: Validate the signer
expected_alb_arn = 'arn:aws:elasticloadbalancing:region-code:account-id:loadbalancer/
app/load-balancer-name/load-balancer-id'
```

```
encoded_jwt = headers.dict['x-amzn-oidc-data']
jwt_headers = encoded_jwt.split('.')[0]
decoded_jwt_headers = base64.b64decode(jwt_headers)
decoded_jwt_headers = decoded_jwt_headers.decode("utf-8")
decoded_json = json.loads(decoded_jwt_headers)
received_alb_arn = decoded_json['signer']

assert expected_alb_arn == received_alb_arn, "Invalid Signer"

# Step 2: Get the key id from JWT headers (the kid field)
kid = decoded_json['kid']

# Step 3: Get the public key from regional endpoint
url = 'https://public-keys.auth.elb.' + region + '.amazonaws.com/' + kid
req = requests.get(url)
pub_key = req.text

# Step 4: Get the payload
payload = jwt.decode(encoded_jwt, pub_key, algorithms=['ES256'])
```

次の例は、Python 2.7 でキー ID、公開キー、およびペイロードを取得する方法を示しています。

```
import jwt
import requests
import base64
import json

# Step 1: Validate the signer
expected_alb_arn = 'arn:aws:elasticloadbalancing:region-code:account-id:loadbalancer/
app/load-balancer-name/load-balancer-id'

encoded_jwt = headers.dict['x-amzn-oidc-data']
jwt_headers = encoded_jwt.split('.')[0]
decoded_jwt_headers = base64.b64decode(jwt_headers)
decoded_json = json.loads(decoded_jwt_headers)
received_alb_arn = decoded_json['signer']

assert expected_alb_arn == received_alb_arn, "Invalid Signer"

# Step 2: Get the key id from JWT headers (the kid field)
kid = decoded_json['kid']
```

```
# Step 3: Get the public key from regional endpoint
url = 'https://public-keys.auth.elb.' + region + '.amazonaws.com/' + kid
req = requests.get(url)
pub_key = req.text

# Step 4: Get the payload
payload = jwt.decode(encoded_jwt, pub_key, algorithms=['ES256'])
```

考慮事項

- これらの例では、トークンの署名を使用して発行者の署名を検証する方法については説明していません。
- 標準ライブラリは、JWT 形式の Application Load Balancer 認証トークンに含まれているパディングと互換性がないことにご注意ください。

タイムアウト

セッションタイムアウト

更新トークンとセッションタイムアウトは連携して次のように動作します。

- セッションタイムアウトがアクセストークンの有効期限より短い場合、ロードバランサーはセッションタイムアウトを優先させます。IdP とのセッションがまだアクティブであれば、ユーザーは再度ログインするように求められないことがあります。それ以外の場合、ユーザーはログインにリダイレクトされます。
  - IdP セッションタイムアウトが Application Load Balancer セッションタイムアウトよりも長い場合、ユーザーは再ログインするために認証情報を入力する必要はありません。代わりに、IdP は新しい認可付与コードを使用して Application Load Balancer にリダイレクトします。認可コードは、再ログインがない場合でも、一度の使用となります。
  - IdP セッションタイムアウトが Application Load Balancer セッションタイムアウト以下の場合、ユーザーは再ログインするための認証情報を指定する必要があります。再ログイン後、IdP は新しい認可コードを使用して Application Load Balancer にリダイレクトし、残りの認証フローはリクエストがバックエンドに到達するまで続行されます。
- セッションタイムアウトがアクセストークンの有効期限よりも長く、IdP が更新トークンをサポートしていない場合、ロードバランサーは認証セッションをタイムアウトまで維持します。その後、ユーザーが再びログインします。

- セッションタイムアウトがアクセストークンの有効期限よりも長く、IdP が更新トークンをサポートしている場合、ロードバランサーはアクセストークンの有効期限が切れるたびにユーザーセッションを更新します。ロードバランサーは、認証セッションがタイムアウトした後、または更新フローに失敗した場合にのみ、ユーザーに再度ログインさせます。

クライアントログインのタイムアウト

クライアントは 15 分以内に認証プロセスを開始して完了する必要があります。クライアントは 15 分以内に認証を完了できなかった場合、ロードバランサーから HTTP 401 エラーを受信します。このタイムアウトは変更または削除できません。

例えば、ユーザーが Application Load Balancer を使用してログインページをロードする場合、15 分以内にログインプロセスを完了する必要があります。15 分間のタイムアウトの期限れ後にユーザーが待機してログインしようとすると、ロードバランサーが HTTP 401 エラーを返します。ユーザーはページを更新して、もう一度ログインする必要があります。

認証ログアウト

アプリケーションで認証済みユーザーをログアウトさせる必要がある場合、認証セッション Cookie の有効期限を -1 に設定し、クライアントを IdP ログアウトエンドポイントにリダイレクト (IdP でサポートされている場合) する必要があります。ユーザーが削除された Cookie を再利用しないようにするには、アクセストークンの有効期限を問題のない範囲でできるだけ短く設定することをお勧めします。有効期限切れのアクセストークンと null 以外の更新トークンがあるセッション Cookie を持つロードバランサーをクライアントが提供する場合、ロードバランサーは IdP に接続してユーザーがまだログインしているかどうかを判別します。

クライアントのログアウトランディングページは、認証されていないページです。つまり、認証を必要とする Application Load Balancer ルールの背後に存在することはできません。

- リクエストがターゲットに送信されると、アプリケーションは、すべての認証 Cookie に対して有効期限を -1 に設定する必要があります。Application Load Balancer は、最大 16K のサイズの Cookie をサポートするため、クライアントに送信するシャードを最大 4 つ作成できます。
  - IdP にログアウトエンドポイントがある場合、IdP ログアウトエンドポイントへのリダイレクトを発行する必要があります。例えば、Amazon Cognito デベロッパーガイドに記載されている[ログアウトエンドポイント]。
  - IdP にログアウトエンドポイントがない場合、リクエストはクライアントのログアウトランディングページに戻り、ログインプロセスが再起動されます。

- IdP にログアウトエンドポイントがあると仮定すると、IdP はアクセストークンを期限切れにしてトークンを更新し、ユーザーをクライアントのログアウトランディングページにリダイレクトし直す必要があります。
- 後続のリクエストは、元の認証フローに従います。

HTTP ヘッダーと Application Load Balancer

HTTP リクエストと HTTP レスポンスは、ヘッダーフィールドを使用して HTTP メッセージに関する情報を送信します。HTTP ヘッダーは自動的に追加されます。ヘッダーフィールドはコロンで区切られた名前と値のペアであり、キャリッジリターン (CR) とラインフィード (LF) で区切ります。HTTP ヘッダーフィールドの標準セットは、RFC 2616 の Message Headers で定義されています。アプリケーションで広く使用されている標準以外の HTTP ヘッダーもあります。標準以外の HTTP ヘッダーには、`X-Forwarded` というプレフィックスが付いている場合があります。Application Load Balancer では、次の `X-Forwarded` ヘッダーがサポートされます。

HTTP 接続の詳細については、Elastic Load Balancing ユーザーガイドの Request routing を参照してください。

X-Forwarded ヘッダー

- X-Forwarded-For
- X-Forwarded-Proto
- X-Forwarded-Port

X-Forwarded-For

`X-Forwarded-For` リクエストヘッダーは、HTTP または HTTPS ロードバランサーを使用する場合に、クライアントの IP アドレスを識別するのに役立ちます。ロードバランサーはクライアント/サーバー間のトラフィックをインターセプトするので、サーバーアクセスログにはロードバランサーの IP アドレスのみが含まれます。クライアントの IP アドレスを確認するには、`routing.http.xff_header_processing.mode` 属性を使用します。この属性を使用すると、Application Load Balancer がターゲットにリクエストを送信する前に、HTTP リクエストの `X-Forwarded-For` ヘッダーを変更、保持、削除できます。この属性に指定できる値は、`append`、`preserve`、および `remove` です。この属性のデフォルト値は `append` です。

> **Important**
>
> X-Forwarded-For ヘッダーは、セキュリティリスクの可能性があるため、慎重に使用する必要があります。エントリは、ネットワーク内で適切に保護されているシステムによって追加された場合にのみ、信頼可能と見なされます。

Append

デフォルトでは、Application Load Balancer は、クライアントの IP アドレスを X-Forwarded-For リクエストヘッダーに格納し、このヘッダーをサーバーに渡します。X-Forwarded-For リクエストヘッダーがオリジナルリクエストに含まれていない場合、ロードバランサーはリクエスト値としてクライアント IP アドレスを持つリクエストヘッダーを作成します。それ以外の場合、ロードバランサーはクライアント IP アドレスを既存のヘッダーに追加し、 ヘッダーをサーバーに渡します。X-Forwarded-For リクエストヘッダーには、カンマで区切られた複数の IP アドレスを含めることができます。

X-Forwarded-For リクエストヘッダーは以下のような形式です。

```
X-Forwarded-For: client-ip-address
```

以下に、IP アドレスが 203.0.113.7 であるクライアントの X-Forwarded-For リクエストヘッダーの例を示します。

```
X-Forwarded-For: 203.0.113.7
```

以下に、IPv6 アドレスが X-Forwarded-For であるクライアントの 2001:DB8::21f:5bff:febf:ce22:8a2e リクエストヘッダーの例を示します。

```
X-Forwarded-For: 2001:DB8::21f:5bff:febf:ce22:8a2e
```

ロードバランサーでクライアントポート保持属性 (routing.http.xff_client_port.enabled) が有効になっている場合、X-Forwarded-For リクエストヘッダーには、client-ip-address の後にコロンで区切って client-port-number が含まれます。ヘッダーは、次のような形式になります。

```
IPv4 -- X-Forwarded-For: client-ip-address:client-port-number
```

```
IPv6 -- X-Forwarded-For: [client-ip-address]:client-port-number
```

IPv6 の場合、ロードバランサーが client-ip-address を既存のヘッダーに追加する際には、アドレスが角括弧で囲まれることに注意してください。

以下に、IPv4 アドレスが 12.34.56.78 で、ポート番号が 8080 であるクライアントの X-Forwarded-For リクエストヘッダーの例を示します。

```
X-Forwarded-For: 12.34.56.78:8080
```

以下に、IPv6 アドレスが 2001:db8:85a3:8d3:1319:8a2e:370:7348 で、ポート番号が 8080 であるクライアントの X-Forwarded-For リクエストヘッダーの例を示します。

```
X-Forwarded-For: [2001:db8:85a3:8d3:1319:8a2e:370:7348]:8080
```

Preserve

属性で preserve モードを指定した場合、HTTP リクエストの X-Forwarded-For ヘッダーは、ターゲットに送信される前に変更されることはありません。

Remove

属性に remove モードを指定した場合、HTTP リクエストの X-Forwarded-For ヘッダーは、ターゲットに送信される前に削除されます。

> **Note**
>
> クライアントポート保持の属性を有効にし (routing.http.xff_client_port.enabled)、かつ routing.http.xff_header_processing.mode 属性 に preserve または remove を選択した場合、Application Load Balancer はクライアントポート保持属性を上書きします。選択したモードに応じて、X-Forwarded-For ヘッダーを変更せずにおくか削除するかして、ターゲットに送信します。

次の表では、append、preserve、remove モードのいずれかを選択した際にターゲットが受信する X-Forwarded-For ヘッダーの例を示します。この例では、ラストホップの IP アドレスは 127.0.0.1 です。

リクエストの説明	リクエストの例	XFF **append** モード	XFF **preserve** モード	XFF **remove** モード
リクエストは XFF ヘッダーを含まずに送信されます	`GET / index.html HTTP/1.1 Host: example.com`	`X-Forwarded-For: 127.0.0.1`	[なし]	[なし]
リクエストは、XFF ヘッダーとクライアント IP アドレスを含んで送信されます。	`GET / index.html HTTP/1.1 Host: example.com X-Forwarded-For: 127.0.0.4`	`X-Forwarded-For: 127.0.0.4, 127.0.0.1`	`X-Forwarded-For: 127.0.0.4`	[なし]
リクエストは、複数のクライアント IP アドレスを含む XFF ヘッダーを含んで送信されます。	`GET / index.html HTTP/1.1 Host: example.com X-Forwarded-For: 127.0.0.4, 127.0.0.8`	`X-Forwarded-For: 127.0.0.4, 127.0.0.8, 127.0.0.1`	`X-Forwarded-For: 127.0.0.4, 127.0.0.8`	[なし]

コンソールを使用して X-Forwarded-For ヘッダーの変更、保持、削除を行うには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[ロードバランサー] を選択します。
3. ロードバランサーを選択します。
4. [属性] タブで、[編集] を選択します。

5. [トラフィックの設定] セクションの [パケット処理] にある [X-Forwarded-For ヘッダー] で、[付加] (デフォルト)、[保持]、または [削除] を選択します。
6. [変更の保存] をクリックします。

を使用して X-Forwarded-Forヘッダーを変更、保存、または削除するには AWS CLI

`routing.http.xff_header_processing.mode` 属性を指定して modify-load-balancer-attributes コマンドを使用します。

X-Forwarded-Proto

`X-Forwarded-Proto` リクエストヘッダーを使用すると、クライアントがロードバランサーへの接続に使用したプロトコル (HTTP または HTTPS) を識別することができます。サーバーアクセスログには、サーバーとロードバランサーの間で使用されたプロトコルのみが含まれ、クライアントとロードバランサーの間で使用されたプロトコルに関する情報は含まれません。クライアントとロードバランサーの間で使用されたプロトコルを判別するには、`X-Forwarded-Proto` リクエストヘッダーを使用します。Elastic Load Balancing は、クライアントとロードバランサーの間で使用されたプロトコルを `X-Forwarded-Proto` リクエストヘッダーに格納し、このヘッダーをサーバーに渡します。

アプリケーションやウェブサイトは `X-Forwarded-Proto` リクエストヘッダーに格納されているプロトコルを使用して、適切な URL にリダイレクトする応答を生成できます。

`X-Forwarded-Proto` リクエストヘッダーは以下のような形式です。

```
X-Forwarded-Proto: originatingProtocol
```

次の例には、HTTPS リクエストとしてクライアントから発信されたリクエストの `X-Forwarded-Proto` リクエストヘッダーが含まれています。

```
X-Forwarded-Proto: https
```

X-Forwarded-Port

`X-Forwarded-Port` リクエストヘッダーは、ロードバランサーへの接続にクライアントが使用した送信先ポートを識別するために役立ちます。

リスナーとルールのタグ

タグは、さまざまな形でリスナーとルールを分類するのに役立ちます。例えば、目的、所有者、環境などに基づいてリソースを分類できます。

各リスナーとルールに複数のタグを追加できます。タグキーは、リスナーとルールごとに一意である必要があります。既にリスナーとルールに関連付けられているキーを持つタグを追加すると、そのタグの値が更新されます。

不要になったタグは、削除することができます。

制限事項

- リソースあたりのタグの最大数 – 50
- キーの最大長 – 127 文字 (Unicode)
- 値の最大長 – 255 文字 (Unicode)
- タグのキーと値は大文字と小文字が区別されます。使用できる文字は、UTF-8 で表現できる文字、スペース、および数字と、特殊文字 (+、-、=、.、_、:、/、@) です。ただし、先頭または末尾にはスペースを使用しないでください。
- タグ名または値に `aws:` プレフィックスを使用しないでください。このプレフィックスは AWS 使用のために予約されています。このプレフィックスが含まれるタグの名前または値は編集または削除できません。このプレフィックスを持つタグは、リソースあたりのタグ数の制限時には計算されません。

リスナータグを更新します。

コンソールを使用してリスナーのタグを更新するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
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. 更新するリスナーを含むロードバランサーの名前を選択し、詳細ページを開きます。
4. [リスナーとルール] タブで、次のいずれかを行います。

   a. [プロトコル:ポート] 列のテキストを選択して、リスナーの詳細ページを開きます。

      [Tags (タグ)] タブで、[Manage tags (タグ管理)] を選択します。

   b. タグを更新するリスナーを選択します。

[リスナーの管理]、[タグの管理] の順に選択します。

c. [タグ] タブにあるリスナーの詳細ページを開くには、[タグ] 列のテキストを選択します。

[Manage tags (タグの管理)] を選択します。

5. [タグの管理] ページで、次の操作を 1 つ以上実行します。

a. タグを更新するには、[キー] と [値] に新しい値を入力します。

b. タグを追加するには、[新しいタグの追加] を選択し、[キー] と [値] に値を入力します。

c. タグを削除するには、タグの横にある [削除] を選択します。

6. タグの更新を完了したら、[変更内容の保存] を選択します。

を使用してリスナーのタグを更新するには AWS CLI

add-tags コマンドと remove-tags コマンドを使用します。

ルールタグの更新

コンソールを使用してルールのタグを更新するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
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. 更新するルールを含むロードバランサーの名前を選択し、詳細ページを開きます。
4. [リスナーとルール] タブで、更新するルールを含むリスナーの [プロトコル:ポート] 列のテキストを選択して、リスナーの詳細ページを開きます。
5. [リスナーの詳細] ページで、次のいずれかの操作を行います。

a. [Name タグ] 列のテキストを選択し、ルールの詳細ページを開きます。

[ルールの詳細] ページで、[タグの管理] を選択します。

b. 更新するルールの [タグ] 列のテキストを選択します。

タグの概要のポップアップで、[タグの管理] を選択します。

6. [タグの管理] ページで、次の操作を 1 つ以上実行します。

a. タグを更新するには、[キー] と [値] に新しい値を入力します。

b. タグを追加するには、[新しいタグの追加] を選択し、[キー] と [値] に値を入力します。

c. タグを削除するには、タグの横にある [削除] を選択します。

7. タグの更新を完了したら、[変更内容の保存] を選択します。

を使用してルールのタグを更新するには AWS CLI

add-tags コマンドと remove-tags コマンドを使用します。

Application Load Balancer のリスナーの削除

リスナーの削除はいつでも行うことができます。ロードバランサーを削除すると、そのリスナーがすべて削除されます。

コンソールを使用してリスナーを削除するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[ロードバランサー] を選択します。
3. ロードバランサーを選択します。
4. [リスナーとルール] タブで、リスナーのチェックボックスを選択し、[リスナーの管理]、[リスナーの削除] を選択します。
5. 確認を求められたら、「**confirm**」を入力し、[削除] を選択します。

を使用してリスナーを削除するには AWS CLI

delete-listener コマンドを使用します。

Application Load Balancer のターゲットグループ

ターゲットグループは、指定されたプロトコルとポート番号を使用して、登録済みのターゲット (EC2 インスタンスなど) ごとにリクエストをルーティングできます。1 つのターゲットを複数のターゲットグループに登録できます。ターゲットグループ単位でヘルスチェックを設定できます。ヘルスチェックは、ロードバランサーのリスナールールに指定されたターゲットグループに登録されたすべてのターゲットで実行されます。

各ターゲットグループは、1 つ以上の登録されているターゲットにリクエストをルーティングするために使用されます。各リスナーのルールを作成するときに、ターゲットグループと条件を指定します。ルールの条件が満たされると、トラフィックが該当するターゲットグループに転送されます。さまざまなタイプのリクエストに応じて別のターゲットグループを作成できます。たとえば、一般的なリクエスト用に 1 つのターゲットグループを作成し、アプリケーションのマイクロサービスへのリクエスト用に別のターゲットグループを作成できます。各ターゲットグループは 1 つのロードバランサーのみで使用できます。詳細については、「Application Load Balancer のコンポーネント」を参照してください。

ロードバランサーのヘルスチェック設定は、ターゲットグループ単位で定義します。各ターゲットグループはデフォルトのヘルスチェック設定を使用します。ただし、ターゲットグループを作成したときや、後で変更したときに上書きした場合を除きます。リスナーのルールでターゲットグループを指定すると、ロードバランサーは、ロードバランサーで有効なアベイラビリティーゾーンにある、ターゲットグループに登録されたすべてのターゲットの状態を継続的にモニタリングします。ロードバランサーは、正常な登録済みターゲットにリクエストをルーティングします。

目次

- ターゲットグループの作成
- ターゲットグループのヘルスチェック
- ターゲットグループに対するクロスゾーン負荷分散
- ターゲットグループの正常性
- ターゲットグループへのターゲットの登録
- Application Load Balancer のスティッキーセッション
- ターゲットとしての Lambda 関数
- ターゲットグループのタグ
- ターゲットグループの削除

ルーティング設定

デフォルトでは、ロードバランサーはターゲットグループの作成時に指定したプロトコルとポート番号を使用して、リクエストをターゲットにルーティングします。または、ターゲットグループへの登録時にターゲットへのトラフィックのルーティングに使用されるポートを上書きすることもできます。

ターゲットグループでは、次のプロトコルとポートがサポートされています。

- プロトコル: HTTP、HTTPS
- ポート: 1 ~ 65535

ターゲットグループが HTTPS プロトコルを使用して設定されているか、HTTPS ヘルスチェックを使用する場合、ターゲットへの TLS 接続には `ELBSecurityPolicy-2016-08` ポリシーのセキュリティ設定が使用されます。ロードバランサーは、ターゲットにインストールする証明書を使用して、ターゲットとの TLS 接続を確立します。ロードバランサーはこれらの証明書を検証しません。したがって、自己署名証明書または期限切れの証明書を使用できます。ロードバランサーとそのターゲットは Virtual Private Cloud (VPC) にあるため、ロードバランサーとターゲット間のトラフィックはパケットレベルで認証されるため、ターゲットの証明書が有効でなくても man-in-the-middle 攻撃やスプーフィングのリスクはありません。から出るトラフィック AWS にはこれらの同じ保護はありません。トラフィックをさらに保護するには、追加の手順が必要になる場合があります。

[Target type (ターゲットタイプ)]

ターゲットグループを作成するときは、そのターゲットの種類を指定します。それにより、このターゲットグループ内でターゲットを登録するときに指定するターゲットの種類が決定されます。ターゲットグループを作成した後で、ターゲットの種類を変更することはできません。

可能なターゲットの種類は次のとおりです。

`instance`

インスタンス ID で指定されたターゲット。

`ip`

ターゲットは IP アドレスです。

`lambda`

ターゲットは Lambda 関数です。

ターゲットの種類が `ip` の場合、次のいずれかの CIDR ブロックから IP アドレスを指定できます。

- ターゲットグループの VPC のサブネット
- 10.0.0.0/8 (RFC 1918)
- 100.64.0.0/10 (RFC 6598)
- 172.16.0.0/12 (RFC 1918)
- 192.168.0.0/16 (RFC 1918)

> **Important**
>
> パブリックにルーティング可能な IP アドレスは指定できません。

サポートされているすべての CIDR ブロックによって、次のターゲットをターゲットグループに登録できます。

- ロードバランサー VPC (同じリージョンまたは異なるリージョン) にピアリングされている VPC 内のインスタンス。
- AWS IP アドレスとポート (データベースなど) でアドレス指定できる リソース。

- AWS Direct Connect または Site-to-Site VPN 接続 AWS を介して にリンクされたオンプレミスリソース。

> **Note**
>
> Local Zone 内にデプロイされた Application Load Balancer の場合、トラフィックを受信するには `ip` ターゲットが同じ Local Zone 内にある必要があります。
> 詳細については、AWS「ローカルゾーンとは」を参照してください。

インスタンス ID を使用してターゲットを指定すると、トラフィックはインスタンスのプライマリネットワークインターフェイスで指定されたプライマリプライベート IP アドレスを使用して、インスタンスにルーティングされます。IP アドレスを使用してターゲットを指定する場合は、1 つまたは複数のネットワークインターフェイスからのプライベート IP アドレスを使用して、トラフィックをインスタンスにルーティングできます。これにより、インスタンスの複数のアプリケーションが同じポートを使用できるようになります。各ネットワークインターフェイスは独自のセキュリティグループを持つことができます。

ターゲットグループのターゲットの種類が `lambda` である場合、1 つの Lambda 関数を登録できます。ロードバランサーが Lambda 関数のリクエストを受け取ると、Lambda 関数を呼び出します。詳細については、「ターゲットとしての Lambda 関数」を参照してください。

Application Load Balancer のターゲットとして、Amazon Elastic Container Service (Amazon ECS) を設定できます。詳細については、「 用 Amazon Elastic Container Service ユーザーガイド」の Application Load Balancer の作成」を参照してください。 AWS Fargate

IP アドレスタイプ

新しいターゲットグループを作成するときは、ターゲットグループの IP アドレスタイプを選択できます。これは、ターゲットとの通信、およびそれらのヘルスステータスのチェックに使用される IP バージョンを制御します。

Application Load Balancer は、IPv4 ターゲットグループと IPv6 ターゲットグループの両方をサポートします。デフォルトで選択されるのは IPv4 です。

考慮事項

- ターゲットグループ内のすべての IP アドレスは、同じ IP アドレスタイプである必要があります。例えば、IPv4 ターゲットを IPv6 ターゲットグループに登録することはできません。

- IPv6 ターゲットグループは、dualstack ロードバランサーのみで使用できます。
- IPv6 ターゲットグループでは、IP およびインスタンスタイプのターゲットがサポートされています。

プロトコルバージョン

デフォルトでは、Application Load Balancer は HTTP/1.1 を使用してターゲットにリクエストを送信します。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リクエストを送信できます。

次の表は、リクエストプロトコルとターゲットグループのプロトコルバージョンの組み合わせの結果をまとめたものです。

リクエストプロトコル	プロトコルバージョン	結果
HTTP/1.1	HTTP/1.1	成功
HTTP/2	HTTP/1.1	成功
gRPC	HTTP/1.1	エラー
HTTP/1.1	HTTP/2	エラー
HTTP/2	HTTP/2	成功
gRPC	HTTP/2	ターゲットが grPC をサポートしている場合は成功
HTTP/1.1	gRPC	エラー
HTTP/2	gRPC	POST リクエストの場合は成功
gRPC	gRPC	成功

gRPC プロトコルバージョンの考慮事項

- サポートされているリスナープロトコルは HTTPS だけです。

- リスナールールでサポートされるアクションタイプは、forward のみです 。
- サポートされているターゲットタイプは、instance と ip のみです。
- ロードバランサーは、gRPC リクエストを解析し、パッケージ、サービス、メソッドに基づいて、適切なターゲットグループに gRPC 呼び出しをルーティングします。
- ロードバランサーは、単項ストリーミング、クライアントサイドストリーミング、サーバーサイドストリーミング、および双方向ストリーミングをサポートします。
- カスタムヘルスチェックメソッドには、/package.service/method という形式で指定する必要があります。
- ターゲットからの正常な応答をチェックするときに使用する gRPC ステータスコードを指定する必要があります。
- Lambda 関数をターゲットとして使用することはできません。

HTTP/2 プロトコルバージョンの考慮事項

- サポートされているリスナープロトコルは HTTPS だけです。
- リスナールールでサポートされるアクションタイプは、forward のみです 。
- サポートされているターゲットタイプは、instance と ip のみです。
- ロードバランサーは、クライアントからのストリーミングをサポートします。ロードバランサーは、ターゲットへのストリーミングをサポートしていません。

登録済みターゲット

ロードバランサーは、クライアントにとって単一の通信先として機能し、正常な登録済みターゲットに受信トラフィックを分散します。各ターゲットは、1 つ以上のターゲットグループに登録できます。

アプリケーションの需要が高まった場合、需要に対処するため、1 つまたは複数のターゲット グループに追加のターゲットを登録できます。ロードバランサーは、登録プロセスが完了し、設定されたしきい値に関係なく、ターゲットが最初の最初のヘルスチェックに合格するとすぐに、新しく登録されたターゲットへのトラフィックのルーティングを開始します。

アプリケーションの需要が低下した場合や、ターゲットを保守する必要がある場合、ターゲットグループからターゲットを登録解除することができます。ターゲットを登録解除するとターゲットグループから削除されますが、ターゲットにそれ以外の影響は及びません。登録解除するとすぐに、

ロードバランサーはターゲットへのリクエストのルーティングを停止します。ターゲットは、未処理のリクエストが完了するまで draining 状態になります。リクエストの受信を再開する準備ができると、ターゲットをターゲットグループに再度登録することができます。

インスタンス ID でターゲットを登録する場合は、Auto Scaling グループでロードバランサーを使用できます。Auto Scaling グループにターゲットグループをアタッチすると、ターゲットの起動時に Auto Scaling によりターゲットグループにターゲットが登録されます。詳細については、Amazon EC2 Auto Scaling ユーザーガイドのAuto Scaling グループへのロードバランサーのアタッチを参照してください。

制限

- 同じ VPC に別の Application Load Balancer の IP アドレスを登録することはできません。もう一方の Application Load Balancer が、ロードバランサー VPC にピアリング接続されている VPC に含まれている場合は、その IP アドレスを登録できます。
- ロードバランサー VPC (同じリージョンまたは異なるリージョン) とピア接続されている VPC にインスタンスがある場合、そのインスタンスをインスタンス ID で登録することはできません。このようなインスタンスは IP アドレスで登録できます。

ターゲットグループの属性

ターゲットグループの種類が instance または ip である場合、以下のターゲットグループ属性がサポートされています。

deregistration_delay.timeout_seconds

ターゲットを登録解除する前に Elastic Load Balancing が待機する時間。範囲は 0 ~ 3600 秒です。デフォルト値は 300 秒です。

load_balancing.algorithm.type

ロードバランシングアルゴリズムは、リクエストをルーティングするときにロードバランサーがターゲットを選択する方法を決定します。値は round_robin、least_outstanding_requests、または ですweighted_random。デフォルトは round_robin です。

load_balancing.algorithm.anomaly_mitigation

load_balancing.algorithm.type が の場合にのみ使用できますweighted_random。異常緩和が有効になっているかどうかを示します。値は on または off です。デフォルト: off。

`load_balancing.cross_zone.enabled`

クロスゾーンロードバランサーが有効かどうかを示します。値は `true`、`false` または `use_load_balancer_configuration` です。デフォルト: `use_load_balancer_configuration`。

`slow_start.duration_seconds`

ロードバランサーが新しく登録されたターゲットに、ターゲットグループに対するトラフィックのシェアを直線的に増加させて送信する期間 (秒)。範囲は 30 ~ 900 秒 (15 分) です。デフォルトは 0 秒 (無効) です。

`stickiness.enabled`

スティッキーセッションが有効かどうかを示します。値は `true` または `false` です。デフォルト: `false`。

`stickiness.app_cookie.cookie_name`

アプリケーション Cookie 名 アプリケーション Cookie 名に、ロードバランサーで使用するために予約されている `AWSALB`、`AWSALBAPP`、または `AWSALBTG` のプレフィックスを含めることはできません。

`stickiness.app_cookie.duration_seconds`

アプリケーションベースの Cookie の有効期間 (秒) この期間が過ぎると、Cookie は古いと見なされます。最小値は 1 秒で、最大値は 7 日間 (604800秒) です。デフォルト値は 1 日 (86400 秒) です。

`stickiness.lb_cookie.duration_seconds`

期間ベースの Cookie の有効期間 (秒) この期間が過ぎると、Cookie は古いと見なされます。最小値は 1 秒で、最大値は 7 日間 (604800秒) です。デフォルト値は 1 日 (86400 秒) です。

`stickiness.type`

維持の種類です。指定できる値は `lb_cookie` および `app_cookie` です。

`target_group_health.dns_failover.minimum_healthy_targets.count`

正常である必要があるターゲットの最小数。正常なターゲットの数がこの値を下回っている場合は、DNS でそのゾーンを異常とマークして、トラフィックが正常なゾーンにのみルーティングされるようにします。指定できる値は `off` または 1 から最大ターゲット数までの整数です。`off` の場合、DNS フェイルアウェイが無効になります。つまり、各ターゲットグループが独立してDNSフェイルオーバーに寄与することになります。デフォルト は 1 です。

`target_group_health.dns_failover.minimum_healthy_targets.percentage`

正常でなければならないターゲットの最小割合。正常なターゲットの割合がこの値を下回っている場合は、DNS でそのゾーンを異常とマークして、トラフィックが正常なゾーンにのみルーティングされるようにします。指定できる値は `off`、または 1 から最大ターゲット数までの整数です。`off` の場合、DNS フェイルアウェイが無効になります。つまり、各ターゲットグループが独立してDNSフェイルオーバーに寄与することになります。デフォルト は 1 です。

`target_group_health.unhealthy_state_routing.minimum_healthy_targets.count`

正常でなければならないターゲットの最小数。正常なターゲットの数がこの値を下回っている場合は、異常なターゲットを含むすべてのターゲットにトラフィックを送信します。範囲は 1 からターゲットの最大数です。デフォルト は 1 です。

`target_group_health.unhealthy_state_routing.minimum_healthy_targets.percentage`

正常でなければならないターゲットの最小割合。正常なターゲットの割合がこの値を下回っている場合は、異常なターゲットを含むすべてのターゲットにトラフィックを送信します。指定できる値は、`off` または 1 から 100 までの整数です。デフォルトは `off` です。

ターゲットグループの種類が `lambda` である場合、以下のターゲット属性がサポートされています。

`lambda.multi_value_headers.enabled`

ロードバランサーと Lambda 関数との間で交換されるリクエストとレスポンスのヘッダーに、値のまたは文字列の配列が含まれるかどうかを示します。使用できる値は、`true` または `false` です。デフォルト値は `false` です。詳細については、「複数値ヘッダー」を参照してください。

ルーティングアルゴリズム

ルーティングアルゴリズムは、リクエストを受信するターゲットを決定するときにロードバランサーが使用する方法です。ラウンドロビンルーティングアルゴリズムは、ターゲットグループレベルでリクエストをルーティングするためにデフォルトで使用されます。最小未処理のリクエストと加重ランダムルーティングアルゴリズムも、アプリケーションのニーズに基づいて利用できます。ターゲットグループには、一度に 1 つのアクティブなルーティングアルゴリズムしか使用できませんが、必要に応じてルーティングアルゴリズムを更新できます。

スティッキーセッションを有効にすると、選択したルーティングアルゴリズムが最初のターゲット選択に使用されます。同じクライアントからの今後のリクエストは、選択したルーティングアルゴリズムをバイパスして、同じターゲットに転送されます。

ラウンドロビン

- ラウンドロビンルーティングアルゴリズムは、ターゲットグループ内の正常なターゲット間でリクエストを順番に均等にルーティングします。
- このアルゴリズムは、受信されるリクエストが複雑さが類似している場合、登録されたターゲットの処理能力が類似している場合、またはターゲット間でリクエストを均等に分散する必要がある場合によく使用されます。

最小の未処理のリクエスト

- 最小未処理のリクエストルーティングアルゴリズムは、進行中のリクエストの数が最も少ないターゲットにリクエストをルーティングします。
- このアルゴリズムは、受信するリクエストの複雑さが異なり、登録されたターゲットの処理能力が変わる場合によく使用されます。
- HTTP/2 をサポートするロードバランサーが HTTP/1.1 のみをサポートするターゲットを使用している場合、リクエストは複数の HTTP/1.1 リクエストに変換されます。この設定では、最小未処理のリクエストアルゴリズムは、各 HTTP/2 リクエストを複数のリクエストとして扱います。
- を使用する場合 WebSockets、ターゲットは最小未処理のリクエストアルゴリズムを使用して選択されます。選択すると、ロードバランサーはターゲットへの接続を作成し、この接続を介してすべてのメッセージを送信します。
- 最小未処理のリクエストルーティングアルゴリズムは、スロースタートモードでは使用できません。

重み付きランダム

- 加重ランダムルーティングアルゴリズムは、ターゲットグループ内の正常なターゲット間でリクエストをランダムな順序で均等にルーティングします。
- このアルゴリズムは、自動ターゲット重み (ATW) 異常緩和をサポートします。
- 加重ランダムルーティングアルゴリズムは、スロースタートモードでは使用できません。

ターゲットグループのルーティングアルゴリズムを変更する

ターゲットグループのルーティングアルゴリズムはいつでも変更できます。

新しいコンソールを使用してルーティングアルゴリズムを変更するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインの [Load Balancing (ロードバランシング)] で [Target Groups (ターゲットグループ)] を選択します。
3. ターゲットグループの名前を選択して、その詳細ページを開きます。
4. ターゲットグループの詳細ページの属性タブで、編集 を選択します。
5. 「ターゲットグループ属性の編集」ページの「トラフィック設定」セクションの「負荷分散アルゴリズム」で、「ラウンドロビン」、「最小未処理リクエスト」、または「重み付きランダム」を選択します。
6. [変更の保存] をクリックします。

を使用してルーティングアルゴリズムを変更するには AWS CLI

`load_balancing.algorithm.type` 属性を指定して modify-target-group-attributes コマンドを使用します。

自動ターゲット重み (ATW)

自動ターゲット重み (ATW) は、アプリケーションを実行しているターゲットを常にモニタリングし、異常と呼ばれる重大なパフォーマンス偏差を検出します。ATW は、リアルタイムのデータ異常検出を通じて、ターゲットにルーティングされるトラフィック量を動的に調整する機能を提供します。

自動ターゲット重み (ATW) は、アカウント内のすべての Application Load Balancer で異常検出を自動的に実行します。異常なターゲットが特定されると、ATW は、異常緩和と呼ばれるルーティングされるトラフィックの量を減らすことで、それらの安定を自動的に試みることができます。ATW は、ターゲットグループ障害率を最小限に抑えながら、ターゲットごとの成功率を最大化するようにトラフィック分散を継続的に最適化します。

考慮事項:

- 異常検出は現在、ターゲットからの HTTP 5xx レスポンスコードとターゲットへの接続失敗をモニタリングしています。異常検出は常にオンであり、オフにすることはできません。

- Lambda をターゲットとして使用する場合、ATW はサポートされていません。

異常検出

ATW 異常検出は、ターゲットグループ内の他のターゲットとの動作に重大な偏差を示しているターゲットをモニタリングします。これらの偏差は異常と呼ばれ、1 つのターゲットのエラー率とターゲットグループ内の他のターゲットのエラー率を比較することで決定されます。これらのエラーは、接続エラーと HTTP エラーコードの両方である可能性があります。その後、ピアよりも大幅に上位の を報告するターゲットは、異常と見なされます。

異常検出には、ターゲットグループ内で少なくとも 3 つの正常なターゲットが必要です。ターゲットがターゲットグループに登録されると、まずヘルスチェックに合格してトラフィックの受信を開始する必要があります。ターゲットがターゲットを受信すると、ATW はターゲットのモニタリングを開始し、異常結果を継続的に発行します。異常のないターゲットの場合、異常結果は ですnormal。異常のあるターゲットの場合、異常結果は ですanomalous。

ATW 異常検出は、ターゲットグループのヘルスチェックとは独立して機能します。ターゲットはすべてのターゲットグループのヘルスチェックに合格できますが、エラー率が高いため、異常とマークされます。ターゲットが異常になっても、ターゲットグループのヘルスチェックのステータスには影響しません。

異常検出ステータス

ATW は、ターゲットに対して実行する異常検出のステータスを継続的に発行します。現在のステータスは、 AWS Management Console または を使用していつでも表示できます AWS CLI。

コンソールを使用して異常検出ステータスを表示するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインの [Load Balancing (ロードバランシング)] で [Target Groups (ターゲットグループ)] を選択します。
3. ターゲットグループの名前を選択して、その詳細ページを開きます。
4. ターゲットグループの詳細ページで、ターゲットタブを選択します。
5. 登録済みターゲットテーブル内で、各ターゲットの異常ステータスを異常検出結果列に表示できます。

   異常が検出されなかった場合、結果は ですnormal。

異常が検出された場合、結果は ですanomalous。

を使用して異常検出結果を表示するには AWS CLI

Include.member.N 属性値を に設定して describe-target-health コマンドを使用しますAnomalyDetection。

異常の軽減

> **Important**
>
> ATW の異常軽減関数は、加重ランダムルーティングアルゴリズムを使用する場合にのみ使用できます。

ATW 異常緩和は、異常なターゲットからトラフィックを自動的にルーティングし、復旧する機会を提供します。

緩和中：

- ATW は、異常なターゲットにルーティングされるトラフィックの量を定期的に調整します。現在、期間は 5 秒ごとです。
- ATW は、異常ターゲットにルーティングされるトラフィックの量を、異常緩和の実行に必要な最小量に減らします。
- 異常として検出されなくなったターゲットは、ターゲットグループ内の他の通常のターゲットと同等になるまで、徐々にトラフィックがルーティングされます。

ATW 異常緩和を有効にする

異常緩和はいつでも有効にできます。

コンソールを使用して異常緩和を有効にするには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインの [Load Balancing (ロードバランシング)] で [Target Groups (ターゲットグループ)] を選択します。
3. ターゲットグループの名前を選択して、その詳細ページを開きます。

4. ターゲットグループの詳細ページの属性タブで、編集 を選択します。
5. 「ターゲットグループ属性の編集」ページの「トラフィック設定」セクションの「負荷分散アルゴリズム」で、重み付きランダムが選択されていることを確認します。

   注： 加重ランダムアルゴリズムを最初に選択すると、異常検出はデフォルトでオンになります。
6. 「異常緩和」で、異常緩和を有効にするが選択されていることを確認します。
7. [変更の保存] をクリックします。

を使用して異常緩和を有効にするには AWS CLI

`load_balancing.algorithm.anomaly_mitigation` 属性を指定して modify-target-group-attributes コマンドを使用します。

異常軽減ステータス

ATW がターゲットに対して緩和を実行するたびに、 AWS Management Console または を使用していつでも現在のステータスを表示できます AWS CLI。

コンソールを使用して異常緩和ステータスを表示するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインの [Load Balancing (ロードバランシング)] で [Target Groups (ターゲットグループ)] を選択します。
3. ターゲットグループの名前を選択して、その詳細ページを開きます。
4. ターゲットグループの詳細ページで、ターゲットタブを選択します。
5. 登録済みターゲットテーブル内では、各ターゲットの異常緩和ステータスを「緩和策が有効」列に表示できます。

   緩和が進行中でない場合、ステータスは ですyes。

   緩和が進行中の場合、ステータスは ですno。

を使用して異常緩和ステータスを表示するには AWS CLI

`Include.member.N` 属性値を に設定して describe-target-health コマンドを使用しますAnomalyDetection。

登録解除の遅延

Elastic Load Balancing は、登録解除するターゲットへのリクエストの送信を停止します。デフォルトでは、Elastic Load Balancing 登録解除プロセスを完了する前に 300 秒待って、ターゲットへ処理中のリクエストが完了するのを助けることができます。Elastic Load Balancing が待機する時間を変更するには、登録解除の遅延値を更新します。

登録解除するターゲットの初期状態は `draining` です。登録解除の遅延が経過すると、登録解除プロセスは完了し、ターゲットの状態は `unused` になります。ターゲットが Auto Scaling グループの一部である場合、ターゲットを終了して置き換えることができます。

登録解除するターゲットに未処理のリクエストやアクティブな接続がない場合は、Elastic Load Balancing は登録解除の遅延時間が経過するのを待たずに、即時登録解除プロセスを完了します。ただし、ターゲットの登録解除が完了しても、ターゲットのステータスは、登録解除の遅延タイムアウトの期限が切れるまで `draining` と表示されます。タイムアウトの期限が切れると、ターゲットは `unused` 状態に移行します。

登録解除の遅延が経過する前に登録解除するターゲットが接続を終了すると、クライアントは 500 レベルのエラー応答を受信します。

コンソールを使用して登録解除の遅延値を更新するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインの [Load Balancing (ロードバランシング)] で [Target Groups (ターゲットグループ)] を選択します。
3. ターゲットグループの名前を選択して、その詳細ページを開きます。
4. [グループの詳細] タブの [属性] セクションで、[編集] を選択します。
5. [属性の編集] ページで、必要に応じて [登録解除の遅延] の値を変更します。
6. [変更の保存] をクリックします。

を使用して登録解除の遅延値を更新するには AWS CLI

`deregistration_delay.timeout_seconds` 属性を指定して modify-target-group-attributes コマンドを使用します。

スロースタートモード

デフォルトでは、ターゲットはターゲットグループを使用して登録され初期ヘルスチェックを渡した後、すぐにリクエストの全シェアを受信し始めます。スロースタートモードを使用すると、ロードバランサーがターゲットにリクエストの全シェアを送信し始めるまでの猶予期間が設定されます。

ターゲットグループのスロースタートを有効にした後、ターゲットグループによってそのターゲットが正常と見なされると、ターゲットはスロースタートモードになります。スロースタートモードのターゲットは、設定されたスロースタート期間が経過するか、ターゲットが異常になると、スロースタートモードを終了します。ロードバランサーは、スロースタートモードのターゲットに送信できるリクエスト数を直線的に増加させます。正常なターゲットがスロースタートモードを終了すると、ロードバランサーはリクエストの全シェアを送信できます。

考慮事項

- ターゲットグループのスロースタートを有効にした時点で、ターゲットグループに登録されていた正常なターゲットは、スロースタートモードになりません。
- 空のターゲットグループでスロースタートを有効にし、その後、単一登録オペレーションを使用してターゲットを登録した場合、それらのターゲットはスロースタートモードになりません。新しく登録されたターゲットは、スロースタートモードになっていない正常なターゲットが 1 つ以上ある場合にのみ、スロースタートモードになります。
- スロースタートモードのターゲットを登録解除すると、そのターゲットはスロースタートモードを終了します。同じターゲットを再度登録すると、ターゲットグループによって正常と見なされたときに、スロースタートモードになります。
- スロースタートモードのターゲットが異常になった場合、ターゲットはスロースタートモードを終了します。ターゲットが正常になると、再びスロースタートモードになります。
- 最小未処理のリクエストまたは加重ランダムルーティングアルゴリズムを使用する場合、スロースタートモードを有効にすることはできません。

コンソールを使用してスロースタート期間の値を更新するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインの [Load Balancing (ロードバランシング)] で [Target Groups (ターゲットグループ)] を選択します。
3. ターゲットグループの名前を選択して、その詳細ページを開きます。
4. [グループの詳細] タブの [属性] セクションで、[編集] を選択します。

5. [属性の編集] ページで、必要に応じて [スロースタート期間] の値を変更します。スロースタートモードを無効にするには、期間を 0 に設定します。
6. [変更の保存] をクリックします。

を使用してスロースタート期間の値を更新するには AWS CLI

`slow_start.duration_seconds` 属性を指定して modify-target-group-attributes コマンドを使用します。

ターゲットグループの作成

ターゲットグループにターゲットを登録します。デフォルトでは、ロードバランサーはターゲットグループに指定したポートとプロトコルを使用して登録済みターゲットにリクエストを送信します。ターゲットグループに各ターゲットを登録するときに、このポートを上書きできます。

ターゲットグループを作成すると、タグを追加できます。

ターゲットグループ内のターゲットにトラフィックをルーティングするには、リスナーを作成するか、リスナーのルールを作成するときに、アクションでターゲットグループを指定します。詳細については、「リスナールール」を参照してください 複数のリスナーで同じターゲットグループを指定できますが、これらのリスナーは同じ Application Load Balancer に属している必要があります。ロードバランサーでターゲットグループを使用するには、ターゲットグループが他のロードバランサーのリスナーによって使用されていないことを確認する必要があります。

ターゲットグループのタグはいつでも追加または削除できます。詳細については、「ターゲットグループへのターゲットの登録」を参照してください。ターゲットグループのヘルスチェック設定を変更することもできます。詳細については、「ターゲットグループのヘルスチェック設定を変更する」を参照してください。

コンソールを使用してターゲットグループを作成するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインの [ロードバランシング] で [ターゲットグループ] を選択します。
3. [ターゲットグループの作成] を選択します。
4. [ターゲットタイプの選択] で、ターゲットをインスタンス ID で登録する場合は [インスタンス]、ターゲットを IP アドレスで登録する場合は [IP アドレス]、ターゲットを Lambda 関数として登録する場合は [Lambda 関数] を選択します。

5. [Target group name] で、ターゲットグループの名前を入力します。この名前はリージョンごと、アカウントごとに一意である必要があり、最大 32 文字の英数字またはハイフンのみを使用する必要があり、先頭と末尾にハイフンを使用することはできません。
6. (オプション) [Protocol] と [Port] で、必要に応じてデフォルト値を変更します。
7. ターゲットタイプが [インスタンス] または [IP アドレス] の場合は、[IPv4] または [IPv6] を [IP アドレスタイプ] として選択します。そうでない場合は、次のステップに進みます。

   このターゲットグループに含めることができるのは、選択した IP アドレスタイプを持つターゲットのみであることに注意してください。ターゲットグループの作成後に IP アドレスタイプを変更することはできません。
8. [VPC] で、Virtual Private Cloud (VPC) を選択します。[IP addresses] (IP アドレス) ターゲットタイプについては、選択可能な VPC が、前のステップで選択した [IP address type] (IP アドレスタイプ) をサポートする VPC であることに注意してください。
9. (オプション) [Protocol version] で、必要に応じてデフォルト値を変更します。
10. (オプション) [ヘルスチェック] セクションで、必要に応じてデフォルト設定を変更します。
11. ターゲットタイプが [Lambda 関数] の場合は、[ヘルスチェック] セクションの [有効化] を選択してヘルスチェックを有効にできます。
12. (オプション) 次のように 1 つ以上のタグを追加します。

    a. [Tags (タグ)] セクションを展開します。
    b. [Add tag] を選択します。
    c. タグキーとタグ値を入力します。
13. [次へ] をクリックします。
14. (オプション) 次のように 1 つ以上のターゲットを追加します。
    - ターゲットタイプがインスタンスである場合は、1 つ以上のインスタンスを選択し、1 つ以上のポートを入力して、[保留中として以下を含める] を選択します。

      注: IPv6 ターゲットグループに登録する場合、インスタンスにプライマリ IPv6 アドレスが割り当てられている必要があります。
    - ターゲットタイプが [IP addresses] (IP アドレス) の場合は、以下を実行してください。

      a. リストからネットワーク [VPC] を選択、または [Other private IP addresses] (その他のプライベート IP アドレス) を選択します。
      b. IP アドレスを手動で入力する、またはインスタンスの詳細を使用して IP アドレスを検索します。IP アドレスは、一度に 5 個まで入力できます。

c. 指定された IP アドレスにトラフィックをルーティングするためのポートを入力します。

d. [Include as pending below] (保留中として以下を含める) をクリックします。

- ターゲットタイプが [Lambda function] (Lambda 関数) の場合は、単一の Lambda 関数を指定する、またはこのステップを省略して、後ほど Lambda 関数を指定します。

15. [ターゲットグループの作成] を選択します。
16. (オプション) リスナールールでターゲットグループを指定できます。詳細については、「リスナールール」を参照してください。

を使用してターゲットグループを作成するには AWS CLI

ターゲットグループを作成するには create-target-group コマンド、ターゲットグループにタグを付けるには add-tags コマンド、ターゲットを追加するには register-targets コマンドを使用します。

ターゲットグループのヘルスチェック

Application Load Balancer は、登録されたターゲットのステータスをテストするため、定期的にリクエストを送信します。これらのテストは、ヘルスチェックと呼ばれます。

各ロードバランサーノードは、ロードバランサーに対して有効になっているアベイラビリティーゾーンの正常なターゲットにのみ、リクエストをルーティングします。各ロードバランサーノードは、ターゲットが登録されているターゲットグループのヘルスチェック設定を使用して、各ターゲットの状態を確認します。ターゲットは、登録後に正常と見なされるためには、1 つのヘルスチェックに合格する必要があります。各ヘルスチェックが完了すると、ロードバランサーノードはヘルスチェック用に確立された接続を終了します。

ターゲットグループに異常な登録済みターゲットのみが含まれている場合、そのヘルスステータスにかかわらず、ロードバランサーはそれらすべてのターゲットにリクエストをルーティングします。つまり、有効なすべてのアベイラビリティーゾーン内で、すべてのターゲットが同時にヘルスチェックに失敗すると、ロードバランサーはオープンに失敗します。フェールオープンの効果は、ヘルスステータスにかかわらず、ロードバランシングのアルゴリズムに基づいて、有効なすべてのアベイラビリティーゾーン内のすべてのターゲットへのトラフィックを許可することです。

ヘルスチェックは をサポートしていません WebSockets。

ヘルスチェックの設定

次の表に示すように、ターゲットグループのターゲットのヘルスチェックを設定します。表で使用される設定名は、API で使用される名前です。ロードバランサーは、指定されたポート、プロトコル、ヘルスチェックパスを使用して、登録された各ターゲットにヘルスチェックリクエストをHealthCheckIntervalSeconds秒ごとに送信します。各ヘルスチェックリクエストは独立しており、結果は間隔全体で存続します。ターゲットが応答するまでにかかる時間は、次のヘルスチェックリクエストまでの間隔に影響を与えません。ヘルスチェックがUnhealthyThresholdCount連続して失敗した場合、ロードバランサーはターゲットをサービス停止にします。ヘルスチェックがHealthyThresholdCount連続した成功を超えると、ロードバランサーはターゲットを稼働状態に戻します。

設定	説明
HealthCheckProtocol	ターゲットでヘルスチェックを実行するときにロードバランサーが使用するプロトコル。使用可能なプロトコルは HTTP および HTTPS です。デフォルトは HTTP プロトコルです。 これらのプロトコルは、HTTP GET メソッドを使用してヘルスチェックリクエストを送信します。
HealthCheckPort	ターゲットでヘルスチェックを実行するときにロードバランサーが使用するポート。デフォルトでは、各ターゲットがロードバランサーからトラフィックを受信するポートが使用されます。
HealthCheckPath	ターゲットでのヘルスチェックの送信先。 プロトコルバージョンが HTTP/1.1 または HTTP/2 の場合は、有効な URI (/ パス ?クエリ) を指定します。デフォルトは / です。 プロトコルバージョンが grPC の場合は、カスタムヘルスチェックメソッドのパスを `/package.service/method` 形式で指定し

設定	説明
	ます。デフォルト: /AWS.ALB/healthcheck 。
HealthCheckTimeoutSeconds	ヘルスチェックを失敗と見なす、ターゲットからレスポンスがない時間 (秒単位)。範囲は 2 ~ 120 秒です。ターゲットタイプが instance または ip の場合のデフォルトは 5 秒で、ターゲットタイプが lambda の場合のデフォルトは 30 秒です。
HealthCheckIntervalSeconds	個々のターゲットのヘルスチェックの概算間隔 (秒単位)。範囲は 5 ~ 300 秒です。ターゲットタイプが instance または ip の場合のデフォルトは 30 秒で、ターゲットタイプが lambda の場合のデフォルトは 35 秒です。
HealthyThresholdCount	非正常なインスタンスが正常であると見なすまでに必要なヘルスチェックの連続成功回数。範囲は 2 ~ 10 です。デフォルトは 5 です。
UnhealthyThresholdCount	非正常なインスタンスが非正常であると見なすまでに必要なヘルスチェックの連続失敗回数。範囲は 2 ~ 10 です。デフォルトは 2 です。

設定	説明
マッチャー	ターゲットからの正常なレスポンスを確認するために使用するコード。これらは、コンソールでは [成功コード] と呼ばれます。 プロトコルバージョンが HTTP/1.1 または HTTP/2 の場合、指定できる値は 200～499 です。複数の値 (例: "200,202") または値の範囲 (例: "200-299") を指定できます。デフォルト値は 200 です。 プロトコルバージョンが grPC の場合、指定できる値は 0～99 です。複数の値 (例: "0,1") または値の範囲 (例: "0-5") を指定できます。デフォルト値は 12 です。

ターゲットヘルスステータス

ロードバランサーがターゲットにヘルスチェックリクエストを送信する前に、ターゲットグループに登録し、リスナールールでターゲットグループを指定して、ターゲットのアベイラビリティーゾーンがロードバランサーに対して有効になっていることを確認する必要があります。ターゲットがロードバランサーからリクエストを受信する前に、最初のヘルスチェックに合格する必要があります。ターゲットが最初のヘルスチェックに合格すると、ステータスは `Healthy` になります。

次の表は、登録されたターゲットのヘルスステータスの可能値を示しています。

値	説明
`initial`	ロードバランサーは、ターゲットを登録中か、ターゲットで最初のヘルスチェックを実行中です。 関連する理由コード:`Elb.RegistrationIn Progress` |`Elb.InitialHealthChecking`
`healthy`	ターゲットは正常です。 関連する理由コード:なし

値	説明
unhealthy	ターゲットはヘルスチェックに応答しなかったか、ヘルスチェックに合格しませんでした。 関連する理由コード :Target.ResponseCodeMismatch |Target.Timeout |Target.FailedHealthChecks |Elb.InternalError
unused	ターゲットがターゲットグループに登録されていないか、ターゲットグループがロードバランサーのリスナールールで使用されていないか、ロードバランサーに対して有効ではないアベイラビリティーゾーンにターゲットがあるか、ターゲットが停止または終了状態にあります。 関連する理由コード :Target.NotRegistered |Target.NotInUse |Target.InvalidState |Target.IpUnusable
draining	ターゲットは登録解除中で、Connection Draining 中です。 関連する理由コード : Target.DeregistrationInProgress
unavailable	ターゲットグループのヘルスチェックは無効になっています。 関連する理由コード : Target.HealthCheckDisabled

ヘルスチェックの理由コード

ターゲットのステータスが Healthy 以外の値の場合、API は問題の理由コードと説明を返し、コンソールで同じ説明が表示されます。Elb で始まる理由コードはロードバランサー側で発生し、Target で始まる理由コードはターゲット側で発生します。ヘルスチェックの失敗が考えられる原因の詳細については、「トラブルシューティング」を参照してください。

理由コード	説明
Elb.InitialHealthChecking	最初のヘルスチェックが進行中です
Elb.InternalError	内部エラーのため、ヘルスチェックに失敗しました
Elb.RegistrationInProgress	ターゲットの登録中です
Target.DeregistrationInProgress	ターゲットの登録解除中です
Target.FailedHealthChecks	ヘルスチェックに失敗しました
Target.HealthCheckDisabled	ヘルスチェックは無効になっています。
Target.InvalidState	ターゲットが停止状態にあります ターゲットは終了状態にあります ターゲットは終了状態か、または停止状態にあります ターゲットは無効な状態にあります
Target.IpUnusable	IP アドレスはロードバランサーによって使用されているので、ターゲットとして使用できません
Target.NotInUse	ターゲットグループは、ロードバランサーからトラフィックを受信するように設定されていません ロードバランサーが有効になっていないアベイラビリティーゾーンにターゲットがあります
Target.NotRegistered	ターゲットはターゲットグループに登録されていません
Target.ResponseCodeMismatch	次のコードでヘルスチェックに失敗しました: [code]
Target.Timeout	リクエストがタイムアウトしました

ターゲットのヘルスステータスをチェックする

ターゲットグループに登録されたターゲットのヘルスステータスをチェックできます。

コンソールを使用してターゲットのヘルスステータスをチェックするには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインの [Load Balancing (ロードバランシング)] で [Target Groups (ターゲットグループ)] を選択します。
3. ターゲットグループの名前を選択して、その詳細ページを開きます。
4. [ターゲット] タブの [ステータス] 列は、各ターゲットのステータスを示します。
5. ステータスの値が `Healthy` 以外の場合は、[ステータスの詳細] 列に詳細情報が表示されます。ヘルスチェックの失敗に関するヘルプについては、「トラブルシューティング」を参照してください。

を使用してターゲットの状態を確認するには AWS CLI

describe-target-healthコマンドを実行します。このコマンドの出力にはターゲットのヘルス状態が含まれます。ステータスの値が `Healthy` 以外の場合は、理由コードも出力に含まれています。

異常なターゲットに関する E メール通知を受信するには

CloudWatch アラームを使用して Lambda 関数をトリガーし、異常なターゲットに関する詳細を送信します。 step-by-step 手順については、次のブログ記事「ロードバランサーの異常なターゲットの特定」を参照してください。

ターゲットグループのヘルスチェック設定を変更する

ターゲットグループのヘルスチェック設定はいつでも変更できます。

コンソールを使用してターゲットグループのヘルスチェック設定を変更するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインの [Load Balancing (ロードバランシング)] で [Target Groups (ターゲットグループ)] を選択します。
3. ターゲットグループの名前を選択して、その詳細ページを開きます。
4. [グループの詳細] タブの [ヘルスチェックの設定] セクションで、[編集] を選択します。

5. [ヘルスチェックの編集の設定] ページで、必要に応じて設定を変更し、[変更内容の保存] を選択します。

を使用してターゲットグループのヘルスチェック設定を変更するには AWS CLI

modify-target-groupコマンドを実行します。

ターゲットグループに対する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オンの場合、各ロードバランサーノードは、登録されたすべてのアベイラビリティーゾーンにある登録済みターゲットに対し、トラフィックを分散します。クロスゾーン負荷分散がオフの場合、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内で登録済みの各ターゲットにのみ、トラフィックを分散します。これは、正常なゾーンが異常なゾーンからの影響を受けないようにする、または全体的なレイテンシーを改善する目的で、リージョン範囲の障害があるドメインよりもゾーン範囲の障害があるドメインの方を優先したい場合などに使用します。

Application Load Balancer では、クロスゾーン負荷分散はロードバランサーレベルで常にオンになっており、オフにすることはできません。ターゲットグループに対しても、ロードバランサーの設定がデフォルトで使用されますが、クロスゾーン負荷分散をターゲットグループレベルで明示的にオフにすることで、デフォルト設定をオーバーライドできます。

考慮事項

- クロスゾーン負荷分散がオフの場合、ターゲットに対するスティッキーなセッションはサポートされません。
- クロスゾーン負荷分散がオフの場合、ターゲットとしての Lambda 関数はサポートされません。
- いずれかのターゲットでパラメータ `AvailabilityZone` が `all` に設定されている場合に、`ModifyTargetGroupAttributes` API を介してクロスゾーン負荷分散をオフにしようとすると、エラーが発生します。
- ターゲットの登録時は、`AvailabilityZone` パラメータは必須です。特定のアベイラビリティゾーン値は、クロスゾーン負荷分散がオフの場合にのみ使用できます。これ以外の場合、このパラメータは無視され `all` が適用されます。

ベストプラクティス

- ターゲットグループごとに、使用する予定のすべてのアベイラビリティーゾーンで十分なターゲット容量を予約します。参加しているすべてのアベイラビリティーゾーンで十分な容量を確保できない場合は、クロスゾーン負荷分散をオンにしておくことをお勧めします。
- Application Load Balancer で複数のターゲットグループを設定する場合には、すべてのターゲットグループが、設定されたリージョン内の同じアベイラビリティーゾーンに参加していることを確認してください。これにより、クロスゾーン負荷分散がオフの状態でアベイラビリティーゾーンが空になり、そのアベイラビリティーゾーンが受け取るすべての HTTP リクエストに対して、503 エラーが返されるようになるのを防ぎます
- 空のサブネットを作成することは避けます。空のサブネットに対しても、Application Load Balancer は、そのゾーン IP アドレスを DNS 経由で公開します。これにより、HTTP リクエストには 503 エラーが返されます。
- クロスゾーン負荷分散がオフになっているターゲットグループで、アベイラビリティーゾーンごとに十分なターゲット容量が予約されているのにも関わらず、アベイラビリティーゾーンのすべてのターゲットに障害が発生することがあります。ターゲットのすべてが障害を起こしているターゲットグループが少なくとも 1 つある場合、対応するロードバランサーノードの IP アドレスは DNS から削除されます。ターゲットグループ内で、ターゲットが 1 つでも正常状態に復帰すると、対象の IP アドレスは DNS に復元されます。

クロスゾーン負荷分散をオフにする

Application Load Balancer のターゲットグループでは、クロスゾーン負荷分散を任意のタイミングでオフにできます。

コンソールを使用して、クロスゾーン負荷分散をオフにするには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインの [Load Balancing] (ロードバランサー) で [Target Groups] (ターゲットグループ) を選択します。
3. ターゲットグループの名前を選択して、その詳細ページを開きます。
4. [Attributes] (属性) タブで、[Edit] (編集) を選択します。
5. [Edit target group attributes] (ターゲットグループ属性の編集) ページの、[Cross-zone load balancing] (クロスゾーン負荷分散) で、[Off] (オフ) を選択します。
6. [Save changes] (変更の保存) をクリックします。

AWS CLI を使用してクロスゾーン負荷分散をオフにするには

`load_balancing.cross_zone.enabled` 属性を `false` に設定しながら、modify-target-group-attributes コマンドを実行します。

```
aws elbv2 modify-target-group-attributes --target-group-arn my-targetgroup-arn --
attributes Key=load_balancing.cross_zone.enabled,Value=false
```

以下に、応答の例を示します。

```
{
    "Attributes": [
        {
            "Key": "load_balancing.cross_zone.enabled",
            "Value": "false"
        },
    ]
}
```

クロスゾーン負荷分散をオンにする

Application Load Balancer のターゲットグループでは、クロスゾーン負荷分散を任意のタイミングでオンにできます。ターゲットグループレベルに対する、クロスゾーン負荷分散の設定は、ロードバランサーレベルの設定よりも優先されます。

コンソールを使用してクロスゾーン負荷分散をオンにするには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインの [Load Balancing] (ロードバランサー) で [Target Groups] (ターゲットグループ) を選択します。
3. ターゲットグループの名前を選択して、その詳細ページを開きます。
4. [Attributes] (属性) タブで、[Edit] (編集) を選択します。
5. [Edit target group attributes] (ターゲットグループ属性の編集) ページの、[Cross-zone load balancing] (クロスゾーン負荷分散) で、[On] (オン) を選択します。
6. [Save changes] (変更の保存) をクリックします。

AWS CLI を使用してクロスゾーン負荷分散をオンにするには

load_balancing.cross_zone.enabled 属性を true に設定しながら、modify-target-group-attributes コマンドを実行します。

```
aws elbv2 modify-target-group-attributes --target-group-arn my-targetgroup-arn --
attributes Key=load_balancing.cross_zone.enabled,Value=true
```

以下に、応答の例を示します。

```
{
    "Attributes": [
        {
            "Key": "load_balancing.cross_zone.enabled",
            "Value": "true"
        },
    ]
}
```

ターゲットグループの正常性

デフォルトでは、ターゲットグループが少なくとも 1 つの正常なターゲットを持っている限り、そのターゲットグループは正常であると見なされます。フリートが大きい場合、トラフィックを処理する正常なターゲットが 1 つだけでは十分ではありません。代わりに、正常でなければならないターゲットの最小数または割合、および正常なターゲットが指定されたしきい値を下回ったときにロードバランサーが実行するアクションを指定できます。これにより、可用性が向上します。

異常な状態アクション

以下のアクションに対して正常なしきい値を設定できます。

- DNS フェイルオーバー - ゾーン内の正常なターゲットがしきい値を下回ると、そのゾーンのロードバランサーノードの IP アドレスが DNS で異常とマークされます。そのため、クライアントがロードバランサーの DNS 名を解決すると、トラフィックは正常なゾーンにのみルーティングされます。
- ルーティングフェイルオーバー - ゾーン内の正常なターゲットがしきい値を下回ると、ロードバランサーは、異常なターゲットを含め、ロードバランサーノードで使用可能なすべてのターゲットにトラフィックを送信します。これにより、特にターゲットが一時的にヘルスチェックに合格しなかった場合に、クライアント接続が成功する可能性が高まり、正常なターゲットが過負荷になるリスクが軽減されます。

要件と考慮事項

- ターゲットが Lambda 関数であるターゲットグループでは、この機能は使用できません。Application Load Balancer が Network Load Balancer または Global Accelerator のターゲットである場合は、DNS フェイルオーバーのしきい値を設定しないでください。
- アクションに両方のタイプのしきい値 (数と割合) を指定すると、どちらかのしきい値に達したときにロードバランサーがアクションを実行します。
- 両方のアクションにしきい値を指定する場合、DNS フェイルオーバーのしきい値はルーティングフェイルオーバーのしきい値以上である必要があります。これにより、DNS フェイルオーバーはルーティングフェイルオーバーの有無にかかわらず発生します。
- しきい値を割合として指定すると、ターゲットグループに登録されているターゲットの総数に基づいて、値が動的に計算されます。
- ターゲットの合計数は、クロスゾーンロードバランサーがオフになっているかオンになっているかによって決まります。クロスゾーンロードバランサーがオフの場合、各ノードは独自のゾーン内のターゲットにのみトラフィックを送信します。つまり、しきい値は有効になっている各ゾーンのターゲット数に個別に適用されます。クロスゾーン負荷分散がオンの場合、各ノードはすべての有効ゾーンのすべてのターゲットにトラフィックを送信します。つまり、指定されたしきい値はすべての有効ゾーンのターゲットの総数に適用されます。
- DNS フェイルオーバーでは、ロードバランサーの DNS ホスト名から異常のあるゾーンの IP アドレスを削除します。ただし、DNS レコードの time-to-live (TTL) が期限切れになるまで (60 秒) 、ローカルクライアント DNS キャッシュにこれらの IP アドレスが含まれる場合があります。
- DNS フェイルオーバーが発生すると、ロードバランサーに関連するすべてのターゲットグループに影響します。特にクロスゾーンロードバランサーがオフになっている場合は、この追加のトラフィックを処理するのに十分な容量が残りのゾーンにあることを確認してください。
- DNS フェイルオーバーでは、すべてのロードバランサーゾーンが異常と見なされると、ロードバランサーは異常なゾーンを含むすべてのゾーンにトラフィックを送信します。
- DNS フェイルオーバーにつながる可能性のある正常なターゲットが十分にあるかどうか以外にも、ゾーンのヘルスなどの要因があります。

モニタリング

ターゲットグループのヘルスをモニタリングするには、CloudWatch 「ターゲットグループのヘルスのメトリクス」を参照してください。

例

次の例は、ターゲットグループのヘルス設定がどのように適用されるかを示しています。

シナリオ

- 2 つのアベイラビリティーゾーン A と B をサポートするロードバランサー
- 各アベイラビリティーゾーンには 10 の登録済みターゲットが含まれています
- ターゲットグループには、次のターゲットグループのヘルス設定があります。
  - DNS フェイルオーバー - 50%
  - ルーティングフェイルオーバー - 50%
- アベイラビリティーゾーン B で 6 つのターゲットが失敗

クロスゾーンロードバランサーがオフの場合

- 各アベイラビリティーゾーンのロードバランサーノードは、アベイラビリティーゾーンの 10 個のターゲットにのみトラフィックを送信できます。
- アベイラビリティーゾーン A には 10 個の正常なターゲットがあり、これは正常なターゲットの必要な割合を満たしています。ロードバランサーは引き続き、10 の正常なターゲット間でトラフィックを分散します。
- アベイラビリティーゾーン B には正常なターゲットが 4 つしかなく、これはアベイラビリティーゾーン B のロードバランサーノードのターゲットの 40% です。これは正常なターゲットの必要なパーセンテージを下回っているため、ロードバランサーは次のアクションを実行します。
  - DNS フェイルオーバー - アベイラビリティーゾーン B が DNS で異常とマークされています。クライアントはロードバランサー名をアベイラビリティーゾーン B のロードバランサーノードに解決できず、アベイラビリティーゾーン A は正常であるため、クライアントはアベイラビリティーゾーン A に新しい接続を送信します。
  - ルーティングフェイルオーバー - 新しい接続がアベイラビリティーゾーン B に明示的に送信されると、ロードバランサーは、異常なターゲットを含むアベイラビリティーゾーン B のすべてのターゲットにトラフィックを分散します。これにより、残りの正常なターゲット間でのシステム停止を防ぐことができます。

クロスゾーンロードバランサーがオンの場合

- 各ロードバランサーノードは、両方のアベイラビリティーゾーンの 20 の登録済みターゲットすべてにトラフィックを送信できます。

- アベイラビリティーゾーン A には 10 個の正常なターゲット、アベイラビリティーゾーン B には 4 個の正常なターゲット、合計 14 個の正常なターゲットがあります。これは両方のアベイラビリティーゾーンのロードバランサーノードのターゲットの 70% であり、正常なターゲットの必要な割合を満たしています。
- ロードバランサーは、両方のアベイラビリティーゾーンの 14 個の正常なターゲット間でトラフィックを分散します。

ターゲットグループのヘルス設定の変更

ターゲットグループに関連するターゲットグループのヘルス設定は、次のように変更できます。

コンソールを使用してターゲットグループのヘルス設定を変更するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインの [ロードバランシング] で [ターゲットグループ] を選択します。
3. ターゲットグループの名前を選択して、その詳細ページを開きます。
4. [Attributes] タブで、[Edit] を選択します。
5. クロスゾーンロードバランサーがオンになっているかオフになっているかを確認します。必要に応じてこの設定を更新して、ゾーンに障害が発生した場合に追加のトラフィックを処理するのに十分な容量を確保してください。
6. [Target group health requirements] (ターゲットグループのヘルス要件) を拡張します。
7. [Configuration type] (設定タイプ) には、両方のアクションに同じしきい値を設定する [Unified configuration] (統合設定) を選択することをお勧めします。
8. [Healthy state requirements] (正常な状態の要件) については、次のいずれかを実行します。
   - [Minimum healthy target count] (正常なターゲットの最小数) を選択し、1 からターゲットグループの最大ターゲット数までの数値を入力します。
   - [Minimum healthy target percentage] (最小の正常なターゲット割合) を選択し、1 から 100 までの数値を入力します。
9. [変更の保存] をクリックします。

を使用してターゲットグループのヘルス設定を変更するには AWS CLI

modify-target-group-attributes コマンドを使用します。次の例では、両方の異常な状態アクションの正常しきい値を 50% に設定しています。

```
aws elbv2 modify-target-group-attributes \
--target-group-arn arn:aws:elasticloadbalancing:region:123456789012:targetgroup/my-targets/73e2d6bc24d8a067 \
--attributes
 Key=target_group_health.dns_failover.minimum_healthy_targets.percentage,Value=50 \

 Key=target_group_health.unhealthy_state_routing.minimum_healthy_targets.percentage,Value=50
```

ロードバランサーで Route 53 DNS フェイルオーバーを使用する

Route 53 を使用して DNS クエリをロードバランサーにルーティングする場合は、同時に Route 53 によりロードバランサーの DNS フェイルオーバーを設定することもできます。フェイルオーバー設定では、ロードバランサー用のターゲットグループのターゲットに関する正常性チェックが Route 53 によって行われ、利用可能かどうかが判断されます。ロードバランサーに正常なターゲットが登録されていない場合、またはロードバランサー自体で不具合が発生している場合、Route 53 は、トラフィックを別の利用可能なリソース (正常なロードバランサーや、Amazon S3 にある静的ウェブサイトなど) にルーティングします。

例えば、www.example.com 用のウェブアプリケーションがあり、異なるリージョンにある 2 つのロードバランサーの背後で冗長なインスタンスを実行するとします。1 つのリージョンのロードバランサーは、主にトラフィックのルーティング先として使用し、もう 1 つのリージョンのロードバランサーは、エラー発生時のバックアップとして使用します DNS フェイルオーバーを設定する場合は、プライマリおよびセカンダリ (バックアップ) ロードバランサーを指定できます。Route 53 は、プライマリロードバランサーが利用可能な場合はプライマリロードバランサーにトラフィックをルーティングし、利用できない場合はセカンダリロードバランサーにルーティングします。

ターゲットの正常性の評価を使用する

- Application Load Balancer のエイリアスレコードでターゲットの正常性の評価が Yes に設定されている場合、Route 53 は alias target 値で指定されたリソースの正常性を評価します。Application Load Balancer の場合、Route 53 は、ロードバランサーに関連付けられたターゲットグループのヘルスチェックを使用します。
- Application Load Balancer 内のすべてのターゲットグループが正常である場合、Route 53 はエイリアスレコードを正常としてマークします。ターゲットグループに少なくとも 1 つの正常なターゲットが含まれている場合、ターゲットグループのヘルスチェックは合格となります。その後、Route 53 は、ルーティングポリシーに従ってレコードを返します。フェイルオーバールーティングポリシーを使用すると、Route 53 はプライマリレコードを返します。

- Application Load Balancer 内のターゲットグループのいずれかが異常な場合、エイリアスレコードは Route 53 ヘルスチェックに失敗します (フェイルオープン)。ターゲットの正常性の評価を使用している場合は、フェイルオーバールーティングポリシーが失敗します。
- Application Load Balancer 内のすべてのターゲットグループが空である (ターゲットがない) 場合、Route 53 はレコードが異常であるとみなします (フェイルオープン)。ターゲットの正常性の評価を使用している場合は、フェイルオーバールーティングポリシーが失敗します。

詳細については、Amazon Route 53 開発者ガイドの「DNS フェイルオーバーの設定」を参照してください。

ターゲットグループへのターゲットの登録

ターゲットグループにターゲットを登録します。ターゲットグループを作成するときは、そのターゲットの種類を指定します。ターゲットの種類は、ターゲットの登録方法を決定します。たとえば、インスタンス ID、IP アドレス、または Lambda 関数を登録できます。詳細については、「Application Load Balancer のターゲットグループ」を参照してください

現在登録されているターゲットの需要が上昇した場合、需要に対応するために追加ターゲットを登録できます。ターゲットがリクエストを処理する準備ができたら、ターゲットグループに登録します。登録処理が完了し、ターゲットが最初のヘルスチェックに合格するとすぐに、ロードバランサーはターゲットへのリクエストのルーティングを開始します。

登録済みターゲットの需要が低下した場合や、ターゲットを保守する必要がある場合、ターゲットグループから登録解除できます。登録解除するとすぐに、ロードバランサーはターゲットへのリクエストのルーティングを停止します。ターゲットがリクエストを受信する準備ができたら、ターゲットグループに再度登録することができます。

ターゲットを登録解除すると、ロードバランサーは未処理のリクエストが完了するまで待機します。これは、Connection Drainingと呼ばれます。Connection Drainingの進行中、ターゲットのステータスは `draining` です。

IP アドレスで登録されたターゲットを登録解除する場合、同じ IP アドレスを再び登録するには、登録解除の遅延が完了するまで待機する必要があります。

インスタンス ID でターゲットを登録する場合は、Auto Scaling グループでロードバランサーを使用できます。Auto Scaling グループにターゲットグループをアタッチし、そのグループがスケールアウトすると、Auto Scaling グループによって起動されたインスタンスが自動的にターゲットグループに

登録されます。Auto Scaling グループからターゲットグループをデタッチした場合、インスタンスはターゲットグループ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」の「Auto Scaling グループへのロードバランサーのアタッチ」を参照してください。

ターゲットセキュリティグループ

EC2 インスタンスをターゲットとして登録した場合、インスタンスのセキュリティグループにより、ロードバランサーがリスナーポートとヘルスチェックポートの両方でインスタンスとの通信が許可されるようにする必要があります。

推奨ルール

Inbound

Source	Port Range	Comment
##################	##########	インスタンスリスナーポートでロードバランサーからのトラフィックを許可する
##################	#######	ヘルスチェックポートでロードバランサーからのトラフィックを許可する

また、パス MTU 検出をサポートするため、インバウンド ICMP トラフィックを許可することをお勧めします。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。 Amazon EC2

共有サブネット

参加者は共有 VPC に Application Load Balancer を作成できます。参加者は、自分と共有されていないサブネットで実行するターゲットを登録することはできません。

ターゲットの登録または登録解除

ターゲットグループのターゲットの種類により、ターゲットグループにターゲットを登録する方法が決定されます。詳細については、「[Target type (ターゲットタイプ)]」を参照してください

目次

インスタンス ID によるターゲットの登録または登録解除

> **Note**
>
> IPv6 ターゲットグループにインスタンス ID でターゲットを登録する場合、ターゲットにはプライマリ IPv6 アドレスが割り当てられている必要があります。詳細については、「Amazon EC2 ユーザーガイド」のIPv6 アドレス」を参照してください。 Amazon EC2

インスタンスは、ターゲットグループに指定された Virtual Private Cloud (VPC) に存在している必要があります。また、インスタンスの登録時の状態は `running` である必要があります。

コンソールを使用してターゲットをインスタンス ID で登録または登録解除するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインの [Load Balancing (ロードバランシング)] で [Target Groups (ターゲットグループ)] を選択します。
3. ターゲットグループの名前を選択して、その詳細ページを開きます。
4. [Targets] タブを選択します。
5. インスタンスを登録するには、[ターゲットの登録] を選択します。1 つ以上のインスタンスを選択し、必要に応じてデフォルトのインスタンスポートを入力して、[保留中として以下を含める] を選択します。インスタンスの追加が完了したら、[保留中のターゲットの登録] を選択します。

   [Note:] (メモ:)

   - IPv6 ターゲットグループに登録する場合、インスタンスにプライマリ IPv6 アドレスが割り当てられている必要があります。
   - AWS GovCloud (US) Regionはコンソールでのプライマリ IPv6 アドレスの割り当てをサポートしていません。でプライマリ IPv6 アドレスを割り当てるには、 API を使用する必要があります AWS GovCloud (US) Region。

6. インスタンスを登録解除するには、インスタンスを選択してから [登録解除] を選択します。

IP アドレスによるターゲットの登録または登録解除

IPv4 ターゲット

登録する IP アドレスは、次のいずれかの CIDR ブロックからのものである必要があります。

- ターゲットグループの VPC のサブネット
- 10.0.0.0/8 (RFC 1918)
- 100.64.0.0/10 (RFC 6598)
- 172.16.0.0/12 (RFC 1918)
- 192.168.0.0/16 (RFC 1918)

同じ VPC に別の Application Load Balancer の IP アドレスを登録することはできません。もう一方の Application Load Balancer が、ロードバランサー VPC にピアリング接続されている VPC に含まれている場合は、その IP アドレスを登録できます。

IPv6 ターゲット

- 登録する IP アドレスは、VPC CIDR ブロック内、またはピア接続された VPC CIDR ブロック内にある必要があります。

コンソールを使用してターゲットを IP アドレスで登録または登録解除するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインの [Load Balancing (ロードバランシング)] で [Target Groups (ターゲットグループ)] を選択します。
3. ターゲットグループの名前を選択して、その詳細ページを開きます。
4. [Targets] タブを選択します。
5. IP アドレスを登録するには、[ターゲットの登録] を選択します。IP アドレスごとにネットワークを選択し、IP アドレスかポートを入力して、[保留中として以下を含める] を選択します。アドレスの指定が終了したら、[保留中のターゲットの登録] を選択します。
6. IP アドレスの登録を解除するには、IP アドレスを選択して [登録解除] を選択します。登録済みの IP アドレスが多い場合は、フィルタを追加したりソート順を変更したりすると便利です。

Lambda 関数の登録または登録解除

各ターゲットグループに単一の Lambda 関数を登録できます。Elastic Load Balancing には、Lambda 関数を呼び出すための権限が必要です。トラフィックを Lambda 関数に送信する必要がなくなった場合は、登録を解除できます。Lambda 関数の登録を解除すると、未処理のリクエストは HTTP 5XX エラーで失敗します。Lambda 関数を置き換えるには、代わりに新しいターゲットグループを作成することをお勧めします。詳細については、「ターゲットとしての Lambda 関数」を参照してください。

コンソールを使用して Lambda 関数を登録または登録解除するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインの [Load Balancing (ロードバランシング)] で [Target Groups (ターゲットグループ)] を選択します。
3. ターゲットグループの名前を選択して、その詳細ページを開きます。
4. [Targets] タブを選択します。
5. 登録された Lambda 関数が表示されない場合は、[登録] を選択します。Lambda 関数を選択し、[登録] を選択します。
6. Lambda 関数を登録解除するには、[登録解除] を選択します。確認を求めるメッセージが表示されたら、[Deregister] を選択します。

AWS CLIを使用してターゲットを登録または登録解除する

ターゲットを追加するには register-targets コマンドを使用し、ターゲットを削除するには deregister-targets コマンドを使用します。

Application Load Balancer のスティッキーセッション

デフォルトでは、Application Load Balancer は、選択したロードバランシングアルゴリズムに基づいて、登録されたターゲットに各リクエストを個別にルーティングします。ただし、スティッキーセッション機能 (セッションアフィニティとも呼ばれます) を使用して、ロードバランサーがユーザーのセッションを特定のターゲットにバインドするように設定できます。これにより、ユーザーのセッション中のすべてのリクエストが同じターゲットに送信されます。これは、クライアントに連続したエクスペリエンスを提供するために状態情報を維持するサーバーに役立ちます。スティッキーセッションを使用するには、クライアントが Cookie をサポートする必要があります。

Application Load Balancer は、期間ベースの Cookie とアプリケーションベースの Cookie の両方をサポートします。ターゲットグループレベルでスティッキーセッションを有効にします。ターゲットグループで、期間ベースの維持、アプリケーションベースの維持、および維持しないの組み合わせを使用できます。

スティッキーセッションの管理において重要なのは、ロードバランサーがユーザーのリクエストを同じターゲットに一貫してルーティングする期間の決定です。アプリケーションに独自のセッション Cookie がある場合は、アプリケーションベースの維持を使用でき、ロードバランサーのセッション Cookie は、アプリケーションのセッション Cookie で指定された期間に従います。アプリケーションに独自のセッション Cookie がない場合、期間ベースの維持を使用して、指定した期間を持つロードバランサーセッション Cookie を生成できます。

ロードバランサーが生成した Cookie の内容は、回転キーを使用して暗号化されます。ロードバランサーが生成した Cookie を復号化または変更することはできません。

どの維持の種類でも、Application Load Balancer はリクエストごとに生成する Cookie の有効期限をリセットします。Cookie の有効期限が切れると、セッションは維持できなくなり、クライアントは Cookie ストアから Cookie を削除する必要があります。

要件

- HTTP/HTTPS ロードバランサー。
- 各アベイラビリティーゾーンに少なくとも 1 つの正常なインスタンスがあること。

考慮事項

- クロスゾーン負荷分散が無効な場合、スティッキーセッションはサポートされません。クロスゾーン負荷分散が無効なときにスティッキーセッションを有効にしようとしてもできません。
- アプリケーションベースの Cookie の場合、ターゲットグループごとに Cookie 名を個別に指定する必要があります。ただし、期間ベースの Cookie の場合、AWSALB はすべてのターゲットグループで使用される唯一の名前です。
- Application Load Balancers の複数のレイヤーを使用している場合、アプリケーションベースの Cookie を使用して、すべてのレイヤーでスティッキーセッションを有効にできます。ただし、期間ベースの Cookie の場合、AWSALB が使用可能な唯一の名前であるため、1 つのレイヤーでのみスティッキーセッションを有効にできます。
- アプリケーションベースの維持は、加重ターゲットグループでは動作しません。

- 複数のターゲットグループを含む転送アクションがあり、1 つ以上のターゲットグループでスティッキーセッションが有効になっている場合、ターゲットグループレベルの維持を有効にする必要があります。
- WebSocket 接続は本質的にスティッキーです。クライアントが への接続アップグレードをリクエストした場合 WebSockets、接続アップグレードを受け入れるために HTTP 101 ステータスコードを返すターゲットは、接続で使用される WebSocketsターゲットです。 WebSockets アップグレードが完了すると、Cookie ベースの維持は使用されません。
- Application Load Balancer は `Max-Age` 属性の代わりに Cookie ヘッダーの `Expires` 属性を使用します。
- Application Load Balancer は、URL エンコードされた Cookie 値をサポートしていません。

期間ベースの維持

期間ベースの維持は、ロードバランサーが生成した Cookie (AWSALB) を使用して、ターゲットグループ内の同じターゲットにリクエストをルーティングします。 Cookie は、セッションをターゲットにマッピングするために使用します。アプリケーションに独自のセッション Cookie がない場合、独自の維持期間を指定し、ロードバランサーがユーザーのリクエストを同じターゲットに一貫してルーティングする期間を管理できます。

ロードバランサーは、クライアントから最初のリクエストを受信すると、選択したアルゴリズムに基づいてリクエストをターゲットにルーティングし、AWSALB という名前の Cookie を生成します。これは、選択したターゲットに関する情報をエンコードしてCookie を暗号化し、クライアントへの応答に Cookie を含めます。ロードバランサーが生成した cookie には 7 日間の有効期限がありますが、これは設定できません。

後続のリクエストでは、クライアントは AWSALB Cookie を含める必要があります。ロードバランサーは Cookie を含むクライアントからリクエストを受信すると、それを検出し、同じターゲットにリクエストをルーティングします。Cookie は存在するがデコードできない場合、あるいは登録解除されたターゲットまたは異常なターゲットを参照している場合、ロードバランサーは新しいターゲットを選択し、新しいターゲットに関する情報で Cookie を更新します。

クロスオリジンリソース共有 (CORS) リクエストの場合、一部のブラウザでは維持を有効にするSameSite=None; Secure必要があります。これらのブラウザをサポートするために、ロードバランサーは常に 2 番目の維持 Cookie を生成します。これにはAWSALBCORS、元の維持 Cookie と同じ情報と SameSite 属性が含まれます。クライアントは、CORS 以外のリクエストを含む両方の Cookie を受け取ります。

コンソールを使用して期間ベースの維持を有効にするには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインの [Load Balancing (ロードバランシング)] で [Target Groups (ターゲットグループ)] を選択します。
3. ターゲットグループの名前を選択して、その詳細ページを開きます。
4. [グループの詳細] タブの [属性] セクションで、[編集] を選択します。
5. [Edit attributes] ページで、以下を実行します。
   a. [維持] を選択します。
   b. [維持の種類] で、[Load balancer generated cookie (ロードバランサーが生成した Cookie)] を選択します。
   c. [Stickiness duration] で、1 秒から 7 日の間の値を指定します。
   d. [変更の保存] をクリックします。

を使用して期間ベースの維持を有効にするには AWS CLI

`stickiness.enabled` および `stickiness.lb_cookie.duration_seconds` 属性を指定して modify-target-group-attributes コマンドを使用します。

期間ベースの維持を有効にするには、次のコマンドを使用します。

```
aws elbv2 modify-target-group-attributes --target-group-arn ARN --attributes
 Key=stickiness.enabled,Value=true
 Key=stickiness.lb_cookie.duration_seconds,Value=time-in-seconds
```

出力は次の例のようになります。

```
    {
       "Attributes": [
          ...
           {
                "Key": "stickiness.enabled",
                "Value": "true"
             },
             {
                 "Key": "stickiness.lb_cookie.duration_seconds",
                 "Value": "86500"
```

```
            },
          ...
        ]
    }
```

アプリケーションベースの維持

アプリケーションベースの維持では、クライアントターゲットの維持の独自の条件を設定できます。アプリケーションベースの維持を有効にすると、ロードバランサーは、選択したアルゴリズムに基づいてターゲットグループ内のターゲットに最初のリクエストをルーティングします。ターゲットは維持を有効にするために、ロードバランサーで設定した Cookie と一致するカスタムアプリケーション Cookie を設定することが想定されています。このカスタム Cookie には、アプリケーションが要求する Cookie 属性を含めることができます。

Application Load Balancer は、ターゲットからカスタムアプリケーション Cookie を受信すると、持続性情報をキャプチャする新しい暗号化アプリケーション Cookie を自動的に生成します。このロードバランサーが生成するアプリケーション Cookie は、アプリケーションベースの持続性が有効になっている各ターゲットグループの持続性情報をキャプチャします。

ロードバランサーが生成するアプリケーション Cookie は、ターゲットが設定するカスタムアプリケーション Cookie の属性をコピーしません。それには独自の 7 日の有効期限があり、設定できません。クライアントへの応答では、Application Load Balancer は、カスタム Cookie がターゲットグループレベルで設定された際に使用された名前のみが検証され、そのカスタム Cookie の値または有効期限属性は検証されません。名前が一致している限り、ロードバランサーは、ターゲットによって設定されたカスタム Cookie とロードバランサーによって生成されたアプリケーション Cookie の両方をクライアントへの応答で送信します。

後続のリクエストでは、クライアントは維持しておくために両方の Cookie を返送する必要があります。ロードバランサーは、アプリケーション Cookie を復号し、設定した持続期間がまだ有効かどうかを確認します。次に、Cookie 内の情報を使用して、ターゲットグループ内の同じターゲットにリクエストを送信し、維持しておきます。また、ロードバランサーは、カスタムアプリケーション Cookie を検査または変更することなく、ターゲットにプロキシします。それ以降の応答では、ロードバランサーが生成したアプリケーション Cookie の有効期限と、ロードバランサーで設定された持続期間がリセットされます。クライアントとターゲット間の持続性を維持するため、Cookie の有効期限と持続時間が経過しないようにします。

ターゲットが失敗した場合または異常が発生した場合、ロードバランサーはそのターゲットへのリクエストのルーティングを停止し、選択した負荷分散アルゴリズムに基づいて、新しい正常なターゲッ

トを選択します。ロードバランサーは、セッションを新しい正常なターゲットに「スタック」しているものとして処理し、失敗したターゲットが戻った場合でも、新しい正常なターゲットへのリクエストのルーティングを続行します。

クロスオリジンリソース共有 (CORS) リクエストの場合、持続性を有効にするために、ロードバランサーはユーザーエージェントのバージョンが Chromium80 以上の場合にのみ `SameSite=None; Secure` 属性をロードバランサーが生成するアプリケーション Cookie に追加します。

ほとんどのブラウザは Cookie のサイズを 4K に制限しているため、ロードバランサーは 4K を超えるアプリケーション Cookie を複数の Cookie にシャードします。Application Load Balancer は、最大 16K のサイズの Cookie をサポートするため、クライアントに送信するシャードを最大 4 つ作成できます。クライアントに表示されるアプリケーション Cookie 名はAWSALBAPP「-」で始まり、フラグメント番号が含まれます。例えば、Cookie のサイズが 0-4Kの場合、クライアントは AWSALBAPP-0 と表示されます。Cookie のサイズが 4～8k の場合、クライアントは AWSALBAPP-0 と AWSALBAPP-1 などを表示します。

コンソールを使用してアプリケーションベースの維持を有効にするには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインの [Load Balancing (ロードバランシング)] で [Target Groups (ターゲットグループ)] を選択します。
3. ターゲットグループの名前を選択して、その詳細ページを開きます。
4. [グループの詳細] タブの [属性] セクションで、[編集] を選択します。
5. [Edit attributes] ページで、以下を実行します。

    a. [維持] を選択します。
    b. [維持の種類] で、[アプリケーションベース Cookie] を選択します。
    c. [Stickiness duration] で、1 秒から 7 日の間の値を指定します。
    d. [アプリ Cookie 名] に、アプリケーションベースの Cookie 名を入力します。

        Cookie 名に `AWSALB`、`AWSALBAPP`、または `AWSALBTG` を使用しないでください。これらは、ロードバランサーで使用するために予約されています。

    e. [変更の保存] をクリックします。

を使用してアプリケーションベースの維持を有効にするには AWS CLI

次の属性で modify-target-group-attributes コマンドを使用します。

- stickiness.enabled
- stickiness.type
- stickiness.app_cookie.cookie_name
- stickiness.app_cookie.duration_seconds

アプリケーションベースの維持を有効にするには、次のコマンドを使用します。

```
aws elbv2 modify-target-group-attributes --target-group-arn ARN --attributes
 Key=stickiness.enabled,Value=true Key=stickiness.type,Value=app_cookie
 Key=stickiness.app_cookie.cookie_name,Value=my-cookie-name
 Key=stickiness.app_cookie.duration_seconds,Value=time-in-seconds
```

出力は次の例のようになります。

```
      {
           "Attributes": [
               ...
               {
                   "Key": "stickiness.enabled",
                   "Value": "true"
               },
               {
                   "Key": "stickiness.app_cookie.cookie_name",
                   "Value": "MyCookie"
               },
               {
                   "Key": "stickiness.type",
                   "Value": "app_cookie"
               },
               {
                   "Key": "stickiness.app_cookie.duration_seconds",
                   "Value": "86500"
               },
               ...
           ]
       }
```

手動再分散

スケールアップ時にターゲット数が大幅に増加すると、維持による負荷の分散が不均等になる可能性があります。このシナリオでは、次の 2 つのオプションを使用して、ターゲットの負荷を再分散できます。

- アプリケーションが生成した Cookie に現在の日付と時刻より前の有効期限を設定します。これにより、クライアントが Application Load Balancer に Cookie を送信できなくなり、維持の確立プロセスが再開されます。
- ロードバランサーのアプリケーションベースの維持設定で、1 秒など非常に短い時間を設定します。これにより、ターゲットが設定した Cookie の有効期限が切れていない場合でも、Application Load Balancer は維持を再確立します。

ターゲットとしての Lambda 関数

Lambda 関数をターゲットとして登録し、Lambda 関数のターゲットグループにリクエストを転送するリスナールールを設定できます。ロードバランサーが Lambda 関数をターゲットとしてターゲットグループにリクエストを転送すると、Lambda 関数を呼び出し、リクエストのコンテンツを JSON 形式で Lambda 関数に渡します。

制限

- Lambda 関数とターゲットグループは、同じアカウントおよび同じリージョンにある必要があります。
- Lambda 関数に送信できるリクエストボディの最大サイズは 1 MB です。関連するサイズ制限の詳細については、HTTP ヘッダーの制限を参照してください。
- Lambda 関数が送信できるレスポンス JSON の最大サイズは 1 MB です。
- WebSockets はサポートされていません。アップグレードのリクエストは HTTP 400 コードで拒否されます。
- Local Zones はサポートされていません。
- 自動ターゲット重み (ATW) はサポートされていません。

内容

- 複数値ヘッダー
- ヘルスチェックの有効化
- Lambda 関数の登録解除

デモについては、Application Load Balancer の Lambda ターゲットを参照してください。

Lambda 関数の準備

Application Load Balancer で Lambda 関数を使用している場合は、以下の推奨事項が適用されます。

Lambda 関数を呼び出すアクセス許可

ターゲットグループを作成し、 AWS Management Consoleを使用して Lambda 関数を登録すると、コンソールは必要なアクセス権限を自動的に Lambda 関数ポリシーに追加します。それ以外の場合は、ターゲットグループを作成し、 を使用して関数を登録した後 AWS CLI、add-permission コマンドを使用して、Lambda 関数を呼び出すアクセス許可を Elastic Load Balancing に付与する必要があります。`aws:SourceAccount` および `aws:SourceArn` の条件キーを使用して、関数の呼び出しを指定のターゲットグループに制限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「「混乱した代理」問題」を参照してください。

```
aws lambda add-permission \
--function-name lambda-function-arn-with-alias-name \
--statement-id elb1 \
--principal elasticloadbalancing.amazonaws.com \
--action lambda:InvokeFunction \
--source-arn target-group-arn \
--source-account target-group-account-id
```

Lambda 関数のバージョニング

ターゲットグループごとに 1 つの Lambda 関数を登録できます。Lambda 関数を変更し、ロードバランサーが常に現行バージョンの Lambda 関数を呼び出せるようにするには、関数のエイリアスを作成し、ロードバランサーに Lambda 関数を登録するときに関数 ARN にエイリアスを含めます。詳細については、AWS Lambda デベロッパーガイドの「AWS Lambda 関数のバージョニングとエイリアス」および「エイリアスを使用したトラフィックの移行」を参照してください。

関数タイムアウト

ロードバランサーは、Lambda 関数が応答またはタイムアウトするまで待機します。予期される実行時間に基づいて Lambda 関数のタイムアウトを設定することをお勧めします。デフォルトのタイムアウト値と、その変更方法の詳細については、「基本的な AWS Lambda 関数を設定する」を参照してください。設定できる最大のタイムアウト値の詳細については、「AWS Lambda の制限」を参照してください。

Lambda 関数のターゲットグループの作成

リクエストルーティングで使用されるターゲットグループを作成します。リクエストのコンテンツが、コンテンツをこのターゲットグループに転送するアクションを含むリスナールールと一致する場合、ロードバランサーは登録された Lambda 関数を呼び出します。

コンソールを使用してターゲットグループを作成し、Lambda 関数を登録するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインの [ロードバランシング] で [ターゲットグループ] を選択します。
3. [ターゲットグループの作成] を選択します。
4. [ターゲットタイプの選択] で [Lambda 関数] を選択します。
5. [Target group name] で、ターゲットグループの名前を入力します。
6. (オプション) ヘルスチェックを有効にするには、[ヘルスチェック] セクションで [有効化] を選択します。
7. (オプション) 次のように 1 つ以上のタグを追加します。
   a. [Tags (タグ)] セクションを展開します。
   b. [Add tag] を選択します。
   c. タグキーとタグ値を入力します。
8. [次へ] をクリックします。
9. 単一の Lambda 関数を指定するか、このステップを省略して、後で Lambda 関数を指定します。
10. [ターゲットグループの作成] を選択します。

ターゲットグループを作成し、 AWS CLIを使用して Lambda 関数を登録するには

create-target-group と register-targets コマンドを使用します。

ロードバランサーからのイベントの受け取り

ロードバランサーは、HTTP と HTTPS の両方でリクエストの Lambda 呼び出しをサポートしています。ロードバランサーは、JSON 形式でイベントを送信します。ロードバランサーは、リクエストごとに X-Amzn-Trace-Id、X-Forwarded-For、X-Forwarded-Port、X-Forwarded-Proto の各ヘッダーを追加します。

content-encoding ヘッダーが存在する場合、ロードバランサー Base64 は本体をエンコードし、isBase64Encoded を true に設定します。

content-encoding ヘッダーが存在しない場合、Base64 エンコーディングはコンテンツタイプによって異なります。タイプが text/*、application/json、application/javascript、application/xml である場合、ロードバランサーは本文をそのまま送信し、isBase64Encoded を false に設定します。それ以外の場合、ロードバランサー Base64 は本文をエンコードし、isBase64Encoded を true に設定します。

以下に示しているのは、イベントの例です。

```
{
    "requestContext": {
        "elb": {
            "targetGroupArn":
 "arn:aws:elasticloadbalancing:region:123456789012:targetgroup/my-target-
group/6d0ecf831eec9f09"
        }
    },
    "httpMethod": "GET",
    "path": "/",
    "queryStringParameters": {parameters},
    "headers": {
        "accept": "text/html,application/xhtml+xml",
        "accept-language": "en-US,en;q=0.8",
        "content-type": "text/plain",
        "cookie": "cookies",
        "host": "lambda-846800462-us-east-2.elb.amazonaws.com",
        "user-agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6)",
        "x-amzn-trace-id": "Root=1-5bdb40ca-556d8b0c50dc66f0511bf520",
        "x-forwarded-for": "72.21.198.66",
        "x-forwarded-port": "443",
        "x-forwarded-proto": "https"
    },
    "isBase64Encoded": false,
```

```
    "body": "request_body"
}
```

ロードバランサーへの応答

Lambda 関数からのレスポンスには、Base64 エンコーディングのステータス、ステータスコード、およびヘッダーが含まれます。本文は省略できます。

レスポンス本文にバイナリコンテンツを含めるには、コンテンツを Base64 でエンコードし、isBase64Encoded を true に設定する必要があります。ロードバランサーはコンテンツをデコードしてバイナリコンテンツを取得し、そのコンテンツを HTTP レスポンスの本文でクライアントに送信します。

ロードバランサーは、 Connectionや などの hop-by-hop ヘッダーを尊重しませんTransfer-Encoding。ロードバランサーがクライアントにレスポンスを送信する前に計算するため、Content-Length ヘッダーは省略できます。

nodejs をベースとした Lambda 関数からのレスポンスの例を次に示します。

```
{
    "isBase64Encoded": false,
    "statusCode": 200,
    "statusDescription": "200 OK",
    "headers": {
        "Set-cookie": "cookies",
        "Content-Type": "application/json"
    },
    "body": "Hello from Lambda (optional)"
}
```

Application Load Balancer で動作する Lambda 関数のテンプレートについては、github の application-load-balancer-serverless-app を参照してください。または、Lambda コンソールを開き、[アプリケーション] から [アプリケーションを作成] を選択し、 AWS Serverless Application Repository から次のいずれかを選択します。

- ALB-Lambda-ターゲット-UploadFiletoS3
- ALB-Lambda-ターゲット-BinaryResponse
- ALB-Lambda-Target-WhatisMyIP

複数値ヘッダー

クライアントからのリクエストまたは Lambda 関数からのレスポンスに複数の値を持つヘッダーが含まれている場合、同じヘッダーが複数回含まれている場合、あるいは同じキーに対して複数の値を持つクエリパラメータが含まれている場合は、複数値のヘッダー構文のサポートを有効にできます。複数値のヘッダーを有効にすると、ロードバランサーと Lambda 関数の間で交換されるヘッダーとクエリパラメータは、文字列ではなく配列を使用します。複数値のヘッダー構文を有効にせず、ヘッダーまたはクエリパラメータに複数の値が含まれている場合、ロードバランサーは受け取った最後の値を使用します。

目次

複数値ヘッダーを持つリクエスト

ヘッダーおよびクエリ文字列パラメータに使用されるフィールドの名前は、ターゲットグループに対して複数値ヘッダーを有効にするかどうかによって異なります。

次のリクエスト例には、同じキーを持つ 2 つのクエリパラメータがあります。

```
http://www.example.com?&myKey=val1&myKey=val2
```

デフォルトの形式では、ロードバランサーはクライアントによって送信された最後の値を使用し、queryStringParameters を使用してクエリ文字列パラメータを含むイベントを送信します。以下に例を示します。

```
"queryStringParameters": { "myKey": "val2"},
```

複数値ヘッダーを有効にした場合、ロードバランサーはクライアントから送信された両方のキー値を使用し、multiValueQueryStringParameters を使用してクエリ文字列パラメータを含むイベントを送信します。以下に例を示します。

```
"multiValueQueryStringParameters": { "myKey": ["val1", "val2"] },
```

同様に、クライアントがヘッダーに 2 つの Cookie を含むリクエストを送信するとします。

```
"cookie": "name1=value1",
"cookie": "name2=value2",
```

デフォルトの形式では、ロードバランサーはクライアントによって送信された最後の Cookie を使用し、headers を使用してヘッダーを含むイベントを送信します。以下に例を示します。

```
"headers": {
    "cookie": "name2=value2",
    ...
},
```

複数値ヘッダーを有効にすると、ロードバランサーはクライアントによって送信された両方の Cookie を使用し、multiValueHeaders を使用してヘッダーを含むイベントを送信します。以下に例を示します。

```
"multiValueHeaders": {
    "cookie": ["name1=value1", "name2=value2"],
    ...
},
```

クエリパラメータが URL エンコードされている場合、ロードバランサーはそれらをデコードしません。その場合は Lambda 関数でデコードする必要があります。

複数値ヘッダーを持つレスポンス

ヘッダーに使用されるフィールドの名前は、ターゲットグループに対して複数値ヘッダーを有効にするかどうかによって異なります。複数値ヘッダーを有効にしている場合は multiValueHeaders を使用し、それ以外の場合は headers を使用する必要があります。

デフォルトの形式では、単一の Cookie を指定できます。

```
{
  "headers": {
      "Set-cookie": "cookie-name=cookie-value;Domain=myweb.com;Secure;HttpOnly",
      "Content-Type": "application/json"
  },
}
```

複数値のヘッダーを有効にした場合、複数の Cookie を以下のように指定する必要があります。

```
{
  "multiValueHeaders": {
      "Set-cookie": ["cookie-name=cookie-
value;Domain=myweb.com;Secure;HttpOnly","cookie-name=cookie-value;Expires=May 8,
 2019"],
      "Content-Type": ["application/json"]
  },
}
```

ロードバランサーは、Lambda レスポンスペイロードで指定されたのとは異なる順序でヘッダーをクライアントに送信する場合があります。したがって、ヘッダーが特定の順序で返されるとは限らないことに留意してください。

複数値ヘッダーの有効化

ターゲットの種類が `lambda` であるターゲットグループに対して、複数値のヘッダーを有効または無効にすることができます。

コンソールを使用して複数値のヘッダーを有効にするには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインの [Load Balancing (ロードバランシング)] で [Target Groups (ターゲットグループ)] を選択します。
3. ターゲットグループの名前を選択して、その詳細ページを開きます。
4. [グループの詳細] タブの [属性] セクションで、[編集] を選択します。
5. [複数値のヘッダー] を選択または選択解除します。
6. [変更の保存] をクリックします。

を使用して複数値ヘッダーを有効にするには AWS CLI

`lambda.multi_value_headers.enabled` 属性を指定して modify-target-group-attributes コマンドを使用します。

ヘルスチェックの有効化

デフォルトでは、ヘルスチェックは種類が `lambda` のターゲットグループに対しては無効になっています。Amazon Route 53 を使用して DNS フェイルオーバーを実装するには、ヘルスチェックを有効にできます。Lambda 関数は、ヘルスチェックリクエストに応答する前に、ダウンストリーム

サービスの状態を確認できます。Lambda 関数からのレスポンスでヘルスチェックの失敗が示された場合、ヘルスチェックの失敗が Route 53 に渡されます。バックアップアプリケーションスタックにフェイルオーバーするよう Route 53 を設定できます。

ヘルスチェックについては、他の Lambda 関数の呼び出しと同じように課金されます。

Lambda 関数に送信されるヘルスチェックの形式は次のとおりです。イベントがヘルスチェックイベントかどうかを確認するには、ユーザーエージェントフィールドの値を確認します。ヘルスチェックのユーザーエージェントは ELB-HealthChecker/2.0 です。

```
{
    "requestContext": {
        "elb": {
            "targetGroupArn":
 "arn:aws:elasticloadbalancing:region:123456789012:targetgroup/my-target-
group/6d0ecf831eec9f09"
        }
    },
    "httpMethod": "GET",
    "path": "/",
    "queryStringParameters": {},
    "headers": {
        "user-agent": "ELB-HealthChecker/2.0"
    },
    "body": "",
    "isBase64Encoded": false
}
```

コンソールを使用してターゲットグループのヘルスチェックを有効にするには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインの [Load Balancing (ロードバランシング)] で [Target Groups (ターゲットグループ)] を選択します。
3. ターゲットグループの名前を選択して、その詳細ページを開きます。
4. [グループの詳細] タブの [ヘルスチェックの設定] セクションで、[編集] を選択します。
5. [ヘルスチェック] で、[有効化] を選択します。
6. [変更の保存] をクリックします。

を使用してターゲットグループのヘルスチェックを有効にするには AWS CLI

modify-target-group コマンドを使用し、`--health-check-enabled` オプションを指定します。

Lambda 関数の登録解除

トラフィックを Lambda 関数に送信する必要がなくなった場合は、登録を解除できます。Lambda 関数の登録を解除すると、未処理のリクエストは HTTP 5XX エラーで失敗します。

Lambda 関数を置き換えるには、新しいターゲットグループを作成し、新しい関数を新しいターゲットグループに登録し、リスナールールを更新して既存のターゲットグループではなく新しいターゲットグループを使用することをお勧めします。

コンソールを使用して Lambda 関数の登録を解除するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインの [Load Balancing (ロードバランシング)] で [Target Groups (ターゲットグループ)] を選択します。
3. ターゲットグループの名前を選択して、その詳細ページを開きます。
4. [ターゲット] タブで、[登録解除] を選択します。
5. 確認を求めるメッセージが表示されたら、[Deregister] を選択します。

を使用して Lambda 関数の登録を解除するには AWS CLI

deregister-targets コマンドを使用します。

ターゲットグループのタグ

タグを使用すると、ターゲットグループを目的、所有者、環境などさまざまな方法で分類することができます。

各ターゲットグループに対して複数のタグを追加できます。タグキーは、各ターゲットグループで一意である必要があります。すでにターゲットグループに関連付けられているキーを持つタグを追加すると、そのキーの値が更新されます。

不要になったタグは、削除することができます。

制限事項

- リソースあたりのタグの最大数 – 50
- キーの最大長 – 127 文字 (Unicode)

- 値の最大長 – 255 文字 (Unicode)
- タグのキーと値は大文字と小文字が区別されます。使用できる文字は、UTF-8 で表現できる文字、スペース、および数字と、特殊文字 (+、-、=、.、_、:、/、@) です。ただし、先頭または末尾にはスペースを使用しないでください。
- タグ名または値に aws: プレフィックスを使用しないでください。このプレフィックスは AWS 用に予約されています。このプレフィックスが含まれるタグの名前または値は編集または削除できません。このプレフィックスを持つタグは、リソースあたりのタグ数の制限時には計算されません。

コンソールを使用してターゲットグループのタグを更新するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインの [Load Balancing (ロードバランシング)] で [Target Groups (ターゲットグループ)] を選択します。
3. ターゲットグループの名前を選択して、その詳細ページを開きます。
4. [タグ] タブで、[タグの管理] を選択し、次の 1 つ以上の操作を行います。
   a. タグを更新するには、[キー] と [値] に新しい値を入力します。
   b. タグを追加するには、[タグの追加] を選択し、[キー] と [値] に値を入力します。
   c. タグを削除するには、タグの横にある [削除] を選択します。
5. タグの更新を完了したら、[変更内容の保存] を選択します。

を使用してターゲットグループのタグを更新するには AWS CLI

add-tags コマンドと remove-tags コマンドを使用します。

ターゲットグループの削除

ターゲットグループがリスナールールの転送アクションによって参照されていない場合は、これを削除できます。ターゲットグループを削除しても、ターゲットグループに登録されたターゲットには影響が及びません。登録済み EC2 インスタンスが必要なくなった場合は停止または終了できます。

コンソールを使用してターゲットグループを削除するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインの [Load Balancing (ロードバランシング)] で [Target Groups (ターゲットグループ)] を選択します。

3. ターゲットグループを選択し、[Actions]、[Delete] を選択します。
4. 確認を求めるメッセージが表示されたら、[はい、削除します] を選択します。

を使用してターゲットグループを削除するには AWS CLI

delete-target-group コマンドを使用します。

Application Load Balancer を監視する

次の機能を使用して、ロードバランサーの監視、トラフィックパターンの分析、ロードバランサーとターゲットに関する問題の解決を実行できます。

CloudWatch メトリクス

Amazon を使用して CloudWatch 、ロードバランサーとターゲットのデータポイントに関する統計を、メトリクス と呼ばれる時系列データの順序付けられたセットとして取得できます。これらのメトリクスを使用して、システムが正常に実行されていることを確認できます。詳細については、「CloudWatch Application Load Balancer の メトリクス」を参照してください。

アクセスログ

アクセスログを使用して、ロードバランサーに対して行われたリクエストの詳細情報をキャプチャし、Amazon S3 でログファイルとして保存できます。これらのアクセスログを使用して、トラフィックパターンの分析や、ターゲットの問題のトラブルシューティングを行うことができます。詳細については、「Application Load Balancer のアクセスログ」を参照してください。

接続ログ

接続ログを使用して、ロードバランサーに送信されたリクエストに関する属性をキャプチャし、ログファイルとして Amazon S3 に保存できます。これらの接続ログを使用して、使用されているクライアントの IP アドレスとポート、クライアント証明書情報、接続結果、TLS 暗号を特定できます。その後、これらの接続ログを使用して、リクエストパターンやその他の傾向を確認できます。詳細については、「Application Load Balancer の接続ログ」を参照してください。

リクエストのトレース

リクエストのトレースを使用して HTTP リクエストを追跡できます。ロードバランサーは、受け取った各リクエストにトレース識別子を持つヘッダーを追加します。詳細については、「Application Load Balancer のリクエストをトレースする」を参照してください。

CloudTrail ログ

を使用して AWS CloudTrail 、Elastic Load Balancing API に対して行われた呼び出しに関する詳細情報をキャプチャし、ログファイルとして Amazon S3 に保存できます。これらの CloudTrail ログを使用して、どの呼び出しが行われたか、呼び出し元の送信元 IP アドレス、呼び出し者、呼び出し日時などを確認できます。詳細については、「AWS CloudTrailによる Application Load Balancer での API 呼び出しのログ記録」を参照してください。

CloudWatch Application Load Balancer の メトリクス

Elastic Load Balancing は、ロードバランサーとターゲット CloudWatch のデータポイントを Amazon に発行します。 CloudWatch を使用すると、これらのデータポイントに関する統計を、メトリクス と呼ばれる時系列データの順序付けられたセットとして取得できます。メトリクスは監視対象の変数、データポイントは時間の経過と共に変わる変数の値と考えることができます。たとえば、指定した期間中のロードバランサーの正常なターゲットの合計数を監視することができます。各データポイントには、タイムスタンプと、オプションの測定単位が関連付けられています。

メトリクスを使用して、システムが正常に実行されていることを確認できます。例えば、指定したメトリクスをモニタリングする CloudWatch アラームを作成し、メトリクスが許容範囲外になった場合にアクション (E メールアドレスへの通知の送信など) を開始できます。

Elastic Load Balancing は、リクエストがロードバランサーを流れる CloudWatch 場合にのみ、メトリクスを にレポートします。ロードバランサーを経由するリクエストがある場合、Elastic Load Balancing は 60 秒間隔でメトリクスを測定し、送信します。ロードバランサーを経由するリクエストがないか、メトリクスのデータがない場合、メトリクスは報告されません。

詳細については、「Amazon ユーザーガイド CloudWatch 」を参照してください。

内容

Application Load Balancer のメトリック

- ロードバランサー
- [Targets] (ターゲット)
- ターゲットグループの正常性
- Lambda 関数
- ユーザー認証

AWS/ApplicationELB 名前空間には、以下のロードバランサーのメトリクスが含まれます。

メトリクス	説明
ActiveConnectionCount	クライアントからロードバランサーへ、およびロードバランサーからターゲットへの、アクティブな同時 TCP 接続の総数。 レポート条件: ゼロ以外の値がある 統計: 最も有用な統計は Sum です。 ディメンション • LoadBalancer • AvailabilityZone , LoadBalancer
AnomalousHostCount	異常で検出されたホストの数。 レポート条件: 常に報告される 統計値: 最も有用な統計値は Average、Minimum、および Maximum です。 ディメンション • TargetGroup , LoadBalancer • TargetGroup , AvailabilityZone , LoadBalancer
ClientTLSNegotiationErrorCount	クライアントにより開始され、TLS エラーのためにロードバランサーとのセッションを確立しなかった、TLS 接続の数。考えられる原因としては、暗号やプロトコルの不一致、クライアントがサーバー証明書を検証できないため接続を閉じるなどがあります。 レポート条件: ゼロ以外の値がある 統計: 最も有用な統計は Sum です。 ディメンション • LoadBalancer • AvailabilityZone , LoadBalancer

メトリクス	説明
ConsumedLCUs	ロードバランサーが使用するロードバランサーキャパシティーユニット (LCU) の数です。1 時間当たりで使用する LCU 数の料金をお支払いいただきます。詳細については、Elastic Load Balancing の料金表を参照してください。 レポート条件: 常に報告される 統計: All ディメンション • LoadBalancer
DesyncMitigationMode_NonCompliant_Request_Count	RFC 7230 に準拠していないリクエストの数。 レポート条件: ゼロ以外の値がある 統計: 最も有用な統計は Sum です。 ディメンション • LoadBalancer • AvailabilityZone , LoadBalancer
DroppedInvalidHeaderRequestCount	リクエストをルーティングする前に、ロードバランサーが無効なヘッダーフィールドを持つ HTTP ヘッダーを削除したリクエストの数。ロードバランサーは、routing.http.drop_invalid_header_fields.enabled 属性が true に設定されている場合にのみこれらのヘッダーを削除します。 レポート条件: ゼロ以外の値がある 統計: All ディメンション • AvailabilityZone , LoadBalancer

メトリクス	説明
MitigatedHostCount	緩和対象のターゲットの数。 レポート条件: 常に報告される 統計値: 最も有用な統計値は Average、Minimum、および Maximum です。 ディメンション • TargetGroup , LoadBalancer • TargetGroup , AvailabilityZone , LoadBalancer
ForwardedInvalidHeaderRequestCount	無効なヘッダーフィールドを持つ HTTP ヘッダーがあるロードバランサーによってルーティングされたリクエストの数。ロードバランサーは、routing.http.drop_invalid_header_fields.enabled 属性が false に設定されている場合にのみ、これらのヘッダーを使用してリクエストを転送します。 レポート条件: 常に報告される 統計: All ディメンション • AvailabilityZone , LoadBalancer
GrpcRequestCount	IPv4 および IPv6 経由で処理された gRPC リクエストの数。 レポート条件: ゼロ以外の値がある 統計 : 最も有用な統計は Sum です。Minimum、Maximum、Average のすべてが 1 を返します。 ディメンション • LoadBalancer • AvailabilityZone , LoadBalancer

メトリクス	説明
HTTP_Fixed_Response_Count	成功した固定レスポンスアクションの数。 レポート条件: ゼロ以外の値がある [Statistics] (統計): 唯一意味のある統計は Sum です。 ディメンション • LoadBalancer • AvailabilityZone , LoadBalancer
HTTP_Redirect_Count	成功したリダイレクトアクションの数。 レポート条件: ゼロ以外の値がある [Statistics] (統計): 唯一意味のある統計は Sum です。 ディメンション • LoadBalancer • AvailabilityZone , LoadBalancer
HTTP_Redirect_Url_Limit_Exceeded_Count	レスポンスの Location ヘッダーの URL が 8K を超えているために、リダイレクトアクションを完了できなかった数。 レポート条件: ゼロ以外の値がある [Statistics] (統計): 唯一意味のある統計は Sum です。 ディメンション • LoadBalancer • AvailabilityZone , LoadBalancer

メトリクス	説明
HTTPCode_ELB_3XX_Count	ロードバランサーから送信された HTTP 3XX リダイレクトコードの数。この数には、ターゲットによって生成される応答コードは含まれません。 レポート条件: ゼロ以外の値がある [Statistics] (統計): 唯一意味のある統計は Sum です。 ディメンション • LoadBalancer • AvailabilityZone , LoadBalancer
HTTPCode_ELB_4XX_Count	ロードバランサーから送信される HTTP 4XX クライアントエラーコードの数。この数には、ターゲットによって生成される応答コードは含まれません。 リクエストの形式が不正な場合、または不完全な場合は、クライアントエラーが生成されます。ロードバランサーが HTTP 460 エラーコードを返す場合を除き、これらのリクエストはターゲットで受信されませんでした。この数には、ターゲットによって生成される応答コードは含まれません。 レポート条件: ゼロ以外の値がある 統計 : 最も有用な統計は Sum です。Minimum、Maximum、Average のすべてが 1 を返します。 ディメンション • LoadBalancer • AvailabilityZone , LoadBalancer

メトリクス	説明
HTTPCode_ELB_5XX_Count	ロードバランサーから送信される HTTP 5XX サーバーエラーコードの数。この数には、ターゲットによって生成される応答コードは含まれません。 レポート条件: ゼロ以外の値がある 統計 : 最も有用な統計は Sum です。Minimum、Maximum、Average のすべてが 1 を返します。 ディメンション • LoadBalancer • AvailabilityZone , LoadBalancer
HTTPCode_ELB_500_Count	ロードバランサーから送信される HTTP 500 エラーコードの数。 レポート条件: ゼロ以外の値がある [Statistics] (統計): 唯一意味のある統計は Sum です。 ディメンション • LoadBalancer • AvailabilityZone , LoadBalancer
HTTPCode_ELB_502_Count	ロードバランサーから送信される HTTP 502 エラーコードの数。 レポート条件: ゼロ以外の値がある [Statistics] (統計): 唯一意味のある統計は Sum です。 ディメンション • LoadBalancer • AvailabilityZone , LoadBalancer

メトリクス	説明
HTTPCode_ELB_503_Count	ロードバランサーから送信される HTTP 503 エラーコードの数。 レポート条件: ゼロ以外の値がある [Statistics] (統計): 唯一意味のある統計は Sum です。 ディメンション • LoadBalancer • AvailabilityZone , LoadBalancer
HTTPCode_ELB_504_Count	ロードバランサーから送信される HTTP 504 エラーコードの数。 レポート条件: ゼロ以外の値がある [Statistics] (統計): 唯一意味のある統計は Sum です。 ディメンション • LoadBalancer • AvailabilityZone , LoadBalancer
IPv6ProcessedBytes	IPv6 を使用したロードバランサーによって処理される総バイト数。この数は ProcessedBytes に含まれています。 レポート条件: ゼロ以外の値がある 統計: 最も有用な統計は Sum です。 ディメンション • LoadBalancer • AvailabilityZone , LoadBalancer

メトリクス	説明
IPv6RequestCount	ロードバランサーによって受信された IPv6 リクエストの数。 レポート条件: ゼロ以外の値がある 統計 : 最も有用な統計は Sum です。Minimum、Maximum、Average のすべてが 1 を返します。 ディメンション • LoadBalancer • AvailabilityZone , LoadBalancer
NewConnectionCount	クライアントからロードバランサーへ、およびロードバランサーからターゲットへの、新たに確立された TCP 接続の総数。 レポート条件: ゼロ以外の値がある 統計: 最も有用な統計は Sum です。 ディメンション • LoadBalancer • AvailabilityZone , LoadBalancer

メトリクス	説明
NonStickyRequestCount	既存のスティッキーセッションを使用できなかったために、ロードバランサーが新しいターゲットを選択したリクエストの数。たとえば、リクエストが新しいクライアントから最初のリクエストで、維持 Cookie が提示されなかった、維持 Cookie が提示されたが、このターゲットグループに登録されたターゲットを指定しなかった、維持 Cookie の形式が誤っているか期限切れであった、内部エラーによりロードバランサーは維持 Cookie を読み取れなかったなどです。 レポート条件: 維持設定がターゲットグループで有効になっている。 [Statistics] (統計): 唯一意味のある統計は Sum です。 ディメンション • LoadBalancer • AvailabilityZone , LoadBalancer
ProcessedBytes	IPv4 および IPv6 を使用したロードバランサーによって処理される総バイト数 (HTTP ヘッダーおよび HTTP ペイロード)。この数には、クライアントとの間および Lambda 関数との間で送受信されるトラフィックに加えて、ユーザー認証が有効な場合は ID プロバイダー (IdP) からのトラフィックが含まれます。 レポート条件: ゼロ以外の値がある 統計: 最も有用な統計は Sum です。 ディメンション • LoadBalancer • AvailabilityZone , LoadBalancer

メトリクス	説明
RejectedConnectionCount	ロードバランサーが接続の最大数に達したため、拒否された接続の数。 レポート条件: ゼロ以外の値がある 統計: 最も有用な統計は Sum です。 ディメンション • LoadBalancer • AvailabilityZone , LoadBalancer
RequestCount	IPv4 および IPv6 経由で正常に処理されたリクエストの数。このメトリクスは、ロードバランサーノードがターゲットを選択できたリクエストに対してのみ増分されます。ターゲットが選択される前に拒否されたリクエストは、このメトリクスには反映されません。 レポート条件: 常に報告される 統計: 最も有用な統計は Sum です。 ディメンション • LoadBalancer • LoadBalancer , AvailabilityZone • LoadBalancer , TargetGroup • LoadBalancer , AvailabilityZone , TargetGroup
RuleEvaluations	1 時間の平均リクエスト頻度に基づいてロードバランサーによって処理されるルールの数。 レポート条件: ゼロ以外の値がある 統計: 最も有用な統計は Sum です。 ディメンション • LoadBalancer

AWS/ApplicationELB 名前空間には、以下のターゲットのメトリクスが含まれます。

メトリクス	説明
HealthyHostCount	正常と見なされるターゲットの数。 レポート条件: ヘルスチェックが有効になっている場合にレポートされます 統計値: 最も有用な統計値は Average、Minimum、および Maximum です。 ディメンション • LoadBalancer , TargetGroup • LoadBalancer , AvailabilityZone , TargetGroup
HTTPCode_Target_2XX_Count , HTTPCode_Target_3XX_Count , HTTPCode_Target_4XX_Count , HTTPCode_Target_5XX_Count	ターゲットによって生成された HTTP 応答コードの数。これには、ロードバランサーによって生成される応答コードは含まれません。 レポート条件: ゼロ以外の値がある 統計 : 最も有用な統計は Sum です。Minimum、Maximum、Average のすべてが 1 を返します。 ディメンション • LoadBalancer • AvailabilityZone , LoadBalancer • TargetGroup , LoadBalancer • TargetGroup , AvailabilityZone , LoadBalancer
RequestCountPerTarget	ターゲットグループ内のターゲットあたりの平均リクエスト数。TargetGroup ディメンションを使用してターゲットグループを指定する必要があります。ターゲットが Lambda 関数である場合、このメトリクスは適用されません。 この数は、ターゲットグループが受信したリクエストの総数を、ターゲットグループ内の正常なターゲットの数で割ったものを使用

メトリクス	説明
	します。ターゲットグループに正常なターゲットがない場合、ターゲットの総数が報告されます。 レポート条件: 常に報告される 統計: 有効な唯一の統計は Sum です。これは合計ではなく平均を表します。 ディメンション • TargetGroup • TargetGroup , AvailabilityZone • LoadBalancer , TargetGroup • LoadBalancer , AvailabilityZone , TargetGroup
TargetConnectionErrorCount	ロードバランサーとターゲット間で正常に確立されなかった接続数。ターゲットが Lambda 関数である場合、このメトリクスは適用されません。 レポート条件: ゼロ以外の値がある 統計: 最も有用な統計は Sum です。 ディメンション • LoadBalancer • AvailabilityZone , LoadBalancer • TargetGroup , LoadBalancer • TargetGroup , AvailabilityZone , LoadBalancer

メトリクス	説明
TargetResponseTime	リクエストがロードバランサーを離れてから、ターゲットがレスポンスヘッダーの送信を開始するまでの経過時間を秒単位で表します。これは、アクセスログの target_processing_time フィールドに相当します。 レポート条件: ゼロ以外の値がある 統計: 最も有用な統計は Average および pNN.NN (パーセンタイル) です。 ディメンション • LoadBalancer • AvailabilityZone , LoadBalancer • TargetGroup , LoadBalancer • TargetGroup , AvailabilityZone , LoadBalancer
TargetTLSNegotiationErrorCount	ロードバランサーにより開始され、ターゲットとのセッションを確立しなかった、TLS 接続の数。暗号化またはプロトコルの不一致が原因である場合があります。ターゲットが Lambda 関数である場合、このメトリクスは適用されません。 レポート条件: ゼロ以外の値がある 統計: 最も有用な統計は Sum です。 ディメンション • LoadBalancer • AvailabilityZone , LoadBalancer • TargetGroup , LoadBalancer • TargetGroup , AvailabilityZone , LoadBalancer

メトリクス	説明
UnHealthyHostCount	異常と見なされるターゲットの数。 レポート条件: ヘルスチェックが有効になっている場合にレポートされます 統計値: 最も有用な統計値は Average、Minimum、および Maximum です。 ディメンション • LoadBalancer , TargetGroup • LoadBalancer , AvailabilityZone , TargetGroup

AWS/ApplicationELB 名前空間には、ターゲットの正常性に関する以下のメトリクスを含んでいます。詳細については、「the section called “ターゲットグループの正常性”」を参照してください。

メトリクス	説明
HealthyStateDNS	DNS により正常状態と判断されたゾーンの数。 統計: 最も有用な統計は Min です。 ディメンション • LoadBalancer , TargetGroup • AvailabilityZone , LoadBalancer , TargetGroup
HealthyStateRouting	ルーティングにより正常状態と判断されたゾーンの数。 統計: 最も有用な統計は Min です。 ディメンション • LoadBalancer , TargetGroup • AvailabilityZone , LoadBalancer , TargetGroup

メトリクス	説明
UnhealthyRoutingRequestCount	ルーティングフェイルオーバーアクション (フェールオープン) を使用してルーティングされたリクエストの数。 統計: 最も有用な統計は Sum です。 ディメンション • LoadBalancer , TargetGroup • AvailabilityZone , LoadBalancer , TargetGroup
UnhealthyStateDNS	DNS での正常状態に関する要件を満たしていないため、DNS により障害があるとマークされたゾーンの数。 統計: 最も有用な統計は Min です。 ディメンション • LoadBalancer , TargetGroup • AvailabilityZone , LoadBalancer , TargetGroup
UnhealthyStateRouting	ルーティングの正常状態に関する要件を満たしていないゾーンの数。ロードバランサーは、このゾーン内の (障害のあるターゲットを含む) すべてのターゲットに対し、トラフィックを分散します。 統計: 最も有用な統計は Min です。 ディメンション • LoadBalancer , TargetGroup • AvailabilityZone , LoadBalancer , TargetGroup

AWS/ApplicationELB 名前空間には、ターゲットとして登録された Lambda 関数の次のメトリクスが含まれています。

メトリクス	説明
LambdaInternalError	ロードバランサーまたは AWS Lambdaの内部的な問題のために失敗した Lambda 関数へのリクエストの数。エラー理由コードを取得するには、アクセスログの error_reason フィールドを確認します。 レポート条件: ゼロ以外の値がある [Statistics] (統計): 唯一意味のある統計は Sum です。 ディメンション • TargetGroup • TargetGroup , LoadBalancer
LambdaTargetProcessedBytes	Lambda 関数へのリクエストおよび Lambda 関数からのレスポンスについて、ロードバランサーによって処理された総バイト数。 レポート条件: ゼロ以外の値がある [Statistics] (統計): 唯一意味のある統計は Sum です。 ディメンション • LoadBalancer
LambdaUserError	Lambda 関数の問題のために失敗した Lambda 関数へのリクエストの数。たとえば、ロードバランサーに関数を呼び出すアクセス権限がなかった、形式が間違っているか必須フィールドがない JSON をロードバランサーが関数から受け取った、リクエストボディまたはレスポンスのサイズが最大サイズの 1 MB を超えているなどです。エラー理由コードを取得するには、アクセスログの error_reason フィールドを確認します。 レポート条件: ゼロ以外の値がある [Statistics] (統計): 唯一意味のある統計は Sum です。 ディメンション • TargetGroup

メトリクス	説明
	• TargetGroup , LoadBalancer

AWS/ApplicationELB 名前空間には、ユーザー認証用に以下のメトリクスが含まれます。

メトリクス	説明
ELBAuthError	認証アクションが誤って設定されたため、ロードバランサーが IdP との接続を確立できなかったため、またはロードバランサーが内部エラーにより認証フローを完了できなかったため、完了できなかったユーザー認証の数。エラー理由コードを取得するには、アクセスログの error_reason フィールドを確認します。 レポート条件: ゼロ以外の値がある [Statistics] (統計): 唯一意味のある統計は Sum です。 ディメンション • LoadBalancer • AvailabilityZone , LoadBalancer
ELBAuthFailure	IdP がユーザーまたは承認コードへのアクセスを拒否したため、または承認コードが複数回使用されため、完了できなかったユーザー認証の数。エラー理由コードを取得するには、アクセスログの error_reason フィールドを確認します。 レポート条件: ゼロ以外の値がある [Statistics] (統計): 唯一意味のある統計は Sum です。 ディメンション • LoadBalancer • AvailabilityZone , LoadBalancer

メトリクス	説明
ELBAuthLatency	ID トークンとユーザー情報を IdP に照会するのにかかった時間 (ミリ秒単位)。これらのオペレーションが 1 つ以上失敗した場合は、エラーになるまでの時間です。 レポート条件: ゼロ以外の値がある [Statistics] (統計): すべての統計は意味があります。 ディメンション • LoadBalancer • AvailabilityZone , LoadBalancer
ELBAuthRefreshTokenSuccess	ロードバランサーが、IdP から提供された更新トークンを使用してユーザークレームを正常に更新した回数。 レポート条件: ゼロ以外の値がある [Statistics] (統計): 唯一意味のある統計は Sum です。 ディメンション • LoadBalancer • AvailabilityZone , LoadBalancer
ELBAuthSuccess	成功した認証アクションの数。ロードバランサーが IdP からユーザークレームを取得した後、認証ワークフローが終了すると、このメトリクスが増分されます。 レポート条件: ゼロ以外の値がある 統計: 最も有用な統計は Sum です。 ディメンション • LoadBalancer • AvailabilityZone , LoadBalancer

メトリクス	説明
ELBAuthUserClaimsSizeExceeded	設定された IdP が、11 KB を超えるサイズのユーザークレームを返した回数。 レポート条件: ゼロ以外の値がある [Statistics] (統計): 唯一意味のある統計は Sum です。 ディメンション • LoadBalancer • AvailabilityZone , LoadBalancer

Application Load Balancer のメトリクスディメンション

Application Load Balancer のメトリクスを絞り込むには、次のディメンションを使用できます。

ディメンション	説明
AvailabilityZone	アベイラビリティーゾーン別にメトリクスデータをフィルタリングします。
LoadBalancer	ロードバランサーでメトリクスデータをフィルタリングします。ロードバランサーを次のように指定します。app/ロードバランサー名/1234567890123456 (ロードバランサー ARN の最後の部分)。
TargetGroup	ターゲットグループでメトリクスデータをフィルタリングします。ターゲットグループを次のように指定します。targetgroup/ターゲットグループ名/1234567890123456 (ターゲットグループ ARN の最後の部分)。

Application Load Balancer の統計

CloudWatch は、Elastic Load Balancing によって発行されたメトリクスデータポイントに基づく統計を提供します。統計とは、メトリクスデータを指定した期間で集約したものです。統計を要求した場合、返されるデータストリームはメトリクス名とディメンションによって識別されます。ディメンションは、メトリクスを一意に識別する名前と値のペアです。たとえば、特定のアベイラビリティー

ゾーンで起動されたロードバランサーの配下のすべての正常な EC2 インスタンスの統計をリクエストできます。

Minimum および Maximum の統計は、各サンプリングウィンドウの個別のロードバランサーノードから報告されるデータポイントの最小値と最大値を反映します。例えば、Application Load Balancer を構成する 2 つのロードバランサーノードがあるとします。一方のノードは、HealthyHostCount の Minimum が 2、Maximum が 10、Average が 6 で、もう一方のノードは HealthyHostCount の Minimum が 1、Maximum が 5、Average が 3 です。このため、ロードバランサーの Minimum は 1、Maximum は 10、Average は約 4 です。

Minimum 統計のゼロ以外の UnHealthyHostCount をモニタリングし、複数のデータポイントでゼロ以外の値が発生した場合にアラームを受け取ることをお勧めします。Minimum を使用すると、ロードバランサーのすべてのノードとアベイラビリティーゾーンによってターゲットが異常であるとみなされた場合に検出されます。Average または Maximum のアラームは、潜在的な問題についてアラートを受け取りたい場合に役立ちます。このメトリクスを確認し、ゼロ以外の発生について調査することをお勧めします。Amazon EC2 Auto Scaling または Amazon Elastic Container Service (Amazon ECS) でロードバランサーのヘルスチェックを使用するベストプラクティスに従って、障害を自動的に緩和できます。

Sum 統計は、すべてのロードバランサーノードにおける集計値です。メトリクスには期間あたり複数のレポートが含まれているため、Sum はすべてのロードバランサーノードで集計されたメトリクスのみに適用されます。

SampleCount 統計は測定されたサンプルの数です。メトリクスはサンプリング間隔とイベントに基づいて集計されるため、通常、この統計は有用ではありません。例えば、HealthyHostCount の SampleCount は、正常なホストの数ではなく各ロードバランサーノードが報告するサンプル数に基づいています。

パーセンタイルは、データセットにおける値の相対的な位置を示します。小数点以下最大 2 桁を使用して、任意のパーセンタイルを指定できます (p95.45 など)。例えば、95 パーセンタイルは、95 パーセントのデータがこの値を下回っており、5 パーセントがこの値を上回っていることを意味します。パーセンタイルは、異常を分離するためによく使用されます。例えば、アプリケーションがほとんどのリクエストをキャッシュから 1 ~ 2 ミリ秒で処理するのに、キャッシュが空の場合は 100 ~ 200 ミリ秒になるとします。最大値は最も速度が遅い場合を反映し、約 200 ミリ秒になります。平均値はデータの分散を示してはいません。パーセンタイルで、アプリケーションのパフォーマンスのより有益なビューが得られます。99 パーセンタイルを Auto Scaling トリガーまたは CloudWatch アラームとして使用することで、処理に 2 ミリ秒以上かかるリクエストは 1% 以下をターゲットにできます。

ロードバランサーの CloudWatch メトリクスを表示する

Amazon EC2 コンソールを使用して、ロードバランサーの CloudWatch メトリクスを表示できます。これらのメトリクスは、モニタリング用のグラフのように表示されます。ロードバランサーがアクティブでリクエストを受信しているときにのみ、モニタリング用のグラフにデータポイントが表示されます。

または、コンソールを使用してロードバランサーの CloudWatchメトリクスを表示することもできます。

コンソールを使用してメトリクスを表示するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ターゲットグループによってフィルタリングされたメトリクスを表示するには、以下の作業を実行します。
   a. ナビゲーションペインで、[Target Groups] を選択します。
   b. ターゲットグループを選択し、[Monitoring] タブを選択します。
   c. (オプション) 結果を時間でフィルタリングするには、[Showing data for] から時間範囲を選択します。
   d. 1 つのメトリクスの大きいビューを取得するには、グラフを選択します。
3. ロードバランサーでフィルタリングされたメトリクスを表示するには、以下の操作を実行します。
   a. ナビゲーションペインで、[Load Balancers] を選択します。
   b. ロードバランサーを選択し、[Monitoring] タブを選択します。
   c. (オプション) 結果を時間でフィルタリングするには、[Showing data for] から時間範囲を選択します。
   d. 1 つのメトリクスの大きいビューを取得するには、グラフを選択します。

CloudWatch コンソールを使用してメトリクスを表示するには

1. https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで [Metrics (メトリクス)] を選択します。
3. 名前空間に [ApplicationELB] を選択します。

4. (オプション) すべてのディメンションでメトリクスを表示するには、検索フィールドに名称を入力します。
5. (オプション) ディメンション別に検索するには、次のいずれかを選択します。

   - ロードバランサーにレポートされるメトリクスのみを表示するには、[Per AppELB Metrics] を選択します。1 つのロードバランサーのメトリクスを表示するには、検索フィールドにその名前を入力します。
   - ターゲットグループにレポートされるメトリクスのみを表示するには、[Per AppELB, per TG Metrics] を選択します。1 つのターゲットグループのメトリクスを表示するには、検索フィールドにその名前を入力します。
   - アベイラビリティーゾーン別に、ロードバランサーにレポートされるメトリクスのみを表示するには、[Per AppELB, per AZ Metrics] を選択します。1 つのロードバランサーのメトリクスを表示するには、検索フィールドにその名前を入力します。1 つのアベイラビリティーゾーンのメトリクスを表示するには、検索フィールドにその名前を入力します。
   - アベイラビリティーゾーンとターゲットグループ別にロードバランサーにレポートされるメトリクスのみを表示するには、[Per AppELB, per AZ, per TG Metrics] を選択します。1 つのロードバランサーのメトリクスを表示するには、検索フィールドにその名前を入力します。1 つのターゲットグループのメトリクスを表示するには、検索フィールドにその名前を入力します。1 つのアベイラビリティーゾーンのメトリクスを表示するには、検索フィールドにその名前を入力します。

を使用してメトリクスを表示するには AWS CLI

使用可能なメトリクスを表示するには、次の list-metrics コマンドを使用します。

```
aws cloudwatch list-metrics --namespace AWS/ApplicationELB
```

を使用してメトリクスの統計を取得するには AWS CLI

次の get-metric-statistics コマンドを使用して、指定されたメトリクスと dimension. CloudWatch treats の各ディメンションの一意の組み合わせの統計を個別のメトリクスとして取得します。特に発行されていないディメンションの組み合わせを使用した統計を取得することはできません。メトリクス作成時に使用した同じディメンションを指定する必要があります。

```
aws cloudwatch get-metric-statistics --namespace AWS/ApplicationELB \
--metric-name UnHealthyHostCount --statistics Average  --period 3600 \
--dimensions Name=LoadBalancer,Value=app/my-load-balancer/50dc6c495c0c9188 \
```

```
Name=TargetGroup,Value=targetgroup/my-targets/73e2d6bc24d8a067 \
--start-time 2016-04-18T00:00:00Z --end-time 2016-04-21T00:00:00Z
```

出力例を次に示します。

```
{
    "Datapoints": [
        {
            "Timestamp": "2016-04-18T22:00:00Z",
            "Average": 0.0,
            "Unit": "Count"
        },
        {
            "Timestamp": "2016-04-18T04:00:00Z",
            "Average": 0.0,
            "Unit": "Count"
        },
        ...
    ],
    "Label": "UnHealthyHostCount"
}
```

Application Load Balancer のアクセスログ

Elastic Load Balancing は、ロードバランサーに送信されるリクエストについての詳細情報をキャプチャしたアクセスログを提供します。各ログには、リクエストを受け取った時刻、クライアントの IP アドレス、レイテンシー、リクエストのパス、サーバーレスポンスなどの情報が含まれます。これらのアクセスログを使用して、トラフィックパターンを分析し、問題のトラブルシューティングを行えます。

アクセスログは、Elastic Load Balancing のオプション機能であり、デフォルトでは無効化されています。ロードバランサーのアクセスログを有効にすると、Elastic Load Balancing はログをキャプチャし、圧縮ファイルとして指定した Amazon S3 バケット内に保存します。アクセスログはいつでも無効にできます。

Amazon S3 のストレージコストは発生しますが、Amazon S3 にログファイルを送信するために Elastic Load Balancing が使用する帯域については料金は発生しません。ストレージコストの詳細については、Amazon S3 の料金を参照してください。

目次

アクセスログファイル

Elastic Load Balancing は各ロードバランサーノードのログファイルを 5 分ごとに発行します。ログ配信には結果整合性があります。ロードバランサーでは、同じ期間について複数のログが発行されることがあります。これは通常、サイトに高トラフィックがある場合に発生します。

アクセスログのファイル名には次の形式を使用します。

```
bucket[/prefix]/AWSLogs/aws-account-id/elasticloadbalancing/region/yyyy/mm/dd/aws-account-id_elasticloadbalancing_region_app.load-balancer-id_end-time_ip-address_random-string.log.gz
```

bucket (バケット)

S3 バケットの名前。

prefix

(オプション) バケットのプレフィックス (論理階層)。指定するプレフィックスに文字列 AWSLogs を含めることはできません。詳細については、「プレフィックスを使用してオブジェクトを整理する」を参照してください。

AWSLogs

指定したバケット名とオプションのプレフィックスの後に、AWSLogs で始まるファイル名部分が追加されます。

aws-account-id

所有者の AWS アカウント ID。

region

ロードバランサーおよび S3 バケットのリージョン。

yyyy/mm/dd

ログが配信された日付。

load-balancer-id

ロードバランサーのリソース ID。リソース ID にスラッシュ (/) が含まれている場合、ピリオド (.) に置換されます。

end-time

ログ作成の間隔が終了した日時。たとえば、終了時間 20140215T2340Z には、UTC または Zulu 時間の 23:35～23:40 に行われたリクエストのエントリが含まれます。

ip-address

リクエストを処理したロードバランサーノードの IP アドレス。内部ロードバランサーの場合、プライベート IP アドレスです。

random-string

システムによって生成されたランダム文字列。

「」をプレフィックスとするログファイル名の例を次に示します。

```
s3://my-bucket/my-prefix/AWSLogs/123456789012/elasticloadbalancing/us-
east-2/2022/05/01/123456789012_elasticloadbalancing_us-east-2_app.my-
loadbalancer.1234567890abcdef_20220215T2340Z_172.160.001.192_20sg8hgm.log.gz
```

プレフィックスが付いていないログファイル名の例は次のようになります。

```
s3://my-bucket/AWSLogs/123456789012/elasticloadbalancing/us-
east-2/2022/05/01/123456789012_elasticloadbalancing_us-east-2_app.my-
loadbalancer.1234567890abcdef_20220215T2340Z_172.160.001.192_20sg8hgm.log.gz
```

必要な場合はログファイルを自身のバケットに保管できますが、ログファイルを自動的にアーカイブまたは削除するように Amazon S3 ライフサイクルルールを定義することもできます。詳細については、Amazon Simple Storage Service ユーザーガイドの「オブジェクトのライフサイクルの管理」を参照してください。

アクセスログのエントリ

Elastic Load Balancing は、ターゲットに到達しなかったリクエストを含め、ロードバランサーに送信されたリクエストを記録します。たとえば、クライアントが誤った形式のリクエストを送信した場合や、リクエストに応答できる正常に動作しているターゲットがない場合も、そのリクエストは記録されます。Elastic Load Balancing はヘルスチェックリクエストをログに記録しません。

各ログエントリには、ロードバランサーに対して行われた単一のリクエスト (または の場合は接続 WebSockets) の詳細が含まれます。の場合 WebSockets、エントリは接続が閉じられた後にのみ書き込まれます。アップグレードされた接続を確立できない場合、エントリは HTTP または HTTPS リクエストと同じです。

> **Important**
>
> Elastic Load Balancing はベストエフォートベースでリクエストを記録します アクセスログは、すべてのリクエストを完全に報告するためのものではなく、リクエストの本質を把握するものとして使用することをお勧めします。

内容

構文

次の表は、アクセスログのエントリのフィールドを順に示しています。すべてのフィールドはスペースで区切られています。新しいフィールドが導入されると、ログエントリの最後に追加されます。予期していなかったログエントリの最後のフィールドは無視する必要があります。

フィールド	説明
type	リクエストまたは接続のタイプ。有効な値は次のとおりです (その他の値は無視してください)。 • `http` — HTTP

フィールド	説明
	• `https` — TLS 経由の HTTP • `h2` — TLS 経由の HTTP/2 • `grpcs` — TLS 経由の gRPC • `ws` — WebSockets • `wss` — TLS WebSockets 経由
time	ロードバランサーがクライアントに対してレスポンスを生成した時刻 (ISO 8601 形式)。の場合 WebSockets、これは接続が閉じられた時刻です。
elb	ロードバランサーのリソース ID。アクセスログエントリを解析する場合、リソース ID にはスラッシュ (/) を含めることができます。
client:port	リクエストを送信したクライアントの IP アドレスとポート。ロードバランサーの前にプロキシがある場合、このフィールドにはプロキシの IP アドレスが含まれています。
target:port	このリクエストを処理したターゲットの IP アドレスとポート。 クライアントがリクエスト全体を送信しなかった場合、ロードバランサーはターゲットにリクエストをディスパッチできず、この値が - に設定されます。 ターゲットが Lambda 関数の場合、この値は - に設定されます。 リクエストが によってブロックされている場合 AWS WAF、この値は - に設定され、elb_status_code の値は 403 に設定されます。

フィールド	説明
request_processing_time	ロードバランサーがリクエストを受け取った時点からターゲットにリクエストを送信するまでの合計経過時間 (ミリ秒精度の秒単位)。 ロードバランサーがリクエストをターゲットにディスパッチできない場合、この値は -1 に設定されます。この状況が発生するのは、ターゲットがアイドルタイムアウト前に接続を閉じた場合か、クライアントが誤った形式のリクエストを送信した場合です。 登録済みターゲットからアイドルタイムアウトまで応答がない場合にも、この値は -1 に設定される場合があります。 Application Load Balancer で AWS WAF が有効になっているか、ターゲットタイプが Lambda 関数の場合、クライアントが POST リクエストに必要なデータを送信するのにかかる時間は にカウントされます `request_processing_time` 。
target_processing_time	ロードバランサーがターゲットにリクエストを送信した時点から、そのターゲットが応答ヘッダーの送信を開始した時点までの合計経過時間 (ミリ秒精度の秒単位)。 ロードバランサーがリクエストをターゲットにディスパッチできない場合、この値は -1 に設定されます。この状況が発生するのは、ターゲットがアイドルタイムアウト前に接続を閉じた場合か、クライアントが誤った形式のリクエストを送信した場合です。 登録済みターゲットからアイドルタイムアウトまで応答がない場合にも、この値は -1 に設定される場合があります。 AWS WAF が Application Load Balancer に対して有効になっていない場合、クライアントが POST リクエストに必要なデータを送信するのにかかる時間は にカウントされます`target_processing_time` 。

フィールド	説明
response_processing_time	ロードバランサーがターゲットから応答ヘッダーを受け取った時点から、クライアントへの応答の送信を開始した時点までの合計経過時間 (ミリ秒精度の秒単位)。これには、ロードバランサーでの待機時間と、ロードバランサーからクライアントへの接続の取得時間の両方が含まれます。 ロードバランサーがターゲットから応答を受け取らない場合、この値は -1 に設定されます。この状況が発生するのは、ターゲットがアイドルタイムアウト前に接続を閉じた場合か、クライアントが誤った形式のリクエストを送信した場合です。
elb_status_code	ロードバランサーからの応答のステータスコード。
target_status_code	ターゲットから応答のステータスコード。この値は、ターゲットへの接続が確立され、ターゲットが応答を送信した場合のみ記録されます。それ以外の場合は、- に設定されます。
received_bytes	クライアント (リクエスタ) から受け取ったリクエストのサイズ (バイト単位)。HTTP リクエストの場合、これにはヘッダーが含まれます。の場合 WebSockets、これは接続でクライアントから受信した合計バイト数です。
sent_bytes	クライアント (リクエスタ) に返される応答のサイズ (バイト単位)。HTTP リクエストの場合、これにはヘッダーが含まれます。の場合 WebSockets、これは接続でクライアントに送信された合計バイト数です。
"request"	クライアントからのリクエスト行は二重引用符で囲まれており、次の形式を使用してログに記録されます。HTTP メソッド + プロトコル://ホスト:ポート/uri + HTTP バージョン。ロードバランサーは、リクエスト URI を記録するときに、クライアントから送信された URL をそのまま保持します。アクセスログファイルのコンテンツタイプは設定されません。このフィールドを処理するときは、クライアントが URL を送信した方法を考慮してください。

フィールド	説明
"user_agent"	リクエスト元のクライアントを特定する User-Agent 文字列 (二重引用符で囲まれます)。この文字列は、1 つ以上の製品 ID (製品[/バージョン]) から構成されます。文字列が 8 KB より長い場合は切り捨てられます。
ssl_cipher	[HTTPS リスナー] SSL 暗号。リスナーが HTTPS リスナーではない場合、この値は - に設定されます。
ssl_protocol	[HTTPS リスナー] SSL プロトコル。リスナーが HTTPS リスナーではない場合、この値は - に設定されます。
target_group_arn	ターゲットグループの Amazon リソースネーム (ARN)。
"trace_id"	X-Amzn-Trace-Id ヘッダーのコンテンツ (二重引用符で囲まれます)。
"domain_name"	[HTTPS リスナー] TLS ハンドシェイク中にクライアントから提供される SNI ドメイン (二重引用符で囲まれます)。クライアントが SNI をサポートしない場合、あるいはドメインが証明書と一致せず、デフォルトの証明書がクライアントに提示された場合、この値は - となります。
"chosen_cert_arn"	[HTTPS リスナー] クライアントに提示される証明書の ARN (二重引用符で囲まれます)。セッションが再利用される場合、この値は `session-reused` に設定されます。リスナーが HTTPS リスナーではない場合、この値は - に設定されます。
matched_rule_priority	リクエストに一致したルールの優先度の値。ルールが一致した場合、この値は 1～50,000 になります。一致するルールがなく、デフォルトのアクションが実行された場合、この値は 0 に設定されます。ルールの評価中にエラーが発生した場合は、-1 に設定されます。その他のエラーの場合は、- に設定されます。
request_creation_time	ロードバランサーがクライアントからリクエストを受け取った時刻 (ISO 8601 形式)。
"actions_executed"	リクエストの処理時に実行されるアクション (二重引用符で囲まれます)。この値は、実行されるアクション で説明されている値を含めることができるカンマ区切りリストです。形式が正しくないリクエストなどでアクションが実行されない場合、この値は - に設定されます。

フィールド	説明
"redirect_url"	HTTP レスポンスの Location ヘッダーのリダイレクトターゲットの URL (二重引用文字で囲む)。リダイレクトアクションが実行されなかった場合、この値は - に設定されます。
"error_reason"	エラー理由コード (二重引用符で囲まれます)。リクエストが失敗した場合、これは エラー理由コード で説明されているいずれかのエラーコードになります。実行されたアクションが認証アクションを含まない、またはターゲットが Lambda 関数ではない場合、この値は - に設定されます。
"target:port_list"	このリクエストを処理したターゲットの IP アドレスとポートのスペース区切りのリスト (二重引用符で囲まれます)。現在、このリストには 1 つの項目を含めることができ、target:port フィールドと一致します。 クライアントがリクエスト全体を送信しなかった場合、ロードバランサーはターゲットにリクエストをディスパッチできず、この値が - に設定されます。 ターゲットが Lambda 関数の場合、この値は - に設定されます。 リクエストが によってブロックされている場合 AWS WAF、この値は - に設定され、elb_status_code の値は 403 に設定されます。
"target_status_code_list"	ターゲットの応答からのステータスコードのスペース区切りのリスト (二重引用符で囲まれます)。現在、このリストには 1 つの項目を含めることができ、target_status_code フィールドと一致します。 この値は、ターゲットへの接続が確立され、ターゲットが応答を送信した場合のみ記録されます。それ以外の場合は、- に設定されます。
"classification"	desync 緩和の分類 (二重引用符で囲まれます)。リクエストが RFC 7230 に準拠していない場合、可能な値は Acceptable、Ambiguous、および Severe です。 リクエストが RFC 7230 に準拠している場合、この値は - に設定されます。

フィールド	説明
"classification_reason"	分類理由コード (二重引用符で囲まれます)。リクエストが RFC 7230 に準拠していない場合、これは 分類の理由 で説明されている分類コードの 1 つです。リクエストが RFC 7230 に準拠している場合、この値は - に設定されます。
conn_trace_id	接続のトレーサビリティ ID は、各接続を識別するために使用される一意の不透明な ID です。クライアントとの接続が確立されると、このクライアントからの後続のリクエストには、それぞれのアクセスログエントリにこの ID が含まれます。この ID は、接続ログとアクセスログ間のリンクを作成するための外部キーとして機能します。

実行されるアクション

ロードバランサーは、実行されたアクションをアクセスログの actions_executed フィールドに格納します。

- `authenticate` — ロードバランサーは、ルール設定で指定されているように、セッションを検証し、ユーザーを認証し、ユーザー情報をリクエストヘッダーに追加しました。
- `fixed-response` — ロードバランサーは、ルール設定で指定された固定レスポンスを発行しました。
- `forward` — ロードバランサーは、ルール設定で指定されているように、リクエストをターゲットに転送しました。
- `redirect` — ロードバランサーは、ルール設定で指定されているように、リクエストを別の URL にリダイレクトしました。
- `waf` — ロードバランサーは、リクエストをターゲットに転送する必要があるかどうかを判別するために、リクエストを AWS WAF に転送しました。これが最後のアクションである場合、 はリクエストを拒否する必要がある AWS WAF と判断しました。
- `waf-failed` — ロードバランサーはリクエストを に転送しようとしましたが AWS WAF、このプロセスは失敗しました。

分類の理由

リクエストが RFC 7230 に準拠していない場合、ロードバランサーはアクセスログの classification_reason フィールドに次のいずれかのコードを保存します。詳細については、「Desync 軽減モード」を参照してください。

コード	説明	分類
AmbiguousUri	リクエスト URI に制御文字が含まれています。	Ambiguous
BadContentLength	Content-Length ヘッダーに解析できない値または無効な数値が含まれています。	Severe
BadHeader	ヘッダーに NULL 文字またはキャリッジリターンが含まれています。	Severe
BadTransferEncoding	Transfer-Encoding ヘッダーに不正な値が含まれています。	Severe
BadUri	リクエスト URI に NULL 文字またはキャリッジリターンが含まれています。	Severe
BadMethod	リクエストメソッドの形式が正しくありません。	Severe
BadVersion	リクエストバージョンの形式が正しくありません。	Severe
BothTeClPresent	リクエストに Transfer-Encoding ヘッダーと Content-Length ヘッダーの両方が含まれています。	Ambiguous
DuplicateContentLength	同じ値を持つ複数の Content-Length ヘッダーがあります。	Ambiguous
EmptyHeader	ヘッダーが空であるか、スペースのみを含む行があります。	Ambiguous

コード	説明	分類
`GetHeadZeroContentLength`	GET または HEAD リクエストの値を 0 とする Content-Length ヘッダーがあります。	Acceptable
`MultipleContentLength`	異なる値を持つ複数の Content-Length ヘッダーがあります。	Severe
`MultipleTransferEncodingChunked`	複数の Transfer-Encoding: chunked ヘッダーがあります。	Severe
`NonCompliantHeader`	ヘッダーに ASCII 以外の文字または制御文字が含まれています。	Acceptable
`NonCompliantVersion`	リクエストバージョンに不正な値が含まれています。	Acceptable
`SpaceInUri`	リクエスト URI に URL エンコードされていないスペースが含まれています。	Acceptable
`SuspiciousHeader`	一般的なテキスト正規化技術を使用して、Transfer-Encoding または Content-Length に正規化できるヘッダーがあります。	Ambiguous
`UndefinedContentLengthSemantics`	GET または HEAD リクエスト用に定義された Content-Length ヘッダーがあります。	Ambiguous
`UndefinedTransferEncodingSemantics`	GET または HEAD リクエスト用に定義された Transfer-Encoding ヘッダーがあります。	Ambiguous

エラー理由コード

ロードバランサーが認証アクションを完了できない場合、ロードバランサーはアクセスログの error_reason フィールドに次のいずれかの理由コードを保存します。ロードバランサーは、対応する CloudWatch メトリクスもインクリメントします。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。

コード	説明	メトリクス
AuthInvalidCookie	認証 Cookie が無効です。	ELBAuthFailure
AuthInvalidGrantError	トークンエンドポイントからの認証付与コードが無効です。	ELBAuthFailure
AuthInvalidIdToken	ID トークンが無効です。	ELBAuthFailure
AuthInvalidStateParam	状態パラメータが無効です。	ELBAuthFailure
AuthInvalidTokenResponse	トークンエンドポイントからのレスポンスが無効です。	ELBAuthFailure
AuthInvalidUserinfoResponse	ユーザー情報エンドポイントからのレスポンスが無効です。	ELBAuthFailure
AuthMissingCodeParam	認証エンドポイントからの認証レスポンスに、「code」という名前のクエリパラメータがありません。	ELBAuthFailure
AuthMissingHostHeader	認証エンドポイントからの認証レスポンスに、ホストヘッダーフィールドがありません。	ELBAuthError
AuthMissingStateParam	認証エンドポイントからの認証レスポンスに、「state」という名前のクエリパラメータがありません。	ELBAuthFailure

コード	説明	メトリクス
AuthTokenEpRequestFailed	トークンエンドポイントからエラーレスポンス (2XX 以外) があります。	ELBAuthError
AuthTokenEpRequestTimeout	ロードバランサーは、トークンエンドポイントと通信できません。	ELBAuthError
AuthUnhandledException	ロードバランサーで処理されない例外が発生しました。	ELBAuthError
AuthUserinfoEpRequestFailed	IdP ユーザー情報エンドポイントからエラーレスポンス (2XX 以外) があります。	ELBAuthError
AuthUserinfoEpRequestTimeout	ロードバランサーは、IdP ユーザー情報エンドポイントと通信できません。	ELBAuthError
AuthUserinfoResponseSizeExceeded	IdP から返されたクレームのサイズが 11K バイトを超えました。	ELBAuthUserClaimsSizeExceeded

加重ターゲットグループへのリクエストが失敗した場合、ロードバランサーはアクセスログの error_reason フィールドに次のいずれかのエラーコードを保存します。

コード	説明
AWSALBTGCookieInvalid	加重ターゲットグループで使用される AWSALBTG Cookie は無効です。たとえば、Cookie 値が URL エンコードされている場合、ロードバランサーはこのエラーを返します。
WeightedTargetGroupsUnhandledException	ロードバランサーで処理されない例外が発生しました。

Lambda 関数へのリクエストが失敗した場合、ロードバランサーはアクセスログの error_reason フィールドに次のいずれかの理由コードを保存します。ロードバランサーは、対応する CloudWatch メトリクスもインクリメントします。詳細については、Lambda 呼び出しアクションを参照してください。

コード	説明	メトリクス
LambdaAccessDenied	ロードバランサーに、Lambda 関数を呼び出すアクセス権限がありませんでした。	LambdaUserError
LambdaBadRequest	クライアントリクエストヘッダーまたは本文に UTF-8 文字のみが含まれていないため、Lambda 呼び出しが失敗しました。	LambdaUserError
LambdaConnectionError	ロードバランサーが Lambda に接続できません。	LambdaInternalError
LambdaConnectionTimeout	Lambda への接続がタイムアウトしました。	LambdaInternalError
LambdaEC2AccessDeniedException	関数の初期化中に Amazon EC2 が Lambda へのアクセスを拒否しました。	LambdaUserError
LambdaEC2ThrottledException	関数の初期化中に Amazon EC2 が Lambda を調整しました。	LambdaUserError
LambdaEC2UnexpectedException	関数の初期化中に Amazon EC2 で予期しない例外が発生しました。	LambdaUserError
LambdaENILimitReachedException	ネットワークインターフェイスの制限を超えたため、Lambda は Lambda 関数の設定で指定された VPC のネットワークインターフェイスを作成できませんでした。	LambdaUserError

コード	説明	メトリクス
LambdaInvalidResponse	Lambda 関数からのレスポンスの形式が間違っているか、必須フィールドが含まれていません。	LambdaUserError
LambdaInvalidRuntimeException	指定されたバージョンの Lambda ランタイムはサポートされていません。	LambdaUserError
LambdaInvalidSecurityGroupIDException	Lambda 関数の設定で指定されたセキュリティグループ ID が無効です。	LambdaUserError
LambdaInvalidSubnetIDException	Lambda 関数の設定で指定されたサブネット ID が無効です。	LambdaUserError
LambdaInvalidZipFileException	Lambda は指定された関数の zip ファイルを解凍できませんでした。	LambdaUserError
LambdaKMSAccessDeniedException	KMS キーへのアクセスが拒否されたため、Lambda は環境変数を復号できませんでした。Lambda 関数の KMS アクセス権限を確認してください。	LambdaUserError
LambdaKMSDisabledException	指定された KMS キーが無効化されたため、Lambda は環境変数を復号できませんでした。Lambda 関数の KMS キー設定を確認してください。	LambdaUserError
LambdaKMSInvalidStateException	KMS キーの状態が無効であるため、Lambda は環境変数を復号できませんでした。Lambda 関数の KMS キー設定を確認してください。	LambdaUserError

コード	説明	メトリクス
LambdaKMSNotFoundException	KMS キーが見つからなかったため、Lambda は環境変数を復号できませんでした。Lambda 関数の KMS キー設定を確認してください。	LambdaUserError
LambdaRequestTooLarge	リクエストボディのサイズは 1 MB を超えています。	LambdaUserError
LambdaResourceNotFound	Lambda 関数は見つかりませんでした。	LambdaUserError
LambdaResponseTooLarge	レスポンスのサイズは 1 MB を超えています。	LambdaUserError
LambdaServiceException	Lambda 内部エラーが発生しました。	LambdaInternalError
LambdaSubnetIPAddressLimitReachedException	Lambda は、1 つ以上のサブネットに使用可能な IP アドレスがないため、Lambda 関数の VPC アクセスを設定できませんでした。	LambdaUserError
LambdaThrottling	リクエストが多すぎるため、Lambda 関数が調整されました。	LambdaUserError
LambdaUnhandled	Lambda 関数で処理されない例外が発生しました。	LambdaUserError
LambdaUnhandledException	ロードバランサーで処理されない例外が発生しました。	LambdaInternalError
LambdaWebsocketNotSupported	WebSockets は Lambda ではサポートされていません。	LambdaUserError

ロードバランサーは、 へのリクエストの転送時にエラーが発生した場合 AWS WAF、次のいずれかのエラーコードをアクセスログの error_reason フィールドに保存します。

Code	説明
WAFConnectionError	ロードバランサーは に接続できません AWS WAF。
WAFConnectionTimeout	への接続がタイムアウト AWS WAF しました。
WAFResponseReadTimeout	へのリクエストがタイムアウト AWS WAF しました。
WAFServiceError	AWS WAF は 5XX エラーを返しました。
WAFUnhandledException	ロードバランサーで処理されない例外が発生しました。

ログエントリの例

以下にログエントリの例を示します。読みやすくするためだけの目的で、テキストは複数の行に表示されています。

HTTP エントリ例

次の例は、HTTP リスナーのログエントリです (ポート 80 からポート 80)。

```
http 2018-07-02T22:23:00.186641Z app/my-loadbalancer/50dc6c495c0c9188
192.168.131.39:2817 10.0.0.1:80 0.000 0.001 0.000 200 200 34 366
"GET http://www.example.com:80/ HTTP/1.1" "curl/7.46.0" - -
arn:aws:elasticloadbalancing:us-east-2:123456789012:targetgroup/my-
targets/73e2d6bc24d8a067
"Root=1-58337262-36d228ad5d99923122bbe354" "-" "-"
0 2018-07-02T22:22:48.364000Z "forward" "-" "-" "10.0.0.1:80" "200" "-" "-"
```

HTTPS エントリ例

次の例は、HTTPS リスナーのログエントリです (ポート 443 からポート 80)。

```
https 2018-07-02T22:23:00.186641Z app/my-loadbalancer/50dc6c495c0c9188
```

```
192.168.131.39:2817 10.0.0.1:80 0.086 0.048 0.037 200 200 0 57
"GET https://www.example.com:443/ HTTP/1.1" "curl/7.46.0" ECDHE-RSA-AES128-GCM-SHA256
 TLSv1.2
arn:aws:elasticloadbalancing:us-east-2:123456789012:targetgroup/my-
targets/73e2d6bc24d8a067
"Root=1-58337281-1d84f3d73c47ec4e58577259" "www.example.com" "arn:aws:acm:us-
east-2:123456789012:certificate/12345678-1234-1234-1234-123456789012"
1 2018-07-02T22:22:48.364000Z "authenticate,forward" "-" "-" "10.0.0.1:80" "200" "-"
 "-" TID_123456
```

HTTP/2 エントリ例

HTTP/2 ストリームのログエントリの例を次に示します。

```
h2 2018-07-02T22:23:00.186641Z app/my-loadbalancer/50dc6c495c0c9188
10.0.1.252:48160 10.0.0.66:9000 0.000 0.002 0.000 200 200 5 257
"GET https://10.0.2.105:773/ HTTP/2.0" "curl/7.46.0" ECDHE-RSA-AES128-GCM-SHA256
 TLSv1.2
arn:aws:elasticloadbalancing:us-east-2:123456789012:targetgroup/my-
targets/73e2d6bc24d8a067
"Root=1-58337327-72bd00b0343d75b906739c42" "-" "-"
1 2018-07-02T22:22:48.364000Z "redirect" "https://example.com:80/" "-" "10.0.0.66:9000"
 "200" "-" "-"
```

WebSockets エントリの例

WebSockets 接続のログエントリの例を次に示します。

```
ws 2018-07-02T22:23:00.186641Z app/my-loadbalancer/50dc6c495c0c9188
10.0.0.140:40914 10.0.1.192:8010 0.001 0.003 0.000 101 101 218 587
"GET http://10.0.0.30:80/ HTTP/1.1" "-" - -
arn:aws:elasticloadbalancing:us-east-2:123456789012:targetgroup/my-
targets/73e2d6bc24d8a067
"Root=1-58337364-23a8c76965a2ef7629b185e3" "-" "-"
1 2018-07-02T22:22:48.364000Z "forward" "-" "-" "10.0.1.192:8010" "101" "-" "-"
```

セキュア WebSockets エントリの例

以下は、セキュリティで保護された WebSockets接続のログエントリの例です。

```
wss 2018-07-02T22:23:00.186641Z app/my-loadbalancer/50dc6c495c0c9188
```

```
10.0.0.140:44244 10.0.0.171:8010 0.000 0.001 0.000 101 101 218 786
"GET https://10.0.0.30:443/ HTTP/1.1" "-" ECDHE-RSA-AES128-GCM-SHA256 TLSv1.2
arn:aws:elasticloadbalancing:us-west-2:123456789012:targetgroup/my-
targets/73e2d6bc24d8a067
"Root=1-58337364-23a8c76965a2ef7629b185e3" "-" "-"
1 2018-07-02T22:22:48.364000Z "forward" "-" "-" "10.0.0.171:8010" "101" "-" "-"
```

Lambda の関数のエントリの例。

成功した Lambda 関数へのリクエストのログエントリ例を次に示します。

```
http 2018-11-30T22:23:00.186641Z app/my-loadbalancer/50dc6c495c0c9188
192.168.131.39:2817 - 0.000 0.001 0.000 200 200 34 366
"GET http://www.example.com:80/ HTTP/1.1" "curl/7.46.0" - -
arn:aws:elasticloadbalancing:us-east-2:123456789012:targetgroup/my-
targets/73e2d6bc24d8a067
"Root=1-58337364-23a8c76965a2ef7629b185e3" "-" "-"
0 2018-11-30T22:22:48.364000Z "forward" "-" "-" "-" "-" "-" "-"
```

失敗した Lambda 関数へのリクエストのログエントリ例を次に示します。

```
http 2018-11-30T22:23:00.186641Z app/my-loadbalancer/50dc6c495c0c9188
192.168.131.39:2817 - 0.000 0.001 0.000 502 - 34 366
"GET http://www.example.com:80/ HTTP/1.1" "curl/7.46.0" - -
arn:aws:elasticloadbalancing:us-east-2:123456789012:targetgroup/my-
targets/73e2d6bc24d8a067
"Root=1-58337364-23a8c76965a2ef7629b185e3" "-" "-"
0 2018-11-30T22:22:48.364000Z "forward" "-" "LambdaInvalidResponse" "-" "-" "-" "-"
```

アクセスログファイルの処理

アクセスログファイルは圧縮されます。Amazon S3 コンソールを使用してファイルを開くと、ファイルは解凍され、情報が表示されます。ファイルをダウンロードする場合、情報を表示するには解凍する必要があります。

ウェブサイトの需要が大きい場合は、ロードバランサーによって数 GB のデータ量のログファイルが生成されることがあります。処理を使用して、このような大量のデータを line-by-line 処理できない場合があります。このため、場合によっては、並列処理ソリューションを提供する分析ツールを使用する必要があります。例えば、次の分析ツールを使用するとアクセスログの分析と処理を行うことができます。

- Amazon Athena はインタラクティブなクエリサービスで、Amazon S3 内のデータを標準 SQL を使用して簡単に分析できるようになります。詳細については、Amazon Athena ユーザーガイドの Querying Application Load Balancer logs を参照してください。
- Loggly
- Splunk
- Sumo logic

Application Load Balancer のアクセスログを有効にする

ロードバランサーのアクセスログを有効にする場合は、ロードバランサーがログを保存する S3 バケットの名前を指定する必要があります。このバケットは、バケットにアクセスログを書き込む許可を Elastic Load Balancing に付与するバケットポリシーが必要です。

タスク

- ステップ 1: S3 バケットを作成する
- ステップ 2: S3 バケットにポリシーをアタッチする
- ステップ 3: アクセスログを設定する
- ステップ 4: バケット許可を確認
- トラブルシューティング

ステップ 1: S3 バケットを作成する

アクセスログを有効にするときは、アクセスログの S3 バケットを指定する必要があります。既存のバケットを使用するか、アクセスログ専用のバケットを作成できます。バケットは、次の要件を満たしている必要があります。

要件

- バケットは、ロードバランサーと同じリージョンに配置されている必要があります。バケットとロードバランサーは、異なるアカウントにより所有できます。
- サポートされている唯一のサーバー側の暗号化オプションは、Amazon S3 マネージドキー (SSE-S3) です。詳細については、「 Amazon S3 マネージド暗号化キー (SSE-S3)」を参照してください。

Amazon S3 コンソールを使用して S3 バケットを作成するには

1. Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [バケットを作成] を選択します。
3. [バケットを作成] ページで、次の操作を実行します。

    a. [バケット名] にバケットの名前を入力します。この名前は、Amazon S3 内で既存の、すべてのバケット名の中で一意である必要があります。リージョンによっては、バケット名にその他の制限が設けられていることがあります。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「バケットの制約と制限」を参照してください。
    b. [AWS リージョン] で、ロードバランサーを作成したリージョンを選択します。
    c. [デフォルトの暗号化] には、[Amazon S3 マネージドキー (SSE-S3)] を選択します。
    d. [バケットを作成] を選択します。

ステップ 2: S3 バケットにポリシーをアタッチする

S3 バケットには、バケットにアクセスログを書き込む許可を Elastic Load Balancing に付与するバケットポリシーが必要です。バケットポリシーは、バケットのアクセス許可を定義するためにアクセスポリシー言語で記述された JSON ステートメントのコレクションです。各ステートメントには 1 つのアクセス許可に関する情報が含まれ、一連のエレメントが使用されます。

既にポリシーがアタッチされている既存のバケットを使用している場合は、Elastic Load Balancing アクセスログのステートメントをポリシーに追加できます。この場合、結果として作成されるアクセス権限のセットが、アクセスログのバケットへのアクセスを必要とするユーザーに対して適切であることを確認するために、このセットを評価することをお勧めします。

使用可能なバケットポリシー

使用するバケットポリシーは、 AWS リージョン とゾーンのタイプによって異なります。

2022 年 8 月以降に利用可能になったリージョン

このポリシーは、指定されたログ配信サービスに許可を付与します。このポリシーは、以下のリージョンのアベイラビリティーゾーンと Local Zones にあるロードバランサーに使用してください。

- アジアパシフィック (ハイデラバード)
- アジアパシフィック (メルボルン)
- カナダ西部 (カルガリー)

- 欧州 (スペイン)
- 欧州 (チューリッヒ)
- イスラエル (テルアビブ)
- 中東 (アラブ首長国連邦)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "logdelivery.elasticloadbalancing.amazonaws.com"
      },
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::bucket-name/prefix/AWSLogs/aws-account-id/*"
    }
  ]
}
```

2022 年 8 月より前に利用可能になったリージョン

このポリシーは、指定された Elastic Load Balancing アカウント ID に許可を付与します。このポリシーは、以下のリストに記載されている リージョンのアベイラビリティーゾーンまたは Local Zones にあるロードバランサーに使用してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::elb-account-id:root"
      },
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::bucket-name/prefix/AWSLogs/aws-account-id/*"
    }
  ]
}
```

elb-account-id を、リージョンの Elastic Load Balancing AWS アカウント の の ID に置き換えます。

- 米国東部 (バージニア北部) – 127311923021
- 米国東部 (オハイオ) — 033677994240
- 米国西部 (北カリフォルニア) – 027434742980
- 米国西部 (オレゴン) — 797873946194
- アフリカ (ケープタウン) - 098369216593
- アジアパシフィック (香港) - 754344448648
- アジアパシフィック (ジャカルタ) – 589379963580
- アジアパシフィック (ムンバイ) – 718504428378
- アジアパシフィック (大阪) – 383597477331
- アジアパシフィック (ソウル) – 600734575887
- アジアパシフィック (シンガポール) – 114774131450
- アジアパシフィック (シドニー) — 783225319266
- アジアパシフィック (東京) — 582318560864
- カナダ (中部) – 985666609251
- 欧州 (フランクフルト) – 054676820928
- 欧州 (アイルランド) – 156460612806
- 欧州 (ロンドン) – 652711504416
- ヨーロッパ (ミラノ) - 635631232127
- 欧州 (パリ) – 009996457667
- 欧州 (ストックホルム) – 897822967062
- 中東 (バーレーン) — 076674570225
- 南米 (サンパウロ) – 507241528517

my-s3-arn はアクセスログを保存する場所の ARN に置き換えてください。指定する ARN は、ステップ 3 でアクセスログを有効にするときにプレフィックスを指定するかどうかによって異なります。

- プレフィックス付きの ARN の例

```
arn:aws:s3:::bucket-name/prefix/AWSLogs/aws-account-id/*
```

- プレフィックスなしの ARN の例

```
arn:aws:s3:::bucket-name/AWSLogs/aws-account-id/*
```

Effect が NotPrincipalの場合、 を使用しますDeny。

Amazon S3 バケットポリシーが を 値Effectとともに使用しDeny、
以下の例NotPrincipalに示すように が含まれている場合は、
logdelivery.elasticloadbalancing.amazonaws.comがServiceリストに含まれていることを確認してください。

```
{
  "Effect": "Deny",
  "NotPrincipal": {
    "Service": [
       "logdelivery.elasticloadbalancing.amazonaws.com",
       "example.com"
    ]
  }
},
```

AWS GovCloud (US) Regions

このポリシーは、指定された Elastic Load Balancing アカウント ID に許可を付与します。このポリシーは、以下のリストの AWS GovCloud (US) リージョンのアベイラビリティーゾーンまたはローカルゾーンのロードバランサーに使用します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws-us-gov:iam::elb-account-id:root"
      },
      "Action": "s3:PutObject",
      "Resource": "my-s3-arn"
```

```
    }
  ]
}
```

elb-account-id を、 AWS GovCloud (US) リージョンの Elastic Load Balancing AWS アカウントの の ID に置き換えます。

- AWS GovCloud （ 米国西部) – 048591011584
- AWS GovCloud （ 米国東部) – 190560391635

my-s3-arn はアクセスログを保存する場所の ARN に置き換えてください。指定する ARN は、ステップ 3 でアクセスログを有効にするときにプレフィックスを指定するかどうかによって異なります。

- プレフィックス付きの ARN の例

```
arn:aws-us-gov:s3:::bucket-name/prefix/AWSLogs/aws-account-id/*
```

- プレフィックスなしの ARN の例

```
arn:aws-us-gov:s3:::bucket-name/AWSLogs/aws-account-id/*
```

Outposts ゾーン

次のポリシーを使用して、指定されたログ配信サービスに許可を付与します。Outposts ゾーンのロードバランサーにはこのポリシーを使用します。

```
{
    "Effect": "Allow",
    "Principal": {
        "Service": "logdelivery.elb.amazonaws.com"
    },
    "Action": "s3:PutObject",
    "Resource": "arn:aws:s3:::bucket-name/prefix/AWSLogs/your-aws-account-id/*",
    "Condition": {
        "StringEquals": {
            "s3:x-amz-acl": "bucket-owner-full-control"
        }
    }
```

```
}
```

Amazon S3 を使用してアクセスログのバケットポリシーをバケットにアタッチする

1. https://console.aws.amazon.com/s3/でAmazon S3 コンソールを開きます。
2. バケットの名前を選択して、その詳細ページを開きます。
3. [許可] を選択してから、[バケットポリシー]、[編集] の順に選択します。
4. 必要な許可を付与するようにバケットポリシーを更新します。
5. [変更の保存] をクリックします。

ステップ 3: アクセスログを設定する

以下の手順を使用して、S3 バケットのログファイルを、収集して配信するように、アクセスログを設定します。

要件

バケットはステップ 1 で説明した要件を満たしている必要があり、ステップ 2 で説明したようにバケットポリシーをアタッチする必要があります。プレフィックスを指定する場合は、文字列「」を含めないでくださいAWSLogs。

コンソールを使用したロードバランサーのアクセスログの有効化

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[ロードバランサー] を選択します。
3. ロードバランサーの名前を選択して、その詳細ページを開きます。
4. [属性] タブで、[編集] を選択します。
5. [モニタリング] で [アクセスログ] をオンにします。
6. [S3 ロケーション] には、ログファイルの S3 URI を入力します。指定する URI は、プレフィックスを使用しているかどうかによって異なります。
   - プレフィックスが付いた URI: `s3://bucket-name/prefix`
   - プレフィックスなしの URI: `s3://bucket-name`
7. [変更の保存] をクリックします。

を使用してアクセスログを有効にするには AWS CLI

modify-load-balancer-attributes コマンドを使用します。

アクセスログの S3 バケットを管理するには

アクセスログ用に設定したバケットを削除する前に、必ずアクセスログを無効にします。これを行わないと、自分が所有していない AWS アカウント に、同じ名前で必要なバケットポリシーを持つ新しいバケットが作成された場合、Elastic Load Balancing がロードバランサーのアクセスログを、その新しいバケットに書き込んでしまう可能性があります。

ステップ 4: バケット許可を確認

アクセスログをロードバランサーで有効にすると、Elastic Load Balancing は S3 バケットを検証し、テストファイルを作成して、バケットポリシーが必要なアクセス権限を指定するようにします。Amazon S3 コンソールを使用して、テストファイルが作成されたことを確認できます。テストファイルは実際のアクセスログファイルではなく、レコード例は含まれていません。

Elastic Load Balancing が S3 バケットにテストファイルを作成したことを確認するには

1. Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. アクセスログ用に指定したバケットの名前を選択します。
3. テストファイル「ELBAccessLogTestFile」に移動します。場所は、プレフィックスを使用しているかどうかによって異なります。

   - プレフィックスが付いている場合の場所: *my-bucket*/*prefix*/AWSLogs/*123456789012*/ELBAccessLogTestFile
   - プレフィックスが付いていない場合の場所: *my-bucket*/AWSLogs/*123456789012*/ELBAccessLogTestFile

トラブルシューティング

アクセス拒否エラーが表示される場合は、以下の原因が考えられます。

- バケットが、バケットにアクセスログを書き込む許可を Elastic Load Balancing に付与しない。そのリージョンに対して正しいバケットポリシーを使用していることを確認してください。リソース ARN で、アクセスログを有効にしたときに指定したのと同じバケット名が使用されていることを確認します。アクセスログを有効にしたときにプレフィックスを指定しなかった場合は、リソース ARN にプレフィックスが含まれていないことを確認してください。
- バケットが、サポートされていないサーバー側の暗号化オプションを使用している。バケットは、Amazon S3 マネージドキー (SSE-S3) を使用する必要があります。

Application Load Balancer のアクセスログを無効にする

ロードバランサーのアクセスログは、いつでも無効にできます。アクセスログを無効にした後は、削除するまでアクセスログは S3 バケットに残されたままになります。詳細については、Amazon Simple Storage Service ユーザーガイドでバケットの使用について参照してください。

コンソールを使用してアクセスログを無効にするには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[ロードバランサー] を選択します。
3. ロードバランサーの名前を選択して、その詳細ページを開きます。
4. [属性] タブで、[編集] を選択します。
5. [モニタリング] で [アクセスログ] をオフにします。
6. [変更の保存] をクリックします。

を使用してアクセスログを無効にするには AWS CLI

modify-load-balancer-attributes コマンドを使用します。

Application Load Balancer の接続ログ

Elastic Load Balancing は、ロードバランサーに送信されたリクエストに関する詳細情報をキャプチャする接続ログを提供します。各ログには、クライアントの IP アドレスとポート、リスナーポート、使用する TLS 暗号とプロトコル、TLS ハンドシェイクレイテンシー、接続ステータス、クライアント証明書の詳細などの情報が含まれます。これらの接続ログを使用して、リクエストパターンを分析し、問題をトラブルシューティングできます。

接続ログは、デフォルトで無効になっている Elastic Load Balancing のオプション機能です。ロードバランサーの接続ログを有効にすると、Elastic Load Balancing はログをキャプチャし、指定した Amazon S3 バケットに圧縮ファイルとして保存します。接続ログはいつでも無効にできます。

Amazon S3 のストレージコストは発生しますが、Amazon S3 にログファイルを送信するために Elastic Load Balancing が使用する帯域については料金は発生しません。ストレージコストの詳細については、Amazon S3 の料金を参照してください。

内容

接続ログファイル

Elastic Load Balancing は各ロードバランサーノードのログファイルを 5 分ごとに発行します。ログ配信には結果整合性があります。ロードバランサーでは、同じ期間について複数のログが発行されることがあります。これは通常、サイトに高トラフィックがある場合に発生します。

接続ログのファイル名は、次の形式を使用します。

```
bucket[/prefix]/AWSLogs/aws-account-id/elasticloadbalancing/region/yyyy/mm/dd/
conn_log.aws-account-id_elasticloadbalancing_region_app.load-balancer-id_end-time_ip-
address_random-string.log.gz
```

bucket (バケット)

S3 バケットの名前。

prefix

(オプション) バケットのプレフィックス (論理階層)。指定するプレフィックスに文字列 AWSLogs を含めることはできません。詳細については、「プレフィックスを使用してオブジェクトを整理する」を参照してください。

AWSLogs

指定したバケット名とオプションのプレフィックスの後に、AWSLogs で始まるファイル名部分が追加されます。

aws-account-id

所有者の AWS アカウント ID。

region

ロードバランサーおよび S3 バケットのリージョン。

yyyy/mm/dd

ログが配信された日付。

load-balancer-id

ロードバランサーのリソース ID。リソース ID にスラッシュ (/) が含まれている場合、ピリオド (.) に置換されます。

end-time

ログ作成の間隔が終了した日時。たとえば、終了時間 20140215T2340Z には、UTC または Zulu 時間の 23:35～23:40 に行われたリクエストのエントリが含まれます。

ip-address

リクエストを処理したロードバランサーノードの IP アドレス。内部ロードバランサーの場合、プライベート IP アドレスです。

random-string

システムによって生成されたランダム文字列。

「」をプレフィックスとするログファイル名の例を次に示します。

```
s3://my-bucket/my-prefix/AWSLogs/123456789012/elasticloadbalancing/us-
east-2/2022/05/01/conn_log.123456789012_elasticloadbalancing_us-east-2_app.my-
loadbalancer.1234567890abcdef_20220215T2340Z_172.160.001.192_20sg8hgm.log.gz
```

プレフィックスが付いていないログファイル名の例は次のようになります。

```
s3://my-bucket/AWSLogs/123456789012/elasticloadbalancing/us-
east-2/2022/05/01/conn_log.123456789012_elasticloadbalancing_us-east-2_app.my-
loadbalancer.1234567890abcdef_20220215T2340Z_172.160.001.192_20sg8hgm.log.gz
```

必要な場合はログファイルを自身のバケットに保管できますが、ログファイルを自動的にアーカイブまたは削除するように Amazon S3 ライフサイクルルールを定義することもできます。詳細については、Amazon Simple Storage Service ユーザーガイドの「オブジェクトのライフサイクルの管理」を参照してください。

接続ログエントリ

各接続試行には、接続ログファイルにエントリがあります。クライアントリクエストの送信方法は、接続が永続か非永続かによって決まります。非永続接続には 1 つのリクエストがあり、アクセスログと接続ログに 1 つのエントリが作成されます。永続接続には複数のリクエストがあり、アクセスログに複数のエントリを作成し、接続ログに 1 つのエントリを作成します。

内容

構文

接続ログエントリは次の形式を使用します。

```
[timestamp] [client_ip] [client_port] [listener_port] [tls_protocol] [tls_cipher]
 [tls_handshake_latency] [leaf_client_cert_subject] [leaf_client_cert_validity]
 [leaf_client_cert_serial_number] [tls_verify_status]
```

次の表は、接続ログエントリのフィールドを順番にまとめたものです。すべてのフィールドはスペースで区切られています。新しいフィールドが導入されると、ログエントリの最後に追加されます。予期していなかったログエントリの最後のフィールドは無視する必要があります。

フィールド	説明
timestamp	ロードバランサーが接続の確立に成功または失敗した ISO 8601 形式の時間。
client_ip	リクエストを送信したクライアントの IP アドレス。
client_port	リクエスト元のクライアントのポート。
listener_port	クライアントリクエストを受信するロードバランサーリスナーのポート。
tls_protocol	[HTTPS リスナー] ハンドシェイク中に使用される SSL/TLS プロトコル。このフィールドは、SSL/TLS 以外のリクエスト-では に設定されます。

フィールド	説明
tls_cipher	[HTTPS リスナー] ハンドシェイク中に使用される SSL/TLS プロトコル。このフィールドは、SSL/TLS 以外のリクエスト-では に設定されます。
tls_handshake_latency	[HTTPS リスナー] ハンドシェイクが成功するまでに経過した、ミリ秒精度の合計時間を秒単位で表したものです。このフィールドは、次の-場合に に設定されます。 • 受信リクエストは SSL/TLS リクエストではありません。 • ハンドシェイクが正常に確立されていません。
leaf_client_cert_Subject	[HTTPS リスナー] リーフクライアント証明書のサブジェクト名。このフィールドは、次の-場合に に設定されます。 • 受信リクエストは SSL/TLS リクエストではありません。 • ロードバランサーのリスナーで mTLS が有効になっていません。 • サーバーはリーフクライアント証明書をロード/解析できません。
leaf_client_cert_validity	[HTTPS リスナー] リーフクライアント証明書の有効性。`not-after` ISO 8601 形式で `not-before` および を使用します。このフィールドは、次の-場合に に設定されます。 • 受信リクエストは SSL/TLS リクエストではありません。 • ロードバランサーのリスナーで mTLS が有効になっていません。 • サーバーはリーフクライアント証明書をロード/解析できません。
leaf_client_cert_serial_number	[HTTPS リスナー] リーフクライアント証明書のシリアル番号。このフィールドは、次の-場合に に設定されます。 • 受信リクエストは SSL/TLS リクエストではありません。 • ロードバランサーのリスナーで mTLS が有効になっていません。 • サーバーはリーフクライアント証明書をロード/解析できません。

フィールド	説明
tls_verify_status	[HTTPS リスナー] 接続リクエストのステータス。この値は、接続が正常に確立Successされた場合に になります。接続に失敗した場合、値はですFailed:$error_code 。
conn_trace_id	接続のトレーサビリティ ID は、各接続を識別するために使用される一意の不透明な ID です。クライアントとの接続が確立されると、このクライアントからの後続のリクエストには、それぞれのアクセスログエントリにこの ID が含まれます。この ID は、接続ログとアクセスログ間のリンクを作成するための外部キーとして機能します。

エラー理由コード

ロードバランサーが接続を確立できない場合、ロードバランサーは次のいずれかの理由コードを接続ログに保存します。

Code	説明
ClientCertMaxChainDepthExceeded	クライアント証明書チェーンの最大深度を超えました
ClientCertMaxSizeExceeded	クライアント証明書の最大サイズを超えました
ClientCertCrlHit	クライアント証明書が CA によって取り消されました
ClientCertCrlProcessingError	CRL 処理エラー
ClientCertUntrusted	クライアント証明書が信頼されていない

Code	説明	
ClientCertNotYetValid	クライアント証明書はまだ有効ではありません	
ClientCertExpired	クライアント証明書の有効期限が切れている	
ClientCertTypeUnsupported	クライアント証明書タイプはサポートされていません	
ClientCertInvalid	クライアント証明書が無効です	
ClientCertRejected	クライアント証明書がカスタムサーバー検証によって拒否されました	
UnmappedConnectionError	マッピングされていないランタイム接続エラー	

ログエントリの例

接続ログエントリの例を次に示します。

以下は、ポート 443 で相互 TLS 検証モードが有効になっている HTTPS リスナーとの正常な接続のログエントリの例です。

```
2023-10-04T17:05:15.514108Z 203.0.113.1 36280 443 TLSv1.2 ECDHE-RSA-AES128-GCM-
SHA256 4.036 "CN=amazondomains.com,O=endEntity,L=Seattle,ST=Washington,C=US"
 NotBefore=2023-09-21T22:43:21Z;NotAfter=2026-06-17T22:43:21Z FEF257372D5C14D4 Success
```

以下は、ポート 443 で相互 TLS 検証モードが有効になっている HTTPS リスナーとの接続に失敗したログエントリの例です。

```
2023-10-04T17:05:15.514108Z 203.0.113.1 36280 443 TLSv1.2 ECDHE-RSA-AES128-
GCM-SHA256 - "CN=amazondomains.com,O=endEntity,L=Seattle,ST=Washington,C=US"
 NotBefore=2023-09-21T22:43:21Z;NotAfter=2026-06-17T22:43:21Z FEF257372D5C14D4
 Failed:ClientCertUntrusted
```

接続ログファイルの処理

接続ログファイルは圧縮されます。Amazon S3 コンソールを使用してファイルを開くと、ファイルは解凍され、情報が表示されます。ファイルをダウンロードする場合、情報を表示するには解凍する必要があります。

ウェブサイトの需要が大きい場合は、ロードバランサーによって数 GB のデータ量のログファイルが生成されることがあります。処理を使用して、このような大量のデータを line-by-line 処理できない場合があります。このため、場合によっては、並列処理ソリューションを提供する分析ツールを使用する必要があります。例えば、次の分析ツールを使用して接続ログを分析および処理できます。

- 「Amazon Athena」は、Amazon S3 内のデータを標準 SQL を使用して簡単に分析できるインタラクティブなクエリサービスです。
- Loggly
- Splunk
- Sumo logic

Application Load Balancer の接続ログを有効にする

ロードバランサーの接続ログを有効にするときは、ロードバランサーがログを保存する S3 バケットの名前を指定する必要があります。このバケットは、バケットにアクセスログを書き込む許可を Elastic Load Balancing に付与するバケットポリシーが必要です。

タスク

- ステップ 1: S3 バケットを作成する
- ステップ 2: S3 バケットにポリシーをアタッチする
- ステップ 3: 接続ログを設定する
- ステップ 4: バケット許可を確認
- トラブルシューティング

ステップ 1: S3 バケットを作成する

接続ログを有効にするときは、接続ログに S3 バケットを指定する必要があります。既存のバケットを使用するか、接続ログ専用のバケットを作成できます。バケットは、次の要件を満たしている必要があります。

要件

- バケットは、ロードバランサーと同じリージョンに配置されている必要があります。バケットとロードバランサーは、異なるアカウントにより所有できます。
- サポートされている唯一のサーバー側の暗号化オプションは、Amazon S3 マネージドキー (SSE-S3) です。詳細については、「Amazon S3 マネージド暗号化キー (SSE-S3)」を参照してください。

Amazon S3 コンソールを使用して S3 バケットを作成するには

1. Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [バケットを作成] を選択します。
3. [バケットを作成] ページで、次の操作を実行します。
   a. [バケット名] にバケットの名前を入力します。この名前は、Amazon S3 内で既存の、すべてのバケット名の中で一意である必要があります。リージョンによっては、バケット名にその他の制限が設けられていることがあります。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「バケットの制約と制限」を参照してください。
   b. [AWS リージョン] で、ロードバランサーを作成したリージョンを選択します。
   c. [デフォルトの暗号化] には、[Amazon S3 マネージドキー (SSE-S3)] を選択します。
   d. [バケットを作成] を選択します。

ステップ 2: S3 バケットにポリシーをアタッチする

S3 バケットには、接続ログをバケットに書き込むアクセス許可を Elastic Load Balancing に付与するバケットポリシーが必要です。バケットポリシーは、バケットのアクセス許可を定義するためにアクセスポリシー言語で記述された JSON ステートメントのコレクションです。各ステートメントには 1 つのアクセス許可に関する情報が含まれ、一連の工レメントが使用されます。

既にポリシーがアタッチされている既存のバケットを使用している場合は、Elastic Load Balancing 接続ログの ステートメントをポリシーに追加できます。その場合、結果として得られるアクセス許可のセットを評価して、接続ログのバケットにアクセスする必要があるユーザーに適していることを確認することをお勧めします。

使用可能なバケットポリシー

使用するバケットポリシーは、 AWS リージョン とゾーンのタイプによって異なります。

2022 年 8 月以降に利用可能になったリージョン

このポリシーは、指定されたログ配信サービスに許可を付与します。このポリシーは、以下のリージョンのアベイラビリティーゾーンと Local Zones にあるロードバランサーに使用してください。

- アジアパシフィック (ハイデラバード)
- アジアパシフィック (メルボルン)
- 欧州 (スペイン)
- 欧州 (チューリッヒ)
- イスラエル (テルアビブ)
- 中東 (アラブ首長国連邦)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "logdelivery.elasticloadbalancing.amazonaws.com"
      },
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::bucket-name/prefix/AWSLogs/aws-account-id/*"
    }
  ]
}
```

2022 年 8 月より前に利用可能になったリージョン

このポリシーは、指定された Elastic Load Balancing アカウント ID に許可を付与します。このポリシーは、以下のリストに記載されている リージョンのアベイラビリティーゾーンまたは Local Zones にあるロードバランサーに使用してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::elb-account-id:root"
      },
```

```
      "Action": "s3:PutObject",
       "Resource": "arn:aws:s3:::bucket-name/prefix/AWSLogs/aws-account-id/*"
    }
  ]
}
```

elb-account-id を、リージョンの Elastic Load Balancing AWS アカウント の の ID に置き換えます。

- 米国東部 (バージニア北部) – 127311923021
- 米国東部 (オハイオ) — 033677994240
- 米国西部 (北カリフォルニア) – 027434742980
- 米国西部 (オレゴン) — 797873946194
- アフリカ (ケープタウン) - 098369216593
- アジアパシフィック (香港) - 754344448648
- アジアパシフィック (ジャカルタ) – 589379963580
- アジアパシフィック (ムンバイ) – 718504428378
- アジアパシフィック (大阪) – 383597477331
- アジアパシフィック (ソウル) – 600734575887
- アジアパシフィック (シンガポール) – 114774131450
- アジアパシフィック (シドニー) — 783225319266
- アジアパシフィック (東京) — 582318560864
- カナダ (中部) – 985666609251
- 欧州 (フランクフルト) – 054676820928
- 欧州 (アイルランド) – 156460612806
- 欧州 (ロンドン) – 652711504416
- ヨーロッパ (ミラノ) - 635631232127
- 欧州 (パリ) – 009996457667
- 欧州 (ストックホルム) – 897822967062
- 中東 (バーレーン) — 076674570225
- 南米 (サンパウロ) – 507241528517
- AWS GovCloud （米国西部) – 048591011584
- AWS GovCloud （米国東部) – 190560391635

my-s3-arn を接続ログの場所の ARN に置き換えます。指定する ARN は、ステップ 3 で接続ログを有効にするときにプレフィックスを指定するかどうかによって異なります。

- プレフィックス付きの ARN の例

```
arn:aws:s3:::bucket-name/prefix/AWSLogs/aws-account-id/*
```

- プレフィックスなしの ARN の例

```
arn:aws:s3:::bucket-name/AWSLogs/aws-account-id/*
```

Effect が NotPrincipalの場合の の使用Deny。

Amazon S3 バケットポリシーが を 値Effectとともに使用しDeny、
以下の例NotPrincipalに示すように が含まれている場合は、
logdelivery.elasticloadbalancing.amazonaws.comがServiceリストに含まれていることを確認してください。

```
{
"Effect": "Deny",
"NotPrincipal": {
    "Service": [
       "logdelivery.elasticloadbalancing.amazonaws.com",
       "example.com"
},
```

Amazon S3 コンソールを使用して接続ログのバケットポリシーをバケットにアタッチするには

1. https://console.aws.amazon.com/s3/でAmazon S3 コンソールを開きます。
2. バケットの名前を選択して、その詳細ページを開きます。
3. [許可] を選択してから、[バケットポリシー]、[編集] の順に選択します。
4. 必要な許可を付与するようにバケットポリシーを更新します。
5. [変更の保存] をクリックします。

ステップ 3: 接続ログを設定する

次の手順を使用して、ログファイルをキャプチャして S3 バケットに配信するように接続ログを設定します。

要件

バケットはステップ 1 で説明した要件を満たしている必要があり、ステップ 2 で説明したようにバケットポリシーをアタッチする必要があります。プレフィックスを指定する場合は、文字列「」を含めないでくださいAWSLogs。

コンソールを使用してロードバランサーの接続ログを有効にするには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[ロードバランサー] を選択します。
3. ロードバランサーの名前を選択して、その詳細ページを開きます。
4. [属性] タブで、[編集] を選択します。
5. モニタリング で、接続ログ をオンにします。
6. [S3 ロケーション] には、ログファイルの S3 URI を入力します。指定する URI は、プレフィックスを使用しているかどうかによって異なります。
   - プレフィックスが付いた URI: `s3://bucket-name/prefix`
   - プレフィックスなしの URI: `s3://bucket-name`
7. [変更の保存] をクリックします。

を使用して接続ログを有効にするには AWS CLI

modify-load-balancer-attributes コマンドを使用します。

接続ログの S3 バケットを管理するには

接続ログ用に設定したバケットを削除する前に、接続ログを必ず無効にしてください。そうしないと、同じ名前で必要なバケットポリシーを持つ新しいバケットがあるが、所有していない で AWS アカウント 作成された場合、Elastic Load Balancing はロードバランサーの接続ログをこの新しいバケットに書き込む可能性があります。

ステップ 4: バケット許可を確認

ロードバランサーの接続ログを有効にすると、Elastic Load Balancing は S3 バケットを検証し、テストファイルを作成して、バケットポリシーが必要なアクセス許可を指定していることを確認します。Amazon S3 コンソールを使用して、テストファイルが作成されたことを確認できます。テストファイルは実際の接続ログファイルではなく、レコード例は含まれていません。

Elastic Load Balancing が S3 バケットにテストファイルを作成したことを確認するには

1. Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. 接続ログに指定したバケットの名前を選択します。
3. テストファイル「ELBConnectionLogTestFile」に移動します。場所は、プレフィックスを使用しているかどうかによって異なります。
   - プレフィックスが付いている場合の場所: *my-bucket*/*prefix*/AWSLogs/*123456789012*/ELBConnectionLogTestFile
   - プレフィックスが付いていない場合の場所: *my-bucket*/AWSLogs/*123456789012*/ELBConnectionLogTestFile

トラブルシューティング

アクセス拒否エラーが表示される場合は、以下の原因が考えられます。

- バケットポリシーは、バケットに接続ログを書き込むアクセス許可を Elastic Load Balancing に付与しません。そのリージョンに対して正しいバケットポリシーを使用していることを確認してください。リソース ARN が、接続ログを有効にしたときに指定したのと同じバケット名を使用していることを確認します。接続ログを有効にしたときにプレフィックスを指定しなかった場合は、リソース ARN にプレフィックスが含まれていないことを確認します。
- バケットが、サポートされていないサーバー側の暗号化オプションを使用している。バケットは、Amazon S3 マネージドキー (SSE-S3) を使用する必要があります。

Application Load Balancer の接続ログを無効にする

ロードバランサーの接続ログはいつでも無効にできます。接続ログを無効にすると、削除するまで接続ログは S3 バケットに残ります。詳細については、Amazon Simple Storage Service ユーザーガイドでバケットの使用について参照してください。

コンソールを使用して接続ログを無効にするには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[ロードバランサー] を選択します。
3. ロードバランサーの名前を選択して、その詳細ページを開きます。
4. [属性] タブで、[編集] を選択します。

5. モニタリング では、接続ログ をオフにします。
6. [変更の保存] をクリックします。

を使用して接続ログを無効にするには AWS CLI

modify-load-balancer-attributes コマンドを使用します。

Application Load Balancer のリクエストをトレースする

ロードバランサーは、クライアントからのリクエストを受信すると、ターゲットにリクエストを送信する前に、[X-Amzn-Trace-Id] ヘッダーを追加、または更新します。ロードバランサーとターゲットとの間のサービスまたはアプリケーションはすべて、このヘッダーの追加、または更新ができます。

クライアントからのターゲットまたは他のサービスへの HTTP リクエストを追跡するため、リクエストのトレースを使用できます。アクセスログを有効にすると、[X-Amzn-Trace-Id] ヘッダーの内容が記録されます。詳細については、「Application Load Balancer のアクセスログ」を参照してください。

構文

[X-Amzn-Trace-Id] ヘッダーには、次の形式を含むフィールドが含まれます。

```
Field=version-time-id
```

フィールド

フィールドの名前。サポートされる値は `Root` と `Self` です。

アプリケーションは独自の目的で任意のフィールドを追加できます。ロードバランサーは、これらのフィールドを維持しますが、使用しません。

バージョン

バージョン番号。

time

エポック時間 (秒)。

id

追跡識別子。

例

[X-Amzn-Trace-Id] ヘッダーが受信リクエストにない場合、ロードバランサーは、Root フィールドのあるヘッダーを作成し、リクエストを転送します。以下に例を示します。

```
X-Amzn-Trace-Id: Root=1-67891233-abcdef012345678912345678
```

[X-Amzn-Trace-Id] ヘッダーがあり、Root フィールドがある場合、ロードバランサーは Self フィールドを挿入し、リクエストを転送します。以下に例を示します。

```
X-Amzn-Trace-Id: Self=1-67891233-12456789abcdef012345678;Root=1-67891233-
abcdef012345678912345678
```

アプリケーションが Root フィールドとカスタムフィールドのあるヘッダーを追加した場合、ロードバランサーは両方のフィールドを保持し、Self フィールドを挿入して、リクエストを転送します。

```
X-Amzn-Trace-Id: Self=1-67891233-12456789abcdef012345678;Root=1-67891233-
abcdef012345678912345678;CalledFrom=app
```

[X-Amzn-Trace-Id] ヘッダーがあり、Self フィールドがある場合、ロードバランサーは Self フィールドの値を更新します。

制限事項

- ロードバランサーは、レスポンスを受信するときではなく、受信リクエストを受け取るときにヘッダーを更新します。
- HTTP ヘッダーが 7 KB より大きい場合、ロードバランサーは、[X-Amzn-Trace-Id] ヘッダーを Root フィールドで書き換えます。
- では WebSockets、アップグレードリクエストが成功するまで追跡できません。

AWS CloudTrailによる Application Load Balancer での API 呼び出しのログ記録

Elastic Load Balancing は AWS CloudTrail、Elastic Load Balancing のユーザー、ロール、または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。は、Elastic Load Balancing のすべての API コールをイベントとして CloudTrail キャプチャしま

す。 Elastic Load Balancing キャプチャされた呼び出しには、 からの呼び出し AWS Management Console 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。証跡を作成する場合は、Elastic Load Balancing の CloudTrail イベントなど、Amazon S3 バケットへのイベントの継続的な配信を有効にすることができます。 Amazon S3 証跡を設定しない場合でも、CloudTrail コンソールのイベント履歴 で最新のイベントを表示できます。によって収集された情報を使用して CloudTrail、Elastic Load Balancing に対するリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

の詳細については CloudTrail、「AWS CloudTrail ユーザーガイド」を参照してください。

クライアントがロードバランサーに対してリクエストを作成するときなど、ロードバランサーに対する他のアクションを監視するには、アクセスログを使用します。詳細については、「Application Load Balancer のアクセスログ」を参照してください。

の Elastic Load Balancing 情報 CloudTrail

CloudTrail AWS アカウントを作成すると、 がアカウントで有効になります。Elastic Load Balancing でアクティビティが発生すると、そのアクティビティは CloudTrail イベント履歴 の他の AWS サービスイベントとともにイベントに記録されます。 AWS アカウントで最近のイベントを表示、検索、ダウンロードできます。詳細については、「イベント履歴 を含む CloudTrail イベントの表示」を参照してください。

Elastic Load Balancing のイベントなど、 AWS アカウント内のイベントの継続的な記録については、証跡を作成します。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。デフォルトでは、コンソールで証跡を作成すると、証跡はすべての AWS リージョンに適用されます。証跡は、 AWS パーティションのすべてのリージョンからのイベントをログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、 CloudTrail ログで収集されたイベントデータをより詳細に分析し、それに基づいて行動するように、他の AWS サービスを設定できます。詳細については、次を参照してください:

- 「証跡作成の概要」
- CloudTrail がサポートするサービスと統合
- の Amazon SNS 通知の設定 CloudTrail
- 複数のリージョンからの CloudTrail ログファイルの受信と複数のアカウントからの CloudTrail ログファイルの受信

Application Load Balancer のすべての Elastic Load Balancing アクションは によってログに記録 CloudTrail され、Elastic Load Balancing API リファレンスバージョン 2015-12-01 に記載されてい

ます。例えば、 および DeleteLoadBalancerアクションを呼び出すCreateLoadBalancerと、CloudTrail ログファイルにエントリが生成されます。

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:

- リクエストが、ルート認証情報を使用して送信されたか
- 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。

詳細については、CloudTrailuserIdentity 要素」を参照してください。

Elastic Load Balancing ログファイルのエントリの理解

証跡は、指定した Amazon S3 バケットにイベントをログファイルとして配信できるようにする設定です。 CloudTrail ログファイルには 1 つ以上のログエントリが含まれます。イベントは任意ソースからの単一リクエストを表し、リクエストされたアクション、アクションの日時、リクエストパラメータなどの情報を含みます。 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、特定の順序では表示されません。

ログファイルには、Elastic Load Balancing AWS API コールだけでなく AWS アカウント、 のすべての API コールのイベントが含まれます。値 eventSource を使用して elasticloadbalancing.amazonaws.com 要素を確認することで、Elastic Load Balancing API に対する呼び出しを見つけることができます。CreateLoadBalancer などの特定のアクションのレコードを表示するには、アクション名で eventName 要素を確認します。

以下は、Application Load Balancer を作成し、 を使用して削除したユーザーの Elastic Load Balancing の CloudTrail ログレコードの例です AWS CLI。 Application Load Balancer 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： 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： 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"
}
```

Application Load Balancer のトラブルシューティング

以下の情報は、Application Load Balancer の問題のトラブルシューティングに役立ちます。

問題点

登録されたターゲットが実行中でない

ターゲットが `InService` 状態になるまでに予想以上に時間がかかっている場合、ヘルスチェックに合格していない可能性があります。ターゲットは、ヘルスチェックに合格するまで実行されません。詳細については、「ターゲットグループのヘルスチェック」を参照してください。

インスタンスがヘルスチェックに合格していないことを確認したら、以下の点を確認します。

セキュリティグループでトラフィックが許可されていない

インスタンスに関連付けられたセキュリティグループでは、ヘルスチェックポートとヘルスチェックプロトコルを使用してロードバランサーからのトラフィックを許可する必要があります。インスタンスセキュリティグループにルールを追加して、ロードバランサーセキュリティグループからのすべてのトラフィックを許可できます。また、ロードバランサーのセキュリティグループは、インスタンスへのトラフィックを許可する必要があります。

ネットワークアクセスコントロールリスト (ACL) ではトラフィックが許可されない

インスタンスのサブネットに関連付けられたネットワーク ACL では、ヘルスチェックポートでインバウンドトラフィックを許可し、一時ポート (1024-65535) で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。ロードバランサーノードのサブネットに関連付けられたネットワーク ACL では、一時ポートでインバウンドトラフィックを許可し、ヘルスチェックおよび一時ポートで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。

ping パスが存在しない

ヘルスチェックのターゲットページを作成し、そのパスを ping パスとして指定します。

接続がタイムアウトする

最初に、ターゲットのプライベート IP アドレスとヘルスチェックプロトコルを使用して、ネットワーク内から直接ターゲットに接続できることを確認します。接続できない場合は、インスタンスの使用率が高すぎないかどうかを確認し、ビジー状態で応答できない場合はさらにターゲットをターゲットグループに追加します。接続できる場合、ヘルスチェックのタイムアウト時間の前に、ターゲットページが応答していない可能性があります。ヘルスチェック用のよりシンプルなターゲットページを選択するか、ヘルスチェックの設定を調整します。

ターゲットが正常なレスポンスコードを返さなかった

デフォルトの成功コードは 200 ですが、ヘルスチェックを設定するときにオプションで成功コードを追加して指定できます。ロードバランサーで予期される成功コードを確認し、成功時にアプリケーションがそれらのコードを返すよう設定されていることを確認します。

ターゲットレスポンスコードの形式が正しくないか、ターゲットへの接続中にエラーが発生しました

アプリケーションがロードバランサーの正常性チェックリクエストに応答することを確認します。一部のアプリケーションでは、正常性チェックに応答するために追加の設定が必要です。例えば、ロードバランサーから送信された HTTP ホストヘッダーに応答するための仮想ホスト設定などです。ホストヘッダー値には、ターゲットのプライベート IP アドレスが含まれ、デフォルトポートを使用しない場合はヘルスチェックポートが続きます。ターゲットがデフォルトのヘルスチェックポートを使用する場合、ホストヘッダー値にはターゲットのプライベート IP アドレスのみが含まれます。例えば、ターゲットのプライベート IP アドレスが `10.0.0.10`で、ヘルスチェックポートが の場合`8080`、ロードバランサーによってヘルスチェックで送信される HTTP ホストヘッダーは ですHost: 10.0.0.10:8080。ターゲットのプライベート IP アドレスがで`10.0.0.10`、ヘルスチェックポートが `80`の場合、ロードバランサーによってヘルスチェックで送信される HTTP ホストヘッダーは ですHost: 10.0.0.10。アプリケーションの正常性チェックを正常に実行するには、そのホストに応答する仮想ホスト設定、またはデフォルト設定

が必要になる場合があります。正常性チェックリクエストは、次の属性を有しています。すなわち、User-Agent が ELB-HealthChecker/2.0 に設定され、メッセージヘッダーフィールドの行終端記号はシーケンス CRLF、ヘッダーは最初の空の行で終了し、その後に CRLF が続きます。

クライアントがインターネット向けロードバランサーに接続できない

ロードバランサーがリクエストに応答しない場合は、以下の点を確認します。

インターネット向けロードバランサーがプライベートサブネットにアタッチされている

ロードバランサーのパブリックサブネットを指定する必要があります。パブリックサブネットには Virtual Private Cloud (VPC) のインターネットゲートウェイへのルートがあります。

セキュリティグループまたはネットワーク ACL でトラフィックが許可されていない

ロードバランサーのセキュリティグループ、およびロードバランサーサブネットのネットワーク ACL で、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リスナーポートで許可する必要があります。

ロードバランサーがカスタムドメインに送信されたリクエストを受信しません

ロードバランサーがカスタムドメインに送信されたリクエストを受信しない場合は、以下の点を確認します。

カスタムドメイン名がロードバランサーの IP アドレスを解決しません

- コマンドラインインターフェイスを使用して、カスタムドメイン名がどの IP アドレスを解決するのかを確認します。
  - Linux、macOS、または Unix — ターミナルで dig コマンドを使用できます。例 `dig example.com`
  - Windows — コマンドプロンプトで nslookup コマンドを使用できます。例 `nslookup example.com`
- コマンドラインインターフェイスを使用して、ロードバランサーの DNS 名がどの IP アドレスを解決するのかを確認します。

- 2 つの出力の結果を比較します。IP アドレスは一致する必要があります。

Route 53 を使用してカスタムドメインをホストしている場合は、Amazon Route 53 開発者ガイドの 「ドメインがインターネットで利用できない」を参照してください。

ロードバランサーに送信された HTTPS リクエストは「NET::ERR_CERT_COMMON_NAME_INVALID」を返します

ロードバランサーから HTTPS リクエストが `NET::ERR_CERT_COMMON_NAME_INVALID` を受信している場合は、次の原因が考えられます。

- HTTPS リクエストで使用されているドメイン名が、リスナーに関連付けられた ACM 証明書で指定されている代替名と一致しません。
- ロードバランサーのデフォルトの DNS 名が使用されています。`*.amazonaws.com` ドメインに対してパブリック証明書をリクエストできないため、デフォルトの DNS 名を使用して HTTPS リクエストを実行できません。

ロードバランサーが処理時間の増加を示しています

ロードバランサーは、設定に基づいて処理時間を異なる方法でカウントします。

- AWS WAF が Application Load Balancer に関連付けられていて、クライアントが HTTP POST リクエストを送信する場合、POST リクエストのデータを送信する時間はロードバランサーのアクセスログの `request_processing_time`フィールドに反映されます。この動作は HTTP POST リクエストで想定されるものです。
- AWS WAF が Application Load Balancer に関連付けられておらず、クライアントが HTTP POST リクエストを送信した場合、POST リクエストのデータを送信する時間はロードバランサーのアクセスログの `target_processing_time`フィールドに反映されます。この動作は HTTP POST リクエストで想定されるものです。

ロードバランサーは、レスポンスコード 000 を送信します。

HTTP/2 接続では、いずれかのヘッダーの圧縮された長さが 8 K バイトを超える場合、または 1 つの接続を介して処理されるリクエスト数が 10,000 を超える場合、ロードバランサーは GOAWAY フレームを送信し、TCP FIN を使用して接続を閉じます。

ロードバランサーが HTTP エラーを生成する

次の HTTP エラーは、ロードバランサーで生成されます。ロードバランサーはクライアントに HTTP コードを送信し、アクセスログにリクエストを保存して、HTTPCode_ELB_4XX_Count または HTTPCode_ELB_5XX_Count メトリクスを増やします。

エラー

- HTTP 400: Bad request
- HTTP 401: Unauthorized
- HTTP 403: Forbidden
- HTTP 405: Method not allowed
- HTTP 408: Request timeout
- HTTP 413: Payload too large
- HTTP 414: URI too long
- HTTP 460
- HTTP 463
- HTTP 464
- HTTP 500: Internal server error
- HTTP 501: Not implemented
- HTTP 502: Bad gateway
- HTTP 503: Service Unavailable
- HTTP 504: Gateway Timeout
- HTTP 505: バージョンはサポートされていません
- HTTP 507: ストレージが不十分
- HTTP 561: Unauthorized

HTTP 400: Bad request

考えられる原因:

- クライアントが HTTP 仕様を満たさない誤った形式のリクエストを送信した。
- リクエストヘッダーが、リクエスト行あたり 16 K、1 つのヘッダーあたり 16 K、またはリクエストヘッダー全体に対して 64 K を超えている。

- クライアントが、リクエスト本文全体を送信する前に接続を閉じた。

HTTP 401: Unauthorized

ユーザーを認証するようリスナールールを設定しましたが、以下のいずれかが該当しません。

- `OnUnauthenticatedRequest` が認証されていないユーザーを拒否するように設定されているか、IdP によってアクセスが拒否されました。
- IdP によって返されるクレームのサイズが、ロードバランサーによってサポートされる最大サイズを超えています。
- クライアントがホストヘッダーなしで HTTP/1.0 リクエストを送信し、ロードバランサーはリダイレクト URL を生成できませんでした。
- リクエストされたスコープで ID トークンが返されません。
- クライアントのログインがタイムアウトする前にログインプロセスが完了しません。詳細については、「クライアントログインタイムアウト」を参照してください。

HTTP 403: Forbidden

Application Load Balancer へのリクエストをモニタリングするように AWS WAF ウェブアクセスコントロールリスト (ウェブ ACL) を設定し、リクエストをブロックしました。

HTTP 405: Method not allowed

クライアントが、Application Load Balancer でサポートされていない TRACE メソッドを使用しました。

HTTP 408: Request timeout

アイドルタイムアウト期間の期限が切れる前に、クライアントからデータが送信されませんでした。TCP キープアライブを送信しても、このタイムアウトを防ぐことはできません。各アイドルタイムアウト期間が経過する前に、1 バイト以上のデータを送信します。必要に応じて、アイドルタイムアウト期間を長くします。

HTTP 413: Payload too large

考えられる原因:

- ターゲットは Lambda 関数で、リクエストボディが 1 MB を超えています。

- リクエストヘッダーが、リクエスト行あたり 16 K、1 つのヘッダーあたり 16 K、またはリクエストヘッダー全体に対して 64 K を超えている。

HTTP 414: URI too long

リクエストの URL またはクエリ文字列パラメータが大きすぎます。

HTTP 460

ロードバランサーはクライアントからリクエストを受信したが、クライアントはアイドルタイムアウトが経過する前に、ロードバランサーとの接続を閉じました。

クライアントのタイムアウト期間がロードバランサーのアイドルタイムアウト期間より長いかどうか確認してください。クライアントのタイムアウト期間が経過する前に、ターゲットがクライアントにレスポンスを返すことを確認するか、クライアントがサポートする場合は、クライアントのタイムアウト期間を増やしてロードバランサーのアイドルタイムアウトに合わせます。

HTTP 463

多すぎる IP アドレスを含む X-Forwarded-For リクエストヘッダーがロードバランサーに送信されました。IP アドレスの上限は 30 です。

HTTP 464

ロードバランサーは、ターゲットグループプロトコルのバージョン構成と互換性のない着信リクエストプロトコルを受信しました。

考えられる原因:

- リクエストプロトコルは HTTP/1.1 であるが、ターゲットグループのプロトコルバージョンが gRPC または HTTP/2 である。
- リクエストプロトコルは gRPC であるが、ターゲットグループのプロトコルバージョンが HTTP/1.1 である。
- リクエストプロトコルは HTTP/2 であり、リクエストは POST ではないが、ターゲットグループのプロトコルバージョンが gRPC である。

HTTP 500: Internal server error

考えられる原因:

- AWS WAF ウェブアクセスコントロールリスト (ウェブ ACL) を設定し、ウェブ ACL ルールの実行中にエラーが発生しました。
- ロードバランサーは、IdP トークンのエンドポイントまたは IdP ユーザー情報エンドポイントと通信できません。
  - IdP の DNS がパブリックに解決可能であることを確認します。
  - ロードバランサーのセキュリティグループおよび VPC のネットワーク ACL がこれらのエンドポイントに対するアウトバウンドアクセスを許可していることを検証します。
  - VPC がインターネット接続されていることを確認します。内部向けロードバランサーがある場合は、NAT ゲートウェイを使用してインターネットアクセスを有効にします。
- IdP から受信したユーザークレームが 11 KB を超えています。

HTTP 501: Not implemented

サポートされていない値を含む Transfer-Encoding ヘッダーがロードバランサーに送信されました。Transfer-Encoding でサポートされている値は、`chunked` と `identity` です。代わりに、Content-Encoding ヘッダーを使用することができます。

HTTP 502: Bad gateway

考えられる原因:

- 接続の確立を試みているときに、ロードバランサーがターゲットから TCP RST を受信した。
- 接続の確立を試みているときに、ロードバランサーがターゲットから予期しないレスポンスを受信した (例: 「ICMP Destination unreachable (Host unreachable) (ICMP 送信先に到達できません (ホストに到達できません))」など)。ターゲットポートでロードバランサーサブネットからターゲットへのトラフィックが許可されているかどうかを確認します。
- ロードバランサーにターゲットへの未処理のリクエストがあるときに、ターゲットが TCP RST または TCP FIN との接続を閉じた。ターゲットのキープアライブ期間がロードバランサーのアイドルタイムアウト値よりも短いことを確認します。
- ターゲットのレスポンス形式が正しくないか、有効でない HTTP ヘッダーが含まれている。
- ターゲットレスポンスヘッダーは、レスポンスヘッダー全体で32 K を超えました。
- 登録解除されたターゲットによって処理されていたリクエストで登録解除の遅延期間が経過しました。時間のかかるオペレーションが完了できるように、遅延期間を増やします。
- ターゲットは Lambda 関数で、レスポンスボディが 1 MB を超えています。
- ターゲットは、設定されたタイムアウトに達する前に応答しなかった Lambda 関数です。

- ターゲットとなるのは、エラーを返した Lambda 関数、または Lambda サービスがスロットリングした関数です。
- ロードバランサーがターゲットへの接続中に SSL ハンドシェイクエラーが発生しました。

詳細については、 AWS サポートナレッジセンターのApplication Load Balancer HTTP 502 エラーのトラブルシューティング方法」を参照してください。

HTTP 503: Service Unavailable

ロードバランサーのターゲットグループに登録済みターゲットがありません。

HTTP 504: Gateway Timeout

考えられる原因:

- ロードバランサーは、接続タイムアウトが期限切れになる (10 秒) 前にターゲットへの接続の確立に失敗した。
- ロードバランサーはターゲットへの接続を確立したが、アイドルタイムアウト期間が経過する前にターゲットが応答しなかった。
- サブネットのネットワーク ACL で、ターゲットから一時ポート (1024-65535) のロードバランサーノードへのトラフィックが許可されなかった。
- ターゲットがエンティティ本文より大きな Content-Length ヘッダーを返した。ロードバランサーが欠落しているバイトを待機してタイムアウトした。
- ターゲットは Lambda 関数であり、接続タイムアウトが期限切れになる前に Lambda サービスが応答しませんでした。
- ロードバランサーがターゲットへの接続中に SSL ハンドシェイクタイムアウト (10 秒) を検出しました。

HTTP 505: バージョンはサポートされていません

ロードバランサーが予期しない HTTP バージョンリクエストを受信しました。例えば、ロードバランサーが HTTP/1 接続を確立したが、HTTP/2 リクエストを受信したとします。

HTTP 507: ストレージが不十分

リダイレクト URL が長すぎます。

HTTP 561: Unauthorized

リスナールールがユーザーを認証するように設定されていますが、IdP がユーザーの認証時にエラーコードを返しました。関連するエラー理由コードについてはアクセスログを確認してください。

ターゲットが HTTP エラーを生成する

ロードバランサーはターゲットからの有効な HTTP レスポンスを、HTTP エラーを含めてクライアントに転送します。ターゲットによって生成された HTTP エラーは、`HTTPCode_Target_4XX_Count` および `HTTPCode_Target_5XX_Count` メトリクスに記録されます。

AWS Certificate Manager 証明書は使用できません

Application Load Balancer で HTTPS リスナーを使用することを決定する場合、 AWS Certificate Manager は証明書を発行する前にドメインの所有権を検証する必要があります。設定中にこのステップを実行しなかった場合、証明書は `Pending Validation` 状態のままとなり、検証されるまで使用できません。

- E メール検証を使用する場合は、「AWS Certificate Manager ユーザーガイド」の「E メール検証」を参照してください。
- DNS での検証を使用する場合は、「AWS Certificate Manager ユーザーガイド」の「DNS での検証」を参照してください。

複数行のヘッダーはサポートされていません

Application Load Balancer は、`message/http` メディアタイプヘッダーを含め、複数行のヘッダーをサポートしていません。複数行のヘッダーが提供されると、Application Load Balancer は、コロン文字「`:`」を付加してターゲットに渡します。

リソースマップを使用した異常なターゲットのトラブルシューティング

Application Load Balancer ターゲットがヘルスチェックに失敗している場合は、リソースマップを使用して異常なターゲットを検索し、失敗理由コードに基づいてアクションを実行できます。詳細については、「Application Load Balancer リソースマップ」を参照してください。

リソースマップには、概要 と異常なターゲットマップ の 2 つのビューがあります。 概要はデフォルトで選択され、ロードバランサーのすべてのリソースが表示されます。異常なターゲットマップビューを選択すると、Application Load Balancer に関連付けられている各ターゲットグループ内の異常なターゲットのみが表示されます。

> **Note**
>
> リソースマップ内の該当するすべてのリソースのヘルスチェックの概要とエラーメッセージを表示するには、リソースの詳細を表示を有効にする必要があります。有効になっていない場合は、各リソースを選択して詳細を表示する必要があります。

ターゲットグループ列には、各ターゲットグループの正常ターゲットと異常ターゲットの概要が表示されます。これにより、すべてのターゲットがヘルスチェックに失敗しているか、特定のターゲットのみが失敗しているかを判断できます。ターゲットグループ内のすべてのターゲットがヘルスチェックに失敗している場合は、ターゲットグループの設定を確認します。ターゲットグループ名を選択して、新しいタブで詳細ページを開きます。

Targets 列には、各ターゲットの TargetID と現在のヘルスチェックステータスが表示されます。ターゲットに異常があると、ヘルスチェックの失敗理由コードが表示されます。1 つのターゲットがヘルスチェックに失敗する場合は、ターゲットに十分なリソースがあることを確認し、ターゲットで実行されているアプリケーションが使用可能であることを確認します。ターゲット ID を選択して、新しいタブで詳細ページを開きます。

Export を選択すると、Application Load Balancer のリソースマップの現在のビューを PDF として エクスポートできます。

インスタンスがヘルスチェックに失敗し、次の問題の失敗理由コードチェックに基づいていることを確認します。

- 異常: HTTP レスポンスの不一致
  - ターゲットで実行されているアプリケーションが、Application Load Balancer のヘルスチェックリクエストに正しい HTTP レスポンスを送信していることを確認します。
  - または、Application Load Balancer のヘルスチェックリクエストを更新して、ターゲットで実行されているアプリケーションからのレスポンスと一致するようにすることもできます。
- 異常: リクエストがタイムアウトしました
  - ターゲットと Application Load Balancer に関連付けられているセキュリティグループとネットワークアクセスコントロールリスト (ACL) が接続をブロックしていないことを確認します。

- ターゲットに、Application Load Balancer からの接続を受け入れるのに十分なリソースがあることを確認します。
- ターゲットで実行されているアプリケーションのステータスを確認します。
- Application Load Balancer のヘルスチェックレスポンスは、各ターゲットのアプリケーションログで表示できます。詳細については、「ヘルスチェックの理由コード」を参照してください。

- 異常： FailedHealthChecks
  - ターゲットで実行されているアプリケーションのステータスを確認します。
  - ターゲットがヘルスチェックポートでトラフィックをリッスンしていることを確認します。

    HTTPS リスナーを使用する場合

    フロントエンド接続に使用するセキュリティポリシーを選択します。バックエンド接続に使用されるセキュリティポリシーは、使用中のフロントエンドセキュリティポリシーに基づいて自動的に選択されます。

    - HTTPS リスナーがフロントエンド接続に TLS 1.3 セキュリティポリシーを使用している場合、`ELBSecurityPolicy-TLS13-1-0-2021-06`セキュリティポリシーはバックエンド接続に使用されます。
    - HTTPS リスナーがフロントエンド接続に TLS 1.3 セキュリティポリシーを使用していない場合、`ELBSecurityPolicy-2016-08`セキュリティポリシーはバックエンド接続に使用されます。

    詳細については、「セキュリティポリシー」を参照してください。
  - ターゲットがサーバー証明書とキーをセキュリティポリシーで指定された正しい形式で提供していることを確認します。
  - ターゲットが 1 つ以上の一致する暗号と、TLS ハンドシェイクを確立するために Application Load Balancer によって提供されるプロトコルをサポートしていることを確認します。

Application Load Balancer のクォータ

AWS アカウントには、AWS のサービスごとにデフォルトのクォータ (以前は制限と呼ばれていました) があります。特に明記されていない限り、クォータは地域固有です。一部のクォータについては引き上げをリクエストできますが、その他のクォータについてはリクエストできません。

Application Load Balancer のクォータを表示するには、Service Quotas コンソールを開きます。ナビゲーションペインで、AWSservices(のサービス)、Elastic Load Balancing の順に選択します。Elastic Load Balancing の describe-account-limits (AWS CLI) コマンドを使用することもできます。

クォータの増加をリクエストするには、Service Quotas ユーザーガイドの Requesting a quota increase を参照してください。クォータが Service Quotas でまだ使用できない場合は、Elastic Load Balancing の制限引き上げフォームを使用します。

ロードバランサー

AWS アカウントには、Application Load Balancer に関連する以下のクォータがあります。

名前	デフォルト	引き上げ可能
リージョンあたりの Application Load Balancer	50	はい
Application Load Balancer あたりの証明書 (デフォルト証明書を除く)	25	はい
Application Load Balancer あたりのリスナー	50	はい
Application Load Balancer ごとの、アクションあたりのターゲットグループ	5	いいえ
Application Load Balancer あたりのターゲットグループ	100	いいえ
Application Load Balancer あたりのターゲット	1,000	はい

ターゲットグループ

次のクォータはターゲットグループ用です。

名前	デフォルト	引き上げ可能
リージョンあたりのターゲットグループ	3,000 *	はい
リージョンごとのターゲットグループあたりのターゲット (インスタンスまたは IP アドレス)	1,000	はい
リージョンごとのターゲットグループあたりのターゲット (Lambda 関数)	1	いいえ
ターゲットグループあたりのロードバランサー	1	いいえ

* このクォータは、Application Load Balancer および Network Load Balancer によって共有されます。

ルール

以下のクォータはルール用です。

名前	デフォルト	引き上げ可能
Application Load Balancer あたりのルール (デフォルトルールを除く)	100	はい
ルールあたりの条件値	5	いいえ
ルールあたりの条件ワイルドカード	5	いいえ
ルールあたりの一致の評価数	5	いいえ

トラストストア

以下のクォータはトラストストア用です。

名前	デフォルト	引き上げ可能
アカウントあたりのトラストストア	20	はい

名前	デフォルト	引き上げ可能
ロードバランサーごとに、検証モードで mTLS を使用するリスナーの数。	2	いいえ

認証機関証明書

以下のクォータは CA 証明書用です。

名前	デフォルト	引き上げ可能
トラストストアあたりの CA 証明書	25	はい
CA 証明書のサイズ	16KB	いいえ
証明書チェーンの最大深度	4	いいえ

証明書失効リスト

次のクォータは、証明書失効リスト用です。

名前	デフォルト	引き上げ可能
トラストストアあたりの失効リスト	30	はい
トラストストアあたりの取り消しエントリ	500,000	はい
失効リストのファイルサイズ	50 MB	いいえ

HTTP ヘッダー

HTTP ヘッダーには次のようなサイズ制限があります。

名前	デフォルト	引き上げ可能
リクエスト行	16 K	いいえ

名前	デフォルト	引き上げ可能
単一ヘッダー	16 K	いいえ
レスポンスのヘッダー全体	32 K	いいえ
リクエストのヘッダー全体	64 K	いいえ

Application Load Balancer のドキュメント履歴

次の表に、Application Load Balancer のリリースを示します。

変更	説明	日付
リソースマップ	このリリースでは、ロードバランサーのリソースと関係を視覚的な形式で表示するためのサポートが追加されました。	2024 年 3 月 8 日
ワンクリック WAF	このリリースでは、ロードバランサーがワンクリック と統合されている場合の動作の設定のサポートが追加されました AWS WAF。	2024 年 2 月 6 日
相互 TLS	このリリースでは、相互 TLS 認証のサポートが追加されています。	2023 年 11 月 26 日
自動ターゲット重み	このリリースでは、自動ターゲット重みアルゴリズムのサポートが追加されました。	2023 年 11 月 26 日
FIPS 140-3 TLS 終了	このリリースでは、TLS 接続を終了するときに FIPS 140-3 暗号化モジュールを使用するセキュリティポリシーが追加されています。	2023年11月20日
IPv6 を使用してターゲットを登録する	このリリースでは、IPv6 で対処されたときにインスタンスをターゲットとして登録するサポートが追加されました。	2023 年 10 月 2 日

TLS 1.3 をサポートするセキュリティポリシー	このリリースでは、TLS 1.3 の事前定義されたセキュリティポリシーのサポートが追加されました。	2023 年 3 月 22 日
ゾーンシフト	このリリースでは、 との統合を通じて、障害のある単一のアベイラビリティーゾーンからトラフィックをルーティングするサポートが追加されました Amazon Route 53 Application Recovery Controller。	2022 年 11 月 28 日
クロスゾーン負荷分散をオフにする	このリリースでは、クロスゾーン負荷分散を無効にするサポートが追加されました。	2022 年 11 月 28 日
ターゲットグループの正常性	このリリースでは、正常でなければならないターゲットの最小数または割合、およびしきい値に達しない場合にロードバランサーが実行するアクションを設定するサポートが追加されています。	2022 年 11 月 28 日
クロスゾーンロードバランサー	このリリースでは、ターゲットグループレベルでクロスゾーン負荷分散を設定するサポートが追加されました。	2022 年 11 月 17 日
IPv6 ターゲットグループ	このリリースでは、Application Load Balancer の IPv6 ターゲットグループの設定に対するサポートが追加されます。	2021 年 11 月 23 日

IPv6 内部ロードバランサー	このリリースでは、Application Load Balancer の IPv6 ターゲットグループの設定に対するサポートが追加されます。	2021 年 11 月 23 日
AWS PrivateLink および静的 IP アドレス	このリリースでは、Network Load Balancer から Application Load Balancer にトラフィックを直接転送することで、静的 IP アドレスの使用 AWS PrivateLink と公開のサポートが追加されました。	2021 年 9 月 27 日
クライアントポートの保持	このリリースは、クライアントがロードバランサーへの接続に使用したソースポートを保持するための属性を追加します。	2021 年 7 月 29 日
TLS ヘッダー	このリリースでは、ネゴシエートされた TLS バージョンと暗号スイートに関する情報を含む TLS ヘッダーがターゲットに送信する前にクライアントリクエストに追加されることを示す 属性が追加されました。	2021 年 7 月 21 日
追加の ACM 証明書	このリリースは、2048 ビット、3072 ビット、および 4096 ビットのキー長の RSA 証明書と、すべての ECDSA 証明書をサポートします。	2021 年 7 月 14 日

アプリケーションベースの維持	このリリースでは、ロードバランサーの維持セッションをサポートするアプリケーションベースの Cookie が追加されました。	2021 年 2 月 8 日
TLS バージョン 1.2 をサポートする FS のセキュリティポリシー	このリリースでは、TLS バージョン 1.2 をサポートする前方秘匿性 (FS) のセキュリティポリシーが追加されました。	2020 年 11 月 24 日
WAF のフェイルオープンサポート	このリリースでは、ロードバランサーが と統合されている場合の動作の設定のサポートが追加されました AWS WAF。	2020 年 11 月 13 日
gRPC と HTTP/2 のサポート	このリリースでは、gRPC ワークロードと end-to-end HTTP/2 のサポートが追加されています。	2020 年 10 月 29 日
Outpost のサポート	で Application Load Balancer をプロビジョニングできます AWS Outposts。	2020 年 9 月 8 日
Desync 軽減モード	このリリースでは、Desync 軽減モードのサポートが追加されました。	2020 年 8 月 17 日
最小の未処理のリクエスト	このリリースでは、最小の未処理リクエストのアルゴリズムのサポートが追加されました。	2019 年 11 月 25 日

加重ターゲットグループ	このリリースでは、複数のターゲットグループを使用した転送アクションのサポートが追加されています。リクエストは、各ターゲットグループに指定した重みに基づいて、これらのターゲットグループに分散されます。	2019 年 11 月 19 日
[New attribute] (新規属性)	このリリースでは、routing.http.drop_invalid_header_fields.enabled 属性のサポートが追加されています。	2019 年 11 月 15 日
FS のセキュリティポリシー	このリリースでは、事前定義された 3 つの前方秘匿性セキュリティポリシーのサポートが追加されました。	2019 年 10 月 8 日
高度なリクエストルーティング	このリリースでは、リスナールールに関する追加の条件タイプのサポートが追加されています。	2019 年 3 月 27 日
ターゲットとしての Lambda 関数	このリリースでは、Lambda 関数をターゲットとして登録する機能のサポートが追加されています。	2018 年 11 月 29 日
リダイレクトアクション	このリリースでは、ロードバランサーでリクエストを別の URL にリダイレクトするためのサポートが追加されています。	2018 年 7 月 25 日

固定レスポンスアクション	このリリースでは、ロードバランサーでカスタムの HTTP レスポンスを返すためのサポートが追加されています。	2018 年 7 月 25 日
FS および TLS 1.2 のセキュリティポリシー	このリリースでは、2 つの事前定義済みの追加セキュリティポリシーへのサポートが追加されました。	2018 年 6 月 6 日
ユーザー認証	このリリースでは、リクエストをルーティングする前に企業 ID またはソーシャル ID を使用してアプリケーションのユーザーを認証するロードバランサーのサポートが追加されます。	2018 年 5 月 30 日
リソースレベルのアクセス許可	このリリースでは、リソースレベルのアクセス許可と条件キーのタグ付け機能に関するサポートが追加されました。	2018 年 5 月 10 日
スロースタートモード	このリリースでは、スロースタートモードのサポートが追加されました。ロードバランサーは、新しく登録されたターゲットがウォームアップを行っている間は、シェアを徐々に増やしながらリクエストを送信します。	2018 年 3 月 24 日
SNI サポート	このリリースでは、Server Name Indication (SNI) へのサポートを追加しています。	2017 年 10 月 10 日

IP アドレスをターゲットに設定	このリリースでは、IP アドレスをターゲットとして登録する機能のサポートが追加されます。	2017 年 8 月 31 日
ホストベースのルーティング	このリリースでは、ホストヘッダーのホスト名に基づいてリクエストをルーティングするサポートが追加されました。	2017 年 4 月 5 日
TLS 1.1 および TLS 1.2 のセキュリティポリシー	このリリースでは、TLS 1.1 と TLS 1.2 のセキュリティポリシーが追加されます。	2017 年 2 月 6 日
IPv6 サポート	このリリースでは、IPv6 アドレスのサポートが追加されます。	2017 年 1 月 25 日
リクエストトレース	このリリースでは、リクエストのトレースのサポートを追加します。	2016 年 11 月 22 日
TargetResponseTime メトリクスのパーセンタイルのサポート	このリリースでは、Amazon でサポートされている新しいパーセンタイル統計のサポートが追加されました CloudWatch。	2016 年 11 月 17 日
新しい種類のロードバランサー	このリリースの Elastic Load Balancing では、Application Load Balancer が導入されています。	2016 年 8 月 11 日

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。