



Classic Load Balancer

# Elastic Load Balancing



# Elastic Load Balancing: Classic Load Balancer

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon の商標およびトレードドレスは Amazon 以外の製品およびサービスに使用することはできません。また、お客様に誤解を与える可能性がある形式で、または Amazon の信用を損なう形式で使用することもできません。Amazon が所有していない他のすべての商標は、それぞれの所有者の所有物であり、Amazon と提携、接続、または後援されている場合とされていない場合があります。

---

# Table of Contents

「Classic Load Balancer とは？」 .....	1
Classic Load Balancer の概要 .....	1
利点 .....	2
開始方法 .....	3
料金 .....	3
インターネット向けロードバランサー .....	4
ロードバランサーのパブリック DNS 名 .....	4
インターネット向けロードバランサーを作成する .....	5
[開始する前に] .....	5
を使用して Classic Load Balancer を作成する AWS マネジメントコンソール .....	6
内部ロードバランサー .....	9
ロードバランサーのパブリック DNS 名 .....	10
内部ロードバランサーの作成 .....	11
前提条件 .....	11
コンソールを使用した内部ロードバランサーの作成 .....	11
を使用して内部ロードバランサーを作成する AWS CLI .....	14
ロードバランサーの設定 .....	16
アイドル接続のタイムアウト .....	17
コンソールを使用したアイドルタイムアウトの設定 .....	17
を使用してアイドルタイムアウトを設定する AWS CLI .....	18
クロスゾーンロードバランサー .....	18
クロスゾーン負荷分散の有効化 .....	19
クロスゾーン負荷分散の無効化 .....	21
Connection Draining .....	22
Connection Drainingの有効化 .....	23
Connection Drainingの無効化 .....	24
スティッキーセッション .....	25
期間ベースのセッションの維持 .....	26
アプリケーション制御によるセッションの維持 .....	29
Desync 軽減モード .....	32
分類 .....	33
モード .....	34
desync 軽減モードの変更 .....	34
Proxy Protocol .....	35

プロキシプロトコルヘッダー .....	36
プロキシプロトコルを有効にするための前提条件 .....	37
を使用してプロキシプロトコルを有効にする AWS CLI .....	37
を使用してプロキシプロトコルを無効にする AWS CLI .....	39
タグ .....	40
タグの制限 .....	40
タグの追加 .....	41
タグの削除 .....	41
サブネットとゾーン .....	42
要件 .....	43
コンソールを使用してサブネットを設定する .....	44
CLI を使用してサブネットを設定する .....	44
セキュリティグループ .....	45
ロードバランサーのセキュリティグループの推奨ルール .....	46
コンソールを使用したセキュリティグループの割り当て .....	47
を使用してセキュリティグループを割り当てる AWS CLI .....	48
ネットワーク ACL .....	48
カスタムドメイン名 .....	50
カスタムドメイン名とロードバランサー名の関連付け .....	51
ロードバランサーの Route 53 DNS フェイルオーバーを使用する .....	51
ドメイン名とロードバランサーの関連付けの解除 .....	52
リスナー .....	53
プロトコル .....	54
TCP/SSL プロトコル .....	54
HTTP/HTTPS プロトコル .....	55
HTTPS/SSL リスナー .....	55
SSL サーバー証明書 .....	55
SSL ネゴシエーション .....	56
バックエンドサーバー認証 .....	56
リスナー設定 .....	56
X-Forwarded ヘッダー .....	59
X-Forwarded-For .....	60
X-Forwarded-Proto .....	60
X-Forwarded-Port .....	61
HTTPS リスナー .....	62
SSL/TLS 証明書 .....	63

を使用して SSL/TLS 証明書を作成またはインポートする AWS Certificate Manager .....	64
IAM を使用する SSL/TLS 証明書のインポート .....	64
SSL ネゴシエーション設定 .....	64
セキュリティポリシー .....	65
SSL プロトコル .....	65
サーバーの優先順位 .....	66
SSL 暗号 .....	66
バックエンド接続用の暗号スイート .....	70
事前定義された SSL セキュリティポリシー .....	71
ポリシー別のプロトコル .....	72
ポリシー別の暗号 .....	73
暗号別のポリシー .....	77
HTTPS ロードバランサーの作成 .....	82
前提条件 .....	83
コンソールを使用した HTTPS ロードバランサーの作成 .....	84
を使用して HTTPS ロードバランサーを作成する AWS CLI .....	88
HTTPS リスナーを設定する .....	100
前提条件 .....	100
コンソールを使用した HTTPS リスナーの追加 .....	100
を使用して HTTPS リスナーを追加する AWS CLI .....	102
SSL 証明書の置き換え .....	104
コンソールを使用した SSL 証明書の置き換え .....	105
を使用して SSL 証明書を置き換える AWS CLI .....	106
SSL ネゴシエーション設定の更新 .....	107
コンソールを使用した SSL ネゴシエーション設定の更新 .....	107
を使用して SSL ネゴシエーション設定を更新する AWS CLI .....	108
登録済みインスタンス .....	113
インスタンスのベストプラクティス .....	113
VPC の推奨事項 .....	114
ロードバランサーにインスタンスを登録するには .....	114
インスタンスの登録 .....	115
ロードバランサーに登録されているインスタンスの表示 .....	116
インスタンスが登録されているロードバランサーの確認 .....	117
インスタンスの登録解除 .....	117
ヘルスチェック .....	118
ヘルスチェックの設定 .....	119

ヘルスチェックの設定の更新 .....	121
インスタンスのヘルスの確認 .....	122
ヘルスチェックのトラブルシューティング .....	122
セキュリティグループ .....	123
ネットワーク ACL .....	123
ロードバランサーのモニタリング .....	125
CloudWatch メトリクス .....	125
Classic Load Balancer のメトリクス .....	126
Classic Load Balancer のメトリクスディメンション .....	136
Classic Load Balancer メトリクスの統計 .....	136
ロードバランサーの CloudWatch メトリクスの表示 .....	137
アクセスログ .....	139
アクセスログファイル .....	140
アクセスログのエントリ .....	142
アクセスログの処理 .....	147
アクセスログの有効化 .....	147
アクセスログの無効化 .....	154
ロードバランサーのトラブルシューティング .....	156
API エラー .....	158
CertificateNotFound: 未定義 .....	158
OutOfService: 一時的なエラーの発生 .....	158
HTTP エラー .....	159
HTTP 400: BAD_REQUEST .....	160
HTTP 405: METHOD_NOT_ALLOWED .....	160
HTTP 408: Request timeout .....	160
HTTP 502: Bad gateway .....	161
HTTP 503: Service Unavailable .....	161
HTTP 504: Gateway Timeout .....	161
レスポンスコードのメトリクス .....	162
HTTPCode_ELB_4XX .....	163
HTTPCode_ELB_5XX .....	163
HTTPCode_Backend_2XX .....	163
HTTPCode_Backend_3XX .....	163
HTTPCode_Backend_4XX .....	163
HTTPCode_Backend_5XX .....	164
ヘルスチェック .....	164

ヘルスチェックのターゲットページのエラー .....	165
インスタンスへの接続がタイムアウトした .....	165
パブリックキー認証が失敗する .....	166
インスタンスがロードバランサーからのトラフィックを受信しない .....	167
インスタンスのポートが開いていない .....	167
Auto Scaling グループのインスタンスが ELB ヘルスチェックに失敗する .....	168
クライアントの接続 .....	168
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 .....	168
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 .....	169
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 .....	169
インスタンス登録 .....	170
EC2 インスタンスの登録に時間がかかりすぎる .....	170
有料 AMI から起動したインスタンスを登録できない .....	170
クォータ .....	171
ドキュメント履歴 .....	172
.....	clxxx

# 「Classic Load Balancer とは？」

## Note

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

Elastic Load Balancing 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ルリングします。また、大半のワークロードに合わせて自動的にスケールできます。

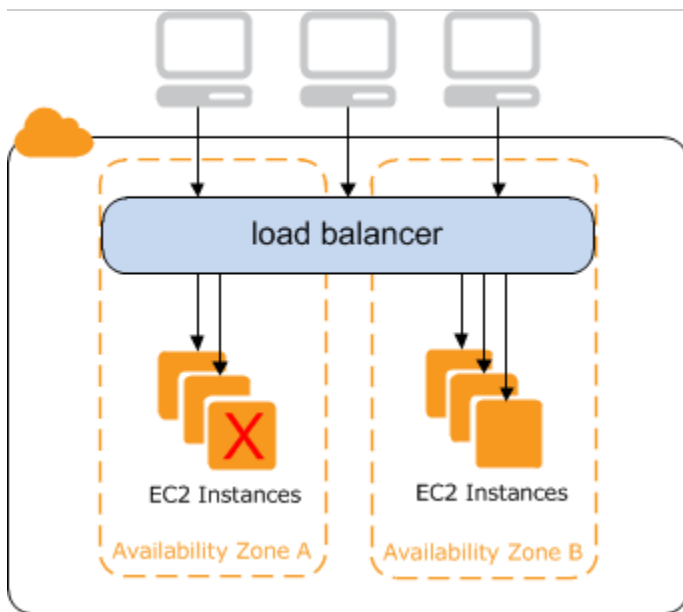
## Classic Load Balancer の概要

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 EC2 インスタンス間で分散します。これにより、アプリケーションの耐障害性が向上します。Elastic Load Balancing は異常なインスタンスを検出し、トラフィックを正常なインスタンスにのみルーティングします。

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。これにより、アプリケーションの可用性が向上します。アプリケーションへのリクエストの流れを中断することなく、ニーズの変化に応じてロードバランサーに対してインスタンス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ルリングします。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ルリングが可能です。

リスナーは、ユーザーが設定する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確認し、ユーザーが設定するプロトコルとポート番号を使用して、1つ以上の登録されたインスタンスにリクエストを転送します。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。

ヘルスチェックを設定して登録したインスタンスのヘルス状態をモニタリングすると、ロードバランサーから正常なインスタンスにのみリクエストを送信できます。



登録されたインスタスが各アベイラビリティゾーンのリクエストロードを処理できるようにするには、ロードバランサーに登録されている各アベイラビリティゾーンにほぼ同じ数のインスタスを維持することが重要です。たとえば、アベイラビリティゾーン us-west-2a に 10 個のインスタンス、アベイラビリティゾーン us-west-2b に 2 個のインスタンスがある場合、リクエストは 2 つのアベイラビリティゾーンに均等に分散されます。その結果、us-west-2b 内の 2 個のインスタンスは、us-west-2a 内の 10 個のインスタンスと同じ量のトラフィックを処理します。代わりに、アベイラビリティゾーンごとに 6 つのインスタスが必要です。

デフォルトでは、ロードバランサーは、ロードバランサーに対して有効にするアベイラビリティゾーン間で均等にトラフィックを分散します。有効にされたすべてのアベイラビリティゾーンのすべての登録されたインスタス間でトラフィックを分散するには、ロードバランサーでクロスゾーン負荷分散を有効にします。ただし、耐障害性を高めるために各アベイラビリティゾーンにおよそ等しい数のインスタスを維持することをお勧めします。

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。

## 利点

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

- TCP および SSL リスナーのサポート
- アプリケーション生成のクッキーを使用したスティッキーセッションのサポート

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。

## 開始方法

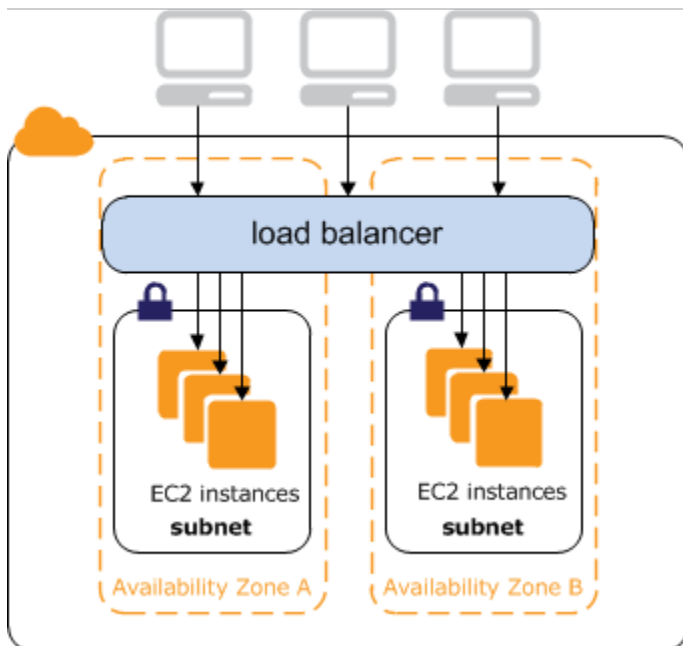
- Classic Load Balancer を作成し、それに EC2 インスタンスを登録する方法については、「[インターネット向け Classic Load Balancer を作成する](#)」を参照してください。
- HTTPS ロードバランサーを作成し、ロードバランサーに EC2 インスタンスを登録する方法を確認するには、「[HTTPS リスナーを使用する Classic Load Balancer の作成](#)」を参照してください。
- Classic Load Balancer でサポートされているさまざまな機能の使用方法については、「[Classic Load Balancer の設定](#)」を参照してください。

## 料金

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

# インターネット向け Classic Load Balancers

Classic Load Balancer を作成するとき、内部向けロードバランサーにすることも、インターネット向けロードバランサーにすることもできます。インターネット向けのロードバランサーには、パブリックに解決可能な DNS 名があるため、クライアントからのリクエストをインターネット経由で、ロードバランサーに登録された EC2 インスタンスにルーティングできます。



内部ロードバランサーの DNS 名は、ノードのプライベート IP アドレスにパブリックに解決可能です。そのため、内部向けロードバランサーは、ロードバランサー用に VPC へのアクセス権を持つクライアントからのみ、リクエストをルーティングできます。詳細については、「[内部ロードバランサー](#)」を参照してください。

## 内容

- [ロードバランサーのパブリック DNS 名](#)
- [インターネット向け Classic Load Balancer を作成する](#)

## ロードバランサーのパブリック DNS 名

ロードバランサーが作成されると、そのロードバランサーはパブリック DNS 名を受け取り、クライアントはそのパブリック DNS 名を使用してリクエストを送信できます。ロードバランサーの DNS 名は、DNS サーバーによってロードバランサーのロードバランサーノードのパブリック IP アドレス

に解決されます。各ロードバランサーノードはプライベート IP アドレスを使用してバックエンドインスタンスに接続されます。

コンソールに、次の形式でパブリック DNS 名が表示されます。

```
name-1234567890.region.elb.amazonaws.com
```

## インターネット向け Classic Load Balancer を作成する

ロードバランサーを作成する際には、リスナーとヘルスチェックを設定し、バックエンドインスタンスを登録します。リスナーを設定するには、フロントエンド (クライアントからロードバランサー) 接続用のプロトコルとポート、およびバックエンド (ロードバランサーからバックエンドインスタンス) 接続用のプロトコルとポートを指定します。ロードバランサーに対して複数のリスナーを設定できます。

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 [AWS マネジメントコンソール Classic Load Balancer](#) を実際に紹介します。ここでは、パブリック HTTP トラフィックを受信し、このトラフィックを EC2 インスタンスに送信するロードバランサーを作成します。

HTTPS リスナーを使用してロードバランサーを作成するには、「[HTTPS リスナーを使用する Classic Load Balancer の作成](#)」を参照してください。

### タスク

- [\[開始する前に\]](#)
- [を使用して Classic Load Balancer を作成する AWS マネジメントコンソール](#)

### [開始する前に]

- 仮想プライベートクラウド (VPC) を作成します。詳細については、「[VPC の推奨事項](#)」を参照してください。
- ロードバランサーに登録する 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。
- Apache や Internet Information Services (IIS) などのウェブサーバーを各インスタンスにインストールし、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その DNS 名を入力して、サーバーのデフォルトページがブラウザに表示されることを確認します。

# を使用して Classic Load Balancer を作成する AWS マネジメントコンソール

次の手順に従って、Classic Load Balancer を作成します。名前やスキームなど、ロードバランサーの基本的な設定情報を指定します。次に、ネットワークと、インスタンスにトラフィックをルーティングするリスナーに関する情報を指定します。

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

1. Amazon EC2 コンソールの <https://console.aws.amazon.com/ec2/> を開いてください。
2. ナビゲーションバーで、ロードバランサーのリージョンを選択します。EC2 インスタンス用に選択したの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 [ロードバランシング] で [ロードバランサー] を選択します。
4. [Create Load Balancer] を選択します。
5. [Classic Load Balancer] セクションを展開し、[作成] を選択します。

## 6. 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。

この Classic Load Balancer の名前は、リージョン内にある Classic Load Balancer の中で一意にする必要があります。使用可能なのは最大 32 文字で、英数字とハイフンのみ使用できます。また、先頭と末尾にハイフンを使用することはできません。

- b. [スキーム] で、[インターネットに接続] を選択します。

## 7. ネットワークマッピング

- a. [VPC] で、インスタンス用に選択したのと同じ VPC を選択します。
- b. マッピングの場合、最初にアベイラビリティーゾーンを選択し、次に利用可能なサブネットからパブリックサブネットを選択します。アベイラビリティーゾーンごとに選択できるサブネットは 1 つだけです。ロードバランサーの可用性を高めるには、複数のアベイラビリティーゾーンからサブネットを選択します。

## 8. セキュリティグループ

- [セキュリティグループ] では、ポート 80 で必要な HTTP トラフィックを許可するように設定されている既存のセキュリティグループを選択します。

## 9. リスナーとルーティング

- a. リスナーの場合は、プロトコルが HTTP でポートが 80 であることを確認してください。

- b. インスタンスの場合は、プロトコルが HTTP、ポートが 80 であることを確認してください。

## 10. ヘルスチェック

- a. Ping プロトコルの場合は、プロトコルが HTTP であることを確認してください。
- b. Ping ポートの場合は、ポートが 80 であることを確認してください。
- c. Ping パスの場合は、パスが / であることを確認してください。
- d. ヘルスチェックの詳細設定には、デフォルト値を使用します。

## 11. インスタンス

- a. [インスタンスの追加] を選択して、インスタンスの選択画面を表示します。
- b. [使用可能なインスタンス] で、現在のネットワーク設定に基づいて、ロードバランサーで使用できる現在のインスタンスから選択できます。
- c. 選択内容に問題がなければ、[確認] を選択して、ロードバランサーに登録するインスタンスを追加します。

## 12. 属性

- [クロスゾーン負荷分散の有効化]、[Connection Draining の有効化]、および [タイムアウト (ドレーニング間隔)] はデフォルト値のままにします。

## 13. ロードバランサータグ (オプション)

- a. [キー] フィールドは必須です。
- b. [値] フィールドはオプションです。
- c. 別のタグを追加するには、[新しいタグを追加] を選択し、[キー] フィールドに値を入力し、オプションで [値] フィールドに値を入力します。
- d. 既存のタグを削除するには、削除したいタグの横にある [削除] を選択します。

## 14. 概要と作成

- a. 設定を変更する必要がある場合は、変更する必要がある設定の横にある [編集] を選択します。
- b. 概要に表示されているすべての設定に問題がなければ、[ロードバランサーの作成] を選択してロードバランサーの作成を開始します。
- c. 最後の作成ページで、[ロードバランサーを表示] を選択して、Amazon EC2 コンソールでロードバランサーを表示します。

## 15. 検証

- a. 新しいロードバランサーを選択します。
- b. [ターゲットインスタンス] タブで、[ヘルスステータス] 列を確認します。少なくとも 1 つの EC2 インスタンスの状態が稼動中であれば、ロードバランサーをテストできます。
- c. [詳細] セクションで、ロードバランサー DNS 名 をコピーします。これは my-load-balancer-1234567890.us-east-1.elb.amazonaws.com のようになります。
- d. ロードバランサーの DNS 名を、パブリック 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貼り付けます。ロードバランサーが正しく機能している場合は、サーバーのデフォルトページが表示されます。

## 16. 削除 (オプション)

- a. ロードバランサーをポイントするドメインの CNAME 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。
- b. Amazon EC2 コンソールの <https://console.aws.amazon.com/ec2/> を開いてください。
- c. ロードバランサーを選択します。
- d. [アクション]、[ロードバランサーを削除] の順に選択します。
- e. 確認を求められたら、「confirm」と入力し、[削除] を選択します。
- f. ロードバランサーを削除しても、そのロードバランサーに登録された EC2 インスタンスは引き続き実行されます。実行が継続される分単位または時間単位でお支払いいただきます。EC2 インスタンスがなくなっただけの場合は、追加料金が発生しないように停止または終了できます。

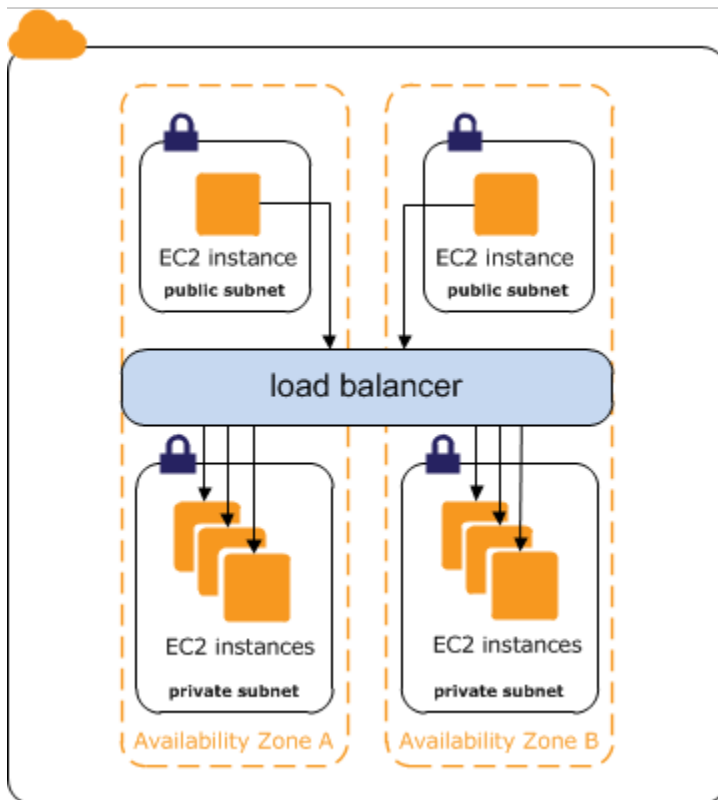
## 内部 Classic Load Balancers

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

インターネット向けロードバランサーのノードにはパブリック IP アドレスが必要です。インターネット向けロードバランサーの DNS 名は、ノードのパブリック IP アドレスにパブリックに解決可能です。したがって、インターネット向けロードバランサーは、クライアントからインターネット経由でリクエストをルーティングできます。詳細については、「[インターネット向け Classic Load Balancers](#)」を参照してください。

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

アプリケーションに複数の層 (インターネットに接続する必要があるウェブサーバーや、ウェブサーバーにのみ接続されているデータベースサーバーなど) がある場合、内部ロードバランサーとインターネット向けロードバランサーの両方を使用するアーキテクチャを設計できます。インターネット接続ロードバランサーを作成し、そこにウェブサーバーを登録します。内部ロードバランサーを作成し、そこにデータベースサーバーを登録します。ウェブサーバーは、インターネット接続ロードバランサーからリクエストを受け取り、データベースサーバー用のリクエストを内部ロードバランサーに送信します。データベースサーバーは、内部ロードバランサーからリクエストを受け取ります。



## 目次

- [ロードバランサーのパブリック DNS 名](#)
- [内部 Classic Load Balancer の作成](#)

## ロードバランサーのパブリック DNS 名

内部ロードバランサーは、作成されると、以下の形式でパブリック DNS 名を受け取ります。

```
internal-name-123456789.region.elb.amazonaws.com
```

ロードバランサーの DNS 名は、DNS サーバーによって内部ロードバランサーのロードバランサーノードのプライベート IP アドレスに解決されます。各ロードバランサーノードは、Elastic Network Interface を使用して、バックエンドインスタンスのプライベート IP アドレスに接続されます。クロスゾーン負荷分散が有効になっている場合、アベイラビリティゾーンにかかわらず、各ノードは各バックエンドインスタンスに接続されます。それ以外の場合、各ノードは、それぞれのアベイラビリティゾーン内にあるインスタンスに接続されるのみです。

## 内部 Classic Load Balancer の作成

内部向けロードバランサーを作成して、ロードバランサー用に VPC へのアクセス権を持つクライアントから EC2 インスタンスにトラフィックを分散できます。

### 内容

- [前提条件](#)
- [コンソールを使用した内部ロードバランサーの作成](#)
- [を使用して内部ロードバランサーを作成する AWS CLI](#)

### 前提条件

- ロードバランサー用の VPC を作成していない場合は、作業を開始する前にその VPC を作成する必要があります。詳細については、「[VPC の推奨事項](#)」を参照してください。
- 内部向けロードバランサーに登録する EC2 インスタンスを起動します。それらのインスタンスは、ロードバランサー用の VPC 内のプライベートサブネットで起動してください。

### コンソールを使用した内部ロードバランサーの作成

次の手順に従って、内部 Classic Load Balancer を作成します。名前やスキームなど、ロードバランサーの基本的な設定情報を指定します。次に、ネットワークと、インスタンスにトラフィックをルーティングするリスナーに関する情報を指定します。

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

1. Amazon EC2 コンソールの <https://console.aws.amazon.com/ec2/> を開いてください。
2. ナビゲーションバーで、ロードバランサーのリージョンを選択します。EC2 インスタンス用に選択したの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 [ロードバランシング] で [ロードバランサー] を選択します。
4. [Create Load Balancer] を選択します。
5. [Classic Load Balancer] セクションを展開し、[作成] を選択します。
6. 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。

この Classic Load Balancer の名前は、リージョン内にある Classic Load Balancer の中で一意にする必要があります。使用可能なのは最大 32 文字で、英数字とハイフンのみ使用できます。また、先頭と末尾にハイフンを使用することはできません。

b. [スキーム] で、[内部] を選択します。

## 7. ネットワークマッピング

a. [VPC] で、インスタンス用に選択したのと同じ VPC を選択します。

b. マッピング の場合は、最初にアベイラビリティーゾーンを選択し、次に使用可能なサブネットからサブネットを選択します。アベイラビリティーゾーンごとに選択できるサブネットは 1 つだけです。ロードバランサーの可用性を高めるには、複数のアベイラビリティーゾーンからサブネットを選択します。

8. [セキュリティグループ] では、ポート 80 で必要な HTTP トラフィックを許可するように設定されている既存のセキュリティグループを選択します。または、アプリケーションが異なるプロトコルとポートを使用している場合は、新しいセキュリティグループを作成できます。

## 9. リスナーとルーティング

a. リスナーの場合は、プロトコルが HTTP でポートが 80 であることを確認してください。

b. インスタンスの場合は、プロトコルが HTTP、ポートが 80 であることを確認してください。

## 10. ヘルスチェック

a. [Ping プロトコル] の場合、デフォルトは HTTP です。

b. [Ping ポート] の場合、デフォルトは 80 です。

c. [Ping パス] のデフォルトは / です。

d. [ヘルスチェックの詳細設定] では、デフォルト値を使用するか、アプリケーションに固有の値を入力します。

## 11. インスタンス

a. [インスタンスの追加] を選択して、インスタンスの選択画面を表示します。

b. [利用可能なインスタンス] で、前に選択したネットワーク設定に基づいて、ロードバランサーで利用可能な現在のインスタンスから選択できます。

c. 選択内容に問題がなければ、[確認] を選択して、ロードバランサーに登録するインスタンスを追加します。

## 12. 属性

- [クロスゾーン負荷分散の有効化]、[Connection Draining の有効化]、および [タイムアウト (ドレーニング間隔)] はデフォルト値のままにします。

### 13. ロードバランサータグ (オプション)

- [キー] フィールドは必須です。
- [値] フィールドはオプションです。
- 別のタグを追加するには、[新しいタグを追加]を選択し、[キー] フィールドに値を入力し、オプションで [値] フィールドに値を入力します。
- 既存のタグを削除するには、削除したいタグの横にある [削除] を選択します。

### 14. 概要と作成

- 設定を変更する必要がある場合は、変更する必要がある設定の横にある [編集] を選択します。
- 概要に表示されているすべての設定に問題がなければ、[ロードバランサーの作成] を選択してロードバランサーの作成を開始します。
- 最後の作成ページで、[ロードバランサーを表示] を選択して、Amazon EC2 コンソールでロードバランサーを表示します。

### 15. 検証

- 新しいロードバランサーを選択します。
- [ターゲットインスタンス] タブで、[ヘルスステータス] 列を確認します。少なくとも 1 つの EC2 インスタンスの状態が稼動中であれば、ロードバランサーをテストできます。
- [詳細] セクションで、ロードバランサー DNS 名 をコピーします。これは my-load-balancer-1234567890.us-east-1.elb.amazonaws.com のようになります。
- ロードバランサーの DNS 名を、パブリック 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貼り付けます。ロードバランサーが正しく機能している場合は、サーバーのデフォルトページが表示されます。

### 16. 削除 (オプション)

- ロードバランサーをポイントするドメインの CNAME 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。
- Amazon EC2 コンソールの <https://console.aws.amazon.com/ec2/> を開いてください。
- ロードバランサーを選択します。
- [アクション]、[ロードバランサーを削除] の順に選択します。

- e. 確認を求められたら、「confirm」と入力し、[削除]を選択します。
- f. ロードバランサーを削除しても、そのロードバランサーに登録された EC2 インスタンスは引き続き実行されます。実行が継続される分単位または時間単位でお支払いいただきます。EC2 インスタンスがなくなっただけの場合は、追加料金が発生しないように停止または終了できます。

## を使用して内部ロードバランサーを作成する AWS CLI

Elastic Load Balancing はデフォルトで、インターネット向けロードバランサーを作成します。次の手順に従って内部向けロードバランサーを作成し、EC2 インスタンスを新しく作成した内部向けロードバランサーに登録します。

内部向けロードバランサーを作成するには

1. 次のように `--scheme` オプションに `internal` を設定しながら、[create-load-balancer](#) コマンドを使用します。

```
aws elb create-load-balancer --load-balancer-name my-internal-loadbalancer --  
listeners Protocol=HTTP,LoadBalancerPort=80,InstanceProtocol=HTTP,InstancePort=80  
--subnets subnet-4e05f721 --scheme internal --security-groups sg-b9ffedd5
```

以下に、応答の例を示します。名前によって、これが内部向けロードバランサーであることが示されている点に注意してください。

```
{  
  "DNSName": "internal-my-internal-loadbalancer-786501203.us-  
west-2.elb.amazonaws.com"  
}
```

2. 次の [register-instances-with-load-balancer](#) コマンドを使用して、インスタンスを追加します。

```
aws elb register-instances-with-load-balancer --load-balancer-name my-internal-  
loadbalancer --instances i-4f8cf126 i-0bb7ca62
```

以下に、応答の例を示します。

```
{  
  "Instances": [  
    {
```

```
        "InstanceId": "i-4f8cf126"
      },
      {
        "InstanceId": "i-0bb7ca62"
      }
    ]
  }
}
```

3. (オプション) 次の [describe-load-balancers](#) コマンドを使用して、内部向けロードバランサーを検証します。

```
aws elb describe-load-balancers --load-balancer-name my-internal-loadbalancer
```

応答には、DNSName フィールドと Scheme フィールドが含まれており、これが内部ロードバランサーであることを示しています。

```
{
  "LoadBalancerDescriptions": [
    {
      ...
      "DNSName": "internal-my-internal-loadbalancer-1234567890.us-
west-2.elb.amazonaws.com",
      "SecurityGroups": [
        "sg-b9ffedd5"
      ],
      "Policies": {
        "LBCookieStickinessPolicies": [],
        "AppCookieStickinessPolicies": [],
        "OtherPolicies": []
      },
      "LoadBalancerName": "my-internal-loadbalancer",
      "CreatedTime": "2014-05-22T20:32:19.920Z",
      "AvailabilityZones": [
        "us-west-2a"
      ],
      "Scheme": "internal",
      ...
    }
  ]
}
```

# Classic Load Balancer の設定

Classic Load Balancer を作成したら、設定を変更できます。例えば、ロードバランサーの属性、サブネット、セキュリティグループを更新できます。

ロードバランサーの属性

## Connection Draining

有効になっている場合、ロードバランサーは、登録解除されたインスタンスや異常なインスタンスからトラフィックを移動する前に、既存のリクエストを完了させることができます。

## クロスゾーンロードバランサー

有効になっている場合、ロードバランサーは、アベイラビリティゾーンに関係なく、リクエストトラフィックをすべてのインスタンスに均等にルーティングします。

## Desync 軽減モード

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。デフォルトは defensive です。

## アイドルタイムアウト

有効になっている場合、ロードバランサーは、指定された期間、接続をアイドル状態にしたままにします (データは接続を介して送信されません)。デフォルト値は 60 秒です。

## スティッキーセッション

Classic Load Balancer は、期間ベースのセッションとアプリケーションベースのセッションの両方の接続維持をサポートします。

ロードバランサーの詳細

## セキュリティグループ

ロードバランサーのセキュリティグループは、リスナーポートと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する必要があります。

## サブネット

ロードバランサーの機能は、追加のサブネットにまで拡大できます。

## [Proxy Protocol](#)

有効にすると、インスタンスに送信される接続情報を含むヘッダーが追加されます。

## [タグ](#)

タグを追加して、ロードバランサーを分類できます。

# Classic Load Balancer でのアイドル接続のタイムアウト設定

クライアントが Classic Load Balancer を通じて行うリクエストごとに、ロードバランサーは 2 つの接続を維持します。フロントエンド接続は、クライアントとロードバランサーの間にあります。バックエンド接続は、ロードバランサーと登録済み EC2 インスタンスの間にあります。ロードバランサーには、その接続に適用される設定済みのアイドルタイムアウト期間があります。アイドルタイムアウトが経過するまでデータが送受信されなかった場合、ロードバランサーは接続を閉じます。ファイルのアップロードなどの長いオペレーションで、完了までの時間を確保するため、各アイドルタイムアウト期間が経過するまでに少なくとも 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の長さを増やします。

HTTP および HTTPS リスナーを使用する場合は、インスタンスで HTTP キープアライブのオプションを有効にすることをお勧めします。インスタンスのウェブサーバー設定で キープアライブ を有効にできます。キープアライブを有効にすると、ロードバランサーはキープアライブタイムアウトが期限切れになるまでバックエンド接続を再利用できます。ロードバランサーでインスタンスへの接続を確実に閉じるには、HTTP キープアライブ時間の設定をロードバランサーのアイドルタイムアウト設定よりも大きい値に設定する必要があります。

TCP キープアライブのプロブはペイロードのデータを送信しないため、ロードバランサーが接続を終了する妨げにはならないことに注意してください。

## 目次

- [コンソールを使用したアイドルタイムアウトの設定](#)
- [を使用してアイドルタイムアウトを設定する AWS CLI](#)

## コンソールを使用したアイドルタイムアウトの設定

Elastic Load Balancing では、ロードバランサーのアイドルタイムアウトはデフォルトで 60 秒に設定されています。アイドルタイムアウトに別の値を設定するには、次の手順を使用します。

コンソールを使用してロードバランサーのアイドルタイムアウトを設定するには

1. Amazon EC2 コンソールの <https://console.aws.amazon.com/ec2/> を開いてください。
2. ナビゲーションペインの [ロードバランシング] で [ロードバランサー] を選択します。
3. 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. [ロードバランサー属性の編集] ページの [トラフィックの設定] セクションで、[アイドルタイムアウト] の値を入力します。アイドルタイムアウトの範囲は 1~4,000 秒です。
6. [Save changes] (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を設定する AWS CLI

次の [modify-load-balancer-attributes](#) コマンドを使用して、ロードバランサーのアイドルタイムアウトを設定します。

```
aws elb modify-load-balancer-attributes --load-balancer-name my-loadbalancer --load-balancer-attributes "{\"ConnectionSettings\":{\"IdleTimeout\":30}}"
```

以下に、応答の例を示します。

```
{
  "LoadBalancerAttributes": {
    "ConnectionSettings": {
      "IdleTimeout": 30
    }
  },
  "LoadBalancerName": "my-loadbalancer"
}
```

## Classic Load Balancer のクロスゾーン負荷分散の設定

クロスゾーン負荷分散を使用すると、Classic Load Balancer の各ロードバランサーノードは、有効なすべてのアベイラビリティゾーンの登録されたインスタンスにリクエストを均等に分散します。クロスゾーン負荷分散が無効の場合は、各ロードバランサーノードは、そのアベイラビリティゾーンの登録されたインスタンスにのみリクエストを均等に分散します。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により、有効な各アベイラビリティゾーンに等しい数のインスタンスを維持する必要性が軽減され、1つ以上のインスタンスの消失を処理するアプリケーションの能力が向上します。ただし、耐障害性を高めるために有効な各アベイラビリティゾーンにおよそ等しい数のインスタンスを維持することをお勧めします。

クライアントが DNS ルックアップをキャッシュする環境では、着信リクエストがいずれかのアベイラビリティゾーンを優先する場合があります。クロスゾーン負荷分散を使用すると、リクエスト負荷の不均衡がリージョン内のすべての利用可能なインスタンスに分散されて、不適切に動作しているクライアントによる影響が軽減されます。

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

## 目次

- [クロスゾーン負荷分散の有効化](#)
- [クロスゾーン負荷分散の無効化](#)

## クロスゾーン負荷分散の有効化

Classic Load Balancer のクロスゾーン負荷分散は、任意のタイミングで有効化できます。

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 [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. [ロードバランサー属性の編集] ページの [アベイラビリティゾーンのルーティング設定] セクションで、[クロスゾーン負荷分散] を有効にします。
6. [Save changes] (変更の保存) をクリックします。

を使用してクロスゾーン負荷分散を有効にするには AWS CLI

1. 次の [modify-load-balancer-attributes](#) コマンドを使用して、ロードバランサーの `CrossZoneLoadBalancing` 属性を `true` に設定します。

```
aws elb modify-load-balancer-attributes --load-balancer-name my-loadbalancer --load-balancer-attributes "{\"CrossZoneLoadBalancing\":{\"Enabled\":true}}"
```

以下に、応答の例を示します。

```
{
  "LoadBalancerAttributes": {
    "CrossZoneLoadBalancing": {
      "Enabled": true
    }
  },
  "LoadBalancerName": "my-loadbalancer"
}
```

2. (省略可) 次の [describe-load-balancer-attributes](#) コマンドを使用して、ロードバランサーのクロスゾーン負荷分散が有効になっていることを確認します。

```
aws elb describe-load-balancer-attributes --load-balancer-name my-loadbalancer
```

以下に、応答の例を示します。

```
{
  "LoadBalancerAttributes": {
    "ConnectionDraining": {
      "Enabled": false,
      "Timeout": 300
    },
    "CrossZoneLoadBalancing": {
      "Enabled": true
    },
    "ConnectionSettings": {
      "IdleTimeout": 60
    },
    "AccessLog": {
      "Enabled": false
    }
  }
}
```

```
}  
}
```

## クロスゾーン負荷分散の無効化

ロードバランサーに対してクロスゾーン負荷分散オプションをいつでも無効にすることができます。

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. [ロードバランサー属性の編集] ページの [アベイラビリティゾーンのルーティング設定] セクションで、[クロスゾーン負荷分散] を無効にします。
6. [Save changes] (変更の保存) をクリックします。

クロスゾーン負荷分散を無効にするには、ロードバランサーの `CrossZoneLoadBalancing` 属性を `false` に設定します。

を使用してクロスゾーン負荷分散を無効にするには AWS CLI

1. 次の [modify-load-balancer-attributes](#) コマンドを使用します。

```
aws elb modify-load-balancer-attributes --load-balancer-name my-loadbalancer --  
load-balancer-attributes "{\"CrossZoneLoadBalancing\":{\"Enabled\":false}}"
```

以下に、応答の例を示します。

```
{  
  "LoadBalancerAttributes": {  
    "CrossZoneLoadBalancing": {  
      "Enabled": false  
    }  
  },  
  "LoadBalancerName": "my-loadbalancer"  
}
```

2. (省略可) 次の `describe-load-balancer-attributes` コマンドを使用して、ロードバランサーのクロスゾーン負荷分散が無効になっていることを確認します。

```
aws elb describe-load-balancer-attributes --load-balancer-name my-loadbalancer
```

以下に、応答の例を示します。

```
{
  "LoadBalancerAttributes": {
    "ConnectionDraining": {
      "Enabled": false,
      "Timeout": 300
    },
    "CrossZoneLoadBalancing": {
      "Enabled": false
    },
    "ConnectionSettings": {
      "IdleTimeout": 60
    },
    "AccessLog": {
      "Enabled": false
    }
  }
}
```

## Classic Load Balancer の Connection Draining の設定

Classic Load Balancer が、既存の接続を開いたまま、登録解除中のインスタンスまたは異常の発生したインスタンスにリクエストを送信しないようにするには、Connection Draining を使用します。これにより、ロードバランサーは、登録解除中のインスタンスまたは異常の発生したインスタンスに対する未処理のリクエストを完了できます。

Connection Draining を有効にするとき、ロードバランサーがインスタンスの登録解除を報告するまで接続を保持する最大時間を指定できます。最大タイムアウト値は 1 ~ 3,600 秒の間で設定できます (デフォルトは 300 秒です)。最大制限時間に達すると、ロードバランサーは登録解除中のインスタンスへの接続を強制的に閉じます。

登録解除するインスタンスに未処理のリクエストやアクティブな接続がない場合、Elastic Load Balancing は直ちに登録解除プロセスを完了します。

未処理のリクエストの処理中、ロードバランサーは登録解除中のインスタンスの状態を `InService: Instance deregistration currently in progress` と報告します。登録解除中のインスタンスが未処理のすべてのリクエストの処理を完了するか、最大制限時間に達すると、ロードバランサーは、インスタンスの状態を `OutOfService: Instance is not currently registered with the LoadBalancer` と報告します。

インスタンスで異常が発生すると、ロードバランサーはインスタンスの状態を `OutOfService` と報告します。異常の発生したインスタンスに対する未処理のリクエストがある場合、それらは完了されません。異常の発生したインスタンスへの接続には、最大制限時間は適用されません。

インスタンスが Auto Scaling グループに属しており、ロードバランサーの Connection Draining が有効になっている場合、Auto Scaling は、スケーリングイベントまたはヘルスチェック交換によりインスタンスを終了する前に、未処理のリクエストが完了するか最大制限時間に達するまで待機します。

ロードバランサーが登録解除中のインスタンスまたは異常の発生したインスタンスへの接続をただちに閉じるように設定するには、Connection Draining を無効にします。Connection Draining が無効になっていると、登録解除中のインスタンスまたは異常の発生したインスタンスに対する未処理のリクエストは完了されません。

## 目次

- [Connection Drainingの有効化](#)
- [Connection Drainingの無効化](#)

## Connection Drainingの有効化

Connection Draining は、ロードバランサーに対していつでも有効にできます。

コンソールを使用して Connection Draining 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 [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. [ロードバランサー属性の編集] ページの [トラフィックの設定] セクションで、[Connection Draining の有効化] を選択します。
6. (オプション) [タイムアウト (ドレーニング間隔)] に、1~3,600 秒の値を入力します。それ以外の場合は、デフォルトの 300 秒が使用されます。

7. [Save changes] (変更の保存) をクリックします。

を使用して接続ドレーニングを有効にするには AWS CLI

次の [modify-load-balancer-attributes](#) コマンドを使用します。

```
aws elb modify-load-balancer-attributes --load-balancer-name my-loadbalancer --load-balancer-attributes "{\"ConnectionDraining\":{\"Enabled\":true,\"Timeout\":300}}"
```

以下に、応答の例を示します。

```
{
  "LoadBalancerAttributes": {
    "ConnectionDraining": {
      "Enabled": true,
      "Timeout": 300
    }
  },
  "LoadBalancerName": "my-loadbalancer"
}
```

## Connection Drainingの無効化

Connection Draining は、ロードバランサーに対していつでも無効にできます。

コンソールを使用して Connection Draining 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. [ロードバランサー属性の編集] ページの [トラフィックの設定] セクションで、[Connection Draining の有効化] の選択を解除します。
6. [Save changes] (変更の保存) をクリックします。

を使用して接続ドレーニングを無効にするには AWS CLI

次の [modify-load-balancer-attributes](#) コマンドを使用します。

```
aws elb modify-load-balancer-attributes --load-balancer-name my-loadbalancer --load-balancer-attributes "{\"ConnectionDraining\":{\"Enabled\":false}}"
```

以下に、応答の例を示します。

```
{
  "LoadBalancerAttributes": {
    "ConnectionDraining": {
      "Enabled": false,
      "Timeout": 300
    }
  },
  "LoadBalancerName": "my-loadbalancer"
}
```

## Classic Load Balancer のステイッキーセッションの設定

Classic Load Balancer のデフォルトでは、各リクエストは登録済みインスタンスに対し、負荷が最小になるように個別にルーティングされます。ステイ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る) を使用することによって、ロードバランサーがユーザーのセッションを特定のアプリケーションインスタンス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インスタンスに送信されます。

ステイ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インスタンス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、Elastic Load Balancing のセッション Cookie を設定し、アプリケーションのセッション Cookie で指定された維持期間を受け入れさせることができます。アプリケーションに独自のセッション Cookie がない場合は、Elastic Load Balancing を設定し、独自の維持期間を指定しながらセッション Cookie を作成させることができます。

Elastic Load Balancing は、AWSELB という名前の Cookie を作成し、セッションをインスタンスにマッピングするために使用します。

### 要件

- HTTP/HTTPS ロードバランサー。
- 各アベイラビリティ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。

## 互換性

- Cookie の path プロパティの RFC では、アンダースコアが許容されています。ただし、Elastic Load Balancing の URI ではアンダースコアは %5F としてエンコードされます。Internet Explorer 7 などの一部のブラウザでは、アンダースコアを %5F としてエンコードする URI が想定されているためです。現在使用されているブラウザに影響が及ぶ可能性があることを考慮し、Elastic Load Balancing では、引き続きアンダースコア文字を URI エンコードしています。例えば、Cookie にプロパティ path=/my\_path がある場合、Elastic Load Balancing は転送されたリクエスト内のこのプロパティを path=/my%5Fpath に変更します。
- 期間ベースのセッション維持 Cookie の secure フラグや HttpOnly フラグを設定することはできません。ただし、これらの Cookie に重要なデータは含まれません。アプリケーション制御によるセッション維持 Cookie に secure または HttpOnly フラグを設定した場合、AWSELB Cookie にも設定されます。
- アプリケーション クッキーの Set-Cookie フィールドで末尾にセミコロンがある場合、ロードバランサーはそのクッキーを無視します。

## 目次

- [期間ベースのセッションの維持](#)
- [アプリケーション制御によるセッションの維持](#)

## 期間ベースのセッションの維持

ロードバランサーは、各リスナーへのリクエストごとに特殊な Cookie である AWSELB を使用してインスタンスを追跡します。ロードバランサーがリクエストを受け取ると、まずこの Cookie がリクエスト内にあるかどうかを調べます。ある場合は、Cookie で指定されているインスタンスにリクエストが送信されます。Cookie がない場合は、ロードバランサーが既存の負荷分散アルゴリズムに基づいてインスタンスを選択します。同じユーザーの後続のリクエストをそのインスタンスにバインドするために Cookie が応答に挿入されます。スティッキーポリシー設定では、各 Cookie の有効期間を設定する Cookie 期限を定義します。ロードバランサーは、Cookie の有効期限を更新せず、Cookie を使用する前に失効しているかどうかを確認しません。クッキーが期限切れになった後、セッションはスティッキーではなくなります。クライアントは、Cookie の失効時に Cookie ストアから削除する必要があります。

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 は別の維持

Cookie として AWSELBCORS を作成します。この Cookie には、元の維持 Cookie と同じ情報に加えて 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。

インスタンスでエラーや異常が発生した場合、ロードバランサーはそのインスタンスへのリクエストのルーティングを停止し、既存の負荷分散アルゴリズムに基づいて新しい正常なインスタンスを選択します。リクエストは、Cookie がなく、セッションが維持されていない場合と同様に、新しいインスタンスにルーティングされます。

クライアントが異なるバックエンドポートを持つリスナーに切り替えた場合、維持設定は失われます。

コンソールを使用してロードバランサーの期間ベースのスティッキーセッションを有効にするには

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 [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. ロードバランサーの名前を選択して、その詳細ページを開きます。
4. [リスナー] タブで、[リスナーを追加] を選択します。
5. [リスナーを管理] ページで、更新するリスナーを見つけて、[Cookie の維持設定] で [編集] を選択します。
6. [Cookie の維持設定] ポップアップで、ロードバランサーによって生成] を選択します。
7. (オプション) [有効期間] に Cookie の有効期間を秒単位で入力します。有効期限を指定しない場合、スティッキーセッションはブラウザセッションが終わるまで保持されます。
8. [変更を保存] をクリックして、ポップアップウィンドウを閉じます。
9. 変更を保存を選択してロードバランサーの詳細ページに戻ります。

を使用してロードバランサーの期間ベースのスティッキーセッションを有効にするには AWS CLI

1. 次の [create-lb-cookie-stickiness-policy](#) コマンドを使用し、Cookie の有効期間を 60 秒に設定して、ロードバランサーが生成する Cookie の維持ポリシーを作成します。

```
aws elb create-lb-cookie-stickiness-policy --load-balancer-name my-loadbalancer --policy-name my-duration-cookie-policy --cookie-expiration-period 60
```

2. 次の [set-load-balancer-policies-of-listener](#) コマンドを使用して、特定のロードバランサーのセッション維持を有効にします。

```
aws elb set-load-balancer-policies-of-listener --load-balancer-name my-loadbalancer
--load-balancer-port 443 --policy-names my-duration-cookie-policy
```

**Note**

set-load-balancer-policies-of-listener コマンドは、指定されたロードバランサーのポートに関連付けられた現在の一連のポリシーを置き換えます。このコマンドを使用するたびに、--policy-names を指定して有効にするすべてのポリシーを一覧表示します。

- (オプション) 次の [describe-load-balancers](#) コマンドを使用して、ポリシーが有効化されていることを確認します。

```
aws elb describe-load-balancers --load-balancer-name my-loadbalancer
```

応答には、指定したポートのリスナーに対してポリシーが有効になっていることを示す次の情報が出力されます。

```
{
  "LoadBalancerDescriptions": [
    {
      ...
      "ListenerDescriptions": [
        {
          "Listener": {
            "InstancePort": 443,
            "SSLCertificateId": "arn:aws:iam::123456789012:server-
certificate/my-server-certificate",
            "LoadBalancerPort": 443,
            "Protocol": "HTTPS",
            "InstanceProtocol": "HTTPS"
          },
          "PolicyNames": [
            "my-duration-cookie-policy",
            "ELBSecurityPolicy-TLS-1-2-2017-01"
          ]
        },
        ...
      ],
    },
    ...
  ],
}
```

```
...
  "Policies": {
    "LBCookieStickinessPolicies": [
      {
        "PolicyName": "my-duration-cookie-policy",
        "CookieExpirationPeriod": 60
      }
    ],
    "AppCookieStickinessPolicies": [],
    "OtherPolicies": [
      "ELBSecurityPolicy-TLS-1-2-2017-01"
    ]
  },
  ...
}
]
```

## アプリケーション制御によるセッションの維持

ロードバランサーは、特殊な Cookie を使用して、最初のリクエストを処理したインスタンスにセッションを関連付けますが、ポリシー設定で指定されているアプリケーション Cookie の有効期間に従います。ロードバランサーは、アプリケーションの応答に新しいアプリケーション Cookie が含まれている場合のみ、新しい維持 Cookie を挿入します。ロードバランサー維持 Cookie はリクエストごとに更新されることはありません。アプリケーション Cookie が明示的に削除されるかまたは有効期限切れになると、新しいアプリケーション Cookie が発行されない限り、セッションは sticky ではなくなります。

### バックエンドインスタンスによって設定された属性

(path、port、domain、secure、httponly、discard、max-age、expires、version、comment、commenturl、および samesite) が Cookie でクライアントに送信されます。

インスタンスでエラーや異常が発生した場合、ロードバランサーはそのインスタンスへのリクエストのルーティングを停止し、既存の負荷分散アルゴリズムに基づいて新しい正常なインスタンスを選択します。ロードバランサーは、セッションが新しい正常なインスタンスで "維持" されていると見なし、エラーが発生したインスタンスが正常な状態に戻っても、継続して新しいインスタンスにリクエストをルーティングします。

コンソールを使用してアプリケーション制御によるセッション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 [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. ロードバランサーの名前を選択して、その詳細ページを開きます。
4. [リスナー] タブで、[リスナーを追加] を選択します。
5. [リスナーを管理] ページで、更新するリスナーを見つけて、[Cookie の維持設定] で [編集] を選択します。
6. [アプリケーションによって生成] を選択します。
7. [Cookie 名] にアプリケーション Cookie の名前を入力します。
8. [Save changes] (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制御セッションの維持を有効にするには AWS CLI

1. 次の [create-app-cookie-stickiness-policy](#) コマンドを使用して、アプリケーションが生成する Cookie の維持ポリシーを作成します。

```
aws elb create-app-cookie-stickiness-policy --load-balancer-name my-loadbalancer --policy-name my-app-cookie-policy --cookie-name my-app-cookie
```

2. 次の [set-load-balancer-policies-of-listener](#) コマンドを使用して、ロードバランサーのセッション維持を有効にします。

```
aws elb set-load-balancer-policies-of-listener --load-balancer-name my-loadbalancer --load-balancer-port 443 --policy-names my-app-cookie-policy
```

#### Note

`set-load-balancer-policies-of-listener` コマンドは、指定されたロードバランサーのポートに関連付けられた現在の一連のポリシーを置き換えます。このコマンドを使用するたびに、`--policy-names` を指定して有効にするすべてのポリシーを一覧表示します。

3. (省略可) 次の [describe-load-balancers](#) コマンドを使用して、維持ポリシーが有効になっていることを確認します。

```
aws elb describe-load-balancers --load-balancer-name my-loadbalancer
```

4. 応答には、指定したポートのリスナーに対してポリシーが有効になっていることを示す次の情報が出力されます。

```
{
  "LoadBalancerDescriptions": [
    {
      ...
      "ListenerDescriptions": [
        {
          "Listener": {
            "InstancePort": 443,
            "SSLCertificateId": "arn:aws:iam::123456789012:server-
certificate/my-server-certificate",
            "LoadBalancerPort": 443,
            "Protocol": "HTTPS",
            "InstanceProtocol": "HTTPS"
          },
          "PolicyNames": [
            "my-app-cookie-policy",
            "ELBSecurityPolicy-TLS-1-2-2017-01"
          ]
        },
        {
          "Listener": {
            "InstancePort": 80,
            "LoadBalancerPort": 80,
            "Protocol": "TCP",
            "InstanceProtocol": "TCP"
          },
          "PolicyNames": []
        }
      ],
      ...
      "Policies": {
        "LBCookieStickinessPolicies": [],
        "AppCookieStickinessPolicies": [
          {
            "PolicyName": "my-app-cookie-policy",
            "CookieName": "my-app-cookie"
          }
        ]
      }
    }
  ]
}
```

```
    ],
    "OtherPolicies": [
      "ELBSecurityPolicy-TLS-1-2-2017-01"
    ]
  },
  ...
}
]
```

## Classic Load Balancer の desync 軽減モードの設定

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、github の [HTTP Desync Guardian](#) を参照してください。

### 目次

- [分類](#)
- [モード](#)
- [desync 軽減モードの変更](#)

#### Tip

この設定は、Classic Load Balancers にのみ適用されます。Application Load Balancer に関する情報については、「[Application Load Balancers の Desync 軽減モード](#)」を参照してください。

## 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。

次のリストでは、分類別の問題について説明します。

### Acceptable

- 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。
- リクエストバージョンに不正な値が含まれています。
- 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。
- 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。

### Ambiguous

- リクエスト URI に制御文字が含まれています。
- 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。
- 同じ値を持つ複数の Content-Length ヘッダーがあります。
- 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。
- 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。
- GET または HEAD リクエストには Content-Length ヘッダーがあります。
- GET または HEAD リクエストには、Transfer-Encoding ヘッダーがあります。

## Severe

- 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。
- 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。
- 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。
- Transfer-Encoding ヘッダーに不正な値が含まれています。
- リクエストメソッドの形式が正しくありません。
- リクエストバージョンの形式が正しくありません。
- 異なる値を持つ複数の Content-Length ヘッダーがあります。
- 複数の Transfer-Encoding: chunked ヘッダーがあります。

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、「[Classic Load Balancer のメトリクス](#)」を参照してください。

## モード

次の表は、Classic Load Balancers 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 <sup>1</sup>	ブロック
Severe	許可	ブロック	ブロック

<sup>1</sup> 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。

## desync 軽減モードの変更

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. [ロードバランサー属性の編集] ページの [トラフィックの設定] で、[防御的 - 推奨]、[最も厳格]、または [モニタリング] を選択します。
6. [Save changes] (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、`elb.http.desyncmitigationmode` 属性に `monitor`、`defensive`、`strictest` のいずれかを設定しながら使用します。

```
aws elb modify-load-balancer-attributes --load-balancer-name my-load-balancer --load-balancer-attributes file://attribute.json
```

`attribute.json` の内容は次のとおりです。

```
{
  "AdditionalAttributes": [
    {
      "Key": "elb.http.desyncmitigationmode",
      "Value": "strictest"
    }
  ]
}
```

## Classic Load Balancer のプロキシプロトコルを設定する

プロキシプロトコルは、接続をリクエストする送信元から接続がリクエストされる送信先に接続情報を伝達するために使用される、インターネットプロトコルです。Elastic Load Balancing のプロキシプロトコルでは、人間にとって可読性があるヘッダー形式を持つバージョン 1 を使用しています。

デフォルトでは、フロントエンド接続とバックエンド接続の両方で Transmission Control Protocol (TCP) を使用した場合に、Classic Load Balancer がインスタンスにリクエストを転送する際、そのリクエストヘッダーは変更されません。プロキシプロトコルを有効にすると、送信元 IP アドレス、送信先 IP アドレス、ポート番号などの接続情報を含むリクエストヘッダーに、人間が判読できるヘッダーが追加されます。これにより、ヘッダーがリクエストの一部としてインスタンスに送信されます。

**Note**

AWS マネジメントコンソール はプロキシプロトコルの有効化をサポートしていません。

## 内容

- [プロキシプロトコルヘッダー](#)
- [プロキシプロトコルを有効にするための前提条件](#)
- [を使用してプロキシプロトコルを有効にする AWS CLI](#)
- [を使用してプロキシプロトコルを無効にする AWS CLI](#)

## プロキシプロトコルヘッダー

プロキシプロトコルヘッダーは、ロードバランサーでバックエンド接続用に TCP 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とインスタンスの間のトラフィックを傍受するため、インスタンス内のアクセスログには、発信元クライアントの IP アドレスでなく、ロードバランサーの IP アドレスが含まれています。リクエストの 1 行めを解析して、クライアントの IP アドレスとポート番号を取得することができます。

IPv6 のヘッダー内のプロキシのアドレスは、ロードバランサーのパブリック IPv6 アドレスです。この IPv6 アドレスは、ipv6 または dualstack で始まるロードバランサーの DNS 名から解決される IP アドレスと一致します。クライアントが IPv4 に接続する場合、ヘッダー内のプロキシのアドレスはロードバランサーのプライベート IPv4 アドレスであるため、DNS ルックアップでは解決できません。

プロキシプロトコルの行は 1 行であり、キャリッジリターンとラインフィード ("\r\n") で終わります。形式は次のとおりです。

```
PROXY_STRING + single space + INET_PROTOCOL + single space + CLIENT_IP + single space + PROXY_IP + single space + CLIENT_PORT + single space + PROXY_PORT + "\r\n"
```

例: IPv4

次に IPv4 のプロキシプロトコルの例を示します。

```
PROXY TCP4 198.51.100.22 203.0.113.7 35646 80\r\n
```

## プロキシプロトコルを有効にするための前提条件

開始する前に、以下を実行します:

- ロードバランサーが、プロキシプロトコルが有効になっているプロキシサーバーの背後にないことを確認します。プロキシプロトコルがプロキシサーバーとロードバランサーの両方で有効になっていると、ロードバランサーは、プロキシサーバーからのヘッダーが既に追加されているリクエストに別のヘッダーを追加します。インスタンの設定によっては、このような重複によってエラーが発生する可能性があります。
- インスタンスでプロキシプロトコル情報を処理できることを確認します。
- リスナー設定でプロキシプロトコルがサポートされることを確認します。詳細については、「[Classic Load Balancer のリスナー設定](#)」を参照してください。

## を使用してプロキシプロトコルを有効にする AWS CLI

プロキシプロトコルを有効にするには、タイプ ProxyProtocolPolicyType のポリシーを作成し、このポリシーをインスタンスのポートで有効にします。

次の手順を使用して、ProxyProtocolPolicyType タイプのロードバランサーの新しいポリシーを作成し、ポート 80 のインスタンスに設定して、そのポリシーが有効になっていることを確認します。

ロードバランサーの Proxy Protocol を有効にするには

1. (オプション) 次の [describe-load-balancer-policy-types](#) コマンドを使用して、Elastic Load Balancing でサポートされているポリシーを一覧表示します。

```
aws elb describe-load-balancer-policy-types
```

応答には、サポートされるポリシータイプの名前と説明が出力されます。ProxyProtocolPolicyType タイプの出力は次のとおりです。

```
{
  "PolicyTypeDescriptions": [
    ...
    {
      "PolicyAttributeTypeDescriptions": [
        {
          "Cardinality": "ONE",
```

```

        "AttributeName": "ProxyProtocol",
        "AttributeType": "Boolean"
    }
  ],
  "PolicyTypeName": "ProxyProtocolPolicyType",
  "Description": "Policy that controls whether to include the IP address
and port of the originating
request for TCP messages. This policy operates on TCP/SSL listeners only"
},
  ...
]
}

```

2. 次の [create-load-balancer-policy](#) コマンドを使用して、プロキシプロトコルを有効にするポリシーを作成します。

```
aws elb create-load-balancer-policy --load-balancer-name my-loadbalancer --policy-name my-ProxyProtocol-policy --policy-type-name ProxyProtocolPolicyType --policy-attributes AttributeName=ProxyProtocol,AttributeValue=true
```

3. 次の [set-load-balancer-policies-for-backend-server](#) コマンドを使用して、新しく作成したポリシーを特定のポートで有効にします。このコマンドは現在有効になっている一連のポリシーを置き換えることに注意してください。したがって、`--policy-names` オプションで、リストに追加するポリシー (`my-ProxyProtocol-policy` など) と現在有効になっているポリシー (`my-existing-policy` など) の両方を指定する必要があります。

```
aws elb set-load-balancer-policies-for-backend-server --load-balancer-name my-loadbalancer --instance-port 80 --policy-names my-ProxyProtocol-policy my-existing-policy
```

4. (オプション) 次の [describe-load-balancers](#) コマンドを使用して、プロキシプロトコルが有効になっていることを確認します。

```
aws elb describe-load-balancers --load-balancer-name my-loadbalancer
```

応答には、`my-ProxyProtocol-policy` ポリシーがポート 80 に関連付けられていることを示す次の情報が出力されます。

```
{
  "LoadBalancerDescriptions": [
    {
```

```
...
  "BackendServerDescriptions": [
    {
      "InstancePort": 80,
      "PolicyNames": [
        "my-ProxyProtocol-policy"
      ]
    }
  ],
  ...
}
]
```

## を使用してプロキシプロトコルを無効にする AWS CLI

インスタンスに関連付けられているポリシーは無効にすることができ、また後で有効にすることができます。

プロキシプロトコルをポリシーを無効にするには

1. 次の [set-load-balancer-policies-for-backend-server](#) コマンドを使用して、`--policy-names` オプションからプロキシプロトコルポリシーを削除して、このポリシーを無効にします。(ただし、`my-existing-policy` など、有効なままにしておく必要のある他のポリシーは残します)。

```
aws elb set-load-balancer-policies-for-backend-server --load-balancer-name my-loadbalancer --instance-port 80 --policy-names my-existing-policy
```

有効にする他のポリシーがない場合は、次のとおり `--policy-names` で空の文字列を指定します。

```
aws elb set-load-balancer-policies-for-backend-server --load-balancer-name my-loadbalancer --instance-port 80 --policy-names "[]"
```

2. (省略可) 次の [describe-load-balancers](#) コマンドを使用して、ポリシーが無効になっていることを確認します。

```
aws elb describe-load-balancers --load-balancer-name my-loadbalancer
```

応答には、ポリシーに関連付けられているポートがないことを示す次の情報が出力されます。

```
{
  "LoadBalancerDescriptions": [
    {
      ...
      "BackendServerDescriptions": [],
      ...
    }
  ]
}
```

## Classic Load Balancer へのタグ付け

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 Classic Load Balancer に対して複数のタグを付けられます。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。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。

### 目次

- [タグの制限](#)
- [タグの追加](#)
- [タグの削除](#)

## タグの制限

タグには以下のような基本制限があります。

- 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。

## タグの追加

ロードバランサーにはいつでもタグを追加できます。

コンソールを使用してタグを追加するには

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. ロードバランサーの名前を選択して、その詳細ページを開きます。
4. [Tags (タグ)] タブで、[Manage tags (タグ管理)] を選択します。
5. [タグの管理] ページで、タグごとに [新しいタグの追加] をクリックし、各タグのキーと値を指定します。
6. タグの追加を完了したら、[変更を保存] を選択します。

を使用してタグを追加するには AWS CLI

指定されたタグを追加するには、次の [add-tags](#) コマンドを使用します。

```
aws elb add-tags --load-balancer-name my-loadbalancer --tag "Key=project,Value=Lima"
```

## タグの削除

タグが不要になったときはいつでも、ロードバランサーからタグを削除できます。

コンソールを使用してタグを削除するには

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 [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. ロードバランサーの名前を選択して、その詳細ページを開きます。
4. [Tags (タグ)] タブで、[Manage tags (タグ管理)] を選択します。
5. [タグの管理] ページで、削除する各タグの横にある [削除] を選択します。
6. タグの削除を終了したら、[変更を保存] を選択します。

を使用してタグを削除するには AWS CLI

指定されたキーを持つタグを削除するには、次の [remove-tags](#) コマンドを使用します。

```
aws elb remove-tags --load-balancer-name my-loadbalancer --tag project
```

## Classic Load Balancer のサブネットの設定

サブネットをロードバランサーに追加すると、Elastic Load Balancing はアベイラビリティゾーンにロードバランサーノードを作成します。ロードバランサーノードは、クライアントからのトラフィックを受け入れ、リクエストを 1 つ以上のアベイラビリティゾーンにある正常な登録済みのインスタンスに送信します。少なくとも 2 つのアベイラビリティゾーンに、それぞれ 1 つのサブネットを追加することをお勧めします。これにより、ロードバランサーの可用性が向上します。ロードバランサーのサブネットは、いつでも変更できます。

インスタンスと同じアベイラビリティゾーンからサブネットを選択します。ロードバランサーがインターネット向けロードバランサーである場合は、バックエンドインスタンス (これがプライベートサブネット内にある場合でも) がロードバランサーからトラフィックを受信できるように、パブリックサブネットを選択する必要があります。ロードバランサーが内部向けロードバランサーである場合、プライベートサブネットを選択することをお勧めします。ロードバランサーのサブネットの詳細については、「[VPC の推奨事項](#)」を参照してください。

サブネットを追加するには、ロードバランサーでアベイラビリティゾーンにインスタンスを登録して、その後同じアベイラビリティゾーンからロードバランサーにサブネットをアタッチします。詳細については、「[Classic Load Balancer に EC2 インスタンスを登録するには](#)」を参照してください。

サブネットを追加したら、ロードバランサーは対応するアベイラビリティゾーン内の登録済みインスタンスにリクエストをルーティングするようになります。デフォルトでは、ロードバランサーはアベイラビリティゾーン間でサブネットにリクエストを均等にルーティングします。アベイラビリティゾーンのサブネットに登録済みのインスタンス間でリクエストを均等にルーティングする

には、クロスゾーン負荷分散を有効にします。詳細については、「[Classic Load Balancer のクロスゾーン負荷分散の設定](#)」を参照してください。

アベイラビリティゾーンに正常な登録済みインスタンスが存在しなくなった場合、または登録済みインスタンスのトラブルシューティングや更新を行う場合は、ロードバランサーからサブネットを一時的に削除することをお勧めします。サブネットを削除すると、ロードバランサーはこのアベイラビリティゾーンの登録済みインスタンスにリクエストをルーティングしなくなります。アベイラビリティゾーン内の残りのサブネットの登録済みインスタンスには、引き続きリクエストをルーティングします。サブネットを削除しても、そのサブネット内のインスタンスはロードバランサーに登録されたままであることに注意してください。ただし、登録を解除することも選択できます。詳細については、「[Classic Load Balancer に EC2 インスタンスを登録するには](#)」を参照してください。

## 内容

- [要件](#)
- [コンソールを使用してサブネットを設定する](#)
- [CLI を使用してサブネットを設定する](#)

## 要件

ロードバランサーのサブネットを更新する場合、以下の要件を満たす必要があります。

- ロードバランサーには、サブネットが常に少なくとも 1 つ必要です。
- アベイラビリティゾーンにつき、多くとも 1 つのサブネットしか追加できません。
- ローカルゾーンサブネットを追加することはできません。

ロードバランサーからサブネットを追加および削除するための別個の API が存在するため、これらの要件を満たすために新しいサブネットの現在のサブネットを入れ替えるときはオペレーションの順序を慎重に検討する必要があります。さらに、ロードバランサーのすべてのサブネットを入れ替える必要がある場合、別のアベイラビリティゾーンから一時的にサブネットを追加する必要があります。たとえば、ロードバランサーに単一のアベイラビリティゾーンがあり、サブネットを別のサブネットに入れられる必要がある場合、まず 2 番目のアベイラビリティゾーンからサブネットを追加する必要があります。その後、元のアベイラビリティゾーンからサブネットを削除し (追加されたサブネットが 1 つ未満にならないようにします)、元のアベイラビリティゾーンから新しいサブネットを追加した後 (アベイラビリティゾーンあたりサブネットが 1 つを超えないようにします)、2 番目のアベイラビリティゾーンからサブネットを削除します (入れ替えの実行に必要な場合のみ)。

## コンソールを使用してサブネットを設定する

次の手順で、コンソールを使用してサブネットを追加または削除します。

コンソールを使用してサブネットを設定するには

1. Amazon EC2 コンソールの <https://console.aws.amazon.com/ec2/> を開いてください。
2. ナビゲーションペインの [ロードバランシング] で [ロードバランサー] を選択します。
3. 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. [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. [サブネットの編集] ページの [ネットワークマッピング] セクションで、必要に応じてサブネットを追加および削除します。
6. 完了したら、[変更を保存] を選択します。

## CLI を使用してサブネットを設定する

AWS CLIを使用してサブネットを追加または削除するには、次の例を使用します。

CLI を使用してサブネットをロードバランサーに追加するには

次の [attach-load-balancer-to-subnets](#) コマンドを使用して、2 つのサブネットをロードバランサーに追加します。

```
aws elb attach-load-balancer-to-subnets --load-balancer-name my-load-balancer --  
subnets subnet-dea770a9 subnet-fb14f6a2
```

応答には、ロードバランサーのすべてのサブネットが一覧表示されます。例えば、次のようになります。

```
{  
  "Subnets": [  
    "subnet-5c11033e",  
    "subnet-dea770a9",  
    "subnet-fb14f6a2"  
  ]  
}
```

を使用してサブネットを削除するには AWS CLI

次の [detach-load-balancer-from-subnets](#) コマンドを使用して、指定したサブネットを特定のロードバランサーから削除します。

```
aws elb detach-load-balancer-from-subnets --load-balancer-name my-loadbalancer --subnets subnet-450f5127
```

応答には、ロードバランサーの残りのサブネットが一覧表示されます。例えば、次のようになります。

```
{
  "Subnets": [
    "subnet-15aaab61"
  ]
}
```

## Classic Load Balancer のセキュリティグループの設定

を使用してロードバランサー AWS マネジメントコンソール を作成する場合は、既存のセキュリティグループを選択するか、新しいセキュリティグループを作成できます。既存のセキュリティグループを選択する場合、ロードバランサーのリスナーポートとヘルスチェックポートの両方で、双方向のトラフィックを許可する必要があります。セキュリティグループを作成する場合、これらのポートのすべてのトラフィックを許可するルールがコンソールによって自動的に追加されます。

[デフォルト以外の VPC] AWS CLI または API を使用してデフォルト以外の VPC にロードバランサーを作成するが、セキュリティグループを指定しない場合、ロードバランサーは VPC のデフォルトのセキュリティグループに自動的に関連付けられます。

[デフォルト VPC] AWS CLI または API を使用してデフォルト VPC にロードバランサーを作成する場合、ロードバランサーの既存のセキュリティグループを選択することはできません。代わりに、ロードバランサー用に指定したポートでのすべてのトラフィックを許可するルールが適用されたセキュリティグループが、Elastic Load Balancing により設定されます。Elastic Load Balancing は、`default_elb_id` という名前 (例: ) で、AWS アカウントごとにそのようなセキュリティグループを 1 つだけ作成します `default_elb_fc5fbcd3-0405-3b7d-a328-ea290EXAMPLE`。今後デフォルト VPC 内に作成するロードバランサーにも、このセキュリティグループが使用されます。新しいロードバランサーのリスナーポートおよびヘルスチェックポートのトラフィックが許可されるように、セキュリティグループルールを必ず確認してください。ロードバランサーを削除した場合でも、このセキュリティグループは自動的に削除されません。

既存のロードバランサーにリスナーを追加する場合、新しいリスナーポートが両方向で許可されるように、セキュリティグループを確認する必要があります。

## 目次

- [ロードバランサーのセキュリティグループの推奨ルール](#)
- [コンソールを使用したセキュリティグループの割り当て](#)
- [を使用してセキュリティグループを割り当てる AWS CLI](#)

## ロードバランサーのセキュリティグループの推奨ルール

ロードバランサーのセキュリティグループでは、インスタンスとの通信が許可される必要があります。推奨ルールは、ロードバランサーのタイプ (インターネット向けまたは内部向け) によって異なります。

### インターネット向けロードバランサー

次の表に、インターネット向けロードバランサーの推奨インバウンドルールを示します。

ソース	プロトコル	ポート範囲	コメント
0.0.0.0/0	TCP	####	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

次の表に、インターネット向けロードバランサーの推奨アウトバウンドルールを示します。

目的地	プロトコル	ポート範囲	コメント
##### #####	TCP	#####	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
##### #####	TCP	#####	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

次の表に、内部ロードバランサーの推奨インバウンドルールを示します。

ソース	プロトコル	ポート範囲	コメント
VPC CIDR	TCP	####	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

次の表に、内部ロードバランサーの推奨アウトバウンドルールを示します。

目的地	プロトコル	ポート範囲	コメント
##### #####	TCP	#####	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
##### #####	TCP	#####	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

## コンソールを使用したセキュリティグループの割り当て

次の手順に従って、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変更を行います。

コンソールを使用してロードバランサーに割り当てられたセキュリティグループを更新するには

1. Amazon EC2 コンソールの <https://console.aws.amazon.com/ec2/> を開いてください。
2. ナビゲーションペインの [ロードバランシング] で [ロードバランサー] を選択します。
3. 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. [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の編集] ページの [セキュリティグループ] で、必要に応じてセキュリティグループを追加または削除します

最大 5 個のセキュリティグループを追加できます。

6. 完了したら、[変更を保存] を選択します。

## を使用してセキュリティグループを割り当てる AWS CLI

次の [apply-security-groups-to-load-balancer](#) コマンドを使用して、セキュリティグループをロードバランサーに関連付けます。指定されたセキュリティグループは、以前に関連付けられたセキュリティグループに上書きされます。

```
aws elb apply-security-groups-to-load-balancer --load-balancer-name my-loadbalancer --security-groups sg-53fae93f
```

以下に、応答の例を示します。

```
{
  "SecurityGroups": [
    "sg-53fae93f"
  ]
}
```

## Classic Load Balancer のネットワーク ACL の設定

VPC のデフォルトネットワークアクセスコントロールリスト (ACL) では、すべての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が許可されます。カスタムネットワーク ACL を作成する場合、ロードバランサーとインスタンスの通信を許可するルールを追加します。

ロードバランサーのサブネットの推奨ルールは、ロードバランサーのタイプ (インターネット向けまたは内部向け) によって異なります。

インターネット向けロードバランサー

インターネット向けロードバランサーの推奨インバウンドルールを次に示します。

ソース	プロトコル	ポート範囲	コメント
0.0.0.0/0	TCP	<i>####</i>	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

ソース	プロトコル	ポート範囲	コメント
<i>VPC CIDR</i>	TCP	1024-65535	一時ポートで VPC CIDR からのインバウンドトラフィックを許可する

インターネット向けロードバランサーの推奨アウトバウンドルールを次に示します。

目的地	プロトコル	ポート範囲	コメント
<i>VPC CIDR</i>	TCP	<i>#####</i>	インスタンスのリスナーポートですべてのアウトバウンドトラフィックを許可する
<i>VPC CIDR</i>	TCP	<i>#####</i>	ヘルスチェックポートですべてのアウトバウンドトラフィックを許可する
0.0.0.0/0	TCP	1024-65535	一時ポートですべて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

内部ロードバランサーの推奨インバウンドルールを次に示します。

ソース	プロトコル	ポート範囲	コメント
<i>VPC CIDR</i>	TCP	<i>####</i>	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
<i>VPC CIDR</i>	TCP	1024-65535	一時ポートで VPC CIDR からのインバウンドトラフィックを許可する

内部ロードバランサーの推奨アウトバウンドルールを次に示します。

目的地	プロトコル	ポート範囲	コメント
VPC CIDR	TCP	#####	インスタンスリスナーポートで VPC CIDR へのアウトバウンドトラフィックを許可する
VPC CIDR	TCP	#####	ヘルスチェックポートで VPC CIDR へのアウトバウンドトラフィックを許可する
VPC CIDR	TCP	1024-65535	一時ポートで VPC CIDR へのアウトバウンドトラフィックを許可する

## Classic Load Balancer のカスタムドメイン名の設定

各 Classic Load Balancer は、デフォルトのドメインネームシステム (DNS) 名を受け取ります。この DNS 名には、ロードバランサーが作成される AWS リージョンの名前が含まれます。例えば、米国西部 (オレゴン) リージョンで my-loadbalancer という名前のロードバランサーを作成した場合、そのロードバランサーは my-loadbalancer-1234567890.us-west-2.elb.amazonaws.com などの DNS 名を受け取ります。インスタンスでウェブサイトにはアクセスするには、ウェブブラウザのアドレスフィールドにこの DNS 名を貼り付けます。ただし、この DNS 名は顧客が簡単に記憶して使用できる名前ではありません。

ロードバランサーのデフォルトの DNS 名を www.example.com などの覚えやすい DNS 名にするには、カスタムドメイン名を作成し、ロードバランサーの DNS 名に関連付けます。このカスタムドメイン名を使用してクライアントがリクエストを生成すると、DNS サーバーがロードバランサーの DNS 名に解決します。

### 目次

- [カスタムドメイン名とロードバランサー名の関連付け](#)
- [ロードバランサーの Route 53 DNS フェイルオーバーを使用する](#)
- [ドメイン名とロードバランサーの関連付けの解除](#)

## カスタムドメイン名とロードバランサー名の関連付け

まず、まだドメイン名を登録していない場合は、ドメイン名を登録します。インターネットのドメイン名は、Internet Corporation for Assigned Names and Numbers (ICANN) によって管理されています。ドメイン名は、ドメイン名のレジストリを管理する ICANN 認定機関であるドメイン名レジストラを使用して登録します。レジストラのウェブサイトで、ドメイン名の登録に関する詳細な手順と料金情報を確認します。詳細については、以下のリソースを参照してください。

- Amazon Route 53 を使用してドメイン名を登録するには、Amazon Route 53 開発者ガイドの「[Route 53 を使用したドメイン名の登録](#)」を参照してください。
- 認定されているレジストラのリストについては、「[List of Accredited Registrars](#)」を参照してください。

次に、ドメインレジストラなどの DNS サービスを使用して、ロードバランサーにクエリをルーティングするために CNAME レコードを作成します。詳細については、DNS サービスのドキュメントを参照してください。

前出と別に、DNS サービスとして Route 53 を使用する方法もあります。インターネット上のトラフィックを (ドメイン内で) ルーティングする方法に関する情報を含むホストゾーンと、ドメイン名へのクエリをロードバランサーにルーティングするエイリアスリソースレコードセットを作成します。Route 53 では、エイリアスレコードセットの DNS クエリには課金されません。エイリアスレコードセットは、ドメインの Zone Apex (例: example.com) のロードバランサーに DNS クエリをルーティングするために使用できます。既存のドメインの DNS サービスを Route 53 に転送する方法については、Amazon Route 53 開発者ガイドの「[DNS サービスとして Amazon Route 53 を設定する](#)」を参照してください。

最後に、Route 53 を使用してドメインのホストゾーンとエイリアスレコードセットを作成します。詳細については、Amazon Route 53 開発者ガイドの「[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。

## ロードバランサーの 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、ロードバランサーに登録された EC2 インスタンスのヘルスチェックが Route 53 によって行われ、利用可能かどうか判断されます。ロードバランサーに正常な EC2 インスタンス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た場合、Route 53 はそのトラ

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

- Classic 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。Classic Load Balancer の場合、Route 53 はロードバランサーに関連付けられたインスタンスのヘルスチェックを使用します。
- Classic Load Balancer に登録されているインスタンスの少なくとも 1 つが正常である場合、Route 53 はエイリアスレコードを正常としてマークします。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。
- Classic Load Balancer に登録されているすべてのインスタンスに異常がある場合、Route 53 はエイリアスレコードを異常とマークします。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が使用されている場合、Route 53 はセカンダリレコードを返します。

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。

## ドメイン名とロードバランサーの関連付けの解除

ロードバランサーインスタンスからドメイン名の関連付けを解除するには、まずホストゾーンのリソースレコードセットを削除してから、ホストゾーンを削除します。詳細については、Amazon Route 53 開発者ガイドの「[レコードの編集](#)」および「[パブリックホストゾーンの削除](#)」を参照してください。

# Classic Load Balancer のリスナー

Elastic Load Balancing の使用を開始する前に、Classic Load Balancer のリスナーを 1 つまたは複数設定する必要があります。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。リスナーは、フロントエンド (クライアントからロードバランサー) 接続用のプロトコルとポート、およびバックエンド (ロードバランサーからバックエンドインスタンス) 接続用のプロトコルとポートを使用して設定します。

Elastic Load Balancing は、次のプロトコルをサポートしています

- HTTP
- HTTPS (セキュア HTTP)
- TCP
- SSL (セキュア TCP)

HTTPS プロトコルは、HTTP レイヤー経由のセキュアな接続を確立するために SSL プロトコルを使用します。SSL プロトコルは、TCP レイヤー経由のセキュアな接続を確立する場合にも使用することができます。

フロントエンド接続が TCP または SSL を使用している場合、バックエンド接続は TCP または SSL を使用できます。フロントエンド接続が HTTP または HTTPS を使用している場合、バックエンド接続は HTTP または HTTPS を使用できます。

バックエンドインスタンスは、ポート 1~65535 をリッスンできます。

ロードバランサーがリッスンできるポート: 1-65535

内容

- [プロトコル](#)
- [HTTPS/SSL リスナー](#)
- [Classic Load Balancer のリスナー設定](#)
- [HTTP ヘッダーと Classic Load Balancer](#)

# プロトコル

一般的なウェブアプリケーション通信は、ハードウェアとソフトウェアの複数のレイヤーを通過します。各レイヤーは固有の通信機能を提供します。通信機能のコントロールはあるレイヤーから次のレイヤーへと順番に渡されます。開放型システム間相互接続 (OSI) は、これらのレイヤーでプロトコルと呼ばれる通信の標準形式を実装するためのモデルフレームワークを定義します。詳細については、Wikipedia の「[OSI model](#)」を参照してください。

Elastic Load Balancing を使用する場合、レイヤー 4 とレイヤー 7 についての基本的な理解が必要です。レイヤー 4 はトランスポートレイヤーで、ロードバランサーを介したクライアントとバックエンドインスタンスとの間の Transmission Control Protocol (TCP) 接続を記述します。レイヤー 4 はロードバランサーで設定可能な一番下のレベルです。レイヤー 7 はアプリケーションレイヤーで、クライアントからロードバランサー、およびロードバランサーからバックエンドインスタンスへの Hypertext Transfer Protocol (HTTP) および HTTPS (セキュア HTTP) 接続の使用について記述します。

Secure Sockets Layer (SSL) プロトコルは、インターネットなどの安全性が低いネットワークでの機密データの暗号化に主に使用されます。SSL プロトコルは、クライアントとバックエンドサーバーの間のセキュアな接続を確立し、クライアントとバックエンドサーバ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と完全性を保証します。

## TCP/SSL プロトコル

フロントエンド接続とバックエンド接続の両方に TCP (レイヤー 4) を使用する場合、ロードバランサーはヘッダーを変更せずにバックエンドインスタンスにリクエストを転送します。ロードバランサーは、リクエストを受け取った後、リスナー設定で指定されたポートを使ってバックエンドインスタンスに対する TCP 接続を開こうと試みます。

ロードバランサーはクライアントとバックエンドインスタンスの間のトラフィックをインターセプトするため、バックエンドインスタンスのアクセスログには、発信元クライアントの IP アドレスでなく、ロードバランサーの IP アドレスが含まれています。プロキシプロトコルを有効にして、送信元 IP アドレス、送信先 IP アドレス、ポート番号などのクライアントの接続情報を含むヘッダーを追加できます。これにより、ヘッダーがリクエストの一部としてバックエンドインスタンスに送信されません。接続情報を取得するには、リクエストの 1 行めを解析します。詳細については、「[Classic Load Balancer のプロキシプロトコルを設定する](#)」を参照してください。

この設定を使用すると、ステイッキーセッション用の Cookie や X-Forwarded ヘッダー用の Cookie は受け取りません。

## HTTP/HTTPS プロトコル

フロントエンド接続とバックエンド接続の両方に HTTP (レイヤー 7) を使用する場合、ロードバランサーはリクエストのヘッダーを解析し、バックエンドインスタンスにリクエストを送信します。

HTTP/HTTPS ロードバランサーの背後にあるあらゆる登録された正常なインスタンスについて、Elastic Load Balancing は 1 つ以上の TCP 接続を開いて維持します。これらの接続によって、HTTP/HTTPS リクエストを受信できる接続が常に確立されていることが保証されます。

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。Elastic Load Balancing は X-Forwarded-For ヘッダーをサポートします。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、X-Forwarded-For リクエストヘッダーを使用します。詳細については、「[X-Forwarded-For](#)」を参照してください。

HTTP/HTTPS を使用するときには、ロードバランサーでステイッキーセッションを有効にできます。ステイッキーセッションは、ユーザーのセッションを特定のバックエンドインスタンスに結び付けます。これにより、セッション中にそのユーザーから来たリクエストをすべて同じバックエンドインスタンスに送信することができます。詳細については、「[Classic Load Balancer のステイッキーセッションの設定](#)」を参照してください。

ロードバランサーはすべての HTTP 拡張機能をサポートしていません。予期しないメソッド、応答コード、またはその他の標準でない HTTP 1.0/1.1 実装のためにロードバランサーがリクエストを終了できない場合、TCP リスナーを使用する必要がある場合もあります。

## HTTPS/SSL リスナー

以下のセキュリティ機能を持つロードバランサーを作成できます。

### SSL サーバー証明書

フロントエンド接続に HTTPS または SSL を使用する場合、ロードバランサーに X.509 証明書 (SSL サーバー証明書) をデプロイする必要があります。ロードバランサーは、クライアントからのリクエストをバックエンドインスタンスに送信する前に、これを復号化します (SSL ターミネーションといいます)。詳細については、「[Classic Load Balancer 用の SSL/TLS 証明書](#)」を参照してください。

ロードバランサーによる SSL ターミネーション (SSL オフローディングと呼ばれる) の処理を希望しない場合には、フロントエンド接続とバックエンド接続のどちらでも TCP を使用して登録されたインスタンスに証明書をデプロイし、リクエストを処理させることができます。

## SSL ネゴシエーション

Elastic Load Balancing には、クライアントとロードバランサーの間に接続が確立されたときに SSL ネゴシエーションに使用される定義済み SSL ネゴシエーション設定があります。SSL ネゴシエーション設定は幅広いクライアントとの互換性を提供し、暗号という強力な暗号アルゴリズムを使用します。ただし、ネットワーク上のすべてのデータを暗号化する必要があるようなユースケースでは、特定の暗号しか使用できない場合があります。一部のセキュリティコンプライアンス基準 (PCI、SOX など) では、セキュリティの基準を確実に満たすために、クライアントからの特定のポートと暗号のセットを要求する場合があります。そのような場合、固有の要件に応じてカスタム SSL ネゴシエーション設定を作成できます。暗号とプロトコルは 30 秒以内に有効になります。詳細については、「[Classic Load Balancers での SSL ネゴシエーション設定](#)」を参照してください。

## バックエンドサーバー認証

バックエンド接続に HTTPS あるいは SSL 接続を使用する場合には、登録されたインスタンスに認証を有効化できます。そして、この認証プロセスを使用して、インスタンスが暗号化された通信のみを受け入れ、それぞれの登録インスタンスに適切なパブリックキーが存在することを確認できます。

詳細については、「[バックエンドサーバー認証の設定](#)」を参照してください。

## Classic Load Balancer のリスナー設定

次の表は、Classic Load Balancer の HTTP および HTTPS リスナーで使用可能な設定を示しています。

ユースケース	フロントエンドプロトコル	フロントエンドのオプション	バックエンドプロトコル	バックエンドのオプション	注意事項
基本的な HTTP ロードバランサー	HTTP	NA	HTTP	NA	<ul style="list-style-type: none"> <li><a href="#">X-Forwarded ヘッダー</a>がサポートされます。</li> </ul>
安全なウェブサイトまたはアプリケーション	HTTPS	<a href="#">SSL ネゴシエーション</a>	HTTP	NA	<ul style="list-style-type: none"> <li><a href="#">X-Forwarded ヘッダー</a>がサ</li> </ul>

ユースケース	フロントエンドプロトコル	フロントエンドのオプション	バックエンドプロトコル	バックエンドのオプション	注意事項
リクエスト ( Elastic Load Balancing を使用して SSL 復号をオフロード )					<p>ポートされます。</p> <ul style="list-style-type: none"> <li>• <a href="#">SSL 証明書</a>がロードバランサーにデプロイされている必要があります。</li> </ul>
安全なウェブサイトまたはアプリケーション (エンドツーエンド暗号化を使用)	HTTPS	<a href="#">SSL ネゴシエーション</a>	HTTPS	バックエンド認証	<ul style="list-style-type: none"> <li>• <a href="#">X-Forwarded ヘッダー</a>がサポートされます。</li> <li>• <a href="#">SSL 証明書</a>がロードバランサーと登録されたインスタンスにデプロイされている必要があります。</li> </ul>

次の表は、Classic Load Balancer の TCP および SSL リスナーで使用可能な設定を示しています。

ユースケース	フロントエンドプロトコル	フロントエンドのオプション	バックエンドプロトコル	バックエンドのオプション	注意事項
基本的な TCP ロード バランサー	TCP	NA	TCP	NA	<ul style="list-style-type: none"> <li>• <a href="#">プロキシプロトコルヘッダー</a>をサポート</li> </ul>
安全なウェブ サイトまたは アプリケーション ( Elastic Load Balancing を使用して SSL 復号をオフロード )	SSL	<a href="#">SSL ネゴシエーション</a>	TCP	NA	<ul style="list-style-type: none"> <li>• <a href="#">SSL 証明書</a>がロード バランサー にデプロイ されている 必要があります。</li> <li>• <a href="#">プロキシプロトコルヘッダー</a>をサポート</li> </ul>
安全なウェブ サイトまたは アプリケーション ( Elastic Load Balancing で エンドツーエンド暗号化を使用 )	SSL	<a href="#">SSL ネゴシエーション</a>	SSL	バックエンド 認証	<ul style="list-style-type: none"> <li>• <a href="#">SSL 証明書</a>がロード バランサー と登録された インスタンスにデプロイ されている 必要があります。</li> <li>• バックエンド SSL 接続で SNI ヘッダーを挿入しません</li> </ul>

ユースケース	フロントエンドプロトコル	フロントエンドのオプション	バックエンドプロトコル	バックエンドのオプション	注意事項
					<ul style="list-style-type: none"> <li>• プロキシプロトコルヘッダーをサポートしません</li> </ul>

## HTTP ヘッダーと Classic 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。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、「[メッセージヘッダー](#)」RFC 2616 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。Classic 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。

### 前提条件

- リスナー設定が X-Forwarded ヘッダーをサポートしていることを確認します。詳細については、「[Classic Load Balancer のリスナー設定](#)」を参照してください。
- クライアント IP アドレスを記録するようウェブサーバーを設定します。

### 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自動的に追加され、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、X-Forwarded-For 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。左端のアドレスは、リクエストが最初に行われたクライアント IP です。この後に、後続のプロキシ識別子が連なって続きます。

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
```

## 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。

# Classic Load Balancer の HTTPS リスナー

暗号化された接続で SSL/TLS プロトコルを使用するロードバランサーを作成できます (SSL オフロードとも呼ばれます)。この機能を使用することにより、ロードバランサーと HTTPS セッションを開始するクライアントとの間でトラフィックを暗号化できます。また、ロードバランサーと EC2 インスタンス間の接続でトラフィックを暗号化することもできます。

Elastic Load Balancing は、クライアントとロードバランサー間の接続をネゴシエーションするために、Secure Sockets Layer (SSL) の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) を使用します。フロントエンド接続に HTTPS/SSL を使用する場合は、定義済みのセキュリティポリシーまたはカスタムセキュリティポリシーのどちらかを使用することができます。ロードバランサーには SSL 証明書をデプロイする必要があります。インスタンスにリクエストを送信する前に、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、バックエンド接続の静的暗号化スイートを使用します。オプションとしてインスタンスで認証を有効にすることもできます。

Classic Load Balancer は Server Name Indication (SNI) をサポートしていません。代わりに、次のいずれかの代替案を使用できます。

- 証明書を 1 つロードバランサーにデプロイし、追加のウェブサイトごとにサブジェクト代替名 (SAN) を追加します。SAN により、1 つの証明書を使用して複数のホスト名を保護できるようになります。証明書ごとにサポートされる SAN の数と SAN の追加および削除方法の詳細については、ご利用の証明書プロバイダにお問い合わせください。
- フロントエンド接続とバックエンド接続のポート 443 で TCP リスナーを使用します。ロードバランサーはリクエストをそのまま渡すため、EC2 インスタンスで HTTPS 終了を処理できます。

Classic Load Balancers は、相互 TLS 認証 (mTLS) をサポートしていません。mTLS のサポートのためには、TCP リスナーを作成します。ロードバランサーはリクエストをそのまま渡すため、EC2 インスタンスで mTLS を実装できます。

## 内容

- [Classic Load Balancer 用の SSL/TLS 証明書](#)
- [Classic Load Balancers での SSL ネゴシエーション設定](#)
- [Classic Load Balancer 用の事前定義済み SSL セキュリティポリシー](#)
- [HTTPS リスナーを使用する Classic Load Balancer の作成](#)

- [Classic Load Balancer の HTTPS リスナーの設定](#)
- [Classic Load Balancer の SSL 証明書の置き換え](#)
- [Classic Load Balancer の SSL ネゴシエーション設定の更新](#)

## Classic Load Balancer 用の SSL/TLS 証明書

フロントエンドリスナーに HTTPS (SSL or TLS) を使用する場合、ロードバランサーに TLS/TLS 証明書をデプロイする必要があります。インスタンスにリクエストを送信する前に、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。

SSL および TLS プロトコルは、X.509 証明書 (SSL/TLS サーバー証明書) を使用して、クライアントとバックエンドアプリケーションの両方を認証します。X.509 証明書は、認証機関 (CA) によって発行された IDENTITY デジタル フォームで、識別情報、有効期間、パブリックキー、シリアルナンバー、発行者のデジタル署名が含まれます。

証明書は、AWS Certificate Manager または OpenSSL などの SSL および TLS プロトコルをサポートするツールを使用して作成できます。この証明書は、ロードバランサーの HTTPS リスナーを作成または更新するときに指定します。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。

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。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、TLS 接続を確認できないため、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ます。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 [件名] 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。

## を使用して SSL/TLS 証明書を作成またはインポートする AWS Certificate Manager

(AWS Certificate Manager ACM) を使用して、ロードバランサーの証明書を作成またはインポートすることをお勧めします。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。ロードバランサーに証明書をデプロイするには、証明書がロードバランサーと同じリージョンにある必要があります。詳細については、AWS Certificate Manager ユーザーガイドの [パブリック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。

ユーザーが AWS マネジメントコンソールを使用して証明書をロードバランサーにデプロイするには、ACM ListCertificates API アクションへのアクセスを許可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 [証明書の一覧表示](#) を参照してください。

### Important

4096 ビット RSA キーまたは EC キーを使用する証明書を、ACM との統合を利用してロードバランサーにインストールすることはできません。4096 ビット RSA キーまたは EC キーを使用する証明書をロードバランサーで使用するには、IAM にアップロードする必要があります。

## IAM を使用する SSL/TLS 証明書のインポート

ACM を使用していない場合は、OpenSSL などの SSL/TLS を使用して署名証明書リクエスト (CSR) を作成し、CA から CSR 署名を取得して証明書を発行し、IAM に証明書をアップロードすることができます。詳細については、IAM ユーザーガイドの [Working with server certificates](#) を参照してください。

## Classic Load Balancers での SSL ネゴシエーション設定

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、SSL プロトコル、SSL 暗号、およびサーバーの優先順位オプションを組み合わせたものです。ロードバランサーの SSL 接続を設定する方法については、「[Classic Load Balancer のリスナー](#)」を参照してください。

### 目次

- [セキュリティポリシー](#)

- [SSL プロトコル](#)
- [サーバーの優先順位](#)
- [SSL 暗号](#)
- [バックエンド接続用の暗号スイート](#)

## セキュリティポリシー

セキュリティポリシーによって、クライアントとロードバランサー間の SSL ネゴシエーションでサポートされる暗号とプロトコルが決まります。Classic Load Balancer では、事前定義済み、もしくはカスタムで作成したセキュリティポリシーのいずれかを使用するように設定できます。

AWS Certificate Manager (ACM) によって提供される証明書には、RSA パブリックキーが含まれていることに注意してください。このため、ACM が提供する証明書を使用する場合は、セキュリティポリシーに RSA を使用する暗号セットを含める必要があります。含めない場合は TLS 接続が失敗します。

### 事前定義されたセキュリティポリシー

事前に定義された最新のセキュリティポリシーの名前には、そのポリシーがリリースされた年月に基づくバージョン情報が含まれています。たとえば、事前に定義されたデフォルトのセキュリティポリシーは ELBSecurityPolicy-2016-08 です。事前に定義されたセキュリティポリシーを新たにリリースするたびに、そのポリシーを使用するよう設定を更新できます。

事前に定義されたセキュリティポリシーで有効なプロトコルと暗号については、「[Classic Load Balancer 用の事前定義済み SSL セキュリティポリシー](#)」を参照してください。

### カスタムセキュリティポリシー

必要な暗号とプロトコルを備えたカスタムネゴシエーション設定を作成できます。たとえば、一部のセキュリティコンプライアンス基準 (PCI や SOC など) では、セキュリティの基準を確実に満たすために、特定のプロトコルと暗号のセットを要求する場合があります。このような場合は、それらの基準を満たすカスタムセキュリティポリシーを作成できます。

セキュリティポリシーの作成についての詳細は、「[Classic Load Balancer の SSL ネゴシエーション設定の更新](#)」を参照してください。

## SSL プロトコル

SSL 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。

Secure Sockets Layer (SSL) と Transport Layer Security (TLS) はインターネットなどの安全性の低いネットワークでの機密データの暗号化に使用される暗号プロトコルです。TLS プロトコルは SSL プロトコルの新しいバージョンです。Elastic Load Balancing のドキュメントでは、SSL と TLS のプロトコルの両方を、SSL プロトコルと呼称しています。

## 推奨プロトコル

ELBSecurityPolicy-TLS-1-2-2017-01 の定義済みセキュリティポリシーで使用されている TLS 1.2 をお勧めします。TLS 1.2 をカスタムセキュリティポリシーで使用することもできます。デフォルトのセキュリティポリシーは TLS 1.2 とそれ以前のバージョンの TLS の両方をサポートしているため、ELBSecurityPolicy-TLS-1-2-2017-01 よりも安全性が低くなります。

## 非推奨のプロトコル

以前にカスタムポリシーで SSL 2.0 プロトコルを有効にした場合、セキュリティポリシーを事前に定義されたセキュリティポリシーに更新することをお勧めします。

## サーバーの優先順位

Elastic Load Balancing では、クライアントとロードバランサー間の接続をネゴシエートする際の、サーバーの優先順位オプションをサポートしています。SSL 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、クライアントのリストで最初にロードバランサーの暗号と一致した暗号が SSL 接続用に選択されます。サーバーの優先順位をサポートするようにロードバランサーが設定されていると、ロードバランサーはクライアントの暗号リストに含まれている暗号と最初に一致する暗号を、ロードバランサーのリストから選択します。このようにして、SSL 接続に使用される暗号がロードバランサーによって決定されます。サーバーの優先順位を有効にしない場合は、クライアントで設定された暗号の順序がクライアントとロードバランサー間の接続のネゴシエーションに使用されます。

## SSL 暗号

SSL 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。SSL プロトコルは、複数の SSL 暗号を使用してインターネット経由のデータを暗号化します。

AWS Certificate Manager (ACM) によって提供される証明書には、RSA パブリックキーが含まれていることに注意してください。このため、ACM が提供する証明書を使用する場合は、セキュリティポリシーに RSA を使用する暗号セットを含める必要があります。含めない場合は TLS 接続が失敗します。

Elastic Load Balancing では、次の暗号の Classic Load Balancer での使用がサポートされています。これらの暗号のサブセットは、事前に定義された SSL ポリシーによって使用されます。これらのすべての暗号は、カスタムポリシーで使用できます。デフォルトのセキュリティポリシー (アスタリスクがあるもの) に含まれている暗号のみを使用することをお勧めします。他の多くの暗号は安全ではなく、お客様の責任にてご使用ください。

## 暗号

- ECDHE-ECDSA-AES128- GCM-SHA256 \*
- ECDHE-RSA-AES128- GCM-SHA256 \*
- ECDHE-ECDSA-AES128-SHA256 \*
- ECDHE-RSA-AES128-SHA256 \*
- ECDHE-ECDSA-AES128-SHA \*
- ECDHE-RSA-AES128-SHA \*
- DHE-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 \*
- AES128-GCM-SHA256 \*
- AES128-SHA256 \*
- AES128-SHA \*
- AES256-GCM-SHA384 \*
- AES256-SHA256 \*
- AES256-SHA \*
- DHE-DSS-AES128-SHA
- CAMELLIA128-SHA
- EDH-RSA-DES-CBC3-SHA
- DES-CBC3-SHA

- ECDHE-RSA-RC4-SHA
- RC4-SHA
- ECDHE-ECDSA-RC4-SHA
- DHE-DSS-AES256-GCM-SHA384
- DHE-RSA-AES256-GCM-SHA384
- DHE-RSA-AES256-SHA256
- DHE-DSS-AES256-SHA256
- DHE-RSA-AES256-SHA
- DHE-DSS-AES256-SHA
- DHE-RSA-CAMELLIA256-SHA
- DHE-DSS-CAMELLIA256-SHA
- CAMELLIA256-SHA
- EDH-DSS-DES-CBC3-SHA
- DHE-DSS-AES128-GCM-SHA256
- DHE-RSA-AES128-GCM-SHA256
- DHE-RSA-AES128-SHA256
- DHE-DSS-AES128-SHA256
- DHE-RSA-CAMELLIA128-SHA
- DHE-DSS-CAMELLIA128-SHA
- ADH-AES128-GCM-SHA256
- ADH-AES128-SHA
- ADH-AES128-SHA256
- ADH-AES256-GCM-SHA384
- ADH-AES256-SHA
- ADH-AES256-SHA256
- ADH-CAMELLIA128-SHA
- ADH-CAMELLIA256-SHA
- ADH-DES-CBC3-SHA
- ADH-DES-CBC-SHA

- ADH-RC4-MD5
- ADH-SEED-SHA
- DES-CBC-SHA
- DHE-DSS-SEED-SHA
- DHE-RSA-SEED-SHA
- EDH-DSS-DES-CBC-SHA
- EDH-RSA-DES-CBC-SHA
- IDEA-CBC-SHA
- RC4-MD5
- SEED-SHA
- DES-CBC3-MD5
- DES-CBC-MD5
- RC2-CBC-MD5
- PSK-AES256-CBC-SHA
- PSK-3DES-EDE-CBC-SHA
- KRB5-DES-CBC3-SHA
- KRB5-DES-CBC3-MD5
- PSK-AES128-CBC-SHA
- PSK-RC4-SHA
- KRB5-RC4-SHA
- KRB5-RC4-MD5
- KRB5-DES-CBC-SHA
- KRB5-DES-CBC-MD5
- EXP-EDH-RSA-DES-CBC-SHA
- EXP-EDH-DSS-DES-CBC-SHA
- EXP-ADH-DES-CBC-SHA
- EXP-DES-CBC-SHA
- EXP-RC2-CBC-MD5
- EXP-KRB5-RC2-CBC-SHA

- EXP-KRB5-DES-CBC-SHA
- EXP-KRB5-RC2-CBC-MD5
- EXP-KRB5-DES-CBC-MD5
- EXP-ADH-RC4-MD5
- EXP-RC4-MD5
- EXP-KRB5-RC4-SHA
- EXP-KRB5-RC4-MD5

\* これらは、デフォルトのセキュリティポリシー ELBSecurityPolicy-2016-08 に含まれる暗号です。

## バックエンド接続用の暗号スイート

Classic Load Balancer は、バックエンド接続の静的暗号化スイートを使用します。Classic Load Balancer と登録されたインスタンスが接続をネゴシエートできない場合は、次のいずれかの暗号を含めます。

- AES256-GCM-SHA384
- AES256-SHA256
- AES256-SHA
- CAMELLIA256-SHA
- AES128-GCM-SHA256
- AES128-SHA256
- AES128-SHA
- CAMELLIA128-SHA
- RC4-SHA
- DES-CBC3-SHA
- DES-CBC-SHA
- DHE-DSS-AES256-GCM-SHA384
- DHE-RSA-AES256-GCM-SHA384
- DHE-RSA-AES256-SHA256
- DHE-DSS-AES256-SHA256

- DHE-RSA-AES256-SHA
- DHE-DSS-AES256-SHA
- DHE-RSA-CAMELLIA256-SHA
- DHE-DSS-CAMELLIA256-SHA
- DHE-DSS-AES128-GCM-SHA256
- DHE-RSA-AES128-GCM-SHA256
- DHE-RSA-AES128-SHA256
- DHE-DSS-AES128-SHA256
- DHE-RSA-AES128-SHA
- DHE-DSS-AES128-SHA
- DHE-RSA-CAMELLIA128-SHA
- DHE-DSS-CAMELLIA128-SHA
- EDH-RSA-DES-CBC3-SHA
- EDH-DSS-DES-CBC3-SHA
- EDH-RSA-DES-CBC-SHA
- EDH-DSS-DES-CBC-SHA

## Classic Load Balancer 用の事前定義済み SSL セキュリティポリシー

HTTPS/SSL リスナー用の事前定義されたセキュリティポリシーのいずれかを選択できます。ELBSecurityPolicy-TLS ポリシーの 1 つを使用して、特定の TLS プロトコルバージョンを無効にする必要があるコンプライアンスとセキュリティの標準を満たすことができます。または、カスタムセキュリティポリシーを作成できます。詳細については、「[SSL ネゴシエーション設定の更新](#)」を参照してください。

RSA および DSA 方式の暗号は、SSL 証明書の作成に使用される署名アルゴリズムに固有です。SSL 証明書は、セキュリティポリシーで有効にされた暗号に基づいた署名アルゴリズムを使用して作成してください。

[サーバーの優先順位] に対応しているポリシーを選択すると、ロードバランサーは、この一覧に指定されている順序で暗号を使用し、クライアントとロードバランサーとの接続をネゴシエートします。それ以外の場合は、ロードバランサーはクライアントが提示した順番で暗号化を使用します。

次のセクションは、有効な SSL プロトコルと SSL 暗号を含め、Classic Load Balancer 用に事前定義された最新のセキュリティポリシーを示しています。事前定義済みのポリシーの詳細を見るには、[describe-load-balancer-policies](#) コマンドを使用することもできます。

### Tip

この情報は、Classic Load Balancer にのみ適用されます。他のロードバランサーに適用される情報については、「[Application Load Balancer のセキュリティポリシー](#)」、および「[Network Load Balancer のセキュリティポリシー](#)」を参照してください。

## 内容

- [ポリシー別のプロトコル](#)
- [ポリシー別の暗号](#)
- [暗号別のポリシー](#)

## ポリシー別のプロトコル

次の表は、各セキュリティポリシーがサポートする TLS プロトコルを示しています。

セキュリティポリシー	TLS 1.2	TLS 1.1	TLS 1.0
ELBSecurityPolicy-TLS-1-2-2017-01	はい	いいえ	いいえ
ELBSecurityPolicy-TLS-1-1-2017-01	はい	はい	いいえ
ELBSecurityPolicy-2016-08	はい	はい	はい
ELBSecurityPolicy-2015-05	はい	はい	はい
ELBSecurityPolicy-2015-03	はい	はい	はい
ELBSecurityPolicy-2015-02	はい	はい	はい

## ポリシー別の暗号

以下の表は、各セキュリティポリシーがサポートする暗号について説明しています。

セキュリティポリシー	暗号
ELBSecurityPolicy-TLS-1-2-2017-01	<ul style="list-style-type: none"> <li>• ECDHE-ECDSA-AES128-GCM-SHA256</li> <li>• ECDHE-RSA-AES128-GCM-SHA256</li> <li>• ECDHE-ECDSA-AES128-SHA256</li> <li>• ECDHE-RSA-AES128-SHA256</li> <li>• ECDHE-ECDSA-AES256-GCM-SHA384</li> <li>• ECDHE-RSA-AES256-GCM-SHA384</li> <li>• ECDHE-ECDSA-AES256-SHA384</li> <li>• ECDHE-RSA-AES256-SHA384</li> <li>• AES128-GCM-SHA256</li> <li>• AES128-SHA256</li> <li>• AES256-GCM-SHA384</li> <li>• AES256-SHA256</li> </ul>
ELBSecurityPolicy-TLS-1-1-2017-01	<ul style="list-style-type: none"> <li>• ECDHE-ECDSA-AES128-GCM-SHA256</li> <li>• ECDHE-RSA-AES128-GCM-SHA256</li> <li>• ECDHE-ECDSA-AES128-SHA256</li> <li>• ECDHE-RSA-AES128-SHA256</li> <li>• ECDHE-ECDSA-AES128-SHA</li> <li>• ECDHE-RSA-AES128-SHA</li> <li>• ECDHE-ECDSA-AES256-GCM-SHA384</li> <li>• ECDHE-RSA-AES256-GCM-SHA384</li> <li>• ECDHE-ECDSA-AES256-SHA384</li> <li>• ECDHE-RSA-AES256-SHA384</li> <li>• ECDHE-ECDSA-AES256-SHA</li> <li>• ECDHE-RSA-AES256-SHA</li> <li>• AES128-GCM-SHA256</li> </ul>

セキュリティポリシー	暗号
	<ul style="list-style-type: none"> <li>• AES128-SHA256</li> <li>• AES128-SHA</li> <li>• AES256-GCM-SHA384</li> <li>• AES256-SHA256</li> <li>• AES256-SHA</li> </ul>
ELBSecurityPolicy-2016-08	<ul style="list-style-type: none"> <li>• ECDHE-ECDSA-AES128-GCM-SHA256</li> <li>• ECDHE-RSA-AES128-GCM-SHA256</li> <li>• ECDHE-ECDSA-AES128-SHA256</li> <li>• ECDHE-RSA-AES128-SHA256</li> <li>• ECDHE-ECDSA-AES128-SHA</li> <li>• ECDHE-RSA-AES128-SHA</li> <li>• ECDHE-ECDSA-AES256-GCM-SHA384</li> <li>• ECDHE-RSA-AES256-GCM-SHA384</li> <li>• ECDHE-ECDSA-AES256-SHA384</li> <li>• ECDHE-RSA-AES256-SHA384</li> <li>• ECDHE-ECDSA-AES256-SHA</li> <li>• ECDHE-RSA-AES256-SHA</li> <li>• AES128-GCM-SHA256</li> <li>• AES128-SHA256</li> <li>• AES128-SHA</li> <li>• AES256-GCM-SHA384</li> <li>• AES256-SHA256</li> <li>• AES256-SHA</li> </ul>

セキュリティポリシー	暗号
ELBSecurityPolicy-2015-05	<ul style="list-style-type: none"><li>• ECDHE-ECDSA-AES128-GCM-SHA256</li><li>• ECDHE-RSA-AES128-GCM-SHA256</li><li>• ECDHE-ECDSA-AES128-SHA256</li><li>• ECDHE-RSA-AES128-SHA256</li><li>• ECDHE-ECDSA-AES128-SHA</li><li>• ECDHE-RSA-AES128-SHA</li><li>• ECDHE-ECDSA-AES256-GCM-SHA384</li><li>• ECDHE-RSA-AES256-GCM-SHA384</li><li>• ECDHE-ECDSA-AES256-SHA384</li><li>• ECDHE-RSA-AES256-SHA384</li><li>• ECDHE-ECDSA-AES256-SHA</li><li>• ECDHE-RSA-AES256-SHA</li><li>• AES128-GCM-SHA256</li><li>• AES128-SHA256</li><li>• AES128-SHA</li><li>• AES256-GCM-SHA384</li><li>• AES256-SHA256</li><li>• AES256-SHA</li><li>• DES-CBC3-SHA</li></ul>

セキュリティポリシー	暗号
ELBSecurityPolicy-2015-03	<ul style="list-style-type: none"><li>• ECDHE-ECDSA-AES128-GCM-SHA256</li><li>• ECDHE-RSA-AES128-GCM-SHA256</li><li>• ECDHE-ECDSA-AES128-SHA256</li><li>• ECDHE-RSA-AES128-SHA256</li><li>• ECDHE-ECDSA-AES128-SHA</li><li>• ECDHE-RSA-AES128-SHA</li><li>• ECDHE-ECDSA-AES256-GCM-SHA384</li><li>• ECDHE-RSA-AES256-GCM-SHA384</li><li>• ECDHE-ECDSA-AES256-SHA384</li><li>• ECDHE-RSA-AES256-SHA384</li><li>• ECDHE-ECDSA-AES256-SHA</li><li>• ECDHE-RSA-AES256-SHA</li><li>• AES128-GCM-SHA256</li><li>• AES128-SHA256</li><li>• AES128-SHA</li><li>• AES256-GCM-SHA384</li><li>• AES256-SHA256</li><li>• AES256-SHA</li><li>• DHE-RSA-AES128-SHA</li><li>• DHE-DSS-AES128-SHA</li><li>• DES-CBC3-SHA</li></ul>

セキュリティポリシー	暗号
ELBSecurityPolicy-2015-02	<ul style="list-style-type: none"> <li>• ECDHE-ECDSA-AES128-GCM-SHA256</li> <li>• ECDHE-RSA-AES128-GCM-SHA256</li> <li>• ECDHE-ECDSA-AES128-SHA256</li> <li>• ECDHE-RSA-AES128-SHA256</li> <li>• ECDHE-ECDSA-AES128-SHA</li> <li>• ECDHE-RSA-AES128-SHA</li> <li>• ECDHE-ECDSA-AES256-GCM-SHA384</li> <li>• ECDHE-RSA-AES256-GCM-SHA384</li> <li>• ECDHE-ECDSA-AES256-SHA384</li> <li>• ECDHE-RSA-AES256-SHA384</li> <li>• ECDHE-ECDSA-AES256-SHA</li> <li>• ECDHE-RSA-AES256-SHA</li> <li>• AES128-GCM-SHA256</li> <li>• AES128-SHA256</li> <li>• AES128-SHA</li> <li>• AES256-GCM-SHA384</li> <li>• AES256-SHA256</li> <li>• AES256-SHA</li> <li>• DHE-RSA-AES128-SHA</li> <li>• DHE-DSS-AES128-SHA</li> </ul>

## 暗号別のポリシー

以下は、各暗号をサポートしているセキュリティポリシーの一覧です。

暗号名	セキュリティポリシー	暗号スイート
OpenSSL – ECDHE-ECDSA-AES128-GCM-SHA256	<ul style="list-style-type: none"> <li>• ELBSecurityPolicy-TLS-1-2-2017-01</li> <li>• ELBSecurityPolicy-TLS-1-1-2017-01</li> </ul>	c02b

暗号名	セキュリティポリシー	暗号スイート
IANA – TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	<ul style="list-style-type: none"> <li>• ELBSecurityPolicy-2016-08</li> <li>• ELBSecurityPolicy-2015-05</li> <li>• ELBSecurityPolicy-2015-03</li> <li>• ELBSecurityPolicy-2015-02</li> </ul>	
OpenSSL – ECDHE-RSA-AES128-GCM-SHA256  IANA – TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	<ul style="list-style-type: none"> <li>• ELBSecurityPolicy-TLS-1-2-2017-01</li> <li>• ELBSecurityPolicy-TLS-1-1-2017-01</li> <li>• ELBSecurityPolicy-2016-08</li> <li>• ELBSecurityPolicy-2015-05</li> <li>• ELBSecurityPolicy-2015-03</li> <li>• ELBSecurityPolicy-2015-02</li> </ul>	c02f
OpenSSL – ECDHE-ECDSA-AES128-SHA256  IANA – TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	<ul style="list-style-type: none"> <li>• ELBSecurityPolicy-TLS-1-2-2017-01</li> <li>• ELBSecurityPolicy-TLS-1-1-2017-01</li> <li>• ELBSecurityPolicy-2016-08</li> <li>• ELBSecurityPolicy-2015-05</li> <li>• ELBSecurityPolicy-2015-03</li> <li>• ELBSecurityPolicy-2015-02</li> </ul>	c023
OpenSSL – ECDHE-RSA-AES128-SHA256  IANA – TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	<ul style="list-style-type: none"> <li>• ELBSecurityPolicy-TLS-1-2-2017-01</li> <li>• ELBSecurityPolicy-TLS-1-1-2017-01</li> <li>• ELBSecurityPolicy-2016-08</li> <li>• ELBSecurityPolicy-2015-05</li> <li>• ELBSecurityPolicy-2015-03</li> <li>• ELBSecurityPolicy-2015-02</li> </ul>	c027

暗号名	セキュリティポリシー	暗号スイート
OpenSSL – ECDHE-ECDSA-AES128-SHA  IANA – TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	<ul style="list-style-type: none"> <li>• ELBSecurityPolicy-TLS-1-1-2017-01</li> <li>• ELBSecurityPolicy-2016-08</li> <li>• ELBSecurityPolicy-2015-05</li> <li>• ELBSecurityPolicy-2015-03</li> <li>• ELBSecurityPolicy-2015-02</li> </ul>	c009
OpenSSL – ECDHE-RSA-AES128-SHA  IANA – TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	<ul style="list-style-type: none"> <li>• ELBSecurityPolicy-TLS-1-1-2017-01</li> <li>• ELBSecurityPolicy-2016-08</li> <li>• ELBSecurityPolicy-2015-05</li> <li>• ELBSecurityPolicy-2015-03</li> <li>• ELBSecurityPolicy-2015-02</li> </ul>	c013
OpenSSL – ECDHE-ECDSA-AES256-GCM-SHA384  IANA – TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	<ul style="list-style-type: none"> <li>• ELBSecurityPolicy-TLS-1-2-2017-01</li> <li>• ELBSecurityPolicy-TLS-1-1-2017-01</li> <li>• ELBSecurityPolicy-2016-08</li> <li>• ELBSecurityPolicy-2015-05</li> <li>• ELBSecurityPolicy-2015-03</li> <li>• ELBSecurityPolicy-2015-02</li> </ul>	c02c
OpenSSL – ECDHE-RSA-AES256-GCM-SHA384  IANA – TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	<ul style="list-style-type: none"> <li>• ELBSecurityPolicy-TLS-1-2-2017-01</li> <li>• ELBSecurityPolicy-TLS-1-1-2017-01</li> <li>• ELBSecurityPolicy-2016-08</li> <li>• ELBSecurityPolicy-2015-05</li> <li>• ELBSecurityPolicy-2015-03</li> <li>• ELBSecurityPolicy-2015-02</li> </ul>	c030

暗号名	セキュリティポリシー	暗号スイート
OpenSSL – ECDHE-ECDSA-AES256-SHA384  IANA – TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	<ul style="list-style-type: none"> <li>• ELBSecurityPolicy-TLS-1-2-2017-01</li> <li>• ELBSecurityPolicy-TLS-1-1-2017-01</li> <li>• ELBSecurityPolicy-2016-08</li> <li>• ELBSecurityPolicy-2015-05</li> <li>• ELBSecurityPolicy-2015-03</li> <li>• ELBSecurityPolicy-2015-02</li> </ul>	c024
OpenSSL – ECDHE-RSA-AES256-SHA384  IANA – TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	<ul style="list-style-type: none"> <li>• ELBSecurityPolicy-TLS-1-2-2017-01</li> <li>• ELBSecurityPolicy-TLS-1-1-2017-01</li> <li>• ELBSecurityPolicy-2016-08</li> <li>• ELBSecurityPolicy-2015-05</li> <li>• ELBSecurityPolicy-2015-03</li> <li>• ELBSecurityPolicy-2015-02</li> </ul>	c028
OpenSSL – ECDHE-ECDSA-AES256-SHA  IANA – TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	<ul style="list-style-type: none"> <li>• ELBSecurityPolicy-TLS-1-1-2017-01</li> <li>• ELBSecurityPolicy-2016-08</li> <li>• ELBSecurityPolicy-2015-05</li> <li>• ELBSecurityPolicy-2015-03</li> <li>• ELBSecurityPolicy-2015-02</li> </ul>	c014
OpenSSL – ECDHE-RSA-AES256-SHA  IANA – TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	<ul style="list-style-type: none"> <li>• ELBSecurityPolicy-TLS-1-1-2017-01</li> <li>• ELBSecurityPolicy-2016-08</li> <li>• ELBSecurityPolicy-2015-05</li> <li>• ELBSecurityPolicy-2015-03</li> <li>• ELBSecurityPolicy-2015-02</li> </ul>	c00a

暗号名	セキュリティポリシー	暗号スイート
OpenSSL – AES128-GCM-SHA256  IANA – TLS_RSA_WITH_AES_128_GCM_SHA256	<ul style="list-style-type: none"> <li>• ELBSecurityPolicy-TLS-1-2-2017-01</li> <li>• ELBSecurityPolicy-TLS-1-1-2017-01</li> <li>• ELBSecurityPolicy-2016-08</li> <li>• ELBSecurityPolicy-2015-05</li> <li>• ELBSecurityPolicy-2015-03</li> <li>• ELBSecurityPolicy-2015-02</li> </ul>	9c
OpenSSL – AES128-SHA256  IANA – TLS_RSA_WITH_AES_128_CBC_SHA256	<ul style="list-style-type: none"> <li>• ELBSecurityPolicy-TLS-1-2-2017-01</li> <li>• ELBSecurityPolicy-TLS-1-1-2017-01</li> <li>• ELBSecurityPolicy-2016-08</li> <li>• ELBSecurityPolicy-2015-05</li> <li>• ELBSecurityPolicy-2015-03</li> <li>• ELBSecurityPolicy-2015-02</li> </ul>	3c
OpenSSL – AES128-SHA  IANA – TLS_RSA_WITH_AES_128_CBC_SHA	<ul style="list-style-type: none"> <li>• ELBSecurityPolicy-TLS-1-1-2017-01</li> <li>• ELBSecurityPolicy-2016-08</li> <li>• ELBSecurityPolicy-2015-05</li> <li>• ELBSecurityPolicy-2015-03</li> <li>• ELBSecurityPolicy-2015-02</li> </ul>	2f
OpenSSL – AES256-GCM-SHA384  IANA – TLS_RSA_WITH_AES_256_GCM_SHA384	<ul style="list-style-type: none"> <li>• ELBSecurityPolicy-TLS-1-2-2017-01</li> <li>• ELBSecurityPolicy-TLS-1-1-2017-01</li> <li>• ELBSecurityPolicy-2016-08</li> <li>• ELBSecurityPolicy-2015-05</li> <li>• ELBSecurityPolicy-2015-03</li> <li>• ELBSecurityPolicy-2015-02</li> </ul>	9d

暗号名	セキュリティポリシー	暗号スイート
OpenSSL – AES256-SHA256 IANA – TLS_RSA_WITH_AES_256_CBC_SHA256	<ul style="list-style-type: none"> <li>• ELBSecurityPolicy-TLS-1-2-2017-01</li> <li>• ELBSecurityPolicy-TLS-1-1-2017-01</li> <li>• ELBSecurityPolicy-2016-08</li> <li>• ELBSecurityPolicy-2015-05</li> <li>• ELBSecurityPolicy-2015-03</li> <li>• ELBSecurityPolicy-2015-02</li> </ul>	3d
OpenSSL – AES256-SHA IANA – TLS_RSA_WITH_AES_256_CBC_SHA	<ul style="list-style-type: none"> <li>• ELBSecurityPolicy-TLS-1-1-2017-01</li> <li>• ELBSecurityPolicy-2016-08</li> <li>• ELBSecurityPolicy-2015-05</li> <li>• ELBSecurityPolicy-2015-03</li> <li>• ELBSecurityPolicy-2015-02</li> </ul>	35
OpenSSL – DHE-RSA-AES128-SHA IANA – TLS_DHE_RSA_WITH_AES_128_CBC_SHA	<ul style="list-style-type: none"> <li>• ELBSecurityPolicy-2015-03</li> <li>• ELBSecurityPolicy-2015-02</li> </ul>	33
OpenSSL – DHE-DSS-AES128-SHA IANA – TLS_DHE_DSS_WITH_AES_128_CBC_SHA	<ul style="list-style-type: none"> <li>• ELBSecurityPolicy-2015-03</li> <li>• ELBSecurityPolicy-2015-02</li> </ul>	32
OpenSSL – DES-CBC3-SHA IANA – TLS_RSA_WITH_3DES_EDE_CBC_SHA	<ul style="list-style-type: none"> <li>• ELBSecurityPolicy-2015-05</li> <li>• ELBSecurityPolicy-2015-03</li> </ul>	0a

## HTTPS リスナーを使用する Classic 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信し、ロードバランサーに登録されている EC2 インスタンスに分散します。

HTTP (80) ポートおよび HTTPS (443) ポートの両方でリッスンするロードバランサーを作成できます。HTTPS リスナーがポート 80 でインスタンスにリクエストを送信する方法を指定すると、ロードバランサーはリクエストを終了し、ロードバランサーからインスタンスへの通信は暗号化されません。HTTPS リスナーがポート 443 でインスタンスにリクエストを送信する場合は、ロードバランサーからインスタンスへの通信は暗号化されます。

ロードバランサーがインスタンスとの通信に暗号化された接続を使用する場合、オプションでインスタンスでの認証を有効にできます。これにより、パブリックキーとロードバランサーにこのために指定したキーが一致した場合に限り、ロードバランサーがインスタンスと通信できます。

既存のロードバランサーに HTTPS リスナーを追加する方法については、「[Classic Load Balancer の HTTPS リスナーの設定](#)」を参照してください。

## 内容

- [前提条件](#)
- [コンソールを使用した HTTPS ロードバランサーの作成](#)
- [を使用して HTTPS ロードバランサーを作成する AWS CLI](#)

## 前提条件

始める前に、以下の前提条件を満たしていることを確認してください。

- 「[VPC の推奨事項](#)」の各ステップを実行します。
- ロードバランサーに登録する 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は、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。
- EC2 インスタンスは、ヘルスチェックの対象に対して、HTTP ステータスコード 200 で応答する必要があります。詳細については、「[Classic Load Balancer のインスタンスのヘルスチェック](#)」を参照してください。
- EC2 インスタンスでキープアライブのオプションを有効にする場合は、キープアライブの設定値をロードバランサーのアイドルタイムアウトの設定値以上に設定をすることをお勧めします。ロードバランサーがインスタンスへの接続を確実に閉じるようにする場合は、キープアライブ時間のインスタンスに設定された値が、ロードバランサーのアイドルタイムアウトの設定値より大きいことを確認します。詳細については、「[Classic Load Balancer でのアイドル接続のタイムアウト設定](#)」を参照してください。
- セキュアリスナーを作成する場合、ロードバランサーに SSL サーバー証明書をデプロイする必要があります。インスタンスにリクエストを送信する前に、ロードバランサーは証明書を使用してリ

クエストを終了し、次にリクエストを復号します。SSL 証明書がない場合は作成します。詳細については、「[Classic Load Balancer 用の SSL/TLS 証明書](#)」を参照してください。

## コンソールを使用した HTTPS ロードバランサーの作成

この例では、ロードバランサー用の 2 つのリスナーを設定します。最初のリスナーは、ポート 80 で HTTP リクエストを受け取り、HTTP を使用してポート 80 でインスタンスにそれらのリクエストを送信します。2 番目のリスナーは、ポート 443 で HTTPS リクエストを受け取り、HTTP を使用してポート 80 で (バックエンド認証を設定する場合は HTTPS を使用してポート 443 で) インスタンスにそれらのリクエストを送信します。

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。リスナーは、フロントエンド (クライアントからロードバランサー) 接続用のプロトコルとポート、およびバックエンド (ロードバランサーからインスタンス) 接続用のプロトコルとポートを使用して設定します。Elastic Load Balancing でサポートされるポート、プロトコル、およびリスナー設定の詳細については、「[Classic Load Balancer のリスナー](#)」を参照してください。

コンソールを使用してセキュアな Classic Load Balancer を作成するには

1. Amazon EC2 コンソールの <https://console.aws.amazon.com/ec2/> を開いてください。
2. ナビゲーションバーで、ロードバランサーのリージョンを選択します。EC2 インスタンス用に選択したの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 [ロードバランシング] で [ロードバランサー] を選択します。
4. [Create Load Balancer] を選択します。
5. [Classic Load Balancer] セクションを展開し、[作成] を選択します。
6. 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。

この Classic Load Balancer の名前は、リージョン内にある Classic Load Balancer の中で一意にする必要があります。使用可能なのは最大 32 文字で、英数字とハイフンのみ使用できます。また、先頭と末尾にハイフンを使用することはできません。

- b. [スキーム] で、[インターネットに接続] を選択します。

7. ネットワークマッピング

- a. [VPC] で、インスタンス用に選択したのと同じ VPC を選択します。

- b. マッピングの場合、最初にアベイラビリティゾーンを選択し、次に利用可能なサブネットからパブリックサブネットを選択します。アベイラビリティゾーンごとに選択できるサブネットは 1 つだけです。ロードバランサーの可用性を高めるには、複数のアベイラビリティゾーンからサブネットを選択します。

## 8. セキュリティグループ

- [セキュリティグループ] では、ポート 80 で必要な HTTP トラフィックとポート 443 での HTTPS トラフィックを許可するように設定された既存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が存在しない場合は、必要なルールを含む新しいセキュリティグループを作成できます。

## 9. リスナーとルーティング


- a. デフォルトのリスナーをデフォルト設定のままにし、[リスナーを追加] を選択します。
- b. 新しいリスナーの Listener で、プロトコルとして HTTPS を選択すると、ポートが 443 に更新されます。デフォルトでは、インスタンスはポート HTTP で 80 プロトコルを使用します。
- c. バックエンド認証が必要な場合は、インスタンスプロトコルを HTTPS に変更します。これにより、インスタンスポートも 443 に更新されます。

## 10. セキュアリスナー設定

フロントエンドリスナーに HTTPS または SSL を使用するとき、ロードバランサーに SSL 証明書をデプロイする必要があります。インスタンスにリクエストを送信する前に、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。また、セキュリティポリシーも指定する必要があります。Elastic Load Balancing には、SSL ネゴシエーション設定が事前定義されたセキュリティポリシーが用意されています。または、独自のカスタムセキュリティポリシーを作成することも可能です。バックエンド接続に HTTPS/SSL を設定した場合は、インスタンスの認証を有効にできます。

- a. [セキュリティポリシー] には、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か、カスタムポリシーを作成することをお勧めします。「[SSL ネゴシエーション設定の更新](#)」を参照してください。
- b. [デフォルトの SSL/TLS 証明書] では、次のオプションが利用可能です。
  - 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。
- c. (オプション) HTTPS リスナーを、暗号化された接続を使用してインスタンスと通信するように設定する場合は、オプションで [バックエンド認証証明書] でインスタンスの認証を設定できます。

 Note

[バックエンド認証証明書] セクションが表示されない場合は、[リスナーとルーティング]に戻り、[インスタンス]のプロトコルとして HTTPS を選択します。

- i. [Certificate name] にパブリックキー証明書の名前を入力します。
- ii. [証明書本文 (PEM エンコード)] に証明書の内容をコピーして貼り付けます。パブリックキーがこのキーと一致した場合のみ、ロードバランサーはインスタンスと通信します。
- iii. 他の証明書を追加するには、[別のバックエンド証明書の追加] を選択します。上限は 5 つです。

## 11. ヘルスチェック

- a. [Ping ターゲット] セクションで、[Ping プロトコル] と [Ping ポート] を選択します。EC2 インスタンスでは、指定された ping ポートでトラフィックを受け入れる必要があります。
- b. Ping ポートの場合は、ポートが 80 であることを確認してください。
- c. [Ping パス] の場合、デフォルト値を半角のスラッシュ (/) で置き換えます。これにより Elastic Load Balancing は、ヘルスチェックリクエストを、ウェブサーバーのデフォルトのホームページ (index.html など) に送信します。
- d. ヘルスチェックの詳細設定には、デフォルト値を使用します。

## 12. インスタンス

- a. [インスタンスの追加] を選択して、インスタンスの選択画面を表示します。
- b. [利用可能なインスタンス] で、前に選択したネットワーク設定に基づいて、ロードバランサーで利用可能な現在のインスタンスから選択できます。
- c. 選択内容に問題がなければ、[確認] を選択して、ロードバランサーに登録するインスタンスを追加します。

## 13. 属性

- [クロスゾーン負荷分散の有効化]、[Connection Draining の有効化]、および [タイムアウト (ドレーニング間隔)] はデフォルト値のままにします。

## 14. ロードバランサータグ (オプション)

- a. [キー] フィールドは必須です。
- b. [値] フィールドはオプションです。
- c. 別のタグを追加するには、[新しいタグを追加] を選択し、[キー] フィールドに値を入力し、オプションで [値] フィールドに値を入力します。
- d. 既存のタグを削除するには、削除したいタグの横にある [削除] を選択します。

## 15. 概要と作成

- a. 設定を変更する必要がある場合は、変更する必要がある設定の横にある [編集] を選択します。
- b. 概要に表示されているすべての設定に問題がなければ、[ロードバランサーの作成] を選択してロードバランサーの作成を開始します。
- c. 最後の作成ページで、[ロードバランサーを表示] を選択して、Amazon EC2 コンソールでロードバランサーを表示します。

## 16. 検証

- a. 新しいロードバランサーを選択します。
- b. [ターゲットインスタンス] タブで、[ヘルスステータス] 列を確認します。少なくとも 1 つの EC2 インスタンスの状態が稼動中であれば、ロードバランサーをテストできます。
- c. [詳細] セクションで、ロードバランサー DNS 名 をコピーします。これは my-load-balancer-1234567890.us-east-1.elb.amazonaws.com のようになります。

- d. ロードバランサの DNS 名を、パブリック 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貼り付けます。ロードバランサーが正しく機能している場合は、サーバーのデフォルトページが表示されます。

## 17. 削除 (オプション)

- a. ロードバランサーをポイントするドメインの CNAME 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。
- b. Amazon EC2 コンソールの <https://console.aws.amazon.com/ec2/> を開いてください。
- c. ロードバランサーを選択します。
- d. [アクション]、[ロードバランサーを削除] の順に選択します。
- e. 確認を求められたら、「confirm」と入力し、[削除] を選択します。
- f. ロードバランサーを削除しても、そのロードバランサーに登録された EC2 インスタンスは引き続き実行されます。実行が継続される分単位または時間単位でお支払いいただきます。EC2 インスタンスがなくなった場合は、追加料金が発生しないように停止または終了できます。

## を使用して HTTPS ロードバランサーを作成する AWS CLI

AWS CLIを使用して HTTPS/SSL ロードバランサーを作成するには、次の手順を実行します。

### タスク

- [ステップ 1: リスナーを設定する](#)
- [ステップ 2: SSL セキュリティポリシーを設定する](#)
- [ステップ 3: バックエンドインスタンス認証を設定する \(オプション\)](#)
- [ステップ 4: ヘルスチェックを設定する \(オプション\)](#)
- [ステップ 5: EC2 インスタンスを登録する](#)
- [ステップ 6: インスタンスを検証する](#)
- [ステップ 7: ロードバランサーを削除する \(オプション\)](#)

### ステップ 1: リスナーを設定する

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。リスナーは、フロントエンド (クライアントからロードバランサー) 接続用のプロトコルとポート、およびバックエンド (ロードバランサーからインスタンス) 接続用のプロトコルとポートを使用して設定します。Elastic Load Balancing でサ

ポートされるポート、プロトコル、およびリスナー設定の詳細については、「[Classic Load Balancer のリスナー](#)」を参照してください。

この例では、ポートおよびプロトコルをフロントエンド接続とバックエンド接続に使用するように指定することで、ロードバランサーに 2 つのリスナーを設定します。最初のリスナーは、ポート 80 で HTTP リクエストを受け取り、HTTP を使用してポート 80 でインスタンスにそれらのリクエストを送信します。2 つ目のリスナーは、ポート 443 で HTTPS リクエストを受け取り、HTTP を使用してポート 80 でインスタンスにリクエストを送信します。

2 番目のリスナーがフロントエンド接続に HTTPS を使用するため、SSL サーバー証明書をロードバランサーにデプロイする必要があります。インスタンスにリクエストを送信する前に、ロードバランサーは証明書を使用してリクエストを終了し、次にリクエストを復号します。

ロードバランサーのリスナーを設定するには

1. SSL 証明書の Amazon リソースネーム (ARN) を取得します。以下に例を示します。

ACM

```
arn:aws:acm:region:123456789012:certificate/12345678-1234-1234-1234-123456789012
```

IAM

```
arn:aws:iam::123456789012:server-certificate/my-server-certificate
```

2. 次の [create-load-balancer](#) コマンドを使用して、ロードバランサーに 2 つのリスナーを構成します。

```
aws elb create-load-balancer --load-balancer-name my-load-balancer --listeners  
"Protocol=http,LoadBalancerPort=80,InstanceProtocol=http,InstancePort=80"  
"Protocol=https,LoadBalancerPort=443,InstanceProtocol=http,InstancePort=80,SSLCertificateI  
--availability-zones us-west-2a
```

以下に、応答の例を示します。

```
{  
  "DNSName": "my-loadbalancer-012345678.us-west-2.elb.amazonaws.com"  
}
```

3. (オプション) 次の [describe-load-balancers](#) コマンドを使用して、ロードバランサーの詳細を表示します。

```
aws elb describe-load-balancers --load-balancer-name my-load-balancer
```

## ステップ 2: SSL セキュリティポリシーを設定する

事前定義されたセキュリティポリシーのいずれかを選択するか、独自のカスタムセキュリティポリシーを作成できます。それ以外の場合、Elastic Load Balancing は、事前定義されたデフォルトのセキュリティポリシー `ELBSecurityPolicy-2016-08` を使用してロードバランサーを構成します。詳細については、「[Classic Load Balancers での SSL ネゴシエーション設定](#)」を参照してください。

ロードバランサーにデフォルトのセキュリティポリシーが関連付けられていることを確認するには次の [describe-load-balancers](#) コマンドを使用します。

```
aws elb describe-load-balancers --load-balancer-name my-loadbalancer
```

以下に、応答の例を示します。ELBSecurityPolicy-2016-08 がポート 443 でロードバランサーに関連付けられることに注意してください。

```
{
  "LoadBalancerDescriptions": [
    {
      ...
      "ListenerDescriptions": [
        {
          "Listener": {
            "InstancePort": 80,
            "SSLCertificateId": "ARN",
            "LoadBalancerPort": 443,
            "Protocol": "HTTPS",
            "InstanceProtocol": "HTTP"
          },
          "PolicyNames": [
            "ELBSecurityPolicy-2016-08"
          ]
        },
        {
```

```
        "Listener": {
            "InstancePort": 80,
            "LoadBalancerPort": 80,
            "Protocol": "HTTP",
            "InstanceProtocol": "HTTP"
        },
        "PolicyNames": []
    }
],
...
}
]
```

必要に応じて、デフォルトのセキュリティポリシーを使用せずにロードバランサーの SSL セキュリティポリシーを設定できます。

(オプション) 事前定義された SSL セキュリティポリシーを使用するには

1. 次の [describe-load-balancer-policies](#) コマンドを使用して、事前定義されたセキュリティポリシーの名前を一覧表示します。

```
aws elb describe-load-balancer-policies
```

事前定義されたセキュリティポリシーの設定については、「[Classic Load Balancer 用の事前定義済み SSL セキュリティポリシー](#)」を参照してください。

2. 次の [create-load-balancer-policy](#) コマンドを使用して、前のステップで一覧表示した事前定義済みセキュリティポリシーのいずれかを利用する、SSL ネゴシエーションのポリシーを作成します。

```
aws elb create-load-balancer-policy --load-balancer-name my-loadbalancer
--policy-name my-SSLNegotiation-policy --policy-type-name SSLNegotiationPolicyType
--policy-attributes AttributeName=Reference-Security-
Policy,AttributeValue=predefined-policy
```

3. (オプション) 次の [describe-load-balancer-policies](#) コマンドを使用して、ポリシーが作成されたことを確認します。

```
aws elb describe-load-balancer-policies --load-balancer-name my-loadbalancer --
policy-name my-SSLNegotiation-policy
```

応答には、ポリシーの説明が含まれます。

4. 次の [set-load-balancer-policies-of-listener](#) コマンドを使用して、ポリシーをロードバランサーのポート 443 で有効にします。

```
aws elb set-load-balancer-policies-of-listener --load-balancer-name my-loadbalancer
--load-balancer-port 443 --policy-names my-SSLNegotiation-policy
```

**Note**

`set-load-balancer-policies-of-listener` コマンドは、指定されたロードバランサーのポートの現在のポリシーのセットを、指定されたポリシーのセットで上書きします。 `--policy-names` リストには、有効にするすべてのポリシーを含める必要があります。現在有効なポリシーを省略すると、そのポリシーは無効になります。

5. (オプション) 次の [describe-load-balancers](#) コマンドを使用して、ポリシーが有効化されていることを確認します。

```
aws elb describe-load-balancers --load-balancer-name my-loadbalancer
```

ポリシーがポート 443 で有効になっていることを示す応答の例を、次に示します。

```
{
  "LoadBalancerDescriptions": [
    {
      ....
      "ListenerDescriptions": [
        {
          "Listener": {
            "InstancePort": 80,
            "SSLCertificateId": "ARN",
            "LoadBalancerPort": 443,
            "Protocol": "HTTPS",
            "InstanceProtocol": "HTTP"
          },
          "PolicyNames": [
            "my-SSLNegotiation-policy"
          ]
        },
        {
```

```
        "Listener": {
            "InstancePort": 80,
            "LoadBalancerPort": 80,
            "Protocol": "HTTP",
            "InstanceProtocol": "HTTP"
        },
        "PolicyNames": []
    }
},
...
}
]
```

カスタムセキュリティポリシーを作成するとき、プロトコルと暗号を少なくとも1つずつ有効にする必要があります。DSA および RSA の暗号は署名アルゴリズムに固有であり、SSL 証明書を作成するために使用されます。すでに SSL 証明書がある場合は、証明書を作成するときに使用された暗号を有効にします。カスタムポリシーの名前は、ELBSecurityPolicy- や ELBSample- で始めないでください。これらのプレフィックスは、事前定義されたセキュリティポリシーの名前用に予約されているためです。

(オプション) カスタム SSL セキュリティポリシーを使用するには

1. [create-load-balancer-policy](#) コマンドを使用して、カスタムセキュリティポリシーを利用する SSL ネゴシエーションポリシーを作成します。以下に例を示します。

```
aws elb create-load-balancer-policy --load-balancer-name my-loadbalancer
--policy-name my-SSLNegotiation-policy --policy-type-name
SSLNegotiationPolicyType
--policy-attributes AttributeName=Protocol-TLSv1.2,AttributeValue=true
AttributeName=Protocol-TLSv1.1,AttributeValue=true
AttributeName=DHE-RSA-AES256-SHA256,AttributeValue=true
AttributeName=Server-Defined-Cipher-Order,AttributeValue=true
```

2. (オプション) 次の [describe-load-balancer-policies](#) コマンドを使用して、ポリシーが作成されたことを確認します。

```
aws elb describe-load-balancer-policies --load-balancer-name my-loadbalancer --
policy-name my-SSLNegotiation-policy
```

応答には、ポリシーの説明が含まれます。

3. 次の [set-load-balancer-policies-of-listener](#) コマンドを使用して、ポリシーをロードバランサーのポート 443 で有効にします。

```
aws elb set-load-balancer-policies-of-listener --load-balancer-name my-loadbalancer
--load-balancer-port 443 --policy-names my-SSLNegotiation-policy
```

**Note**

`set-load-balancer-policies-of-listener` コマンドは、指定されたロードバランサーのポートの現在のポリシーのセットを、指定されたポリシーのセットで上書きします。 `--policy-names` リストには、有効にするすべてのポリシーを含める必要があります。現在有効なポリシーを省略すると、そのポリシーは無効になります。

4. (オプション) 次の [describe-load-balancers](#) コマンドを使用して、ポリシーが有効化されていることを確認します。

```
aws elb describe-load-balancers --load-balancer-name my-loadbalancer
```

ポリシーがポート 443 で有効になっていることを示す応答の例を、次に示します。

```
{
  "LoadBalancerDescriptions": [
    {
      ....
      "ListenerDescriptions": [
        {
          "Listener": {
            "InstancePort": 80,
            "SSLCertificateId": "ARN",
            "LoadBalancerPort": 443,
            "Protocol": "HTTPS",
            "InstanceProtocol": "HTTP"
          },
          "PolicyNames": [
            "my-SSLNegotiation-policy"
          ]
        },
        {
```

```
        "Listener": {
            "InstancePort": 80,
            "LoadBalancerPort": 80,
            "Protocol": "HTTP",
            "InstanceProtocol": "HTTP"
        },
        "PolicyNames": []
    }
},
...
}
]
```

### ステップ 3: バックエンドインスタンス認証を設定する (オプション)

バックエンド接続に HTTPS/SSL をセットアップした場合、オプションでインスタンスの認証を設定できます。

バックエンドインスタンス認証をセットアップする際、パブリックキーポリシーを作成します。次に、このパブリックキーポリシーを使用して、バックエンドインスタンス認証ポリシーを作成します。最後に、HTTPS プロトコルのインスタンスポートでバックエンドインスタンス認証ポリシーを設定します。

ロードバランサーは、インスタンスがロードバランサーに提示するパブリックキーがロードバランサーの認証ポリシーのパブリックキーと一致する場合のみ、インスタンスと通信します。

バックエンドインスタンス認証を設定するには

1. 次のコマンドを使用してパブリックキーを取得します。

```
openssl x509 -in your X509 certificate PublicKey -pubkey -noout
```

2. 次の [create-load-balancer-policy](#) コマンドを使用して、パブリックキーポリシーを作成します。

```
aws elb create-load-balancer-policy --load-balancer-name my-loadbalancer --policy-name my-PublicKey-policy \
--policy-type-name PublicKeyPolicyType --policy-attributes
Attribute=PublicKey,AttributeValue=MIICiTCCAfICCQD6m7oRw0uX0jANBgkqhkiG9w
0BAQUFADCBiDELMakGA1UEBhMVCVVMxCzAJBgNVBAgTA1dBMRAwDgYDVQQHEwdTZ
WF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xZDASBgNVBAwTC01BTSBDb25zb2x1MRIw
```

```
EAYDVQQDEw1UZxN0Q21sYWMxHzAdBgkqhkiG9w0BCQEWEG5vb251QGftYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcNMTEwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQHQEwdTZWF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBA5TC01BTSBDb25zb2x1MRIwEAYDVQQDEw1UZxN0Q21sYWMxHzAdBgkqhkiG9w0BCQEWEG5vb251QGftYXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEI03IyNoH/f0wYK8m9TrDHudUZg3qX4waLG5M43q7Wgc/MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpEiBb30hjZnzcVQAaRHhd1QWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0FkbFFBjvSfpJI1J00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp3780D8uTs7fLvJx79LjSTbNYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
```

### Note

--policy-attributes にパブリックキーの値を指定するには、パブリックキーの先頭の行と末尾の行 (「-----BEGIN PUBLIC KEY-----」を含む行と「-----END PUBLIC KEY-----」を含む行) を削除します。AWS CLI では、空白文字は使用できません--policy-attributes。

- 次の [create-load-balancer-policy](#) コマンドを使用し、my-PublicKey-policy によりバックエンドインスタンス認証ポリシーを作成します。

```
aws elb create-load-balancer-policy --load-balancer-name my-Loadbalancer --policy-name my-authentication-policy --policy-type-name BackendServerAuthenticationPolicyType --policy-attributes AttributeName=PublicKeyPolicyName,AttributeValue=my-PublicKey-policy
```

オプションで複数のパブリックキーポリシーを仕様できます。ロードバランサーはすべてのキーを1つずつ試して認証を行います。インスタンスが提示するパブリックキーがこれらのパブリックキーのいずれか1つに一致すれば、インスタンスは認証されます。

- 次の [set-load-balancer-policies-for-backend-server](#) コマンドを使用して、my-authentication-policy を HTTPS のインスタンスポートに設定します。この例では、インスタンスポートはポート 443 です。

```
aws elb set-load-balancer-policies-for-backend-server --load-balancer-name my-Loadbalancer --instance-port 443 --policy-names my-authentication-policy
```

5. (オプション) 次の [describe-load-balancer-policies](#) コマンドを使用して、ロードバランサーのすべてのポリシーを一覧表示します。

```
aws elb describe-load-balancer-policies --load-balancer-name my-loadbalancer
```

6. (オプション) 次の [describe-load-balancer-policies](#) コマンドを使用して、ポリシーの詳細を表示します。

```
aws elb describe-load-balancer-policies --load-balancer-name my-loadbalancer --  
policy-names my-authentication-policy
```

## ステップ 4: ヘルスチェックを設定する (オプション)

Elastic Load Balancing は、ユーザーが作成したヘルスチェック設定に基づいて、登録されている各 EC2 インスタンスの状態を定期的にチェックします。Elastic Load Balancing で 問題のあるインスタンスが検出されると、そのインスタンスへのトラフィックの送信は停止され、正常なインスタンスにトラフィックがルーティングされます。詳細については、「[Classic Load Balancer のインスタンスのヘルスチェック](#)」を参照してください。

ロードバランサーの作成時、Elastic Load Balancing のヘルスチェックにはデフォルト設定が使用されます。デフォルト設定を使用する代わりに、必要に応じて、ロードバランサーのヘルスチェック設定を変更できます。

インスタンスのヘルスチェックを設定するには

次の [configure-health-check](#) コマンドを使用します。

```
aws elb configure-health-check --load-balancer-name my-loadbalancer --health-check  
Target=HTTP:80/ping,Interval=30,UnhealthyThreshold=2,HealthyThreshold=2,Timeout=3
```

以下に、応答の例を示します。

```
{  
  "HealthCheck": {  
    "HealthyThreshold": 2,  
    "Interval": 30,  
    "Target": "HTTP:80/ping",  
    "Timeout": 3,  
    "UnhealthyThreshold": 2
```

```
}  
}
```

## ステップ 5: EC2 インスタンスを登録する

ロードバランサーを作成したら、そのロードバランサーに EC2 インスタンスを登録する必要があります。同じリージョン内の 1 つまたは複数のアベイラビリティゾーンにある EC2 インスタンスをロードバランサーとして選択できます。詳細については、「[Classic Load Balancer の登録済みインスタンス](#)」を参照してください。

[register-instances-with-load-balancer](#) コマンドを次のように使用します。

```
aws elb register-instances-with-load-balancer --load-balancer-name my-loadbalancer --  
instances i-4f8cf126 i-0bb7ca62
```

以下に、応答の例を示します。

```
{  
  "Instances": [  
    {  
      "InstanceId": "i-4f8cf126"  
    },  
    {  
      "InstanceId": "i-0bb7ca62"  
    }  
  ]  
}
```

## ステップ 6: インスタンスを検証する

登録したインスタンスのいずれかが InService 状態になると、ロードバランサーを使用できるようになります。

新しく登録した EC2 インスタンスの状態を確認するには、次の [describe-instance-health](#) コマンドを使用します。

```
aws elb describe-instance-health --load-balancer-name my-loadbalancer --  
instances i-4f8cf126 i-0bb7ca62
```

以下に、応答の例を示します。

```
{
  "InstanceStates": [
    {
      "InstanceId": "i-4f8cf126",
      "ReasonCode": "N/A",
      "State": "InService",
      "Description": "N/A"
    },
    {
      "InstanceId": "i-0bb7ca62",
      "ReasonCode": "Instance",
      "State": "OutOfService",
      "Description": "Instance registration is still in progress"
    }
  ]
}
```

インスタンスの State フィールドが OutOfService である場合は、インスタンスがまだ登録中の可能性があります。詳細については、「[Classic Load Balancer のトラブルシューティング: インスタンスの登録](#)」を参照してください。

少なくとも 1 つのインスタンスの状態が InService であれば、ロードバランサーをテストできます。ロードバランサーをテストするには、ロードバランサーの DNS 名をコピーして、インターネットに接続されているウェブブラウザのアドレスフィールドに貼り付けます。ロードバランサーが実行中の場合は、HTTP サーバーのデフォルトページが表示されます。

## ステップ 7: ロードバランサーを削除する (オプション)

ロードバランサーを削除すると、関連する EC2 インスタンスが自動的に登録解除されます。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。ただし、EC2 インスタンスは引き続き実行され、利用料金も継続して発生します。

ロードバランサーを削除するには、次の [delete-load-balancer](#) コマンドを使用します。

```
aws elb delete-load-balancer --load-balancer-name my-loadbalancer
```

EC2 インスタンスを停止するには、[stop-instances](#) コマンドを使用します。EC2 インスタンスを終了するには、[terminate-instances](#) コマンドを使用します。

## Classic Load Balancer の HTTPS リスナーの設定

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。リスナーは、フロントエンド (クライアントからロードバランサー) 接続用のプロトコルとポート、およびバックエンド (ロードバランサーからインスタンス) 接続用のプロトコルとポートを使用して設定します。Elastic Load Balancing でサポートされるポート、プロトコル、およびリスナー設定の詳細については、「[Classic Load Balancer のリスナー](#)」を参照してください。

ポート 80 で HTTP リクエストを受け付けるリスナーを持つロードバランサーがある場合は、ポート 443 で HTTPS リクエストを受け付けるリスナーを追加できます。HTTPS リスナーがポート 80 でインスタンスに SSL リクエストを送信する方法を指定すると、ロードバランサーはリクエストを終了し、ロードバランサーからインスタンスへの通信は暗号化されません。HTTPS リスナーがポート 443 でインスタンスにリクエストを送信する場合は、ロードバランサーからインスタンスへの通信は暗号化されます。

ロードバランサーがインスタンスとの通信に暗号化された接続を使用する場合、オプションでインスタンスでの認証を有効にできます。これにより、パブリックキーとロードバランサーにこのために指定したキーが一致した場合に限り、ロードバランサーがインスタンスと通信できます。

新しい HTTPS リスナーの作成については、「[HTTPS リスナーを使用する Classic Load Balancer の作成](#)」を参照してください。

### 内容

- [前提条件](#)
- [コンソールを使用した HTTPS リスナーの追加](#)
- [を使用して HTTPS リスナーを追加する AWS CLI](#)

## 前提条件

HTTPS リスナーの HTTPS サポートを有効にするには、ロードバランサーに SSL サーバー証明書をデプロイする必要があります。インスタンスにリクエストを送信する前に、ロードバランサーは証明書を使用してリクエストを終了し、次にリクエストを復号します。SSL 証明書がない場合は作成します。詳細については、「[Classic Load Balancer 用の SSL/TLS 証明書](#)」を参照してください。

## コンソールを使用した HTTPS リスナーの追加

既存のロードバランサーに、HTTPS リスナーを追加できます。

コンソールを使用してロードバランサーに HTTPS リスナーを追加するには

1. Amazon EC2 コンソールの <https://console.aws.amazon.com/ec2/> を開いてください。
2. ナビゲーションペインの [ロードバランシング] で [ロードバランサー] を選択します。
3. 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. [リスナー] タブで、[リスナーを追加] を選択します。
5. [リスナーを管理] ページの [リスナー] セクションで、[リスナーの追加] を選択します。
6. [リスナープロトコル] で [HTTPS] を選択します。

**⚠ Important**

デフォルトでは、インスタンスプロトコルは HTTP です。バックエンドインスタンス認証を設定するには、インスタンスプロトコルを HTTPS に変更します。

7. [セキュリティポリシー] には、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。別の事前定義されたセキュリティポリシーを使用する場合や、カスタムポリシーを作成する場合は、「[SSL ネゴシエーション設定の更新](#)」を参照してください。
8. [デフォルトの SSL 証明書] で、[編集] を選択し、次のいずれかの操作を行います。
  - 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 から選択し、リストから証明書を選択し、変更を保存 を選択します。

**i Note**

このオプションは、がサポートされているリージョンでのみ使用できます AWS Certificate Manager

- IAM を使用して証明書をインポートした場合は、[IAM から] を選択し、リストから証明書を選択して、[変更内容の保存] を選択します。
- ACM にインポートする SSL 証明書がある場合は、[インポート] と [ACM へ] を選択します。[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いて、ブラウザがその証明書を暗黙的に受け入れることが重要ではない場合以外は、[証明書チェーン - オプション] 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。

- インポートする SSL 証明書があるが、このリージョンで ACM がサポートされていない場合は、[インポート] と [IAM へ] を選択します。[証明書名] 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いて、ブラウザーがその証明書を暗黙的に受け入れることが重要ではない場合以外は、[証明書チェーン - オプション] 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。
  - [Save changes] (変更の保存) をクリックします。
9. Cookie ステイックネスの場合、デフォルトは無効です。これを変更するには、[編集] を選択します。[ロードバランサーによって生成] を選択した場合は、[有効期限] を指定する必要があります。[アプリケーションによって生成] を選択した場合は、Cookie 名 を指定する必要があります。選択後、[変更を保存] を選択します。。
  10. (オプション) [リスナーを追加] を選択してその他のリスナーを追加します。
  11. [変更を保存] を選択して、先ほど設定したリスナーを追加します。
  12. (オプション) 既存のロードバランサーのバックエンドインスタンス認証を設定するには、AWS CLI または API を使用する必要があります。このタスクは コンソールではサポートされていないためです。詳細については、「[バックエンドインスタンス認証の設定](#)」を参照してください。

## を使用して HTTPS リスナーを追加する AWS CLI

既存のロードバランサーに、HTTPS リスナーを追加できます。

を使用してロードバランサーに HTTPS リスナーを追加するには AWS CLI

1. SSL 証明書の Amazon リソースネーム (ARN) を取得します。以下に例を示します。

ACM

```
arn:aws:acm:region:123456789012:certificate/12345678-1234-1234-1234-123456789012
```

IAM

```
arn:aws:iam::123456789012:server-certificate/my-server-certificate
```

2. (ポート 443 で HTTPS リクエストを受け付け、そのリクエストを HTTP を使用してポート 80 でインスタンスに送信している) ロードバランサーに対しリスナーを追加するには、次の [create-load-balancer-listeners](#) コマンドを使用します。

```
aws elb create-load-balancer-listeners --load-balancer-name my-load-balancer --  
listeners  
Protocol=HTTPS,LoadBalancerPort=443,InstanceProtocol=HTTP,InstancePort=80,SSLCertificateId
```

バックエンドインスタンス認証をセットアップするには、次のコマンドを使用して、ポート 443 で HTTPS リクエストを受け付け、HTTPS を使用してポート 443 でインスタンスにリクエストを送信するリスナーを追加します。

```
aws elb create-load-balancer-listeners --load-balancer-name my-load-balancer --  
listeners  
Protocol=HTTPS,LoadBalancerPort=443,InstanceProtocol=HTTPS,InstancePort=443,SSLCertificate
```

3. (オプション) 次の [describe-load-balancers](#) コマンドを使用して、ロードバランサーの更新後の詳細を表示できます。

```
aws elb describe-load-balancers --load-balancer-name my-load-balancer
```

以下に、応答の例を示します。

```
{  
  "LoadBalancerDescriptions": [  
    {  
      ...  
      "ListenerDescriptions": [  
        {  
          "Listener": {  
            "InstancePort": 80,  
            "SSLCertificateId": "ARN",  
            "LoadBalancerPort": 443,  
            "Protocol": "HTTPS",  
            "InstanceProtocol": "HTTP"  
          },  
          "PolicyNames": [  
            "ELBSecurityPolicy-2016-08"  
          ]  
        },  
        {  
          "Listener": {  
            "InstancePort": 80,  
            "LoadBalancerPort": 80,
```

```
        "Protocol": "HTTP",
        "InstanceProtocol": "HTTP"
    },
    "PolicyNames": []
}
],
...
}
]
```

- (オプション) HTTPS リスナーは、デフォルトのセキュリティポリシーを使用して作成されています。異なる事前定義済みセキュリティポリシーや、カスタムのセキュリティポリシーを指定する場合は、[create-load-balancer-policy](#) コマンドおよび [set-load-balancer-policies-of-listener](#) コマンドを使用します。詳細については、「[を使用して SSL ネゴシエーション設定を更新する AWS CLI](#)」を参照してください。
- (オプション) バックエンドインスタンス認証をセットアップするには、[set-load-balancer-policies-for-backend-server](#) コマンドを使用します。詳細については、「[バックエンドインスタンス認証の設定](#)」を参照してください。

## Classic Load Balancer の SSL 証明書の置き換え

HTTPS リスナーがある場合、リスナー作成時に SSL サーバー証明書をロードバランサーにデプロイしました。各証明書には有効期間が記載されています。有効期間が終わる前に、証明書を更新するか、置き換える必要があります。

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。ロードバランサーにデプロイされた証明書の更新後、新しいリクエストでは更新された証明書が使用されます。

証明書を置き換えるには、現在の証明書を作成したときと同じステップで、最初に新しい証明書を作成する必要があります。その後、証明書を置き換えることができます。ロードバランサーにデプロイされた証明書の置き換え後、新しいリクエストでは新しい証明書が使用されます。

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によって既に受け取られ、正常なターゲットへのルーティングを保留中のリクエストには影響しません。

## 目次

- [コンソールを使用した SSL 証明書の置き換え](#)
- [を使用して SSL 証明書を置き換える AWS CLI](#)

## コンソールを使用した SSL 証明書の置き換え

ロードバランサーにデプロイされた証明書は、ACM が提供する証明書、または IAM にアップロードされた証明書で置き換えることができます。

コンソールを使用して HTTPS ロードバランサーの SSL 証明書を置き換えるには

1. Amazon EC2 コンソールの <https://console.aws.amazon.com/ec2/> を開いてください。
2. ナビゲーションペインの [ロードバランシング] で [ロードバランサー] を選択します。
3. 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. [リスナー] タブで、[リスナーを追加] を選択します。
5. リスナーを管理ページで、更新するリスナーを探し、[デフォルトの SSL 証明書] の下の [編集] を選択し、次のいずれかを実行します。
  - 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 から選択し、リストから証明書を選択し、変更を保存 を選択します。

### Note

このオプションは、がサポートされているリージョンでのみ使用できます AWS Certificate Manager

- IAM を使用して証明書をインポートした場合は、[IAM から] を選択し、リストから証明書を選択して、[変更内容の保存] を選択します。
- ACM にインポートする SSL 証明書がある場合は、[インポート] と [ACM へ] を選択します。[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いて、ブラウザがその証明書を暗黙的に受け入れることが重要ではない場合以外は、[証明書チェーン - オブ

ション] 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。

- インポートする SSL 証明書があるが、このリージョンで ACM がサポートされていない場合は、[インポート] と [IAM へ] を選択します。[証明書名] 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いて、ブラウザがその証明書を暗黙的に受け入れることが重要ではない場合以外は、[証明書チェーン - オプション] 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。
- [Save changes] (変更の保存) をクリックします。

## を使用して SSL 証明書を置き換える AWS CLI

ロードバランサーにデプロイされた証明書は、ACM が提供する証明書、または IAM にアップロードされた証明書で置き換えることができます。

SSL 証明書を ACM が提供する証明書で置き換えるには

1. 次の [request-certificate](#) コマンドを使用して、新しい証明書をリクエストします。

```
aws acm request-certificate --domain-name www.example.com
```

2. 次の [set-load-balancer-listener-ssl-certificate](#) コマンドを使用して、証明書を設定します。

```
aws elb set-load-balancer-listener-ssl-certificate --load-balancer-name my-load-balancer --load-balancer-port 443 --ssl-certificate-id arn:aws:acm:region:123456789012:certificate/12345678-1234-1234-1234-123456789012
```

SSL 証明書を IAM にアップロードされた証明書で置き換えるには

1. SSL 証明書を保持しておりまだアップロードしていない場合は、IAM ユーザーガイドの「[サーバー証明書のアップロード](#)」を参照してください。
2. 次の [get-server-certificate](#) コマンドを使用して、証明書の ARN を取得します。

```
aws iam get-server-certificate --server-certificate-name my-new-certificate
```

3. 次の [set-load-balancer-listener-ssl-certificate](#) コマンドを使用して、証明書を設定します。

```
aws elb set-load-balancer-listener-ssl-certificate --load-balancer-name my-load-balancer --load-balancer-port 443 --ssl-certificate-id arn:aws:iam::123456789012:server-certificate/my-new-certificate
```

## Classic Load Balancer の SSL ネゴシエーション設定の更新

Elastic Load Balancing は、クライアントとロードバランサーの間の SSL 接続でのネゴシエーションに使用する SSL ネゴシエーション設定を事前定義した、セキュリティポリシーを提供します。リスナーに HTTPS/SSL プロトコルを使用する場合は、事前定義されたセキュリティポリシーのいずれか、または独自のカスタムセキュリティポリシーを使用できます。

セキュリティポリシーの詳細については、「[Classic Load Balancers での SSL ネゴシエーション設定](#)」を参照してください。Elastic Load Balancing で提供されるセキュリティポリシーの設定については、「[Classic Load Balancer 用の事前定義済み SSL セキュリティポリシー](#)」を参照してください。

セキュリティポリシーを関連付けずに HTTPS/SSL リスナーを作成した場合、事前定義されたデフォルトのセキュリティポリシー `ELBSecurityPolicy-2016-08` が、Elastic Load Balancing によりロードバランサーに関連付けられます。

希望に応じて、カスタム設定を作成できます。ロードバランサー設定をアップグレードする前に、セキュリティポリシーをテストすることを強くお勧めします。

次の例は、HTTPS/SSL リスナーの SSL ネゴシエーション設定を更新する方法を示しています。変更は、ロードバランサーノードが受け取ったリクエスト及び正常なインスタンスへのルーティング待ちリクエストに影響を与えません。変更された設定は新規のリクエストに対して適用されます。

### 目次

- [コンソールを使用した SSL ネゴシエーション設定の更新](#)
- [を使用して SSL ネゴシエーション設定を更新する AWS CLI](#)

## コンソールを使用した SSL ネゴシエーション設定の更新

デフォルトでは、Elastic Load Balancing は事前定義された最新のセキュリティポリシーをロードバランサーに関連付けます。新しい定義済みポリシーが追加された場合は、新しい定義済みポリシーを使用するようにロードバランサーを更新することをお勧めします。また、別の事前定義されたセキュリティポリシーやカスタムポリシーを選択することもできます。

コンソールを使用して HTTPS/SSL ロードバランサーの SSL ネゴシエーション設定を更新するには

1. Amazon EC2 コンソールの <https://console.aws.amazon.com/ec2/> を開いてください。
2. ナビゲーションペインの [ロードバランシング] で [ロードバランサー] を選択します。
3. 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. [リスナー] タブで、[リスナーを追加] を選択します。
5. [リスナーを管理] ページで、更新するリスナーを見つけ、[セキュリティポリシー] で [編集] を選択します。
  - デフォルトのポリシー [ELBSecurityPolicy-2016-08] をそのまま使用し、[変更内容の保存] を選択します。
  - デフォルト以外の定義済みポリシーを選択し、[変更内容の保存] を選択します。
  - [カスタム] を選択し、次のようにプロトコルと暗号を少なくとも 1 つずつ有効にします。
    - a. [SSL Protocols] で、有効にするプロトコルを 1 つ以上選択します。
    - b. [Classic Load Balancer 用の事前定義済み SSL セキュリティポリシー](#) にリストされた順序で SSL ネゴシエーションに使用するには、[SSL オプション] で [サーバーの優先順位] をクリックします。
    - c. [SSL Ciphers] で、有効にする暗号を 1 つ以上選択します。既に SSL 証明書がある場合、DSA および RSA 暗号化は署名アルゴリズムに固有になるため、その証明書を作成する際に使用した暗号を有効にする必要があります。
    - d. [Save changes] (変更の保存) をクリックします。

## を使用して SSL ネゴシエーション設定を更新する AWS CLI

事前定義されたデフォルトのセキュリティポリシー ELBSecurityPolicy-2016-08、別の事前定義されたセキュリティポリシー、またはカスタムセキュリティポリシーを使用できます。

事前定義された SSL セキュリティポリシーを使用するには

1. 次の [describe-load-balancer-policies](#) コマンドを使用して、Elastic Load Balancing から提供された、事前定義済みセキュリティポリシーを一覧表示します。使用する構文は、使用しているオペレーティングシステムとシェルによって異なります。

Linux

```
aws elb describe-load-balancer-policies --query 'PolicyDescriptions[?
PolicyTypeName==`SSLNegotiationPolicyType`].{PolicyName:PolicyName}' --output table
```

## Windows

```
aws elb describe-load-balancer-policies --query "PolicyDescriptions[?
PolicyTypeName==`SSLNegotiationPolicyType`].{PolicyName:PolicyName}" --output table
```

出力例を次に示します。

```
-----
| DescribeLoadBalancerPolicies |
+-----+
| PolicyName |
+-----+
| ELBSecurityPolicy-2016-08 |
| ELBSecurityPolicy-TLS-1-2-2017-01 |
| ELBSecurityPolicy-TLS-1-1-2017-01 |
| ELBSecurityPolicy-2015-05 |
| ELBSecurityPolicy-2015-03 |
| ELBSecurityPolicy-2015-02 |
| ELBSecurityPolicy-2014-10 |
| ELBSecurityPolicy-2014-01 |
| ELBSecurityPolicy-2011-08 |
| ELBSample-ELBDefaultCipherPolicy |
| ELBSample-OpenSSLDefaultCipherPolicy |
+-----+
```

ポリシーで有効な暗号を確認するには、次のコマンドを使用します。

```
aws elb describe-load-balancer-policies --policy-names ELBSecurityPolicy-2016-08 --
output table
```

事前定義されたセキュリティポリシーの設定については、「[Classic Load Balancer 用の事前定義済み SSL セキュリティポリシー](#)」を参照してください。

2. [create-load-balancer-policy](#) コマンドを使用して、前のステップで一覧表示した事前定義済みセキュリティポリシーのいずれかを使用する、SSL ネゴシエーションポリシーを作成します。たとえば、次のコマンドでは事前定義されたデフォルトのセキュリティポリシーが使用されます。

```
aws elb create-load-balancer-policy --load-balancer-name my-loadbalancer
--policy-name my-SSLNegotiation-policy --policy-type-name SSLNegotiationPolicyType
--policy-attributes AttributeName=Reference-Security-
Policy,AttributeValue=ELBSecurityPolicy-2016-08
```

ロードバランサーのポリシー数の制限を超えた場合は、[delete-load-balancer-policy](#) コマンドを使用して、未使用のポリシーを削除します。

3. (オプション) 次の [describe-load-balancer-policies](#) コマンドを使用して、ポリシーが作成されたことを確認します。

```
aws elb describe-load-balancer-policies --load-balancer-name my-loadbalancer --
policy-name my-SSLNegotiation-policy
```

応答には、ポリシーの説明が含まれます。

4. 次の [set-load-balancer-policies-of-listener](#) コマンドを使用して、ポリシーをロードバランサーのポート 443 で有効にします。

```
aws elb set-load-balancer-policies-of-listener --load-balancer-name my-loadbalancer
--load-balancer-port 443 --policy-names my-SSLNegotiation-policy
```

#### Note

`set-load-balancer-policies-of-listener` コマンドは、指定されたロードバランサーのポートの現在のポリシーのセットを、指定されたポリシーのセットで上書きします。 `--policy-names` リストには、有効にするすべてのポリシーを含める必要があります。現在有効なポリシーを省略すると、そのポリシーは無効になります。

5. (オプション) 次の [describe-load-balancers](#) コマンドを使用して、新しいポリシーが、ロードバランサーのポートで有効化されていることを確認します。

```
aws elb describe-load-balancers --load-balancer-name my-loadbalancer
```

応答には、ポリシーがポート 443 で有効になっていることが示されます。

```
...
{
  "Listener": {
```

```

        "InstancePort": 443,
        "SSLCertificateId": "ARN",
        "LoadBalancerPort": 443,
        "Protocol": "HTTPS",
        "InstanceProtocol": "HTTPS"
    },
    "PolicyNames": [
        "my-SSLNegotiation-policy"
    ]
}
...

```

カスタムセキュリティポリシーを作成するとき、プロトコルと暗号を少なくとも1つずつ有効にする必要があります。DSA および RSA の暗号は署名アルゴリズムに固有であり、SSL 証明書を作成するために使用されます。すでに SSL 証明書がある場合は、証明書を作成するときに使用された暗号を有効にします。カスタムポリシーの名前は、ELBSecurityPolicy- や ELBSample- で始めないでください。これらのプレフィックスは、事前定義されたセキュリティポリシーの名前用に予約されているためです。

カスタム SSL セキュリティポリシーを使用するには

1. [create-load-balancer-policy](#) コマンドを使用して、カスタムセキュリティポリシーを利用する SSL ネゴシエーションポリシーを作成します。以下に例を示します。

```

aws elb create-load-balancer-policy --load-balancer-name my-loadbalancer
--policy-name my-SSLNegotiation-policy --policy-type-name
SSLNegotiationPolicyType
--policy-attributes AttributeName=Protocol-TLSv1.2,AttributeValue=true
AttributeName=Protocol-TLSv1.1,AttributeValue=true
AttributeName=DHE-RSA-AES256-SHA256,AttributeValue=true
AttributeName=Server-Defined-Cipher-Order,AttributeValue=true

```

ロードバランサーのポリシー数の制限を超えた場合は、[delete-load-balancer-policy](#) コマンドを使用して、未使用のポリシーを削除します。

2. (オプション) 次の [describe-load-balancer-policies](#) コマンドを使用して、ポリシーが作成されたことを確認します。

```

aws elb describe-load-balancer-policies --load-balancer-name my-loadbalancer --
policy-name my-SSLNegotiation-policy

```

応答には、ポリシーの説明が含まれます。

3. 次の [set-load-balancer-policies-of-listener](#) コマンドを使用して、ポリシーをロードバランサーのポート 443 で有効にします。

```
aws elb set-load-balancer-policies-of-listener --load-balancer-name my-loadbalancer
--load-balancer-port 443 --policy-names my-SSLNegotiation-policy
```

**Note**

`set-load-balancer-policies-of-listener` コマンドは、指定されたロードバランサーのポートの現在のポリシーのセットを、指定されたポリシーのセットで上書きします。 `--policy-names` リストには、有効にするすべてのポリシーを含める必要があります。現在有効なポリシーを省略すると、そのポリシーは無効になります。

4. (オプション) 次の [describe-load-balancers](#) コマンドを使用して、新しいポリシーが、ロードバランサーのポートで有効化されていることを確認します。

```
aws elb describe-load-balancers --load-balancer-name my-loadbalancer
```

応答には、ポリシーがポート 443 で有効になっていることが示されます。

```
...
{
  "Listener": {
    "InstancePort": 443,
    "SSLCertificateId": "ARN",
    "LoadBalancerPort": 443,
    "Protocol": "HTTPS",
    "InstanceProtocol": "HTTPS"
  },
  "PolicyNames": [
    "my-SSLNegotiation-policy"
  ]
}
...
```

# Classic Load Balancer の登録済みインスタンス

Classic Load Balancer を作成した後、そのロードバランサーに対し EC2 インスタンスを登録する必要があります。同じリージョン内の 1 つまたは複数のアベイラビリティゾーンにある EC2 インスタンスをロードバランサーとして選択できます。Elastic Load Balancing は、登録済み EC2 インスタンスに関するヘルスチェックを定期的に行い、正常な登録済み EC2 インスタンスに対し、ロードバランサーの DNS 名への受信リクエストを自動的に分散させます。

## 目次

- [インスタンスのベストプラクティス](#)
- [VPC の推奨事項](#)
- [Classic Load Balancer に EC2 インスタンスを登録するには](#)
- [Classic Load Balancer のインスタンスのヘルスチェック](#)
- [Classic Load Balancer のインスタンスのセキュリティグループ](#)
- [Classic Load Balancer のインスタンスのネットワーク ACL](#)

## インスタンスのベストプラクティス

- 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、インスタンスと通信できることを確認する必要があります。詳細については、「[Classic Load Balancer のセキュリティグループの設定](#)」を参照してください。インスタンスのセキュリティグループは、両方のポート（ロードバランサー用の各サブネットのポート）で、双方向のトラフィックを許可する必要があります。
- ロードバランサーに登録するすべてのインスタンスに Apache や Internet Information Services (IIS) などのウェブサーバーをインストールします。
- HTTP および HTTPS リスナーを使用する場合は、ロードバランサーが複数のクライアントリクエストに対応するためインスタンスへの接続を再利用できるようにするキープアライブオプションを EC2 インスタンスで有効にすることをお勧めします。これによってウェブサーバーの負荷が軽減し、ロードバランサーのスループットが向上します。キープアライブのタイムアウトは、ロードバランサーがインスタンスへの接続を確実に閉じることができるよう、60 秒以上にする必要があります。
- Elastic Load Balancing では、パスの最大送信単位 (MTU) 検出をサポートしています。パス MTU 検出が正しく機能するようにするため、インスタンスのセキュリティグループで ICMP フラグメ

ント化に必要な (タイプ 3、コード 4) メッセージが許可されるようにする必要があります。詳細については、「Amazon EC2 ユーザーガイド」の「[パス MTU 検出](#)」を参照してください。

## VPC の推奨事項

### Virtual Private Cloud (VPC)

2014 年 AWS アカウント より前に を作成していない限り、各リージョンにデフォルトの VPC があります。ロードバランサーには既存のデフォルト VPC を使用するが、新しい VPC を作成できます。詳細については、[Amazon VPC ユーザーガイド](#)を参照してください。

### ロードバランサーのサブネット

ロードバランサーが正しくスケーリングできるように、ロードバランサー毎に各サブネットの 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 8 個の空き IP アドレスを用意してください。ロードバランサーはこれらの IP アドレスを使用してインスタンスとの接続を確立し、必要に応じてスケールアウトします。IP アドレスが不十分な場合、ロードバランサーがスケーリングできず、容量不足が原因で 503 エラーが発生する可能性があります。

インスタンスを起動する各アベイラビリティゾーンにサブネットを作成します。アプリケーションに応じて、パブリックサブネット、プライベートサブネット、またはパブリックサブネットとプライベートサブネットの組み合わせでインスタンスを起動できます。パブリックサブネットには、インターネットゲートウェイへのルートがあります。デフォルト VPC には、アベイラビリティゾーンごとにデフォルトで 1 つのパブリックサブネットがあることに注意してください。

ロードバランサーを作成する場合は、ロードバランサーに 1 つ以上のパブリックサブネットを追加する必要があります。インスタンスがプライベートサブネットにある場合、インスタンスのサブネットと同じアベイラビリティゾーンにパブリックサブネットを作成し、このパブリックサブネットをロードバランサーに追加します。

### ネットワーク ACL

VPC のネットワーク ACL では、リスナーポートおよびヘルスチェックポートで両方向のトラフィックが許可される必要があります。詳細については、「[Classic Load Balancer のインスタンスのネットワーク ACL](#)」を参照してください。

## Classic Load Balancer に EC2 インスタンスを登録するには

EC2 インスタンスを登録すると、ロードバランサーに追加されます。ロードバランサーは、登録済みインスタンスの状態を、有効になっているアベイラビリティゾーンで常に監視し、正常なインス

タンスにリクエストをルーティングします。インスタンスの需要が上昇した場合、需要に対応できるようにロードバランサーに追加インスタンスを登録できます。

EC2 インスタンスを登録解除すると、ロードバランサーから削除されます。登録解除するとすぐに、ロードバランサーはインスタンスへのリクエストのルーティングを停止します。需要が低下した場合や、インスタンスを保守する必要がある場合、ロードバランサーからインスタンスを登録解除することができます。登録解除されたインスタンスは実行され続けますが、ロードバランサーからトラフィックを受信しなくなります。準備ができたなら、再度ロードバランサーに登録することができます。

インスタンスの登録解除時に Connection Drainingが有効化されている場合は、Elastic Load Balancing は実行中のリクエストが完了するまで待機します。詳細については、「[Classic Load Balancer の Connection Draining の設定](#)」を参照してください。

ロードバランサーが Auto Scaling グループにアタッチされている場合、グループ内のインスタンスがロードバランサーに自動的に登録されます。Auto Scaling グループからロードバランサーをデタッチした場合には、グループ内のインスタンスは登録解除されます。

Elastic Load Balancing は、EC2 インスタンスの IP アドレスを使用して、そのインスタンスをロードバランサーに登録します。

[EC2-VPC] アタッチされている Elastic Network Interface (ENI) にインスタンスを登録するとき、ロードバランサーは、インスタンスのプライマリインターフェイス (eth0) のプライマリ IP アドレスにリクエストをルーティングします。

## 内容

- [インスタンスの登録](#)
- [ロードバランサーに登録されているインスタンスの表示](#)
- [インスタンスが登録されているロードバランサーの確認](#)
- [インスタンスの登録解除](#)

## インスタンスの登録

準備が整ったら、ロードバランサーにインスタンスを登録します。インスタンスがロードバランサーで有効なアベイラビリティゾーンにある場合、そのインスタンスは、必要な数のヘルスチェックに合格するとすぐに、ロードバランサーからトラフィックを受信する準備が整います。

コンソールを使用してインスタンスを登録するには

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 [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. [ターゲットインスタンス] タブで、[インスタンスを管理] を選択します。
5. [インスタンスを管理] ページの [使用可能なインスタンス] テーブル内で、ロードバランサーに登録するインスタンスを選択します。
6. 登録する必要があるインスタンスが [選択済みインスタンスを確認] テーブル内に入力されていることを確認します。
7. [Save changes] (変更の保存) をクリックします。

を使用してインスタンスを登録するには AWS CLI

次の [register-instances-with-load-balancer](#) コマンドを使用します。

```
aws elb register-instances-with-load-balancer --load-balancer-name my-loadbalancer --instances i-4e05f721
```

ロードバランサーに登録されているインスタンスを一覧表示する応答の例を次に示します。

```
{
  "Instances": [
    {
      "InstanceId": "i-315b7e51"
    },
    {
      "InstanceId": "i-4e05f721"
    }
  ]
}
```

## ロードバランサーに登録されているインスタンスの表示

指定したロードバランサーに登録されているインスタンスを一覧表示するには、次の [describe-load-balancers](#) コマンドを使用します。

```
aws elb describe-load-balancers --load-balancer-names my-load-balancer --output text --query "LoadBalancerDescriptions[*].Instances[*].InstanceId"
```

出力例を次に示します。

```
i-e905622e  
i-315b7e51  
i-4e05f721
```

## インスタンスが登録されているロードバランサーの確認

指定したインスタンスが登録されているロードバランサーの名前を取得するには、次の [describe-load-balancers](#) コマンドを使用します。

```
aws elb describe-load-balancers --output text --query "LoadBalancerDescriptions[?Instances[?InstanceId=='i-e905622e']].[LoadBalancerName]"
```

出力例を次に示します。

```
my-load-balancer
```

## インスタンスの登録解除

容量がなくなったりインスタンスを保守する必要がある場合、ロードバランサーからインスタンスを登録解除することができます。

ロードバランサーが Auto Scaling グループにアタッチされている場合、グループからインスタンスをデタッチするとロードバランサーからも登録解除されます。詳細については、Amazon EC2 Auto Scaling ユーザーガイドの「[Auto Scaling グループから EC2 インスタンスをデタッチする](#)」を参照してください。

コンソールを使用してインスタンスを登録解除するには

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 [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. [ターゲットインスタンス] タブで、[インスタンスを管理] を選択します。

5. [インスタンスを管理] ページの [使用可能なインスタンス] テーブル内で、ロードバランサーから登録を解除するインスタンスの選択を解除します。
6. 登録解除する必要があるインスタンスが [選択済みインスタンスを確認] テーブル内に入力されていないことを確認してください。
7. [Save changes] (変更の保存) をクリックします。

を使用してインスタンスの登録を解除するには AWS CLI

次の [deregister-instances-from-load-balancer](#) コマンドを使用します。

```
aws elb deregister-instances-from-load-balancer --load-balancer-name my-loadbalancer --instances i-4e05f721
```

ロードバランサーに登録されている残りのインスタンスをリスト表示する応答の例を次に示します。

```
{
  "Instances": [
    {
      "InstanceId": "i-315b7e51"
    }
  ]
}
```

## Classic Load Balancer のインスタンスのヘルスチェック

Classic Load Balancer は、登録されたインスタンス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。ヘルスチェック時に正常なインスタンスのステータスは、InService となります。ヘルスチェック時に異常があるインスタンスのステータスは、OutOfService となります。ロードバランサーは、インスタンスの状態が正常であるか異常であるかにかかわらず、すべての登録済みインスタンスでヘルスチェックを実行します。

ロードバランサーは、正常なインスタンスのみにリクエストをルーティングします。インスタンスが異常であると判断した場合、ロードバランサーはそのインスタンスへのリクエストのルーティングを中止します。インスタンスが正常な状態に戻ると、ロードバランサーはそのインスタンスへのリクエストのルーティングを再開します。

ロードバランサーは Elastic Load Balancing によって提供されるデフォルトのヘルスチェック設定、またはユーザーが作成したヘルスチェック設定を使用して、登録されたインスタンスの状態をチェックします。

Auto Scaling グループに Classic Load Balancer を関連付けておくと、ロードバランサーのヘルスチェックを使用して、Auto Scaling グループ内のインスタンスのヘルス状態を判断することが可能になります。デフォルトでは、Auto Scaling グループ内の各インスタンスのヘルス状態が定期的に判断されます。詳細については、Amazon EC2 Auto Scaling 開発者ガイドの「[Auto Scaling グループへの Elastic Load Balancing ヘルスチェックの追加](#)」を参照してください。

## 目次

- [ヘルスチェックの設定](#)
- [ヘルスチェックの設定の更新](#)
- [インスタンスのヘルスの確認](#)
- [ヘルスチェックのトラブルシューティング](#)

## ヘルスチェックの設定

ヘルス設定には、登録されているインスタンスのヘルス状態を判断するために、ロードバランサーが使用する情報が含まれています。次の表は、ヘルスチェック設定のフィールドの説明を示しています。

フィールド	説明
プロトコル	インスタンスと接続するために使用するプロトコル。  有効な値: TCP、HTTP、HTTPS、および SSL  コンソールのデフォルト: HTTP  CLI/API のデフォルト: TCP
ポート	インスタンスに接続するために使用するポート (protocol:port ペアなど)。ロードバランサーが、設定された応答タイムアウト期間内に指定されたポートでインスタンスに接続できない場合、インスタンスは異常と見なされます。

フィールド	説明
	<p>プロトコル: TCP、HTTP、HTTPS、および SSL</p> <p>ポート範囲: 1 ~ 65535</p> <p>コンソールのデフォルト: HTTP:80</p> <p>CLI/API のデフォルト: TCP:80</p>
パス	<p>HTTP または HTTPS リクエストの送信先。</p> <p>HTTP または HTTPS GET リクエストがポートとパス上のインスタンスに発行されます。ロードバランサーが応答タイムアウト時間内に "200 OK" 以外の応答を受信した場合、インスタンスは異常と見なされます。応答に本文が含まれている場合、アプリケーションは Content-Length ヘッダーを 0 以上の値に設定するか、値を "chunked" に設定した Transfer-Encoding を指定する必要があります。</p> <p>デフォルト: /index.html</p>
応答タイムアウト	<p>ヘルスチェックからの応答を受け取るまで待つ時間 (秒単位)。</p> <p>有効な値: 2 ~ 60</p> <p>デフォルト: 5</p>
HealthCheck 間隔	<p>個々のインスタンスのヘルスチェックの間隔 (秒単位)。</p> <p>有効な値: 5 ~ 300</p> <p>デフォルト: 30</p>

フィールド	説明
非正常のしきい値	<p>EC2 インスタンスで異常が発生していることを宣言する前に連続して失敗したヘルスチェックの数。</p> <p>有効な値: 2 ~ 10</p> <p>デフォルト: 2</p>
正常のしきい値	<p>EC2 インスタンスが正常であることを宣言する前に連続して成功したヘルスチェックの数。</p> <p>有効な値: 2 ~ 10</p> <p>デフォルト: 10</p>

ロードバランサーは、指定されたポート、プロトコル、およびパスを使用して、Interval 秒ごとに、登録された各インスタンスにヘルスチェックリクエストを送信します。各ヘルスチェックリクエストは独立しており、間隔全体で存続します。インスタンスが応答するまでにかかる時間は、次のヘルスチェックまでの間隔に影響を与えません。ヘルスチェックが UnhealthyThresholdCount 連続失敗数のしきい値を超えると、ロードバランサーはインスタンスをサービス停止中の状態にします。ヘルスチェックが HealthyThresholdCount 連続成功数のしきい値を超えると、ロードバランサーはインスタンスを実行中の状態に戻します。

インスタンスがヘルスチェックの間隔内に 200 応答コードを返せば、HTTP/HTTPS ヘルスチェックは正常です。TCP 接続が成功すれば、TCP ヘルスチェックは正常です。SSL ハンドシェイクが成功すれば、SSL ヘルスチェックは正常です。

## ヘルスチェックの設定の更新

ロードバランサーのヘルスチェックの設定はいつでも更新できます。

コンソールを使用してロードバランサーのヘルスチェックの設定を更新するには

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. 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. [ヘルスチェック] タブで、[編集] を選択します。
5. [ヘルスチェックの編集の設定] ページの [ヘルス チェック] で、必要に応じて設定を更新します。
6. 選択した内容でよければ、[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ヘルスチェック設定を更新するには AWS CLI

次の [configure-health-check](#) コマンドを使用します。

```
aws elb configure-health-check --load-balancer-name my-load-balancer --health-check Target=HTTP:80/path,Interval=30,UnhealthyThreshold=2,HealthyThreshold=2,Timeout=3
```

## インスタンスのヘルスの確認

登録済みインスタンス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インスタンス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 [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. ロードバランサーの名前を選択して、その詳細ページを開きます。
4. [詳細] セクションの [ステータス] は、サービス中のインスタンスの数を示します。
5. [ターゲットインスタンス] タブの [ターゲットインスタンス] テーブル内の [ヘルスステータス] 列は、登録されている各インスタンスの特定のステータスを示します。

を使用してインスタンスのヘルスステータスを確認するには AWS CLI

次の [describe-instance-health](#) コマンドを使用します。

```
aws elb describe-instance-health --load-balancer-name my-load-balancer
```

## ヘルスチェックのトラブルシューティング

登録済みインスタンスが、いくつかの理由からロードバランサーのヘルスチェックに失敗する場合があります。ヘルスチェックの失敗理由として最も多いのは、EC2 インスタンスがロードバランサーへの接続を閉じている場合や、EC2 インスタンスからの応答がタイムアウトになった場合です。考

えられる原因、および失敗したヘルスチェックの問題を解決するための手順については、「[Classic Load Balancer のトラブルシューティング: ヘルスチェック](#)」を参照してください。

## Classic Load Balancer のインスタンスのセキュリティグループ

セキュリティグループは、1 つ以上のインスタンス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。EC2 インスタンスを起動するとき、1 つ以上のセキュリティグループをインスタンスに関連付けることができます。セキュリティグループごとに、トラフィックを許可する 1 つ以上のルールを追加します。セキュリティグループのルールは、いつでも変更できます。新しいルールは、セキュリティグループに関連付けられているすべてのインスタンスに自動的に適用されます。詳細については、「Amazon EC2 ユーザーガイド」の「[Amazon EC2 セキュリティグループ](#)」を参照してください。

インスタンスのセキュリティグループでは、ロードバランサーとの通信が許可される必要があります。次の表に、推奨されるインバウンドルールを示します。

ソース	プロトコル	ポート範囲	コメント
#####	TCP	##### ##	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
#####	TCP	#####	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。

## Classic Load Balancer のインスタンスのネットワーク ACL

ネットワークアクセスコントロールリスト (ACL) は、サブネットレベルで特定のインバウンドまたはアウトバウンドのトラフィックを許可または拒否します。VPC のデフォルトのネットワーク ACL を使用するか、セキュリティグループと同様のルールを使用して VPC のカスタムネットワーク ACL を作成し、セキュリティの追加レイヤーを VPC に追加できます。

VPC のデフォルトネットワークアクセスコントロールリスト (ACL) では、すべての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が許可されます。カスタムネットワーク ACL を作成する場合、ロードバランサーとインスタンスの通信を許可するルールを追加します。

インスタンスのサブネットの推奨ルールは、サブネットがプライベートとパブリックのどちらであるかによって異なります。次のルールは、プライベートサブネット用です。インスタンスがパブリックサブネット内にある場合、送信元と宛先を VPC の CIDR から 0.0.0.0/0 に変更します。

推奨されるインバウンドルールを次に示します。

ソース	プロトコル	ポート範囲	コメント
<i>VPC CIDR</i>	TCP	#####	インスタンスリスナーポートで VPC CIDR からのインバウンドトラフィックを許可する
<i>VPC CIDR</i>	TCP	#####	ヘルスチェックポートで VPC CIDR からのインバウンドトラフィックを許可する

推奨されるアウトバウンドルールを次に示します。

目的地	プロトコル	ポート範囲	コメント
<i>VPC CIDR</i>	TCP	1024-65535	一時ポートで VPC CIDR へのアウトバウンドトラフィックを許可する

# Classic Load Balancer のモニタリング

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バックエンドインスタンスに関する問題の解決を実行できます。

## CloudWatch メトリクス

Elastic Load Balancing は、ロードバランサーとバックエンドインスタンスに関するデータポイントを Amazon CloudWatch に発行します。CloudWatch では、それらのデータポイントについての統計を、(メトリクスと呼ばれる) 順序付けられた時系列データの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、「[Classic Load Balancer の CloudWatch メトリクス](#)」を参照してください。

## Elastic Load Balancing のアクセスログ

Elastic Load Balancing のアクセスログは、ロードバランサーに対し作成されたリクエストの詳細情報をキャプチャし、ログファイルの形で指定した Amazon S3 バケットに保存します。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詳細が含まれます。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の分析や、バックエンドアプリケーション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、「[Classic Load Balancer のアクセスログ](#)」を参照してください。

## CloudTrail ログ

AWS CloudTrail では、アカウントによって、または AWS アカウントに代わって Elastic Load Balancing API に対して行われた呼び出しを追跡できます。CloudTrail は、その情報をログファイルの形で指定した Amazon S3 バケットに格納します。これらのログファイルを使用して、作成されたリクエスト、リクエストの送信元のソース IP アドレス、リクエストの作成者、リクエストが作成された日時などを判別することにより、ロードバランサーのアクティビティを監視することができます。詳細については、「[CloudTrail を使用した Elastic Load Balancing の API コールのログ記録](#)」を参照してください。

# Classic Load Balancer の CloudWatch メトリクス

Elastic Load Balancing は、ロードバランサーとバックエンドインスタンス用として、データポイントを Amazon CloudWatch に発行します。CloudWatch では、それらのデータポイントについての統計を、(メトリクスと呼ばれる) 順序付けられた時系列データの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。例えば、指定した期間中のロードバランサーの正常な EC2 インスタンス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、メトリクスが許容範囲外になる場合、CloudWatch アラームを作成して、指定されたメトリクスを監視し、アクション (E メールアドレスに通知を送信するなど) を開始することができます。

Elastic Load Balancing は、ロードバランサー経由でリクエストが伝達される場合にのみ、メトリクスを CloudWatch 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。

Amazon CloudWatch の詳細については、「[Amazon CloudWatch ユーザーガイド](#)」を参照してください。

## 目次

- [Classic Load Balancer のメトリクス](#)
- [Classic Load Balancer のメトリクスディメンション](#)
- [Classic Load Balancer メトリクスの統計](#)
- [ロードバランサーの CloudWatch メトリクスの表示](#)

## Classic Load Balancer のメトリクス

AWS/ELB 名前空間には、次のメトリクスが含まれます。

メトリクス	説明
BackendConnectionErrors	<p>ロードバランサーと登録されたインスタンス間で正常に確立されなかった接続数。エラーが発生すると、ロードバランサーは接続を再試行するため、このカウントはリクエストレートを上回ります。この数には、ヘルスチェックに関連する接続エラーも含まれます。</p> <p>レポート条件: ゼロ以外の値がある</p>

メトリクス	説明
	<p>統計: 最も有用な統計は Sum です。Average、Minimum、Maximum はロードバランサーノードごとに報告されるため、通常有益ではありません。ただし、最大と最小の差 (ピーク値対平均、または平均対底値) は、1 つのロードバランサーノードが異常値かどうかを判断する上で有益な場合があります。</p> <p>例: ロードバランサーにインスタンスがあり、そのうちの 2 つのインスタンスは us-west-2a に、もう 2 つのインスタンスは us-west-2b にあるとします。また、us-west-2a の 1 つのインスタンスへの接続を試みると、バックエンド接続エラーになるとします。us-west-2a の合計にはこれらの接続エラーが含まれますが、us-west-2b の合計には含まれません。このため、ロードバランサーの合計は us-west-2a の合計と同じになります。</p>
DesyncMitigationMode_NonCompliant_Request_Count	<p>[HTTP リスナー] RFC 7230 に準拠していないリクエストの数。</p> <p>レポート条件: ゼロ以外の値がある</p> <p>統計: 最も有用な統計は Sum です。</p>

メトリクス	説明
HealthyHostCount	<p>ロードバランサーに登録された、正常なインスタンスの数。新しく登録されたインスタンスは、最初のヘルスチェックに合格すると、正常な状態と見なされます。クロスゾーンの負荷分散が有効な場合、すべてのアベイラビリティゾーンにわたって LoadBalancerName デイメンションの正常なインスタンスの数が算出されます。それ以外の場合、アベイラビリティゾーンごとに計算されます。</p> <p>レポート要件: 登録されたインスタンスがある</p> <p>Statistics: 最も有用な統計は Average および Maximum です。これらの統計はロードバランサーノードによって決まります。短い間、インスタンスを異常と判断するロードバランサーノードがあったり、インスタンスを正常と判断するノードがあったりします。</p> <p>例: ロードバランサーにインスタンスがあり、そのうちの2つのインスタンスは us-west-2a に、もう2つのインスタンスは us-west-2b にあるとします。また、us-west-2a には1つの異常なインスタンスがあり、us-west-2b には異常なインスタンスはないとします。AvailabilityZone デイメンションでは、us-west-2a の正常なインスタンスの平均数は1、異常なインスタンスの平均数は1、us-west-2b の正常なインスタンスの平均数は2、異常なインスタンスの平均数は0です。</p>

メトリクス	説明
HTTPCode_Backend_2XX , HTTPCode_Backend_3XX , HTTPCode_Backend_4XX , HTTPCode_Backend_5XX	<p>[HTTP リスナー] 登録されたインスタンスによって生成された HTTP 応答コードの数。この数には、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。</p> <p>レポート条件: ゼロ以外の値がある</p> <p>統計: 最も有用な統計は Sum です。Minimum、Maximum、および Average はすべて 1 です。</p> <p>例: ロードバランサーにインスタンスがあり、そのうちの 2 つのインスタンスは us-west-2a に、もう 2 つのインスタンスは us-west-2b にあるとします。また、us-west-2a の 1 つのインスタンスに送信されたリクエストは HTTP 500 応答となるとします。us-west-2a の合計にはこれらのエラーレスポンスが含まれますが、us-west-2b の合計には含まれません。このため、ロードバランサーの合計は us-west-2a の合計と同じになります。</p>
HTTPCode_ELB_4XX	<p>[HTTP リスナー] ロードバランサーで生成される HTTP 4XX クライアントのエラーコードの数。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。</p> <p>レポート条件: ゼロ以外の値がある</p> <p>統計: 最も有用な統計は Sum です。Minimum、Maximum、および Average はすべて 1 です。</p> <p>例: ロードバランサーで us-west-2a および us-west-2b が有効になっていて、クライアントリクエストに正しい形式でないリクエスト URL が含まれているとします。その結果、すべてのアベイラビリティゾーンでのクライアントエラーが増加する可能性があります。ロードバランサーの合計はアベイラビリティゾーンの値の合計です。</p>

メトリクス	説明
HTTPCode_ELB_5XX	<p>[HTTP リスナー] ロードバランサーで生成される HTTP 5XX サーバーのエラーコードの数。この数には、登録されたインスタンスによって生成される応答コードは含まれません。ロードバランサーに登録されている正常なインスタスがない場合、またはリクエストレートがインスタンスまたはロードバランサーの容量を超える場合 (スピルオーバー)、このメトリクスが報告されます。</p> <p>レポート条件: ゼロ以外の値がある</p> <p>統計: 最も有用な統計は Sum です。Minimum、Maximum、および Average はすべて 1 です。</p> <p>例: ロードバランサーで us-west-2a および us-west-2b が有効になっていて、us-west-2a のインスタンスで高いレイテンシーが発生し、リクエストに回答するまでに時間がかかっているとします。その結果、us-west-2a のロードバランサーノードのサージキューはいっぱいになり、クライアントは 503 エラーを受け取ります。us-west-2b が正常に回答を継続する場合、ロードバランサーの合計は us-west-2a の場合の合計と同じになります。</p>

メトリクス	説明
Latency	<p>(HTTP リスナーの場合) ロードバランサーが登録済みインスタンスにリクエストを送信した時点から、そのインスタンス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 (秒単位)。</p> <p>[TCP リスナー] ロードバランサーが、登録済みインスタンスへの接続を正常に確立するまでの合計経過時間 (秒)。</p> <p>レポート条件: ゼロ以外の値がある</p> <p>統計: 最も有用な統計は Average です。Maximum を使用して、一部のリクエストに平均を大幅に上回る時間がかかっているかどうかを判断します。通常、Minimum は有用ではありません。</p> <p>例: ロードバランサーにインスタンスがあり、そのうちの 2 つのインスタンスは us-west-2a に、もう 2 つのインスタンスは us-west-2b にあるとします。また、us-west-2a の 1 つのインスタンスに送信されたリクエストのレイテンシーがより高いとします。us-west-2a の平均値は、us-west-2b の平均よりも高いとします。</p>

メトリクス	説明
RequestCount	<p>指定された間隔 (1 分または 5 分) の間に完了したリクエストの数、または接続の数。</p> <p>[HTTP リスナー] 登録されたインスタンスからの HTTP エラーレスポンスを含めて、受け取ったリクエストとルーティングされたリクエストの数。</p> <p>[TCPリスナー] 登録されたインスタンスに対して行われた接続の数。</p> <p>レポート条件: ゼロ以外の値がある</p> <p>統計: 最も有用な統計は Sum です。Minimum、Maximum、および Average はすべて 1 を返すことに注意してください。</p> <p>例: ロードバランサーにインスタンスがあり、そのうちの 2 つのインスタンスは us-west-2a に、もう 2 つのインスタンスは us-west-2b にあるとします。また、100 件のリクエストがロードバランサーに送信されるとします。60 件のリクエストが us-west-2a に送信され、各インスタンスがそれぞれ 30 件のリクエストを受信します。40 件のリクエストが us-west-2b に送信され、各インスタンスがそれぞれ 20 件のリクエストを受信します。AvailabilityZone デイメンションでは、us-west-2a の合計リクエスト数は 60 件、us-west-2b の合計リクエスト数は 40 件です。LoadBalancerName デイメンションでは、合計 100 件のリクエストがあります。</p>

メトリクス	説明
SpilloverCount	<p>サージキューがいっぱいなため、拒否されたリクエストの総数。</p> <p>[HTTP リスナー] ロードバランサーは、HTTP 503 エラーコードを返します。</p> <p>[TCP リスナー] ロードバランサーは接続を終了します。</p> <p>レポート条件: ゼロ以外の値がある</p> <p>統計: 最も有用な統計は Sum です。Average、Minimum、Maximum はロードバランサーノードごとに報告されるため、通常有益ではありません。</p> <p>例: ロードバランサーで us-west-2a および us-west-2b が有効になっていて、us-west-2a のインスタンスで高いレイテンシーが発生し、リクエストに回答するまでに時間がかかっているとします。その結果、us-west-2a のロードバランサーノードのサージキューがいっぱいになり、スピルオーバーが発生します。us-west-2b が正常に回答を継続する場合、ロードバランサーの合計は us-west-2a の場合の合計と同じになります。</p>

メトリクス	説明
SurgeQueueLength	<p>正常なインスタンスへのルーティングを保留中のリクエスト (HTTP リスナー) または接続 (TCP リスナー) の合計数。キューの最大サイズは 1,024 です。追加のリクエストまたは接続は、キューがいっぱいになると拒否されます。詳細については、「SpilloverCount」を参照してください。</p> <p>レポート条件: ゼロ以外の値がある。</p> <p>統計: Maximum はキューに送信されたリクエストのピークを表すため、最も有用な統計です。Average 統計は Minimum および Maximum と組み合わせて、キューに送信されたリクエストの範囲を確認する場合にも有用です。Sum は有用ではありません。</p> <p>例: ロードバランサーで us-west-2a および us-west-2b が有効になっていて、us-west-2a のインスタンスで高いレイテンシーが発生し、リクエストに回答するまでに時間がかかっているとします。その結果、us-west-2a のロードバランサーノードのサージキューがいっぱいになり、クライアントに回答時間の増加が発生している可能性があります。これが継続すると、ロードバランサーにスピルオーバーが発生する可能性が高くなります (SpilloverCount メトリクスを参照してください)。us-west-2b が正常に回答を継続する場合、ロードバランサーの max は us-west-2a の場合の max と同じになります。</p>

メトリクス	説明
UnHealthyHostCount	<p>ロードバランサーに登録された、異常なインスタンスの数。インスタンスは、ヘルスチェックに対して構成された異常なしきい値を超えると、異常な状態と見なされます。異常なインスタンスは、ヘルスチェックに設定されている状態のしきい値を満たせば再び正常な状態と見なされます。</p> <p>レポート要件: 登録されたインスタンスがある</p> <p>Statistics: 最も有用な統計は Average および Minimum です。これらの統計はロードバランサーノードによって決まります。短い間、インスタンスを異常と判断するロードバランサーノードがあったり、インスタンスを正常と判断するノードがあったりします。</p> <p>例: HealthyHostCount を参照してください。</p>

次のメトリクスを使用すると、Classic Load Balancer を Application Load Balancer に移行する場合のコストを見積もることができます。これらのメトリクスは、単なる情報提供を意図しており、CloudWatch アラームでの使用を目的にしていません。Classic Load Balancer に複数のリスナーがある場合、これらのメトリクスはすべてのリスナーを集計したものになります。

この見積りは、1つのロードバランサーで1つのデフォルトルールと1つの証明書(2K サイズ)を使用した場合に基づいています。サイズが4K以上の証明書を使用する場合にコストを見積もる方法としては、移行ツールを使用しながら Classic Load Balancer に基づいた Application Load Balancer を作成し、その Application Load Balancer で ConsumedLCUs メトリクスをモニタリングすることをお勧めします。詳細については、Elastic Load Balancing ユーザーガイドの [Classic Load Balancer の移行](#) を参照してください。

メトリクス	説明
EstimatedALBActiveConnectionCount	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 接続の予測数。

メトリクス	説明
EstimatedALBConsumedLCUs	Application Load Balancer で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) の予測数。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、 <a href="#">Elastic Load Balancing の料金表</a> を参照してください。
EstimatedALBNewConnectionCount	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、新たに確立される TCP 接続の予測数。
EstimatedProcessedBytes	Application Load Balancer で処理されるバイトの予測数。

## Classic Load Balancer のメトリクスディメンション

Classic 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	指定されたアベイラビリティゾーンでメトリクスデータをフィルタリングします。
LoadBalancerName	指定された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。

## Classic Load Balancer メトリクスの統計

CloudWatch では、Elastic Load Balancing で発行されたメトリクスのデータポイントに基づいた統計が提供されます。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/値のペアです。例えば、特定のアベイラビリティ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 統計と Maximum 統計には、個々のロードバランサーノードによって報告される最小値と最大値が反映されます。例えば、ロードバランサーノードが 2 つあるとします。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。

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、RequestCount、HTTPCode\_ELB\_XXX、HTTPCode\_Backend\_XXX、BackendConnectionErrors、などの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 番目のパーセンタイルを Auto Scaling のトリガーあるいは CloudWatch アラームとして使用すると、2 ミリ秒以上の処理時間がかかるリクエスト数が 1% を超えないようターゲットを設定できます。

## ロードバランサーの CloudWatch メトリクスの表示

Amazon EC2 コンソールを使用して、ロードバランサーに関する 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。

別の方法としては、ロードバランサーのメトリクスの表示に、CloudWatch コンソールを使用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 コンソールの <https://console.aws.amazon.com/ec2/> を開いてください。

2. ナビゲーションペインの [ロードバランシング] で [ロードバランサー] を選択します。
3. 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. [モニタリング] タブを選択します。
5. 1つのメトリクスを拡大して表示するには、グラフ上にカーソルを移動させ、Maximize アイコンを選択します。以下のメトリクスが利用可能です。
  - 正常なホスト — HealthyHostCount
  - 非正常なホスト — UnHealthyHostCount
  - 平均レイテンシー — Latency
  - リクエスト — RequestCount
  - バックエンド接続エラー — BackendConnectionErrors
  - キュー長の急増 — SurgeQueueLength
  - 過剰数 — SpilloverCount
  - HTTP 2XXs — HTTPCode\_Backend\_2XX
  - HTTP 3XXs — HTTPCode\_Backend\_3XX
  - HTTP 4XXs — HTTPCode\_Backend\_4XX
  - HTTP 5XXs — HTTPCode\_Backend\_5XX
  - ELB HTTP 4XXs — HTTPCode\_ELB\_4XX
  - ELB HTTP 5XXs — HTTPCode\_ELB\_5XX
  - 推定処理バイト数 — EstimatedProcessedBytes
  - ALB の推定消費 LCU 数 — EstimatedALBConsumedLCUs
  - ALB の推定アクティブ接続数 — EstimatedALBActiveConnectionCount
  - ALB の推定新規接続数 — EstimatedALBNewConnectionCount

CloudWatch コンソールを使用してメトリクスを表示するには

1. CloudWatch コンソール (<https://console.aws.amazon.com/cloudwatch/>) を開きます。
2. ナビゲーションペインで [Metrics (メトリクス)] を選択してください。
3. [ELB] 名前空間を選択します。
4. 次のいずれかを行ってください。
  - メトリクスディメンションを選択して、ロードバランサーごと、アベイラビリティーゾーンごと、あるいはすべてのロードバランサーのメトリクスを表示できます。

- 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。
- 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。

## Classic 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関する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の分析や、問題のトラブルシューティングを行うことができます。

アクセスログの作成は、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の作成を有効にすると、Elastic Load Balancing はログをキャプチャし、そのログを指定した Amazon S3 バケット内に保存します。アクセスログの作成はいつでも無効にできます。

各アクセスログファイルは、S3 バケットに保存される前に、SSE-S3 を使用して自動的に暗号化され、アクセス時に復号化されます。お客様によるアクションは必要はありません。暗号化と復号は透過的に実行されます。各ログファイルは一意のキーで暗号化されます。この一意のキー自体が、定期的に更新される KMS キーで更新されます。詳細については、「[Amazon S3 ユーザーガイド](#)」の「[Amazon S3 が管理する暗号化キーによるサーバー側の暗号化 \(SSE-S3\) を使用したデータの保護](#)」を参照してください。

アクセスログに対する追加料金はありません。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](#)
- [アクセスログのエントリ](#)
- [アクセスログの処理](#)

- [Classic Load Balancer のアクセスログの有効化](#)
- [Classic Load Balancer のアクセスログの無効化](#)

## アクセスログファイル

Elastic Load Balancing は、指定した間隔で各ロードバランサーノードにログファイルを発行します。ロードバランサーのアクセスログを有効にするときに、5 分または 60 分の発行間隔を指定できます。デフォルトでは、Elastic Load Balancing は 60 分間隔でログを発行します。間隔を 5 分に設定すると、ログは、1:05、1:10、1:15、のように発行されます。ログの配信開始は、間隔が 5 分に設定されている場合は最大 5 分の遅延があり、間隔が 60 分に設定されている場合は最大 15 分の遅延があります。発行間隔はいつでも変更できます。

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、サイトのトラフィックが大量である場合、ロードバランサーノードが複数ある場合、およびログ発行間隔が短い場合に発生します。

アクセスログのファイル名には次の形式を使用します。

```
amzn-s3-demo-loadbalancer-logs[/logging-prefix]/AWSLogs/aws-account-id/  
elasticloadbalancing/region/yyyy/mm/dd/aws-account-id_elasticloadbalancing_region_load-  
balancer-name_end-time_ip-address_random-string.log
```

amzn-s3-demo-loadbalancer-logs

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-name

ロードバランサーの名前。

## end-time

ログ作成の間隔が終了した日時。例えば、発行間隔が 5 分の場合、終了時刻が 20140215T2340Z であれば 23:35 から 23:40 の間に作成されたリクエストに関するエントリが含まれています。

## 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。

「my-app」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://amzn-s3-demo-loadbalancer-logs/my-app/AWSLogs/123456789012/elasticloadbalancing/us-west-2/2018/02/15/123456789012_elasticloadbalancing_us-west-2_my-loadbalancer_20180215T2340Z_172.160.001.192_20sg8hgm.log
```

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://amzn-s3-demo-loadbalancer-logs/AWSLogs/123456789012/elasticloadbalancing/us-west-2/2018/02/15/123456789012_elasticloadbalancing_us-west-2_my-loadbalancer_20180215T2340Z_172.160.001.192_20sg8hgm.log
```

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 S3 ユーザーガイド」の [「オブジェクトのライフサイクル管理」](#) を参照してください。

## アクセスログのエントリ

Elastic Load Balancing は、バックエンドのインスタンス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。例えば、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、応答できる正常に動作しているインスタンスがない場合も、そのリクエストは記録されます。

### 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。

### 構文

各ログエントリには、ロードバランサーに対する 1 件のリクエストの詳細が含まれます。ログエントリのすべてのフィールドはスペースで区切られています。ログファイルの各エントリは次の形式になります。

```
timestamp elb client:port backend:port request_processing_time backend_processing_time
response_processing_time elb_status_code backend_status_code received_bytes sent_bytes
"request" "user_agent" ssl_cipher ssl_protocol
```

次の表は、アクセスログのエントリのフィールドを示しています。

フィールド	説明
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)。
elb	ロードバランサーの名前
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。
backend:port	このリクエストを処理した登録済みインスタンスの IP アドレスとポート。

フィールド	説明
	<p>ロードバランサーが登録されたインスタンスにリクエストを送信できない場合、または応答が送信される前にインスタンスが接続を閉じた場合、この値は - に設定されます。</p> <p>登録済みインスタンス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 - に設定される場合があります。</p>
request_processing_time	<p>[HTTP リスナー] ロードバランサーがリクエストを受け取った時点から登録済みインスタンスに送信するまでの合計経過時間 (秒単位)。</p> <p>[TCPリスナー] ロードバランサーがクライアントから TCP/SSL 接続を受け入れた時点から、ロードバランサーがデータの最初のバイトを登録されたインスタンスに送信するまでの合計経過時間 (秒単位)。</p> <p>ロードバランサーがリクエストを登録されたインスタンス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、登録済みインスタンス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。さらに、TCP リスナーの場合、クライアントがロードバランサーと接続を確立するが、データを送信しない場合にもこの状況が発生する可能性があります。</p> <p>登録済みインスタンス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。</p>

フィールド	説明
backend_processing_time	<p>(HTTP リスナーの場合) ロードバランサーが登録済みインスタンスにリクエストを送信した時点から、そのインスタンス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 (秒単位)。</p> <p>[TCP リスナー] ロードバランサーが、登録済みインスタンスへの接続を正常に確立するまでの合計経過時間 (秒)。</p> <p>ロードバランサーがリクエストを登録されたインスタンス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、登録済みインスタンス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。</p> <p>登録済みインスタンス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。</p>
response_processing_time	<p>(HTTP リスナーの場合) ロードバランサーが登録済みインスタンス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 (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。</p> <p>(TCP リスナーの場合) ロードバランサーが登録済みインスタンスから最初のバイト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 (秒単位)。</p> <p>ロードバランサーがリクエストを登録されたインスタンス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、登録済みインスタンス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。</p> <p>登録済みインスタンス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。</p>
elb_status_code	<p>(HTTP リスナーの場合) ロードバランサーからの応答のステータスコード。</p>

フィールド	説明
backend_status_code	(HTTP リスナーの場合) 登録済みインスタンスからの応答のステータスコード。
received_bytes	<p>クライアント (リクエスト) から受け取ったリクエストのサイズ (バイト単位)。</p> <p>(HTTP リスナーの場合) この値にはリクエストの本文が含まれますがヘッダーは含まれません。</p> <p>(TCP リスナーの場合) この値にはリクエストの本文とヘッダーが含まれます。</p>
sent_bytes	<p>クライアント (リクエスト) に返される応答のサイズ (バイト単位)。</p> <p>(HTTP リスナーの場合) この値には応答の本文が含まれますがヘッダーは含まれません。</p> <p>(TCP リスナーの場合) この値にはリクエストの本文とヘッダーが含まれます。</p>
リクエスト	<p>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、次の形式でログに記録されます。HTTP メソッド + プロトコル://ホストヘッダー:ポート + パス + 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。</p> <p>(TCP リスナーの場合) URL は、3 個のダッシュをそれぞれスペース 1 個で区切り、末尾がスペースになります ("---")。</p>
user_agent	[HTTP/HTTPS リスナー] リクエスト元のクライアントを特定する User-Agent 文字列。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。

フィールド	説明
ssl_cipher	[HTTPS/SSL リスナー] SSL 暗号。正常なネゴシエーションの後に受信 SSL/TLS 接続が確立した場合にのみ、この値が記録されます。それ以外の場合、値は - に設定されます。
ssl_protocol	[HTTPS/SSL リスナー] SSL プロトコル。正常なネゴシエーションの後に受信 SSL/TLS 接続が確立した場合にのみ、この値が記録されます。それ以外の場合、値は - に設定されます。

## 例

### 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)。

```
2015-05-13T23:39:43.945958Z my-loadbalancer 192.168.131.39:2817 10.0.0.1:80 0.000073
0.001048 0.000057 200 200 0 29 "GET http://www.example.com:80/ HTTP/1.1" "curl/7.38.0"
- -
```

### 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)。

```
2015-05-13T23:39:43.945958Z my-loadbalancer 192.168.131.39:2817 10.0.0.1:80
0.000086 0.001048 0.001337 200 200 0 57 "GET https://www.example.com:443/ HTTP/1.1"
"curl/7.38.0" DHE-RSA-AES128-SHA TLSv1.2
```

### TCP エントリ例

次の例は、TCP リスナーのログエントリです (ポート 8080 からポート 80)。

```
2015-05-13T23:39:43.945958Z my-loadbalancer 192.168.131.39:2817 10.0.0.1:80 0.001069
0.000028 0.000041 - - 82 305 "- - -" "- - -"
```

### SSL エントリ例

次の例は、SSL リスナーのログエントリです (ポート 8443 からポート 80)。

```
2015-05-13T23:39:43.945958Z my-loadbalancer 192.168.131.39:2817 10.0.0.1:80 0.001065
0.000015 0.000023 - - 57 502 "- - - " "- " ECDHE-ECDSA-AES128-GCM-SHA256 TLSv1.2
```

## アクセスログの処理

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。このような大容量のデータは、行単位で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「[Classic Load Balancer ログのクエリ](#)」を参照してください。
- [Loggly](#)
- [Splunk](#)
- [Sumo Logic](#)

## Classic Load Balancer のアクセスログの有効化

ロードバランサーのアクセスログを有効にするには、ロードバランサーがログの格納先として使用する、Amazon S3 バケットの名前を指定する必要があります。また、バケットに書き込むためのアクセス権限を Elastic Load Balancing に付与するバケットポリシーを、このバケットにアタッチする必要があります。

### 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

アクセスログの作成を有効にするときは、アクセスログファイルの S3 バケットを指定する必要があります。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 S3 ユーザーガイド」の「[バケットのクォータ、制約、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。

### バケットポリシー

このポリシーは、ログ配信サービスにアクセス許可を付与します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "logdelivery.elasticloadbalancing.amazonaws.com"
      },
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/prefix/AWSLogs/123456789012/*"
    }
  ]
}
```

Resource に、ポリシー例に示す形式を使用して、アクセスログの場所の ARN を入力します。S3 バケット ARN のリソースパスには、ロードバランサーを持つアカウントのアカウント ID を必ず含めてください。これにより、指定されたアカウントのロードバランサーのみが S3 バケットにアクセスログを書き込むことができます。

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含めるかどうかによって変わってきます。

プレフィックスを含む S3 バケット ARN の例

S3 バケット名は amzn-s3-demo-logging-bucket で、プレフィックスは logging-prefix です。

```
arn:aws:s3:::amzn-s3-demo-logging-bucket/logging-prefix/AWSLogs/123456789012/*
```

[AWS GovCloud (US)] 次の例では、AWS GovCloud (US) Regions の ARN 構文を使用します。

```
arn:aws-us-gov:s3:::amzn-s3-demo-logging-bucket/logging-prefix/AWSLogs/123456789012/*
```

プレフィックスを持たない S3 バケット ARN の例

S3 バケット名は amzn-s3-demo-logging-bucket です。S3 バケット ARN にプレフィックス部分はありません。

```
arn:aws:s3:::amzn-s3-demo-logging-bucket/AWSLogs/123456789012/*
```

[AWS GovCloud (US)] 次の例では、AWS GovCloud (US) Regionsの ARN 構文を使用します。

```
arn:aws-us-gov:s3:::amzn-s3-demo-logging-bucket/AWSLogs/123456789012/*
```

## レガシーバケットポリシー

以前は、2022年8月より前に利用可能であったリージョンでは、リージョンに固有の Elastic Load Balancing アカウントにアクセス許可を付与するポリシーが必要でした。このレガシーポリシーは引き続きサポートされていますが、上記の新しいポリシーに置き換えることをお勧めします。レガシーバケットポリシー (ここには示されていません) を引き続き使用してもかまいません。

参考までに、Principal で指定する Elastic Load Balancing アカウントの ID を次に示します。このリストに含まれていないリージョンは、これまで一度もレガシーバケットポリシーをサポートしていないことに注意してください。

- 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 (米国東部) – 190560391635
- AWS GovCloud (米国西部) – 048591011584

## セキュリティのベストプラクティス

セキュリティを強化するには、正確な S3 バケット ARN を使用します。

- S3 バケット ARN だけでなく、完全なリソースパスを使用します。
- S3 バケット ARN のアカウント ID 部分を含めます。
- S3 バケット ARN のアカウント ID 部分にワイルドカード (\*) を使用しないでください。

バケットポリシーを作成したら、Amazon S3 コンソールや AWS CLI コマンドなどの Amazon S3 インターフェイスを使用して、バケットポリシーを S3 バケットにアタッチします。

コンソールを使用してバケットポリシーをバケットにアタッチするには

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. [Save changes] (変更の保存) をクリックします。

を使用して S3 バケットにバケットポリシーをアタッチするには AWS CLI

[put-bucket-policy](#) コマンドを使用します。この例では、バケットポリシーは指定された .json ファイルに保存されます。

```
aws s3api put-bucket-policy \  
  --bucket amzn-s3-demo-bucket \  
  --policy file://access-log-policy.json
```

## ステップ 3: アクセスログを設定する

以下の手順を使って、リクエスト情報を取り込み S3 バケットにログファイルを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、「AWSLogs」を含めることはできません。

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を設定するには

1. Amazon EC2 コンソールの <https://console.aws.amazon.com/ec2/> を開いてください。
2. ナビゲーションペインの [ロードバランシング] で [ロードバランサー] を選択します。
3. 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. [ロードバランサー属性の編集] ページの [モニタリング] セクションで、次の操作を実行します。
  - a. [アクセスログ] を有効にします。
  - b. [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://amzn-s3-demo-logging-bucket/logging-prefix`
    - プレフィックスなしの URI: `s3://amzn-s3-demo-logging-bucket`
  - c. [ロギング間隔] は 60 minutes - default のままにしておきます。
  - d. [Save changes] (変更の保存) をクリックします。

を使用してロードバランサーのアクセスログを設定するには AWS CLI

まず、Elastic Load Balancing が 60 分ごとにログをキャプチャし、ログ用に作成した S3 バケットにそれを配信できるようにする `.json` ファイルを作成します。

```
{
  "AccessLog": {
    "Enabled": true,
    "S3BucketName": "amzn-s3-demo-logging-bucket",
```

```
"EmitInterval": 60,  
"S3BucketPrefix": "my-app"  
}  
}
```

次に、以下のように [modify-load-balancer-attributes](#) コマンドでこの .json ファイルを定義します。

```
aws elb modify-load-balancer-attributes --load-balancer-name my-loadbalancer --load-balancer-attributes file://my-json-file.json
```

以下に、応答の例を示します。

```
{  
  "LoadBalancerAttributes": {  
    "AccessLog": {  
      "Enabled": true,  
      "EmitInterval": 60,  
      "S3BucketName": "amzn-s3-demo-logging-bucket",  
      "S3BucketPrefix": "my-app"  
    }  
  },  
  "LoadBalancerName": "my-loadbalancer"  
}
```

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。そうしないと、同じ名前の新しいバケットと AWS アカウント、所有していないで作成された必要なバケットポリシーがある場合、Elastic Load Balancing はロードバランサーのアクセス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

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。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 S3 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。
  - プレフィックスがある場合の場所：`amzn-s3-demo-loadbalancer-logs/logging-prefix/AWSLogs/123456789012/ELBAccessLogTestFile`
  - プレフィックスのない場合の場所：`amzn-s3-demo-loadbalancer-logs/AWSLogs/123456789012/ELBAccessLogTestFile`

## トラブルシューティング

バケットへのアクセスが拒否されました: **bucket-name**。S3 バケットの許可を確認してください  
この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。

## Classic Load Balancer のアクセスログの無効化

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。アクセスログの作成を無効にした後でも、それを削除するまでアクセスログは Amazon S3 に保持されます。詳細については、「Amazon S3 ユーザーガイド」の「[S3 バケットの使用](#)」を参照してください。

コソールを使用してロードバランサーのアクセスログを無効にするには

1. Amazon EC2 コンソールの <https://console.aws.amazon.com/ec2/> を開いてください。
2. ナビゲーションペインの [ロードバランシング] で [ロードバランサー] を選択します。
3. 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. [ロードバランサー属性の編集] ページの [モニタリング] セクションで、[アクセス ログ] を無効にします。

を使用してアクセスログを無効にするには AWS CLI

次の [modify-load-balancer-attributes](#) コマンドを使用して、アクセスログを無効にします。

```
aws elb modify-load-balancer-attributes --load-balancer-name my-loadbalancer --load-balancer-attributes "{\"AccessLog\":{\"Enabled\":false}}"
```

以下に、応答の例を示します。

```
{
  "LoadBalancerName": "my-loadbalancer",
  "LoadBalancerAttributes": {
    "AccessLog": {
      "S3BucketName": "amzn-s3-demo-loadbalancer-logs",
      "EmitInterval": 60,
      "Enabled": false,
      "S3BucketPrefix": "my-app"
    }
  }
}
```

# Classic Load Balancer のトラブルシューティングを行う

次の表は、Classic Load Balancer を利用する際に役立つトラブルシューティングのリソースの一覧です。

## API エラー

### エラー

[CertificateNotFound: 未定義](#)

[OutOfService: 一時的なエラーの発生](#)

## HTTP エラー

### エラー

[HTTP 400: BAD\\_REQUEST](#)

[HTTP 405: METHOD\\_NOT\\_ALLOWED](#)

[HTTP 408: Request timeout](#)

[HTTP 502: Bad gateway](#)

[HTTP 503: Service Unavailable](#)

[HTTP 504: Gateway Timeout](#)

## レスポンスコードのメトリクス

### レスポンスコードのメトリクス

[HTTPCode\\_ELB\\_4XX](#)

[HTTPCode\\_ELB\\_5XX](#)

[HTTPCode\\_Backend\\_2XX](#)

## レスポンスコードのメトリクス

[HTTPCode\\_Backend\\_3XX](#)

[HTTPCode\\_Backend\\_4XX](#)

[HTTPCode\\_Backend\\_5XX](#)

## ヘルスチェックの問題

### 問題

[ヘルスチェックのターゲットページのエラー](#)

[インスタンスへの接続がタイムアウトした](#)

[パブリックキー認証が失敗する](#)

[インスタンスがロードバランサーからのトラフィックを受信しない](#)

[インスタンスのポートが開いていない](#)

[Auto Scaling グループのインスタンスが ELB ヘルスチェックに失敗する](#)

## 接続の問題

### 問題

[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](#)

[ロードバランサーに送信された HTTPS リクエストは「NET::ERR\\_CERT\\_COMMON\\_NAME\\_INVALID」を返します](#)

## インスタンス登録の問題

### 問題

[EC2 インスタンスの登録に時間がかかりすぎる](#)

[有料 AMI から起動したインスタンスを登録できない](#)

## Classic Load Balancer のトラブルシューティング: API エラー

次は、Elastic Load Balancing API によって返されるエラーメッセージ、考えられる原因、および問題の解決のために取るべき手順を示しています。

### エラーメッセージ

- [CertificateNotFound: 未定義](#)
- [OutOfService: 一時的なエラーの発生](#)

### CertificateNotFound: 未定義

原因 1: AWS マネジメントコンソールを使用して作成した証明書は、すべてのリージョンに伝達されるまでに遅延があります。この遅延が発生すると、ロードバランサーの作成プロセスの最終ステップでエラーメッセージが表示されます。

解決方法 1: 15 分ほど待つから、再度試してください。問題が解決しない場合は、[AWS サポートセンター](#)にアクセスしてサポートをお求めください。

原因 2: AWS CLI または API を直接使用している場合、存在しない証明書に Amazon リソースネーム (ARN) を指定すると、このエラーが表示されることがあります。

解決策 2: AWS Identity and Access Management (IAM) アクション [GetServerCertificate](#) を使用して証明書 ARN を取得し、ARN に正しい値を指定したことを確認します。

### OutOfService: 一時的なエラーの発生

原因: Elastic Load Balancing サービスまたはその基盤ネットワーク内に一時的な問題が発生しています。この一時的な問題は、ロードバランサーとその登録済みインスタンスの状態を Elastic Load Balancing がクエリした際にも、発生する場合があります。

解決方法: API 呼び出しを再試行してください。問題が解決しない場合は、[AWS サポートセンター](#)にアクセスしてサポートをお求めください。

## Classic Load Balancer のトラブルシューティング: HTTP エラー

HTTP メソッド (HTTP 動詞ともいう) は、HTTP リクエストを受信するリソースに対して実行するアクションを指定します。HTTP リクエストの標準メソッドは、RFC 2616 の「[Method Definitions](#)」で定義されています。標準メソッドには GET、POST、PUT、HEAD、および OPTIONS があります。ウェブアプリケーションによっては、HTTP/1.1 メソッドの拡張であるメソッドが必要とされます (導入されている場合もあります)。一般的な HTTP 拡張メソッドには PATCH、REPORT、MKCOL、PROPFIND、MOVE、LOCK などがあります。Elastic Load Balancing では、標準 HTTP メソッドも標準外の HTTP メソッドもすべて受け付けます。

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。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が RFC 2616 の「[Message Headers](#)」で定義されています。詳細については、「[HTTP ヘッダーと Classic Load Balancer](#)」を参照してください。

ロードバランサーは、HTTP リクエストを受信すると、誤った形式のリクエストがないかどうかをチェックすると共に、メソッドの長さをチェックします。ロードバランサーへの HTTP リクエスト内のメソッドの長さの合計は、127 文字以下にする必要があります。これら 2 つのチェックを渡す HTTP リクエストは、ロードバランサーにより EC2 インスタンスに送信されます。リクエストのメソッドフィールドの形式が正しくなければ、[HTTP 400: BAD\\_REQUEST](#) エラーが返されます。リクエストのメソッドフィールドの長さが 127 文字を超えていれば、[HTTP 405: METHOD\\_NOT\\_ALLOWED](#) エラーが返されます。

EC2 インスタンスは、リクエストに含まれるメソッドを実装し、応答をクライアントに返信することによって、有効なリクエストを処理します。サポートされているメソッドもサポートされていないメソッドも処理するように、インスタンスを設定しておく必要があります。

以下はロードバランサーによって返されるエラーメッセージ、考えられる原因、問題の解決のためのステップです。

### エラーメッセージ

- [HTTP 400: BAD\\_REQUEST](#)
- [HTTP 405: METHOD\\_NOT\\_ALLOWED](#)
- [HTTP 408: Request timeout](#)

- [HTTP 502: Bad gateway](#)
- [HTTP 503: Service Unavailable](#)
- [HTTP 504: Gateway Timeout](#)

## HTTP 400: BAD\_REQUEST

説明: クライアントが無効なリクエストを送信したことを示します。

原因 1: 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ました。たとえば、リクエストの URL にスペースを含めることはできません。

原因 2: クライアントが HTTP CONNECT メソッドを使用しました。このメソッドは Elastic Load Balancing ではサポートされていません。

解決方法: 直接インスタンスに接続し、クライアントリクエストの詳細をキャプチャします。ヘッダーと URL で誤った形式のリクエストを確認します。リクエストが HTTP 仕様を満たすことを確認します。HTTP CONNECT が使用されていないことを確認します。

## HTTP 405: METHOD\_NOT\_ALLOWED

説明: メソッドの長さが無効であることを示しています。

原因: リクエストヘッダー内のメソッドの長さが 127 文字を超えています。

解決方法: メソッドの長さを確認します。

## HTTP 408: Request timeout

説明: クライアントがリクエストをキャンセルしたが、リクエスト全体の送信に失敗したことを示します。

原因 1: ネットワークの中断またはリクエストの構造の問題 (ヘッダーの形式が完全ではない、指定されたコンテンツのサイズが実際に送信されたコンテンツのサイズと一致しないなど)。

解決方法 1: リクエストを生成しているコードを調べ、そのコードを実際のリクエストをより詳細に調べることができる登録済みインスタンス (または開発/テスト環境) に直接送信します。

原因 2: クライアントとの接続が閉じています (ロードバランサーは応答を送信できません)。

解決方法 2: リクエストを送信するマシンでパケットスニッファーを使用して、応答が送信される前にクライアントが接続を閉じないことを確認してください。

## HTTP 502: Bad gateway

説明: 登録されたインスタンスから送信された応答をロードバランサーが解析できなかったことを示します。

原因: インスタンスからの応答の形式が適切でないか、ロードバランサーに問題があります。

解決方法: インスタンスから送信された応答が HTTP 仕様に準拠していることを確認します。[AWS サポートセンター](#)にアクセスしてサポートをお求めください。

## HTTP 503: Service Unavailable

説明: ロードバランサーまたは登録されたインスタンスが原因でエラーが発生していることを示します。

原因 1: ロードバランサーにリクエストを処理する能力が不足しています。

解決方法: これは一時的な問題であり、数分以上は継続しません。問題が解決しない場合は、[AWS サポートセンター](#)にアクセスしてサポートをお求めください。

原因 2: 登録されたインスタンスがありません。

解決方法 2: ロードバランサーによる応答が設定された利用可能ゾーンごとに 1 つ以上のインスタンスを登録します。これを確認するには、CloudWatch の HealthyHostCount メトリクスを確認します。各アベイラビリティゾーンにインスタンスが登録されているかどうか不明な場合は、クロスゾーン負荷分散を有効にすることをお勧めします。詳細については、「[Classic Load Balancer のクロスゾーン負荷分散の設定](#)」を参照してください。

原因 3: 正常なインスタンスがありません。

解決方法 3: ロードバランサーの応答を設定しているすべての利用可能ゾーンに正常なインスタンスがあることを確認します。これを確認するには、HealthyHostCount メトリクスを確認します。

原因 4: サージキューがいっぱいです。

解決方法 4: インスタンスに、リクエスト率に対応するための十分な容量があることを確認します。これを確認するには、SpilloverCount メトリクスを確認します。

## HTTP 504: Gateway Timeout

説明: リクエストがアイドルタイムアウト期間内に完了しなかったためロードバランサーが接続を閉じたことを示します。

原因 1: アプリケーションの応答が、設定されているアイドルタイムアウトよりも長くかかっています。

解決方法 1: HTTPCode\_ELB\_5XX および Latency メトリクスをモニタリングします。これらのメトリクスに増加があった場合は、アイドルタイムアウト期間内に応答しないアプリケーションが原因である可能性があります。タイムアウトしているリクエストの詳細については、ロードバランサーのアクセスログを有効にし、Elastic Load Balancing によって生成されたログ内の 504 レスポンスコードを確認します。必要に応じて、キャパシティーを増やしたり、設定されているアイドルタイムアウトを長くしたりして、時間のかかるオペレーション (大容量ファイルのアップロードなど) が完了できるようにします。詳細については、[Classic Load Balancer でのアイドル接続のタイムアウト設定](#) および「[Elastic Load Balancing のレイテンシーが高い場合のトラブルシューティング方法を教えてください。](#)」を参照してください。

原因 2: 登録されたインスタンスが Elastic Load Balancing への接続を終了中です。

解決方法 2: EC2 インスタンスのキープアライブ設定を有効にし、キープアライブタイムアウトが、ロードバランサーのアイドルタイムアウト設定より大きい値に設定されていることを確認します。

## Classic Load Balancer のトラブルシューティング: レスポンスコードのメトリクス

ロードバランサーは、クライアントに送信された HTTP 応答コードのメトリクスを Amazon CloudWatch に送信することで、エラーの原因がロードバランサーなのか、登録済みインスタンスなのかを特定します。ロードバランサーに CloudWatch から返されるメトリクスを、問題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、「[Classic Load Balancer の CloudWatch メトリクス](#)」を参照してください。

次は、CloudWatch からロードバランサーに返される応答コードのメトリクス、考えられる原因、および問題の解決のために取るべきステップを示しています。

### 応答コードのメトリクス

- [HTTPCode\\_ELB\\_4XX](#)
- [HTTPCode\\_ELB\\_5XX](#)
- [HTTPCode\\_Backend\\_2XX](#)
- [HTTPCode\\_Backend\\_3XX](#)
- [HTTPCode\\_Backend\\_4XX](#)
- [HTTPCode\\_Backend\\_5XX](#)

## HTTPCode\_ELB\_4XX

原因: クライアントからのリクエストが誤った形式であるか、キャンセルされました。

### ソリューション

- 「[HTTP 400: BAD\\_REQUEST](#)」を参照してください
- 「[HTTP 405: METHOD\\_NOT\\_ALLOWED](#)」を参照してください
- 「」を参照してください[HTTP 408: Request timeout](#)

## HTTPCode\_ELB\_5XX

原因: ロードバランサーまたは登録されたインスタンスがエラーの原因であるか、またはロードバランサーが応答を解析できませんでした。

### ソリューション

- 「[HTTP 502: Bad gateway](#)」を参照してください
- 「[HTTP 503: Service Unavailable](#)」を参照してください
- 「」を参照してください[HTTP 504: Gateway Timeout](#)

## HTTPCode\_Backend\_2XX

原因: 登録済みインスタンスから正常な応答が返されました。

解決方法: ありません。

## HTTPCode\_Backend\_3XX

原因: 登録済みインスタンスからリダイレクト応答が送信されました。

解決方法: インスタンスのアクセスログまたはエラーログを確認して、原因を特定します。リクエストをインスタンスに直接送信 (ロードバランサーをバイパス) して応答を表示します。

## HTTPCode\_Backend\_4XX

原因: 登録済みインスタンスからクライアントエラー応答が送信されました。

解決方法: インスタンスのアクセスログまたはエラーログを表示して、問題の原因を判断します。リクエストをインスタンスに直接送信 (ロードバランサーをバイパス) して応答を表示します。

#### Note

クライアントが Transfer-Encoding: chunked ヘッダーで開始された HTTP リクエストをキャンセルする場合は、クライアントがリクエストをキャンセルしても、ロードバランサーはそのリクエストをインスタンスに転送するという既知の問題があります。これにより、バックエンドエラーが発生する場合があります。

## HTTPCode\_Backend\_5XX

原因: 登録済みインスタンスからサーバーエラー応答が送信されました。

解決方法: インスタンスのアクセスログまたはエラーログを確認して、原因を特定します。リクエストをインスタンスに直接送信 (ロードバランサーをバイパス) して応答を表示します。

#### Note

クライアントが Transfer-Encoding: chunked ヘッダーで開始された HTTP リクエストをキャンセルする場合は、クライアントがリクエストをキャンセルしても、ロードバランサーはそのリクエストをインスタンスに転送するという既知の問題があります。これにより、バックエンドエラーが発生する場合があります。

## Classic Load Balancer のトラブルシューティング: ヘルスチェック

ロードバランサーは Elastic Load Balancing によって提供されたデフォルトのヘルスチェック設定、またはユーザーが指定するヘルスチェック設定を使用して、登録されたインスタンスの状態をチェックします。ヘルスチェック設定には、プロトコル、ping ポート、ping パス、応答タイムアウト、ヘルスチェック間隔などの情報が含まれます。インスタンスは、ヘルスチェックの間隔内に 200 応答コードを返せば、正常と見なされます。詳細については、「[Classic Load Balancer のインスタンスのヘルスチェック](#)」を参照してください。

一部またはすべてのインスタンスの現在の状態が OutOfService で、説明フィールドに Instance has failed at least the Unhealthy Threshold number of health checks consecutively というメッセージが表示された場合は、インスタンスはロードバランサーのヘル

スチェックに失敗しています。以下は確認する必要がある問題、考えられる原因、問題の解決のためのステップです。

## 問題点

- [ヘルスチェックのターゲットページのエラー](#)
- [インスタンスへの接続がタイムアウトした](#)
- [パブリックキー認証が失敗する](#)
- [インスタンスがロードバランサーからのトラフィックを受信しない](#)
- [インスタンスのポートが開いていない](#)
- [Auto Scaling グループのインスタンスが ELB ヘルスチェックに失敗する](#)

## ヘルスチェックのターゲットページのエラー

問題: 指定された ping ポートと ping パス (例: HTTP:80/index.html) 上のインスタンスに対して発行された HTTP GET リクエストで、200 以外の応答コードが受信されます。

原因 1: インスタンスでターゲットページが設定されていない。

ソリューション 1: 各登録インスタンスで、ターゲットページ (index.html など) を作成し、そのパスを ping パスとして指定します。

原因 2: 応答の Content-Length ヘッダーの値が設定されていない。

解決方法 2: 応答に本文が含まれている場合、Content-Length ヘッダーの値を 0 以上に設定するか、Transfer-Encoding の値を "chunked" に設定します。

原因 3: アプリケーションがロードバランサーからリクエストを受け取るか、200 応答コードを返すように設定されていない。

解決方法 3: インスタンス上のアプリケーションを確認し、原因を調査します。

## インスタンスへの接続がタイムアウトした

問題: ロードバランサーから EC2 インスタンスへのヘルスチェックのリクエストがタイムアウトしたか、断続的に失敗しています。

まず、インスタンスに直接接続して問題を確認します。そのインスタンスのプライベート IP アドレスを使用して、ネットワーク内からインスタンスに接続することをお勧めします。

TCP 接続では、次のコマンドを使用します。

```
telnet private-IP-address-of-the-instance port
```

HTTP または HTTPS 接続では、次のコマンドを使用します。

```
curl -I private-IP-address-of-the-instance:port/health-check-target-page
```

HTTP/HTTPS 接続を使用している場合に、200 以外の応答が返されたときは、「[ヘルスチェックのターゲットページのエラー](#)」を参照してください。インスタンスに直接接続できる場合は、以下を確認してください。

原因 1: インスタンスが、設定された応答タイムアウト期間内に応答できない。

解決方法 1: ロードバランサーのヘルスチェック設定で、応答タイムアウト設定を調整します。

原因 2: インスタンスに大きな負荷がかかっている、設定された応答タイムアウト期間よりも応答に時間がかかっている。

解決策 2:

- モニタリンググラフで、CPU の過度の使用率について確認します。詳細については、Amazon EC2 ユーザーガイドの「[特定の EC2 インスタンスの統計を取得する](#)」を参照してください。
- EC2 インスタンスに接続して、メモリ、制限など他のアプリケーションリソースの使用状況を確認します。
- 必要に応じて、さらにインスタンスを追加するか、Auto Scaling を有効にします。詳細については [Amazon EC2 Auto Scaling ユーザーガイド](#) を参照してください。

原因 3: HTTP または HTTPS 接続を使用していて、ping パスフィールドで指定したターゲットページ (例: HTTP:80/index.html) でヘルスチェックを実行している場合、ターゲットページからの応答は、設定したタイムアウトより長くかかる場合があります。

解決方法 3: よりシンプルなヘルスチェックのターゲットページを使用するか、ヘルスチェック間隔の設定を調整します。

## パブリックキー認証が失敗する

問題: バックエンド認証が有効で、HTTPS または SSL プロトコルを使用するように設定されたロードバランサーが、パブリックキー認証に失敗します。

原因: SSL 証明書のパブリックキーが、ロードバランサーで設定されたパブリックキーと一致しない。s\_client コマンドを使用して、証明書チェーン内のサーバー証明書のリストを表示します。詳細については、OpenSSL ドキュメントの「[s\\_client](#)」を参照してください。

解決方法: SSL 証明書の更新が必要な場合があります。SSL 証明書が最新状態である場合は、その証明書をロードバランサーに再インストールしてみます。詳細については、「[Classic Load Balancer の SSL 証明書の置き換え](#)」を参照してください。

## インスタンスがロードバランサーからのトラフィックを受信しない

問題: インスタンスのセキュリティグループが、ロードバランサーからのトラフィックをブロックしています。

インスタンスでパケットキャプチャを実行して、問題を確認します。次のコマンドを使用します。

```
# tcpdump port health-check-port
```

原因 1: インスタンス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トラフィックを許可していません。

解決方法 1: ロードバランサーからのトラフィックを許可するようにインスタンスのセキュリティグループを編集します。ロードバランサーのセキュリティグループからのトラフィックを許可するルールを追加します。

原因 2: ロードバランサーのセキュリティグループで、EC2 インスタンスへのトラフィックが許可されていません。

解決方法 2: ロードバランサーのセキュリティグループを編集して、サブネットおよび EC2 インスタンスへのトラフィックを許可します。

セキュリティグループの管理方法の詳細については、「[Classic Load Balancer のセキュリティグループの設定](#)」を参照してください。

## インスタンスのポートが開いていない

問題: ロードバランサーによって EC2 インスタンスに送信されるヘルスチェックが、ポートまたはファイアウォールによってブロックされます。

次のコマンドを使用して問題を確認します。

```
netstat -ant
```

原因: 指定されたポートまたはリスナーポート (異なる設定の場合) が開いていません。ヘルスチェックに指定したポートと、リスナーポートの両方が開いていて、リッスンしている必要があります。

解決方法: リスナーポートと、ヘルスチェックで指定したポート (異なる設定の場合) をインスタンスで開き、ロードバランサーのトラフィックを受信できるようにします。

## Auto Scaling グループのインスタンスが ELB ヘルスチェックに失敗する

問題: Auto Scaling グループのインスタンスがデフォルトの Auto Scaling ヘルスチェックには合格するにもかかわらず、ELB ヘルスチェックには合格しません。

原因: Auto Scaling は EC2 ヘルスチェックを使用して、インスタンスに存在するハードウェアとソフトウェアの問題を検出します。一方、ロードバランサーはリクエストをインスタンスに送信して 200 応答コードを待つか、インスタンスとの TCP 接続を確立することによって (TCP ベースのヘルスチェックの場合) ヘルスチェックを実行します。

インスタンスは、インスタンスで実行するアプリケーションに、ロードバランサーがインスタンスを停止中と判断する問題があることが原因で、ELB ヘルスチェックに失敗する場合があります。このインスタンスは、Auto Scaling ヘルスチェックに合格する場合があります。EC2 のステータスチェックに基づいて正常な状態と見なされているため、Auto Scaling ポリシーによる置き換えは行われません。

解決方法: Auto Scaling グループに対し ELB ヘルスチェックを使用します。ELB ヘルスチェックを使用すると、Auto Scaling は、インスタンスのステータスチェックと ELB ヘルスチェックの両方の結果を確認して、インスタンスのヘルスステータスを判断します。詳細については、Amazon EC2 Auto Scaling 開発者ガイドの「[Auto Scaling グループへの Elastic Load Balancing ヘルスチェックの追加](#)」を参照してください。

## Classic Load Balancer のトラブルシューティング: クライアント接続

### 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。詳細については、「[Classic Load Balancer のセキュリティグループの設定](#)」を参照してください。

## 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。

## 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。

# Classic Load Balancer のトラブルシューティング: インスタンスの登録

ロードバランサーにインスタンスを登録する場合、ロードバランサーがインスタンスへのリクエストの送信を開始する前に行うべき手順があります。

以下は EC2 インスタンスを登録するときにロードバランサーが遭遇する可能性のある問題、考えられる原因、問題の解決のためのステップです。

## 問題点

- [EC2 インスタンスの登録に時間がかかりすぎる](#)
- [有料 AMI から起動したインスタンスを登録できない](#)

## EC2 インスタンスの登録に時間がかかりすぎる

問題: 登録した EC2 インスタンスが InService 状態になるまで予期されるより時間がかかっています。

原因: インスタンスのヘルスチェックが失敗している可能性があります。最初のインスタンスを登録するステップ (最大約 30 秒かかります) が完了した後、ロードバランサーがヘルスチェックリクエストの送信を開始します。ヘルスチェックが 1 つ成功するまで、インスタンスは InService になりません。

解決方法: 「[インスタンスへの接続がタイムアウトした](#)」を参照してください。

## 有料 AMI から起動したインスタンスを登録できない

問題: 有料 AMI を使用して起動したインスタンスが Elastic Load Balancing に登録されません。

原因: このインスタンスは、有料 AMI を使用しながら [Amazon DevPay](#) から起動された可能性があります。

解決方法: Elastic Load Balancing では、有料 AMI を使用して [Amazon DevPay](#) から起動したインスタンスの登録はサポートされていません。[AWS Marketplace](#) からであれば、有料 AMI を使用可能です。から有料 AMI を既に使用 AWS Marketplace していて、その有料 AMI から起動されたインスタンスを登録できない場合は、[AWS サポート センター](#)にアクセスしてください。

## Classic Load Balancer のクォータ

AWS アカウントには、サービスごとに AWS、以前は制限と呼ばれていたデフォルトのクォータがあります。特に明記していない限り、クォータはリージョン固有です。

Classic Load Balancer のクォータを表示するには、[サービスクォータコンソール](#)を開きます。ナビゲーションペインで、[AWS services]、[Elastic Load Balancing] の順に選択します。また、Elastic Load Balancing 用に [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。

AWS アカウントには、Classic Load Balancer に関連する次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 Classic Load Balancer	20	<a href="#">あり</a>
Classic Load Balancer あたりのリスナー	100	<a href="#">あり</a>
Classic Load Balancer あたりの登録済みインスタンス	1,000	<a href="#">あり</a>

# Classic Load Balancer のドキュメント履歴

次の表に、Classic Load Balancer のリリース情報を示します。

変更	説明	日付
<a href="#">アクセスログと接続ログのバケットポリシー</a>	このリリース以前にユーザーが使用したバケットポリシーは、リージョンが利用可能になったのが 2022 年 8 月より前か後かによって異なりました。このリリースでは、新しいバケットポリシーがすべてのリージョンでサポートされています。レガシーバケットポリシーは引き続きサポートされています。	2025 年 9 月 10 日
<a href="#">Desync 軽減モード</a>	desync 軽減モードのサポートが追加されました。詳細は、「 <a href="#">Classic Load Balancer の desync 軽減モードの設定</a> 」を参照してください。	2020 年 8 月 17 日
<a href="#">Classic Load Balancer</a>	Application Load Balancer と Network Load Balancer の導入により、2016-06-01 API で作成されたロードバランサーは、Classic Load Balancer と呼ばれるようになりました。これらのロードバランサーの違いの詳細については、「 <a href="#">Elastic Load Balancing の特徴</a> 」を参照してください。	2016 年 8 月 11 日
<a href="#">AWS Certificate Manager (ACM) のサポート</a>	ACM から SSL/TLS 証明書をリクエストし、その証明書を	2016 年 1 月 21 日

ロードバランサーにデプロイできません。詳細については、[「Classic Load Balancer 用の SSL/TLS 証明書」](#)を参照してください。

### [追加のポートのサポート](#)

ロードバランサーは、範囲 1 ~ 65535 のすべてのポートをリッスンできます。詳細については、の [「Classic Load Balancer のリスナー」](#)を参照してください。

2015 年 9 月 15 日

### [アクセスログエントリの追加のフィールド](#)

user\_agent 、ssl\_cipher 、および ssl\_protocol フィールドを追加しました。詳細については、[「アクセスログファイル」](#)を参照してください。

2015 年 5 月 18 日

### [ロードバランサーのタグ付けのサポート](#)

このリリース以降、Elastic Load Balancing CLI (ELB CLI) は、複数の AWS サービスを管理するための統合ツールである AWS Command Line Interface (AWS CLI) に置き換えられました。ELB CLI バージョン 1.0.35.0 (2014 年 7 月 24 日付け) 以降にリリースされた新機能は、AWS CLI のみに含まれています。現在 ELB CLI を使用しているお客様には、AWS CLI への切り替えを推奨しています。詳細については、「AWS Command Line Interface ユーザーガイド」を参照してください。

2014 年 8 月 11 日

<a href="#">アイドル接続のタイムアウト</a>	ロードバランサーに対するアイドル接続のタイムアウトを設定できます。	2014 年 7 月 24 日
<a href="#">ユーザーおよびグループに特定のロードバランサーまたは API アクションに対するアクセスを許可する機能のサポート</a>	ユーザーおよびグループに特定のロードバランサーまたは API アクションに対するアクセスを許可するポリシーを作成することができます。	2014 年 5 月 12 日
<a href="#">のサポート AWS CloudTrail</a>	CloudTrail を使用して、ELB API、ELB CLI AWS マネジメントコンソール、または AWS アカウント を使用して、によって、またはに代わって行われた API コールをキャプチャできます AWS CLI。	2014 年 4 月 4 日
<a href="#">Connection Draining</a>	Connection Draining に関する情報を追加しました。このサポートを使用すると、登録済みインスタスが登録解除中であるとき、または登録済みインスタスで異常が発生したときに、ロードバランサーが、既存の接続を開いたまま、そのインスタスに新しいリクエストを送信しないようにできます。詳細については、「 <a href="#">Classic Load Balancer の Connection Draining の設定</a> 」を参照してください。	2014 年 3 月 20 日

## [アクセスログ](#)

ロードバランサーに送信されるリクエストに関する詳細情報を取得し、Amazon S3 バケットに保存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、「[Classic Load Balancer のアクセスログ](#)」を参照してください。

2014 年 3 月 6 日

## [TLSv1.1-1.2 のサポート](#)

HTTPS/SSL リスナーを使用したロードバランサーの TLSv1.1-1.2 プロトコル サポートに関する情報を追加しました。このサポートでは、Elastic Load Balancing は、事前定義された SSL ネゴシエーション設定も更新しません。更新された事前定義済み SSL ネゴシエーション設定の詳細については、「[Classic Load Balancers での SSL ネゴシエーション設定](#)」を参照してください。現在の SSL ネゴシエーション設定の更新については、「[Classic Load Balancer の SSL ネゴシエーション設定の更新](#)」を参照してください。

2014 年 2 月 19 日

## [クロスゾーンロードバランサー](#)

ロードバランサーのクロスゾーン負荷分散の有効化に関する情報を追加しました。詳細については、「[Classic Load Balancer のクロスゾーン負荷分散の設定](#)」を参照してください。

2013 年 11 月 6 日

## [追加の CloudWatch のメトリクス](#)

Elastic Load Balancing によりレポートされる、CloudWatch の新しいメトリクスに関する情報を追加しました。詳細については、「[Classic Load Balancer の CloudWatch メトリクス](#)」を参照してください。

2013 年 10 月 28 日

## [プロキシプロトコルのサポート](#)

TCP/SSL 接続用に設定されたロードバランサーのプロキシプロトコルサポートに関する情報を追加しました。詳細については、「[プロキシプロトコルヘッダー](#)」を参照してください。

2013 年 7 月 30 日

## [DNS フェイルオーバーのサポート](#)

ロードバランサーに対する Amazon Route 53 DNS フェイルオーバーの設定に関する情報を追加。詳細については、「[ロードバランサーの Amazon Route 53 DNS フェイルオーバーを使用する](#)」を参照してください。

2013 年 6 月 3 日

## [コンソールでの CloudWatch メトリクス表示とアラーム作成のサポート](#)

コンソールを使用して、指定されたロードバランサーについて CloudWatch メトリクスを表示したりアラームを作成したりするための情報を追加しました。詳細については、「[Classic Load Balancer の CloudWatch メトリクス](#)」を参照してください。

2013 年 3 月 28 日

[デフォルト VPC 内での EC2 インスタンスの登録のサポート](#)

デフォルト VPC 内で起動された EC2 インスタンスに関するサポートを追加しました。

2013 年 3 月 11 日

[内部ロードバランサー](#)

このリリースでは、Virtual Private Cloud (VPC) のロードバランサーを内部向けまたはインターネット向けに作成できます。内部のロードバランサーには、プライベート IP アドレスに解決する、パブリックに解決可能な DNS 名が含まれています。インターネット向けロードバランサーには、パブリック IP アドレスに解決される、パブリックに解決可能な DNS 名が含まれています。詳細については、「[内部 Classic Load Balancer の作成](#)」を参照してください。

2012 年 6 月 10 日

[リスナー、暗号設定、SSL 証明書を管理するためのコンソールでのサポート](#)

詳細については、「[Classic Load Balancer の HTTPS リスナーの設定](#)」と、「[Classic Load Balancer の SSL 証明書の置き換え](#)」を参照してください。

2012 年 5 月 18 日

[Amazon VPC での Elastic Load Balancing のサポート](#)

Virtual Private Cloud (VPC) でロードバランサーを作成するためのサポートを追加しました。

2011 年 11 月 21 日

<a href="#">Amazon CloudWatch</a>	CloudWatch を使用してロードバランサーをモニタリングできません。詳細については、「 <a href="#">Classic Load Balancer の CloudWatch メトリクス</a> 」を参照してください。	2011 年 10 月 17 日
<a href="#">追加のセキュリティ機能</a>	SSL 暗号、バックエンド SSL、バックエンドサーバー認証を設定できません。詳細は、「 <a href="#">HTTPS リスナーを使用する Classic Load Balancer の作成</a> 」を参照してください。	2011 年 8 月 30 日
<a href="#">Zone Apex ドメイン名</a>	詳細は、「 <a href="#">Classic Load Balancer のカスタムドメイン名の設定</a> 」	2011 年 5 月 24 日
<a href="#">X-Forwarded-Proto ヘッダーと X-Forwarded-Port ヘッダーのサポート</a>	X-Forwarded-Proto ヘッダーは発信元のリクエストのプロトコルを示し、X-Forwarded-Port ヘッダーは発信元のリクエストのポートを示します。リクエストにこれらのヘッダーを追加することによって、お客様はロードバランサーへの着信リクエストが暗号化されているかどうかや、ロードバランサーでリクエストを受信したポートを特定できません。Network Load Balancer の詳細については、「 <a href="#">HTTP ヘッダーと Classic Load Balancer</a> 」を参照してください。	2010 年 10 月 27 日

<a href="#">HTTPS のサポート</a>	このリリースによって、SSL/TLS プロトコルを利用してトラフィックを暗号化することや、SSL 処理をアプリケーションインスタンスからロードバランサーにオフロードすることができます。また、この機能によって、個々のアプリケーションインスタンスで証明書を管理するのではなく、ロードバランサーで SSL サーバー証明書を集中管理できます。	2010 年 10 月 14 日
<a href="#">AWS Identity and Access Management (IAM) のサポート</a>	IAM のサポートが追加されました。	2010 年 9 月 2 日
<a href="#">スティッキーセッション</a>	詳細は、「 <a href="#">Classic Load Balancer のスティッキーセッションの設定</a> 」を参照してください。	2010 年 4 月 7 日
<a href="#">AWS SDK for Java</a>	SDK for Java のサポートが追加されました。	2010 年 3 月 22 日
<a href="#">AWS SDK for .NET</a>	のサポートが追加されました SDK for .NET。	2009 年 11 月 11 日
<a href="#">新しいサービス</a>	Elastic Load Balancing の最初のパブリックベータリリース。	2009 年 5 月 18 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。