



開発者ガイド

Amazon CloudFront



Amazon CloudFront: 開発者ガイド

Copyright © 2024 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 のものではない製品またはサービスにも関連して、お客様に混乱を招いたり Amazon の信用を傷つけたり失わせたりするいかなる形においても使用することはできません。Amazon が所有しない他の商標はすべてそれぞれの所有者に帰属します。所有者は必ずしも Amazon との提携や関連があるわけではありません。また、Amazon の支援を受けているとはかぎりません。

Table of Contents

Amazon とは CloudFront	1
コンテンツを配信 CloudFront するように を設定する方法	2
ユースケース	4
静的ウェブサイトのコンテンツ配信の加速	4
オンデマンドビデオおよびライブストリーミングビデオの配信	5
システム処理全体で特定のフィールドを暗号化する	5
エッジのカスタマイズ	5
Lambda@Edge カスタマイズを使用したプライベートコンテンツの供給	6
がコンテンツを CloudFront 配信する方法	7
がユーザーにコンテンツを CloudFront 配信する方法	7
がリージョン別エッジキャッシュと CloudFront 連携する方法	8
CloudFront エッジサーバー	10
CloudFront マネージドプレフィックスリストを使用する	11
アクセス CloudFront	12
CloudFront 料金	12
Savings Bundle	14
CloudFront デистриビューションの価格クラスを選択	19
AWS SDK の操作	21
開始	22
設定	22
AWS アカウントへのサインアップ	22
管理ユーザーの作成	23
AWS Command Line Interface または AWS Tools for Windows PowerShell の設定	24
AWS SDK のダウンロード	24
簡単なデистриビューションの使用を開始する	24
前提条件	25
ステップ 1: バケットを作成する	25
ステップ 2: コンテンツをアップロードする	26
ステップ 3: デистриビューションを作成する	27
ステップ 4: コンテンツにアクセスする	30
ステップ 5: クリーンアップ	31
ヒント	31
安全な静的ウェブサイトの使用開始	32
ソリューションの概要	33

ソリューションのデプロイ	33
ディストリビューションの使用	39
ディストリビューションの概要	39
ディストリビューションで使用できるアクション	40
ディストリビューションの作成および更新で必須である API フィールド	41
ディストリビューションの作成、更新、および削除	45
ディストリビューションを作成するためのステップ	45
ディストリビューションの作成	47
指定する値	48
表示される値	84
ディストリビューションのテスト	86
ディストリビューションの更新	87
ディストリビューションのタグ付け	88
ディストリビューションを削除する	91
継続的デプロイを使用して変更を安全にテストする	92
CloudFront 継続的デプロイを使用するワークフロー	94
ステージングディストリビューションと継続的デプロイポリシーの使用	95
ステージングディストリビューションのモニタリング	105
継続的デプロイの仕組みを理解する	105
継続的デプロイに関するクォータとその他の考慮事項	108
さまざまなオリジンの使用	109
Amazon S3 バケットの使用	109
MediaStore コンテナまたは MediaPackage チャンネルの使用	118
Application Load Balancer の使用	118
Lambda 関数 URL の使用	119
Amazon EC2 (または他のカスタムオリジン) の使用	120
CloudFront オリジングループの使用	121
カスタム URL の使用	121
代替ドメイン名の追加	122
代替ドメイン名を別のディストリビューションに移動する	126
代替ドメイン名の削除	132
代替ドメイン名での * ワイルドカードの使用	133
代替ドメイン名を使用するための要件	134
代替ドメイン名の使用に対する制限	135
の使用 WebSockets	137
WebSocket プロトコルの仕組み	138

WebSocket の要件	138
推奨される設定	139
ポリシーの使用	140
キャッシュキーの管理	141
キャッシュポリシーの作成	141
キャッシュポリシーについて	146
管理キャッシュポリシーの使用	153
キャッシュキーについて	157
オリジンリクエストの制御	161
オリジンリクエストポリシーの作成	162
オリジンリクエストポリシーについて	167
管理オリジンリクエストポリシーの使用	169
CloudFront リクエストヘッダーの追加	175
オリジンリクエストポリシーとキャッシュポリシーの連携方法を理解する	179
レスポンスヘッダーの追加または削除	182
レスポンスヘッダーポリシーの作成	183
マネージドレスポンスヘッダーポリシーの使用	190
レスポンスヘッダーポリシーの理解	196
コンテンツの追加、削除、または置き換え	211
コンテンツの追加とアクセス	211
既存のコンテンツの更新	212
バージョン付きのファイル名を使用した既存ファイルの更新	212
同じファイル名を使用した既存コンテンツの更新	213
コンテンツを削除 CloudFront して配信しない	214
ファイルの URL のカスタマイズ	214
独自のドメイン名を使用する (example.com)	215
URL に末尾のスラッシュ (/) を使用する	215
制限されたコンテンツの署名付き URL の作成	216
デフォルトのルートオブジェクトの指定	216
デフォルトのルートオブジェクトを指定する方法	216
デフォルトのルートオブジェクトの仕組み	218
ルートオブジェクトを定義しない場合の CloudFront の動作	219
ファイルの無効化	220
ファイルは無効化するか、バージョン付きファイル名を使用するかを選択	221
無効にするファイルの決定	222
無効にするファイルの指定	222

コンソールを使用したファイルの無効化	226
CloudFront API を使用したファイルの無効化	229
同時無効化リクエストの最大制限	229
ファイルの無効化に対する支払い	230
圧縮ファイルの供給	230
オブジェクトを圧縮 CloudFront するための の設定	231
CloudFront 圧縮の仕組み	231
CloudFront 圧縮に関する注意事項	233
CloudFront が圧縮するファイルタイプ	235
ETag ヘッダーの変換	236
カスタムエラーレスポンスの生成	237
エラーレスポンス動作を設定する	238
特定の HTTP ステータスコードに対応するカスタムエラーページの作成	239
オブジェクトとカスタムエラーページを別の場所に格納する	241
によって返されるレスポンスコードの変更 CloudFront	242
がエラーを CloudFront キャッシュする時間の制御	243
AWS WAF 保護の使用	245
新しいディストリビューションで AWS WAF を有効にする	246
既存のウェブ ACL の使用	247
既存のディストリビューションで AWS WAF を有効にする	247
既存のウェブ ACL の使用	248
セキュリティ保護を無効にする	248
レート制限の設定	249
CloudFront セキュリティダッシュボードの使用	249
AWS WAF の有効化	250
トレンドデータの理解	251
Bot Control の有効化	252
ログの理解	254
CloudFront 地理的制限の管理	255
セキュリティダッシュボードの料金	255
コンテンツへのセキュアなアクセスとアクセス制限の設定	256
で HTTPS を使用する CloudFront	256
ビューワーと の間で HTTPS を必須にする CloudFront	258
カスタムオリジンに対して HTTPS を必須にする	260
Amazon S3 オリジンに対して HTTPS を必須にする	263
ビューワーと の間でサポートされているプロトコルと暗号 CloudFront	265

とオリジン間でサポートされているプロトコル CloudFront と暗号	271
HTTPS 接続料金	273
代替ドメイン名と HTTPS の使用	273
が HTTPS リクエスト CloudFront を処理する方法の選択	274
で SSL/TLS 証明書を使用するための要件 CloudFront	277
での SSL/TLS 証明書の使用に関するクォータ CloudFront (ビューワーと間の HTTPS CloudFront のみ)	282
代替ドメイン名と HTTPS の設定	284
SSL/TLS RSA 証明書内のパブリック (公開) キーのサイズの確認	288
SSL/TLS 証明書のクォータの引き上げ	289
SSL/TLS 証明書の更新	290
カスタム SSL/TLS 証明書からデフォルト CloudFront 証明書に戻す	292
独自 SSL/TLS 証明書を専用 IP アドレスから SNI に切り替える	293
署名付き URL と署名付き Cookie を使用したコンテンツの制限	294
プライベートコンテンツ提供の概要	294
プライベートコンテンツ提供のためのタスクリスト	297
署名者の指定	298
署名付き URL と署名付き Cookie の選択	308
署名付き URL の使用	309
署名付き Cookie の使用	332
Linux コマンドおよび OpenSSL を使用した Base64 エンコードおよび暗号化	354
署名付き URL のコード例	355
AWS オリジンへのアクセスの制限	384
オリジンへのアクセス MediaStoreの制限	384
Amazon S3 オリジンへのアクセスの制限	392
Application Load Balancers へのアクセスを制限する	407
リクエスト CloudFront にカスタム HTTP ヘッダーを追加するための の設定	408
特定のヘッダーを含むリクエストだけを転送するように Application Load Balancer を構成 する	410
(オプション) このソリューションのセキュリティ向上	415
コンテンツを地理的に制限する	416
CloudFront 地理的制限の使用	416
サードパーティーの位置情報サービスの使用	418
フィールドレベル暗号化を使用した機密データの保護	420
フィールドレベル暗号化の概要	422
フィールドレベル暗号化の設定	422

オリジンでのデータフィールドの復号化	428
キャッシュの最適化と可用性	432
エッジロケーションでのキャッシュ	432
キャッシュヒット率の向上	433
がオブジェクトを CloudFront キャッシュする期間を指定する	433
Origin Shield の使用	433
クエリ文字列パラメータに基づくキャッシュ	434
Cookie 値に基づくキャッシュ	434
リクエストヘッダーに基づくキャッシュ	435
圧縮が不要な場合に Accept-Encoding ヘッダーを削除する	436
HTTP を使用したメディアコンテンツの提供	437
Origin Shield の使用	437
Origin Shield のユースケース	438
Origin Shield の AWS リージョンの選択	444
Origin Shield の有効化	446
Origin Shield のコストの見積もり	449
Origin Shield の高可用性	449
Origin Shield が他の CloudFront 機能とやり取りする方法	450
オリジンフェイルオーバーによる可用性の向上	451
オリジングループの作成	453
オリジンのタイムアウトと試行の制御	453
Lambda@Edge 関数でのオリジンフェイルオーバーの使用	455
オリジンフェイルオーバーでのカスタムエラーページの使用	456
キャッシュの有効期限の管理	457
ヘッダーを使用した個々のオブジェクトのキャッシュ保持期間の制御	458
古い (期限切れの) コンテンツの提供	459
がオブジェクトを CloudFront キャッシュする時間を指定する	461
Amazon S3 コンソールを使用したオブジェクトへのヘッダーの追加	468
キャッシュおよびクエリ文字列パラメータ	468
クエリ文字列の転送とキャッシュのためのコンソールおよび API の設定	470
キャッシュの最適化	471
クエリ文字列パラメータと CloudFront 標準ログ (アクセスログ)	472
Cookie に基づくコンテンツのキャッシュ	473
リクエストヘッダーに基づくコンテンツのキャッシュ	476
ヘッダーとディストリビューションの概要	476
キャッシュ条件に使用するヘッダーを選択する	478

CORS 設定を優先 CloudFront するように を設定する	479
デバイスタイプに基づいてキャッシュを設定する	480
ビューワーの言語に基づいてキャッシュを設定する	480
ビューワーの場所に基づいてキャッシュを設定する	480
リクエストのプロトコルに基づいてキャッシュを設定する	480
圧縮ファイルのキャッシュの設定	481
ヘッダーに基づくキャッシュがパフォーマンスに及ぼす影響	481
ヘッダーとヘッダー値の大文字小文字がキャッシュに及ぼす影響	481
がビューワーに CloudFront 返すヘッダー	481
トラブルシューティング	483
トラブルシューティング: ディストリビューション	483
CloudFront 代替ドメイン名を追加しようとする、 がInvalidViewerCertificateエラーを返します。	483
ディストリビューション内のファイルを表示できません	485
エラーメッセージ: Certificate: <certificate-id> is used by CloudFront	486
オリジンからのエラーレスポンスのトラブルシューティング	487
HTTP 400 ステータスコード (Bad Request)	487
HTTP 502 ステータスコード (Bad Gateway)	488
HTTP 502 ステータスコード (Lambda validation error)	491
HTTP 502 ステータスコード (DNS error)	492
HTTP 503 ステータスコード (関数実行エラー)	493
HTTP 503 ステータスコード (Lambda limit exceeded)	493
HTTP 503 ステータスコード (Service Unavailable)	494
HTTP 504 ステータスコード (Gateway Timeout)	495
負荷テスト CloudFront	499
リクエストとレスポンスの動作	501
Amazon S3 オリジンに対するリクエストとレスポンスの動作	501
が HTTP および HTTPS リクエスト CloudFront を処理する方法	501
がリクエスト CloudFront を処理して Amazon S3 オリジンに転送する方法	502
が Amazon S3 オリジンからのレスポンス CloudFront を処理する方法	509
「カスタムオリジンのリクエストとレスポンスの動作」	511
がリクエスト CloudFront を処理してカスタムオリジンに転送する方法	511
がカスタムオリジンからのレスポンス CloudFront を処理する方法	529
オリジングループに対するリクエストとレスポンスの動作	534
オリジンリクエストへのカスタムヘッダーの追加	535
オリジンのカスタムヘッダーのユースケース	535

オリジンリクエスト CloudFront にカスタムヘッダーを追加するための の設定	536
がオリジンリクエストに追加 CloudFront できないカスタムヘッダー	537
Authorization ヘッダーを転送する CloudFront ための の設定	538
レンジ GET の処理方法	538
範囲リクエストを使用して大きなオブジェクトをキャッシュする	539
がオリジンからの HTTP 3xx ステータスコード CloudFront を処理する方法	540
がオリジンからの HTTP 4xx および 5xx ステータスコード CloudFront を処理してキャッシュ する方法	540
カスタムエラーページを設定した場合に がエラー CloudFront を処理する方法	542
カスタムエラーページを設定していない場合に がエラー CloudFront を処理する方法	544
キャッシュする CloudFront HTTP 4xx および 5xx ステータスコード	545
ビデオオンデマンド (VOD) およびライブストリーミングビデオ	547
ストリーミングビデオについて: オンデマンドおよびライブストリーミングビデオ	547
ビデオオンデマンド (VOD) を配信する	548
Microsoft Smooth Streaming のビデオオンデマンドの設定	549
ライブストリーミングビデオの配信	551
AWS Elemental MediaStore をオリジンとして使用する動画の配信	552
AWS Elemental MediaPackage でフォーマットされたライブ動画の配信	553
エッジの関数	560
どの関数タイプを使うべきか	560
CloudFront 関数	563
チュートリアル: 単純な関数	564
チュートリアル: キー値のある関数	567
関数コードの記述	569
関数の管理	650
の使用 CloudFront KeyValueStore	667
Lambda@Edge を使用したカスタマイズ	680
開始	681
IAM のアクセス許可とロールの設定	692
関数の記述と作成	700
トリガーの追加	705
テストとデバッグ	713
関数とレプリカの削除	720
イベントの構造	721
リクエストとレスポンスを使用する	738
関数の例	744

エッジ関数に対する制限	782
すべてのエッジ機能に対する制限	782
CloudFront 関数の制限	789
Lambda@Edge に対する制限	790
レポート、メトリクス、ログ	794
AWS の 請求および使用状況レポート CloudFront	794
AWS の 請求レポート CloudFront	795
AWS の 使用状況レポート CloudFront	796
のAWS請求書とAWS使用状況レポートの解釈 CloudFront	797
CloudFront コンソールレポート	802
CloudFront キャッシュ統計レポート	805
CloudFront 人気オブジェクトレポート	810
CloudFront トップリファラレポート	816
CloudFront 使用状況レポート	819
CloudFront ビューワーレポート	826
Amazon による CloudFront メトリクスのモニタリング CloudWatch	838
CloudFront およびエッジ関数メトリクスの表示	839
アラームの作成	847
メトリクスデータのダウンロード	848
API を使用したメトリクスの取得	851
CloudFront およびエッジ関数のログ記録	857
リクエストのログ記録	857
エッジ関数をログ記録する	858
サービスアクティビティのログ記録	858
標準ログ (アクセスログ) の使用	859
リアルタイムログ	879
エッジ関数のログ	898
を使用した API リクエストのキャプチャ CloudTrail	900
AWS Config による設定変更の追跡	908
AWS Config で をセットアップする CloudFront	908
CloudFront 設定履歴の表示	910
セキュリティ	911
データ保護	912
転送中の暗号化	913
保管中の暗号化	914
コンテンツへのアクセス制限	914

Identity and Access Management	915
対象者	916
アイデンティティによる認証	916
ポリシーを使用したアクセス権の管理	920
Amazon と IAM の CloudFront 連携方法	922
アイデンティティベースポリシーの例	930
AWS マネージドポリシー	940
トラブルシューティング	945
ロギングとモニタリング	947
コンプライアンス検証	948
CloudFront コンプライアンスのベストプラクティス	949
耐障害性	951
CloudFront オリジンフェイルオーバー	951
インフラストラクチャセキュリティ	951
クォータ	953
一般的なクォータ	953
ディストリビューションの一般的なクォータ	954
ポリシーの一般的なクォータ	956
CloudFront 関数のクォータ	958
キー値ストアのクォータ	959
Lambda@Edge のクォータ	959
SSL 証明書のクォータ	961
無効化のクォータ	962
キーグループのクォータ	962
WebSocket 接続のクォータ	963
フィールドレベル暗号化のクォータ	963
Cookie のクォータ (従来のキャッシュ設定)	964
クエリ文字列のクォータ (従来のキャッシュ設定)	964
ヘッダーのクォータ	965
コードサンプル	967
アクション	967
ディストリビューションを作成する	968
関数の作成	978
キーグループを作成する	981
ディストリビューションを削除する	982
署名リソースを削除する	985

配信構成を取得する	987
配信の一覧表示	992
ディストリビューションを更新する	1001
パブリックキーをアップロードする	1014
シナリオ	1017
URL および cookies に署名する	1017
関連情報	1021
Amazon CloudFront のその他のドキュメント	1021
サポート情報	1021
CloudFront デベロッパーツールと SDKs	1022
Amazon Web Services ブログからのヒント	1022
ドキュメント履歴	1023
2022 より前の更新	1029
AWS 用語集	1038
.....	mxxxix

Amazon とは CloudFront

Amazon CloudFront は、.html、.css、.js、画像ファイルなどの静的および動的なウェブコンテンツをユーザーに配信するウェブサービスです。は、エッジロケーションと呼ばれるデータセンターの世界中のネットワークを通じてコンテンツを CloudFront 配信します。ユーザーが提供されているコンテンツをリクエストすると CloudFront、リクエストは最小のレイテンシー (遅延時間) を提供するエッジロケーションにルーティングされ、コンテンツは可能な限り最高のパフォーマンスで配信されます。

- コンテンツがすでにエッジロケーションにあり、レイテンシーが最も低い場合、はコンテンツをすぐに CloudFront 配信します。
- コンテンツがそのエッジロケーションにない場合、は、コンテンツの最終バージョンのソースとして識別した Amazon S3 バケット、MediaPackage チャンネル、HTTP サーバー (ウェブサーバーなど) など、定義したオリジンからコンテンツ CloudFront を取得します。

たとえば、イメージが CloudFront からではなく従来のウェブサーバーから供給されているとします。たとえば、sunsetphoto.png というイメージを、URL `https://example.com/sunsetphoto.png` を使用して提供するとします。

ユーザーは簡単にこの URL にアクセスしてそのイメージを表示できます。ただし、そのイメージが見つかるまで、リクエストがネットワークから別のネットワークに (インターネットを構成する相互接続ネットワークの複雑な集合経由で) ルーティングされたということを、おそらくユーザーは認識しません。

CloudFront は、各ユーザーリクエストをAWSバックボーンネットワーク経由で、コンテンツに最適なエッジロケーションにルーティングすることで、コンテンツの配信を高速化します。通常、これはビューワーに最速の配信を提供する CloudFront エッジサーバーです。AWS ネットワークを使用することでユーザーのリクエストが通過しなければならないネットワークの数が大幅に減少するので、パフォーマンスが向上します。ユーザーが経験するレイテンシー (ファイルの最初のバイトがロードされるまでの時間) が低くなり、データ転送速度が高くなります。

お客様のファイル (オブジェクトとしても知られる) のコピーが世界中の複数のエッジロケーションに保持される (つまりキャッシュされる) ので、信頼性と可用性の向上も得られます。

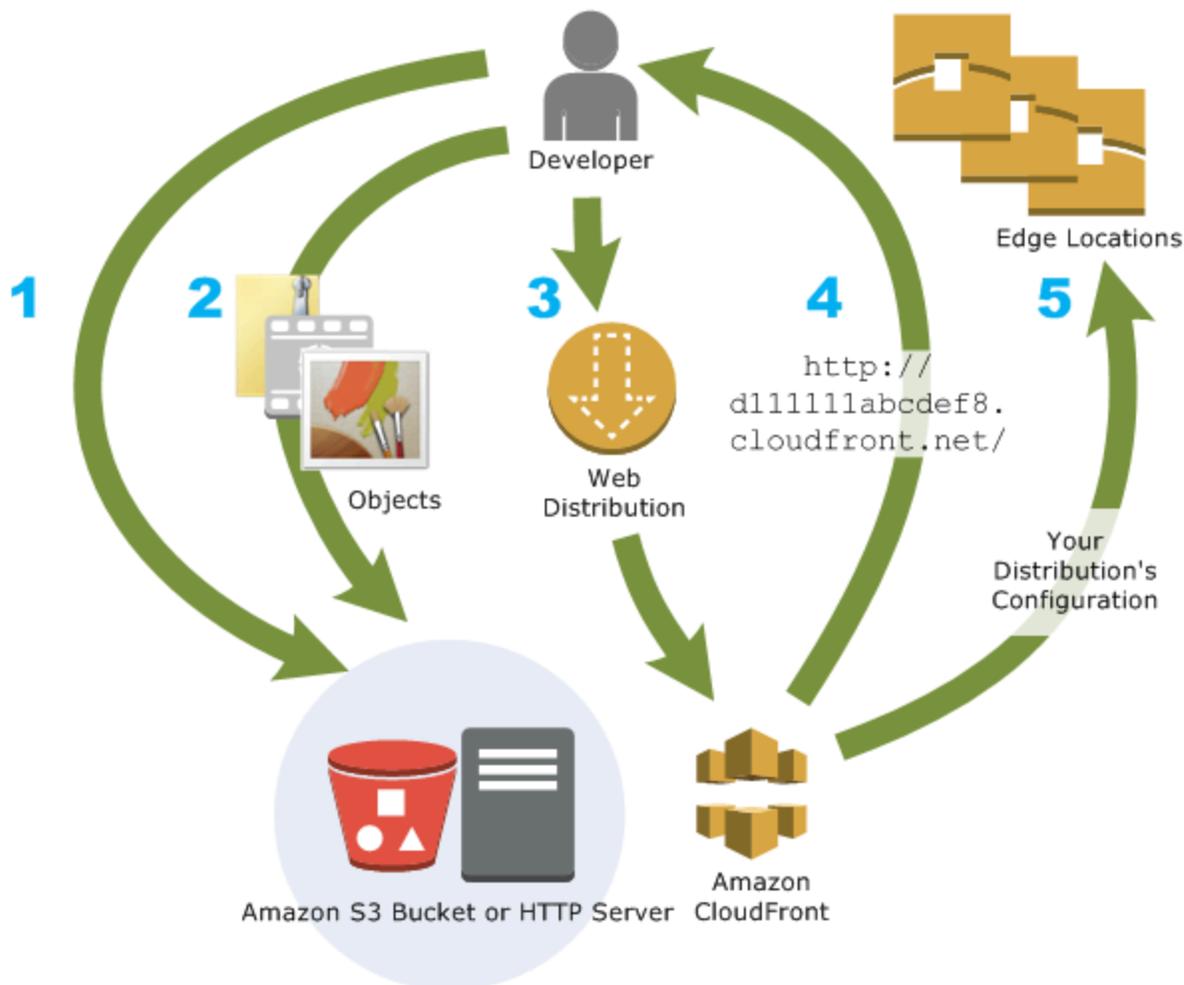
トピック

- [コンテンツを配信 CloudFront するように を設定する方法](#)
- [CloudFront ユースケース](#)

- [がコンテンツを CloudFront 配信する方法](#)
- [CloudFront エッジサーバーの場所と IP アドレス範囲](#)
- [アクセス CloudFront](#)
- [CloudFront 料金](#)
- [AWS SDK CloudFront で を使用する](#)

コンテンツを配信 CloudFront するように を設定する方法

デイス CloudFront トリビューションを作成して、コンテンツの配信 CloudFront 元と、コンテンツ配信の追跡と管理の方法に関する詳細を に伝えます。次に、ビューワーに近いコンピューター、エッジサーバー CloudFront を使用して、誰かがコンテンツを表示または使用したいときに、そのコンテンツをすばやく配信します。



コンテンツを配信 CloudFront するように を設定する方法

1. Amazon S3 バケットや独自の HTTP サーバーなどのオリジンサーバー を指定します。このサーバーは、ファイル CloudFront を取得し、そこから世界中の CloudFront エッジロケーションから配信されます。

オリジンサーバーには、お客様のオブジェクトのオリジナルの最終バージョンが保存されます。コンテンツを HTTP 経由で提供する場合、オリジンサーバーは Amazon S3 バケットまたは HTTP サーバー (ウェブサーバーなど) になります。HTTP サーバーは、Amazon Elastic Compute Cloud (Amazon EC2) インスタンス、またはお客様が管理するサーバー (カスタムオリジンとも呼ばれる) で実行できます。

2. ファイルをオリジンサーバーにアップロードします。ファイル (オブジェクトとも呼ばれます) には、通常、ウェブページ、イメージ、メディアファイルが含まれますが、HTTP 経由で提供できるものであれば何でもかまいません。

Amazon S3 バケットをオリジンサーバーとして使用している場合は、バケット内のオブジェクトをパブリックに読み取り可能にできるため、オブジェクト CloudFront URLs を知っているすべてのユーザーがオブジェクトにアクセスできます。オブジェクトを非公開にして、オブジェクトにアクセスするユーザーを制限することもできます。「[署名付き URL と署名付き Cookie を使用したプライベートコンテンツの提供](#)」を参照してください。

3. CloudFront ディストリビューション を作成します。これは、ユーザーがウェブサイトまたはアプリケーションを通じてファイルをリクエストしたときに、ファイルを取得する CloudFront オリジンサーバーを に指示します。同時に、すべてのリクエスト CloudFront を記録するかどうか、ディストリビューションの作成後すぐにディストリビューションを有効にするかどうかなどの詳細を指定します。
4. CloudFront は、新しいディストリビューションにドメイン名を割り当てます。ドメイン名は、CloudFront コンソールで確認できます。また、API リクエストなどのプログラムによるリクエストに対するレスポンスで返されます。必要に応じて、代わりに使用する代替ドメイン名を追加できます。
5. CloudFront は、ディストリビューションの設定 (コンテンツではなく) を、そのすべてのエッジロケーションまたは Point of Presence (POPs) に送信します。POP は、 がファイルのコピーを CloudFront キャッシュする地理的に分散したデータセンター内のサーバーのコレクションです。

ウェブサイトまたはアプリケーションを開発するときは、 が URLs CloudFront に提供するドメイン名を使用します。例えば、 がディストリビューションのドメイン

名 `d111111abcdef8.cloudfront.net` としてを CloudFront 返す場合、Amazon S3 バケット (または HTTP サーバーのルートディレクトリ) 内の `logo.jpg` の URL は `https://d111111abcdef8.cloudfront.net/logo.jpg` です。

または、ディストリビューションで独自のドメイン名を使用する CloudFront ようにを設定することもできます。この場合、URL は `https://www.example.com/logo.jpg` のようになります。

オプションで、ファイルにヘッダーを追加するようにオリジンサーバーを設定して、エッジ CloudFront ロケーションのキャッシュにファイルを保持する期間を指定できます。デフォルトでは、各ファイルはエッジロケーションに 24 時間保持された後に有効期限切れになります。最小の有効期限切れ時間は 0 秒です。有効期限切れ時間の上限はありません。詳細については、「[コンテンツがキャッシュに保持される期間 \(有効期限\) の管理](#)」を参照してください。

CloudFront ユースケース

を使用すると CloudFront、さまざまな目標を達成できます。このセクションでは、詳細情報へのリンクとともに、可能性についてのアイデアをいくつか示します。

トピック

- [静的ウェブサイトのコンテンツ配信の加速](#)
- [オンデマンドビデオおよびライブストリーミングビデオの配信](#)
- [システム処理全体で特定のフィールドを暗号化する](#)
- [エッジのカスタマイズ](#)
- [Lambda@Edge カスタマイズを使用したプライベートコンテンツの供給](#)

静的ウェブサイトのコンテンツ配信の加速

CloudFront は、世界中のビューワーへの静的コンテンツ (イメージ、スタイルシートなど JavaScript) の配信を高速化できます。を使用すると CloudFront、AWS バックボーンネットワークサーバーと CloudFront エッジサーバーを活用して、ビューワーがウェブサイトにアクセスする際に高速、安全、信頼性の高いエクスペリエンスを提供できます。

Amazon S3 バケットを使用すると、静的コンテンツを簡単に格納し、配信することができます。S3 をと一緒に使用すると、[オリジンアクセスコントロール](#)を使用して S3 コンテンツへのアクセスを簡単に制限するオプションなど、多くの利点 CloudFront があります。

S3 と を併用する方法の詳細については CloudFront、すぐに使用を開始するためのAWS CloudFormationテンプレートなど、[「Amazon S3 + Amazon CloudFront: クラウドでのマッチングの作成」](#)を参照してください。

オンデマンドビデオおよびライブストリーミングビデオの配信

CloudFront には、録画済みのファイルとライブイベントの両方について、世界中のビューワーにメディアをストリーミングするためのオプションがいくつか用意されています。

- ビデオオンデマンド (VOD) ストリーミングの場合、CloudFront を使用して MPEG DASH、Apple HLS、Microsoft Smooth Streaming、CMAF などの一般的な形式で任意のデバイスにストリーミングできます。
- ライブストリームをブロードキャストする場合は、フラグメントを正しい順序で配信するマニフェストファイルに対する複数のリクエストを結合して、オリジンサーバーの負荷を軽減できるように、メディアフラグメントをエッジにキャッシュすることができます。

でストリーミングコンテンツを配信する方法の詳細については CloudFront、[「」](#)を参照してください。[によるビデオオンデマンドおよびライブストリーミングビデオ CloudFront](#)。

システム処理全体で特定のフィールドを暗号化する

で HTTPS を設定すると CloudFront、オリジンサーバーへの安全な end-to-end 接続が既に確保されています。フィールドレベルの暗号化を追加する場合、HTTPS セキュリティに加えて、システムの処理中に特定のデータにオリジンの特定のアプリケーションのみがアクセスできるように、そのデータを保護できます。

フィールドレベル暗号化を設定するには、にパブリックキーを追加し CloudFront、そのキーで暗号化するフィールドのセットを指定します。詳細については、[「フィールドレベル暗号化を使用した機密データの保護」](#)を参照してください。

エッジのカスタマイズ

エッジでサーバーレスコードを実行することには、ビューワーのコンテンツとエクスペリエンスをカスタマイズして、待ち時間を短縮することができる様々な可能性があります。たとえば、メンテナンスのためにオリジンサーバーがダウンしているときにカスタムエラーメッセージを表示することができるため、ビューワーには一般的な HTTP エラーメッセージは表示されません。または、がリクエストをオリジン CloudFront に転送する前に、関数を使用してユーザーを承認し、コンテンツへのアクセスを制御することもできます。

で Lambda@Edge CloudFront を使用すると、が CloudFront 配信するコンテンツをさまざまな方法でカスタマイズできます。Lambda@Edge の詳細、および CloudFront で関数を作成してデプロイする方法の詳細については、「[Lambda@Edge を使用したエッジでのカスタマイズ](#)」を参照してください。独自のソリューションのためにカスタマイズできる多くのコードサンプルについては、[Lambda@Edge 関数の例](#) を参照してください。

Lambda@Edge カスタマイズを使用したプライベートコンテンツの供給

Lambda@Edge を使用すると、署名URLs または署名付き Cookie を使用するだけでなく、独自のカスタムオリジンからプライベートコンテンツを提供するように CloudFront デイストリビューションを設定できます。

を使用してプライベートコンテンツを供給するには CloudFront、次の操作を行います。

- [署名付き URL または署名付き Cookie](#) を使用してコンテンツにアクセスするようにユーザー (ビューワー) にリクエストします。
- オリジンへのアクセスを制限して、CloudFrontのオリジン向けサーバーからのみ使用できるようにします。これは、次のいずれかで実行できます。
 - Amazon S3 オリジンでは、[オリジンアクセスコントロール \(OAC\) を使用](#)できます。
 - カスタムオリジンでは、次のように実行できます。
 - カスタムオリジンが Amazon VPC セキュリティグループまたは によって保護されている場合AWS Firewall Manager、[CloudFront マネージドプレフィックスリスト](#)を使用して、CloudFrontのオリジン向け IP アドレスのみからオリジンへのインバウンドトラフィックを許可できます。
 - カスタム HTTP ヘッダーを使用して、からのリクエストのみにアクセスを制限します CloudFront。詳細については、「[the section called “カスタムオリジン上のファイルへのアクセス制限”](#)」および「[the section called “オリジンリクエストへのカスタムヘッダーの追加”](#)」を参照してください。カスタムヘッダーを使用して Application Load Balancer オリジンへのアクセスを制限する例については、[the section called “Application Load Balancers へのアクセスを制限する”](#) を参照してください。
 - カスタムオリジンがカスタムアクセスコントロールロジックを必要とする場合は、ブログ記事「[Amazon CloudFront & Lambda@Edge を使用したプライベートコンテンツの提供](#)」で説明されているように、[Lambda@Edge](#) を使用してそのロジックを実装できます。

がコンテンツを CloudFront 配信する方法

一部の初期設定後、CloudFront はウェブサイトまたはアプリケーションと連携して、コンテンツの配信を高速化します。このセクションでは、ビューワーがコンテンツを要求したときに がコンテンツ CloudFront を提供する方法について説明します。

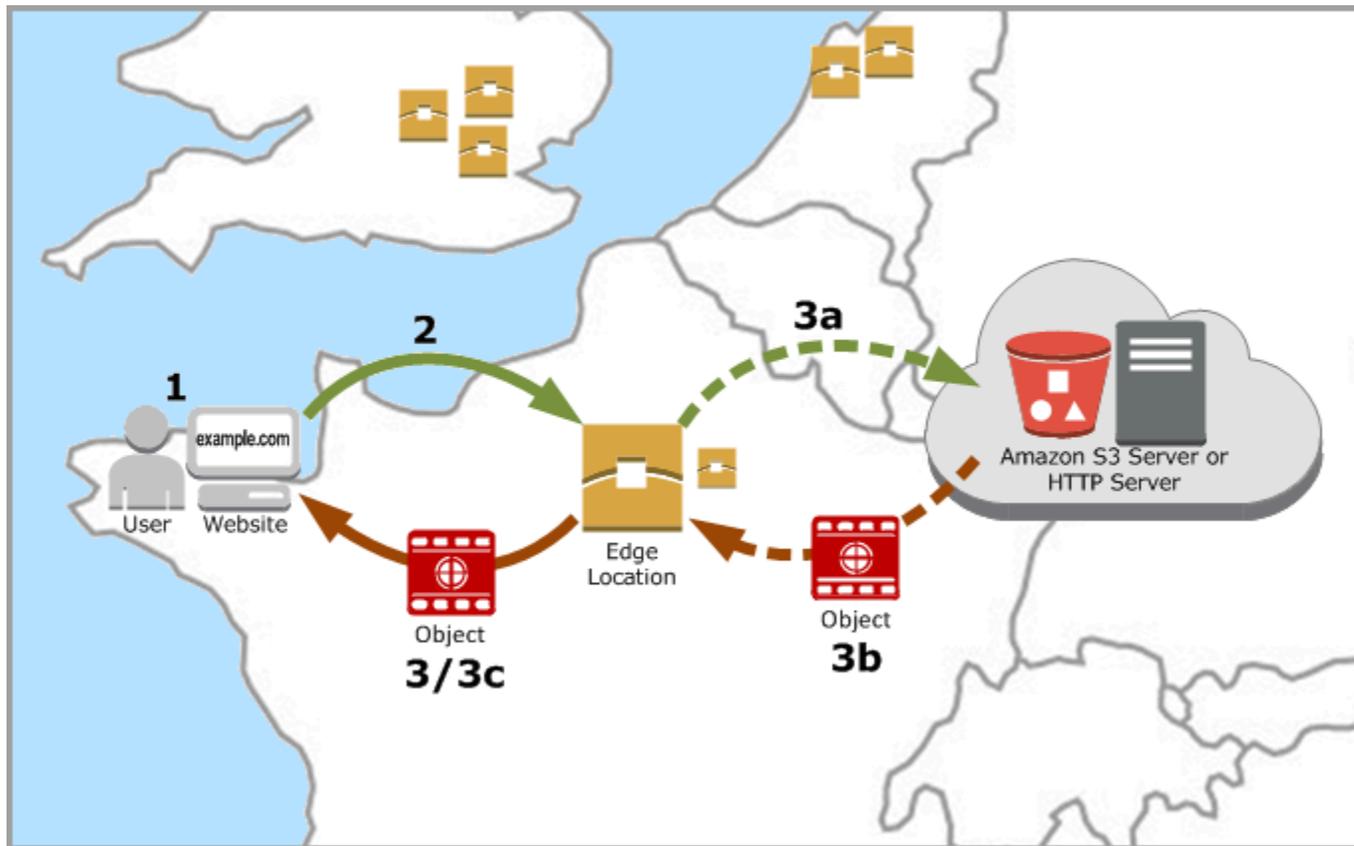
トピック

- [がユーザーにコンテンツを CloudFront 配信する方法](#)
- [がリージョン別エッジキャッシュと CloudFront 連携する方法](#)

がユーザーにコンテンツを CloudFront 配信する方法

コンテンツを配信 CloudFront するように を設定した後、ユーザーがオブジェクトをリクエストすると次のようになります。

1. ユーザーがウェブサイトまたはアプリケーションにアクセスして、イメージファイルや HTML ファイルなどのオブジェクトに対するリクエストを送信します。
2. DNS は、リクエストを最適な方法で処理できる CloudFront POP (エッジロケーション) にリクエストをルーティングします。通常はレイテンシーの点で最も近い CloudFront POP です。
3. CloudFront は、リクエストされたオブジェクトのキャッシュをチェックします。オブジェクトがキャッシュにある場合、 はそれをユーザーに CloudFront 返します。オブジェクトがキャッシュにない場合、 CloudFront は以下を実行します。
 - a. CloudFront は、リクエストをディストリビューションの仕様と比較し、Amazon S3 バケットや HTTP サーバーなど、対応するオブジェクトのオリジンサーバーにリクエストを転送します。
 - b. そのオリジンサーバーが、エッジロケーションにオブジェクトを返します。
 - c. オリジンから最初のバイトが到着するとすぐに、 はオブジェクトをユーザーに転送するために CloudFront 開始します。CloudFront また、 は次にリクエストしたときにオブジェクトをキャッシュに追加します。



がリージョン別エッジキャッシュと CloudFront 連携する方法

CloudFront POP またはエッジロケーション (とも呼ばれ POPs) により、人気のあるコンテンツをビューワーにすばやく提供できます。CloudFront また、POP にとどまるほど人気がない場合でも、より多くのコンテンツをビューワーに近づけるリージョン別エッジキャッシュがあり、そのコンテンツのパフォーマンスを向上させることができます。

リージョン別エッジキャッシュは、すべてのタイプのコンテンツ (特に、時間の経過とともに人気落ちる傾向にあるコンテンツ) に役立ちます。この例には、ビデオ、写真、アートワークのようなユーザーが生成したコンテンツ、製品の写真やビデオのような e コマースアセット、突然新たに人気が出る可能性があるニュースやイベント関連のコンテンツがあります。

リージョン別キャッシュの動作

リージョン別エッジキャッシュは、ビューワーに近いグローバルにデプロイされる CloudFront ロケーションです。オリジンサーバーと、ビューワーに直接コンテンツを提供する POP (世界各地のエッジロケーション) の間にあります。オブジェクトの人気下がると、個別の POP では、これらのオブジェクトを削除し、より人気の高いコンテンツ用に容量を確保する場合があります。リージョン別エッジキャッシュのキャッシュは個別の POP よりも大きいため、オブジェクトは最も近いリー

地域別エッジキャッシュロケーションでより長くキャッシュに残ります。これにより、より多くのコンテンツをビューワーに近づけ、オリジンサーバー CloudFront に戻る必要性を減らし、ビューワーの全体的なパフォーマンスを向上させることができます。

ビューワーがウェブサイトで、またはアプリケーション経由でリクエストを実行すると、DNS はユーザーのリクエストに対応できる最適な POP にリクエストをルーティングします。この場所は通常、レイテンシーの点で最も近い CloudFront エッジロケーションです。POP で、CloudFront はリクエストされたオブジェクトのキャッシュをチェックします。オブジェクトがキャッシュにある場合、CloudFront はそれをユーザーに返します。オブジェクトがキャッシュにない場合、POP は最も近い地域別エッジキャッシュをフェッチします。POP が地域別エッジキャッシュをスキップして直接オリジンに移動する場合の詳細については、次の注記を参照してください。

地域別エッジキャッシュロケーションで、CloudFront は要求されたオブジェクトのキャッシュをチェックします。オブジェクトがキャッシュにある CloudFront 場合は、それをリクエストした POP に転送します。地域別エッジキャッシュロケーションから最初のバイトが到着するとすぐに、はオブジェクトをユーザーに転送するために CloudFront 開始します。CloudFront また、は、次回リクエストしたときに、POP のキャッシュにオブジェクトを追加します。

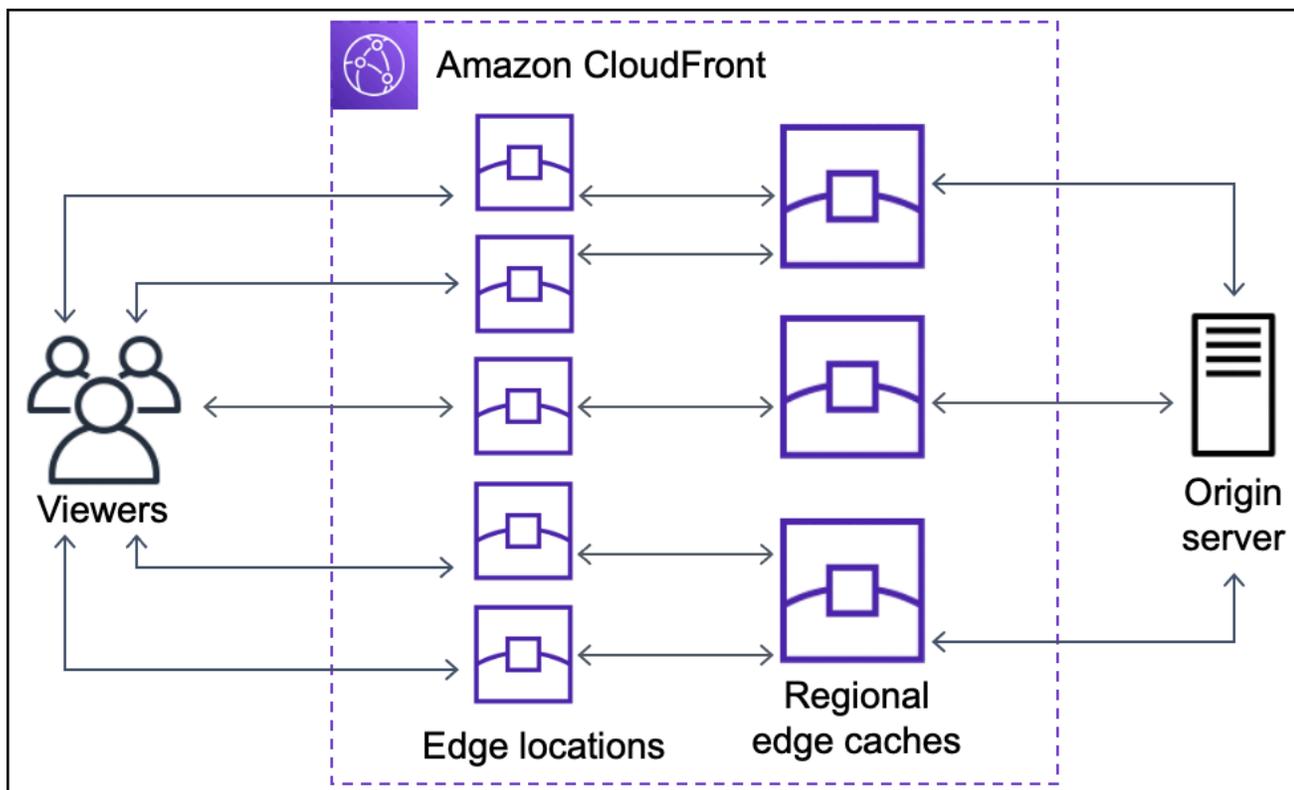
POP または地域別エッジキャッシュロケーションのいずれにもキャッシュされていないオブジェクトの場合、はディストリビューションの仕様とリクエスト CloudFront を比較し、リクエストをオリジンサーバーに転送します。オリジンサーバーがオブジェクトを地域別エッジキャッシュロケーションに送り返した後、POP に転送され、その後ユーザー CloudFront に転送されます。この場合、は、次回ビューワーがリクエストしたときに、POP に加えて地域別エッジキャッシュロケーションのキャッシュにオブジェクト CloudFront を追加します。これにより、地域内のすべての POPs がローカルキャッシュを共有され、オリジンサーバーへの複数のリクエストがなくなります。CloudFront また、はオリジンサーバーとの永続的な接続も維持されるため、オブジェクトは可能な限り迅速にオリジンから取得されます。

Note

- 地域別エッジキャッシュには、POP と同等の機能があります。たとえば、キャッシュ無効化リクエストでは、有効期限が切れる前に、POP と地域別エッジキャッシュの両方からオブジェクトが削除されます。エンドユーザーが次にオブジェクトを要求したときに、CloudFront はオリジンに戻ってオブジェクトの最新バージョンをフェッチします。
- プロキシ HTTP メソッド (PUT、POST、PATCH、OPTIONS、DELETE) は POP からオリジンに直接送信され、地域別エッジキャッシュを経由してプロキシを実行しません。

- リクエスト時に決定される動的リクエストは、リージョン別エッジキャッシュを通過せず、直接オリジンに送信されます。
- オリジンが Amazon S3 バケットで、リクエストの最適なリージョン別エッジキャッシュが S3 バケットと同じ AWS リージョンである場合、POP はリージョン別エッジキャッシュをスキップして S3 バケットに直接移動します。

次の図は、リクエストとレスポンスが CloudFront エッジロケーションとリージョン別エッジキャッシュを通過する方法を示しています。



CloudFront エッジサーバーの場所と IP アドレス範囲

CloudFront エッジサーバーの場所のリストについては、[「Amazon CloudFront Global Edge Network」](#) ページを参照してください。

Amazon Web Services (AWS) は、その現在の IP アドレス範囲を JSON 形式で公開します。現在の範囲を参照するには、[ip-ranges.json](#) ファイルをダウンロードします。詳細については、Amazon Web Services 全般のリファレンスの [AWS IP アドレスの範囲](#) をご参照ください。

CloudFront エッジサーバーに関連付けられている IP アドレス範囲を見つけるには、ip-ranges.json で次の文字列を検索します。

```
"region": "GLOBAL",  
"service": "CLOUDFRONT"
```

または、で CloudFront IP 範囲のみを表示できます <https://d7uri8nf7uskq.cloudfront.net/tools/list-cloudfront-ips>。

CloudFront マネージドプレフィックスリストを使用する

CloudFront マネージドプレフィックスリストには、CloudFrontのグローバルに分散されたオリジン向けサーバーの IP アドレス範囲がすべて含まれています。オリジンがでホストAWSされ、Amazon VPC [セキュリティグループ](#) によって保護されている場合、CloudFront マネージドプレフィックスリストを使用して、CloudFrontのオリジン向けサーバーからのみオリジンへのインバウンドトラフィックを許可できます。CloudFront トラフィック以外の がオリジンに到達しないようにします。マネージドプレフィックスリストは、のすべての CloudFront グローバルオリジン向けサーバーの IP アドレスで常に最新になるように CloudFront 維持されます。CloudFront マネージドプレフィックスリストを使用すると、IP アドレス範囲のリストを自分で読み取ったり維持したりする必要はありません。

例えば、オリジンが欧州 (ロンドン) (eu-west-2) リージョンの Amazon EC2 インスタンスであると想定します。インスタンスが VPC 内にある場合は、CloudFront マネージドプレフィックスリストからのインバウンド HTTPS アクセスを許可するセキュリティグループルールを作成できます。これにより、CloudFrontのすべてのグローバルオリジン向けサーバーがインスタンスに到達できるようになります。セキュリティグループから他のすべてのインバウンドルールを削除すると、CloudFront トラフィック以外の がインスタンスに到達できなくなります。

CloudFront マネージドプレフィックスリストの名前は com.amazonaws.global.cloudfront.origin-facing です。このプレフィックスリストは、アジアパシフィック (ジャカルタ) (ap-southeast-3) を除く、すべての AWS リージョンで使用できます。詳細については、Amazon VPC ユーザーガイドの「[AWS マネージドプレフィックスリストの使用](#)」を参照してください。

Important

CloudFront マネージドプレフィックスリストは、Amazon VPC クォータに適用される点で一意です。詳細については、Amazon VPC ユーザーガイドの「[AWS マネージドプレフィックスリストの重み](#)」を参照してください。

アクセス CloudFront

Amazon には CloudFront、次の方法でアクセスできます。

- AWS Management Console – このガイドでは、AWS Management Console を使用してタスクを実行する方法について説明しています。
- AWS SDK - AWS が SDK を提供するプログラミング言語を使用している場合は、SDK を使用して CloudFront にアクセスできます。SDKs 認証を簡素化し、開発環境と簡単に統合して、CloudFront コマンドへのアクセスを提供します。詳細については、「[Amazon ウェブ サービスのツール](#)」を参照してください。
- CloudFront API – SDK が使用できないプログラミング言語を使用している場合は、「[Amazon CloudFront API リファレンス](#)」で API アクションの詳細と API リクエストの作成方法を参照してください。
- AWS Command Line Interface – 詳細については、AWS Command Line Interface ユーザーガイドの「[AWS Command Line Interface でのセットアップ](#)」を参照してください。
- AWS Tools for Windows PowerShell - 詳細については、AWS Tools for Windows PowerShell ユーザーガイドの [AWS Tools for Windows PowerShell のセットアップ](#) を参照してください。

CloudFront 料金

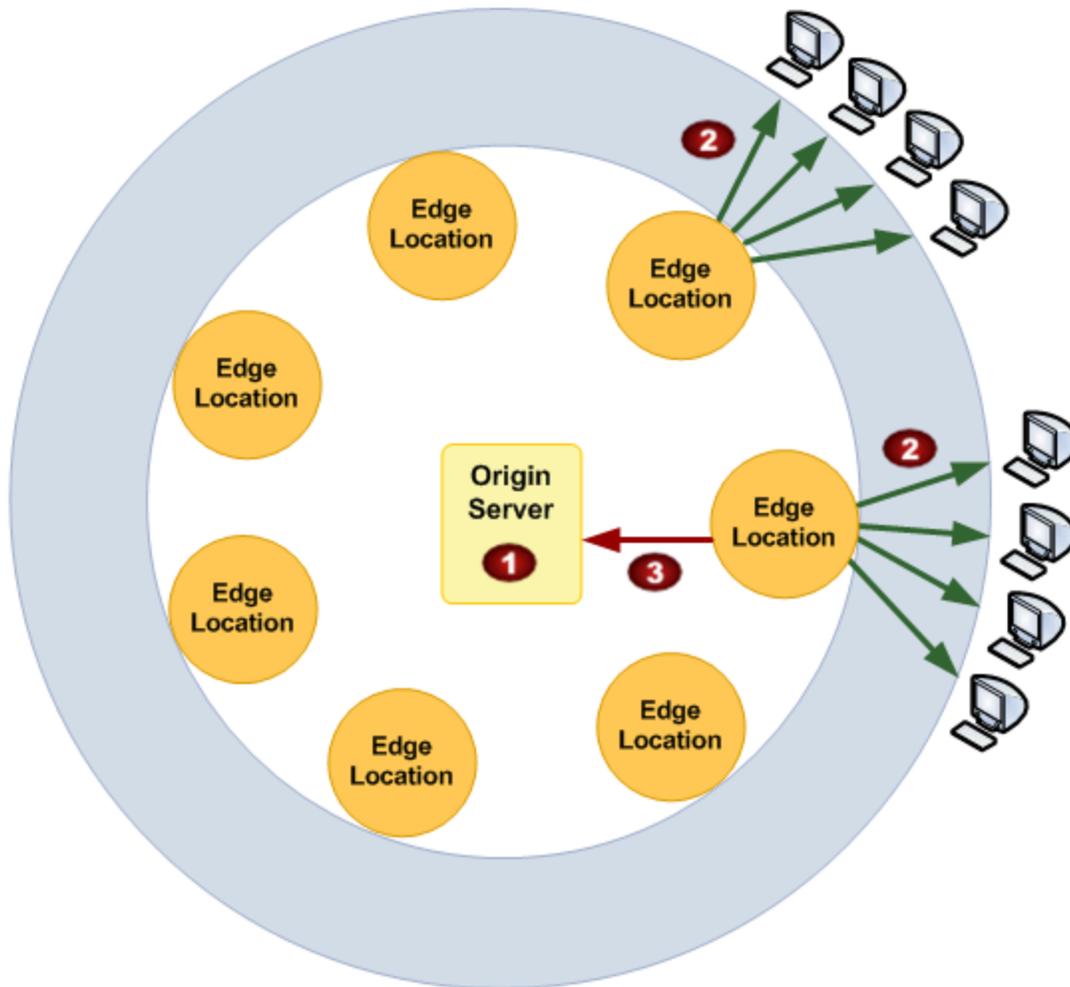
Amazon CloudFront は、前払い料金を支払う必要も、コンテンツの量にコミットする必要もありません。他の AWS サービスと同様に、使用分の料金だけを支払う従量制料金になっています。料金の詳細については、「[Amazon 料金表 CloudFront](#)」を参照してください。

Tip

CloudFront (または任意のAWSサービス)からの予想外の請求を避けるために、AWS Budgets を使用できます。AWS Budgets では、コストのしきい値を設定して、実際の料金や予測した料金がしきい値を超えた場合に E メールまたは Amazon SNS トピックで通知を受け取ることができます。詳細については、AWS Billing and Cost Management ユーザーガイドの「[AWS Budgets を使用したコストの管理](#)」と「[予算の作成](#)」を参照してください。使用を開始するには、[コンソールの AWS Budgets](#) に移動します。

AWS には CloudFront、請求レポートと使用状況を要約したレポートの 2 つの使用状況レポートが用意されています。これらのレポートの詳細については、「[AWS の 請求および使用状況レポート CloudFront](#)」を参照してください。

以下の図とリストを使用して、CloudFront の使用料の概要を説明します。



AWS からの毎月の請求書では、AWS のサービスと機能ごとに使用量と金額が振り分けられています。以下で、前の図で説明した料金を説明します。料金の詳細については、「[Amazon 料金表 CloudFront](#)」を参照してください。

1. Amazon S3 バケットのストレージに対する課金。バケットにオブジェクトを保存するための通常の Amazon S3 ストレージ料金をお支払いいただきます。この料金は、AWS ステートメントの Amazon S3 の部分に表示されます。
2. エッジロケーションからのオブジェクトの供給に対する課金。がオブジェクトのリクエストに CloudFront 応答すると、CloudFront 料金が発生します。料金には、サーバーからクライアントへ

の WebSocket データ転送が含まれます。CloudFront 料金は、AWSステートメントの CloudFront 部分に *region* -DataTransfer-Out-Bytes として表示されます。

- データの送信に対する課金。ユーザーが、、、DELETEOPTIONSPATCH、POST、および PUT リクエストを含むオリジンまたは [エッジ関数](#) にデータを転送したときに CloudFront 料金が発生します。料金には、クライアントからサーバーへの WebSocket データ転送が含まれます。料金は、AWSステートメントの CloudFront CloudFront 部分に *region* -DataTransfer-Out-OBytes として表示されます。

以下の点に注意してください。

- HTTPS リクエストには追加料金も発生します。また、フィールドレベルの暗号化が有効になっているリクエストや、[Origin Shield](#) をインクリメンタルキャッシュレイヤーとして使用する場合にも追加料金が発生します。料金の詳細については、「[Amazon 料金表 CloudFront](#)」を参照してください。
- オリジングループを使用する場合、CloudFront 追加料金は発生しません。他のオリジン AWS または AWS 以外のオリジン CloudFront でを使用する場合と同じリクエスト料金とデータ転送レートを引き続き支払います。詳細については、「[CloudFront オリジングループの使用](#)」を参照してください。

CloudFront Security Savings Bundle

CloudFront Security Savings Bundle は、前払いコミットメントを行うときに AWS 請求書の CloudFront 料金を最大 30% 節約できる簡単な方法です。Savings Bundle を購入すると、一般的な [AWS WAF](#) ウェブエクспロイトから CloudFront ディストリビューションを保護するウェブアプリケーションファイアウォールである のクレジットも取得できます。

詳細については、次のセクションを参照してください。Savings Bundle を購入するには、[CloudFront コンソールの Savings Bundle の概要ページ](#) に移動します。

セクション

- [Savings Bundle の概要](#)
- [Savings Bundle の例](#)
- [Savings Bundle の購入](#)
- [Savings Bundle の表示と更新](#)
- [Savings Bundle を管理するアクセス許可](#)

- [Savings Bundle の詳細情報](#)

Savings Bundle の概要

CloudFront Security Savings Bundle の仕組みは次のとおりです。

1. Savings Bundle を購入するには、CloudFront 1 年間にわたって一貫した月額 (月額) を支払うことをコミットします。Savings Bundle を購入した請求期間から、12 か月間、毎月約束した金額が請求されます。請求期間の最終日に Savings Bundle を購入した場合、料金とクレジットは次の請求期間から始まります。
2. お客様のコミットメントと引き換えに、は 1 年間の 12 請求期間ごとにAWS自動的にクレジットを請求書 CloudFront に適用します。これらのクレジットの値は、コミットされた支払い金額を超えるため、コミットされた金額に対する CloudFrontの標準料金が最大 30% 割引されます。これらのクレジットは、AWS 請求書の CloudFront 請求額を自動的に補います。詳細な例については、次のセクションを参照してください。
3. CloudFront クレジットに加えて、の使用に対するリクエストごとの料金を補うクレジットも得られますAWS WAF。AWS WAF クレジットの金額は、毎月の CloudFront コミットメント金額の 10% までです。AWS WAF でを使用する方法の詳細については CloudFront、「」を参照してください[AWS WAF 保護の使用](#)。

Savings Bundle の例

CloudFront 使用料が通常 1 か月あたり 600 USD になるシナリオを考えてみましょう。CloudFront Security Savings Bundle を最大限に活用するには、1 年間、CloudFront毎月 420 USD のお支払いをコミットします。これは、通常の使用料金より 30% 少なく (600 USD x 0.7) なります。このコミットメントと引き換えに、は、次の 12 回の請求期間ごとに CloudFront 毎月のAWS請求額に適用される 600 USD 分のクレジット CloudFront を提供します。これらのクレジットはの CloudFront 料金に自動的に適用され、引き続き標準料金で請求に表示されます。実際には、1 か月あたり 600 USD の使用に対して 1 か月あたり 420 USD のコストが発生します CloudFront 。

さらに、デイス CloudFront トリビューションでを使用するコストを補うためにAWS WAF、42 USD のAWS WAFクレジットも獲得できます。

CloudFront Security Savings Bundle を 420 USD の月額コミットメントで購入すると、推定される年間削減額は最大 2,664 USD になります。

Savings Bundle の購入

Savings Bundle を購入するには、[CloudFront コンソールの Savings Bundle の概要ページ](#)に移動し、開始方法を選択します。

最初のステップでは、過去数か月間の履歴に基づいて、コンソール CloudFront に推奨される月額コミットメント額が表示されます。計算ツールタブを選択すると、使用量計算ツールを使用して推定 CloudFront 使用量を入力し、推定に基づいて月額コミットメント額の推奨を受け取ることができます。

推奨されるコミットメントを表示し、[Next] を選択すると、毎月のコミットメント額と、毎年 Savings Plan を自動的に更新するかどうかを選択できます。毎月のコミットメント額に基づいて、特典の概要を確認できます。

Purchase commitment

Term	Start month	Monthly commitment payment
<input checked="" type="radio"/> 1-year	February 2021 (this month)	Enter monthly commitment amount (in US dollars)
		<input type="text" value="420"/>
Payment option	Auto renew	
<input checked="" type="radio"/> Monthly	<input checked="" type="checkbox"/> Automatically renew saving bundle	

Purchase summary

Monthly payment	Total cost over term
\$420.00	\$5,040.00

Benefits summary

Monthly CloudFront charges covered	Monthly WAF charges covered	Total estimated savings over term
\$600.00	\$42.00	up to \$2,664.00

次へを選択して、CloudFront Security Savings Bundle を確認して購入します。

Savings Bundle の表示と更新

購入済みの Savings Bundle を表示または更新するには、[CloudFront コンソールの Savings Bundle インベントリページ](#)に移動します。このページには、購入した Savings Bundle が表示され、購入した Savings Bundle の自動更新を有効または無効にできます。

複数の Savings Bundle を購入でき、同時に複数の Savings Bundle を有効にすることができます。Savings Bundle を購入し、毎月の CloudFront 使用量がバンドルのクレジットを一貫して超えている場合は、別のバンドルを購入して追加の節約できます。

Savings Bundle を管理するアクセス許可

CloudFront Security Savings Bundle を管理するには、IAM アイデンティティに必要なアクセス許可が必要です。CloudFront (`cloudfront:*`) へのフルアクセス権を持つ ID は、これらのアクセス許可を自動的に継承します。他の ID の場合、以下のアクセス許可を手動で追加できます。

- 次の読み取り専用アクセス許可により、ID は、コンソールで CloudFront レコメンデーションや推定削減額を表示するために必要な情報など、既存の CloudFront Security Savings Bundle に関連する情報を取得できます。
 - `cloudfront:ListSavingsPlans`
 - `cloudfront:GetSavingsPlan`
 - `cloudfront:ListRateCards`
 - `cloudfront:ListUsages`
- `cloudfront:CreateSavingsPlan` — ID が CloudFront Security Savings Bundle を購入できるようにします。
- `cloudfront:UpdateSavingsPlan` — 購入した CloudFront Security Savings Bundle の自動更新を有効または無効にすることを ID に許可します。

Savings Bundle の詳細情報

以下の質問と回答を参考にして、CloudFront Security Savings Bundle に関する追加の詳細を理解してください。

Savings Bundle のクレジットは、特定のディストリビューションに適用されますか、またはすべてのディストリビューションに適用されますか？

クレジットは、AWS アカウント内のすべての CloudFront 使用量に AWS アカウントレベルで適用されます。

クレジットはすべてのタイプの CloudFront 使用量に適用されますか？

はい。クレジットは、データ転送 CloudFront 料金、リクエスト料金、Lambda@Edge 料金を含むすべての料金に適用されます。

一括請求で CloudFront Security Savings Bundle を使用できますか？

はい。クレジット共有が有効になっている限り使用できます (これは、AWS Billing and Cost Management コンソールの[請求設定ページ](#)を表示することで確認できます)。支払いアカウント (管理アカウント) で Savings Bundle を購入します。クレジットは、最初に支払いアカウントで発生した CloudFront 料金に適用され、次にメンバーアカウントで発生した CloudFront 料金に適用されます。これは、アカウントが組織に参加または組織を離れる時期によって異なります。単一、および複数のアカウント間での AWS クレジットの適用方法に関する詳細は、AWS Billing ユーザーガイドの「[AWS クレジット](#)」を参照してください。

特定の請求期間にすべてのクレジットを使用しない場合はどうなりますか？

クレジットは、請求期間ごとに AWS 請求書に適用され、その請求期間内で使用する必要があります。請求期間の終了時に未使用のクレジットがある場合は、有効期限が切れます。クレジットは、次の請求期間には持ち越されません。

自分の CloudFront または AWS WAF の使用量がクレジットの量を超えた場合はどうなりますか？

発生する CloudFront および AWS WAF 料金は、CloudFront Security Savings Bundle のクレジットによって相殺されます。使用量がその請求期間で利用可能なクレジットを超えた場合は、差額が標準料金で請求されます。

請求額またはクレジットは、一部の月について日割り計算されますか？

いいえ。CloudFront Security Savings Plan を購入すると、現在の請求期間の請求額に料金が適用されます。同様に、クレジットは、現在の請求期間の料金に適用されます。請求期間の最終日に Savings Bundle を購入した場合、料金とクレジットは次の請求期間 (翌日) から始まります。

Savings Bundle の有効期限が切れたらどうなりますか？

1 年の期間の終了時にバンドルを自動的に更新するかどうかを選択できます。自動更新しないことを選択した場合、Savings Bundle の有効期限は 1 年後に失効します。その場合、クレジットは AWS 請求に適用されなくなり、CloudFront および AWS WAF の使用に対して標準料金が請求されます。

CloudFront 使用量が Savings Bundle のクレジットでカバーされる金額を超えた場合、通知を受け取ることはできますか？

はい。AWS Budgets では、コストまたは使用量のしきい値を設定できます。の実際の料金または予測料金がしきい値 CloudFront を超えると、E メールまたは Amazon SNS トピックで通知

を受け取ります。用にフィルタリングされたカスタム予算を作成し CloudFront、予算のしきい値を CloudFront Security Savings Bundle の対象となる使用量に設定できます。詳細については、AWS Billing and Cost Management ユーザーガイドの「[AWS Budgets を使用したコストの管理](#)」と「[予算の作成](#)」を参照してください。使用を開始するには、[コンソールの AWS Budgets](#) に移動します。

CloudFront Security Savings Bundle は請求にどのように表示されますか？

コミットメント金額の料金は、毎月の請求の CloudFront 「セキュリティバンドル」セクションに表示されます。クレジットは、CloudFront Security Savings Bundle の対象となる使用量の説明とともに、請求書の CloudFront および AWS WAF セクションに表示されます。

詳細については、AWS ウェブサイトの「[Amazon CloudFront の料金](#)」を参照してください。

CloudFront デイストリビューションの価格クラスの選択

CloudFront には、[世界中のエッジロケーションがあります](#)。各エッジロケーションのコストは異なるため、請求する料金は、どのエッジロケーションからリクエストを処理するかに応じて異なります。

CloudFront エッジロケーションは地理的リージョンにグループ化され、次の表に示すように、リージョンは料金クラスにグループ化されています。価格クラスは、CloudFront デイストリビューションを[作成](#)または[更新](#)するときに選択します。

	北米 (米国、 メキシ コ、カ ナダ)	ヨー ロッパ および イスラ エル	南アフ リカ、 ケニア 、中東	南米	日本	オース トラ リア および ニュー ジーラ ンド	香港、 インド ネシア 、フィ リピン 、シン ガポー ル、韓 国、台 湾、タ イ	インド
料金ク ラス All	はい	はい	はい	はい	はい	はい	はい	はい
料金ク ラス 200	はい	はい	はい	いいえ	はい	いいえ	はい	はい
料金ク ラス 100	はい	はい	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ

デフォルトでは、はパフォーマンスのみに基づいてリクエスト CloudFront に対応します。オブジェクトは、ビューワのレイテンシーが最も低いエッジロケーションから提供されます。コスト削減を優先し、一部の地理的リージョンにおけるビューワの潜在的なレイテンシーの延長を容認できるのであれば、地理的リージョンが限定された料金クラスを選択することもできます。一部のビューワ、特に料金クラスに含まれていない地理的リージョンのビューワでは、コンテンツがすべての CloudFront エッジロケーションから供給された場合よりもレイテンシーが高くなる場合があります。例えば、料金クラス 100 を選択した場合、インドのビューワのレイテンシーは料金クラス 200 を選択した場合よりも長くなる場合があります。

すべてのエッジロケーションを含まない価格クラスを選択した場合 CloudFront でも、価格クラスに含まれていないリージョンのエッジロケーションからリクエストを処理することがあります。この場合、コストの高いリージョンの料金は請求されません。代わりに、料金クラスのリージョンの中で最もコストの低いリージョンの料金が請求されます。

CloudFront 料金クラスと料金クラスの詳細については、「[Amazon CloudFront 料金表](#)」を参照してください。

AWS SDK CloudFront で を使用する

AWS ソフトウェア開発キット (SDK) は、多くの一般的なプログラミング言語で使用できます。各 SDK には、デベロッパーが好みの言語でアプリケーションを簡単に構築できるようにする API、コード例、およびドキュメントが提供されています。

SDK ドキュメント	コードの例
AWS SDK for C++	AWS SDK for C++ コードの例
AWS SDK for Go	AWS SDK for Go コードの例
AWS SDK for Java	AWS SDK for Java コードの例
AWS SDK for JavaScript	AWS SDK for JavaScript コードの例
AWS SDK for Kotlin	AWS SDK for Kotlin コードの例
AWS SDK for .NET	AWS SDK for .NET コードの例
AWS SDK for PHP	AWS SDK for PHP コードの例
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) コードの例
AWS SDK for Ruby	AWS SDK for Ruby コードの例
AWS SDK for Rust	AWS SDK for Rust コードの例
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP コードの例
AWS SDK for Swift	AWS SDK for Swift コードの例

可用性の例

必要なものが見つからなかった場合。このページの下側にある [Provide feedback (フィードバックを送信)] リンクから、コードの例をリクエストしてください。

Amazon の開始方法 CloudFront

シンプルなディストリビューションまたは安全な静的ウェブサイト CloudFront を作成して、コンテンツを配信するための基本的な手順を開始します。

トピック

- [設定](#)
- [シンプルな CloudFront ディストリビューションの開始方法](#)
- [安全な静的ウェブサイトの使用開始](#)

設定

このトピックでは、Amazon を使用する準備をするための AWS アカウントの作成などの準備手順について説明します CloudFront。

トピック

- [AWS アカウントへのサインアップ](#)
- [管理ユーザーの作成](#)
- [AWS Command Line Interface または AWS Tools for Windows PowerShell の設定](#)
- [AWS SDK のダウンロード](#)

AWS アカウントへのサインアップ

AWS アカウントがない場合は、以下のステップを実行して作成します。

AWS アカウント にサインアップするには

1. <https://portal.aws.amazon.com/billing/signup> を開きます。
2. オンラインの手順に従います。

サインアップ手順の一環として、通話呼び出しを受け取り、電話のキーパッドを使用して検証コードを入力するように求められます。

AWS アカウントにサインアップすると、AWS アカウントのルートユーザーが作成されます。ルートユーザーには、アカウントのすべての AWS のサービスとリソースへのアクセス権があ

ります。セキュリティのベストプラクティスとして、[管理ユーザーに管理アクセスを割り当て、ルートユーザーアクセスが必要なタスク](#)を実行する場合にのみ、ルートユーザーを使用してください。

サインアップ処理が完了すると、AWS からユーザーに確認メールが送信されます。<https://aws.amazon.com/> の アカウント をクリックして、いつでもアカウントの現在のアクティビティを表示し、アカウントを管理することができます。

管理ユーザーの作成

AWS アカウント にサインアップしたら、AWS アカウントのルートユーザー をセキュリティで保護し、AWS IAM Identity Center を有効にして管理ユーザーを作成します。日常的なタスクには、管理ユーザーを使用し、ルートユーザーを使用しないようにします。

AWS アカウントのルートユーザーをセキュリティで保護する

1. ルートユーザー を選択し、AWS アカウント のメールアドレスを入力して、アカウント所有者として [AWS Management Console](#) にサインインします。次のページでパスワードを入力します。

ルートユーザーを使用してサインインする方法については、「AWS サインイン User Guide」の「[Signing in as the root user](#)」を参照してください。

2. ルートユーザーの多要素認証 (MFA) を有効にします。

手順については、「IAM ユーザーガイド」の「[AWS アカウントのルートユーザーの仮想 MFA デバイスを有効にする \(コンソール\)](#)」を参照してください。

管理ユーザーを作成する

1. IAM アイデンティティセンターを有効にします。

手順については、「AWS IAM Identity Center ユーザーガイド」の「[AWS IAM Identity Center の有効化](#)」を参照してください。

2. IAM アイデンティティセンターで、管理ユーザーに管理アクセス権を付与します。

IAM アイデンティティセンターディレクトリ をアイデンティティソースとして使用するチュートリアルについては、「AWS IAM Identity Center ユーザーガイド」の「[IAM アイデンティティセンターディレクトリ デフォルトでのユーザーアクセスの設定](#)」を参照してください。

管理ユーザーとしてサインインする

- IAM アイデンティティセンターのユーザーとしてサインインするには、IAM アイデンティティセンターのユーザーの作成時に E メールアドレスに送信されたサインイン URL を使用します。
IAM アイデンティティセンターのユーザーを使用してサインインする方法については、「AWS サインイン ユーザーガイド」の「[AWS アクセスポータルにサインイン](#)」を参照してください。

AWS Command Line Interface または AWS Tools for Windows PowerShell の設定

AWS Command Line Interface(AWS CLI) は、AWSのサービスを管理するための統合ツールです。AWS CLI をインストールして設定する方法については、AWS Command Line Interface ユーザーガイドの「[AWS Command Line Interface でのセットアップ](#)」を参照してください。

Windows の使用経験がある場合は PowerShell、 の使用をお勧めしますAWS Tools for Windows PowerShell。詳細については、AWS Tools for Windows PowerShell ユーザーガイドの「[AWS Tools for Windows PowerShell のセットアップ](#)」を参照してください。

AWS SDK のダウンロード

が SDK AWSを提供するプログラミング言語を使用している場合は、Amazon CloudFront API の代わりに SDK を使用することをお勧めします。SDKs、認証が簡素化され、開発環境との統合が容易になり、CloudFront コマンドに簡単にアクセスできます。詳細については、「[AWS での構築ツール](#)」を参照してください。

シンプルな CloudFront デイストリビューションの開始方法

このセクションの手順では、CloudFront を使用して、以下を実行する基本設定をセットアップする方法を示します。

- 誰もが読み取りアクセスできるバケットを作成する
- オブジェクトの元のバージョンを Amazon Simple Storage Service (Amazon S3) バケットに保存する
- オブジェクトの URLs に CloudFront ドメイン名を使用します (例: `https://d1111111abcdef8.cloudfront.net/index.html`)。
- オブジェクトを CloudFront 24 時間のデフォルト期間 (最小期間は 0 秒) エッジロケーションに保持します。

大半のオプションはカスタマイズ可能です。CloudFront デイストリビューションオプションをカスタマイズする方法については、「」を参照してください[デイストリビューションを作成するためのステップ \(概要\)](#)。

トピック

- [前提条件](#)
- [ステップ 1: アクセス可能なバケットを作成する](#)
- [ステップ 2: コンテンツをバケットにアップロードする](#)
- [ステップ 3: CloudFront デイストリビューションを作成する](#)
- [ステップ 4: を通じてコンテンツにアクセスする CloudFront](#)
- [ステップ 5: クリーンアップ](#)
- [ヒント](#)

前提条件

この作業を開始する前に、必ず「[設定](#)」のステップを完了してください。

ステップ 1: アクセス可能なバケットを作成する

Amazon S3 バケットは、ファイル (オブジェクト) やフォルダのためのコンテナです。CloudFront は、Amazon S3 バケットがソースである場合、ほぼすべてのタイプのファイルを配信できます。例えば、テキスト、イメージ、動画を配信 CloudFront できます。Amazon S3 に保存できるデータ量に上限はありません。

バケットを作成するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (<https://console.aws.amazon.com/s3/>) を開きます。
2. この「使用開始」では、hello world のサンプルを使用することをお勧めします。「hello world」ウェブページをダウンロードします: [hello-world-html.zip](#)。次に、解凍します (3 つのファイルが含まれています)。3 つのファイルを、ブラウザを実行しているデスクトップなどの使いやすい場所に配置します。
3. [バケットを作成] を選択します。
4. [バケット名] に入力します。この「使用開始」では、**my-sample-bucket** という名前を使用します。この名前は、「Amazon Simple Storage Service ユーザーガイド」の「[DNS 名の要件](#)」に準拠しています。

5. [リージョン] で、地理的に近い AWS リージョンを選択し、レイテンシーとコストを削減します。
6. [オブジェクト所有者] セクションを以下のように設定します。
 - [ACL が有効になっています] を選択します。
 - [オブジェクト所有者] で、[希望するバケット所有者] を選択します。
 - [すべてのパブリックアクセスをブロック] を選択解除します。
 - 警告セクションで、[現在の設定により、このバケットとバケット内のオブジェクトがパブリックになる可能性があることを了承します] チェックボックスをオンにします。

このページの ACL とパブリックアクセスに関するすべての情報に目を通すことをお勧めします。本番環境に移行したときに、ディストリビューションに適用するセキュリティについて、予備知識を得ることができます。

7. 他のすべての設定はデフォルトのままにして、[バケットを作成] を選択します。

ステップ 2: コンテンツをバケットにアップロードする

コンテンツを Amazon S3 にアップロードするには

1. [バケット] セクションで、新しいバケットを選択します。バケットのページが表示されます。[Upload (アップロード)] を選択します。
2. [アップロード] ページで、3 つの hello world ファイルをドロップ領域にドラッグします。
3. 他のすべての設定はデフォルトのままにして、[アップロード] を選択します。
4. アップロードが完了したら、Amazon S3 の URL をウェブブラウザに入力し、コンテンツが公開されていることを確認できます。この URL は自分用のショートカットであることに注意してください。顧客がこの URL を使用してコンテンツにアクセスすることはありません。次の 2 つのステップで説明するように、顧客はディストリビューションの URL を使用します。

構文は次のとおりです。

```
https://<bucket name>.s3-<AWS Region>.amazonaws.com/<object name>
```

例:

```
https://my-sample-bucket.s3-us-west-2.amazonaws.com/index.html
```

米国東部 (バージニア北部) リージョン (us-east-1) でバケットを作成している場合は、URL から **<AWS Region>** の部分を削除します。例:

```
https://my-sample-bucket.s3.amazonaws.com/index.html
```

ステップ 3: CloudFront デイストリビューションを作成する

CloudFront デイストリビューションを作成するには

1. で CloudFront コンソールを開きます <https://console.aws.amazon.com/cloudfront/v4/home>。
2. [デイストリビューションの作成] を選択します。次の表に示すように設定します。

セクション	設定	説明						
オリジン	オリジンドメイン	作成した Amazon S3 バケットを選択します。						
	その他の設定	[オリジン]のその他の設定では、デフォルト値をその						

セクション	設定	説明						
		そのまま使用します。						
デフォルトのキャッシュヘイビア	すべての設定	デフォルト値をそのまま使用します。 キャッシュヘイビアのオプションの詳細については、 「キャッシュ動作の設定」 を参照してください。						

セクション	設定	説明						
ウェブアプリケーションファイアウォール (WAF)		組織のセキュリティポリシーに合ったオプションを選択します。						
その他のセクション	すべての設定	デフォルト値をそのまま使用します。 これらのパラメータの詳細については、 「ディストリビューションの設定」 を参照してください。						

3. [ディストリビューションの作成] を選択します。新しいディストリビューションの [詳細] セクションを確認します。数分後、[最終変更日] フィールドが [デプロイ中] から日時に変わります。
4. がディストリビューション CloudFront に割り当てるドメイン名を記録します。この表示は以下のようになります: `d111111abcdef8.cloudfront.net`

ステップ 4: を通じてコンテンツにアクセスする CloudFront

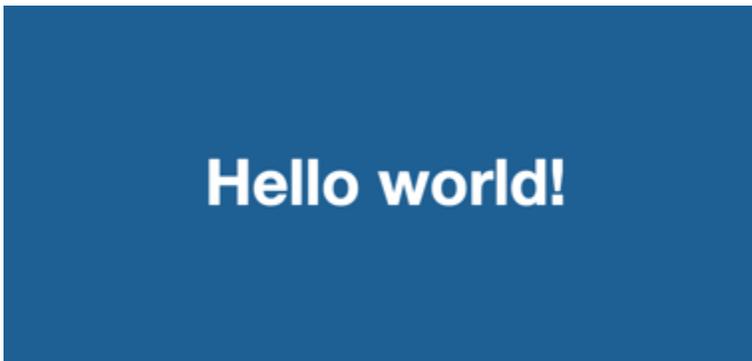
を介してコンテンツにアクセスするには CloudFront、ディストリビューションのドメイン名とコンテンツのメインページを組み合わせます。

- ディストリビューションのドメイン名は、`d111111abcdef8.cloudfront.net` のようになります。
- 通常、ウェブサイトのメインページへのパスは `/index.html` です。

したがって、を介して CloudFront コンテンツにアクセスするための URL は次のようになります。

```
https://d111111abcdef8.cloudfront.net/index.html
```

前の手順に従い、単純な Hello world ウェブページを使用している場合は、次のようにコンテンツが表示されます。



この S3 バケットにさらにコンテンツをアップロードする場合、CloudFront ディストリビューションドメイン名と S3 バケット内のオブジェクトへのパスを組み合わせる CloudFront ことで、を通じてコンテンツにアクセスできます。例えば、`new-page.html` という名前の新しいファイルを S3 バケットのルートにアップロードした場合、URL は次のようになります。

```
https://d111111abcdef8.cloudfront.net/new-page.html
```

ステップ 5 : クリーンアップ

Warning

この「使用開始」で作成したバケットを本番環境で使用しないでください。

この「使用開始」では、ACL を有効にしてバケットを作成しました。これは、すべてのユーザーにバケットへの読み取りアクセス権をすばやく付与できるようにするためです。この目的で ACL を使用することはお勧めしません。本番環境でアクセス権を設定するための現在の推奨事項については、「[コンテンツへのセキュアなアクセスとアクセス制限の設定](#)」を参照してください。

ここで Amazon S3 バケットとそのオブジェクトを削除することを強くお勧めします。

ヒント

この「使用開始」では、ディストリビューションを作成するための基本事項について説明しています。以下の拡張機能を試してみることをお勧めします。

- この「使用開始」では、最寄りのリージョンを選択することをお勧めします。ただし、規制要件に対応する場合などは、別のリージョンを選択することもできます。
- デフォルトでは、Amazon S3 バケット内のファイル (オブジェクト) はプライベートとして設定されます。バケットを作成した AWS アカウントだけが、バケット内のファイルの読み取りまたは書き込みのアクセス許可を持ちます。CloudFront URLs を使用して Amazon S3 バケット内のファイルへのアクセスを誰かに許可する場合は、オブジェクトにパブリック読み取りアクセス許可を付与する必要があります。
- CloudFront プライベートコンテンツ機能を使用して、Amazon S3 バケット内のコンテンツへのアクセスを制限できます。プライベートコンテンツの配信の詳細については、「[署名付き URL と署名付き Cookie を使用したプライベートコンテンツの提供](#)」を参照してください。
- カスタムドメイン名を使用するようにディストリビューションを設定できます (例: `www.example.com` の代わりに `d111111abcdef8.cloudfront.net`)。詳細については、「[カスタム URL の使用](#)」を参照してください。

安全な静的ウェブサイトの使用開始

Amazon の使用を開始するには、このトピックで説明するソリューション CloudFront を使用して、ドメイン名用の安全な静的ウェブサイトを作成します。静的ウェブサイトは、HTML、CSS、画像 JavaScript、動画などの静的ファイルのみを使用し、サーバーやサーバー側の処理は必要ありません。このソリューションを使用すると、ウェブサイトには次の利点があります。

- [Amazon Simple Storage Service \(Amazon S3\)](#) の耐久性のあるストレージを使用 - このソリューションでは、静的ウェブサイトのコンテンツをホストする Amazon S3 バケットを作成します。ウェブサイトを更新するには、新しいファイルを S3 バケットにアップロードするだけです。
- Amazon CloudFront コンテンツ配信ネットワークによって高速化される — このソリューションは、低レイテンシーでビューワーにウェブサイトを提供するディストリビューション CloudFront ションを作成します。ディストリビューションには、S3 から直接ではなく CloudFront、を通じてのみウェブサイトにアクセスできるように、[オリジンアクセスアイデンティティ](#)が設定されています。
- HTTPS および追加のセキュリティヘッダーによって保護される — このソリューションは、[AWS Certificate Manager \(ACM\)](#) に SSL/TLS 証明書を作成し、ディストリビューションにアタッチします。この証明書により、ディストリビューションが HTTPS を使用してドメインの Web サイトに安全にサービスを提供できるようになります。

このソリューションでは、[Lambda @Edge](#) を使用して、すべてのサーバーレスポンスにセキュリティヘッダーを追加します。セキュリティヘッダーは、ウェブサーバーレスポンス内のヘッダーのグループであり、追加のセキュリティ対策を講じるようにウェブブラウザに指示します。詳細については、ブログ記事「[Adding HTTP Security Headers Using Lambda@Edge and Amazon CloudFront](#)」を参照してください。

- [AWS CloudFormation](#) を使用した設定とデプロイ - このソリューションは AWS CloudFormation テンプレートを使用してすべてのコンポーネントをセットアップするため、コンポーネントの設定よりもウェブサイトのコンテンツに集中できます。

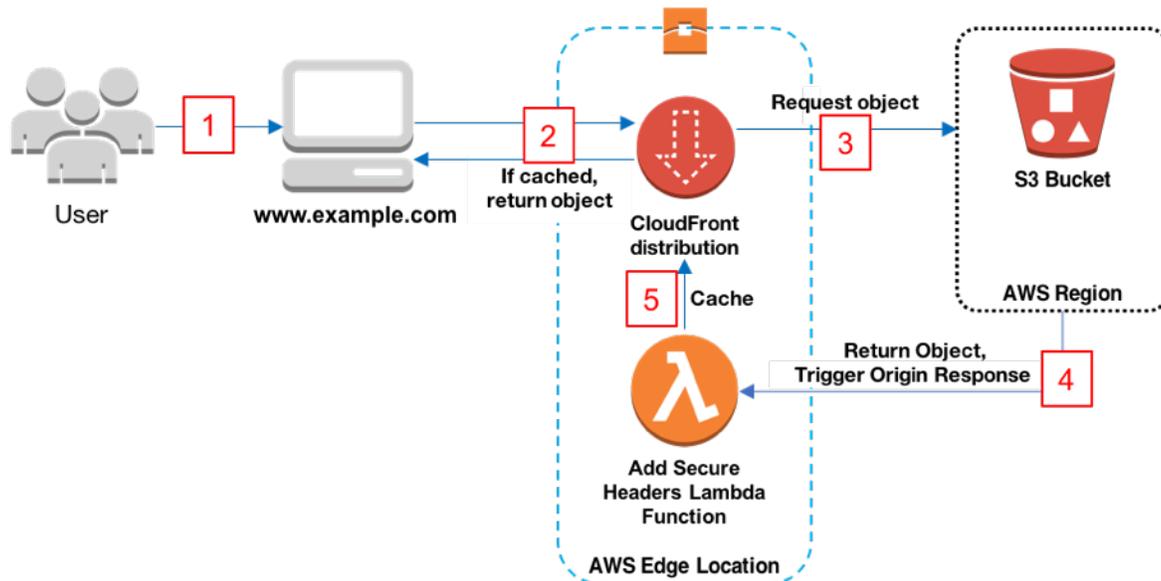
このソリューションは のオープンソースです GitHub。コードを表示したり、プルリクエストを送信したり、問題を提起したりするには、「」に進みます <https://github.com/aws-samples/amazon-cloudfront-secure-static-site>

トピック

- [ソリューションの概要](#)
- [ソリューションのデプロイ](#)

ソリューションの概要

次の図は、この静的ウェブサイトソリューションの動作の概要を示しています。



1. ビューワーが `www.example.com` のウェブサイトをリクエストします。
2. リクエストされたオブジェクトがキャッシュされている場合、 はオブジェクトをキャッシュからビューワーに CloudFront 返します。
3. オブジェクトが CloudFrontのキャッシュにない場合、 はオリジン (S3 バケット) からオブジェクトを CloudFront リクエストします。
4. S3 は、 オブジェクトを に返します。これにより CloudFront、Lambda@Edge [オリジンレスポンスイベント](#) がトリガーされます。
5. Lambda@Edge 関数によって追加されたセキュリティヘッダーを含む オブジェクトが、 CloudFrontのキャッシュに追加されます。
6. (非表示) オブジェクトがビューワーに返されます。同じ CloudFront エッジロケーションに送信されるオブジェクトに対する後続のリクエストは、 キャッシュから処理されます CloudFront。

ソリューションのデプロイ

この安全な静的ウェブサイトソリューションをデプロイするには、次のいずれかのオプションを選択できます。

- AWS CloudFormation コンソールを使用してデフォルトコンテンツでソリューションをデプロイしてから、ウェブサイトのコンテンツを Amazon S3 にアップロードします。

- ソリューションをコンピュータにクローンして、ウェブサイトのコンテンツを追加します。次に、AWS Command Line Interface (AWS CLI) を使用してソリューションをデプロイします。

Note

CloudFormation テンプレートをデプロイするには、米国東部 (バージニア北部) リージョンを使用する必要があります。

トピック

- [前提条件](#)
- [AWS CloudFormation コンソールを使用する場合](#)
- [ローカルでのソリューションのクローン作成](#)
- [アクセスログの検索](#)

前提条件

このソリューションを使用するには、次の前提条件が必要です。

- Amazon Route 53 ホストゾーンを指している登録済みドメイン名 (example.com など)。ホストゾーンは、このソリューションをデプロイするAWS アカウントのと同じにある必要があります。登録済みドメイン名がない場合、[Route 53 で登録](#)できます。登録済みドメイン名を持っていても、Route 53 のホストゾーンをポイントしていないという場合は、[Route 53 を DNS サービスとして設定](#)します。
- AWS Identity and Access Management IAM ロールを作成する CloudFormation 起動テンプレートへの (IAM) アクセス許可と、ソリューション内のすべてのAWSリソースを作成するアクセス許可。

このソリューションの使用中に発生したコストは、お客様の負担となります。コストの詳細については、[各の料金ページAWSのサービス](#)を参照してください。

AWS CloudFormation コンソールを使用する場合

CloudFormation コンソールを使用してデプロイするには

1. [Launch on AWS] (Launch で起動) を選択して、AWS CloudFormation コンソールでこのソリューションを開きます。必要に応じて、にサインインしますAWS アカウント。



2. スタックの作成ウィザードが CloudFormation コンソールで開き、このソリューション CloudFormation のテンプレートを指定するフィールドがあらかじめ入力されています。

ページの最下部にある [Next] を選択します。

3. [スタック詳細の指定] ページで、次のフィールドに値を入力します。
 - SubDomain – ウェブサイトに使用するサブドメインを入力します。たとえば、サブドメインが `www` の場合、ウェブサイトは `www.example.com` で利用できます。(次の箇条書きで説明するように、`example.com` をドメイン名に置き換えます)。
 - DomainName – `example.com` などのドメイン名を入力します。このドメインは Route 53 ホストゾーンを指している必要があります。
 - HostedZoneId – ドメイン名の Route 53 ホストゾーン ID。

完了したら、[次へ] を選択します。

4. (オプション) [スタックオプションの設定] ページで、[タグおよびその他のスタックオプションを追加します](#)。

完了したら、[次へ] を選択します。

5. [レビュー] ページで、ページの下部までスクロールし、[機能] セクションの 2 つのボックスを選択します。これらの機能は、AWS CloudFormation がスタックのリソースへのアクセスを許可する IAM ロールを作成し、リソースを動的に命名することを可能にします。
6. [スタックの作成] を選択します。
7. スタックの作成が完了するまで待ちます。スタックはネストされたスタックを作成します。完了までに数分かかることがあります。完了すると、[ステータス] が [CREATE_COMPLETE] に変わります。

ステータスが [CREATE_COMPLETE] の場合、<https://www.example.com> に移動してウェブサイトを表示します (`www.example.com` は、ステップ 3 で指定したサブドメイン名およびドメイン名に置き換えます)。ウェブサイトのデフォルトコンテンツが表示されます。

I am a static website!

Great, huh? [Here's a link to another page.](#)

ウェブサイトのデフォルトのコンテンツを独自のコンテンツに置き換えるには

1. <https://console.aws.amazon.com/s3/>でAmazon S3 コンソールを開きます。
2. 名前が `amazon-cloudfront-secure-static-site-s3bucketroot-` で始まるバケットを選択します。

Note

s3bucketlogs ではなく、名前に s3bucketroot を含むバケットを選択してください。名前に s3bucketroot が含まれるバケットには、ウェブサイトのコンテンツが含まれていません。s3bucketlogs が含まれるバケットには、ログファイルのみ含まれています。

3. ウェブサイトのデフォルトコンテンツを削除し、独自のコンテンツをアップロードします。

Note

このソリューションのデフォルトコンテンツでウェブサイトを表示した場合、デフォルトコンテンツの一部が CloudFront エッジロケーションにキャッシュされている可能性があります。更新されたウェブサイトのコンテンツがビューワーに表示されるようにするには、ファイルを無効にして CloudFront、エッジロケーションからキャッシュされたコピーを削除します。詳細については、「[ファイルの無効化](#)」を参照してください。

ローカルでのソリューションのクローン作成

前提条件

このソリューションをデプロイする前にウェブサイトのコンテンツを追加するには、ソリューションのアーティファクトをローカルでパッケージ化する必要があります。これには、Node.js と npm が必要です。詳細については、「<https://www.npmjs.com/get-npm>」を参照してください。

ウェブサイトのコンテンツを追加し、ソリューションをデプロイするには

1. からソリューションをクローンまたはダウンロードします<https://github.com/aws-samples/amazon-cloudfront-secure-static-site> クローンを作成またはダウンロードしたら、コマンドプロンプトまたはターミナルを開き、`amazon-cloudfront-secure-static-site` フォルダに移動します。
2. 次のコマンドを実行し、ソリューションのアーティファクトをインストールしてパッケージ化します。

```
make package-static
```

3. ウェブサイトのコンテンツを `www` フォルダにコピーし、デフォルト Web サイトのコンテンツを上書きします。
4. 次の AWS CLI コマンドを実行して、ソリューションのアーティファクトを保存する Amazon S3 バケットを作成します。を独自のバケット名 `example-bucket-for-artifacts` に置き換えます。

```
aws s3 mb s3://example-bucket-for-artifacts --region us-east-1
```

5. 次の AWS CLI コマンドを実行し、ソリューションのアーティファクトを AWS CloudFormation テンプレートとしてパッケージ化します。を、前のステップで作成したバケットの名前 `example-bucket-for-artifacts` に置き換えます。

```
aws cloudformation package \  
  --region us-east-1 \  
  --template-file templates/main.yaml \  
  --s3-bucket example-bucket-for-artifacts \  
  --output-template-file packaged.template
```

6. 次のコマンドを実行し、AWS CloudFormation を使用してソリューションをデプロイします。次の値を置き換えます。
 - `your-CloudFormation-stack-name` – をAWS CloudFormationスタックの名前に置き換えます。
 - `example.com` - ドメイン名で置き換えます。このドメインは、同じ AWS アカウントの Route 53 ホストゾーンをポイントしている必要があります。
 - `www` - ウェブサイトで使用するサブドメインに置き換えます。たとえば、サブドメインが `www` の場合、ウェブサイトは `www.example.com` で利用できます。

```
aws cloudformation deploy \  
  --region us-east-1 \  
  --stack-name your-CloudFormation-stack-name \  
  --template-file packaged.template \  
  --capabilities CAPABILITY_NAMED_IAM CAPABILITY_AUTO_EXPAND \  
  --parameter-overrides DomainName=example.com SubDomain=www
```

7. AWS CloudFormation スタックの作成が完了するまで待ちます。スタックはネストされたスタックを作成します。完了までに数分かかることがあります。完了すると、[ステータス] が [CREATE_COMPLETE] に変わります。

ステータスが [CREATE_COMPLETE] に変わったら、<https://www.example.com> に移動してウェブサイトを表示します (www.example.com は、前のステップで指定したサブドメイン名およびドメイン名に置き換えます)。ウェブサイトのコンテンツが表示されます。

アクセスログの検索

このソリューションにより、ディストリビューションの[アクセスログ](#)が有効になります CloudFront。ディストリビューションのアクセスログを見つけるには、以下のステップを実行します。

ディストリビューションのアクセスログを見つけるには

1. <https://console.aws.amazon.com/s3/>でAmazon S3 コンソールを開きます。
2. `-amazon-cloudfront-secure-staticsite-s3bucketlogs-` で始まる名前のバケットを選択します。

Note

s3bucketroot ではなく、名前に s3bucketlogs を含むバケットを選択してください。名前に s3bucketlogs が含まれるバケットには、ログファイルが含まれています。s3bucketroot が含まれるバケットには、ウェブサイトのコンテンツが含まれていません。

3. `cdn` という名前のフォルダには、CloudFront アクセスログが含まれています。

ディストリビューションの使用

Amazon CloudFront ディストリビューションを作成して、コンテンツの配信先と、コンテンツ配信の追跡と管理の方法に関する詳細 CloudFront をに伝えます。以下のトピックでは、CloudFront ディストリビューションに関するいくつかの基本を説明し、ビジネスニーズに合わせてディストリビューションを設定するために選択できる設定に関する詳細情報を提供します。

トピック

- [ディストリビューションの概要](#)
- [ディストリビューションの作成、更新、および削除](#)
- [CloudFront 継続的デプロイを使用して CDN 設定の変更を安全にテストする](#)
- [ディストリビューションでの CloudFront さまざまなオリジンの使用](#)
- [代替ドメイン名 \(CNAME\) を追加することによるカスタム URL の使用](#)
- [ディストリビューション WebSockets での CloudFront の使用](#)

ディストリビューションの概要

CloudFront を使用してコンテンツを配信する場合は、ディストリビューションを作成し、次の構成設定から選択します。

- コンテンツオリジン — 配信するファイル CloudFront を取得する Amazon S3 バケット、AWS Elemental MediaPackage チャンネル、AWS Elemental MediaStore コンテナ、Elastic Load Balancing ロードバランサー、または HTTP サーバー。1 つのディストリビューションで、最大で 25 のオリジンの任意に組み合わせて指定できます。
- アクセス - ファイルをすべてのユーザーが使用できるようにするか、または一部のユーザーにアクセスを制限するか。
- セキュリティ - AWS WAF 保護を有効にして、HTTPS を使用したコンテンツへのアクセスを必須にするかどうか。
- キャッシュキー - キャッシュキーに含める値 (存在する場合)。キャッシュキーは、特定のディストリビューションのキャッシュ内の各ファイルを一意に識別します。
- オリジンリクエスト設定 — オリジンに送信するリクエストに HTTP ヘッダー、Cookie、またはクエリ文字列 CloudFront を含めるかどうか。
- 地理的制限 — 選択した国のユーザーがコンテンツにアクセスできない CloudFront ようにするかどうか。

- ログ — 標準ログを作成するか、ビューワーのアクティビティを示すリアルタイムログ CloudFront を作成するか。

AWS アカウントごとに作成できるディストリビューションの現在の最大数については、「[ディストリビューションの一般的なクォータ](#)」を参照してください。ディストリビューションごとにサービスが可能なファイルの最大数はありません。

ディストリビューションを使用して、HTTP または HTTPS 経由で以下のコンテンツを供給できます。

- HTTP または HTTPS を使用した、HTML、CSS JavaScript、画像ファイルなどの静的および動的ダウンロードコンテンツ。
- Apple HTTP Live Streaming (HLS) や Microsoft Smooth Streaming など、さまざまな形式のビデオオンデマンド。詳細については、「[によるビデオオンデマンド \(VOD\) の配信 CloudFront](#)」を参照してください。
- ライブイベント。リアルタイムのミーティング、会議、コンサートなど。ライブストリーミングの場合は、AWS CloudFormation スタックを使用して自動的にディストリビューションを作成できます。詳細については、「[CloudFront とAWSメディアサービスによるライブストリーミングビデオの配信](#)」を参照してください。

ディストリビューションの作成については、[ディストリビューションを作成するためのステップ \(概要\)](#) を参照してください。

ディストリビューションで使用できるアクション

次の表に、ディストリビューションを操作するために実行できる CloudFront アクションを示します。この表には、CloudFront コンソールと CloudFront APIs。

[アクション]	CloudFront コンソールの使用	CloudFront API の使用
ディストリビューションを作成する	「ディストリビューションを作成するためのステップ (概要)」 を参照してください。	CreateDistribution に進みます。
		ListDistributions に進みます。

[アクション]	CloudFront コンソールの使用	CloudFront API の使用
ディストリビューションのリストを表示する	「 ディストリビューションの更新 」を参照してください。	
ディストリビューションに関するすべての情報を取得する	「 ディストリビューションの更新 」を参照してください。	GetDistribution に進みます。
ディストリビューション構成を取得する	「 ディストリビューションの更新 」を参照してください。	GetDistributionConfig に進みます。
ディストリビューションを更新する	「 ディストリビューションの更新 」を参照してください。	UpdateDistribution に進みます。
ディストリビューションを削除する	「 ディストリビューションを削除する 」を参照してください。	DeleteDistribution に進みます。

ディストリビューションの作成および更新で必須である API フィールド

[UpdateDistribution](#) CloudFront API アクションを使用してディストリビューションを更新する場合、[CreateDistribution](#) を使用してディストリビューションを作成する場合よりも多くの必須フィールドがあります。[CreateDistribution](#)。ディストリビューションを更新するには、次のステップを実行します。

1. [GetDistribution](#) を使用して、更新するディストリビューションの現在の設定を取得します。
2. 更新するディストリビューション設定のフィールドを変更します。ETag フィールドの名前を IfMatch に変更します。ただし、フィールドの値は変更しないでください。
3. [UpdateDistribution](#) を使用してディストリビューションを更新し、変更したフィールドや変更していないフィールドなど、ディストリビューション設定全体を提供します。

次の表は、ディストリビューションの作成と更新に必要なフィールドをまとめたものです。

DistributionConfig

メンバー	CreateDistribution API コール で必須	UpdateDistribution API コール で必須
CallerReference	はい	はい
エイリアス	-	必要 (このフィールドは必須ですが、項目がなく数量 0 の場合は有効です)
DefaultRootObject	-	必要 (このフィールドは必須ですが、空の文字列は有効な値です)
Origins	はい	はい
OriginGroups	-	-
DefaultCacheBehavior	はい	はい
CacheBehaviors	-	必要 (このフィールドは必須ですが、項目がなく数量 0 の場合は有効です)
CustomErrorResponses	-	必要 (このフィールドは必須ですが、項目がなく数量 0 の場合は有効です)
コメント	必要 (このフィールドは必須ですが、空の文字列は有効な値です)	必要 (このフィールドは必須ですが、空の文字列は有効な値です)
ログ記録	-	はい
PriceClass	-	はい

メンバー	CreateDistribution API コールで必須	UpdateDistribution API コールで必須
有効	はい	はい
ViewerCertificate	-	はい
制限事項	-	必要 (このフィールドは必須ですが、項目がなく RestrictionType が none で、数量が 0 の場合は有効です)
WebACLId	-	必要 (このフィールドは必須ですが、空の文字列は有効な値です)
HttpVersion	-	はい
IsIPV6Enabled	-	-

CacheBehavior (を含む DefaultCacheBehavior)

メンバー	CreateDistribution API コールで必須	UpdateDistribution API コールで必須
PathPattern (このフィールドは には適用されません DefaultCacheBehavior)	はい	はい
TargetOriginId	はい	はい
TrustedSigners	-	-
TrustedKeyGroups	-	-
ViewerProtocolPolicy	はい	はい

メンバー	CreateDistribution API コール で必須	UpdateDistribution API コール で必須
AllowedMethods	-	はい
SmoothStreaming	-	はい
Compress	-	はい
LambdaFunctionAssociations	-	必要 (このフィールドは必須ですが、項目がなく数量 0 の場合は有効です)
FunctionAssociations	-	-
FieldLevelEncryptionId	-	必要 (このフィールドは必須ですが、空の文字列は有効な値です)
RealtimeLogConfigArn	-	-
CachePolicyId	必要 (次の廃止フィールドを使用する場合、CachePolicyId は不要です。非推奨フィールド: Forwarded Values、MinTTL、DefaultTTL、および MaxTTL)	必要 (次の廃止フィールドを使用する場合、CachePolicyId は不要です。非推奨フィールド: Forwarded Values、MinTTL、DefaultTTL、および MaxTTL)
OriginRequestPolicyId	-	-
ResponseHeadersPolicyId	-	-

ディストリビューションの作成、更新、および削除

以下のトピックでは、Amazon CloudFront ディストリビューションを作成、更新、削除する方法について説明します。

トピック

- [ディストリビューションを作成するためのステップ \(概要\)](#)
- [ディストリビューションの作成](#)
- [ディストリビューションを作成または更新する場合に指定する値](#)
- [コンソール CloudFront に表示される値](#)
- [ディストリビューションのテスト](#)
- [ディストリビューションの更新](#)
- [Amazon CloudFront ディストリビューションのタグ付け](#)
- [ディストリビューションを削除する](#)

ディストリビューションを作成するためのステップ (概要)

以下のタスクリストでは、ディストリビューションの作成処理の概要について説明します。

ディストリビューションを作成するには

1. 1 つ以上の Amazon S3 バケットを作成するか、HTTP サーバーをオリジンサーバーとして構成します。オリジンとは、コンテンツのオリジナルバージョンを保存する場所です。ガブファイルのリクエスト CloudFront を取得すると、オリジンに移動して、エッジロケーションで配信するファイルを取得します。オリジンサーバーとして、10 個の Amazon S3 バケットと HTTP サーバーの任意の組み合わせを使用できます。

Amazon S3 を使用する場合は、バケット名をすべて小文字にする必要があります。また、バケット名にスペースを含めることはできません。

Amazon EC2 サーバーまたは別のカスタムオリジンを使用する場合は、「[Amazon EC2 \(または他のカスタムオリジン\) の使用](#)」を確認してください。

ディストリビューションに対して作成できるオリジンの現在の最大数について、またはクォータの引き上げを要求するには、「[ディストリビューションの一般的なクォータ](#)」を参照してください。

2. コンテンツをオリジンサーバーにアップロードします。オブジェクトをパブリックに読み取り可能にするか、CloudFront 署名URLs を使用してコンテンツへのアクセスを制限できます。

⚠ Important

オリジンサーバーのセキュリティを確保する責任はお客様にあります。CloudFront にサーバーへのアクセス許可があり、セキュリティ設定によってコンテンツが保護されていることを確認する必要があります。

3. CloudFront デイストリビューションを作成します。
 - CloudFront コンソールを使用してデイストリビューションを作成する方法の詳細については、「」を参照してください[デイストリビューションの作成](#)。
 - CloudFront APIs 「Amazon CloudFront API リファレンス[CreateDistribution](#)」の「」を参照してください。
4. (オプション) CloudFront コンソールを使用してデイストリビューションを作成する場合は、デイストリビューションのキャッシュ動作またはオリジンをさらに作成します。動作およびオリジンの詳細については、「[CloudFront デイストリビューションを更新するには](#)」を参照してください。
5. デイストリビューションをテストします。テストの詳細については、「[デイストリビューションのテスト](#)」を参照してください。
6. ステップ 3 でデイストリビューションを作成した後に CloudFront から返されたドメイン名を使用して、そのコンテンツにアクセスするためのウェブサイトまたはアプリケーションを開発します。例えば、がデイストリビューションのドメイン名として `d111111abcdef8.cloudfront.net` を CloudFront 返す場合、Amazon S3 バケットまたは HTTP サーバーのルートディレクトリ `image.jpg` にあるファイルの URL は `https://d111111abcdef8.cloudfront.net/image.jpg` です。

デイストリビューションの作成時に 1 つ以上の代替ドメイン名 (CNAME) を指定した場合、独自のドメイン名を使用できます。この場合、`image.jpg` の URL を `https://www.example.com/image.jpg` にすることができます。

次の点に注意してください。

- 署名付き URL を使用してコンテンツへのアクセスを制限する場合は、「[署名付き URL と署名付き Cookie を使用したプライベートコンテンツの提供](#)」を参照してください。
- 圧縮されたコンテンツを供給する場合は、「[圧縮ファイルの供給](#)」を参照してください。

- Amazon S3 およびカスタムオリジンの CloudFront リクエストとレスポンスの動作については、「」を参照してください[リクエストとレスポンスの動作](#)。 Amazon S3

ディストリビューションの作成

このトピックでは、CloudFront コンソールを使用してディストリビューションを作成する方法について説明します。

CloudFront APIs 「Amazon CloudFront API リファレンス」の[「ディストリビューションの作成」](#)を参照してください。

ディストリビューションの更新の詳細については、このセクションの後半にある[「ディストリビューションの更新」](#)を参照してください。

AWS アカウントごとに作成できるディストリビューションの現在の最大数、またはクォータの引き上げを要求するには、「[ディストリビューションの一般的なクォータ](#)」を参照してください。

ディストリビューションを作成するには (コンソール)

1. にサインインAWS Management Consoleし、 で CloudFront コンソールを開きます<https://console.aws.amazon.com/cloudfront/v4/home>。
2. ナビゲーションペインで [ディストリビューション] を選択し、[ディストリビューションを作成] を選択します。
3. ディストリビューションの設定項目を指定します。詳細については、「[ディストリビューションを作成または更新する場合に指定する値](#)」を参照してください。
4. 変更を保存します。
5. がディストリビューション CloudFront を作成すると、ディストリビューションのステータス列の値が からデプロイされた InProgressに変更されます。ディストリビューションを有効にするように選択した場合、ステータスが [Deployed (デプロイ済み)] に切り替わると、リクエストを処理する準備ができています。

がディストリビューション CloudFront に割り当てるドメイン名がディストリビューションのリストに表示されます。(ドメイン名は、選択されたディストリビューションの [General] タブにも表示されます)。

i Tip

「」の CloudFront 手順に従って、によって割り当てられた名前の代わりに代替ドメイン名を使用できます [代替ドメイン名 \(CNAME\) を追加することによるカスタム URL の使用](#)。

6. ディストリビューションがデプロイされたら、新しい CloudFront URL または CNAME を使用してコンテンツにアクセスできることを確認します。詳細については、「[ディストリビューションのテスト](#)」を参照してください。

ディストリビューションを更新 (たとえば、キャッシュ動作の追加または変更) するには、[ディストリビューションの更新](#) を参照してください。

ディストリビューションを作成または更新する場合に指定する値

[CloudFront コンソール](#) を使用して新しいディストリビューションを作成するか、既存のディストリビューションを更新する場合は、次の値を指定します。

[the section called “オリジンの設定”](#)

- [the section called “オリジンドメイン”](#)
- [the section called “プロトコル \(カスタムオリジンのみ\)”](#)
- [the section called “オリジンのパス”](#)
- [the section called “名前”](#)
- [the section called “オリジンアクセス \(Amazon S3 オリジンのみ\)”](#)
- [the section called “カスタムヘッダーを追加する”](#)
- [the section called “Origin Shield を有効にする”](#)
- [the section called “接続の試行”](#)
- [the section called “接続タイムアウト”](#)
- [the section called “応答タイムアウト \(カスタムオリジンのみ\)”](#)
- [the section called “キープアライブタイムアウト \(カスタムオリジンのみ\)”](#)

[キャッシュ動作の設定](#)

以下の値は、ディストリビューションを作成するときに [デフォルトのキャッシュ動作の設定] に適用されます。これらは後で作成する他のキャッシュ動作にも適用されます。

- [パスパターン](#)
- [オリジンまたはオリジングループ](#) (既存のディストリビューションのキャッシュ動作を作成または更新する場合のみ適用されます)
- [ビューワプロトコルポリシー](#)
- [許可される HTTP メソッド](#)
- [フィールドレベル暗号化の設定](#)
- [キャッシュされる HTTP メソッド](#)
- [選択されたリクエストヘッダーに基づいたキャッシュ](#)
- [許可リストヘッダー](#) ([[選択されたリクエストヘッダーに基づくキャッシュ](#)] で [許可リスト] を選択した場合にのみ適用されます)
- [オブジェクトキャッシュ](#)
- [最小 TTL](#)
- [最大 TTL](#)
- [デフォルト TTL](#)
- [cookie の転送](#)
- [許可リストCookie](#) ([[Cookies を転送する](#)] で [許可リスト] を選択した場合にのみ適用されます)
- [クエリ文字列の転送とキャッシュ](#)
- [クエリ文字列の許可リスト](#) ([[クエリ文字列の転送とキャッシュ](#)] で [すべて転送 - 許可リストに基づいてキャッシュ] を選択した場合にのみ適用されます)
- [スムーズストリーミング](#)
- [ビューワのアクセス制限 \(署名付き URL または署名付き cookie の使用\)](#)
- [信頼された署名者](#) ([[Restrict Viewer Access \(Use Signed URLs or Signed Cookies\)](#)] (ビューワのアクセスを制限 (署名付き URL または署名付き Cookie の使用))] で [Yes (はい)] を選択した場合にのみ適用されます)
- [AWS アカウント番号](#) ([[Trusted Signers \(信頼された署名者\)](#)] で [[Specify Accounts \(アカウントを指定\)](#)] を選択した場合にのみ適用されます)
- [オブジェクトを自動的に圧縮する](#)

以下の値は、[[Lambda Function Associations \(Lambda 関数の関連付け\)](#)] に適用されます。

- [CloudFront イベント](#)
- [Lambda 関数の ARN](#)
- [本文を含める](#)

[ディストリビューションの設定](#)

- [価格クラス](#)
- [AWS WAF ウェブ ACL](#)
- [代替ドメイン名 \(CNAME\)](#)
- [SSL 証明書](#)
- [独自 SSL クライアントのサポート](#) ([SSL Certificate (SSL 証明書)] で [Custom SSL Certificate (example.com) (独自 SSL 証明書 (example.com))] を選択した場合にのみ適用されます)
- [セキュリティポリシー](#) (SSL/TLS の最小バージョン)
- [サポートされる HTTP バージョン](#)
- [デフォルトのルートオブジェクト](#)
- [ログ記録](#)
- [ログ用のバケット](#)
- [ログのプレフィックス](#)
- [cookie のログ作成](#)
- [IPv6 を有効にする](#)
- [コメント](#)
- [ディストリビューションの状態](#)

[カスタムエラーページとエラーキャッシュ](#)

- [HTTP エラーコード](#)
- [Response page path \(レスポンスページのパス\)](#)
- [HTTP レスポンスコード](#)
- [Error caching minimum TTL \(seconds\) \(エラーキャッシュ最小 TTL \(秒\)\)](#)

[地理的制限](#)

CloudFront コンソールを使用してディストリビューションを作成または更新する方法の詳細については、「[the section called “ディストリビューションの作成”](#)」または「[the section called “ディストリビューションの更新”](#)」を参照してください。

オリジンの設定

CloudFront コンソールを使用してディストリビューションを作成または更新するときは、オリジンと呼ばれる 1 つ以上の場所に関する情報を指定します。ここでは、ウェブコンテンツの元のバージョンを保存します。CloudFront は、オリジンからウェブコンテンツを取得し、エッジサーバーの世界ネットワークを介してビューワーに提供します。

ディストリビューションに対して作成できるオリジンの現在の最大数について、またはクォータの引き上げを要求するには、「[the section called “ディストリビューションの一般的なクォータ”](#)」を参照してください。

オリジンを削除する場合は、まず、そのオリジンに関連付けられたキャッシュ動作を編集または削除する必要があります。

Important

オリジンを削除する場合は、そのオリジンによって以前供給されていたファイルが別のオリジンで利用可能であり、現在、そのファイルへのリクエストがキャッシュ動作によって新しいオリジンにルーティングされていることを確認します。

ディストリビューションを作成または更新する場合、オリジンごとに以下の値を指定します。

オリジンドメイン

オリジンドメインは、このオリジンのオブジェクト CloudFront を取得する Amazon S3 バケットまたは HTTP サーバーの DNS ドメイン名です。次に例を示します。

- Amazon S3 バケット – *DOC-EXAMPLE-BUCKET.s3.us-west-2.amazonaws.com*

Note

最近 S3 バケットを作成した場合、CloudFront ディストリビューションは最大 24 時間 HTTP 307 Temporary Redirect レスポンスを返すことがあります。S3 バケット名がすべての AWS リージョンに反映されるまでに、最大 24 時間かかることがあります。反

映が完了すると、ディストリビューションは自動的にこれらのリダイレクトレスポンスの送信を停止します。何もする必要はありません。詳細については、「[Why am I getting an HTTP 307 Temporary Redirect response from Amazon S3?](#)」と「[一時的なリクエストのリダイレクト](#)」を参照してください。

- ウェブサイトとして設定された Amazon S3 バケット – `DOC-EXAMPLE-BUCKET.s3-website.us-west-2.amazonaws.com`
- MediaStore コンテナ – `examplemediastore.data.mediastore.us-west-1.amazonaws.com`
- MediaPackage エンドポイント – `examplemediapackage.mediapackage.us-west-1.amazonaws.com`
- Amazon EC2 インスタンス – `ec2-203-0-113-25.compute-1.amazonaws.com`
- Elastic Load Balancing ロードバランサー – `example-load-balancer-1234567890.us-west-2.elb.amazonaws.com`
- 独自のウェブサーバー – `https://www.example.com`

[Origin Domain Name] (オリジンドメイン名) フィールドでドメイン名を選択するか、名前を入力します。ドメイン名では、大文字と小文字が区別されません。

オリジンが Amazon S3 バケットの場合は、次の点に注意してください。

- バケットがウェブサイトとして構成されている場合は、バケットの Amazon S3 静的ウェブサイトホスティングエンドポイントを入力します。[Origin domain] (オリジンドメイン) フィールドのバケット一覧からバケット名を選択しないでください。静的ウェブサイトホスティングエンドポイントは、Amazon S3 コンソールの [Static website hosting] (静的ウェブサイトホスティング) の [Properties] (プロパティ) ページに表示されます。詳細については、「[the section called “ウェブサイトのエンドポイントとして設定された Amazon S3 バケットの使用”](#)」を参照してください。
- バケットに Amazon S3 Transfer Acceleration を設定した場合、[Origin domain] (オリジンドメイン) に `s3-accelerate` エンドポイントを指定しないでください。
- 使用しているバケットが別の AWS アカウントのものであり、そのバケットがウェブサイトとして構成されていない場合は、次の形式で名前を入力します。

`bucket-name.s3.region.amazonaws.com`

バケットが米国リージョンにあり、Amazon S3 がバージニア北部の施設にリクエストをルーティングするように設定する場合は、次の形式を使用します。

`bucket-name.s3.us-east-1.amazonaws.com`

- CloudFront オリジンアクセスコントロールを使用して Amazon S3 のコンテンツを保護しない限り、ファイルはパブリックに読み取り可能である必要があります。アクセスコントロールの詳細については、「[the section called “Amazon S3 オリジンへのアクセスの制限”](#)」を参照してください。

⚠ Important

オリジンが Amazon S3 バケットの場合、バケット名は DNS 命名要件に準拠する必要があります。詳細については、Amazon Simple Storage Service ユーザーガイドの[バケットの制約と制限](#)を参照してください。

オリジンのオリジンドメインの値を変更すると、CloudFront は CloudFront エッジロケーションへの変更のレプリケーションを直ちに開始します。特定のエッジロケーションでディストリビューション設定が更新されるまで、CloudFront はリクエストを以前のオリジンに転送します。そのエッジロケーションでディストリビューション設定が更新されるとすぐに、CloudFront は新しいオリジンへのリクエストの転送を開始します。

オリジンを変更 CloudFront しても、エッジキャッシュに新しいオリジンのオブジェクトを再入力する必要はありません。アプリケーションのビューワーリクエストが変更されていない限り、は、各オブジェクトの TTL の有効期限が切れるか、リクエスト頻度の低いオブジェクトが削除されるまで、エッジキャッシュに既に存在するオブジェクトの処理 CloudFront を継続します。

オリジンのパス

オリジンのディレクトリからコンテンツを CloudFront リクエストする場合は、スラッシュ (/) で始まるディレクトリパスを入力します。は、などのオリジンドメインの値にディレクトリパス CloudFront を追加します `cf-origin.example.com/production/images`。パスの末尾にはスラッシュ (/) を付けしないでください。

例えば、特定のディストリビューションに対して次の値を指定したとします。

- [Origin domain] (オリジンドメイン) - **DOC-EXAMPLE-BUCKET** と名前の付いた Amazon S3 バケット
- オリジンのパス - **/production**
- 代替ドメイン名 (CNAME) - **example.com**

ユーザーがブラウザexample.com/index.htmlに を入力すると、 CloudFront は のリクエストを Amazon S3 に送信しますDOC-EXAMPLE-BUCKET/production/index.html。

ユーザーがブラウザexample.com/acme/index.htmlに を入力すると、 は のリクエストを Amazon S3 CloudFront に送信しますDOC-EXAMPLE-BUCKET/production/acme/index.html。

名前

名前は、このディストリビューション内でこのオリジンを一意に識別する文字列です。デフォルトの キャッシュ動作に加えてキャッシュ動作を作成する場合は、ここで指定した名前を使用して、リクエストがそのキャッシュ動作のパスパターンに一致したときにリクエストをルーティング CloudFront するオリジンを識別します。

カスタムヘッダーを追加する

リクエスト CloudFront をオリジンに送信するたびにカスタムヘッダーを追加する場合は、ヘッダー名とその値を指定します。詳細については、「[the section called “オリジンリクエストへのカスタムヘッダーの追加”](#)」を参照してください。

現在、追加できるカスタムヘッダーの最大数、カスタムヘッダー名と値の最大長、すべてのヘッダー名と値の全長の最大長については、[クォータ](#) を参照してください。

Origin Shield を有効にする

はい を選択して CloudFront Origin Shield を有効にします。Origin Shield の詳細については、[the section called “Origin Shield の使用”](#) を参照してください。

接続の試行

がオリジンへの接続 CloudFront を試みる回数を設定できます。試行回数として 1、2、または 3 を指定できます。デフォルト値 (特に指定しない場合) は 3 です。

この設定を接続タイムアウトとともに使用して、セカンダリオリジンに接続しようとしたり、ビューワーにエラーレスポンスを返したりするまでの CloudFront 待機時間を指定します。デフォルトでは、 はセカンダリオリジンに接続しようとしたり、エラーレスポンスを返したりする前に、30 秒 (それぞれ 10 秒間の試行が 3 回) CloudFront 待機します。試行回数を減らすか、接続タイムアウトを短くするか、その両方を行うことで、この時間を短縮できます。

指定された接続試行回数が失敗した場合、 は次のいずれか CloudFront を実行します。

- オリジンがオリジングループの一部である場合、CloudFront はセカンダリオリジンへの接続を試みます。セカンダリオリジンへの接続試行が指定された回数失敗した場合、はビューワーにエラーレスポンス CloudFront を返します。
- オリジンがオリジングループの一部でない場合、はビューワーにエラーレスポンス CloudFront を返します。

カスタムオリジン (静的ウェブサイトホスティングで設定された Amazon S3 バケットを含む) の場合、この設定では、がオリジンからレスポンスを取得 CloudFront しようとする回数も指定します。詳細については、「[the section called “応答タイムアウト \(カスタムオリジンのみ\)”](#)」を参照してください。

接続タイムアウト

接続タイムアウトは、オリジンへの接続を確立しようとしたときに が CloudFront 待機する秒数です。1 ~ 10 (両端の値を含む) の秒数を指定できます。デフォルトのタイムアウト (特に指定しない場合) は 10 秒です。

この設定を Connection と組み合わせて使用すると、がセカンダリオリジンに接続しようとしたり、エラーレスポンスをビューワーに返したりするまでの CloudFront 待機時間を指定できます。デフォルトでは、CloudFront はセカンダリオリジンへの接続を試行したり、エラーレスポンスを返したりする前に 30 秒 (それぞれ 10 秒間の試行が 3 回) 待機します。試行回数を減らすか、接続タイムアウトを短くするか、その両方を行うことで、この時間を短縮できます。

CloudFront が指定された秒数以内にオリジンへの接続を確立しない場合、は次のいずれか CloudFront を実行します。

- 指定された接続試行回数が 1 回を超える場合、は接続の確立を CloudFront 再試行します。は接続 CloudFront 試行 の値によって決まるように、最大 3 回試行します。
- すべての接続試行が失敗し、オリジンがオリジングループの一部である場合、CloudFront はセカンダリオリジンへの接続を試みます。セカンダリオリジンへの接続試行が指定された回数失敗した場合、はビューワーにエラーレスポンス CloudFront を返します。
- すべての接続試行が失敗し、オリジンがオリジングループの一部でない場合、はビューワーにエラーレスポンス CloudFront を返します。

応答タイムアウト (カスタムオリジンのみ)

Note

これは、カスタムオリジンにのみ適用されます。

オリジン応答タイムアウト (オリジンの読み取りタイムアウトまたはオリジンリクエストタイムアウトとも呼ばれる) は、次の両方の値に適用されます。

- リクエストをオリジンに転送した後のレスポンスの CloudFront 待機時間 (秒単位)。
- オリジンからレスポンスのパケットを受信してから次のパケットを受信してから CloudFront 待機する時間 (秒単位)。

デフォルトのタイムアウトは 30 秒です。この値は、1 ~ 60 秒の範囲で変更できます。この範囲外のタイムアウト値が必要な場合は、[デケースを作成します AWS Support Center Console](#)。

Tip

ビューワーで HTTP 504 ステータスコードエラーが発生しているために、タイムアウト値を引き上げる必要がある場合は、タイムアウト値を変更する前に、それらのエラーを回避するその他の方法を検討します。「[the section called “HTTP 504 ステータスコード \(Gateway Timeout\)”](#)」でトラブルシューティングのヒントを参照してください。

CloudFront 動作は、ビューワーリクエストの HTTP メソッドによって異なります。

- GET および HEAD リクエスト – オリジンが応答しないか、応答タイムアウトの期間内に応答を停止した場合、は接続を CloudFront ドロップし、の値に従って接続 CloudFront を再試行します [the section called “接続の試行”](#)。
- DELETE、OPTIONS、PATCH、および PUTPOST リクエスト – オリジンが読み取りタイムアウトの間応答しない場合、CloudFront は接続を中断し、オリジンへの接続を再試行しません。クライアントは、必要に応じてリクエストを再送信できます。

キープアライブタイムアウト (カスタムオリジンのみ)

Note

これは、カスタムオリジンにのみ適用されます。

キープアライブタイムアウトは、レスポンスの最後のパケットを取得した後、カスタムオリジンへの接続を維持 CloudFront しようとする時間 (秒単位) です。持続的接続を維持すると、TCP 接続の再構築に必要な時間と後続のリクエストに対する別の TLS ハンドシェイクの実行に必要な時間を節約できます。キープアライブタイムアウトを増やすと、ディストリビューションの request-per-connection メトリクスが向上します。

Note

[Origin Keep-alive Timeout] (オリジンキープアライブタイムアウト) の値が有効になるためには、永続接続を許可するようにオリジンが設定されている必要があります。

デフォルトのタイムアウトは 5 秒です。この値は、1~60 秒の範囲で変更できます。60 秒を超えるキープアライブタイムアウトが必要な場合は、[でケースを作成しますAWS Support Center Console](#)。

オリジンアクセス (Amazon S3 オリジンのみ)

Note

これは、Amazon S3 バケットオリジン (S3 静的ウェブサイトエンドポイントを使用していないオリジン) にのみ適用されます。

Amazon S3 バケットオリジンへのアクセスを特定のディストリビューションのみに制限できるようにするには、オリジンアクセスコントロール設定 (推奨) を選択します。CloudFront

Amazon S3 バケットオリジンがパブリックにアクセス可能な場合は、[パブリック] を選択します。

詳細については、「[the section called “Amazon S3 オリジンへのアクセスの制限”](#)」を参照してください。

CloudFront URLs 「」を参照してください [the section called “カスタムオリジン上のファイルへのアクセス制限”](#)。

プロトコル (カスタムオリジンのみ)

Note

これは、カスタムオリジンにのみ適用されます。

オリジンからオブジェクトを取得するときに CloudFront 使用するプロトコルポリシー。

次のいずれかの値を選択します。

- HTTP のみ： CloudFront オリジンへのアクセスには HTTP のみを使用します。

Important

Amazon S3 は静的ウェブサイトホスティングエンドポイントをサポートしていないため、オリジンが Amazon S3 の静的ウェブサイトホスティングエンドポイントの場合、[HTTPS only] (HTTP のみ) がデフォルト設定です。CloudFront コンソールは、エンドポイントをホストする Amazon S3 静的ウェブサイトのこの設定の変更をサポートしていません。

- HTTPS のみ： CloudFront オリジンへのアクセスには HTTPS のみを使用します。
- ビューワーの一致： CloudFront ビューワーリクエストのプロトコルに応じて、HTTP または HTTPS を使用してオリジンと通信します。ビューワーが HTTP プロトコルと HTTPS プロトコルの両方を使用してリクエストを行った場合でも、オブジェクトは 1 回だけ CloudFront キャッシュされます。

Important

がこのオリジン CloudFront に転送する HTTPS ビューワーリクエストの場合、オリジンサーバーの SSL/TLS 証明書のドメイン名の 1 つが、オリジンドメインに指定したドメイン名と一致する必要があります。それ以外の場合は、リクエストされたオブジェクトを返す代わりに、HTTP ステータスコード 502 (Bad Gateway) でビューワーリクエスト CloudFront に応答します。詳細については、「[the section called “で SSL/TLS 証明書を使用するための要件 CloudFront”](#)」を参照してください。

HTTP ポート

Note

これは、カスタムオリジンにのみ適用されます。

(オプション) カスタムオリジンがリスンする HTTP ポートを指定できます。有効な値には、ポート 80、443、および 1024 ~ 65535 が含まれます。デフォルト値はポート 80 です。

Important

オリジンが Amazon S3 の静的ウェブサイトホスティングエンドポイントの場合、ポート 80 がデフォルト設定です。Amazon S3 は、静的ウェブサイトホスティングエンドポイントではポート 80 のみサポートするためです。CloudFront コンソールは、エンドポイントをホストする Amazon S3 静的ウェブサイトのこの設定の変更をサポートしていません。

HTTPS ポート

Note

これは、カスタムオリジンにのみ適用されます。

(オプション) カスタムオリジンがリスンする HTTPS ポートを指定できます。有効な値には、ポート 80、443、および 1024 ~ 65535 が含まれます。デフォルト値はポート 443 です。[Protocol] (プロトコル) は、[HTTP only] (HTTP のみ) に設定されます。[HTTPS port] (HTTPS ポート) の値を指定することはできません。

最小限のオリジン SSL プロトコル

Note

これは、カスタムオリジンにのみ適用されます。

オリジンへの HTTPS 接続を確立するときに CloudFront が使用できる最低限の TLS/SSL プロトコルを選択します。より低い TLS プロトコルは安全性が低いため、オリジンがサポートする最新の TLS

を使用することが推奨されます。[Protocol] (プロトコル) は、[HTTP only] (HTTP のみ) に設定されます。[Minimum origin SSL protocol] (最小限のオリジン SSL プロトコル) の値を指定することはできません。

CloudFront API を使用して CloudFront で使用する TLS/SSL プロトコルを設定する場合、最小プロトコルを設定することはできません。代わりに、[がオリジン CloudFront で使用できるすべての TLS/SSL プロトコルを指定します](#)。詳細については、「[Amazon CloudFront API リファレンス OriginSslProtocols](#)」の「」を参照してください。

キャッシュ動作の設定

キャッシュ動作を設定することで、ウェブサイト上のファイルの特定の URL パターンにさまざまな CloudFront 機能を設定できます。たとえば、CloudFront のオリジンサーバーとして使用しているウェブサーバーの images ディレクトリ内にあるすべての .jpg ファイルに 1 つのキャッシュ動作を適用します。キャッシュ動作ごとに構成可能な機能には以下のようなものがあります。

- パスパターン
- CloudFront ディストリビューションに複数のオリジンを設定している場合、リクエスト CloudFront を転送するオリジン
- クエリ文字列をオリジンに転送するかどうか
- 指定したファイルへのアクセスに署名付き URL を必要とするかどうか
- これらのファイルへのアクセスに HTTPS を使用するようユーザーに要求するかどうか
- オリジンがファイルに追加する Cache-Control ヘッダーの値に関係なく、それらのファイルが CloudFront キャッシュに保持される最小時間

新しいディストリビューションを作成する場合、デフォルトのキャッシュ動作の設定を指定します。デフォルトのキャッシュ動作では、ディストリビューションの作成時に指定されたオリジンにすべてのリクエストが自動的に転送されます。ディストリビューションを作成したら、パスパターンに一致するオブジェクトのリクエストを受信したときの CloudFront 応答を定義する追加のキャッシュ動作を作成できます。例えば、`です*.jpg`。追加のキャッシュ動作を定義した場合、デフォルトのキャッシュ動作は常に最後に処理されます。その他のキャッシュ動作は、CloudFront コンソールにリストされている順序で処理されます。CloudFront API を使用している場合は、ディストリビューションの DistributionConfig 要素にリストされている順序で処理されます。詳細については、「[パスパターン](#)」を参照してください。

キャッシュ動作を作成するときは、オブジェクトを取得するオリジンを 1 つ指定 CloudFront します。そのため、すべてのオリジンからオブジェクト CloudFront を配信する場合は、少なくともオ

オリジンと同じ数のキャッシュ動作 (デフォルトのキャッシュ動作を含む) が必要です。例えば、オリジンが 2 つあり、デフォルトのキャッシュ動作のみがある場合、デフォルトのキャッシュ動作により、はオリジンの 1 つからオブジェクト CloudFront を取得しますが、もう 1 つのオリジンは使用されません。

ディストリビューションに対して作成できるキャッシュ動作の現在の最大数、またはクォータの引き上げを要求するキャッシュ動作の最大数については、「[ディストリビューションの一般的なクォータ](#)」を参照してください。

パスパターン

パスパターン (例: `images/*.jpg`) は、このキャッシュ動作をどのリクエストに割り当てるかを指定します。がエンドユーザーリクエスト CloudFront を受信すると、リクエストされたパスは、キャッシュ動作がディストリビューションにリストされている順序でパスパターンと比較されます。最初の一一致によって、そのリクエストに適用されるキャッシュ動作が決まります。たとえば、以下の 3 つのパスパターンを持つ 3 つのキャッシュ動作がこの順序で設定されているとします。

- `images/*.jpg`
- `images/*`
- `*.gif`

Note

オプションで、パスパターンの先頭にスラッシュ (/) を含めることができます `/images/*.jpg`。例えば、. CloudFront behavior は先頭の / の有無にかかわらず同じです。パスの先頭に / を指定しない場合、この文字は自動的に暗示され、先頭の / の有無にかかわらず同じパスが CloudFront 処理されます。例えば、CloudFront は `/*product.jpg` と同じ処理を行います。 `*product.jpg`

ファイル `images/sample.gif` のリクエストは 1 番目のパスパターンを満たさないため、関連付けられたキャッシュ動作はこのリクエストに適用されません。ファイルは 2 番目のパスパターンを満たします。リクエストは 3 番目のパスパターンにも一致しますが、2 番目のパスパターンに関連付けられたキャッシュ動作が適用されます。

Note

新しいディストリビューションを作成すると、デフォルトのキャッシュ動作の [Path Pattern (パスパターン)] の値は * (すべてのファイル) に設定され、この値は変更できません。この値を指定する CloudFront と、 はオブジェクトに対するすべてのリクエストを [オリジンメイン](#) フィールドで指定したオリジンに転送します。オブジェクトのリクエストが他のキャッシュ動作のパスパターンと一致しない場合、 はデフォルトのキャッシュ動作で指定した動作 CloudFront を適用します。

Important

パスパターンとその順序を慎重に定義します。適切に定義されていない場合、コンテンツへの意図されないアクセスがユーザーに与えられる場合があります。たとえば、リクエストが、2つのキャッシュ動作のパスパターンに一致したと仮定します。最初のキャッシュ動作は署名付き URL を要求しませんが、2番目のキャッシュ動作は署名付き URL を要求します。は最初の一致に関連するキャッシュ動作 CloudFront を処理するため、ユーザーは署名付き URL を使用せずにオブジェクトにアクセスできます。

MediaPackage チャンネルを使用する場合は、オリジンのエンドポイントタイプに対して定義するキャッシュ動作の特定のパスパターンを含める必要があります。たとえば、DASH エンドポイントの場合は、[Path Pattern (パスパターン)] に「*.mpd」と入力します。詳細と具体的な手順については、「[AWS Elemental MediaPackage でフォーマットされたライブ動画の配信](#)」を参照してください。

指定したパスは、指定したディレクトリ内および指定した directory. CloudFront does 以下のサブディレクトリ内のすべてのファイルのリクエストに適用されます。パスパターンを評価する場合、クエリ文字列や Cookie は考慮されません。たとえば、images ディレクトリに product1 および product2 サブディレクトリが含まれる場合、パスパターン images/*.jpg は、images、images/product1、および images/product2 ディレクトリ内のあらゆる .jpg ファイルのリクエストに適用されます。異なるキャッシュ動作を、images/product1 および images ディレクトリのファイルではなく images/product2 ディレクトリのファイルに割り当てる場合、images/product1 用の独立したキャッシュ動作を作成し、そのキャッシュ動作を images ディレクトリ用のキャッシュ動作の上 (前) の位置に移動します。

パスパターンには、以下のワイルドカード文字を使用できます。

- * は、0 個以上の文字に一致します。
- ? は、正確に 1 個の文字に一致します。

以下の例を使用して、ワイルドカード文字がどのように機能するかを示します。

パスパターン	パスパターンに一致するファイル
*.jpg	すべての .jpg ファイル。
images/*.jpg	images ディレクトリ内、および images ディレクトリ下のサブディレクトリ内のすべての .jpg ファイル。
a*.jpg	<ul style="list-style-type: none"> • ファイル名が a で始まるすべての .jpg ファイル (例: apple.jpg、appalachian_trail_2012_05_21.jpg)。 • ファイルパスが a で始まるすべての .jpg ファイル (例: abra/cadabra/magic.jpg)。
a??.jpg	ファイル名が a で始まり、ファイル名の後に正確に他の 2 文字が続くすべての .jpg ファイル (例: ant.jpg、abe.jpg)。
.doc	ファイル名拡張子が .doc で始まるすべてのファイル (例: .doc、.docx、.docm ファイル)。この場合、パスパターン *.doc? を使用することはできません。このパスパターンは .doc ファイルのリクエストに適用されないためです。? ワイルドカード文字は正確に 1 個の文字を置き換えるものです。

パスパターンの最大長は 255 文字です。値には以下の文字を含めることができます。

- A ~ Z、a ~ z

パスパターンでは大文字と小文字が区別されるので、パスパターン *.jpg はファイル LOGO.JPG には適用されません。

- 0-9
- _ - . * \$ / ~ " ' @ : +
- & (& として受け渡しされます)

オリジンまたはオリジングループ

既存のオリジンまたはオリジングループの値を入力します。これにより、リクエスト (https://example.com/logo.jpg など) がキャッシュ動作 (*.jpg など) またはデフォルトのキャッシュ動作 (*) のパスパターンと一致する場合に、リクエストをルーティング CloudFront するオリジンまたはオリジングループを識別します。

ビューワープロトコルポリシー

CloudFront エッジロケーションのコンテンツにアクセスするためにビューワーが使用するプロトコルポリシーを選択します。

- [HTTP and HTTPS (HTTP と HTTPS)]: ビューワーは両方のプロトコルを使用できます。
- [Redirect HTTP to HTTPS (HTTP を HTTPS にリダイレクト)]: ビューワーは両方のプロトコルを使用できますが、HTTP リクエストは自動的に HTTPS リクエストにリダイレクトされます。
- [HTTPS Only (HTTPS のみ)]: ビューワーは HTTPS を使用している場合にのみコンテンツにアクセスできます。

詳細については、「[ビューワーと間の通信に HTTPS を必須にする CloudFront](#)」を参照してください。

許可される HTTP メソッド

処理 CloudFront してオリジンに転送する HTTP メソッドを指定します。

- GET、HEAD: は、オリジンからオブジェクトを取得したり、オブジェクトヘッダーを取得したりするために CloudFront のみ使用できます。
- [GET, HEAD, OPTIONS (GET、HEAD、OPTIONS)]: CloudFront を使用して、オブジェクト、オブジェクトヘッダー、またはオリジンサーバーがサポートしているオプションのリストのいずれかのみをオリジンから取得できます。

- GET、HEAD、OPTIONS、PUT、POST、PATCH、DELETE: CloudFront を使用して、オブジェクトの取得、追加、更新、削除、およびオブジェクトヘッダーの取得を行うことができます。また、ウェブフォームからのデータの送信など、その他の POST 操作も実行できます。

Note

CloudFront は、GET リクエストと HEAD リクエスト、およびオプションで OPTIONS リクエストへのレスポンスをキャッシュします。OPTIONS リクエストへのレスポンスは、GET および HEAD リクエストへのレスポンスとは別にキャッシュされます (OPTIONS メソッドは OPTIONS リクエストの [キャッシュキー](#) に含まれます)。CloudFront は、他のメソッドを使用するリクエストへのレスポンスをキャッシュしません。

Important

[GET, HEAD, OPTIONS] または [GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE] を選択した場合は、望ましくない操作がユーザーによって実行されないように、Amazon S3 バケットまたはカスタムオリジンへのアクセスを制限する必要があることがあります。以下の例は、アクセスを制限する方法を示しています。

- ディストリビューションのオリジンとして Amazon S3 を使用している場合: Amazon S3 コンテンツへのアクセスを制限する CloudFront オリジンアクセスコントロールを作成し、オリジンアクセスコントロールにアクセス許可を付与します。例えば、を使用したいというだけの理由で、これらのメソッドを受け入れて転送 CloudFront するようにを設定しても PUT、DELETE リクエストを適切に処理するように Amazon S3 バケットポリシーを設定する必要があります。詳細については、「[Amazon S3 オリジンへのアクセスの制限](#)」を参照してください。
- カスタムオリジンを使用している場合: すべてのメソッドを処理するようにオリジンサーバーを設定します。例えば、を使用したいというだけの理由で、これらのメソッドを受け入れて転送 CloudFront するようにを設定しても POST、DELETE リクエストを適切に処理するようにオリジンサーバーを設定する必要があります。

フィールドレベル暗号化の設定

特定のデータフィールドにフィールドレベル暗号化を適用する場合は、ドロップダウンリストからフィールドレベル暗号化の設定を選択します。

詳細については、「[フィールドレベル暗号化を使用した機密データの保護](#)」を参照してください。

キャッシュされる HTTP メソッド

ビューワーが OPTIONS リクエストを送信したときに、オリジンからのレスポンスを CloudFront キャッシュするかどうかを指定します。CloudFront always は、リクエスト GET と HEAD リクエストにレスポンスをキャッシュします。

選択されたリクエストヘッダーに基づいたキャッシュ

指定されたヘッダーの値に基づいてオブジェクトを CloudFront キャッシュするかどうかを指定します。

- なし (キャッシュを改善) — CloudFront ヘッダー値に基づいてオブジェクトをキャッシュしません。
- Allowlist – 指定されたヘッダーの値のみに基づいてオブジェクトを CloudFront キャッシュします。Allowlist Headers CloudFront を使用して、キャッシュの基準にするヘッダーを選択します。
- All – このキャッシュ動作に関連付けられているオブジェクトはキャッシュ CloudFront されません。代わりに、はすべてのリクエストをオリジン CloudFront に送信します。(Amazon S3 オリジンには推奨されません)。

選択したオプションにかかわらず、は特定のヘッダーをオリジン CloudFront に転送し、転送したヘッダーに基づいて特定のアクションを実行します。がヘッダー転送 CloudFront を処理する方法の詳細については、「」を参照してください[HTTP リクエストヘッダーと CloudFront 動作 \(カスタムオリジンと Amazon S3 オリジン\)](#)。

リクエストヘッダー CloudFront を使用して でキャッシュを設定する方法の詳細については、「」を参照してください[リクエストヘッダーに基づくコンテンツのキャッシュ](#)。

許可リストヘッダー

オブジェクト CloudFront をキャッシュするときに考慮するヘッダーを指定します。使用可能なヘッダーのリストからヘッダーを選択し、[Add (追加)] を選択します。カスタムヘッダーを転送するには、フィールドにそのヘッダーの名前を入力して [Add Custom (カスタムの追加)] を選択します。

キャッシュ動作ごとに許可リストに追加できるヘッダーの現在の最大数、またはクォータ (以前は制限と呼ばれていました) の引き上げを要求するヘッダーの最大数については、「[ヘッダーのクォータ](#)」を参照してください。

オブジェクトキャッシュ

オリジンサーバーがオブジェクトに Cache-Control ヘッダーを追加して、オブジェクトが CloudFront キャッシュに保持される期間を制御し、Cache-Control 値を変更しない場合は、オリジンキャッシュヘッダーの使用を選択します。

Cache-Control ヘッダーに関係なくオブジェクトがキャッシュに CloudFront 保持される最小時間と最大時間、および Cache-Control ヘッダーがオブジェクトに見つからないときにオブジェクトが CloudFront キャッシュに保持されるデフォルトの時間を指定するには、のカスタマイズを選択します。次に、[Minimum TTL (最小 TTL)]、[Default TTL (デフォルト TTL)]、[Maximum TTL (最大 TTL)] の各フィールドで値を指定します。

詳細については、「[コンテンツがキャッシュに保持される期間 \(有効期限\) の管理](#)」を参照してください。

最小 TTL

がオリジンに別のリクエスト CloudFront を送信してオブジェクトが更新されたかどうかを判断する前に、オブジェクトを CloudFront キャッシュに保持する最小時間を秒単位で指定します。

詳細については、「[コンテンツがキャッシュに保持される期間 \(有効期限\) の管理](#)」を参照してください。

最大 TTL

がオリジンに CloudFront クエリを実行してオブジェクトが更新されたかどうかを確認するまでに、オブジェクトを CloudFront キャッシュに保持する最大時間を秒単位で指定します。[Maximum TTL (最大 TTL)] に指定する値は、オリジンが Cache-Control max-age、Cache-Control s-maxage、Expires などの HTTP ヘッダーをオブジェクトに追加するときのみ適用されます。詳細については、「[コンテンツがキャッシュに保持される期間 \(有効期限\) の管理](#)」を参照してください。

[Maximum TTL (最大 TTL)] の値を指定するには、[Object Caching (オブジェクトキャッシュ)] 設定で [Customize (カスタマイズ)] オプションを選択する必要があります。

[Maximum TTL (最大 TTL)] のデフォルト値は 31,536,000 (秒)、つまり 1 年です。[Minimum TTL (最小 TTL)] または [Default TTL (デフォルト TTL)] の値を 31,536,000 (秒) より大きい値に変更する場合は、[Maximum TTL (最大 TTL)] のデフォルト値を [Default TTL (デフォルト TTL)] の値に変更します。

デフォルト TTL

がオリジンに別のリクエストを CloudFront 転送してオブジェクトが更新されたかどうかを判断する前に、オブジェクトを CloudFront キャッシュに保持するデフォルトの時間を秒単位で指定します。[Default TTL] (デフォルト TTL) に指定する値が適用されるのは、オリジンが Cache-Control max-age、Cache-Control s-maxage、Expires などの HTTP ヘッダーをオブジェクトに追加しない場合のみです。詳細については、「[コンテンツがキャッシュに保持される期間 \(有効期限\) の管理](#)」を参照してください。

[Default TTL (デフォルト TTL)] の値を指定するには、[Object Caching (オブジェクトキャッシュ)] 設定で [Customize (カスタマイズ)] オプションを選択する必要があります。

[Default TTL (デフォルト TTL)] のデフォルト値は 86,400 (秒)、つまり 1 日です。[Minimum TTL (最小 TTL)] の値を 86,400 (秒) より大きい値に変更する場合は、[Default TTL (デフォルト TTL)] のデフォルト値を [Minimum TTL (最小 TTL)] の値に変更します。

cookie の転送

Note

Amazon S3 オリジンの場合、このオプションは、ウェブサイトエンドポイントとして設定されているバケットにのみ適用されます。

Cookie CloudFront をオリジンサーバーに転送するかどうか、および転送する場合はどれを転送するかを指定します。選択された Cookie (Cookie の許可リスト) のみを転送するように選択した場合、Cookie 名を [許可リスト Cookie] フィールドに入力します。[All (すべて)] を選択した場合、アプリケーションで使用されている Cookie の数に関係なく、CloudFront はすべての Cookie を転送します。

Amazon S3 は、Cookie を処理しません。オリジンに Cookie を転送すると、キャッシュ能力が低下します。リクエストを Amazon S3 オリジンに転送するキャッシュ動作の場合は、[Forward Cookies (Cookie の転送)] で [None (なし)] を選択します。

オリジンへの Cookie の転送の詳細については、「[Cookie に基づくコンテンツのキャッシュ](#)」を参照してください。

許可リスト Cookie

Note

Amazon S3 オリジンの場合、このオプションは、ウェブサイトエンドポイントとして設定されているバケットにのみ適用されます。

転送 Cookie リストで許可リストを選択した場合は、許可リスト Cookie フィールドに、このキャッシュ動作のためにオリジンサーバーに転送する CloudFront Cookie の名前を入力します。各 Cookie 名を新しい行に入力します。

以下のワイルドカード文字を使用して Cookie 名を指定することができます。

- * は、Cookie 名に含まれる 0 個以上の文字に一致します。
- ? は、Cookie 名に含まれる 1 文字に一致します。

たとえば、オブジェクトに対するビューワーリクエストに、次の名前の Cookie が含まれているとします。

`userid_`*member-number*

member-number は、各ユーザーに割り当てられた一意の値です。メンバーごとにオブジェクトの個別のバージョンをキャッシュ CloudFront する場合。これを行うには、すべての Cookie をオリジンに転送しますが、ビューワーリクエストには CloudFront キャッシュしたくない Cookie が含まれています。または、Cookie 名として次の値を指定することもできます。これにより、CloudFront はで始まるすべての Cookie をオリジンに転送しますuserid_。

`userid_*`

キャッシュ動作ごとに許可リストに追加できる Cookie 名の現在の最大数、またはクォータの引き上げ (以前は制限と呼ばれていました) を要求する Cookie 名の最大数については、「[Cookie のクォータ \(従来のキャッシュ設定\)](#)」を参照してください。

クエリ文字列の転送とキャッシュ

CloudFront は、クエリ文字列パラメータの値に基づいて、コンテンツの異なるバージョンをキャッシュできます。以下のオプションのいずれかを選択します。

None (Improves Caching)

オリジンがクエリ文字列パラメータの値に関係なくオブジェクトの同じバージョンを返す場合、このオプションを選択します。これにより、がキャッシュからのリクエストを処理 CloudFront できる可能性が高くなり、パフォーマンスが向上し、オリジンの負荷が軽減されます。

すべて転送、許可リストに基づいてキャッシュ

オリジンサーバーが 1 つ以上のクエリ文字列パラメータに基づいてオブジェクトの異なるバージョンを返す場合、このオプションを選択します。次に、キャッシュの基準として使用するパラメータ CloudFront を [クエリ文字列の許可リスト](#) フィールドで指定します。

Forward all, cache based on all

オリジンサーバーがすべてのクエリ文字列パラメータについてオブジェクトの異なるバージョンを返す場合、このオプションを選択します。

パフォーマンスを向上する方法を含む、クエリ文字列パラメータに基づくキャッシュについては、「[クエリ文字列パラメータに基づくコンテンツのキャッシュ](#)」を参照してください。

クエリ文字列の許可リスト

すべて転送 を選択した場合、 の許可リストに基づいてキャッシュする場合は [クエリ文字列の転送とキャッシュ](#)、キャッシュの基準 CloudFront として使用するクエリ文字列パラメータを指定します。

スムーズストリーミング

Microsoft Smooth Streaming 形式のメディアファイルを配信するが、IIS サーバーがない場合は、[Yes (はい)] を選択します。

Microsoft Smooth Streaming 形式のメディアファイルを配信するためのオリジンとして使用する Microsoft IIS サーバーがある場合、または Smooth Streaming メディアファイルを配信しない場合は、[No (いいえ)] を選択します。

Note

[Yes (はい)] を指定した場合でも、他のコンテンツが [Path Pattern (パスパターン)] の値と一致すれば、このキャッシュ動作を使用してそのコンテンツを配信できます。

詳細については、「[Microsoft Smooth Streaming のビデオオンデマンドの設定](#)」を参照してください。

ビューワのアクセス制限 (署名付き URL または署名付き cookie の使用)

このキャッシュ動作の PathPattern に一致するオブジェクトのリクエストでパブリック URL を使用する場合、[No (いいえ)] を選択します。

このキャッシュ動作の PathPattern に一致するオブジェクトのリクエストで署名付き URL を使用する場合、[Yes (はい)] を選択します。次に、署名付き URL の作成に使用する AWS アカウントを指定します。このアカウントは信頼された署名者として知られています。

信頼された署名者の詳細については、「[署名付き URL と署名付き Cookie を作成できる署名者の指定](#)」を参照してください。

信頼された署名者

このキャッシュ動作の信頼された署名者として使用する AWS アカウントを選択します。

- Self: 信頼された署名者として AWS Management Console へのサインインに現在使用しているアカウントを使用します。現在、IAM ユーザーとしてサインインしている場合は、関連付けられた AWS アカウントを、信頼された署名者として追加します。
- Specify Accounts: 信頼された署名者のアカウント番号を [AWS Account Numbers] フィールドに入力します。

署名URLs、AWSアカウントに少なくとも 1 つのアクティブな CloudFrontキーペアが必要です。

Important

コンテンツの配信で既に使用されているディストリビューションを更新する場合は、オブジェクトの署名付き URL の生成を開始する準備ができたときにのみ、信頼された署名者を追加します。信頼された署名者がディストリビューションに追加されると、ユーザーは、このキャッシュ動作の PathPattern に一致するオブジェクトへのアクセスに、署名付き URL を使用する必要があります。

AWS アカウント番号

現在のアカウントに加えて、または現在のアカウントの代わりに、AWS アカウントを使用して署名付き URL を作成する場合、このフィールドの行ごとに 1 つの AWS アカウント番号を入力します。次の点に注意してください。

- 指定するアカウントに少なくとも 1 つのアクティブな CloudFront キーペアが必要です。詳細については、「[署名者のキーペアの作成](#)」を参照してください。
- IAM ユーザーの CloudFront キーペアを作成できないため、IAM ユーザーを信頼された署名者として使用することはできません。
- アカウントの AWS アカウント番号を取得する方法については、「Amazon Web Services 全般のリファレンス」の「[AWS アカウント ID](#)」を参照してください。
- 現在のアカウントのアカウント番号を入力すると、CloudFront は自動的にセルフチェックボックスをオンにし、アカウント番号リストから AWS アカウント番号を削除します。

オブジェクトを自動的に圧縮する

ビューワーが圧縮コンテンツをサポートしているときに、特定のタイプのファイル CloudFront を自動的に圧縮する場合は、はいを選択します。CloudFront がコンテンツを圧縮する際、ファイルが小さいためダウンロードはより高速に行われ、ウェブページは高速にレンダリングされます。詳細については、「[圧縮ファイルの供給](#)」を参照してください。

CloudFront イベント

以下の CloudFront イベントが 1 つ以上発生した場合に Lambda 関数を実行するように選択できます。

- がビューワーからリクエスト CloudFront を受け取る場合 (ビューワーリクエスト)
- がリクエストをオリジン CloudFront に転送する前に (オリジンリクエスト)
- がオリジンからレスポンス CloudFront を受信する場合 (オリジンレスポンス)
- がビューワーにレスポンスを CloudFront 返す前 (ビューワーレスポンス)

詳細については、「[Lambda@Edge 関数のトリガーに使用する CloudFront イベントを決定する方法](#)」を参照してください。

Lambda 関数の ARN

トリガーを追加する Lambda 関数の Amazon リソースネーム (ARN) を指定します。関数の ARN を取得する方法については、「[CloudFront 「コンソールを使用してトリガーを追加する」](#)」の手順のステップ 1 を参照してください。

ディストリビューションの設定

以下の値はディストリビューション全体に適用されます。

価格クラス

CloudFront サービスに支払う上限価格に対応する価格クラスを選択します。デフォルトでは、はすべての CloudFront リージョンのエッジロケーションからオブジェクト CloudFront を提供します。

料金クラスの詳細と、選択した料金クラスがディストリビューション CloudFront のパフォーマンスにどのように影響するかについては、「」を参照してください [CloudFront ディストリビューションの価格クラスの選択](#)。CloudFront 料金クラスが CloudFront リージョンにどのようにマッピングされるかなど、料金の詳細については、「[Amazon CloudFront 料金表](#)」を参照してください。

AWS WAF ウェブ ACL

ウェブアプリケーションと APIs を保護してリクエストがサーバーに到達する前にブロックできるようにするウェブアプリケーションファイアウォール [AWS WAF](#) である を使用してディストリビューション CloudFront を保護できます。CloudFront ディストリビューションを作成または編集 [新しいディストリビューションで AWS WAF を有効にする](#) するときを実行できます。

オプションで、後で AWS WAF コンソール (<https://console.aws.amazon.com/wafv2/>) から、アプリケーションに固有の他の脅威に対する追加のセキュリティ保護を設定できます。

AWS WAF の詳細については、[AWS WAF 開発者ガイド](#)を参照してください。

代替ドメイン名 (CNAME)

オプション。ディストリビューションの作成時に が CloudFront 割り当てるドメイン名の代わりに、オブジェクトURLs に使用する 1 つ以上のドメイン名を指定します。ドメイン名を所有しているか、あるいはこのドメイン名を使用する許可があることが必要です。この許可は、SSL/TLS 証明書を追加することで検証します。

たとえば、次のオブジェクトの URL があります。

```
/images/image.jpg
```

この URL を次のように表示します。

```
https://www.example.com/images/image.jpg
```

次のようには指定しません。

```
https://d1111111abcdef8.cloudfront.net/images/image.jpg
```

この場合、www.example.com の CNAME を追加します。

⚠ Important

ディストリビューションに `www.example.com` の CNAME を追加する場合、さらに以下を実行する必要があります。

- DNS サービスを使用して CNAME レコードを作成 (または更新) して、`www.example.com` のクエリを `d111111abcdef8.cloudfront.net` にルーティングします。
- ディストリビューションに追加するドメイン名 (CNAME) を対象とする信頼された認証機関 (CA) CloudFront から 証明書を追加して、ドメイン名を使用する権限を検証します。

ドメインの DNS サービスプロバイダーがある CNAME レコードを作成する許可が必要です。通常、これはドメインを所有している、またはドメイン所有者向けにアプリケーションを開発していることを示します。

ディストリビューションに対して作成できる代替ドメイン名の現在の最大数、またはクォータの引き上げを要求する代替ドメイン名の最大数については、「[ディストリビューションの一般的なクォータ](#)」を参照してください。

代替ドメイン名の詳細については、「[代替ドメイン名 \(CNAME\) を追加することによるカスタム URL の使用](#)」を参照してください。CloudFront URLs 「」を参照してくださいの[ファイルの URL 形式のカスタマイズ CloudFront](#)。

SSL 証明書

ディストリビューションで使用する代替ドメイン名を指定した場合は、[Custom SSL Certificate (カスタム SSL 証明書)] を選択してこの代替ドメイン名を使用する許可を検証し、これを対象とする証明書を選択します。ビューワーが HTTPS を使用してオブジェクトにアクセスするようにする場合は、それをサポートしている設定を選択します。

i Note

カスタム SSL 証明書を指定する前に、有効な代替ドメイン名を指定する必要があります。詳細については、「[代替ドメイン名を使用するための要件](#)」および「[代替ドメイン名と HTTPS の使用](#)」を参照してください。

- デフォルトの CloudFront 証明書 (*.cloudfront.net) – などのオブジェクトの URLs で CloudFront ドメイン名を使用する場合は、このオプションを選択します `https://d111111abcdef8.cloudfront.net/image1.jpg`。
- 独自 SSL 証明書 – オブジェクトの URL で独自のドメイン名を代替ドメイン名として使用する場合 (`https://example.com/image1.jpg` など)、このオプションを選択します。次に、代替ドメイン名を対象とする証明書を使用するために選択します。証明書のリストには、次のいずれかを含めることができます。
 - AWS Certificate Manager から提供される証明書
 - サードパーティー認証機関から購入して ACM にアップロードした証明書
 - サードパーティー認証機関から購入して IAM 証明書ストアにアップロードした証明書

この設定を選択した場合、オブジェクト URL でのみ代替ドメイン名を使用することをお勧めします (`https://example.com/logo.jpg`)。CloudFront ディストリビューションドメイン名 (`https://d111111abcdef8.cloudfront.net/logo.jpg`) を使用し、クライアントが SNI をサポートしていない古いビューワーを使用している場合、ビューワーの応答方法は、サポートされるクライアントに選択した値によって異なります。

- すべてのクライアント: CloudFront ドメイン名が SSL/TLS 証明書のドメイン名と一致しないため、ビューワーに警告が表示されます。
- Server Name Indication (SNI) をサポートするクライアントのみ: オブジェクトを返さずにビューワーとの接続を CloudFront ドロップします。

独自 SSL クライアントのサポート

ディストリビューションに 1 つ以上の代替ドメイン名とカスタム SSL 証明書を指定した場合は、HTTPS リクエスト CloudFront の処理方法を選択します。

- [Clients that Support Server Name Indication (SNI) - (Recommended) (Server Name Indication (SNI) をサポートするクライアント - (推奨))] – この設定では、ほとんどすべての最新のウェブブラウザとクライアントは、SNI をサポートしているため、ディストリビューションに接続できます。ただし、一部のビューワーは SNI をサポートしていない古いウェブブラウザやクライアントを使用している可能性があります。つまり、一部のビューワーはディストリビューションに接続できません。

CloudFront API を使用してこの設定を適用するには、`SSLSupportMethod` フィールド `sni-only` を指定します。AWS CloudFormation では、このフィールドの名前は `SslSupportMethod` です (大文字と小文字の変更に注意してください)。

- [Legacy Clients Support (レガシークライアントサポート)] – この設定では、SNI をサポートしていない古いウェブブラウザとクライアントはディストリビューションに接続できます。ただし、この設定では、追加の月額料金が発生します。正確な料金については、[Amazon CloudFront 料金表](#) ページに移動し、専用 IP カスタム SSL のページを検索します。

CloudFront API を使用してこの設定を適用するには、`SSLSupportMethod` フィールドで `vip` を指定します。AWS CloudFormation では、このフィールドの名前は `SslSupportMethod` です (大文字と小文字の変更に注意してください)。

詳細については、「[が HTTPS リクエスト CloudFront を処理する方法の選択](#)」を参照してください。

セキュリティポリシー

ビューワー (クライアント) との HTTPS 接続 CloudFront に使用するセキュリティポリシーを指定します。セキュリティポリシーは 2 通りの設定を決定します。

- がビューワーとの通信 CloudFront に使用する最小 SSL/TLS プロトコル。
- がビューワーに返すコンテンツを暗号化するために CloudFront 使用できる暗号。

セキュリティポリシー (各ポリシーに含まれるプロトコルや暗号など) の詳細については、「[ビューワーとの間でサポートされているプロトコルと暗号 CloudFront](#)」を参照してください。

使用できるセキュリティポリシーは、SSL 証明書とカスタム SSL クライアントサポート (CloudFront API `SSLSupportMethod` では `CloudFrontDefaultCertificate` と呼ばれます) に指定する値によって異なります。

- SSL 証明書がデフォルト CloudFront 証明書 (`*.cloudfront.net`) の場合 (`CloudFrontDefaultCertificate` が `API true` にある場合)、 はセキュリティポリシー `CloudFront` を自動的に TLSv1 に設定します。
- [SSL Certificate] (SSL 証明書) が [Custom SSL Certificate (example.com)] (カスタム SSL 証明書) で、かつ [Custom SSL Client Support] (カスタム SSL クライアントサポート) が [Clients that Support Server Name Indication (SNI) - (Recommended)] (Server Name Indication (SNI) をサポートするクライアント) - (推奨)、 (つまり API で `CloudFrontDefaultCertificate` が `false` かつ `SSLSupportMethod` が `sni-only` である場合) 次のセキュリティポリシーから選択できません。
 - `TLSv1.2_2021`

- TLSv1.2_2019
- TLSv1.2_2018
- TLSv1.1_2016
- TLSv1_2016
- TLSv1
- [SSL Certificate] (SSL 証明書) が [Custom SSL Certificate (example.com)] (カスタム SSL 証明書) で、かつ [Custom SSL Client Support] (カスタム SSL クライアントサポート) が [Legacy Clients Support] (レガシークライアントサポート)、(つまり API で CloudFrontDefaultCertificate が false かつ SSLSupportMethod が vip である場合) 次のセキュリティポリシーから選択できません。
 - TLSv1
 - SSLv3

この設定では、TLSv1.2_2021、TLSv1.2_2019、TLSv1.2_2018、TLSv1.1_2016、および TLSv1_2016 セキュリティポリシーは、CloudFront コンソールまたは API では使用できません。これらのセキュリティポリシーのいずれかを使用する場合は、以下のオプションを指定できます。

- ディストリビューションが専用 IP アドレスを使用したレガシークライアントサポートを必要としているかどうかを評価します。ビューワーが [\[Server Name Indication \(SNI\)\]](#) (Server Name Indication (SNI)) をサポートしている場合は、ディストリビューションの [Custom SSL Client Support] (カスタム SSL クライアントサポート) の設定を [Clients that Support Server Name Indication (SNI)] (Server Name Indication (SNI)) をサポートするクライアントに更新 (API で SSLSupportMethod を sni-only に設定する) することをお勧めします。これにより、使用可能な TLS セキュリティポリシーのいずれかを使用でき、料金も削減できます CloudFront。
- 専用 IP アドレスを使用するレガシークライアントのサポートを維持する必要がある場合は、[AWS サポートセンター](#)でケースを作成することによって、他の TLS セキュリティポリシー (TLSv1.2_2021、TLSv1.2_2019、TLSv1.2_2018、TLSv1.1_2016、または TLSv1_2016) のいずれかをリクエストできます。

Note

AWS サポートに連絡してこの変更をリクエストする前に、以下の点を考慮してください。

- これらのセキュリティポリシー (TLSv1.2_2021、TLSv1.2_2019、TLSv1.2_2018、TLSv1.1_2016、または TLSv1_2016) のいずれかをレガシークライアントサポートディストリビューション

に追加すると、セキュリティポリシーが AWS アカウントのすべてのレガシークライアントサポートディストリビューションに対する SNI 以外のすべてのビューワーリクエストに適用されます。ただし、ビューワーが、レガシークライアントサポートを使用するディストリビューションに SNI リクエストを送信した場合は、そのディストリビューションのセキュリティポリシーが適用されます。AWS アカウントのすべてのレガシークライアントサポートディストリビューションに送信されたすべてのビューワーリクエストに望ましいセキュリティポリシーが適用されることを確実にするには、望ましいセキュリティポリシーをディストリビューションごとに個別に追加します。

- 定義上、新しいセキュリティポリシーは、古いセキュリティポリシーと同じ暗号やプロトコルをサポートしません。たとえば、ディストリビューションのセキュリティポリシーを TLSv1 から TLSv1.1_2016 にアップグレードすることを選択した場合、そのディストリビューションは DES-CBC3-SHA 暗号をサポートしなくなります。各セキュリティポリシーがサポートする暗号とプロトコルの詳細については、「[ビューワーとの間でサポートされているプロトコルと暗号 CloudFront](#)」を参照してください。

サポートされる HTTP バージョン

ビューワーがと通信するときにディストリビューションがサポートする HTTP バージョンを選択します CloudFront。

ビューワーと CloudFront が HTTP/2 を使用するには、ビューワーが TLSv1.2 以降をサポートしている必要があります。Server Name Indication (SNI) は HTTP/2 経由の gRPC のネイティブサポートを提供し CloudFront ません。

ビューワーと HTTP/3 CloudFront を使用するには、ビューワーが TLSv1.3 と Server Name Indication (SNI) をサポートしている必要があります。は HTTP/3 接続の移行 CloudFront をサポートしているため、ビューワーは接続を失うことなくネットワークを切り替えることができます。接続の移行の詳細については、RFC 9000 で「[Connection Migration](#)」を参照してください。

Note

サポートされる TLSv1.3 暗号の詳細については、「[ビューワーとの間でサポートされているプロトコルと暗号 CloudFront](#)」を参照してください。

デフォルトのルートオブジェクト

オプション。ビューワーがディストリビューション内のオブジェクト (`index.html`) ではなくディストリビューションのルート URL () をリクエストするとき、CloudFront オリジンにリクエストするオブジェクト (例: `https://www.example.com/`) `https://www.example.com/product-description.html`。デフォルトのルートオブジェクトを指定すると、ディストリビューションのコンテンツが公開されなくなります。

名前の最大長は 255 文字です。名前には以下の文字を含めることができます。

- A ~ Z、a ~ z
- 0-9
- `_ - . * $ / ~ "`
- `&` (& として受け渡しされます)

デフォルトのルートオブジェクトを指定する場合、オブジェクト名のみを指定します (例: `index.html`)。オブジェクト名の前に `/` を追加しないでください。

詳細については、「[デフォルトのルートオブジェクトの指定](#)」を参照してください。

ログ記録

オブジェクトの各リクエストに関する情報をログ CloudFront に記録し、ログファイルを Amazon S3 バケットに保存するかどうか。ログ作成はいつでも有効または無効にできます。ログ作成を有効にしても追加料金はかかりませんが、Amazon S3 バケットにおけるファイルの保存とファイルへのアクセスについては通常の Amazon S3 料金が発生します。ログの削除はいつでも行うことができます。CloudFront アクセスログの詳細については、「[標準ログ \(アクセスログ\) の設定および使用](#)」を参照してください。

ログ用のバケット

ログ記録に On を選択した場合、アクセスログ CloudFront を保存する Amazon S3 バケット。例えば、`myLogs-DOC-EXAMPLE-BUCKET.s3.amazonaws.com`。

Important

は、これらのリージョンのバケットに標準ログを配信しないため、以下のリージョンでは Amazon S3 バケットを選択しないでください。CloudFront

- アフリカ (ケープタウン)
- アジアパシフィック (香港)
- アジアパシフィック (ハイデラバード)
- アジアパシフィック (ジャカルタ)
- アジアパシフィック (メルボルン)
- カナダ (中部)
- 欧州 (ミラノ)
- 欧州 (スペイン)
- 欧州 (チューリッヒ)
- イスラエル (テルアビブ)
- 中東 (バーレーン)
- 中東 (アラブ首長国連邦)

ログ記録を有効にすると、はオブジェクトに対する各エンドユーザーリクエストに関する情報を CloudFront 記録し、指定した Amazon S3 バケットにファイルを保存します。ログ作成はいつでも有効または無効にできます。CloudFront アクセスログの詳細については、「[標準ログ \(アクセスログ\) の設定および使用](#)」を参照してください。

Note

Amazon S3 バケット ACL を取得して更新するための権限が必要です。また、バケットの S3 ACL から FULL_CONTROL が付与される必要があります。これにより CloudFront、はawslogsdeliveryアカウントに対して、ログファイルをバケットに保存するためのアクセス許可を付与できます。詳細については、「[標準ログ記録の設定およびログファイルへのアクセスに必要なアクセス許可](#)」を参照してください。

ログのプレフィックス

オプション。[Logging] で [On] を選択した場合、このディストリビューションのアクセスログファイル名の先頭に CloudFront が追加する文字列 (ある場合) を指定します (例: exampleprefix/)。末尾のスラッシュ (/) はオプションですが、ログファイルの参照を容易にするためにこれを使用することをお勧めします。CloudFront アクセスログの詳細については、「」を参照してください [標準ログ \(アクセスログ\) の設定および使用](#)。

cookie のログ作成

アクセスログに Cookie CloudFront を含める場合は、`Cookie Logging` を選択します。Cookie をログに含めるように選択した場合、CloudFront はすべての Cookie をログに記録します。このディストリビューションのキャッシュ動作がどのように構成されているか (オリジンにすべての Cookie を転送するか、Cookie を転送しないか、指定された一連の Cookie を転送するか) は関係ありません。

Amazon S3 は Cookie を処理しません。したがって、ディストリビューションに Amazon EC2 または他のカスタムオリジンも含まれていない限り、[Cookie Logging (Cookie ログ記録)] の値に [Off (オフ)] を選択することをお勧めします。

Cookie の詳細については、「[Cookie に基づくコンテンツのキャッシュ](#)」を参照してください。

IPv6 を有効にする

IPv6 は、IP プロトコルの新しいバージョンです。これは最終的に IPv4 に置き換わり、より大きなアドレス空間を使用します。CloudFront 常に IPv4 リクエストに応答します。IPv4 IP アドレス (192.0.2.44 など) からのリクエストと IPv6 アドレス (2001:0db8:85a3::8a2e:0370:7334 など) からのリクエストに CloudFront 応答する場合は、IPv6 を有効にするを選択します。

一般的に、IPv6 ネットワークのユーザーがいてコンテンツにアクセスする場合は、IPv6 を有効にする必要があります。ただし、コンテンツへのアクセスを制限するために署名付き URL または署名付き Cookie を使用していて、コンテンツへのアクセスが可能な IP アドレスを制限する IP アドレスを含む `IpAddress` パラメータを使用している場合、IPv6 は有効にしません。一部のコンテンツへのアクセスを IP アドレスで制限し、他のコンテンツへのアクセスを制限しない場合 (またはアクセスを制限するが IP アドレスでは行わない場合)、2 つのディストリビューションを作成します。カスタムポリシーを使用して署名付き URL を作成する方法については、[カスタムポリシーを使用する署名付き URL の作成](#)を参照してください。カスタムポリシーを使用して署名付き Cookie を作成する方法については、[カスタムポリシーを使用する署名付き Cookie の設定](#)を参照してください。

Route 53 エイリアスリソースレコードセットを使用してトラフィックを CloudFront ディストリビューションにルーティングする場合は、以下の両方に該当するときに 2 つ目のエイリアスリソースレコードセットを作成する必要があります。

- ディストリビューションで IPv6 を有効にする
- オブジェクトの URL で代替ドメイン名を使用している

詳細については、「[Amazon Route 53 デベロッパーガイド](#)」の「[ドメイン名を使用したトラフィックの Amazon CloudFront ディストリビューションへのルーティング](#)」を参照してください。

Route 53 または別の DNS サービスで CNAME リソースレコードセットを作成した場合、変更を行う必要はありません。ビューワーリクエストの IP アドレスフォーマットに関係なく、CNAME レコードはトラフィックをディストリビューションにルーティングします。

IPv6 と CloudFront アクセスログを有効にすると、`c-ip`列には IPv4 および IPv6 形式の値が含まれます。詳細については、「[標準ログ \(アクセスログ\) の設定および使用](#)」を参照してください。

Note

高いカスタマー可用性を維持するために、`is-ip-ipv4` は、IPv4 がより良いユーザーエクスペリエンスを提供するとデータが示す場合、IPv4 を使用してビューワーリクエスト CloudFront に対応します。IPv6 で処理 CloudFront されているリクエストの割合を調べるには、ディストリビューションの CloudFront ログ記録を有効にし、リクエストを行ったビューワーの IP アドレスを含む `c-ip`列を解析します。この割合 (%) は時間とともに大きくなりますが、IPv6 は世界中のすべてのビューワーネットワークでサポートされているわけではないため、過半数のトラフィックになることはないでしょう。ビューワーネットワークによっては IPv6 が十分にサポートされていますが、IPv6 をまったくサポートしないものもあります (ビューワーネットワークは、ホームインターネットやワイヤレスキャリアに似ています)。

IPv6 のサポートの詳細については、「[CloudFront に関するよくある質問](#)」を参照してください。アクセスログを有効にする方法の詳細については、[ログ記録](#)、[ログ用のバケット](#)、および [ログのプレフィックス](#) のフィールドを参照してください。

コメント

(オプション)。ディストリビューションを作成するときに、最大で 128 文字のコメントを含めることができます。コメントの更新はいつでも行うことができます。

ディストリビューションの状態

ディストリビューションがデプロイされた後にディストリビューションを有効または無効のどちらにするかを示します。

- Enabled (有効): ディストリビューションが完全にデプロイされると、ディストリビューションのドメイン名を使用するリンクをデプロイでき、ユーザーはコンテンツを取得できます。ディストリビューションを有効にすると、`is-ip-ipv4`はそのディストリビューションに関連付けられたドメイン名を使用するコンテンツに対するエンドユーザーリクエスト CloudFront を受け入れて処理します。

CloudFront ディストリビューションを作成、変更、または削除すると、変更が CloudFront データベースに反映されるまでに時間がかかります。変更直後に出したディストリビューションに関する

情報のリクエストには、変更が反映されていない可能性があります。通常、伝達は数分以内で完了しますが、システムの高負荷またはネットワークパーティションによっては、それより時間がかかる可能性があります。

- Disabled (無効): デイストリビューションがデプロイされていて、使用準備ができていても、ユーザーはデイストリビューションを使用できません。デイストリビューションを無効にするたびに、CloudFront はそのデイストリビューションに関連付けられたドメイン名を使用するエンドユーザーリクエストを受け入れません。(デイストリビューションの構成を更新することで) デイストリビューションを無効から有効に切り替えるまで、誰もデイストリビューションを使用できません。

デイストリビューションの無効と有効は何度でも切り替えることができます。デイストリビューションの構成を更新するプロセスに従います。詳細については、「[デイストリビューションの更新](#)」を参照してください。

カスタムエラーページとエラーキャッシュ

Amazon S3 またはカスタムオリジンが HTTP 4xx または 5xx ステータスコードを に返すときに、 がオブジェクトをビューワーに CloudFront 返すことができます (HTML ファイルなど) CloudFront。オリジンからのエラーレスポンスまたはカスタムエラーページが CloudFront エッジキャッシュにキャッシュされる期間を指定することもできます。詳細については、「[特定の HTTP ステータスコードに対応するカスタムエラーページの作成](#)」を参照してください。

Note

以下の値は [Create Distribution] ウィザードに含まれていないため、デイストリビューションを更新するときのみ、カスタムエラーページを構成することができます。

HTTP エラーコード

カスタムエラーページ CloudFront を返す HTTP ステータスコード。が CloudFront キャッシュする HTTP ステータスコードの一部または全部について、カスタムエラーページを返す CloudFront ように を設定できます。

Error caching minimum TTL (seconds) (エラーキャッシュ最小 TTL (秒))

オリジンサーバーからのエラーレスポンスをキャッシュ CloudFront する最小時間。

Response page path (レスポンスページのパス)

オリジンがエラーコードに指定した HTTP ステータスコード (403 など/`4xx-errors/403-forbidden.html`) を返すときにビューワーに返す CloudFront カスタムエラーページへのパス (など)。オブジェクトとカスタムエラーページを別の場所に保存する場合は、次の状況に該当するときに適用されるキャッシュ動作をディストリビューションに組み込む必要があります。

- [Path Pattern (パスパターン)] の値が、カスタムエラーメッセージのパスと一致している。たとえば、4xx エラーのカスタムエラーページを `/4xx-errors` というディレクトリの Amazon S3 バケットに保存したとします。このとき、パスパターンによってカスタムエラーページのリクエストのルーティング先である場所に対するキャッシュ動作を、ディストリビューションに組み込む必要があります (例: `/4xx-errors/*`)。
- [Origin (オリジン)] の値は、カスタムエラーページが含まれているオリジンの [Origin ID (オリジン ID)] の値を指定しています。

HTTP レスポンスコード

カスタムエラーページとともにビューワーに返す CloudFront の HTTP ステータスコード。

地理的制限

選択した国のユーザーがコンテンツにアクセスできないようにする必要がある場合は、許可リストまたはブロックリストを使用して CloudFront ディストリビューションを設定できます。地理的制限の設定には追加料金が発生しません。詳細については、「[コンテンツの地理的ディストリビューションの制限](#)」を参照してください。

コンソール CloudFront に表示される値

新しいディストリビューションを作成するか、既存のディストリビューションを更新すると、は CloudFront コンソールに次の情報 CloudFront を表示します。

Note

アクティブな信頼された署名者、つまりアクティブな CloudFront キーペアを持ち、有効な署名 URL の作成に使用できる AWS アカウントは、現在 CloudFront コンソールに表示されません。

ディストリビューション ID

CloudFront API を使用してディストリビューションに対してアクションを実行する場合、ディストリビューション ID を使用して、使用するディストリビューションを指定します。例えば、です EFDVBD6EXAMPLE。ディストリビューションのディストリビューション ID を変更することはできません。

デプロイとステータス

ディストリビューションをデプロイすると、最終更新日時列の下にデプロイステータスが表示されます。ディストリビューションのデプロイが完了するのを待ち、ステータス列に有効と表示されることを確認します。詳細については、「[ディストリビューションの状態](#)」を参照してください。

最終更新日時

ディストリビューションが最後に変更された日時。ISO 8601 形式が使用されます (例: 2012-05-19T19:37:58Z)。詳しくは、「<https://www.w3.org/TR/NOTE-datetime>」を参照してください。

ドメイン名

オブジェクトへのリンク内で、ディストリビューションのドメイン名を使用します。たとえば、ディストリビューションのドメイン名が `d111111abcdef8.cloudfront.net` の場合、`/images/image.jpg` へのリンクは `https://d111111abcdef8.cloudfront.net/images/image.jpg` になります。ディストリビューションの CloudFront ドメイン名を変更することはできません。オブジェクトへのリンクの CloudFront URLs 「」を参照してくださいの [ファイルの URL 形式のカスタマイズ CloudFront](#)。

1 つ以上の代替ドメイン名 (CNAMEs を指定した場合は、ドメイン名を使用する代わりに、オブジェクトへのリンクに独自の CloudFront ドメイン名を使用できます。CNAME の詳細については、「[代替ドメイン名 \(CNAME\)](#)」を参照してください。

Note

CloudFront ドメイン名は一意です。ディストリビューションのドメイン名は以前のディストリビューションで使用されておらず、今後の別のディストリビューションでも再利用されません。

ディストリビューションのテスト

ディストリビューションを作成したら、はオリジンサーバーの場所 CloudFront を認識し、ディストリビューションに関連付けられたドメイン名がわかっています。CloudFront ドメイン名を使用してオブジェクトへのリンクを作成すると、CloudFront はオブジェクトをウェブページまたはアプリケーションに配信します。

Note

リンクのテストを行う前に、ディストリビューションのステータスが [Deployed (デプロイ済み)] に変わるまで待つ必要があります。

ウェブディストリビューション内のオブジェクトへのリンクを作成するには

1. 以下の HTML コードを新しいファイルにコピーし、*domain-name* をディストリビューションのドメイン名で置き換え、*object-name* をオブジェクトの名前で置き換えます。

```
<html>
<head>My CloudFront Test</head>
<body>
<p>My text content goes here.</p>
<p>
</html>
```

たとえば、ドメイン名が `d111111abcdef8.cloudfront.net` で、オブジェクトが `image.jpg` の場合、リンクの URL は次のようになります。

```
https://d111111abcdef8.cloudfront.net/image.jpg.
```

オブジェクトがオリジンサーバー内のフォルダにある場合は、そのフォルダも URL に含める必要があります。たとえば、`image.jpg` がオリジンサーバーのイメージフォルダにある場合、URL は次のようになります。

```
https://d111111abcdef8.cloudfront.net/images/image.jpg
```

2. HTML コードを、html ファイル名拡張子の付いたファイルに保存します。
3. ブラウザでウェブページを開いて、オブジェクトが見られるかどうかを確認します。

ブラウザは、オブジェクトを供給するのに適切である CloudFront と判断したエッジロケーションから提供されたイメージファイルが埋め込まれたページを返します。

ディストリビューションの更新

CloudFront コンソールでは、AWSアカウントに関連付けられている CloudFront ディストリビューションを表示したり、ディストリビューションの設定を表示したり、ほとんどの設定を更新したりできます。設定に変更を加えても、ディストリビューションが AWS エッジロケーションに伝達されるまで有効にならないことに注意してください。

CloudFront ディストリビューションを更新するには

1. にサインインAWS Management Consoleし、 で CloudFront コンソールを開きます<https://console.aws.amazon.com/cloudfront/v4/home>。
2. ディストリビューションの ID を選択します。リストには、CloudFront コンソールへのサインインに使用したAWSアカウントに関連付けられたすべてのディストリビューションが含まれます。
3. ディストリビューションの設定を編集するには、[Distribution Settings (ディストリビューションの設定)] タブを選択します。
4. 全般設定を更新するには、[Edit (編集)] を選択します。それ以外の場合は、更新する設定のタブ ([Origins (オリジン)] または [Behaviors (動作)]) を選択します。
5. 更新してから変更内容を保存するには、[Yes, Edit (はい、編集します)] を選択します。フィールドの詳細については、以下の各トピックを参照してください。
 - 全般設定: [ディストリビューションの設定](#)
 - オリジンの設定: [オリジンの設定](#)
 - キャッシュ動作の設定: [キャッシュ動作の設定](#)
6. ディストリビューションのオリジンを削除する場合は、次の操作を行います。
 - a. [Behaviors (動作)] を選択し、そのオリジンに関連付けられているデフォルトのキャッシュ動作が別のオリジンに移動済みであることを確認します。
 - b. [Origins (オリジン)] で、オリジンを選択します。
 - c. [Delete] (削除) を選択します。

CloudFront API を使用してディストリビューションを更新することもできます。

- デистриビューションを更新するには、「Amazon CloudFront API リファレンス」の[UpdateDistribution](#)「」を参照してください。

Important

デистриビューションを更新する際は、デистриビューションの作成時には必要でないいくつかの追加フィールドが必須であることに注意してください。デистриビューションを作成または更新するときの必須フィールドの概要については、「[デистриビューションの作成および更新で必須である API フィールド](#)」を参照してください。CloudFront API を使用してデистриビューションを更新するときに必須フィールドがすべて含まれるようにするには、「Amazon CloudFront API リファレンス」の[UpdateDistribution](#)「」で説明されている手順に従います。

デистриビューション設定に変更を保存すると、はすべてのエッジロケーションに変更を伝達し CloudFront 始めます。連続した設定の変更は、それぞれの順序で伝播されます。エッジロケーションで構成が更新されるまで、CloudFront は以前の構成に基づいて、そのエッジロケーションからコンテンツを引き続き供給します。エッジロケーションで構成が更新されると、CloudFront は新しい構成に基づいて、そのエッジロケーションからコンテンツを直ちに供給し始めます。

変更は、すべてのエッジロケーションにすぐに伝達されるわけではありません。伝達が完了すると、デистриビューションのステータスが からデプロイ済み InProgress に変わります。CloudFront が変更を伝達している間、特定のエッジロケーションが以前の設定または新しい設定に基づいてコンテンツを提供しているかどうかを判断することはできません。

Amazon CloudFront デистриビューションのタグ付け

タグは、AWS リソースを特定し、整理するのに使用できる単語または語句です。各リソースには複数のタグを追加でき、各タグにはユーザーが定義したキーと値が含まれています。たとえば、キーが "domain" で値が "example.com" というタグを付けることができます。追加したタグに基づいて、リソースを検索したりフィルタ処理したりできます。

以下の 2 つの例では、CloudFront でタグを使用する便利な方法を示しています。

- タグを使用して、さまざまなカテゴリの請求情報を追跡する。デистриビューションまたは他の AWS リソース (Amazon EC2 インスタンスや Amazon S3 バケットなど) に CloudFront タグを適用してタグをアクティブ化すると、は、アクティブなタグ別に使用量とコストを集計したコスト配分レポートをカンマ区切り値 (CSV) ファイルとして AWS 生成します。自社のカテゴリ たとえばコ

ストセンター、アプリケーション名、所有者を表すタグを適用すると、複数のサービスにわたってコストを分類することができます。タグを使用したコスト配分の詳細については、「AWS Billing ユーザーガイド」の「[コスト配分タグの使用](#)」を参照してください。

- タグを使用して、CloudFront デイストリビューションにタグベースのアクセス許可を適用します。詳細については、「[での ABAC CloudFront](#)」を参照してください。

次の点に注意してください。

- デイストリビューションをタグ付けできますが、オリジンアクセスアイデンティティや無効化をタグ付けすることはできません。
- [タグエディタ](#)と[リソースグループ](#)は、現在ではサポートされていません CloudFront。

デイストリビューションに追加できるタグの数の現在の最大制限については、「[クォータ](#)」を参照してください。クォータ (以前は制限と呼ばれていました) の引き上げをリクエストするには、AWS サポートセンターで[ケースを作成](#)してください。

CloudFront API、SDKsAWS CLI、および [および](#) を使用して、リソースにタグを適用することもできます AWS Tools for Windows PowerShell。詳細については、次のドキュメントを参照してください。

- CloudFront API – Amazon CloudFront API リファレンスの以下のオペレーションを参照してください。
 - [ListTagsForResource](#)
 - [TagResource](#)
 - [UntagResource](#)
- AWS CLI – AWS CLI コマンドリファレンスの「[cloudfront](#)」を参照
- SDK – [AWS ドキュメント](#) ページの該当する SDK ドキュメントを参照
- Tools for Windows PowerShell – コマンドレットリファレンスの「[Amazon CloudFront](#)」を参照してください。 [AWS Tools for PowerShell](#)

トピック

- [タグの制限](#)
- [デイストリビューションに対するタグの追加、編集、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 – 128 文字 (Unicode)
- 値の最大長 – 256 文字 (Unicode)
- キーと値の有効な値 – a~z、A~Z、0~9、スペース、特殊文字 (_ . : / = + - @)
- タグのキーと値は大文字と小文字が区別されます。
- aws: をキーのプレフィックスとして使用しないでください。AWS 用に予約済みです。

ディストリビューションに対するタグの追加、編集、削除

次の手順では、CloudFront コンソールでディストリビューションのタグを追加、編集、削除する方法について説明します。

ディストリビューションにタグを追加、編集、または削除するには

1. にサインインAWS Management Consoleし、 で CloudFront コンソールを開きます <https://console.aws.amazon.com/cloudfront/v4/home>。
2. 更新するディストリビューションの ID を選択します。
3. [タグ] タブを選択します。
4. [タグを管理] を選択します。
5. [Manage tags] (タグの管理) ページで、次の操作を実行できます。
 - タグを追加するには、キーと、タグの値 (オプション) を入力します。さらにタグを追加するには、[Add new tag] (新規タグの追加) ボタンをクリックします。
 - タグを編集するには、タグのキーまたはその値、あるいはその両方を変更します。タグの値を削除することはできますが、キーが必要です。
 - タグを削除するには、タグの横にある [Remove] (削除) ボタンを選択します。
6. [変更の保存] をクリックします。

ディストリビューションを削除する

ディストリビューションが不要になった場合は、CloudFront コンソールまたは CloudFront API を使用してディストリビューションを削除できます。

ディストリビューションを削除する前に、これを無効にする必要があることに注意してください。そのためには、ディストリビューションを更新するアクセス許可が必要です。

Note

代替ドメイン名が関連付けられているディストリビューションを無効にすると、別のディストリビューションに同じドメイン (*.www.example.com など) と一致するワイルドカード (*) を持つ代替ドメイン名がある場合でも、はそのドメイン名 (example.com など) のトラフィックの受け入れを CloudFront 停止します。

CloudFront ディストリビューションを削除するには

1. にサインインAWS Management Consoleし、 で CloudFront コンソールを開きます<https://console.aws.amazon.com/cloudfront/v4/home>。
2. CloudFront コンソールの右ペインで、削除するディストリビューションを見つけます。
 - ステータス列に Disabled と表示されている場合は、ステップ 6 に進みます。
 - ステータスが Enabled と表示されているが、ディストリビューションが最終更新列にまだデプロイ中と表示されている場合は、デプロイが完了するのを待ってからステップ 3 に進みます。
3. CloudFront コンソールの右ペインで、削除するディストリビューションのチェックボックスをオンにします。
4. [Disable (無効)] を選択してディストリビューションを無効にし、[Yes, Disable (はい、無効にします)] を選択して確定します。次に、[Close] (閉じる) を選択します。

Note

CloudFront はこの変更をすべてのエッジロケーションに伝達する必要があるため、更新が完了してディストリビューションを削除できるようになるまでに数分かかる場合があります。

5. Status 列の値は、すぐに Disabled に変わります。最終更新日時列の下に新しいタイムスタンプが表示されるまで待ちます。
6. 削除するディストリビューションのチェックボックスをオンにします。
7. [Delete] (削除) を選択し、削除します。

Note

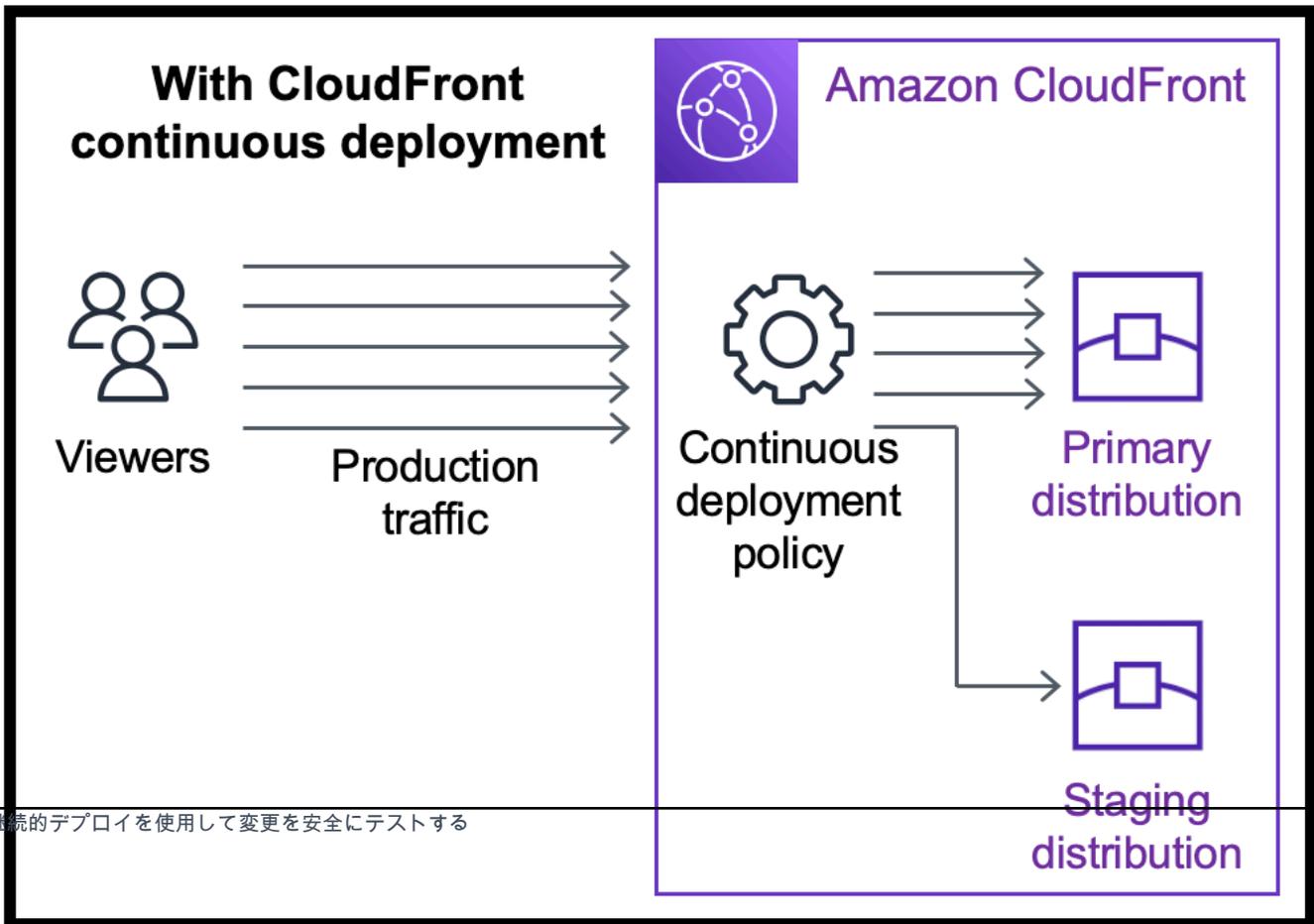
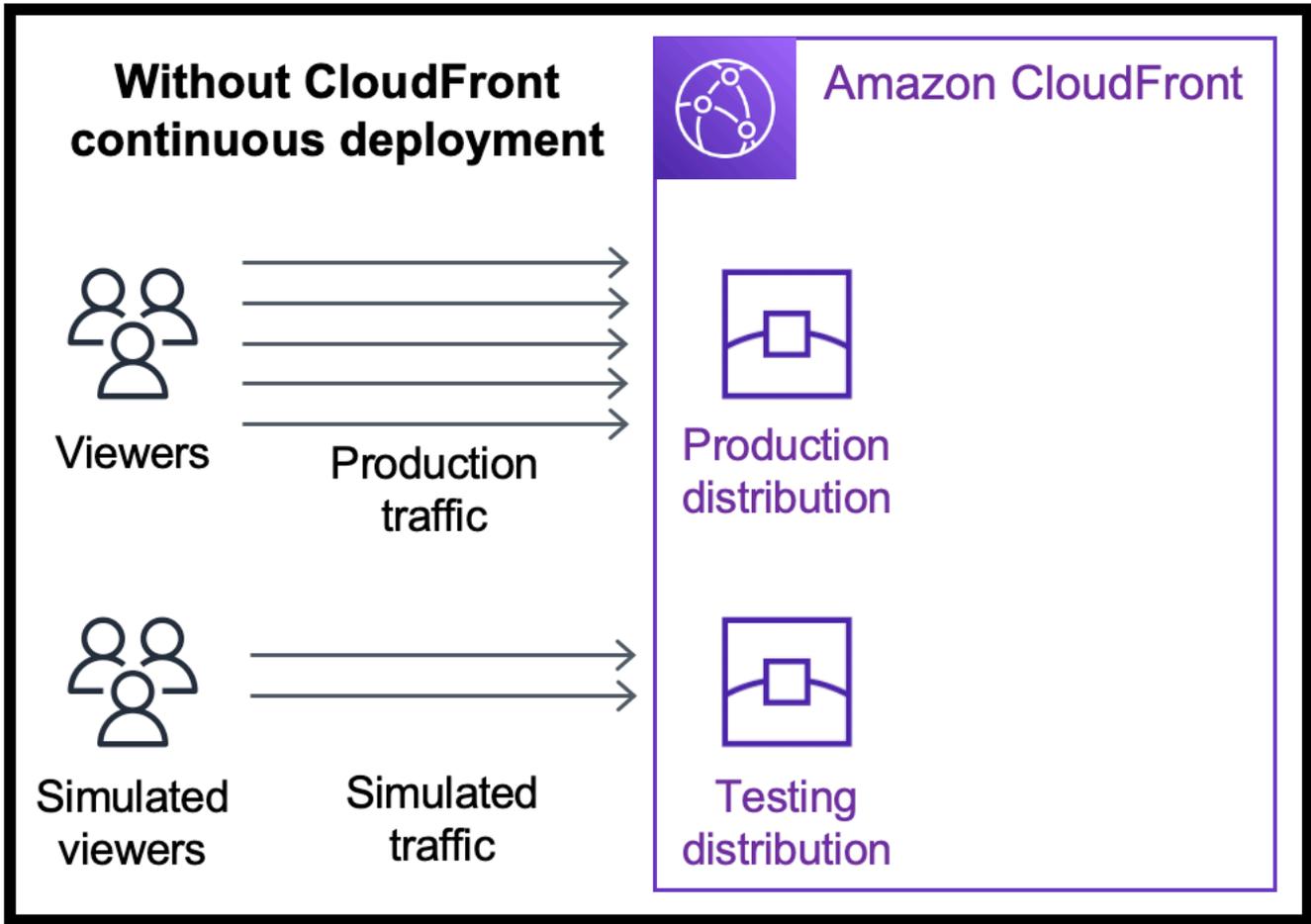
ディストリビューションを無効にマークしたばかりの場合は、エッジロケーションに変更が反映されるまでにさらに数分かかる CloudFront ことがあります。伝達が完了するまでは、[削除]オプションを使用することができません。

CloudFront API を使用してディストリビューションを削除することもできます。詳細については、「Amazon CloudFront API リファレンス [DeleteDistribution](#)」の「」を参照してください。

CloudFront 継続的デプロイを使用して CDN 設定の変更を安全にテストする

Amazon CloudFront の継続的デプロイでは、最初に本番トラフィックのサブセットでテストすることで、CDN 設定に変更を安全にデプロイできます。ステージングディストリビューションと継続的デプロイポリシーを使用して、実際 (本番環境) のビューワーからのトラフィックを新しい CDN 設定に送信し、正常に動作することを確認できます。新しい設定のパフォーマンスをリアルタイムでモニタリングし、準備ができたら、新しい設定を昇格させてすべてのトラフィックをプライマリディストリビューションで処理できます。

次の図は、CloudFront 継続的デプロイを使用する利点を示しています。これを使用しない場合は、シミュレートしたトラフィックで CDN 設定の変更をテストする必要があります。継続的デプロイでは、本番トラフィックのサブセットで変更をテストし、準備ができたら、変更をプライマリディストリビューションに昇格させることができます。



トピック

- [CloudFront 継続的デプロイを使用するワークフロー](#)
- [ステージングディストリビューションと継続的デプロイポリシーの使用](#)
- [ステージングディストリビューションのモニタリング](#)
- [継続的デプロイの仕組みを理解する](#)
- [継続的デプロイに関するクォータとその他の考慮事項](#)

CloudFront 継続的デプロイを使用するワークフロー

次の大まかなワークフローでは、CloudFront 継続的デプロイで設定変更を安全にテストしてデプロイする方法について説明します。

1. プライマリディストリビューションとして使用するディストリビューションを選択します。プライマリディストリビューションは、本番トラフィックを現在処理しているディストリビューションです。
2. プライマリディストリビューションから、ステージングディストリビューションを作成します。ステージングディストリビューションは、プライマリディストリビューションのコピーとして開始します。
3. 継続的デプロイポリシー内にトラフィック設定を作成し、それをプライマリディストリビューションにアタッチします。これにより、[ガトラフィックをステージングディストリビューションに CloudFront ルーティングする方法が決まります](#)。ステージングディストリビューションにリクエストをルーティングする方法の詳細については、「[the section called “ステージングディストリビューションへのリクエストのルーティング”](#)」を参照してください。
4. ステージングディストリビューションの設定を更新します。更新できる設定の詳細については、「[the section called “プライマリディストリビューションとステージングディストリビューションの更新”](#)」を参照してください。
5. ステージングディストリビューションをモニタリングして、設定の変更が正常に動作するかどうかを確認します。ステージングディストリビューションのモニタリングの詳細については、「[the section called “ステージングディストリビューションのモニタリング”](#)」を参照してください。

ステージングディストリビューションをモニタリングしながら、以下のことを実行できます。

- ステージングディストリビューションの設定を再度更新し、設定の変更のテストを続行します。
- 継続的デプロイポリシー (トラフィック設定) を更新して、ステージングディストリビューションに送信するトラフィックを増減します。

6. ステージングディストリビューションのパフォーマンスに満足したら、ステージングディストリビューションの設定をプライマリディストリビューションに昇格させます。これにより、ステージングディストリビューションの設定がプライマリディストリビューションにコピーされます。これにより、継続的デプロイポリシーも無効になります。つまり、はすべてのトラフィックをプライマリディストリビューションに CloudFront ルーティングします。

オートメーションを構築することで、ステージングディストリビューションのパフォーマンスをモニタリングして (ステップ 5)、特定の基準を満たしたときに自動的に設定を昇格させる (ステップ 6) ことができます。

設定を昇格させると、設定の変更を次回テストするときに、同じステージングディストリビューションを再利用できます。

CloudFront コンソール、AWS CLI または CloudFront API でのステージングディストリビューションと継続的デプロイポリシーの使用の詳細については、次のセクションを参照してください。

ステージングディストリビューションと継続的デプロイポリシーの使用

CloudFront コンソール、AWS Command Line Interface (AWS CLI)、または CloudFront API を使用して、ステージングディストリビューションと継続的デプロイポリシーを作成、更新、変更できます。

Console

ステージングディストリビューションと継続的デプロイポリシーを AWS Management Console で使用するには、次の手順に従います。

ステージングディストリビューションと継続的デプロイポリシーを作成するには (コンソール)

1. にサインインAWS Management Consoleし、 で CloudFront コンソールを開きます <https://console.aws.amazon.com/cloudfront/v4/home>。
2. ナビゲーションペインで、[Distribution] (ディストリビューション) を選択します。
3. プライマリディストリビューションとして使用するディストリビューションを選択します。プライマリーディストリビューションは、本番トラフィックを現在処理しているディストリビューションであり、これを元にしてステージングディストリビューションを作成します。
4. [Continuous deployment] (継続的デプロイ) セクションで、[Create staging distribution] (ステージングディストリビューションを作成) を選択します。[Create staging distribution] (ステージングディストリビューションを作成) ウィザードが開きます。

5. [Create staging distribution] (ステージングディストリビューションを作成) ウィザードで、以下の操作を行います。
 - a. (オプション) ステージングディストリビューションの説明を入力します。
 - b. [Next] (次へ) を選択します。
 - c. ステージングディストリビューションの設定を変更します。更新できる設定の詳細については、「[the section called “プライマリーディストリビューションとステージングディストリビューションの更新”](#)」を参照してください。

ステージングディストリビューションの設定の変更が完了したら、[Next] (次へ) を選択します。

- d. コンソールを使用して [Traffic configuration] (トラフィック設定) を指定します。これにより、ガトラフィックをステージングディストリビューションに CloudFront ルーティングする方法が決まります。(トラフィック設定は継続的デプロイポリシーに CloudFront 保存されます)。

[Traffic configuration] (トラフィック設定) のオプションの詳細については、「[the section called “ステージングディストリビューションへのリクエストのルーティング”](#)」を参照してください。

[Traffic configuration] (トラフィック設定) が完了したら、[Next] (次へ) を選択します。

- e. トラフィック設定を含む、ステージングディストリビューションの設定を確認したら、[Create staging distribution] (ステージングディストリビューションを作成) を選択します。

CloudFront コンソールでステージングディストリビューションの作成ウィザードを終了すると、は次の CloudFront 操作を行います。

- ステップ 5c で指定した設定を使用して、ステージングディストリビューションを作成します。
- ステップ 5d で指定したトラフィック設定を使用して、継続的デプロイポリシーを作成します。
- ステージングディストリビューションの作成元のプライマリディストリビューションに継続的デプロイポリシーをアタッチします。

継続的デプロイポリシーがアタッチされたプライマリディストリビューションの設定がエッジロケーションにデプロイされると、トラフィック設定に基づいてトラフィックの指定された部分をステージングディストリビューションに送信 CloudFront し始めます。

ステージングディストリビューションを更新するには (コンソール)

1. で CloudFront コンソールを開きます <https://console.aws.amazon.com/cloudfront/v4/home>。
2. ナビゲーションペインで、[Distribution] (ディストリビューション) を選択します。
3. プライマリディストリビューションを選択します。これは、本番トラフィックを現在処理しているディストリビューションで、ステージングディストリビューションの作成元のディストリビューションです。
4. [View staging distribution] (ステージングディストリビューションを表示) を選択します。
5. コンソールを使用して、ステージングディストリビューションの設定を変更します。更新できる設定の詳細については、「[the section called “プライマリーディストリビューションとステージングディストリビューションの更新”](#)」を参照してください。

ステージングディストリビューションの設定をエッジロケーションにデプロイするとすぐに、ステージングディストリビューションにルーティングされた受信トラフィックに設定が適用されます。

継続的デプロイポリシーを更新するには (コンソール)

1. で CloudFront コンソールを開きます <https://console.aws.amazon.com/cloudfront/v4/home>。
2. ナビゲーションペインで、[Distribution] (ディストリビューション) を選択します。
3. プライマリディストリビューションを選択します。これは、本番トラフィックを現在処理しているディストリビューションで、ステージングディストリビューションの作成元のディストリビューションです。
4. [Continuous deployment] (継続的デプロイ) セクションで、[Edit policy] (ポリシーの編集) を選択します。
5. 継続的デプロイポリシーのトラフィック設定を変更します。完了したら、[変更を保存] を選択します。

更新された継続的デプロイポリシーを持つプライマリディストリビューションの設定がエッジロケーションにデプロイされると、は更新されたトラフィック設定に基づいてステージングディストリビューションへのトラフィックの送信 CloudFront を開始します。

ステージングディストリビューションの設定を昇格させるには (コンソール)

1. で CloudFront コンソールを開きます <https://console.aws.amazon.com/cloudfront/v4/home>。
2. ナビゲーションペインで、[Distribution] (ディストリビューション) を選択します。
3. プライマリディストリビューションを選択します。これは、本番トラフィックを現在処理しているディストリビューションで、ステージングディストリビューションの作成元のディストリビューションです。
4. [Continuous deployment] (継続的デプロイ) セクションで、[Promote] (昇格) を選択します。
5. 「**confirm**」と入力して [Promote] (昇格) を選択します。

ステージングディストリビューションを昇格させると、はステージングディストリビューションからプライマリディストリビューションに設定 CloudFront をコピーします。CloudFront また、は継続的デプロイポリシーを無効にし、すべてのトラフィックをプライマリディストリビューションにルーティングします。

設定を昇格させると、設定の変更を次回テストするときに、同じステージングディストリビューションを再利用できます。

CLI

ステージングディストリビューションと継続的デプロイポリシーを AWS CLI で使用するには、次の手順に従います。

ステージングディストリビューションを作成するには (CLI)

1. `aws cloudfront get-distribution` コマンドと `grep` コマンドと一緒に使用して、プライマリディストリビューションとして使用するディストリビューションの ETag 値を取得します。プライマリディストリビューションは、本番トラフィックを現在処理しているディストリビューションであり、これを元にしてステージングディストリビューションを作成します。

次のコマンドでは、例を示しています。次の例では、`primary_distribution_ID` をプライマリディストリビューションの ID に置き換えます。

```
aws cloudfront get-distribution --id primary_distribution_ID | grep 'ETag'
```

ETag 値をコピーします (次のステップで必要になります)。

2. `aws cloudfront copy-distribution` コマンドを使用してステージングディストリビューションを作成します。次のコマンド例では、読みやすくするためにエスケープ文字 (\) と改行を使用していますが、これらはコマンドから省略してください。次のコマンドの例で以下の操作を行います。
 - `primary_distribution_ID` をプライマリディストリビューションの ID に置き換えます。
 - `primary_distribution_ETag` をプライマリディストリビューションの ETag 値 (前のステップで取得したもの) に置き換えます。
 - (オプション) `CLI_example` を目的の発信者リファレンス ID に置き換えます。

```
aws cloudfront copy-distribution --primary-distribution-id primary_distribution_ID \  
                                --if-match primary_distribution_ETag \  
                                --staging \  
                                --caller-reference 'CLI_example'
```

コマンドの出力に、ステージングディストリビューションとその設定に関する情報が表示されます。ステージングディストリビューションの CloudFront ドメイン名をコピーします。これは、次のステップで必要になります。

継続的デプロイポリシーを作成するには (CLI および入力ファイル)

1. 次のコマンドを使用して、`continuous-deployment-policy.yaml` コマンドのすべての入力パラメータを含む、`create-continuous-deployment-policy` という名前のファイルを作成します。次のコマンドでは、読みやすくするためにエスケープ文字 (\) と改行を使用していますが、これらはコマンドから省略してください。

```
aws cloudfront create-continuous-deployment-policy --generate-cli-skeleton yaml-input \  
                                                    > continuous-deployment-policy.yaml
```

2. 先ほど作成した `continuous-deployment-policy.yaml` という名前のファイルを開きます。このファイルを編集して、必要な継続的デプロイポリシー設定を指定し、ファイルを保存します。ファイルは以下のように編集します。

- StagingDistributionDnsNames セクションでの編集
 - Quantity の値を 1 に変更します。
 - には Items、ステージングディストリビューションの CloudFront ドメイン名を貼り付けます (前の手順で保存したもの)。
- TrafficConfig セクションでの編集
 - Type として、SingleWeight または SingleHeader を選択します。
 - 他のタイプの設定を削除します。例えば、重みベースのトラフィック設定が必要な場合は、Type を SingleWeight に設定し、SingleHeaderConfig 設定を削除します。
 - 重みベースのトラフィック設定を使用するには、Weight の値を .01 (1%) から .15 (15%) までの 10 進数に設定します。

TrafficConfig のオプションの詳細については、「[the section called “ステージングディストリビューションへのリクエストのルーティング”](#)」および「[the section called “重みベースの設定におけるセッションの維持”](#)」を参照してください。

3. 次のコマンドで continuous-deployment-policy.yaml ファイルの入力パラメータを使用し、継続的デプロイポリシーを作成します。

```
aws cloudfront create-continuous-deployment-policy --cli-input-yaml file://
continuous-deployment-policy.yaml
```

コマンドの出力の Id 値をコピーします。これは継続的デプロイポリシー ID で、次のステップで必要になります。

継続的デプロイポリシーをプライマリディストリビューションにアタッチするには (CLI および入力ファイル)

1. 次のコマンドを使用して、プライマリディストリビューションの設定を primary-distribution.yaml という名前のファイルに保存します。 *primary_distribution_ID* をプライマリディストリビューションの ID に置き換えます。

```
aws cloudfront get-distribution-config --id primary_distribution_ID --output
yaml > primary-distribution.yaml
```

- 先ほど作成した `primary-distribution.yaml` という名前のファイルを開きます。ファイルを編集し、以下の変更を加えます。
 - 継続的デプロイポリシー ID (前のステップでコピーしたもの) を `ContinuousDeploymentPolicyId` フィールドに貼り付けます。
 - ETag フィールドの名前を `IfMatch` に変更します。ただし、フィールドの値は変更しないでください。

完了したら、ファイルを保存します。

- 次のコマンドを使用して、継続的デプロイポリシーを使用するようにプライマリディストリビューションを更新します。`primary_distribution_ID` をプライマリディストリビューションの ID に置き換えます。

```
aws cloudfront update-distribution --id primary_distribution_ID --cli-input-yaml file://primary-distribution.yaml
```

継続的デプロイポリシーがアタッチされたプライマリディストリビューションの設定がエッジロケーションにデプロイされると、トラフィック設定に基づいてトラフィックの指定された部分をステージングディストリビューションに送信 CloudFront し始めます。

ステージングディストリビューションを更新するには (CLI および入力ファイル)

- 次のコマンドを使用して、ステージングディストリビューションの設定を `staging-distribution.yaml` という名前のファイルに保存します。`staging_distribution_ID` をステージングディストリビューションの ID に置き換えます。

```
aws cloudfront get-distribution-config --id staging_distribution_ID --output yaml > staging-distribution.yaml
```

- 先ほど作成した `staging-distribution.yaml` という名前のファイルを開きます。ファイルを編集し、以下の変更を加えます。
 - ステージングディストリビューションの設定を変更します。更新できる設定の詳細については、「[the section called “プライマリディストリビューションとステージングディストリビューションの更新”](#)」を参照してください。

- ETag フィールドの名前を IfMatch に変更します。ただし、フィールドの値は変更しないでください。

完了したら、ファイルを保存します。

3. 次のコマンドを使用して、ステージングディストリビューションの設定を更新します。*staging_distribution_ID* をステージングディストリビューションの ID に置き換えます。

```
aws cloudfront update-distribution --id staging_distribution_ID --cli-input-yaml
file://staging-distribution.yaml
```

ステージングディストリビューションの設定をエッジロケーションにデプロイするとすぐに、ステージングディストリビューションにルーティングされた受信トラフィックに設定が適用されます。

継続的デプロイポリシーを更新するには (CLI および入力ファイル)

1. 次のコマンドを使用して、継続的デプロイポリシーの設定を `continuous-deployment-policy.yaml` という名前のファイルに保存します。*continuous_deployment_policy_ID* を継続的デプロイポリシーの ID に置き換えます。次のコマンドでは、読みやすくするためにエスケープ文字 (\) と改行を使用していますが、これらはコマンドから省略してください。

```
aws cloudfront get-continuous-deployment-policy-config --
id continuous_deployment_policy_ID \
                                                    --output yaml >
continuous-deployment-policy.yaml
```

2. 先ほど作成した `continuous-deployment-policy.yaml` という名前のファイルを開きます。ファイルを編集し、以下の変更を加えます。
 - 継続的デプロイポリシーの設定を必要に応じて変更します。例えば、トラフィック設定をヘッダーベースから重みベースに変更したり、重みベースの設定でトラフィックの割合 (重み) を変更したりできます。詳細については、「[the section called “ステージングディストリビューションへのリクエストのルーティング”](#)」および「[the section called “重みベースの設定におけるセッションの維持”](#)」を参照してください。

- ETag フィールドの名前を IfMatch に変更します。ただし、フィールドの値は変更しないでください。

完了したら、ファイルを保存します。

3. 次のコマンドを使用して、継続的デプロイポリシーを更新します。*continuous_deployment_policy_ID* を継続的デプロイポリシーの ID に置き換えます。次のコマンドでは、読みやすくするためにエスケープ文字 (\) と改行を使用していますが、これらはコマンドから省略してください。

```
aws cloudfront update-continuous-deployment-policy --  
id continuous_deployment_policy_ID \  
                                                    --cli-input-yaml file://  
continuous-deployment-policy.yaml
```

更新された継続的デプロイポリシーを持つプライマリディストリビューションの設定がエッジロケーションにデプロイされると、は更新されたトラフィック設定に基づいてステージングディストリビューションへのトラフィックの送信 CloudFront を開始します。

ステージングディストリビューションの設定を昇格させるには (CLI)

- `aws cloudfront update-distribution-with-staging-config` コマンドを使用して、ステージングディストリビューションの設定をプライマリディストリビューションに昇格させます。次のコマンド例では、読みやすくするためにエスケープ文字 (\) と改行を使用していますが、これらはコマンドから省略してください。次のコマンドの例で以下の操作を行います。
 - *primary_distribution_ID* をプライマリディストリビューションの ID に置き換えます。
 - *staging_distribution_ID* をステージングディストリビューションの ID に置き換えます。
 - *primary_distribution_ETag* と *staging_distribution_ETag* をプライマリディストリビューションとステージングディストリビューションの ETag 値に置き換えます。例に示すように、プライマリディストリビューションの値が最初になっていることを確認します。

```
aws cloudfront update-distribution-with-staging-config --
id primary_distribution_ID \
                                                    --staging-distribution-
id staging_distribution_ID \
                                                    --if-match
'primary_distribution_ETag, staging_distribution_ETag'
```

ステージングディストリビューションを昇格させると、はステージングディストリビューションからプライマリディストリビューションに設定 CloudFront をコピーします。CloudFront また、は継続的デプロイポリシーを無効にし、すべてのトラフィックをプライマリディストリビューションにルーティングします。

設定を昇格させると、設定の変更を次回テストするときに、同じステージングディストリビューションを再利用できます。

API

CloudFront API を使用してステージングディストリビューションと継続的デプロイポリシーを作成するには、次の API オペレーションを使用します。

- [CopyDistribution](#)
- [CreateContinuousDeploymentPolicy](#)

これらの API コールで指定するフィールドの詳細については、以下を参照してください。

- [the section called “ステージングディストリビューションへのリクエストのルーティング”](#)
- [the section called “重みベースの設定におけるセッションの維持”](#)
- AWS SDK またはその他の API クライアントの API リファレンスドキュメント

ステージングディストリビューションと継続的デプロイポリシーを作成したら、[UpdateDistribution](#) (プライマリディストリビューションで) を使用して継続的デプロイポリシーをプライマリディストリビューションにアタッチします。

ステージングディストリビューションの設定を更新するには、[UpdateDistribution](#) (ステージングディストリビューションで) を使用してステージングディストリビューションの設定を変更します。更新できる設定の詳細については、「[the section called “プライマリーディストリビューションとステージングディストリビューションの更新”](#)」を参照してください。

ステージングディストリビューションの設定をプライマリディストリビューションに昇格させるには、[UpdateDistributionWithStagingConfig](#)を使用します。

この API コールで指定するフィールドの詳細については、AWS SDK またはその他の API クライアントの API リファレンスドキュメントを参照してください。

ステージングディストリビューションのモニタリング

ステージングディストリビューションのパフォーマンスをモニタリングするには、[がすべてのディストリビューションに対して CloudFront 提供するのと同じメトリクス、ログ、レポート](#)を使用できます。例:

- [デフォルトの CloudFrontディストリビューションメトリクス](#) (総リクエスト数やエラー率など) は CloudFrontコンソールで表示でき、[追加のメトリクス \(キャッシュヒット率やステータスコード別のエラー率など\)](#) は追加料金で有効にできます。これらのメトリクスに基づいてアラームを作成することもできます。
- [標準ログ](#)と[リアルタイムログ](#)を表示して、ステージングディストリビューションが受信したリクエストに関する詳細情報を取得できます。標準ログには、リクエストが最初に送信されたプライマリディストリビューションを特定してからステージングディストリビューションに CloudFront ルーティングするのに役立つ 2 つのフィールド `primary-distribution-id`と `primary-distribution-dns-name`が含まれています。
- キャッシュ統計[レポート](#)など、コンソールで CloudFrontレポートを表示およびダウンロードできます。

継続的デプロイの仕組みを理解する

以下のトピックでは、CloudFront 継続的デプロイの仕組みについて説明します。

トピック

- [ステージングディストリビューションへのリクエストのルーティング](#)
- [重みベースの設定におけるセッションの維持](#)
- [プライマリディストリビューションとステージングディストリビューションの更新](#)
- [プライマリディストリビューションとステージングディストリビューションはキャッシュを共有しない](#)

ステージングディストリビューションへのリクエストのルーティング

CloudFront 継続的デプロイを使用する場合、ビューワーリクエストについて何も変更する必要はありません。ビューワーは、DNS 名、IP アドレス、または CNAME を使用してステージングディストリビューションにリクエストを直接送信することはできません。代わりに、ビューワーはプライマリ (本番稼働) ディストリビューションにリクエストを送信し、継続的デプロイポリシーのトラフィック設定に基づいて、これらのリクエストの一部をステージングディストリビューションに CloudFront ルーティングします。トラフィック設定には次の 2 つの種類があります。

重みベース

重みベースの設定では、ビューワーリクエストの指定された割合をステージングディストリビューションにルーティングします。重みベースの設定を使用する場合、セッションの維持を有効にすることもできます。これにより、は同じビューワーからのリクエストを 1 つのセッションの一部として CloudFront 処理できます。詳細については、「[the section called “重みベースの設定におけるセッションの維持”](#)」を参照してください。

ヘッダーベース

ビューワーリクエストに特定の HTTP ヘッダーが含まれている (ヘッダーと値が指定されている) 場合、ヘッダーベースの設定は、リクエストをステージングディストリビューションにルーティングします。ヘッダーと値が指定されていないリクエストは、プライマリディストリビューションにルーティングされます。この設定は、ローカルでテストする場合や、ビューワーリクエストが制御可能である場合に便利です。

Note

ステージングディストリビューションにルーティングするヘッダーには、プレフィックス `aws-cf-cd-` が含まれている必要があります。

重みベースの設定におけるセッションの維持

重みベースの設定を使用してトラフィックをステージングディストリビューションにルーティングする場合、セッションの維持を有効にすることもできます。これにより、は同じビューワーからのリクエストを 1 つのセッションとして CloudFront 処理できます。セッションの維持を有効にすると、は Cookie CloudFront を設定し、1 つのセッションで同じビューワーからのすべてのリクエストが、プライマリまたはステージングの 1 つのディストリビューションによって処理されるようになります。

セッションの維持を有効にするときに、アイドル期間を指定することもできます。ビューワーがこの時間アイドル状態である (リクエストを送信しない) 場合、セッションは期限切れになり、このビューワーからの今後のリクエストを新しいセッションとして CloudFront 扱います。アイドル期間は 300 (5 分) から 3,600 (1 時間) までの秒数で指定します。

次の場合、はすべてのセッション (アクティブなセッションも含む) CloudFront をリセットし、すべてのリクエストを新しいセッションと見なします。

- 継続的デプロイポリシーを無効または有効にする。
- セッションの維持の設定を無効または有効にする。

プライマリーディストリビューションとステージングディストリビューションの更新

プライマリディストリビューションに継続的デプロイポリシーがアタッチされている場合、プライマリーディストリビューションとステージングディストリビューションの両方で、以下の設定の変更が可能です。

- すべてのキャッシュ動作設定 (デフォルトのキャッシュ動作を含む)
- すべてのオリジン設定 (オリジンとオリジングループ)
- カスタムエラーレスポンス (エラーページ)
- 地理的制限
- デフォルトのルートオブジェクト
- ログ記録の設定
- 説明 (コメント)

キャッシュポリシー、レスポンスヘッダーポリシー、CloudFront 関数、Lambda@Edge 関数など、ディストリビューションの設定で参照されている外部リソースを更新することもできます。

プライマリディストリビューションとステージングディストリビューションはキャッシュを共有しない

プライマリディストリビューションとステージングディストリビューションはキャッシュを共有しません。が最初のリクエストをステージングディストリビューション CloudFront に送信すると、そのキャッシュは空になります。リクエストがステージングディストリビューションに到着すると、レスポンスのキャッシュが開始されます (そのように設定している場合)。

継続的デプロイに関するクォータとその他の考慮事項

CloudFront 継続的デプロイには、以下のクォータおよびその他の考慮事項が適用されます。

クォータ

- AWS アカウント あたりのステージングディストリビューションの最大数: 20
- AWS アカウント あたりの継続的デプロイポリシーの最大数: 20
- 重みベースの設定でステージングディストリビューションに送信できるトラフィックの最大割合: 15%
- セッションの維持のアイドル期間の最小値と最大値: 300 ~ 3,600 秒

詳細については、「[クォータ](#)」を参照してください。

Note

継続的デプロイを使用し、プライマリディストリビューションに S3 バケットアクセス用の OAC が設定されている場合は、S3 バケットポリシーを更新してステージングディストリビューションへのアクセスを許可します。S3 バケットポリシーの例については、「」を参照してください [the section called “S3 バケットへのアクセス許可をオリジンアクセスコントロールに付与する”](#)。

HTTP/3

HTTP/3 をサポートするディストリビューションでは継続的デプロイを使用できません。

がすべてのリクエストをプライマリディストリビューション CloudFront に送信する場合

リソース使用率が高い期間など、状況によっては、継続的デプロイポリシーで指定されている内容に関係なく、すべてのリクエストをプライマリディストリビューションに送信する CloudFront 場合があります。

CloudFront は、継続的デプロイポリシーで指定されている内容に関係なく、ピークトラフィック時間中にすべてのリクエストをプライマリディストリビューションに送信します。ピークトラフィックとは、ディストリビューションのトラフィックではなく、CloudFront サービスのトラフィックを指します。

ディストリビューションでの CloudFront さまざまなオリジンの使用

ディストリビューションを作成するときは、ファイルのリクエスト CloudFront を送信するオリジンを指定します。では、さまざまな種類のオリジンを使用できません CloudFront。例えば、Amazon S3 バケット、MediaStore コンテナ、MediaPackage チャンネル、Application Load Balancer、または AWS Lambda 関数 URL を使用できます。

トピック

- [Amazon S3 バケットの使用](#)
- [MediaStore コンテナまたは MediaPackage チャンネルの使用](#)
- [Application Load Balancer の使用](#)
- [Lambda 関数 URL の使用](#)
- [Amazon EC2 \(または他のカスタムオリジン\) の使用](#)
- [CloudFront オリジングループの使用](#)

Amazon S3 バケットの使用

以下のトピックでは、Amazon S3 バケットを CloudFront ディストリビューションのオリジンとして使用するさまざまな方法について説明します。

トピック

- [標準的な Amazon S3 バケットの使用](#)
- [Amazon S3 Object Lambda の使用](#)
- [ウェブサイトのエンドポイントとして設定された Amazon S3 バケットの使用](#)
- [既存の Amazon S3 バケット CloudFront への追加](#)
- [Amazon S3 バケットを別の AWS リージョンに移動する](#)

標準的な Amazon S3 バケットの使用

ディストリビューションのオリジンとして Amazon S3 を使用する場合は、CloudFront 配信するオブジェクトを Amazon S3 バケットに配置します。オブジェクトを Amazon S3 に保存するには、Amazon S3 でサポートされる任意の方法を使用できます。例えば、Amazon S3 コンソールや

API、サードパーティーのツールを使用できます。他の標準 Amazon S3 バケットと同様に、バケット内に階層を作成してオブジェクトを保存できます。

既存の Amazon S3 バケットを CloudFront オリジンサーバーとして使用しても、バケットは変更されません。通常の Amazon S3 料金で Amazon S3 オブジェクトを保存してアクセスする場合と同じように使用できます。Amazon S3 バケットへのオブジェクトの保存には、通常の Amazon S3 料金が発生します。の使用料金の詳細については CloudFront、[「Amazon CloudFront 料金表」](#)を参照してください。既存の S3 バケット CloudFront でを使用する方法の詳細については、「」を参照してください[the section called “既存の Amazon S3 バケット CloudFront への追加”](#)。

⚠ Important

バケットでを使用するには CloudFront、名前が DNS 命名要件に準拠している必要があります。詳細については、Amazon Simple Storage Service ユーザーガイドの「[バケットの命名規則](#)」を参照してください。

のオリジンとして Amazon S3 バケットを指定する場合は CloudFront、次の形式を使用することを勧めします。

`bucket-name.s3.region.amazonaws.com`

この形式でバケット名を指定した場合は、以下の CloudFront 機能を使用できます。

- SSL/TLS を使用して Amazon S3 バケットと通信 CloudFront するようにを設定します。詳細については、「[the section called “で HTTPS を使用する CloudFront”](#)」を参照してください。
- オリジンアクセスコントロールを使用して、Amazon S3 URLsではなく CloudFront URLs を使用してコンテンツにアクセスすることをビューワーに要求します。Amazon S3 詳細については、「[the section called “Amazon S3 オリジンへのアクセスの制限”](#)」を参照してください。
- に POSTおよび PUTリクエストを送信して、バケットのコンテンツを更新します CloudFront。詳細については、トピック「[the section called “がリクエスト CloudFront を処理して Amazon S3 オリジンに転送する方法”](#)」の「[the section called “HTTP メソッド”](#)」を参照してください。

以下の形式を使用してバケットを指定しないでください。

- Amazon S3 パススタイル: `s3.amazonaws.com/bucket-name`
- Amazon S3 の CNAME

Amazon S3 Object Lambda の使用

[Object Lambda アクセスポイントを作成する](#)と、Amazon S3 は Object Lambda アクセスポイントの固有のエイリアスを自動的に生成します。Amazon S3 バケット名の代わりに[このエイリアス](#)を CloudFront ディストリビューションのオリジンとして使用できます。

Object Lambda アクセスポイントエイリアスを のオリジンとして使用する場合は CloudFront、次の形式を使用することをお勧めします。

`alias.s3.region.amazonaws.com`

`alias` の検索の詳細については、「Amazon S3 ユーザーガイド」の「[S3 バケット Object Lambda アクセスポイントにおけるバケット形式のエイリアスの使用](#)」を参照してください。

Important

Object Lambda アクセスポイントを のオリジンとして使用する場合は CloudFront、[オリジンアクセスコントロール](#)を使用する必要があります。

ユースケースの例については、「[Amazon S3 Object Lambda と Amazon CloudFront を使用してエンドユーザーのコンテンツを調整する](#)」を参照してください。

CloudFront は、Object Lambda アクセスポイントのオリジンを[標準の Amazon S3 バケットオリジン](#)と同じように扱います。

Amazon S3 Object Lambda をディストリビューションのオリジンとして使用する場合は、次の 4 つのアクセス許可を設定する必要があります。

Object Lambda アクセスポイントのアクセス許可

1. AWS Management Console にサインインし、Amazon S3 コンソール (<https://console.aws.amazon.com/s3/>) を開きます。
2. 左のナビゲーションペインで、[Object Lambda アクセスポイント] を選択します。
3. 使用する Object Lambda アクセスポイントを選択します。
4. [アクセス許可] タブを選択します。
5. [Object Lambda アクセスポイントポリシー] セクションで [編集] を選択します。
6. 以下のポリシーを [ポリシー] フィールドに貼り付けます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudfront.amazonaws.com"
      },
      "Action": "s3-object-lambda:Get*",
      "Resource": "arn:aws:s3-object-lambda:<region>:<AWS #####
ID>:accesspoint/<Object Lambda Access Point name>",
      "Condition": {
        "StringEquals": {
          "aws:SourceArn": "arn:aws:cloudfront::<AWS #####
ID>:distribution/<CloudFront distribution ID>"
        }
      }
    }
  ]
}
```

7. [変更の保存] をクリックします。

Amazon S3 アクセスポイントのアクセス許可

1. AWS Management Console にサインインし、Amazon S3 コンソール (<https://console.aws.amazon.com/s3/>) を開きます。
2. ナビゲーションペインで、[アクセスポイント] を選択します。
3. 使用する Amazon S3 アクセスポイントを選択します。
4. [アクセス許可] タブを選択します。
5. [アクセスポイントポリシー] セクションで [編集] を選択します。
6. 以下のポリシーを [ポリシー] フィールドに貼り付けます。

```
{
  "Version": "2012-10-17",
  "Id": "default",
  "Statement": [
    {
      "Sid": "s3objlambda",
```

```
        "Effect": "Allow",
        "Principal": {
            "Service": "cloudfront.amazonaws.com"
        },
        "Action": "s3:*",
        "Resource": [
            "arn:aws:s3:<region>:<AWS ##### ID>:accesspoint/<Access Point name>",
            "arn:aws:s3:<region>:<AWS ##### ID>:accesspoint/<Access Point name>/object/*"
        ],
        "Condition": {
            "ForAnyValue:StringEquals": {
                "aws:CalledVia": "s3-object-lambda.amazonaws.com"
            }
        }
    ]
}
```

7. [保存] を選択します。

Amazon S3 バケットのアクセス許可

1. AWS Management Console にサインインし、Amazon S3 コンソール (<https://console.aws.amazon.com/s3/>) を開きます。
2. ナビゲーションペインで、バケットを選択します。
3. 使用する Amazon S3 バケットを選択します。
4. [アクセス許可] タブを選択します。
5. [バケットポリシー] セクションで、[編集] を選択します。
6. 以下のポリシーを [ポリシー] フィールドに貼り付けます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
```

```
    "Action": "*",
    "Resource": [
      "arn:aws:s3:::<bucket name>",
      "arn:aws:s3:::<bucket name>/*"
    ],
    "Condition": {
      "StringEquals": {
        "s3:DataAccessPointAccount": "<AWS ##### ID>"
      }
    }
  }
]
```

7. [変更の保存] をクリックします。

AWS Lambda アクセス許可

1. AWS Management Console にサインインして AWS Lambda コンソール (<https://console.aws.amazon.com/lambda/>) を開きます。
2. ナビゲーションペインで、[関数] を選択します。
3. 使用する AWS Lambda 関数を選択します。
4. [設定] タブを開き、次に [アクセス許可] をクリックします。
5. [リソーススペースのポリシーステートメント] セクションで [アクセス許可の追加] を選択します。
6. [AWS アカウント] を選択します。
7. [ステートメント ID] の名前を入力します。
8. [プリンシパル] の `cloudfront.amazonaws.com` を入力します。
9. [アクション] ドロップダウンメニューから `lambda:InvokeFunction` を選択します。
10. [保存] を選択します。

ウェブサイトのエンドポイントとして設定された Amazon S3 バケットの使用

ウェブサイトエンドポイントとして設定された Amazon S3 バケットを、のカスタムオリジンとして使用できます CloudFront。 CloudFront デイストリビューションを設定するときに、オリジンにバケットのエンドポイントをホストする Amazon S3 静的ウェブサイトを入力します。この値は、[Amazon S3 コンソール](#)の [Static Website Hosting] (静的ウェブサイトホスティング) ペインの [Properties] (プロパティ) タブに表示されます。例:

`http://bucket-name.s3-website-region.amazonaws.com`

Amazon S3 静的ウェブサイトエンドポイントを指定する方法の詳細については、Amazon Simple Storage Service ユーザーガイドの「[ウェブサイトエンドポイント](#)」を参照してください。

この形式でバケット名をオリジンとして指定すると、Amazon S3 リダイレクトと Amazon S3 カスタムエラーロードキュメントを使用できません。詳細については、[Amazon Simple Storage Service ユーザーガイド](#)の「[カスタムエラーロードキュメントの設定](#)」および「[リダイレクトの設定](#)」を参照してください。(CloudFront はカスタムエラーページも提供します。詳細については、「」を参照してくださいthe section called “[特定の HTTP ステータスコードに対応するカスタムエラーページの作成](#)”。)

Amazon S3 バケットを CloudFront オリジンサーバーとして使用しても、バケットは変更されません。通常使用しているとおり使用でき、通常の Amazon S3 料金が発生します。の使用料金の詳細については CloudFront、「[Amazon の CloudFront 料金](#)」を参照してください。

Note

CloudFront API を使用してウェブサイトエンドポイントとして設定された Amazon S3 バケットでディストリビューションを作成する場合は、ウェブサイトが Amazon S3 バケットでホストされている場合でも CustomOriginConfig、を使用してディストリビューションを設定する必要があります。Amazon S3 CloudFront API を使用してディストリビューションを作成する方法の詳細については、Amazon API リファレンス [CreateDistribution](#) の「」を参照してください。 CloudFront

既存の Amazon S3 バケット CloudFront への追加

オブジェクトを Amazon S3 バケットに保存する場合、ユーザーに S3 から直接オブジェクトを取得するか、S3 からオブジェクトを取得してユーザーに配信 CloudFront するようにを設定できます。を使用すると CloudFront、データ転送の料金が Amazon S3 CloudFront データ転送の料金よりも低くなるため、ユーザーが頻繁にオブジェクトにアクセスする場合、コスト効率が向上します。さらに、オブジェクトはユーザーの近くに保存されるため、Amazon S3 のみの場合 CloudFront よりものダウンロードが高速です。

Note

Amazon S3 の Cross-Origin Resource Sharing 設定を優先 CloudFront する場合は、Originヘッダーを Amazon S3 CloudFront に転送するようにを設定します。Amazon S3

詳細については、「[the section called “リクエストヘッダーに基づくコンテンツのキャッシュ”](#)」を参照してください。

現在、Amazon S3 バケットのドメイン名 (example.com など) の代わりに独自のドメイン名 (DOC-EXAMPLE-BUCKET.s3.us-west-2.amazonaws.com など) を使用して Amazon S3 バケットから直接コンテンツを配信している場合は、次の手順を使用して中断することなく を追加できます CloudFront。

Amazon S3 からコンテンツをすでに配信している CloudFront ときに を追加するには

1. CloudFront ディストリビューションを作成します。詳細については、「[the section called “ディストリビューションを作成するためのステップ”](#)」を参照してください。

ディストリビューションを作成するときに、オリジンサーバーとして Amazon S3 バケットの名前を指定します。

⚠ Important

バケットで を使用するには CloudFront、名前が DNS 命名要件に準拠している必要があります。詳細については、Amazon Simple Storage Service ユーザーガイドの「[バケットの命名規則](#)」を参照してください。

Amazon S3 で CNAME を使用している場合、ディストリビューションにもその CNAME を指定します。

2. Amazon S3 バケット内にあるパブリックに読み出し可能なオブジェクトへのリンクが含まれているテストウェブページを作成して、リンクをテストします。この初期テストでは、オブジェクト URL にディストリビューションの CloudFront ドメイン名 (例: `https://d111111abcdef8.cloudfront.net/images/image.jpg`) を使用します。

CloudFront URLs 「」を参照してください [the section called “ファイルの URL のカスタマイズ”](#)。

3. Amazon S3 の CNAME を使用している場合、アプリケーションでは、バケットの名前 (例: DOC-EXAMPLE-BUCKET.s3.amazonaws.com) を使用する代わりにお客様のドメイン名 (例: example.com) を使用して、Amazon S3 バケットのオブジェクトを参照しています。ディストリビューションのドメイン名 (d111111abcdef8.cloudfront.net など) を使用する代わりに、CloudFront 引き続きドメイン名を使用してオブジェクトを参照するには、DNS サービスプロバイダーで設定を更新する必要があります。

Amazon S3 の CNAME が機能するためには、DNS サービスプロバイダーに、現在ドメインに対するクエリを Amazon S3 バケットにルーティングしているドメインの CNAME リソースレコードセットが必要です。たとえば、ユーザーが次のオブジェクトをリクエストした場合、

```
https://example.com/images/image.jpg
```

このリクエストは自動的に再ルーティングされ、次のオブジェクトがユーザーに表示されます。

```
https://DOC-EXAMPLE-BUCKET.s3.amazonaws.com/images/image.jpg
```

Amazon S3 バケットではなく CloudFront デイストリビューションにクエリをルーティングするには、DNS サービスプロバイダーが提供する方法を使用して、ドメインの CNAME リソースレコードセットを更新する必要があります。この更新された CNAME レコードは、DNS クエリをドメインからデイストリビューションの CloudFront ドメイン名にリダイレクトします。詳細については、DNS サービスプロバイダーから提供されたドキュメントを参照してください。

Note

Route 53 を DNS サービスとして使用している場合は、CNAME リソースレコードセットまたはエイリアスリソースレコードセットを使用できます。リソースレコードセットの編集については、「[レコードの編集](#)」を参照してください。エイリアスリソースレコードセットの詳細については、「[エイリアスおよび非エイリアスリソースレコードの選択](#)」を参照してください。どちらのトピックも、Amazon Route 53 開発者ガイドにあります。

での CNAMEs 「」を参照してください [the section called “カスタム URL の使用”](#)。CloudFront

CNAME リソースレコードセットを更新してから変更が DNS システム全体に伝達されるまで最大で 72 時間かかりますが、通常は、それよりも早く終了します。この間、コンテンツに対する一部のリクエストは引き続き Amazon S3 バケットにルーティングされ、その他のリクエストはにルーティングされます CloudFront。

Amazon S3 バケットを別の AWS リージョンに移動する

CloudFront デイストリビューションのオリジンとして Amazon S3 を使用していて、バケットを別のに移動する場合 AWS リージョン、次の条件の両方に当てはまる場合、新しいリージョンを使用するようにレコードを更新するまでに最大 1 時間かかる CloudFront ことがあります。

- CloudFront オリジンアクセスアイデンティティ (OAI) を使用してバケットへのアクセスを制限している。
- バケットの移動先の Amazon S3 リージョンで認証に署名バージョン 4 が要求される

OAI、 はリージョン (特に値) CloudFront を使用して、バケットからオブジェクトをリクエストするために使用する署名を計算します。OAI の詳細については、「[the section called “オリジンアクセスアイデンティティの使用 \(レガシー、非推奨\)”](#)」を参照してください。署名バージョン 2 をサポートする AWS リージョンのリストについては、「Amazon Web Services 全般のリファレンス」の「[署名バージョン 2 の署名プロセス](#)」を参照してください。

CloudFrontのレコードをより速く更新するには、CloudFront コンソールの 全般タブの説明フィールドを更新するなどして、ディストリビューションを更新できます CloudFront。ディストリビューションを更新すると、 はバケットがあるリージョンを CloudFront 直ちに確認します。すべてのエッジロケーションへの変更の伝達には数分しかかかりません。

MediaStore コンテナまたは MediaPackage チャンネルの使用

を使用してビデオをストリーミングするには CloudFront、コンテナとして MediaStore設定された Amazon S3 バケットを設定するか、 でチャンネルとエンドポイントを作成します MediaPackage。次に、 でディストリビューションを作成して設定し CloudFront、ビデオをストリーミングします。

詳細と step-by-step 手順については、以下のトピックを参照してください。

- [the section called “AWS Elemental MediaStore をオリジンとして使用する動画の配信”](#)
- [the section called “AWS Elemental MediaPackage でフォーマットされたライブ動画の配信”](#)

Application Load Balancer の使用

オリジンが 1 つ以上の Amazon EC2 インスタンスでホストされている 1 つ以上の HTTP サーバー (ウェブサーバー) である場合、Application Load Balancer を使用してインスタンスにトラフィックを分散できます。Application Load Balancer を のオリジンとして使用方法の詳細については CloudFront、「」を参照してください。ロードバランサーに直接アクセスして CloudFront ではなく、ビューワが を介してのみウェブサーバーにアクセスできることを確認する方法などです [the section called “Application Load Balancers へのアクセスを制限する”](#)。

Lambda 関数 URL の使用

[Lambda 関数 URL](#) は、AWS Lambda 関数専用の HTTPS エンドポイントです。Lambda 関数 URL を使用して、サーバーレスウェブアプリケーションを完全に AWS Lambda 内で構築できます。API Gateway または Application Load Balancer と統合する必要なく、関数 URL を介して Lambda ウェブアプリケーションを直接呼び出すことができます。

関数 URLs、 を追加して次の利点 CloudFront を得ることができます。

- コンテンツを視聴者の近くにキャッシュしてアプリケーションを高速化する
- ウェブアプリケーションのカスタムドメイン名を使用する
- CloudFront キャッシュ動作を使用して異なる Lambda 関数に異なる URL パスをルーティングする
- CloudFront 地理的制限または AWS WAF (またはその両方) を使用して特定のリクエストをブロックする
- AWS WAF と CloudFront を併用することで、悪意のあるボットからアプリケーションを保護し、一般的なアプリケーションの悪用を防ぎ、DDoS 攻撃からの保護を強化できます。

Lambda 関数 URL を CloudFront デイストリビューションのオリジンとして使用するには、Lambda 関数 URL の完全なドメイン名をオリジンドメインとして指定します。Lambda 関数 URL ドメイン名は、次の形式を使用します。

function-URL-ID.lambda-url.AWS-Region.on.aws

Lambda 関数 URL を CloudFront デイストリビューションのオリジンとして使用する場合は、関数 URL がパブリックにアクセス可能であることを確認する必要があります。これを行うには、関数の URL の AuthType パラメータを NONE に設定し、リソースベースのポリシーで `lambda:InvokeFunctionUrl` 許可を付与します。詳細については、「AWS Lambda デベロッパーガイド」の「[NONE の使用 AuthType](#)」を参照してください。ただし、[がオリジンに送信するリクエストにカスタムオリジンヘッダーを追加](#)し、ヘッダーがリクエストに存在しない場合はエラーレスポンスを返す関数コードを記述することもできます。CloudFront これにより、ユーザーは Lambda 関数 URL を直接使用するのではなく CloudFront、 を介してのみウェブアプリケーションにアクセスできるようになります。

Lambda 関数 URL の詳細については、AWS Lambda デベロッパーガイドの以下のトピックを参照してください。

- [Lambda 関数 URL](#) – Lambda 関数 URL 機能の一般的な概要

- [Lambda 関数 URL の呼び出し](#) – サーバーレスウェブアプリケーションのコーディングに使用するリクエストおよびレスポンスのペイロードに関する詳細が含まれます。

Amazon EC2 (または他のカスタムオリジン) の使用

カスタムオリジンは、HTTP サーバーです (例: ウェブサーバー)。HTTP サーバーとして、Amazon EC2 インスタンス、または別の場所でホストしている HTTP サーバーを使用できます。ウェブサイトエンドポイントとして設定された Amazon S3 オリジンは、カスタムオリジンと見なされます。

独自の HTTP サーバーをカスタムオリジンとして使用する場合は、サーバーの DNS 名と、オリジンからオブジェクトを取得するときに CloudFront 使用する HTTP ポート、HTTPS ポート、プロトコルを指定します。

ほとんどの CloudFront 機能は、プライベートコンテンツを除くカスタムオリジンを使用する場合にサポートされます。署名付き URL を使用してカスタムオリジンからコンテンツを配信することはできますが、CloudFront がカスタムオリジンにアクセスするには、オリジンがパブリックにアクセス可能である必要があります。詳細については、「[the section called “署名付き URL と署名付き Cookie を使用したコンテンツの制限”](#)」を参照してください。

で Amazon EC2 インスタンスおよびその他のカスタムオリジンを使用するには、以下のガイドラインに従ってくださいCloudFront。

- 同じ CloudFront オリジンのコンテンツを提供するすべてのサーバーで同じコンテンツをホストし、提供します。詳細については、「[the section called “指定する値”](#)」トピックの「[the section called “オリジンの設定”](#)」を参照してください。
- デバッグにこの値を使用する必要がある場合AWS Supportや使用する必要がある場合は、すべてのサーバーで X-Amz-Cf-IdヘッダーエントリCloudFront をログに記録します。
- カスタムオリジンがリッスンしている HTTP および HTTPS ポートへのリクエストを制限します。
- 実装内のすべてのサーバーの時計を同期します。では、署名付き URLs と署名付き Cookie、ログ、レポートには協定世界時 (UTC) CloudFront が使用されます。さらに、メトリクスを使用して CloudFront CloudWatchアクティビティをモニタリングする場合は、CloudWatch も UTC を使用していることに注意してください。
- 冗長サーバーを使用して障害に対処します。
- カスタムオリジンを使用したプライベートコンテンツ供給の詳細については、「[the section called “カスタムオリジン上のファイルへのアクセス制限”](#)」を参照してください。
- リクエストとレスポンス動作、およびサポートされる HTTP ステータスコードについては、「[リクエストとレスポンスの動作](#)」を参照してください。

カスタムオリジンで Amazon EC2 を使用する場合は、以下の操作を行うことをお勧めします。

- ウェブサーバーのソフトウェアを自動的にインストールする Amazon マシンイメージを使用します。詳細については、「[Amazon EC2 ドキュメント](#)」を参照してください。
- Elastic Load Balancing ロードバランサーを使用して、複数の Amazon EC2 インスタンスにわたるトラフィックを処理するほかに、Amazon EC2 インスタンスの変更からアプリケーションを隔離します。たとえば、ロードバランサーを使用する場合、アプリケーションを変更せずに Amazon EC2 インスタンスの追加と削除ができます。詳細については、「[Elastic Load Balancing ドキュメント](#)」を参照してください。
- CloudFront デイストリビューションを作成するときは、オリジンサーバーのドメイン名にロードバランサーの URL を指定します。詳細については、「[the section called “デイストリビューションの作成”](#)」を参照してください。

CloudFront オリジングループの使用

例えば、高可用性が必要なシナリオでオリジンフェイルオーバーを設定する場合は、CloudFront オリジンにオリジングループを指定できます。オリジンフェイルオーバーを使用して、プライマリオリジン CloudFront と、プライマリオリジンが特定の HTTP ステータスコードの失敗レスポンスを返すときに CloudFront 自動的に切り替わる 2 番目のオリジンを指定します。

オリジングループをセットアップするステップを含む詳細については、「[the section called “オリジンフェイルオーバーによる可用性の向上”](#)」を参照してください。

代替ドメイン名 (CNAME) を追加することによるカスタム URL の使用

では CloudFront、CNAME と呼ばれる代替ドメイン名により、[が](#) デイストリビューション CloudFront に割り当てるドメイン名を使用する代わりに、ファイルの URLs で独自のドメイン名 (www.example.com など) を使用できます。

デイストリビューションを作成すると、[は](#) d111111abcdef8.cloudfront.net などのデイストリビューションのドメイン名 CloudFront を提供します。

cloudfront.net ドメイン名の代わりに独自のドメイン名 (www.example.com など) を使用する場合は、代替ドメイン名をデイストリビューションに追加することができます。

トピック

- [代替ドメイン名の追加](#)
- [代替ドメイン名を別のディストリビューションに移動する](#)
- [代替ドメイン名の削除](#)
- [代替ドメイン名での * ワイルドカードの使用](#)
- [代替ドメイン名を使用するための要件](#)
- [代替ドメイン名の使用に対する制限](#)

代替ドメイン名の追加

次のタスクリストでは、CloudFront コンソールを使用してディストリビューションに代替ドメイン名を追加する方法について説明します。これにより、CloudFront ドメイン名の代わりに独自のドメイン名をリンクで使用できるようになります。CloudFront API を使用してディストリビューションを更新する方法については、「」を参照してください[ディストリビューションの使用](#)。

Note

ビューワーで代替ドメイン名を含む HTTPS を使用する場合は、「[代替ドメイン名と HTTPS の使用](#)」を参照してください。

開始する前に: ディストリビューションを更新して代替ドメイン名を追加する前に、以下の操作必ず実行してください。

- ドメイン名を Route 53 または別のドメインレジストラに登録します。
- ドメイン名を対象とする許可された認定権限 (CA) から SSL/TLS 証明書を取得します。証明書をディストリビューションに追加して、ドメインを使用する権限があることを確認します。詳細については、「[代替ドメイン名を使用するための要件](#)」を参照してください。

代替ドメイン名の追加

1. にサインインAWS Management Consoleし、 で CloudFront コンソールを開きます<https://console.aws.amazon.com/cloudfront/v4/home>。
2. 更新するディストリビューションの ID を選択します。
3. [General] タブで、[Edit] を選択します。
4. 以下の値を更新します。

代替ドメイン名 (CNAME)

代替ドメイン名を追加します。ドメイン名をコンマで区切るか、新しい行にドメイン名を1つずつ入力します。

SSL 証明書

次の設定を選択します。

- HTTPS を使用する – [Custom SSL Certificate (独自 SSL 証明書)] を選択して、リストから証明書を選択します。このリストには、AWS Certificate Manager(ACM)によってプロビジョニングされた証明書、別のCAから購入してACMにアップロードした証明書、および別のCAから購入してIAM証明書ストアにアップロードした証明書が含まれています。

IAM 証明書ストアに証明書をアップロードする場合で、それがリストに表示されない場合は、「[SSL/TLS 証明書をインポートする](#)」の手順を確認して、証明書が正しくアップロードされたことを確認します。

この設定を選択した場合、オブジェクト URL でのみ代替ドメイン名を使用することをお勧めします (<https://www.example.com/logo.jpg>)。CloudFront デイストリビューションドメイン名 (<https://d1111111abcdef8.cloudfront.net.cloudfront.net/logo.jpg>) を使用する場合、サポートされるクライアントに選択した値に応じて、ビューワーが次のように動作することがあります。

- すべてのクライアント: ビューワーが SNI をサポートしていない場合、CloudFront ドメイン名が TLS/SSL 証明書のドメイン名と一致しないため、警告が表示されます。
- Server Name Indication (SNI) をサポートするクライアントのみ: オブジェクトを返さずにビューワーとの接続を CloudFront ドロップします。

Clients Supported (サポートされるクライアント)

次のいずれかのオプションを選択します。

- すべてのクライアント: 専用 IP アドレスを使用して HTTPS コンテンツ CloudFront を提供します。このオプションを選択した場合、有効になっているデイストリビューションに SSL/TLS 証明書を関連付けると、追加料金がかかります。詳細については、「[Amazon CloudFront 料金表](#)」を参照してください。
- [Only Clients that Support Server Name Indication (SNI) (Server Name Indication (SNI) をサポートしているクライアントのみ) (推奨)]: SNI をサポートしていない旧式のブラウザやクライアントでは、別の方法を使用してコンテンツにアクセスする必要があります。

詳細については、「[が HTTPS リクエスト CloudFront を処理する方法の選択](#)」を参照してください。

5. [Yes, Edit (はい、編集します)] を選択します。
6. ディストリビューションの [General (全般)] タブで、[Distribution Status (ディストリビューションのステータス)] が [Deployed (デプロイ済み)] に変わっていることを確認します。ディストリビューションに対する更新がデプロイされる前に代替ドメインの使用を試みた場合、以下のステップで作成するリンクは機能しません。
7. 代替ドメイン名 (www.example.com など) の DNS サービスを設定し、トラフィックをディストリビューションの CloudFront ドメイン名 (d111111abcdef8.cloudfront.net など) にルーティングします。使用する方法は、ドメインの DNS サービスプロバイダーとして、または別のプロバイダーとして Route 53 を使用しているかどうかによって異なります。

Note

DNS レコードが既に指しているのが、更新中のディストリビューション以外のディストリビューションである場合、DNS を更新した後のみディストリビューションに代替ドメイン名を追加できます。詳細については、「[代替ドメイン名の使用に対する制限](#)」を参照してください。

Route 53

エイリアスリソースレコードセットを作成します。エイリアスリソースレコードセットを使用した場合、Route 53 クエリに対する料金はかかりません。また、ルートドメイン名 (example.com) に対してエイリアスリソースレコードセットを作成することもできます。DNS では CNAME の使用が許可されていません。詳細については、「[Amazon Route 53 デベロッパーガイド](#)」の「[ドメイン名を使用したトラフィックの Amazon CloudFront ウェブディストリビューションへのルーティング](#)」を参照してください。

別の DNS サービスプロバイダー

DNS サービスプロバイダーが提供する方法を使用して、ドメインの CNAME レコードを追加します。この新しい CNAME レコードは、代替ドメイン名 (www.example.com など) からディストリビューションの CloudFront ドメイン名 (d111111abcdef8.cloudfront.net など) に DNS クエリをリダイレクトします。詳細については、DNS サービスプロバイダーから提供されたドキュメントを参照してください。

⚠ Important

代替ドメイン名の既存の CNAME レコードがある場合は、そのレコードを更新するか、ディストリビューションの CloudFront ドメイン名を指す新しいレコードに置き換えます。

8. dig などの DNS ツールを使用して、前のステップで作成した DNS 設定がディストリビューションのドメイン名を指していることを確認します。

以下の例は、www.example.com ドメインへの dig リクエストと、応答のうちの関連する部分を示しています。

```
PROMPT> dig www.example.com

; <<> DiG 9.3.3rc2 <<> www.example.com
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15917
;; flags: qr rd ra; QUERY: 1, ANSWER: 9, AUTHORITY: 2, ADDITIONAL: 0

;; QUESTION SECTION:
;www.example.com.      IN      A

;; ANSWER SECTION:
www.example.com. 10800 IN CNAME d111111abcdef8.cloudfront.net.
...
```

回答セクションには、www.example.com のクエリを CloudFront ディストリビューションドメイン名 d111111abcdef8.cloudfront.net にルーティングする CNAME レコードが表示されます。の右側の名前 CNAME が CloudFront ディストリビューションのドメイン名である場合、CNAME レコードは正しく設定されています。その名前が、Amazon S3 バケットのドメイン名などの別の値であれば、CNAME レコードの設定が間違っています。この場合、ステップ 7 に戻り、ディストリビューションのドメイン名を指すように CNAME レコードを修正します。

9. ディストリビューションの CloudFront ドメイン名ではなくお客様のドメイン名を持つ URL にアクセスして、代替ドメイン名をテストします。
10. アプリケーションで、ディストリビューションのドメイン名の代わりに代替ドメイン名を使用するようにオブジェクトの URLs を変更します。

代替ドメイン名を別のディストリビューションに移動する

ディストリビューションに代替ドメイン名を追加しようとしても、代替ドメイン名が別のディストリビューションで既に使用されている場合は、CNAMEAlreadyExists エラー (指定した 1 つ以上の CNAME が既に別のリソースに関連付けられています) が発生します。例えば、www.example.com をディストリビューションに追加しようとしても、www.example.com が別のディストリビューションに既に関連付けられている場合は、このエラーが発生します。

その場合、既存の代替ドメイン名を 1 つのディストリビューション (ソースディストリビューション) から別のディストリビューション (ターゲットディストリビューション) に移動できます。プロセスの概要を次のステップに示します。詳細については、概要の各ステップにあるリンクを参照してください。

代替ドメイン名を移動するには

1. ターゲットディストリビューションを設定します。このディストリビューションには、移動する代替ドメイン名を対象とする SSL/TLS 証明書が必要です。詳細については、「[ターゲットディストリビューションを設定する](#)」を参照してください。
2. ソースディストリビューションを検索します。AWS Command Line Interface (AWS CLI) を使用して、代替ドメイン名が関連付けられているディストリビューションを検索できます。詳細については、「[ソースディストリビューションを検索する](#)」を参照してください。
3. 代替ドメイン名を移動します。これを行う方法は、ソースディストリビューションとターゲットディストリビューションが同じ AWS アカウントにあるかどうかによって異なります。詳細については、「[the section called “代替ドメイン名を移動する”](#)」を参照してください。

ターゲットディストリビューションを設定する

代替ドメイン名を移動する前に、ターゲットディストリビューション (代替ドメイン名の移動先のディストリビューション) を設定する必要があります。

ターゲットディストリビューションを設定するには

1. 移動する代替ドメイン名を含む SSL/TLS 証明書を取得します。お持ちでない場合は、[AWS Certificate Manager \(ACM\)](#) にリクエストするか、別の認定権限 (CA) から取得して ACM にインポートします。米国東部 (バージニア北部) (us-east-1) リージョンの証明書をリクエストまたはインポートしていることを確認します。

- ターゲットディストリビューションを作成していない場合は、新しく作成します。ターゲットディストリビューションの作成の一環として、(前のステップの) 証明書をディストリビューションに関連付けます。詳細については、「[ディストリビューションの作成](#)」を参照してください。

ターゲットディストリビューションを既に作成している場合は、(前のステップの) 証明書をターゲットディストリビューションに関連付けます。詳細については、「[ディストリビューションの更新](#)」を参照してください。

- 代替ドメイン名をターゲットディストリビューションのディストリビューションドメイン名に関連付ける DNS TXT レコードを作成します。代替ドメイン名の前にアンダースコア (_) を付けて TXT レコードを作成します。次に、DNS の TXT レコードの例を示します。

```
_www.example.com TXT d111111abcdef8.cloudfront.net
```

CloudFront はこの TXT レコードを使用して、代替ドメイン名の所有権を検証します。

ソースディストリビューションを検索する

1 つのディストリビューションから別のディストリビューションに代替ドメイン名を移動する前に、ソースディストリビューション (代替ドメイン名が現在使用されているディストリビューション) を検索する必要があります。ソースディストリビューションとターゲットディストリビューションの両方の AWS アカウント ID が分かっている場合は、代替ドメイン名の移動方法を決定できます。

代替ドメイン名のソースディストリビューションを検索するには

- 次の例に示すように [CloudFront list-conflicting-aliases](#)、[AWS Command Line Interface \(AWS CLI\)](#) で [コマンド](#) を使用します。 `www.example.com` を代替ドメイン名に置き換え、 `EDFDVBD6EXAMPLE` を、 [以前に設定した](#) ターゲットディストリビューションの ID に置き換えます。ターゲットディストリビューションと同じ AWS アカウントにある認証情報を使用して、このコマンドを実行します。このコマンドを使用するには、ターゲットディストリビューションに対する `cloudfront:GetDistribution` と `cloudfront:ListConflictingAlias` アクセス許可が必要です。

```
aws cloudfront list-conflicting-aliases --alias www.example.com --distribution-id EDFDVBD6EXAMPLE
```

コマンドの出力には、指定されたドメイン名と競合または重複するすべての代替ドメイン名のリストが表示されます。次に例を示します。

- コマンドに `www.example.com` を指定すると、コマンドの出力には `www.example.com` および、重複するワイルドカード付き代替ドメイン名 (`*.example.com`) が含まれます。
- コマンドに `*.example.com` を指定すると、コマンドの出力には、`*.example.com` と、そのワイルドカードでカバーされる代替ドメイン名 (例えば、`www.example.com`、`test.example.com`、`dev.example.com` など) が含まれます。

コマンドの出力内の代替ドメイン名ごとに、それが関連付けられているディストリビューションの ID と、ディストリビューションを所有する AWS アカウント ID を確認できます。ディストリビューション ID とアカウント ID は部分的に非表示になっているため、所有しているディストリビューションとアカウントを識別できますが、所有していないディストリビューションとアカウントの情報は保護されます。

2. コマンドの出力から、移動する代替ドメイン名のディストリビューションを検索し、ソースディストリビューションの AWS アカウント ID を書き留めます。ソースディストリビューションのアカウント ID とターゲットディストリビューションを作成したアカウント ID を比較し、これら 2 つのディストリビューションが同じ AWS アカウントにあるかどうかを判断します。これは、代替ドメイン名の移動方法を決定するのに役立ちます。

代替ドメイン名の移動については、次のトピックを参照してください。

代替ドメイン名を移動する

代替ドメインの移動方法については、状況に応じて、次の方法から選択します。

ソースディストリビューションとターゲットディストリビューションが同じ AWS アカウントにある場合

AWS CLI の `associate-alias` コマンドを使用して、代替ドメイン名を移動します。この方法は、代替ドメイン名が apex ドメイン (ルートドメインとも呼ばれます。例: `example.com`) である場合を含めて、同じアカウントでのすべての移動で機能します。詳細については、「[the section called “associate-alias を使用して代替ドメイン名を移動する”](#)」を参照してください。

ソースディストリビューションとターゲットディストリビューションが異なる AWS アカウントにある場合

ソースディストリビューションにアクセスでき、代替ドメイン名が apex ドメイン (ルートドメインとも呼ばれます。例: `example.com`) 以外のもので、その代替ドメイン名と重複するワイルドカードをまだ使用していない場合は、ワイルドカードを使用して代替ドメイン名を移動します。

詳細については、「[the section called “ワイルドカードを使用して代替ドメイン名を移動する”](#)」を参照してください。

ソースディストリビューションの AWS アカウントにアクセスできない場合は、AWS CLI の `associate-alias` コマンドを使用して代替ドメイン名を移動します。ソースディストリビューションが無効になっている場合は、代替ドメイン名を移動できます。詳細については、「[the section called “associate-alias を使用して代替ドメイン名を移動する”](#)」を参照してください。 `associate-alias` コマンドが機能しない場合は、AWS Support に問い合わせてください。詳細については、「[the section called “AWS Support にお問い合わせいただき、代替ドメイン名を移動する”](#)」を参照してください。

associate-alias を使用して代替ドメイン名を移動する

ソースディストリビューションがターゲットディストリビューションと同じAWSアカウントにある場合、または別のアカウントにあるが無効な場合は、[CloudFront associate-alias のコマンドAWS CLI](#)を使用して代替ドメイン名を移動できます。

関連エイリアスを使用して代替ドメイン名を移動するには

1. 次の例に示すように、を使用してコマンドAWS CLIを実行します CloudFront `associate-alias`。 `www.example.com` を代替ドメイン名に、 `EDFDVBD6EXAMPLE` をターゲットディストリビューションの ID に置き換えます。ターゲットディストリビューションと同じ AWS アカウントにある認証情報を使用して、このコマンドを実行します。このコマンドの使用には、次の制約があることに注意してください。
 - ターゲットディストリビューションに対する `cloudfront:AssociateAlias` と `cloudfront:UpdateDistribution` アクセス許可が必要です。
 - ソースディストリビューションとターゲットディストリビューションが同じ AWS アカウントにある場合、ソースディストリビューションに対する `cloudfront:UpdateDistribution` アクセス許可が必要です。
 - ソースディストリビューションとターゲットディストリビューションが異なる AWS アカウントにある場合、ソースディストリビューションを無効にする必要があります。
 - ターゲットディストリビューションは、「[the section called “ターゲットディストリビューションを設定する”](#)」で説明されているように設定する必要があります。

```
aws cloudfront associate-alias --alias www.example.com --target-distribution-id EDFDVBD6EXAMPLE
```

このコマンドは、ソースディストリビューションから代替ドメイン名を削除し、ターゲットディストリビューションに追加することで、両方のディストリビューションを更新します。

2. ターゲットディストリビューションが完全にデプロイされたら、代替ドメイン名の DNS レコードがターゲットディストリビューションのディストリビューションドメイン名を参照するように DNS 構成を更新します。

ワイルドカードを使用して代替ドメイン名を移動する

ソースディストリビューションがターゲットディストリビューションと異なる AWS アカウントにあり、ソースディストリビューションが有効になっている場合は、ワイルドカードを使用して代替ドメイン名を移動できます。

Note

ワイルドカードを使用して apex ドメイン (example.com など) を移動することはできません。ソースディストリビューションとターゲットディストリビューションが異なる AWS アカウントにある場合の apex ドメインの移動については、AWS Support にお問い合わせください。詳細については、「[the section called “AWS Support にお問い合わせいただき、代替ドメイン名を移動する”](#)」を参照してください。

ワイルドカードを使用して代替ドメイン名を移動するには

Note

このプロセスでは、ディストリビューションを複数回更新します。各ディストリビューションが最新の変更を完全にデプロイするまで待つから次のステップに進みます。

1. ターゲットディストリビューションを更新して、移動する代替ドメイン名に対応したワイルドカード付き代替ドメイン名を追加します。例えば、移動する代替ドメイン名を *www.example.com* にしている場合は、代替ドメイン名 **.example.com* をターゲットディストリビューションに追加します。これを行うには、ターゲットディストリビューションの SSL/TLS 証明書にワイルドカード付きのドメイン名を含める必要があります。詳細については、「[the section called “ディストリビューションの更新”](#)」を参照してください。

2. 代替ドメイン名の DNS 設定を更新して、ターゲットディストリビューションのドメイン名を参照します。例えば、移動する代替ドメイン名が `www.example.com` の場合、`www.example.com` の DNS レコードを更新して、トラフィックをターゲットディストリビューションのドメイン名 (例: `d1111111abcdef8.cloudfront.net`) にルーティングします。

 Note

DNS 設定を更新した後でも、代替ドメイン名はソースディストリビューションによって引き続き提供されます。これは、代替ドメイン名がソースディストリビューションで現在設定されているためです。

3. ソースディストリビューションを更新して代替ドメイン名を削除します。詳細については、「[ディストリビューションの更新](#)」を参照してください。
4. ターゲットディストリビューションを更新して代替ドメイン名を追加します。詳細については、「[ディストリビューションの更新](#)」を参照してください。
5. `dig` (または類似の DNS クエリツール) を使用して、代替ドメイン名の DNS レコードの解決先がターゲットディストリビューションのドメイン名になっていることを検証します。
6. (オプション) ターゲットディストリビューションを更新してワイルドカード付き代替ドメイン名を削除します。

AWS Support にお問い合わせいただき、代替ドメイン名を移動する

ソースディストリビューションとターゲットディストリビューションが異なる AWS アカウントにある場合に、ソースディストリビューションの AWS アカウントにアクセスできないか、ソースディストリビューションを無効にできない場合は、AWS Support にお問い合わせいただき代替ドメイン名を移動できます。

AWS Support にお問い合わせいただき、代替ドメイン名を移動するには

1. ターゲットディストリビューションをセットアップします。これには、ターゲットディストリビューションを参照する DNS TXT レコードも含まれます。詳細については、「[ターゲットディストリビューションを設定する](#)」を参照してください。
2. [に連絡してAWS Support](#)、ドメインの所有権を確認し、ドメインを新しい CloudFront ディストリビューションに移動するように依頼します。
3. ターゲットディストリビューションが完全にデプロイされたら、代替ドメイン名の DNS レコードがターゲットディストリビューションのディストリビューションドメイン名を参照するように DNS 構成を更新します。

代替ドメイン名の削除

ドメインまたはサブドメインのトラフィックのディストリビューションへのルーティングを停止する場合は、このセクションの手順に従って DNS 設定と CloudFront ディストリビューションの両方を更新します。

ディストリビューションから代替ドメイン名を削除し、同時に DNS 設定を更新することが重要です。これにより、ドメイン名を別の CloudFront ディストリビューションに関連付ける場合に、後で問題が発生するのを防ぐことができます。代替ドメイン名が既に 1 つのディストリビューションに関連付けられている場合、別のディストリビューションで設定することはできません。

Note

このディストリビューションから代替ドメイン名を削除して、別のディストリビューションに追加するには、「[代替ドメイン名を別のディストリビューションに移動する](#)」の手順に従います。代わりにここに示すステップに従って (ドメインを削除するために)、ドメインを別のディストリビューションに追加した場合、[エッジロケーションの更新](#) CloudFront に反映されるため、ドメインが新しいディストリビューションにリンクされない期間があります。

ディストリビューションから代替ドメイン名を削除するには

1. まず、ドメインのインターネットトラフィックを、Elastic Load Balancing ロードバランサーなど、CloudFront ディストリビューションではない別のリソースにルーティングします。または、[トラフィックをルーティングしている DNS レコードを削除することもできます](#) CloudFront。

ドメインの DNS サービスに応じて、次のいずれかを実行します。

- Route 53 を使用している場合、エイリアスレコードまたは CNAME レコードを更新または削除します。詳細については、「[レコードの編集](#)」または「[レコードの削除](#)」を参照してください。
- 別の DNS サービスプロバイダーを使用している場合は、その DNS サービスプロバイダーが提供している方法を使用して、CloudFront にトラフィックを送っている CNAME レコードを更新または削除します。詳細については、DNS サービスプロバイダーから提供されたドキュメントを参照してください。

2. ドメインの DNS レコードを更新したら、この変更が伝達され、DNS リゾルバーが新しいリソースにトラフィックをルーティングするまで待ちます。URL でドメインを使用するテストリンクを作成することで、以上が完了したことを確認できます。
3. にサインインAWS Management Consoleし、 で CloudFront コンソールを開き <https://console.aws.amazon.com/cloudfront/v4/home>、デイス CloudFront トリビューションを更新してドメイン名を削除します。
 - a. 更新するデイストリビューションの ID を選択します。
 - b. [General] タブで、[Edit] を選択します。
 - c. [Alternate Domain Names (CNAMEs) (代替ドメイン名 (CNAME))] で、デイストリビューションに使用しない代替ドメイン名 (1 つ以上) を削除します。
 - d. [Yes, Edit (はい、編集します)] を選択します。

代替ドメイン名での * ワイルドカードの使用

代替ドメイン名を追加するとき、サブドメインを個別に追加する代わりに、ドメイン名の最初に * ワイルドカードを使用できます。例えば、代替ドメイン名を *.example.com にしている場合は、「www.example.com」、「product-name.example.com」、「marketing.product-name.example.com」などの、末尾が「example.com」であるドメイン名であればどれも URL に使用できます。オブジェクトのパスは、ドメイン名に関係なく同じです。例えば、次のようになります。

- www.example.com/images/image.jpg
- product-name.example.com/images/image.jpg
- marketing.product-name.example.com/images/image.jpg

ワイルドカードを含む代替ドメイン名については、以下の要件に従ってください。

- 代替ドメイン名は、アスタリスクとドット (*) で始まる必要があります。
- *.domain.example.com のように、サブドメイン名の一部をワイルドカードで置き換えることはできません。
- 「subdomain.*.example.com」のように、ドメイン名の途中にあるサブドメインを置き換えることはできません。
- ワイルドカードを使用する代替ドメイン名を含むすべての代替ドメイン名は、証明書のサブジェクト代替名 (SAN) でカバーされている必要があります。

ワイルドカード付き代替ドメイン名 (*.example.com など) には、使用されている別の代替ドメイン名 (example.com など) を含めることができます。

代替ドメイン名を使用するための要件

www.example.com などの代替ドメイン名を CloudFront デイストリビューションに追加する場合、次の要件があります。

代替ドメイン名は小文字を使用する必要があります

すべての代替ドメイン名 (CNAME) には小文字を使用する必要があります。

代替ドメイン名は有効な SSL/TLS 証明書の対象であることが必要です

デイス CloudFront トリビューションに代替ドメイン名 (CNAME) を追加するには、代替ドメイン名を対象とする信頼された有効な SSL/TLS 証明書をデイス トリビューションにアタッチする必要があります。これにより、ドメインの証明書にアクセスできるユーザーのみが CloudFront、ドメインに関連する CNAME に関連付けることができます。

信頼された証明書とは、AWS Certificate Manager (ACM) または別の有効な認証機関 (CA) によって発行される証明書です。自己署名証明書を使用して既存の CNAME を検証できますが、新しい CNAME については検証できません。は Mozilla と同じ認証機関 CloudFront をサポートします。最新のリストは、「[Mozilla に付属する CA 証明書一覧](#)」を参照してください。

ワイルドカードを含む代替ドメイン名を含め、アタッチした証明書を使用して代替ドメイン名を検証するには、は証明書のサブジェクト代替名 (SAN) CloudFront をチェックします。追加する代替ドメイン名は、SAN の対象である必要があります。

Note

一度に 1 つの CloudFront デイストリビューションにアタッチできる証明書は 1 つだけです。

デイス トリビューションに特定の代替ドメイン名を追加する許可があることを証明するには、次のいずれかを実行します。

- 代替ドメイン名 (product-name.example.com など) を含む証明書を添付します。
- ドメイン名の先頭に * ワイルドカードを含む証明書をアタッチして、1 つの証明書で複数のサブドメインを対象とします。ワイルドカードを指定する場合、複数のサブドメインを CloudFront の代替ドメイン名として追加できます。

次の例では、証明書のドメイン名のワイルドカードを使用して特定の代替ドメイン名を CloudFront に追加することを許可する方法を示しています。

- marketing.example.com を代替ドメイン名として追加するとします。証明書にドメイン名 *.example.com をリストします。この証明書を にアタッチすると CloudFront、marketing.example.com トリビューションの代替ドメイン名を追加して、など、そのレベルでワイルドカードを置き換えることができます。また、たとえば次の代替ドメイン名を追加することもできます。
 - product.example.com
 - api.example.com

ただし、ワイルドカードより高いあるいは低い位置に代替ドメイン名を追加することはできません。例えば、代替ドメイン名 example.com や marketing.product.example.com を追加することはできません。

- example.com を代替ドメイン名として追加するとします。これを行うには、ディストリビューションにアタッチした証明書にこのドメイン名 example.com 自体をリストする必要があります。
- marketing.example.com を代替ドメイン名として追加するとします。これを行うには、証明書に *.product.example.com をリストするか、または証明書に marketing.product.example.com 自体をリストできます。

DNS 設定を変更する権限

代替ドメイン名を追加するときは、CNAME レコードを作成して、代替ドメイン名の DNS クエリを CloudFront ディストリビューションにルーティングする必要があります。これを行うには、DNS サービスプロバイダーを使用して使用する代替ドメイン名に CNAME を作成する権限があることが必要です。通常、これはドメインを所有していることを指しますが、ドメイン所有者向けにアプリケーションを開発している場合にも当てはまります。

代替ドメイン名と HTTPS

代替ドメイン名を含む HTTPS をビューワーが使用するように構成する場合は、いくつかの追加設定を実行する必要があります。詳細については、「[代替ドメイン名と HTTPS の使用](#)」を参照してください。

代替ドメイン名の使用に対する制限

代替ドメイン名の使用には、以下の制限があることに注意してください。

代替ドメイン名の最大数

ディストリビューションに対して作成できる代替ドメイン名の現在の最大数、またはクォータの引き上げを要求する代替ドメイン名の最大数については、「[ディストリビューションの一般的なクォータ](#)」を参照してください。

重複する代替ドメイン名

AWS アカウントが他の CloudFront ディストリビューションを所有している場合でも、同じ代替ドメイン名が別のディストリビューションに既に存在する場合、代替ドメイン名をディストリビューションに追加することはできません。

ただし、*.example.com のようなワイルドカード付きの代替ドメイン名を追加できます。これには、www.example.com のような非ワイルドカードの代替ドメイン名を含む (つまり重複している) が含まれます。2 つのディストリビューションで代替ドメイン名が重複している場合、は、DNS レコードが指すディストリビューションに関係なく、より具体的な名前が一致するディストリビューションにリクエスト CloudFront を送信します。例えば、marketing.domain.com は *.domain.com より具体的です。

ドメインフロンティング

CloudFront には、異なる AWS アカウント間で発生するドメインフロンティングに対する保護が含まれます。ドメインフロンティングとは、非標準クライアントが 1 つの AWS アカウントのドメイン名への TLS/SSL 接続を作成し、別の AWS アカウントの関連しない名前の HTTPS リクエストを行うシナリオです。例えば、TLS 接続が www.example.com に接続を行い、次に www.example.org に HTTP リクエストを送信するような場合です。

ドメインフロンティングが異なる AWS アカウントを横断するケースを防ぐため、は、特定の接続にサービスを提供する証明書を所有する AWS アカウントが、その同じ接続で処理するリクエストを所有する AWS アカウントと常に一致する CloudFront ことを確認します。

2 つの AWS アカウント番号が一致しない場合、は HTTP 421 Misdirected Request レスポンスで CloudFront 応答し、正しいドメインを使用して接続する機会をクライアントに付与します。

ドメインのトップノード (zone apex) の代替ドメイン名の追加

ディストリビューションに代替ドメイン名を追加する場合、通常、DNS 設定で CNAME レコードを作成して、ドメイン名の DNS クエリを CloudFront ディストリビューションにルーティングします。ただし、DNS プロトコルでは、zone apex と呼ばれる、DNS 名前空間の最上位ノードに対して CNAME レコードを作成することができません。例えば、example.com という DNS 名を登録する場合、Zone Apex は example.com になります。「example.com」に対して CNAME

レコードを作成することはできませんが、www.example.com、newproduct.example.com などに対しては CNAME レコードを作成できます。

DNS サービスとして Route 53 を使用している場合、エイリアスリソースレコードセットを作成できます。これは、CNAME レコードに比べて 2 つの利点があります。トップノードのドメイン名 (example.com) に対してエイリアスリソースレコードセットを作成することもできます。また、エイリアスリソースレコードセットを使用した場合、Route 53 クエリに対する料金はかかりません。

Note

IPv6 を有効にする場合、2 つのエイリアスリソースレコードセットを作成する必要があります。IPv6 トラフィック (A レコード) をルーティングするため、および IPv4 トラフィック (AAAA レコード) をルーティングするためです。詳細については、トピック「[ディストリビューションを作成または更新するときに指定する値](#)」の「[IPv6 を有効にする](#)」を参照してください。

詳細については、[「Amazon Route 53 デベロッパーガイド」の「ドメイン名を使用したトラフィックの Amazon CloudFront ウェブディストリビューションへのルーティング」](#)を参照してください。

ディストリビューション WebSockets での CloudFront の使用

Amazon は WebSocket、クライアントとサーバー間の存続期間の長い双方向接続が必要な場合に便利な TCP ベースのプロトコルである の使用 CloudFront をサポートしています。永続的な接続は、多くの場合、リアルタイムアプリケーションでの要件です。使用するシナリオ WebSockets には、ソーシャルチャットプラットフォーム、オンラインコラボレーションワークスペース、マルチプレイヤーゲーム、金融取引プラットフォームなどのリアルタイムのデータフィードを提供するサービスなどがあります。WebSocket 接続上のデータは、全二重通信では両方向に流れる可能性があります。

CloudFront は WebSocket、追加の設定を必要とせずに、接続をグローバルにサポートします。クライアントとサーバーの両方が WebSocket プロトコルをサポートしている限り、すべての CloudFront ディストリビューションにはプロトコルサポートが組み込まれています。

WebSocket プロトコルの仕組み

WebSocket プロトコルは、HTTP のオーバーヘッドやレイテンシーの増加を回避する、独立した TCP ベースのプロトコルです。

WebSocket 接続を確立するために、クライアントは HTTP のアップグレードセマンティクスを使用してプロトコルを変更する通常の HTTP リクエストを送信します。その後、サーバーはハンドシェイクを完了できます。WebSocket 接続は開いたままで、クライアントまたはサーバーは毎回新しい接続を確立する必要なく、互いにデータフレームを送信できます。

デフォルトでは、WebSocket プロトコルは通常の WebSocket 接続にはポート 80 を使用し、TLS/SSL 経由 WebSocket の接続にはポート 443 を使用します。と CloudFront [ビューワープロトコルポリシー](#) に選択したオプションは、WebSocket 接続および HTTP トラフィック [プロトコル \(カスタムオリジンのみ\)](#) に適用されます。

WebSocket の要件

WebSocket リクエストは、次の標準形式で [RFC 6455](#) に準拠している必要があります。

サンプルクライアントリクエスト:

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGh1IHNhbXBsZSBub25jZQ==
Origin: https://example.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
```

サンプルサーバー応答:

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+x0o=
Sec-WebSocket-Protocol: chat
```

クライアントまたはサーバー、またはネットワークの中断によって WebSocket 接続が切断された場合、クライアントアプリケーションはサーバーとの接続を再開する必要があります。

推奨される設定

を使用する際に予期しない圧縮関連の問題を回避するには WebSockets、[オリジンリクエストポリシー](#)に次のヘッダーを含めることをお勧めします。

- Sec-WebSocket-Key
- Sec-WebSocket-Version
- Sec-WebSocket-Protocol
- Sec-WebSocket-Accept
- Sec-WebSocket-Extensions

ポリシーの使用

Amazon CloudFront では、CloudFront 次の方法でカスタマイズできる 3 種類のポリシーを提供しています。

キャッシュの設定と圧縮の設定を指定する

CloudFront キャッシュポリシー では、キャッシュキー CloudFront に含まれる HTTP ヘッダー、Cookie、クエリ文字列を指定できます。キャッシュキーは、ビューワの HTTP リクエストによってキャッシュヒットが発生するかどうかを決定します (オブジェクトは CloudFront キャッシュからビューワに提供されます)。キャッシュキーに含まれる値が少なくなると、キャッシュヒットの可能性が高まります。

また、キャッシュポリシーを使用して、CloudFront キャッシュ内のオブジェクトの有効期限 (TTL) 設定を指定し、で圧縮オブジェクト CloudFront をリクエストおよびキャッシュすることもできます。

オリジンリクエスト (キャッシュキーではなく) に含める値を指定する

CloudFront オリジンリクエストポリシー では、オリジンリクエスト CloudFront に含まれる HTTP ヘッダー、Cookie、クエリ文字列を指定できます。これらは、キャッシュミスが発生したときに がオリジン CloudFront に送信するリクエストです。

キャッシュポリシーのすべての値はオリジンリクエストに自動的に含まれますが、オリジンリクエストポリシーを使用して、それらをキャッシュキーに含めずにオリジンリクエストに追加の値を含めることができます。

ビューワレスポンスで追加または削除する HTTP ヘッダーを指定する

CloudFront レスponseヘッダーポリシー を使用すると、ビューワ (ウェブブラウザまたは他のクライアント) に送信する HTTP レスponseに CloudFront が含まれる HTTP ヘッダーを制御できます。オリジンを変更したりコードを記述したりすることなく、オリジンの HTTP レスponseからヘッダーを削除したり、 がビューワ CloudFront に送信するレスponseに HTTP ヘッダーを追加したりできます。

詳細については、以下のトピックを参照してください。

トピック

- [the section called “キャッシュキーの管理”](#)

- [the section called “オリジンリクエストの制御”](#)
- [レスポンスヘッダーの追加または削除](#)

キャッシュキーの管理

Amazon では CloudFront、CloudFront エッジロケーションにキャッシュされるオブジェクトのキャッシュキーを制御できます。キャッシュキーは、キャッシュ内のすべてのオブジェクトの一意的識別子であり、ビューワーリクエストによってキャッシュヒットが発生するかどうかを決定します。キャッシュヒットが発生するのは、ビューワーリクエストが以前のリクエストと同じキャッシュキーを生成し、そのキャッシュキーのオブジェクトがエッジロケーションのキャッシュにあり、有効な場合です。キャッシュヒットが発生すると、オブジェクトは CloudFront エッジロケーションからビューワーに提供されます。これには次の利点があります。

- オリジンサーバーの負荷を軽減
- ビューワーのレイテンシーを低減

キャッシュヒット率が高い (キャッシュヒットにつながるビューワーリクエストの割合が高い) ほど、ウェブサイトやアプリケーションのパフォーマンスが高くなります。キャッシュヒット率を改善する 1 つの方法は、キャッシュキーに必要な最小値のみを含めることです。詳細については、「[キャッシュキーについて](#)」を参照してください。

キャッシュキーを制御するには、CloudFront キャッシュポリシーを使用します。キャッシュポリシーは、CloudFront デイストリビューションの 1 つ以上のキャッシュ動作にアタッチします。

トピック

- [キャッシュポリシーの作成](#)
- [キャッシュポリシーについて](#)
- [管理キャッシュポリシーの使用](#)
- [キャッシュキーについて](#)

キャッシュポリシーの作成

キャッシュポリシーを使用して、キャッシュキーに含まれる値 (URL クエリ文字列、HTTP ヘッダー、Cookie) を管理することで、キャッシュヒット率を改善できます。キャッシュポリシーは、CloudFront コンソール、AWS Command Line Interface (AWS CLI)、または CloudFront API を使用して作成できます。

キャッシュポリシーを作成したら、デイス CloudFront トリビューションの 1 つ以上のキャッシュ動作にアタッチします。

Console

キャッシュポリシーを作成するには (コンソール)

1. にサインインAWS Management Consoleし、 CloudFront コンソールのポリシーページを開きます<https://console.aws.amazon.com/cloudfront/v4/home?#/policies>。
2. [キャッシュポリシーの作成] を選択します。
3. このキャッシュポリシーに目的の設定を選択します。詳細については、「[キャッシュポリシーについて](#)」を参照してください。
4. 終了したら、[作成] を選択します。

キャッシュポリシーを作成したら、それをキャッシュ動作にアタッチできます。

既存のデイス トリビューションにキャッシュポリシーをアタッチするには (コンソール)

1. の CloudFront コンソールでデイス トリビューションページを開きます<https://console.aws.amazon.com/cloudfront/v4/home#/distributions>。
2. 更新するデイス トリビューションを選択し、[動作] タブを選択します。
3. 更新するキャッシュ動作を選択し、[編集] を選択します。

または、新しいキャッシュ動作を作成するには、[動作を作成] を選択します。

4. [キャッシュキーとオリジンリクエスト] セクションで、[キャッシュポリシーとオリジンリクエストポリシー] が選択されていることを確認します。
5. [キャッシュポリシー] では、このキャッシュ動作にアタッチするキャッシュポリシーを選択します。
6. ページの最下部で [変更の保存] を選択します。

新しいデイス トリビューションにキャッシュポリシーをアタッチするには (コンソール)

1. で CloudFront コンソールを開きます<https://console.aws.amazon.com/cloudfront/v4/home>。
2. [デイス トリビューションの作成] を選択します。
3. [キャッシュキーとオリジンリクエスト] セクションで、[キャッシュポリシーとオリジンリクエストポリシー] が選択されていることを確認します。

4. [Cache policy] (キャッシュポリシー) で、このディストリビューションのデフォルトのキャッシュ動作にアタッチするキャッシュポリシーを選択します。
5. オリジン、デフォルトのキャッシュ動作、その他のディストリビューション設定に必要な設定を選択します。詳細については、「[ディストリビューションを作成または更新する場合に指定する値](#)」を参照してください。
6. 終了したら、[ディストリビューションの作成] を選択します。

CLI

AWS Command Line Interface (AWS CLI) でキャッシュポリシーを作成するには、aws cloudfront create-cache-policy コマンドを使用します。コマンドの入力パラメータは、コマンドライン入力として個別に指定せずに、入力ファイルを使用して指定できます。

キャッシュポリシーを作成するには (入力ファイルを含む CLI)

1. 次のコマンドを使用して、cache-policy.yaml コマンドのすべての入力パラメータを含む create-cache-policy という名前のファイルを作成します。

```
aws cloudfront create-cache-policy --generate-cli-skeleton yaml-input > cache-policy.yaml
```

2. 先ほど作成した cache-policy.yaml という名前のファイルを開きます。ファイルを編集して、必要なキャッシュポリシー設定を指定し、ファイルを保存します。ファイルからオプションのフィールドを削除することはできますが、必須フィールドは削除しないでください。

キャッシュポリシー設定の詳細については、「[キャッシュポリシーについて](#)」を参照してください。

3. 次のコマンドを使用して、cache-policy.yaml ファイルの入力パラメータを使用し、キャッシュポリシーを作成します。

```
aws cloudfront create-cache-policy --cli-input-yaml file://cache-policy.yaml
```

コマンドの出力の Id 値を書き留めます。これはキャッシュポリシー ID であり、ディストリビューションのキャッシュ動作にキャッシュポリシーをアタッチするために必要です。

既存のディストリビューションにキャッシュポリシーをアタッチするには (入力ファイルを含む CLI)

1. 次のコマンドを使用して、更新するディストリビューションの CloudFront ディストリビューション設定を保存します。 *distribution_ID* をディストリビューションの ID に置き換えます。

```
aws cloudfront get-distribution-config --id distribution_ID --output yaml >
dist-config.yaml
```

2. 先ほど作成した `dist-config.yaml` という名前のファイルを開きます。ファイルを編集し、キャッシュポリシーを使用するように更新する各キャッシュ動作に次の変更を加えます。
 - キャッシュ動作で、`CachePolicyId` という名前のフィールドを追加します。フィールドの値には、ポリシーの作成後に書き留めたキャッシュポリシー ID を使用します。
 - キャッシュ動作から `MinTTL`、`MaxTTL`、`DefaultTTL`、および `ForwardedValues` フィールドを削除します。これらの設定はキャッシュポリシー内に指定するため、これらのフィールドとキャッシュポリシーを同じキャッシュ動作に含めることはできません。
 - `Etag` フィールドの名前を `IfMatch` に変更します。ただし、フィールドの値は変更しないでください。

完了したら、ファイルを保存します。

3. キャッシュポリシーを使用するようにディストリビューションを更新するには、次のコマンドを使用します。 *distribution_ID* をディストリビューションの ID に置き換えます。

```
aws cloudfront update-distribution --id distribution_ID --cli-input-yaml file://
dist-config.yaml
```

新しいディストリビューションにキャッシュポリシーをアタッチするには (入力ファイルを含む CLI)

1. 次のコマンドを使用して、`distribution.yaml` コマンドのすべての入力パラメータを含む `create-distribution` という名前のファイルを作成します。

```
aws cloudfront create-distribution --generate-cli-skeleton yml-input >
distribution.yml
```

2. 先ほど作成した `distribution.yml` という名前のファイルを開きます。デフォルトのキャッシュ動作の `[CachePolicyId]` フィールドに、ポリシーの作成後に書き留めたキャッシュポリシー ID を入力します。ファイルの編集を続行して必要なディストリビューション設定を指定し、完了したらファイルを保存します。

ディストリビューション設定の詳細については、「[ディストリビューションを作成または更新する場合に指定する値](#)」を参照してください。

3. 次のコマンドを使用して、`distribution.yml` ファイルの入力パラメータを使用し、ディストリビューションを作成します。

```
aws cloudfront create-distribution --cli-input-yml file://distribution.yml
```

API

CloudFront API を使用してキャッシュポリシーを作成するには、[aws cloudfront create-cache-policy](#) を使用します。この API コールで指定するフィールドの詳細については、「[キャッシュポリシーについて](#)」、および AWS SDK やその他の API クライアントの API リファレンスドキュメントを参照してください。

キャッシュポリシーを作成したら、次の API コールのいずれかを使用して、それをキャッシュ動作にアタッチできます。

- 既存のディストリビューションのキャッシュ動作にアタッチするには、[aws cloudfront update-distribution](#) を使用します。
- 新しいディストリビューションのキャッシュ動作にアタッチするには、[aws cloudfront create-distribution](#) を使用します。

これらの API コールの両方について、キャッシュ動作内で、`CachePolicyId` フィールドにキャッシュポリシーの ID を指定します。これらの API コールで指定するその他フィールドの詳細については、「[ディストリビューションを作成または更新する場合に指定する値](#)」と、AWS SDK またはその他 API クライアントの API リファレンスドキュメントを参照してください。

キャッシュポリシーについて

キャッシュポリシーを使用して、キャッシュキーに含まれる値 (URL クエリ文字列、HTTP ヘッダー、Cookie) を制御することで、キャッシュヒット率を向上させることができます。CloudFront には、一般的なユースケース用に、管理ポリシーと呼ばれる事前定義されたキャッシュポリシーがいくつか用意されています。これらのマネージドポリシーを使用することも、ユーザーのニーズ別に独自のキャッシュポリシーを作成することもできます。マネージドポリシーの詳細については、「[管理キャッシュポリシーの使用](#)」を参照してください。

キャッシュポリシーには以下の設定が含まれます。設定は、ポリシー情報、Time to live (TTL) 設定、およびキャッシュキー設定に分類されます。

ポリシー情報

名前

キャッシュポリシーを特定する名前。コンソールでは、名前を使用して、キャッシュポリシーをキャッシュ動作にアタッチします。

説明

キャッシュポリシーを説明するコメント。これはオプションですが、キャッシュポリシーの目的を特定するのに役立ちます。

Time to live (TTL) 設定

有効期限 (TTL) 設定は、Cache-Control および HTTP Expires ヘッダー (オリジンレスポンスにある場合) と連携して、CloudFront キャッシュ内のオブジェクトが有効である期間を決定します。

最小 TTL

がオリジンをチェックしてオブジェクトが更新されたかどうかを確認するまでに、オブジェクトを CloudFront キャッシュに保持する最小時間を秒単位で CloudFront 指定します。詳細については、「[コンテンツがキャッシュに保持される期間 \(有効期限\) の管理](#)」を参照してください。

最大 TTL

がオリジン CloudFront をチェックしてオブジェクトが更新されたかどうかを確認するまでに、そのオブジェクトがキャッシュに CloudFront 保持される秒単位の最大時間。は、オリジンがオブジェクトで Cache-Control または Expires ヘッダーを送信する場合にのみ、この設定

CloudFront を使用します。詳細については、「[コンテンツがキャッシュに保持される期間 \(有効期限\) の管理](#)」を参照してください。

デフォルト TTL

がオリジン CloudFront をチェックしてオブジェクトが更新されたかどうかを確認する前に、オブジェクトを CloudFront キャッシュに保持する秒単位のデフォルトの時間。は、オリジンがオブジェクトで Cache-Control または Expires ヘッダーを送信しない場合にのみ、CloudFront この設定の値をオブジェクトの TTL として使用します。詳細については、「[コンテンツがキャッシュに保持される期間 \(有効期限\) の管理](#)」を参照してください。

Note

最小 TTL、最大 TTL、およびデフォルト TTL 設定がすべて 0 に設定されている場合、CloudFront キャッシュは無効になります。

キャッシュキー設定

キャッシュキー設定では、がキャッシュキー CloudFront に含めるビューワーリクエストの値を指定します。値には、URL クエリ文字列、HTTP ヘッダー、および Cookie を含めることができます。キャッシュキーに含める値は、オリジンリクエストと呼ばれる、がオリジン CloudFront に送信するリクエストに自動的に含まれます。キャッシュキーに影響を与えずにオリジンリクエストを管理する方法については、「[オリジンリクエストの制御](#)」を参照してください。

キャッシュキーには次のような設定があります。

- [ヘッダー](#)
- [Cookie](#)
- [クエリ文字列](#)
- [圧縮のサポート](#)

ヘッダー

がキャッシュキーおよびオリジンリクエスト CloudFront に含めるビューワーリクエストの HTTP ヘッダー。ヘッダーには、以下のいずれかの設定を選択できます。

- [None (なし)] - ビューワーリクエストの HTTP ヘッダーはキャッシュキーに含まれず、オリジンリクエストに自動的に含まれません。

- [次のヘッダーを含める]- ビューワーリクエストのどの HTTP ヘッダーをキャッシュキーに含め、オリジンリクエストに自動的に含めるかを指定します。

[次のヘッダーを含める] 設定を使用する場合、HTTP ヘッダーは値ではなく名前で指定します。たとえば、次の HTTP ヘッダーを考えてみます。

```
Accept-Language: en-US,en;q=0.5
```

この場合、ヘッダーを `Accept-Language: en-US,en;q=0.5` としてではなく、`Accept-Language` として指定します。ただし、CloudFront はキャッシュキーとオリジンリクエストに、その値を含む完全なヘッダーを含めます。

によって生成された特定のヘッダー CloudFront をキャッシュキーに含めることもできます。詳細については、「[the section called “ CloudFront リクエストヘッダーの追加”](#)」を参照してください。

Cookie

がキャッシュキーとオリジンリクエスト CloudFront に含めるビューワーリクエストの Cookie。Cookie には、以下のいずれかの設定を選択できます。

- [None (なし)]- ビューワーリクエストの Cookie はキャッシュキーに含まれず、オリジンリクエストに自動的に含まれません。
- [All (すべて)]- ビューワーリクエストのすべての Cookie は、キャッシュキーに含まれ、オリジンリクエストに自動的に含まれます。
- [指定した Cookie を含める]- ビューワーリクエストのどの Cookie をキャッシュキーに含め、オリジンリクエストに自動的に含めるかを指定します。
- [次を除くすべての Cookie を含める]- ビューワーリクエストのどの Cookie をキャッシュキーに含めず、オリジンリクエストに自動的に含めないかを指定します。指定した Cookie 以外の他のすべての Cookie は、キャッシュキーに含まれ、自動的にオリジンリクエストに含まれます。

[指定した Cookie を含める] または [次を除くすべての Cookie を含める] 設定を使用する場合、Cookie は値ではなく名前で指定します。たとえば、次の Cookie ヘッダーを考えてみます。

```
Cookie: session_ID=abcd1234
```

この場合、Cookie を `session_ID=abcd1234` としてではなく、`session_ID` として指定します。ただし、キャッシュキーとオリジンリクエストには、その値を含む完全な Cookie CloudFront が含まれます。

クエリ文字列

キャッシュキーとオリジンリクエストに CloudFront が含まれるビューワーリクエストの URL クエリ文字列。クエリ文字列には、以下のいずれかの設定を選択できます。

- [None (なし)] - ビューワーリクエストのクエリ文字列はキャッシュキーに含まれず、オリジンリクエストに自動的に含まれません。
- [All (すべて)] - ビューワーリクエストのすべてのクエリ文字列は、キャッシュキーに含まれ、オリジンリクエストにも自動的に含まれます。
- [指定したクエリ文字列を含める] - ビューワーリクエストのどのクエリ文字列をキャッシュキーに含め、オリジンリクエストに自動的に含めるかを指定します。
- [次を除くすべてのクエリ文字列を含める] - ビューワーリクエストのどのクエリ文字列をキャッシュキーに含めず、オリジンリクエストに自動的に含めないかを指定します。指定したクエリ文字列以外の他のすべてのクエリ文字列は、キャッシュキーに含まれ、自動的にオリジンリクエストに含まれます。

[指定したクエリ文字列を含める] または [次を除くすべてのクエリ文字列を含める] 設定を使用する場合、クエリ文字列は値ではなく名前で指定します。たとえば、次の URL パスを考えてみます。

```
/content/stories/example-story.html?split-pages=false
```

この場合、クエリ文字列を `split-pages=false` としてではなく、`split-pages` として指定します。ただし、キャッシュキーとオリジンリクエストには、その値を含む完全なクエリ文字列 CloudFront が含まれます。

圧縮のサポート

これらの設定により CloudFront、ビューワーが Gzip または Brotli 圧縮形式をサポートしている場合、は Gzip または Brotli 圧縮形式で圧縮されたオブジェクトをリクエストおよびキャッシュできます。これらの設定により、[CloudFront 圧縮](#)も機能します。ビューワーは Accept-Encoding HTTP ヘッダーを使用して、これらの圧縮形式のサポートの可否を示します。

Note

ウェブブラウザ Chrome および Firefox では、HTTPS を使用してリクエストを送信する場合のみ、Brotli 圧縮がサポートされます。これらのブラウザでは、HTTP リクエストで Brotli がサポートされません。

次のいずれかに該当する場合は、これらの設定を有効にします。

- ビューワーによってサポートされている場合にオリジンが Gzip 圧縮オブジェクトを返す (リクエストには、HTTPヘッダー Accept-Encoding と値 gzip が含まれます)。この場合、Gzip 対応設定を使用します (CloudFront API、 AWS SDKs、 AWS CLI または true では EnableAcceptEncodingGzip に設定します AWS CloudFormation)。
- ビューワーによってサポートされている場合にオリジンが Brotli 圧縮オブジェクトを返す (リクエストには、HTTPヘッダー Accept-Encoding と値 br が含まれます)。この場合、Brotli 対応設定を使用します (CloudFront API、 AWS SDKs、 AWS CLI または true では EnableAcceptEncodingBrotli に設定します AWS CloudFormation)。
- このキャッシュポリシーがアタッチされているキャッシュ動作は、 [CloudFront 圧縮](#) で設定されます。この場合、Gzip または Brotli のいずれか、またはその両方に対してキャッシュを有効にできます。CloudFront 圧縮が有効になっている場合、両方の形式でキャッシュを有効にすると、インターネットへのデータ転送コストを削減できます。

Note

これらの圧縮形式のいずれかまたは両方でキャッシュを有効にする場合は、同じキャッシュ動作に関連付けられた [オリジンリクエストポリシー](#) に Accept-Encoding ヘッダーを含めないでください。CloudFront は、これらの形式のいずれかでキャッシュが有効になっている場合、常にこのヘッダーをオリジンリクエストに含めます。そのため、オリジンリクエストポリシー Accept-Encoding に を含めても効果はありません。

オリジンサーバーが Gzip または Brotli 圧縮オブジェクトを返さない場合、またはキャッシュ動作が CloudFront 圧縮で設定されていない場合は、圧縮オブジェクトのキャッシュを有効にしないでください。有効にすると、 [キャッシュヒット率](#) が低下する可能性があります。

次に、これらの設定が CloudFront デイストリビューションにどのように影響するかを説明します。次のシナリオはすべて、ビューワーリクエストに Accept-Encoding ヘッダーが含まれていることを前提としています。ビューワーリクエストに Accept-Encoding ヘッダーが含まれて

いない場合、このヘッダーをキャッシュキーに含め CloudFront ず、対応するオリジンリクエストに含めません。

圧縮されたオブジェクトのキャッシュが両方の圧縮形式に対応している場合

ビューワーが Gzip と Brotli の両方をサポートしている場合、つまり gzip と の両方の br 値がビューワーリクエストの Accept-Encoding ヘッダーにある場合、CloudFront は次の操作を行います。

- ヘッダーを Accept-Encoding: br, gzip の形式に正規化し、正規化されたヘッダーをキャッシュキーに含めます。キャッシュキーには、ビューワーから送信された Accept-Encoding ヘッダーに存在していた他の値は含まれません。
- エッジロケーションのキャッシュに、リクエストに一致し有効期限が切れていない Brotli または Gzip 圧縮オブジェクトがある場合、エッジロケーションはそのオブジェクトをビューワーに返します。
- エッジロケーションのキャッシュに、リクエストに一致して有効期限が切れていない Brotli または Gzip 圧縮オブジェクトがない場合、 は対応するオリジンリクエストに正規化されたヘッダー (Accept-Encoding: br, gzip) CloudFront を含めます。オリジンリクエストには、ビューワーから送信された Accept-Encoding ヘッダーに存在していた他の値は含まれません。

ビューワーが一方の圧縮形式をサポートしているが、もう一方の圧縮形式をサポートしていない場合、例えば、ビューワーリクエストの Accept-Encoding ヘッダーの gzip 値が であるが br ではない場合、次の CloudFront 操作を行います。

- ヘッダーを Accept-Encoding: gzip の形式に正規化し、正規化されたヘッダーをキャッシュキーに含めます。キャッシュキーには、ビューワーから送信された Accept-Encoding ヘッダーに存在していた他の値は含まれません。
- エッジロケーションのキャッシュに、リクエストに一致し有効期限が切れていない Gzip 圧縮オブジェクトがある場合、エッジロケーションはそのオブジェクトをビューワーに返します。
- エッジロケーションのキャッシュに、リクエストに一致して有効期限が切れていない Gzip 圧縮オブジェクトがない場合、 は対応するオリジンリクエストに正規化されたヘッダー (Accept-Encoding: gzip) CloudFront を含めます。オリジンリクエストには、ビューワーから送信された Accept-Encoding ヘッダーに存在していた他の値は含まれません。

ビューワーが Brotli をサポートしているが Gzip をサポートしていない場合 CloudFront の動作を理解するには、前の例の 2 つの圧縮形式を互いに置き換えます。

ビューワーが Brotli または Gzip をサポートしていない場合、つまりビューワーリクエストの Accept-Encoding ヘッダーに値 gzip として br または が含まれていない場合 CloudFront :

- キャッシュキーに Accept-Encoding ヘッダーを含めません。
- 対応するオリジンリクエストに Accept-Encoding: identity を含めます。オリジンリクエストには、ビューワーから送信された Accept-Encoding ヘッダーに存在していた他の値は含まれません。

圧縮されたオブジェクトのキャッシュが 1 つの圧縮形式に対して有効で、もう 1 つの圧縮形式に対して有効でない場合

ビューワーがキャッシュが有効になっている形式をサポートしている場合。例えば、圧縮オブジェクトのキャッシュが Gzip に対して有効になっており、ビューワーが Gzip をサポートしている場合 (gzip はビューワーリクエストの Accept-Encoding ヘッダーの値の 1 つ) は、次の CloudFront 操作を行います。

- ヘッダーを Accept-Encoding: gzip の形式に正規化し、正規化されたヘッダーをキャッシュキーに含めます。
- エッジロケーションのキャッシュに、リクエストに一致し有効期限が切れていない Gzip 圧縮オブジェクトがある場合、エッジロケーションはそのオブジェクトをビューワーに返します。
- エッジロケーションのキャッシュに、リクエストに一致して有効期限が切れていない Gzip 圧縮オブジェクトがない場合、 は対応するオリジンリクエストに正規化されたヘッダー (Accept-Encoding: gzip) CloudFront を含めます。オリジンリクエストには、ビューワーから送信された Accept-Encoding ヘッダーに存在していた他の値は含まれません。

この動作は、ビューワーが Gzip と Brotli の両方をサポートしている場合と同じです (ビューワーリクエストの Accept-Encoding ヘッダーには gzip と br の両方の値が含まれます)。このシナリオでは、圧縮オブジェクトのキャッシュが Brotli に対応していないためです。

圧縮オブジェクトのキャッシュが Brotli に対して有効で、Gzip に対して有効になっていない場合 CloudFront の動作を理解するには、前の例で 2 つの圧縮形式を相互に置き換えます。

ビューワーがキャッシュが有効になっている圧縮形式をサポートしていない場合 (ビューワーリクエストの Accept-Encoding ヘッダーにその形式の値が含まれていない場合) CloudFront :

- キャッシュキーに Accept-Encoding ヘッダーを含めません。
- 対応するオリジンリクエストに Accept-Encoding: identity を含めます。オリジンリクエストには、ビューワーから送信された Accept-Encoding ヘッダーに存在していた他の値は含まれません。

圧縮されたオブジェクトのキャッシュが両方の圧縮形式に対応していない場合

圧縮されたオブジェクトのキャッシュが両方の圧縮形式で無効になっている場合、はビューワーリクエストの他の HTTP Accept-Encoding ヘッダーと同じようにヘッダーを CloudFront 扱います。デフォルトでは、キャッシュキーには含まれず、オリジンリクエストにも含まれません。他の HTTP ヘッダーと同一のキャッシュポリシーまたはオリジンリクエストポリシーのヘッダーリストに含めることができます。

管理キャッシュポリシーの使用

CloudFront には、ディストリビューションのキャッシュ動作にアタッチできる一連のマネージドキャッシュポリシーが用意されています。管理キャッシュポリシーを使用すると、独自のキャッシュポリシーを記述したり、維持したりする必要はありません。マネージドポリシーでは、特定のユースケースに最適化された設定を使用します。

トピック

- [管理キャッシュポリシーのアタッチ](#)
- [使用可能な管理キャッシュポリシー](#)

管理キャッシュポリシーのアタッチ

管理キャッシュポリシーを使用するには、ディストリビューションのキャッシュ動作にそのポリシーをアタッチします。このプロセスは、キャッシュポリシーを作成するときと同じですが、新しいキャッシュポリシーを作成するのではなく、管理キャッシュポリシーの 1 つをアタッチするだけです。ポリシーをアタッチするには、名前 (コンソールの場合) または ID (AWS CLI または SDK の場合) を使用します。名前と ID は、次のセクションに記載されています。

詳細については、「[キャッシュポリシーの作成](#)」を参照してください。

使用可能な管理キャッシュポリシー

次のトピックでは、使用可能な管理キャッシュポリシーについて説明します。

トピック

- [Amplify](#)
- [CachingDisabled](#)
- [CachingOptimized](#)

- [CachingOptimizedForUncompressedObjects](#)
- [Elemental-MediaPackage](#)

Amplify

[CloudFront コンソールでこのポリシーを表示する](#)

このポリシーは、[AWS Amplify](#) ウェブアプリケーションであるオリジンで使用するよう設計されています。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

2e54312d-136d-493c-8eb9-b001f22f67d2

このポリシーの設定は以下のとおりです。

- 最小 TTL: 2 秒
- 最大 TTL: 600 秒 (10 分)
- デフォルト TTL: 2 秒
- キャッシュキーに含まれるヘッダー:
 - Authorization
 - CloudFront-Viewer-Country
 - Host

圧縮オブジェクトのキャッシュの設定が有効になっているため、正規化された Accept-Encoding ヘッダーも含まれています。詳細については、「[圧縮のサポート](#)」を参照してください。

- キャッシュキーに含まれる Cookie: すべての Cookie が含まれています。
- キャッシュキーに含まれるクエリ文字列: すべてのクエリ文字列が含まれています。
- 圧縮オブジェクトのキャッシュの設定: 有効。詳細については、「[圧縮のサポート](#)」を参照してください。

CachingDisabled

[CloudFront コンソールでこのポリシーを表示する](#)

このポリシーは、キャッシュを無効にします。このポリシーは、動的コンテンツとキャッシュできないリクエストに役立ちます。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

4135ea2d-6df8-44a3-9df3-4b5a84be39ad

このポリシーの設定は以下のとおりです。

- 最小 TTL: 0 秒
- 最大 TTL: 0 秒
- デフォルト TTL: 0 秒
- キャッシュキーに含まれるヘッダー: なし
- キャッシュキーに含まれる Cookie: なし
- キャッシュキーに含まれるクエリ文字列: なし
- 圧縮オブジェクトのキャッシュの設定: 無効

CachingOptimized

[CloudFront コンソールでこのポリシーを表示する](#)

このポリシーは、キャッシュキーに CloudFront が含まれる値を最小化することで、キャッシュ効率を最適化するように設計されています。CloudFront キャッシュキーにはクエリ文字列や Cookie が含まれず、正規化された Accept-Encoding ヘッダーのみが含まれます。これにより、オリジンがオブジェクト CloudFront を返すとき、または [CloudFront エッジ圧縮が有効になっているときに、は Gzip 圧縮形式と Brotli 圧縮形式でオブジェクトを個別にキャッシュできます。](#)

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

658327ea-f89d-4fab-a63d-7e88639e58f6

このポリシーの設定は以下のとおりです。

- 最小 TTL: 1 秒。
- 最大 TTL: 31,536,000 秒 (365 日)。

- デフォルト TTL: 86,400 秒 (24 時間)。
- キャッシュキーに含まれるヘッダー: 明示的に含まれているものはありません。圧縮オブジェクトのキャッシュの設定が有効になっているため、正規化された Accept-Encoding ヘッダーが含まれます。詳細については、「[圧縮のサポート](#)」を参照してください。
- キャッシュキーに含まれる Cookie: なし。
- キャッシュキーに含まれるクエリ文字列: なし。
- 圧縮オブジェクトのキャッシュの設定: 有効。詳細については、「[圧縮のサポート](#)」を参照してください。

CachingOptimizedForUncompressedObjects

[CloudFront コンソールでこのポリシーを表示する](#)

このポリシーは、キャッシュキーに含まれる値を最小限に抑えることで、キャッシュ効率を最適化するよう設計されています。クエリ文字列、ヘッダー、または Cookie は含まれません。このポリシーは、前のポリシーと同じですが、圧縮オブジェクトのキャッシュの設定が無効になります。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

b2884449-e4de-46a7-ac36-70bc7f1ddd6d

このポリシーの設定は以下のとおりです。

- 最小 TTL: 1 秒
- 最大 TTL: 31,536,000 秒 (365 日)
- デフォルト TTL: 86,400 秒 (24 時間)
- キャッシュキーに含まれるヘッダー: なし
- キャッシュキーに含まれる Cookie: なし
- キャッシュキーに含まれるクエリ文字列: なし
- 圧縮オブジェクトのキャッシュの設定: 無効

Elemental-MediaPackage

[CloudFront コンソールでこのポリシーを表示する](#)

このポリシーは、オリジンとしての AWS Elemental MediaPackage 用に設計されています。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

08627262-05a9-4f76-9ded-b50ca2e3a84f

このポリシーの設定は以下のとおりです。

- 最小 TTL: 0 秒
- 最大 TTL: 31,536,000 秒 (365 日)
- デフォルト TTL: 86,400 秒 (24 時間)
- キャッシュキーに含まれるヘッダー:
 - Origin

圧縮オブジェクトのキャッシュの設定が Gzip に対して有効になっているため、正規化された Accept-Encoding ヘッダーも含まれています。詳細については、「[圧縮のサポート](#)」を参照してください。

- キャッシュキーに含まれる Cookie: なし
- キャッシュキーに含まれるクエリ文字列:
 - aws.manifestfilter
 - start
 - end
 - m
- 圧縮オブジェクトのキャッシュの設定: Gzip に対して有効にする。詳細については、「[圧縮のサポート](#)」を参照してください。

キャッシュキーについて

キャッシュキーは、CloudFront エッジロケーションへのビューワーリクエストがキャッシュヒットになるかどうかを決定します。キャッシュキーは、キャッシュ内のオブジェクトの固有の識別子です。キャッシュ内の各オブジェクトには、固有のキャッシュキーがあります。

キャッシュヒットは、ビューワーリクエストが以前のリクエストと同じキャッシュキーを生成し、そのキャッシュキーのオブジェクトがエッジロケーションのキャッシュにあり、有効な場合に発生し

ます。キャッシュヒットが発生すると、リクエストされたオブジェクトは CloudFront エッジロケーションからビューワーに提供されます。これには次の利点があります。

- オリジンサーバーの負荷を軽減
- ビューワーのレイテンシーを低減

キャッシュヒット率が高い (キャッシュヒットにつながるビューワーリクエストの割合が高い) ほど、ウェブサイトやアプリケーションのパフォーマンスが高くなります。キャッシュヒット率を改善する 1 つの方法は、キャッシュキーに必要な最小値のみを含めることです。詳細については、次のセクションを参照してください。

[キャッシュポリシー](#)を使用して、キャッシュキーの値 (URL クエリ文字列、HTTP ヘッダー、および Cookie) を変更できます。 ([Lambda@Edge 関数](#)を使用してキャッシュキーを変更することもできます)。キャッシュキーを変更する前に、アプリケーションの設計方法と、ビューワーリクエストの特性に基づいてさまざまなレスポンスをいつ、どのように処理するかを理解することが重要です。ビューワーリクエストの値によってオリジンが返すレスポンスが決定された場合は、その値をキャッシュキーに含める必要があります。しかし、オリジンが返すレスポンスに影響を与えない値をキャッシュキーに含めると、重複したオブジェクトをキャッシュしてしまう可能性があります。

デフォルトのキャッシュキー

デフォルトでは、CloudFront デイストリビューションのキャッシュキーには次の情報が含まれます。

- CloudFront デイストリビューションのドメイン名 (d1111111abcdef8.cloudfront.net など)
- リクエストされたオブジェクトの URL パス (例: /content/stories/example-story.html)

Note

この OPTIONS メソッドは、OPTIONS リクエストのキャッシュキーに含まれます。つまり、OPTIONS リクエストへのレスポンスは、GET リクエストおよび HEAD リクエストへのレスポンスとは別にキャッシュされます。

デフォルトでは、ビューワーリクエストのその他の値はキャッシュキーに含まれません。ウェブブラウザからの次の HTTP リクエストを考えてみましょう。

```
GET /content/stories/example-story.html?ref=0123abc&split-pages=false
HTTP/1.1
Host: d111111abcdef8.cloudfront.net
User-Agent: Mozilla/5.0 Gecko/20100101 Firefox/68.0
Accept: text/html,*/*
Accept-Language: en-US,en
Cookie: session_id=01234abcd
Referer: https://news.example.com/
```

この例のようなビューワーリクエストが CloudFront エッジロケーションに到達すると、は キャッシュキー CloudFront を使用してキャッシュヒットがあるかどうかを判断します。デフォルトでは、リクエストの `/content/stories/example-story.html` および `d111111abcdef8.cloudfront.net` コンポーネントのみがキャッシュキーに含まれます。リクエストされたオブジェクトがキャッシュにない場合 (キャッシュミス)、はオブジェクトを取得するリクエストをオリジン CloudFront に送信します。オブジェクトを取得した後、はオブジェクトをビューワーに CloudFront 返し、エッジロケーションのキャッシュに保存します。

がキャッシュキーによって決定される同じオブジェクトに対する別のリクエスト CloudFront を受信すると、は、オリジンにリクエストを送信せずに、キャッシュされたオブジェクトをビューワーに即座に CloudFront 保存します。たとえば、前のリクエストの後に入ってくる次の HTTP リクエストを考えてみましょう。

```
GET /content/stories/example-story.html?ref=xyz987&split-pages=true
HTTP/1.1
Host: d111111abcdef8.cloudfront.net
User-Agent: Mozilla/5.0 AppleWebKit/537.36 Chrome/83.0.4103.116
Accept: text/html,*/*
Accept-Language: en-US,en
Cookie: session_id=wxyz9876
Referer: https://rss.news.example.net/
```

このリクエストは、前のリクエストと同じオブジェクトに対するものですが、前のリクエストとは異なります。これは、異なる URL クエリ文字列、異なる Referer ヘッダーと User-Agent ヘッダー、および異なる session_id Cookie を持っています。ただし、これらの値はデフォルトではキャッシュキーの一部ではないので、この 2 番目のリクエストはキャッシュヒットになります。

キャッシュキーのカスタマイズ

場合によっては、キャッシュキーにさらに多くの情報を含める必要がありますが、そうするとキャッシュのヒット数が少なくなることがあります。[キャッシュポリシー](#)を使用して、キャッシュキーに含める内容を指定します。

たとえば、オリジンサーバーがビューワーリクエストで Accept-Language HTTP ヘッダーを使用して、ビューワーの言語に基づいて異なるコンテンツを返す場合、このヘッダーをキャッシュキーに含めることができます。これを行うと、はこのヘッダー CloudFront を使用してキャッシュヒットを判断し、オリジンリクエスト (キャッシュミスが発生したときに がオリジン CloudFront に送信するリクエスト) に ヘッダーを含めます。

キャッシュキー CloudFrontに追加の値を含めると、ビューワーリクエストで発生する可能性のある変動により、重複オブジェクトがキャッシュされる可能性があります。たとえば、ビューワーは、Accept-Language ヘッダーに対して次の値のいずれかを送信できます。

- en-US, en
- en, en-US
- en-US, en
- en-US

これらの異なる値はすべて、ビューワーの言語が英語であることを示していますが、バリエーションによって が同じオブジェクト CloudFront を複数回キャッシュする可能性があります。これにより、キャッシュヒットを減らし、オリジンリクエストの数を増やすことができます。キャッシュキーに Accept-Language ヘッダーを含めず、異なる言語のコンテンツに異なる URL を使用するようウェブサイトまたはアプリケーションを設定することで、この重複を避けることができます (例: /en-US/content/stories/example-story.html)。

キャッシュキーに含める任意の値については、その値のバリエーションがビューワーリクエストに表示される可能性があることを理解しておく必要があります。特定のリクエスト値については、キャッシュキーに含めることはほとんど意味がありません。たとえば、User-Agent ヘッダーには何千もの固有のバリエーションがあるため、通常はキャッシュキーに含めるには適していません。ユーザー固有の値またはセッション固有の値を持ち、何千もの (または数百万) のリクエストで固有の Cookie も、キャッシュキーを含む候補には適していません。キャッシュキーにこれらの値を含めると、それぞれ固有のバリエーションによって、キャッシュ内のオブジェクトの別のコピーが作成されます。オブジェクトのこれらのコピーが固有でない場合、またはわずかに異なるオブジェクトが多数作成され、各オブジェクトが少数のキャッシュヒットのみを取得する場合は、別のアプローチを検討するこ

とをお勧めします。これらの非常に可変性の高い値をキャッシュキーから除外することも、オブジェクトをキャッシュ不可能としてマークすることもできます。

キャッシュキーをカスタマイズするときは注意が必要です。時には望ましいことがあります、重複オブジェクトのキャッシュ、キャッシュヒット率の低下、オリジンリクエスト数の増加など、意図しない結果が生じる可能性があります。オリジンのウェブサイトまたはアプリケーションが、分析、テレメトリー、またはその他の用途に対するビューワーリクエストから特定の値を受け取る必要があるが、これらの値によってオリジンが返すオブジェクトが変更されない場合は、[オリジンリクエストポリシー](#)を使用してこれらの値をオリジンリクエストに含めますが、キャッシュキーには含めません。

オリジンリクエストの制御

へのビューワーリクエスト CloudFront によってキャッシュミスが発生すると (リクエストされたオブジェクトはエッジロケーションでキャッシュされません)、 はオブジェクトを取得するリクエストをオリジン CloudFront に送信します。これは、オリジンリクエストと呼ばれます。オリジンリクエストには、ビューワーリクエストの次の情報が常に含まれます。

- URL パス (URL クエリ文字列またはドメイン名を含まないパスのみ)
- リクエストボディ (存在する場合)
- 、 、 など User-Agent、すべてのオリジンリクエストに自動的に CloudFront 含まれる HTTP Host ヘッダー X-Amz-Cf-Id

ビューワーリクエストのその他の情報 (URL クエリ文字列、HTTP ヘッダー、Cookie など) は、デフォルトではオリジンリクエストに含まれません。(例外: レガシーキャッシュ設定では、 はデフォルトでヘッダーをオリジン CloudFront に転送します)。ただし、分析やテレメトリーのためにデータを収集するなど、オリジンでこのその他の情報を受信することができます。オリジンリクエストポリシーを使用して、オリジンリクエストに含まれる情報を制御できます。

オリジンリクエストポリシーは、キャッシュキーを制御する [キャッシュポリシー](#) とは別のものです。この分離により、オリジンで追加情報を受信し、適切なキャッシュヒット率 (キャッシュヒットとなるビューワーリクエストの割合) を維持することができます。これを行うには、オリジンリクエストに含める情報 (オリジンリクエストポリシーを使用) とキャッシュキーに含める情報 (キャッシュポリシーを使用) を個別に制御します。

2 種類のポリシーは別々のものですが、関連性があります。キャッシュキーに含めるすべての URL クエリ文字列、HTTP ヘッダー、および Cookie (キャッシュポリシーを使用) は、オリジンリクエストに自動的に含まれます。オリジンリクエストポリシーを使用して、オリジンリクエストに含める

が、キャッシュキーには含めない情報を指定します。キャッシュポリシーと同様に、オリジンリクエストポリシーを CloudFront デイストリビューションの 1 つ以上のキャッシュ動作にアタッチします。

オリジンリクエストポリシーを使用して、ビューワーリクエストに含まれていないオリジンリクエストに追加の HTTP ヘッダーを追加することもできます。これらの追加のヘッダーは、オリジンリクエストを送信する CloudFront 前に よって追加され、ビューワーリクエストに基づいて自動的に決定されるヘッダー値が含まれます。詳細については、「[the section called “ CloudFront リクエストヘッダーの追加”](#)」を参照してください。

トピック

- [オリジンリクエストポリシーの作成](#)
- [オリジンリクエストポリシーについて](#)
- [管理オリジンリクエストポリシーの使用](#)
- [CloudFront リクエストヘッダーの追加](#)
- [オリジンリクエストポリシーとキャッシュポリシーの連携方法を理解する](#)

オリジンリクエストポリシーの作成

オリジンリクエストポリシーを使用して、 がオリジン CloudFront に送信するリクエストに含まれる値 (URL クエリ文字列、HTTP ヘッダー、Cookie) を制御できます。オリジンリクエストポリシーは、CloudFront コンソール、AWS Command Line Interface (AWS CLI)、または CloudFront API を使用して作成できます。

オリジンリクエストポリシーを作成したら、デイス CloudFront トリビューションの 1 つ以上のキャッシュ動作にアタッチします。

オリジンリクエストポリシーは必須ではありません。キャッシュ動作にオリジンリクエストポリシーがアタッチされていない場合、オリジンリクエストには、[キャッシュポリシー](#)で指定されたすべての値が含まれますが、それ以上は含まれません。

Note

オリジンリクエストポリシーを使用するには、キャッシュ動作でも[キャッシュポリシー](#)を使用する必要があります。キャッシュポリシーがないと、キャッシュ動作でオリジンリクエストポリシーを使用することはできません。

Console

オリジンリクエストポリシーを作成するには (コンソール)

1. にサインインAWS Management Consoleし、 CloudFront コンソールのポリシーページを開きます<https://console.aws.amazon.com/cloudfront/v4/home?#/policies>。
2. [オリジンリクエスト] を選択してから、[オリジンリクエストポリシーの作成] を選択します。
3. このオリジンリクエストポリシーに目的の設定を選択します。詳細については、「[オリジンリクエストポリシーについて](#)」を参照してください。
4. 終了したら、[作成] を選択します。

オリジンリクエストポリシーを作成したら、それをキャッシュ動作にアタッチできます。

オリジンリクエストポリシーを既存のディストリビューションにアタッチするには (コンソール)

1. の CloudFront コンソールでディストリビューションページを開きます<https://console.aws.amazon.com/cloudfront/v4/home#/distributions>。
2. 更新するディストリビューションを選択し、[動作] タブを選択します。
3. 更新するキャッシュ動作を選択し、[編集] を選択します。
または、新しいキャッシュ動作を作成するには、[動作を作成] を選択します。
4. [キャッシュキーとオリジンリクエスト] セクションで、[キャッシュポリシーとオリジンリクエストポリシー] が選択されていることを確認します。
5. [オリジンリクエストポリシー] では、このキャッシュ動作にアタッチするオリジンリクエストポリシーを選択します。
6. ページの最下部で [変更の保存] を選択します。

オリジンリクエストポリシーを新しいディストリビューションにアタッチするには (コンソール)

1. で CloudFront コンソールを開きます<https://console.aws.amazon.com/cloudfront/v4/home>。
2. [ディストリビューションの作成] を選択します。
3. [キャッシュキーとオリジンリクエスト] セクションで、[キャッシュポリシーとオリジンリクエストポリシー] が選択されていることを確認します。
4. [Origin request policy] (オリジンリクエストポリシー) で、このディストリビューションのデフォルトのキャッシュ動作にアタッチするオリジンリクエストポリシーを選択します。

5. オリジン、デフォルトのキャッシュ動作、その他のディストリビューション設定に必要な設定を選択します。詳細については、「[ディストリビューションを作成または更新する場合に指定する値](#)」を参照してください。
6. 終了したら、[ディストリビューションの作成] を選択します。

CLI

AWS Command Line Interface (AWS CLI) でオリジンリクエストポリシーを作成するには、`aws cloudfront create-origin-request-policy` コマンドを使用します。コマンドの入力パラメータは、コマンドライン入力として個別に指定せずに、入力ファイルを使用して指定できます。

オリジンリクエストポリシーを作成するには (入力ファイルを含む CLI)

1. 次のコマンドを使用して、`origin-request-policy.yaml` コマンドのすべての入力パラメータを含む `create-origin-request-policy` という名前のファイルを作成します。

```
aws cloudfront create-origin-request-policy --generate-cli-skeleton yaml-input > origin-request-policy.yaml
```

2. 先ほど作成した `origin-request-policy.yaml` という名前のファイルを開きます。ファイルを編集して、必要なオリジンリクエストポリシー設定を指定し、ファイルを保存します。ファイルからオプションのフィールドを削除することはできますが、必須フィールドは削除しないでください。

オリジンリクエストポリシー設定の詳細については、「[オリジンリクエストポリシーについて](#)」を参照してください。

3. 次のコマンドを使用して、`origin-request-policy.yaml` ファイルの入力パラメータを使用し、オリジンリクエストポリシーを作成します。

```
aws cloudfront create-origin-request-policy --cli-input-yaml file://origin-request-policy.yaml
```

コマンドの出力の `Id` 値を書き留めます。これはオリジンリクエストポリシー ID であり、オリジンリクエストポリシーを CloudFront ディストリビューションのキャッシュ動作にアタッチするために必要です。

オリジンリクエストポリシーを既存のディストリビューションにアタッチするには (入力ファイルを含む CLI)

1. 次のコマンドを使用して、更新するディストリビューションの CloudFront ディストリビューション設定を保存します。 *distribution_ID* をディストリビューションの ID に置き換えます。

```
aws cloudfront get-distribution-config --id distribution_ID --output yaml >
dist-config.yaml
```

2. 先ほど作成した `dist-config.yaml` という名前のファイルを開きます。ファイルを編集し、オリジンリクエストポリシーを使用するように更新する各キャッシュ動作に次の変更を加えます。
 - キャッシュ動作で、`OriginRequestPolicyId` という名前のフィールドを追加します。フィールドの値には、ポリシーの作成後に書き留めたオリジンリクエストポリシー ID を使用します。
 - `ETag` フィールドの名前を `IfMatch` に変更します。ただし、フィールドの値は変更しないでください。

完了したら、ファイルを保存します。

3. オリジンリクエストポリシーを使用するようにディストリビューションを更新するには、次のコマンドを使用します。 *distribution_ID* をディストリビューションの ID に置き換えます。

```
aws cloudfront update-distribution --id distribution_ID --cli-input-yaml file://
dist-config.yaml
```

オリジンリクエストポリシーを新しいディストリビューションにアタッチするには (入力ファイルを含む CLI)

1. 次のコマンドを使用して、`distribution.yaml` コマンドのすべての入力パラメータを含む `create-distribution` という名前のファイルを作成します。

```
aws cloudfront create-distribution --generate-cli-skeleton yml-input >
distribution.yml
```

2. 先ほど作成した `distribution.yml` という名前のファイルを開きます。デフォルトのキャッシュ動作の `[OriginRequestPolicyId]` フィールドに、ポリシーの作成後に書き留めたオリジンリクエストポリシー ID を入力します。ファイルの編集を続行して必要なディストリビューション設定を指定し、完了したらファイルを保存します。

ディストリビューション設定の詳細については、「[ディストリビューションを作成または更新する場合に指定する値](#)」を参照してください。

3. 次のコマンドを使用して、`distribution.yml` ファイルの入力パラメータを使用し、ディストリビューションを作成します。

```
aws cloudfront create-distribution --cli-input-yml file://distribution.yml
```

API

CloudFront API を使用してオリジンリクエストポリシーを作成するには、[CreateOriginRequestPolicy](#) を使用します。この API コールで指定するフィールドの詳細については、「[オリジンリクエストポリシーについて](#)」、および AWS SDK やその他の API クライアントの API リファレンスドキュメントを参照してください。

オリジンリクエストポリシーを作成したら、次の API コールのいずれかを使用して、それをキャッシュ動作にアタッチできます。

- 既存のディストリビューションのキャッシュ動作にアタッチするには、[UpdateDistribution](#) を使用します。
- 新しいディストリビューションのキャッシュ動作にアタッチするには、[CreateDistribution](#) を使用します。

これらの API コールの両方について、キャッシュ動作内で、`OriginRequestPolicyId` フィールドにオリジンリクエストポリシーの ID を指定します。これらの API コールで指定するその他のフィールドの詳細については、「[ディストリビューションを作成または更新する場合に指定する値](#)」と、AWS SDK またはその他 API クライアントの API リファレンスドキュメントを参照してください。

オリジンリクエストポリシーについて

CloudFront では、一般的なユースケース向けに、管理ポリシーと呼ばれる事前定義されたオリジンリクエストポリシーを提供しています。これらのマネージドポリシーを使用することも、ユーザーのニーズ別に独自のオリジンリクエストポリシーを作成することもできます。マネージドポリシーの詳細については、「[管理オリジンリクエストポリシーの使用](#)」を参照してください。

オリジンリクエストポリシーには以下の設定が含まれます。設定は、ポリシー情報とオリジンリクエスト設定に分類されます。

ポリシー情報

名前

オリジンリクエストポリシーを識別する名前。コンソールでは、名前を使用して、オリジンリクエストポリシーをキャッシュ動作にアタッチします。

説明

オリジンリクエストポリシーを説明するコメント。これはオプションです。

オリジンリクエスト設定

オリジンリクエスト設定では、`Origin` CloudFront に送信するリクエスト (オリジンリクエストと呼ばれる) に含まれるビューワーリクエストの値を指定します。値には、URL クエリ文字列、HTTP ヘッダー、および Cookie を含めることができます。指定した値は、オリジンリクエストに含まれますが、キャッシュキーには含まれません。キャッシュキーの制御については、「[キャッシュキーの管理](#)」を参照してください。

ヘッダー

オリジンリクエストに含まれるビューワーリクエスト CloudFront の HTTP ヘッダー。ヘッダーには、以下のいずれかの設定を選択できます。

- [Note (なし)] – ビューワーリクエストの HTTP ヘッダーは、オリジンリクエストに含まれません。
- [All viewer headers (すべてのビューワーヘッダー)] – ビューワーリクエストのすべての HTTP ヘッダーは、オリジンリクエストに含まれます。
- すべてのビューワーヘッダーと以下の CloudFront ヘッダー – ビューワーリクエストのすべての HTTP ヘッダーは、オリジンリクエストに含まれます。さらに、オリジンリクエストに追加す

る CloudFront ヘッダーを指定します。CloudFront ヘッダーの詳細については、「」を参照してください [the section called “ CloudFront リクエストヘッダーの追加”](#)。

- [次のヘッダーを含める] – オリジンリクエストに含める HTTP ヘッダーを指定します。

Note

[オリジンのカスタムヘッダー] 設定に既に含まれているヘッダーを指定しないでください。詳細については、「[オリジンリクエスト CloudFront にカスタムヘッダーを追加するための の設定](#)」を参照してください。

- [以下を除くすべてのビューワーヘッダー] – オリジンリクエストに含まれない HTTP ヘッダーを指定します。指定されたものを除いて、ビューワーリクエストの他のすべての HTTP ヘッダーが含まれます。

すべてのビューワーヘッダーと次の CloudFront ヘッダー、次のヘッダーを含める、または設定を除くすべてのビューワーヘッダーを使用する場合、ヘッダー名のみで HTTP ヘッダーを指定します。オリジンリクエストには、その値を含む完全なヘッダー CloudFront が含まれます。

Note

設定以外のすべてのビューワーヘッダーを使用してビューワーの Host ヘッダーを削除すると、はオリジンのドメイン名を含む新しい Host ヘッダーをオリジンリクエスト CloudFront に追加します。

Cookie

オリジンリクエストに含まれるビューワーリクエスト CloudFront の Cookie。Cookie には、以下のいずれかの設定を選択できます。

- [None (なし)] – ビューワーリクエストの Cookie は、オリジンリクエストに含まれません。
- [All (すべて)] – ビューワーリクエストのすべての Cookie は、オリジンリクエストに含まれます。
- [次のヘッダーを含める] – ビューワーリクエストのどの Cookie をオリジンリクエストに含めるかを指定します。
- [次を除くすべての Cookie] – ビューワーリクエストのどの Cookie をオリジンリクエストに含めないかを指定します。ビューワーリクエストの他のすべての Cookie が含まれます。

次の Cookie を含める または 設定を除くすべての Cookie を使用する場合は、名前のみを使用して Cookie を指定します。オリジンリクエストには、その値を含む完全な Cookie CloudFront が含まれます。

クエリ文字列

オリジンリクエストに含まれるビューワーリクエスト CloudFront の URL クエリ文字列。クエリ文字列には、以下のいずれかの設定を選択できます。

- [None (なし)] – ビューワーリクエストのクエリ文字列は、オリジンリクエストに含まれません。
- [All (すべて)] – ビューワーリクエストのすべてのクエリ文字列は、オリジンリクエストに含まれます。
- [次のクエリ文字列を含める] – ビューワーリクエストのどのクエリ文字列をオリジンリクエストに含めるかを指定します。
- [次を除くすべてのクエリ文字列] – ビューワーリクエストのどのクエリ文字列をオリジンリクエストに含めないかを指定します。その他すべてのクエリ文字列が含まれます。

次のクエリ文字列を含めるまたは設定以外のすべてのクエリ文字列を使用する場合は、名前のみを使用してクエリ文字列を指定します。オリジンリクエストには、その値を含む完全なクエリ文字列 CloudFront が含まれます。

管理オリジンリクエストポリシーの使用

CloudFront には、ディストリビューションのキャッシュ動作にアタッチできる一連のマネージドオリジンリクエストポリシーが用意されています。管理オリジンリクエストポリシーを使用すると、独自のオリジンリクエストポリシーを記述したり、維持したりする必要はありません。マネージドポリシーでは、特定のユースケースに最適化された設定を使用します。

トピック

- [管理オリジンリクエストポリシーのアタッチ](#)
- [使用可能な管理オリジンリクエストポリシー](#)

管理オリジンリクエストポリシーのアタッチ

管理オリジンリクエストポリシーを使用するには、ディストリビューションのキャッシュ動作にそのポリシーをアタッチします。このプロセスは、オリジンリクエストポリシーを作成するときと同じで

すが、新しいオリジンリクエストポリシーを作成するのではなく、管理オリジンリクエストポリシーの1つをアタッチするだけです。ポリシーをアタッチするには、名前 (コンソールの場合) または ID (AWS CLI または SDK の場合) を使用します。名前と ID は、次のセクションに記載されています。

詳細については、「[オリジンリクエストポリシーの作成](#)」を参照してください。

使用可能な管理オリジンリクエストポリシー

次のトピックでは、使用可能な管理オリジンリクエストポリシーについて説明します。

トピック

- [AllViewer](#)
- [AllViewerAndCloudFrontHeaders-2022年6月](#)
- [AllViewerExceptHostHeader](#)
- [CORS-CustomOrigin](#)
- [CORS-S3Origin](#)
- [Elemental-MediaTailor-PersonalizedManifests](#)
- [UserAgentRefererHeaders](#)

AllViewer

[CloudFront コンソールでこのポリシーを表示する](#)

このポリシーには、ビューワーリクエストのすべての値 (ヘッダー、Cookie、クエリ文字列) が含まれます。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

```
216adef6-5c7f-47e4-b989-5492eafa07d3
```

このポリシーの設定は以下のとおりです。

- オリジンリクエストに含まれるヘッダー: ビューワーリクエスト内のすべてのヘッダー
- オリジンリクエストに含まれる Cookie: すべて
- オリジンリクエストに含まれるクエリ文字列: すべて

AllViewerAndCloudFrontHeaders-2022 年 6 月

[CloudFront コンソールでこのポリシーを表示する](#)

このポリシーには、ビューワーリクエストのすべての値 (ヘッダー、Cookie、クエリ文字列) と、2022 年 6 月までにリリースされたすべての [CloudFront ヘッダー](#) (2022 年 6 月以降にリリースされた CloudFront ヘッダーは含まれません) が含まれます。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

33f36d7e-f396-46d9-90e0-52428a34d9dc

このポリシーの設定は以下のとおりです。

- オリジンリクエストに含まれるヘッダー：ビューワーリクエスト内のすべてのヘッダー、および次の CloudFront ヘッダー。
 - CloudFront-Forwarded-Proto
 - CloudFront-Is-Android-Viewer
 - CloudFront-Is-Desktop-Viewer
 - CloudFront-Is-IOS-Viewer
 - CloudFront-Is-Mobile-Viewer
 - CloudFront-Is-SmartTV-Viewer
 - CloudFront-Is-Tablet-Viewer
 - CloudFront-Viewer-Address
 - CloudFront-Viewer-ASN
 - CloudFront-Viewer-City
 - CloudFront-Viewer-Country
 - CloudFront-Viewer-Country-Name
 - CloudFront-Viewer-Country-Region
 - CloudFront-Viewer-Country-Region-Name
 - CloudFront-Viewer-Http-Version
 - CloudFront-Viewer-Latitude
 - CloudFront-Viewer-Longitude
 - CloudFront-Viewer-Metro-Code

- CloudFront-Viewer-Postal-Code
- CloudFront-Viewer-Time-Zone
- CloudFront-Viewer-TLS
- オリジンリクエストに含まれる Cookie: すべて
- オリジンリクエストに含まれるクエリ文字列: すべて

AllViewerExceptHostHeader

[CloudFront コンソールでこのポリシーを表示する](#)

このポリシーには、ビューワーリクエストの Host ヘッダーは含まれませんが、ビューワーリクエストの他のすべての値 (ヘッダー、Cookie、クエリ文字列) が含まれます。

このポリシーには、HTTP プロトコル、HTTP バージョン、TLS バージョン、およびすべてのデバイスタイプとビューワーロケーションヘッダー用の追加の [CloudFront リクエスト](#) ヘッダーも含まれています。

このポリシーは、Amazon API Gateway と AWS Lambda 関数 URL オリジンでの使用を目的としています。これらのオリジンは、Hostヘッダーにデイス CloudFront トリビューションのドメイン名ではなく、オリジンドメイン名が含まれていることを想定しています。ビューワーリクエストの Host ヘッダーをこれらのオリジンに転送すると、それらが機能しなくなる可能性があります。

Note

このマネージドオリジンリクエストポリシーを使用してビューワーの Hostヘッダーを削除すると、はオリジンのドメイン名を持つ新しい Hostヘッダーをオリジンリクエスト CloudFront に追加します。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

b689b0a8-53d0-40ab-baf2-68738e2966ac

このポリシーの設定は以下のとおりです。

- オリジンリクエストに含まれるヘッダー: Host ヘッダーを除くビューワーリクエスト内のすべてのヘッダー

- オリジンリクエストに含まれる Cookie: すべて
- オリジンリクエストに含まれるクエリ文字列: すべて

CORS-CustomOrigin

[CloudFront コンソールでこのポリシーを表示する](#)

このポリシーには、オリジンがカスタムオリジンである場合に、Cross-Origin Resource Sharing (CORS) リクエストを有効にするヘッダーが含まれます。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

59781a5b-3903-41f3-afcb-af62929ccde1

このポリシーの設定は以下のとおりです。

- オリジンリクエストに含まれるヘッダー:
 - Origin
- オリジンリクエストに含まれる Cookie: なし
- オリジンリクエストに含まれるクエリ文字列: なし

CORS-S3Origin

[CloudFront コンソールでこのポリシーを表示する](#)

このポリシーには、オリジンが Amazon S3 バケットである場合に、Cross-Origin Resource Sharing (CORS) リクエストを有効にするヘッダーが含まれます。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

88a5eaf4-2fd4-4709-b370-b4c650ea3fcf

このポリシーの設定は以下のとおりです。

- オリジンリクエストに含まれるヘッダー:
 - Origin

- Access-Control-Request-Headers
- Access-Control-Request-Method
- オリジンリクエストに含まれる Cookie: なし
- オリジンリクエストに含まれるクエリ文字列: なし

Elemental-MediaTailor-PersonalizedManifests

[CloudFront コンソールでこのポリシーを表示する](#)

このポリシーは、オリジンとしての AWS Elemental MediaTailor エンドポイント用に意図されています。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

775133bc-15f2-49f9-abea-afb2e0bf67d2

このポリシーの設定は以下のとおりです。

- オリジンリクエストに含まれるヘッダー:
 - Origin
 - Access-Control-Request-Headers
 - Access-Control-Request-Method
 - User-Agent
 - X-Forwarded-For
- オリジンリクエストに含まれる Cookie: なし
- オリジンリクエストに含まれるクエリ文字列: すべて

UserAgentRefererHeaders

[CloudFront コンソールでこのポリシーを表示する](#)

このポリシーには、User-Agent ヘッダーと Referer ヘッダーのみが含まれます。クエリ文字列や Cookie は含まれません。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

acba4595-bd28-49b8-b9fe-13317c0390fa

このポリシーの設定は以下のとおりです。

- オリジンリクエストに含まれるヘッダー:
 - User-Agent
 - Referer
- オリジンリクエストに含まれる Cookie: なし
- オリジンリクエストに含まれるクエリ文字列: なし

CloudFront リクエストヘッダーの追加

ビューワーから CloudFront を受信し、オリジンまたは [エッジ関数](#) に転送するリクエストに特定の HTTP ヘッダーを追加する CloudFront ように `Cache-Control` を設定できます。これらの HTTP ヘッダーの値は、ビューワーまたはビューワーリクエストの特性に基づいています。ヘッダーは、ビューワーのデバイスタイプ、IP アドレス、地理的位置、リクエストプロトコル (HTTP または HTTPS)、HTTP バージョン、TLS 接続の詳細、および [JA3 フィンガープリント](#) に関する情報を提供します。

これらのヘッダーを使用すると、オリジンまたはエッジ関数でビューワーに関する情報を受け取ることができ、この情報を特定するための独自のコードを記述する必要はありません。オリジンがこれらのヘッダーの情報に基づいて異なるレスポンスを返す場合、`Cache-Control` を個別に CloudFront キャッシュするように、それらをキャッシュキーに含めることができます。例えば、オリジンは、ビューワーが居住する国に基づく特定の言語のコンテンツや、特定のデバイスタイプに合わせたコンテンツで応答する場合があります。オリジンは、これらのヘッダーをログファイルに書き込むこともあります。ログファイルを使用して、ビューワーの場所、ビューワーのデバイスの種類などの情報を判断できます。

キャッシュキーにこれらのヘッダーを含めるには、キャッシュポリシーを使用します。詳細については、「[the section called “キャッシュキーの管理”](#)」および「[the section called “キャッシュキーについて”](#)」を参照してください。

オリジンでヘッダーを受信するが、キャッシュキーに含めないようにするには、オリジンリクエストポリシーを使用します。詳しくは、「[the section called “オリジンリクエストの制御”](#)」を参照してください。

トピック

- [ビューワーのデバイスタイプを特定するためのヘッダー](#)
- [ビューワーの場所を特定するためのヘッダー](#)

- [ビューワーのヘッダー構造を特定するためのヘッダー](#)
- [その他の CloudFront ヘッダー](#)

ビューワーのデバイスタイプを特定するためのヘッダー

ビューワーのデバイスタイプを特定するために、次のヘッダーが追加できます。User-Agent ヘッダーの値に基づいて、これらのヘッダーの値を true または CloudFront に設定します。false。デバイスが複数のカテゴリに属している場合、複数の値が true になる場合があります。例えば、一部のタブレットデバイスでは、CloudFront は CloudFront-Is-Mobile-Viewer との両方 CloudFront-Is-Tablet-Viewer を に設定します true。

- CloudFront-Is-Android-Viewer - ビューワーが Android オペレーティングシステムを搭載したデバイスである true CloudFront と判断された場合、 を に設定します。
- CloudFront-Is-Desktop-Viewer - ビューワーがデスクトップデバイスである true CloudFront と判断された場合、 に設定します。
- CloudFront-Is-IOS-Viewer - ビューワーが iPhone、iPod タッチ、一部の iPad デバイスなどの Apple モバイルオペレーティングシステムを搭載したデバイスであると が CloudFront 判断 true した場合、 に設定します。
- CloudFront-Is-Mobile-Viewer - ビューワーがモバイルデバイスである true CloudFront と判断された場合、 に設定します。
- CloudFront-Is-SmartTV-Viewer - ビューワーがスマートテレビである true CloudFront と判断された場合、 に設定します。
- CloudFront-Is-Tablet-Viewer - ビューワーがタブレットである CloudFront と判断した true ときに に設定します。

ビューワーの場所を特定するためのヘッダー

次のヘッダーを追加して、ビューワーの location. CloudFront determines をビューワーの IP アドレスに基づいて決定できます。これらのヘッダーの値に含まれる ASCII 以外の文字の場合、は [RFC 3986 のセクション 1.2](#) に従って文字を CloudFront パーセントエンコードします。

- CloudFront-Viewer-Address - ビューワーの IP アドレスと、リクエストのソースポートを示します。例えば、198.51.100.10:46532 のヘッダー値は、ビューワーの IP アドレスが 198.51.100.10 で、リクエストのソースポートが 46532 であることを意味します。
- CloudFront-Viewer-ASN - ビューワーの AS 番号 (ASN) を示します。

Note

CloudFront-Viewer-Address と CloudFront-Viewer-ASN は、オリジンリクエストポリシーで追加できますが、キャッシュポリシーでは追加できません。

- CloudFront-Viewer-Country - ビューワーの国の 2 文字の国コードが含まれています。国コードの一覧については、「[ISO 3166-1 alpha-2](#)」を参照してください。

次のヘッダーを追加すると、CloudFront は、AWS ネットワークから発信されるリクエストを除くすべてのリクエストに適用します。

- CloudFront-Viewer-City - ビューワーの市区町村名が含まれています。
- CloudFront-Viewer-Country-Name - ビューワーの国名が含まれています。
- CloudFront-Viewer-Country-Region - ビューワーのリージョンを表すコード (最大 3 文字) が含まれています。地域は、[ISO 3166-2](#)コードの第 1 レベルのサブディビジョン (最も広いまたは狭い) です。
- CloudFront-Viewer-Country-Region-Name - ビューワーのリージョン名が含まれています。地域は、[ISO 3166-2](#)コードの第 1 レベルのサブディビジョン (最も広いまたは狭い) です。
- CloudFront-Viewer-Latitude - ビューワーのおよその緯度が含まれています。
- CloudFront-Viewer-Longitude - ビューワーのおよその経度が含まれています。
- CloudFront-Viewer-Metro-Code - ビューワーのメトロコードが含まれています。これは、ビューワーが米国にいる場合にのみ表示されます。
- CloudFront-Viewer-Postal-Code - ビューワーの郵便番号が含まれています。
- CloudFront-Viewer-Time-Zone [IANA タイムゾーンデータベース形式 \(例: America/Los_Angeles\)](#) で、ビューワーのタイムゾーンが含まれています。

ビューワーのヘッダー構造を特定するためのヘッダー

送信するヘッダーに基づいてビューワーを特定しやすくするために、以下のヘッダーを追加できます。例えば、異なるブラウザごとに HTTP ヘッダーを送信する順序は異なる場合があります。User-Agent ヘッダーに指定されているブラウザが、そのブラウザの想定されたヘッダーの順序と一致しない場合は、リクエストを拒否できます。また、CloudFront-Viewer-Header-Count 値が CloudFront-Viewer-Header-Order 内のヘッダー数と一致しない場合は、リクエストを拒否できます。

- CloudFront-Viewer-Header-Order — ビューワーのヘッダー名をリクエスト順にコロンで区切って示します。例: CloudFront-Viewer-Header-Order: Host:User-Agent:Accept:Accept-Encoding。文字数の制限である 7,680 文字を超えるヘッダーは切り捨てられます。
- CloudFront-Viewer-Header-Count — ビューワーのヘッダーの総数を示します。

その他の CloudFront ヘッダー

ビューワーのプロトコル、バージョン、JA3 フィンガープリント、および TLS 接続の詳細を特定するために、以下のヘッダーを追加できます。

- CloudFront-Forwarded-Proto - ビューワーのリクエストのプロトコル (HTTP または HTTPS) を示します。
- CloudFront-Viewer-Http-Version - ビューワーのリクエストの HTTP バージョンを示します。
- CloudFront-Viewer-JA3-Fingerprint — ビューワーの [JA3 フィンガープリント](#) を示します。JA3 フィンガープリントは、既知のクライアントからのリクエストであるか、マルウェアまたは悪意のあるボットであるか、想定された (許可リストに登録されている) アプリケーションであるかを判断するのに役立ちます。このヘッダーは、ビューワーの SSL/TLS Client Hello パケットに依存し、HTTPS リクエスト専用となっています。

Note

CloudFront-Viewer-JA3-Fingerprint の追加は、[オリジンリクエストポリシー](#) ではありますが、[キャッシュポリシー](#) ではありません。

- CloudFront-Viewer-TLS – SSL/TLS バージョン、暗号、およびビューワーと 間の接続に使用された SSL/TLS ハンドシェイクに関する情報が含まれます CloudFront。ヘッダー値は次の形式です。

SSL/TLS_version:cipher:handshake_information

handshake_information の場合、ヘッダーには以下のいずれかの値が含まれます。

- fullHandshake – SSL/TLS セッションに対して完全なハンドシェイクが実行された。
- sessionResumed – 前の SSL/TLS セッションが再開された。
- connectionReused – 前の SSL/TLS 接続が再利用された。

以下に示しているのは、このヘッダーの値の例です。

```
TLSv1.3:TLS_AES_128_GCM_SHA256:sessionResumed
```

```
TLSv1.2:ECDHE-ECDSA-AES128-GCM-SHA256:connectionReused
```

```
TLSv1.1:ECDHE-RSA-AES128-SHA256:fullHandshake
```

```
TLSv1:ECDHE-RSA-AES256-SHA:fullHandshake
```

このヘッダー値に含まれる SSL/TLS バージョンと暗号の完全なリストについては、「[the section called “ビューワーとの間でサポートされているプロトコルと暗号 CloudFront”](#)」を参照してください。

Note

CloudFront-Viewer-TLS の追加は、[オリジンリクエストポリシー](#)ではできますが、[キャッシュポリシー](#)ではできません。

オリジンリクエストポリシーとキャッシュポリシーの連携方法を理解する

CloudFront [オリジンリクエストポリシー](#)を使用して、がオリジンリクエスト CloudFront に送信するリクエストを制御できます。オリジンリクエストポリシーを使用するには、同じキャッシュ動作に[キャッシュポリシー](#)をアタッチする必要があります。キャッシュポリシーがないと、キャッシュ動作でオリジンリクエストポリシーを使用することはできません。詳細については、「[the section called “オリジンリクエストの制御”](#)」を参照してください。

オリジンリクエストポリシーとキャッシュポリシーは CloudFront連携して、オリジンリクエストに含まれる値を決定します。キャッシュキーに指定するすべての URL クエリ文字列、HTTP ヘッダー、および Cookie (キャッシュポリシーを使用) は、オリジンリクエストに自動的に含まれます。オリジンリクエストポリシーで指定した追加のクエリ文字列、ヘッダー、および Cookie もオリジンリクエストに含まれます (キャッシュキーには含まれません)。

オリジンリクエストポリシーとキャッシュポリシーには、互いに競合しているように見える設定があります。例えば、あるポリシーでは特定の値を許可し、別のポリシーではそれらをブロックするなど

です。次の表では、オリジンリクエストポリシーとキャッシュポリシーの設定を一緒に使用する場合に、オリジンリクエスト CloudFront に含まれる値について説明します。これらの設定は、一般的にすべてのタイプの値 (クエリ文字列、ヘッダー、Cookie) に適用されます。ただし、キャッシュポリシーですべてのヘッダーを指定したり、ヘッダーブロックリストを使用することはできません。

	オリジンリクエストポリシー			
	なし	[All] (すべて)	許可リスト	ブロックリスト

キャッシュポリシー

なし	すべてのオリジンリクエストに含まれるデフォルトを除き、ビューワーリクエストの値はオリジンリクエストには含まれません。詳細については、「 the section called “オリジンリクエストの制御” 」を参照してください。	ビューワーリクエストのすべての値がオリジンリクエストに含まれます。	オリジンリクエストポリシーで指定された値のみがオリジンリクエストに含まれます。	オリジンリクエストポリシーで指定された値を除くビューワーリクエストのすべての値がオリジンリクエストに含まれます。
[All] (すべて) 注意: キャッシュポリシーですべてのヘッダーを指定することはできません。	ビューワーリクエストのすべてのクエリ文字列と Cookie は、オリジンリクエストに含まれます。	ビューワーリクエストのすべての値がオリジンリクエストに含まれます。	ビューワーリクエストのすべてのクエリ文字列と Cookie、およびオリジンリクエストポリシーで指定されたヘッダーは、オリジンリクエストに含まれます。	ビューワーリクエストのすべてのクエリ文字列と Cookie は、オリジンリクエストポリシーのブロックリストで指定されているものも含め、すべてオリジンリクエストに含まれます。

オリジンリクエストポリシー				
	なし	[All] (すべて)	許可リスト	ブロックリスト
				れます。キャッシュポリシーの設定は、オリジンリクエストポリシーのブロックリストより優先されます。
許可リスト	ビューワーリクエストから指定された値のみが、オリジンリクエストに含まれます。	ビューワーリクエストのすべての値がオリジンリクエストに含まれます。	キャッシュポリシーまたはオリジンリクエストポリシーで指定されているすべての値は、オリジンリクエストに含まれます。	キャッシュポリシーで指定された値は、オリジンリクエストポリシーのブロックリストで同じ値が指定されている場合でも、オリジンリクエストに含まれません。キャッシュポリシーの許可リストは、オリジンリクエストポリシーの禁止リストより優先されます。

	オリジンリクエストポリシー			
	なし	[All] (すべて)	許可リスト	ブロックリスト
ブロックリスト 注意: キャッシュポリシーのブロックリストではヘッダーを指定できません。	指定されたものを除くビューワーリクエストのすべてのクエリ文字列と Cookie は、オリジンリクエストに含まれます。	ビューワーリクエストのすべての値がオリジンリクエストに含まれます。	オリジンリクエストポリシーで指定された値は、キャッシュポリシーのブロックリストに同じ値が指定されている場合でも、オリジンリクエストに含まれます。オリジンリクエストポリシーの許可リストは、キャッシュポリシーのブロックリストより優先されません。	キャッシュポリシーまたはオリジンリクエストポリシーで指定された値を除くビューワーリクエストのすべての値がオリジンリクエストに含まれます。

レスポンスでの CloudFront HTTP ヘッダーの追加または削除

ビューワーに送信するレスポンスの HTTP ヘッダーを変更する CloudFront ように を設定できます。ビューワーにレスポンスを送信する前に、オリジンから受信したヘッダーを削除したり、レスポンスにヘッダーを追加 CloudFront したりできます。これらの変更を行うために、コードを記述したり、オリジンを変更したりする必要はありません。

例えば、X-Powered-By や などのヘッダーを削除 Vary して、CloudFront がビューワーに送信するレスポンスにこれらのヘッダーを含めないようにすることができます。または、以下のような HTTP ヘッダーを追加できます。

- ブラウザのキャッシュを制御する Cache-Control ヘッダー。
- Cross-Origin Resource Sharing (CORS) を有効にする Access-Control-Allow-Origin ヘッダー また、他の CORS ヘッダーを追加することもできます。

- Strict-Transport-Security、Content-Security-Policy、X-Frame-Options のような一般的なセキュリティヘッダーのセット。
- を介したリクエストとレスポンスの両方のパフォーマンスとルーティングに関連する情報を表示する Server-Timing ヘッダー CloudFront。

HTTP レスポンスで CloudFront 追加または削除するヘッダーを指定するには、レスポンスヘッダーポリシーを使用します。レスポンスヘッダーポリシーを 1 つ以上のキャッシュ動作にアタッチし、キャッシュ動作に一致するリクエストに送信するレスポンスのヘッダー CloudFront を変更します。は、キャッシュから提供するレスポンスのヘッダーとオリジンから転送するヘッダー CloudFront を変更します。オリジンレスポンスに、レスポンスヘッダーポリシーに追加されるヘッダーが 1 つ以上含まれている場合、ポリシーは、オリジンから受信したヘッダー CloudFront を使用するか、レスポンスヘッダーポリシーのヘッダーでそのヘッダーを上書きするかを指定できます。

CloudFront は、一般的なユースケース向けに、マネージドポリシーと呼ばれる事前定義されたレスポンスヘッダーポリシーを提供します。[これらのマネージドポリシーを使用することも](#)、独自のポリシーを作成することもできます。AWS アカウントの複数のディストリビューションにおける複数のキャッシュ動作に単一のレスポンスヘッダーポリシーをアタッチすることができます。

詳細については、以下のトピックを参照してください。

トピック

- [レスポンスヘッダーポリシーの作成](#)
- [マネージドレスポンスヘッダーポリシーの使用](#)
- [レスポンスヘッダーポリシーの理解](#)

レスポンスヘッダーポリシーの作成

レスポンスヘッダーポリシーを使用して、Amazon が HTTP レスポンスで CloudFront 追加または削除する HTTP ヘッダーを指定できます。レスポンスヘッダーポリシーおよびそれらを使用する理由の詳細については、「[the section called “レスポンスヘッダーの追加または削除”](#)」を参照してください。

CloudFront コンソールでレスポンスヘッダーポリシーを作成できます。または、AWS Command Line Interface (AWS CLI) AWS CloudFormation、または CloudFront API を使用して作成することもできます。レスポンスヘッダーポリシーを作成したら、ディストリビューションの 1 つ以上のキャッシュ動作にアタッチします。

カスタムレスポンスヘッダーポリシーを作成する前に、[マネージドレスポンスヘッダーポリシー](#)の1つがユースケースに適合するかどうか確認します。適合する場合は、そのポリシーをキャッシュ動作にアタッチできます。これにより、独自のレスポンスヘッダーポリシーを作成したり管理したりする必要はありません。

Console

レスポンスヘッダーポリシーを作成するには (コンソール)

1. にサインインしAWS Management Console、コンソールのポリシーページのレスポンスヘッダータブに移動します CloudFront <https://console.aws.amazon.com/cloudfront/v4/home#/policies/responseHeaders>。
2. [Create response headers policy] (レスポンスヘッダーポリシーの作成) を選択します。
3. [Create response headers policy] (レスポンスヘッダーポリシーの作成) フォームで、次の操作を行います。
 - a. [Details] (詳細) パネルで、レスポンスヘッダーポリシーおよび (任意で) ポリシーの目的を説明する [Description] (説明) に「名前」を入力します。
 - b. [Cross-Origin Resource Sharing (CORS)] パネルで、[Configure CORS] (CORS を設定する) トグルを選択し、ポリシーに追加する CORS ヘッダーを設定します。設定済みヘッダーで、オリジンから が CloudFront 受信するヘッダーを上書きする場合は、オリジン上書きチェックボックスをオンにします。

CORS ヘッダー設定の詳細については、「[the section called “CORS ヘッダー”](#)」を参照してください。

- c. [Security headers] (セキュリティヘッダー) パネルで、トグルを選択し、ポリシーに追加する各セキュリティヘッダーを設定します。

セキュリティヘッダー設定の詳細については、「[the section called “セキュリティヘッダー”](#)」を参照してください。

- d. [Custom headers] (カスタムヘッダー) パネルで、ポリシーに含めるカスタムヘッダーを追加します。

カスタムヘッダー設定の詳細については、「[the section called “カスタムヘッダー”](#)」を参照してください。

- e. ヘッダーの削除パネル CloudFront で、オリジンのレスポンスから削除し、 が CloudFrontビューワーに送信するレスポンスに含めないヘッダーの名前を追加します。

ヘッダーの削除設定の詳細については、「[the section called “ヘッダーを削除”](#)」を参照してください。

- f. [Server-Timing header] (Server-Timing ヘッダー) パネルで、[Enable] (有効化) トグルを選択して、サンプリングレート (0~100 の数値) を入力します。

Server-Timing ヘッダーの詳細については、「[the section called “Server-Timing ヘッダー”](#)」を参照してください。

4. [Create] (作成) を選択して、ポリシーを作成します。

レスポンスヘッダーポリシーを作成したら、デイス CloudFront トリビューションのキャッシュ動作にアタッチできます。

既存のデイス トリビューションにレスポンスヘッダーポリシーをアタッチするには (コンソール)

1. の CloudFront コンソールでデイス トリビューションページを開きます<https://console.aws.amazon.com/cloudfront/v4/home#/distributions>。
2. 更新するデイス トリビューションを選択し、[動作] タブを選択します。
3. 更新するキャッシュ動作を選び、[Edit] (編集) を選択します。

または、新しいキャッシュ動作を作成するには、[動作を作成] を選択します。

4. [Response headers policy] (レスポンスヘッダーポリシー) の場合は、キャッシュ動作に追加するポリシーを選択します。
5. [Save changes] (変更を保存) を選択して、キャッシュ動作を更新します。新しいキャッシュ動作を作成する場合は、[Create behavior] (動作を作成) を選択します。

新しいデイス トリビューションにレスポンスヘッダーポリシーをアタッチするには (コンソール)

1. で CloudFront コンソールを開きます<https://console.aws.amazon.com/cloudfront/v4/home>。
2. [デイス トリビューションの作成] を選択します。
3. [Response headers policy] (レスポンスヘッダーポリシー) の場合は、キャッシュ動作に追加するポリシーを選択します。
4. その他のデイス トリビューションの設定を選択します。詳細については、「[the section called “指定する値”](#)」を参照してください。
5. [Create distribution] (デイス トリビューションを作成) を選択して、デイス トリビューションを作成します。

AWS CloudFormation

AWS CloudFormation を使用してレスポンスヘッダーポリシーを作成するには、`AWS::CloudFront::ResponseHeadersPolicy` リソースタイプを使用します。以下に例を示します。AWS CloudFormationレスポンスヘッダーポリシーを作成するための YAML 形式のテンプレート構文。

```
Type: AWS::CloudFront::ResponseHeadersPolicy
Properties:
  ResponseHeadersPolicyConfig:
    Name: EXAMPLE-Response-Headers-Policy
    Comment: Example response headers policy for the documentation
    CorsConfig:
      AccessControlAllowCredentials: false
      AccessControlAllowHeaders:
        Items:
          - '*'
      AccessControlAllowMethods:
        Items:
          - GET
          - OPTIONS
      AccessControlAllowOrigins:
        Items:
          - https://example.com
          - https://docs.example.com
      AccessControlExposeHeaders:
        Items:
          - '*'
      AccessControlMaxAgeSec: 600
      OriginOverride: false
    CustomHeadersConfig:
      Items:
        - Header: Example-Custom-Header-1
          Value: value-1
          Override: true
        - Header: Example-Custom-Header-2
          Value: value-2
          Override: true
    SecurityHeadersConfig:
      ContentSecurityPolicy:
        ContentSecurityPolicy: default-src 'none'; img-src 'self'; script-src
'self'; style-src 'self'; object-src 'none'; frame-ancestors 'none'
        Override: false
```

```
    ContentTypeOptions: # You don't need to specify a value for 'X-Content-Type-Options'.
                        # Simply including it in the template sets its value to
                        'nosniff'.
      Override: false
    FrameOptions:
      FrameOption: DENY
      Override: false
    ReferrerPolicy:
      ReferrerPolicy: same-origin
      Override: false
    StrictTransportSecurity:
      AccessControlMaxAgeSec: 63072000
      IncludeSubdomains: true
      Preload: true
      Override: false
    XSSProtection:
      ModeBlock: true # You can set ModeBlock to 'true' OR set a value for
ReportUri, but not both
      Protection: true
      Override: false
    ServerTimingHeadersConfig:
      Enabled: true
      SamplingRate: 50
    RemoveHeadersConfig:
      Items:
        - Header: Vary
        - Header: X-Powered-By
```

詳細については、「ユーザーガイド」の[AWS::CloudFront::ResponseHeaders「ポリシー」](#)を参照してください。AWS CloudFormation

CLI

AWS Command Line Interface (AWS CLI) でレスポンスヘッダーポリシーを作成するには、`aws cloudfront create-response-headers-policy` コマンドを使用します。コマンドの入力パラメータは、コマンドライン入力として個別に指定せずに、入力ファイルを使用して指定できます。

レスポンスヘッダーポリシーを作成するには (入力ファイルを含む CLI)

1. 次のコマンドを使用して、`response-headers-policy.yaml` という名前のファイルを作成します。このファイルには、`create-response-headers-policy` コマンドのすべての入力パラメータが含まれます。

```
aws cloudfront create-response-headers-policy --generate-cli-skeleton yml-input > response-headers-policy.yml
```

2. 先ほど作成した `response-headers-policy.yml` ファイルを開きます。ファイルを編集してポリシー名と必要なレスポンスヘッダーポリシー設定を指定し、ファイルを保存します。

レスポンスヘッダーポリシー設定の詳細については、「[the section called “レスポンスヘッダーポリシーの理解”](#)」を参照してください。

3. 次のコマンドを使用してレスポンスヘッダーポリシーを作成します。作成するポリシーでは、`response-headers-policy.yml` ファイルからの入力パラメータを使用します。

```
aws cloudfront create-response-headers-policy --cli-input-yml file://response-headers-policy.yml
```

コマンド出力の `Id` 値を書き留めます。これはレスポンスヘッダーポリシー ID です。ポリシーを CloudFront ディストリビューションのキャッシュ動作にアタッチする必要があります。

既存のディストリビューションにレスポンスヘッダーポリシーをアタッチするには (入力ファイルを含む CLI)

1. 次のコマンドを使用して、更新するディストリビューションの CloudFront ディストリビューション設定を保存します。`distribution_ID` をディストリビューション ID に置き換えます。

```
aws cloudfront get-distribution-config --id distribution_ID --output yml > dist-config.yml
```

2. 先ほど作成した `dist-config.yml` という名前のファイルを開きます。ファイルを編集し、レスポンスヘッダーポリシーを使用するように、次の変更をキャッシュ動作に加えます。

- キャッシュ動作で、ResponseHeadersPolicyId という名前のフィールドを追加します。フィールドの値には、ポリシーの作成後に書き留めたレスポンスヘッダーポリシー ID を使用します。
- ETag フィールドの名前を IfMatch に変更します。ただし、フィールドの値は変更しないでください。

完了したら、ファイルを保存します。

3. レスポンスヘッダーポリシーを使用するようにディストリビューションを更新するには、次のコマンドを使用します。*distribution_ID* をディストリビューション ID に置き換えます。

```
aws cloudfront update-distribution --id distribution_ID --cli-input-yaml file://  
dist-config.yaml
```

新しいディストリビューションにレスポンスヘッダーポリシーをアタッチするには (入力ファイルを含む CLI)

1. 次のコマンドを使用して、distribution.yaml という名前のファイルを作成します。このファイルには、create-distribution コマンドのすべての入力パラメータが含まれます。

```
aws cloudfront create-distribution --generate-cli-skeleton yaml-input >  
distribution.yaml
```

2. 先ほど作成した distribution.yaml ファイルを開きます。デフォルトのキャッシュ動作の [ResponseHeadersPolicyId] フィールドに、ポリシーの作成後に書き留めたレスポンスヘッダーポリシー ID を入力します。ファイルの編集を続行して必要なディストリビューション設定を指定し、完了したらファイルを保存します。

ディストリビューション設定の詳細については、「[ディストリビューションを作成または更新する場合に指定する値](#)」を参照してください。

3. 次のコマンドを使用して、distribution.yaml ファイルの入力パラメータを使用し、ディストリビューションを作成します。

```
aws cloudfront create-distribution --cli-input-yaml file://distribution.yaml
```

API

CloudFront API を使用してレスポンスヘッダーポリシーを作成するには、[を使用します](#) [CreateResponseHeadersPolicy](#)。この API コールで指定するフィールドの詳細については、「[the section called “レスポンスヘッダーポリシーの理解”](#)」、および AWS SDK やその他の API クライアントの API リファレンスドキュメントを参照してください。

レスポンスヘッダーポリシーを作成したら、次の API コールのいずれかを使用して、それをキャッシュ動作にアタッチできます。

- 既存のディストリビューションのキャッシュ動作にアタッチするには、[を使用します](#) [UpdateDistribution](#)。
- 新しいディストリビューションのキャッシュ動作にアタッチするには、[を使用します](#) [CreateDistribution](#)。

これらの API コールの両方について、キャッシュ動作内で、ResponseHeadersPolicyId フィールドにレスポンスヘッダーポリシー ID を指定します。これらの API コールで指定するその他フィールドの詳細については、「[ディストリビューションを作成または更新する場合に指定する値](#)」と、AWS SDK またはその他 API クライアントの API リファレンスドキュメントを参照してください。

マネージドレスポンスヘッダーポリシーの使用

CloudFront レスポンスヘッダーポリシーを使用すると、Amazon CloudFront がビューワーに送信するレスポンスで削除または追加する HTTP ヘッダーを指定できます。レスポンスヘッダーポリシーおよびそれらを使用する理由の詳細については、「[the section called “レスポンスヘッダーの追加または削除”](#)」を参照してください。

CloudFront には、CloudFront ディストリビューションのキャッシュ動作にアタッチできるマネージドレスポンスヘッダーポリシーが用意されています。マネージドレスポンスヘッダーポリシーを使用すると、独自のポリシーを記述したり、維持したりする必要がありません。このマネージドポリシーには、一般的ユースケース用の HTTP レスポンスヘッダーのセットが含まれています。

トピック

- [マネージドレスポンスヘッダーポリシーのアタッチ](#)

• [使用可能なマネージドレスポンスヘッダーポリシー](#)

マネージドレスポンスヘッダーポリシーのアタッチ

マネージドレスポンスヘッダーポリシーを使用するには、ディストリビューションのキャッシュ動作にそのポリシーをアタッチします。このプロセスは、カスタムレスポンスヘッダーポリシーを作成するときと同じです。ただし、新しいポリシーを作成する代わりに、マネージドポリシーの1つをアタッチします。ポリシーは、名前 (コンソールを使用) または ID (AWS CloudFormation、AWS CLI、または AWS SDK を使用) を使用してアタッチします。名前と ID は、次のセクションに記載されています。

詳細については、「[the section called “レスポンスヘッダーポリシーの作成”](#)」を参照してください。

使用可能なマネージドレスポンスヘッダーポリシー

次のトピックでは、使用可能なマネージドレスポンスヘッダーポリシーについて説明します。

トピック

- [CORS と SecurityHeadersPolicy](#)
- [CORS-With-Preflight](#)
- [CORS-with-preflight-and-SecurityHeadersPolicy](#)
- [SecurityHeadersPolicy](#)
- [SimpleCORS](#)

CORS と SecurityHeadersPolicy

[CloudFront コンソールでこのポリシーを表示する](#)

このマネージドポリシーを使用して、オリジンからの単一の CORS リクエストが許可されます。このポリシーは、ビューワー CloudFront に送信するすべてのレスポンスにセキュリティヘッダーのセットも追加します。このポリシーによって、[the section called “SimpleCORS”](#) ポリシーおよび [the section called “SecurityHeadersPolicy”](#) ポリシーを1つにまとめます。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

e61eb60c-9c35-4d20-a928-2b84e02af89c

ポリシー設定

	ヘッダー名	ヘッダー値	オーバーライド オリジンです か。
CORS ヘッ ダー:	Access-Control-Allow- Origin	*	いいえ
セキュリテイ ヘッダー:	Referrer-Policy	strict-origin- when-cross-or igin	いいえ
	Strict-Transport-S ecurity	max-age=3 1536000	いいえ
	X-Content-Type-Options	nosniff	はい
	X-Frame-Options	SAMEORIGIN	いいえ
	X-XSS-Protection	1; mode=block	いいえ

CORS-With-Preflight

[CloudFront コンソールでこのポリシーを表示する](#)

このマネージドポリシーを使用して、プリフライトリクエストを含むオリジンからの CORS リクエストが許可されます。プリフライトリクエスト (HTTP OPTIONSメソッドを使用) の場合、は次の 3 つのヘッダーすべてをレスポンス CloudFront に追加します。シンプルな CORS リクエストの場合、は Access-Control-Allow-Originヘッダーのみ CloudFront を追加します。

がオリジンから CloudFront 受け取るレスポンスにこれらのヘッダーのいずれかが含まれている場合、はビューワーへのレスポンスで受信したヘッダー (およびその値) CloudFront を使用します。CloudFront はこのポリシーの ヘッダーを使用しません。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

5cc3b908-e619-4b99-88e5-2cf7f45965bd

ポリシー設定

	ヘッダー名	ヘッダー値	オーバーライド オリジンです か。
CORS ヘッ ダー:	Access-Control-Allow- Methods	DELETE, GET, HEAD, OPTIONS, PATCH, POST, PUT	いいえ
	Access-Control-Allow- Origin	*	
	Access-Control-Expose- Headers	*	

CORS-with-preflight-and-SecurityHeadersPolicy

[CloudFront コンソールでこのポリシーを表示する](#)

このマネージドポリシーを使用して、オリジンからの CORS リクエストが許可されます。これには、プリフライトリクエストが含まれます。このポリシーは、[がビューワー CloudFront に送信するすべてのレスポンスにセキュリティヘッダーのセットも追加します。](#)このポリシーによって、[the section called “CORS-With-Preflight”](#) ポリシーおよび [the section called “SecurityHeadersPolicy”](#) ポリシーを 1 つにまとめます。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

eaab4381-ed33-4a86-88ca-d9558dc6cd63

ポリシー設定

	ヘッダー名	ヘッダー値	オーバーライド オリジンです か。
CORS ヘッ ダー:	Access-Control-Allow- Methods	DELETE, GET, HEAD, OPTIONS, PATCH, POST, PUT	いいえ

	ヘッダー名	ヘッダー値	オーバーライド オリジンです か。
	Access-Control-Allow-Origin	*	
	Access-Control-Expose-Headers	*	
セキュリティ ヘッダー:	Referrer-Policy	strict-origin-when-cross-origin	いいえ
	Strict-Transport-Security	max-age=31536000	いいえ
	X-Content-Type-Options	nosniff	はい
	X-Frame-Options	SAMEORIGIN	いいえ
	X-XSS-Protection	1; mode=block	いいえ

SecurityHeadersPolicy

[CloudFront コンソールでこのポリシーを表示する](#)

このマネージドポリシーを使用して、 がビューワー CloudFront に送信するすべてのレスポンスにセキュリティヘッダーのセットを追加します。これらのセキュリティヘッダーの詳細については、 [Mozilla のウェブセキュリティに関するガイドライン](#) を参照してください。

このレスポンスヘッダーポリシーでは、 はすべてのレスポンスX-Content-Type-Options: nosniffに CloudFront を追加します。これは、オリジンから受信したレスポンス CloudFrontにこのヘッダーが含まれ、含まれていない場合に当てはまります。このポリシーの他のすべてのヘッダーでは、 がオリジンから CloudFront 受信するレスポンスに ヘッダーが含まれている場合、 はビューワーへのレスポンスで受信したヘッダー (およびその値) CloudFront を使用します。CloudFront はこのポリシーのヘッダーを使用しません。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

67f7725c-6f97-4210-82d7-5512b31e9d03

ポリシー設定

	ヘッダー名	ヘッダー値	オーバーライドオリジンですか。
セキュリティ ヘッダー:	Referrer-Policy	strict-origin-when-cross-origin	いいえ
	Strict-Transport-Security	max-age=31536000	いいえ
	X-Content-Type-Options	nosniff	はい
	X-Frame-Options	SAMEORIGIN	いいえ
	X-XSS-Protection	1; mode=block	いいえ

SimpleCORS

[CloudFront コンソールでこのポリシーを表示する](#)

このマネージドポリシーを使用して、オリジンからの[単一の CORS リクエスト](#)が許可されます。このポリシーでは、は単純な CORS リクエストのすべてのレスポンス Access-Control-Allow-Origin: * にヘッダー CloudFront を追加します。

がオリジンから CloudFront 受け取るレスポンスに Access-Control-Allow-Originヘッダーが含まれている場合、は viewer. CloudFront doeses へのレスポンスでそのヘッダー (およびその値) CloudFront を使用します。このポリシーのヘッダーは使用しません。

AWS CloudFormation、AWS CLI、または CloudFront API を使用する場合、このポリシーの ID は次のとおりです。

60669652-455b-4ae9-85a4-c4c02393f86c

ポリシー設定

	ヘッダー名	ヘッダー値	オーバーライド オリジンです か。
CORS ヘッ ダー:	Access-Control-Allow- Origin	*	いいえ

レスポンスヘッダーポリシーの理解

レスポンスヘッダーポリシーを使用して、Amazon がビューワーに送信するレスポンスで CloudFront 削除または追加する HTTP ヘッダーを指定できます。レスポンスヘッダーポリシーおよびそれらを使用する理由の詳細については、「[the section called “レスポンスヘッダーの追加または削除”](#)」を参照してください。

以下のトピックでは、レスポンスヘッダーポリシーの設定について説明します。設定はカテゴリに分類され、そのことについて次のトピックで説明します。

トピック

- [ポリシーの詳細 \(メタデータ\)](#)
- [CORS ヘッダー](#)
- [セキュリティヘッダー](#)
- [カスタムヘッダー](#)
- [ヘッダーを削除](#)
- [Server-Timing ヘッダー](#)

ポリシーの詳細 (メタデータ)

ポリシーの詳細設定には、レスポンスヘッダーポリシーに関するメタデータが含まれます。

- 名前 - レスポンスヘッダーポリシーを識別するための名前。コンソールでは、名前を使用して、このポリシーをキャッシュ動作にアタッチします。
- 説明 (オプション) - レスポンスヘッダーポリシーを説明するコメント。これはオプションですが、このポリシーの目的を特定するのに役立ちます。

CORS ヘッダー

Cross-Origin Resource Sharing (CORS) 設定を使用して、レスポンスヘッダーポリシーに CORS ヘッダーを追加および設定できます。

このリストは、レスポンスヘッダーポリシーの設定および有効な値の指定方法に焦点を合わせています。これらの各ヘッダーと、実際の CORS リクエストとレスポンスの使用の詳細については、MDN ウェブドキュメントと [CORS プロトコルの仕様](#)にある「[クロスオリジンリソース共有](#)」を参照してください。

Access-Control-Allow-Credentials

これは、CORS リクエストへのレスポンスに Access-Control-Allow-Credentialsヘッダー CloudFront を追加するかどうかを決定するブール設定 (true または false) です。この設定が true に設定されている場合 true、は CORS リクエストへのレスポンスに Access-Control-Allow-Credentials: trueヘッダー CloudFront を追加します。それ以外の場合は、このヘッダーをレスポンスに追加 CloudFront しません。

Access-Control-Allow-Headers

CORS プリフライトリクエストへのレスポンスで、がヘッダーの値 CloudFront として使用する Access-Control-Allow-Headersヘッダー名を指定します。この設定の有効な値には、HTTP ヘッダー名、またはすべてのヘッダーを許可することを示すワイルドカード文字 (*) が含まれます。Authorization ヘッダーにはワイルドカードを使用できず、常に明示的にリストする必要があることに注意してください。

ワイルドカード文字の有効な使用例を次の表に示します。

例	一致する	一致しない
x-amz-*	x-amz-test x-amz-	x-amz
x-*-amz	x-test-amz x--amz	
*	Authorization を除くすべてのヘッダー	Authorization

Access-Control-Allow-Methods

CORS プリフライトリクエストへのレスポンスの Access-Control-Allow-Methods ヘッダーの値としてが CloudFront 使用する HTTP メソッドを指定します。有効な値は、GET、DELETE、HEAD、OPTIONS、PATCH、POST、PUT、ALL です。ALL は、リストされているすべての HTTP メソッドが含まれている特別な値です。

Access-Control-Allow-Origin

が Access-Control-Allow-Origin レスポンスヘッダー CloudFront で使用できる値を指定します。この設定の有効な値には、特定のオリジン (<http://www.example.com> など)、またはすべてのオリジンを許可することを示すワイルドカード文字 (*) が含まれます。次の表の例を参照してください。

Note

ワイルドカード文字 (*) は、ドメイン (*.example.org) の左端部分として使用できません。

ワイルドカード文字 (*) は以下の位置では使用できません。

- 最上位ドメイン (example.*)
- サブドメイン (test.*.example.org) の右側
- 規約の内部 (exa*mples.org)

ワイルドカード文字の有効な使用例を次の表に示します。

例	一致する	一致しない
http://*.example.org	http://www.example.com	https://test.example.org
	http://test.example.org	https://test.example.org:123
	http://test.example.org:123	
*.example.org	test.example.org	

例	一致する	一致しない
	test.test.example.org .example.org http://test.example.org https://test.example.org http://test.example.org:123 https://test.example.org:123	
example.org	http://example.org https://example.org	
http://example.org		https://example.org http://example.org:123
http://example.org:*	http://example.org:123 http://example.org	
http://example.org:1*3	http://example.org:123 http://example.org:1893 http://example.org:13	

例	一致する	一致しない
.example.org:1	test.example.org:123	

Access-Control-Expose-Headers

CORS リクエストへのレスポンスでヘッダーの値としてが CloudFront 使用する Access-Control-Expose-Headers ヘッダー名を指定します。この設定の有効な値には、HTTP ヘッダー名、またはワイルドカード文字 (*) が含まれます。

Access-Control-Max-Age

CORS プリフライトリクエストへのレスポンスの Access-Control-Max-Age ヘッダーの値として CloudFront を使用する秒数。

オリジンのオーバーライド

これは、オリジンからのレスポンスに、ポリシーにも含まれている CORS ヘッダーの 1 つが含まれている場合 CloudFront の動作を決定するブール値設定 (true または false) です。

この設定が true に設定され、オリジンレスポンスにポリシーにも CORS ヘッダーが含まれている場合、はビューワーに送信するレスポンスにポリシーの CORS ヘッダー CloudFront を追加します。オリジンから受け取ったヘッダーは無視されます。

この設定が false で、オリジンレスポンスにポリシーにも CORS ヘッダーが含まれている場合、ビューワーに送信するレスポンスにオリジンから受信した CORS ヘッダー CloudFront が含まれます。

オリジンレスポンスにポリシー内の CORS ヘッダーが含まれていない場合、はポリシー内の CORS ヘッダーを、この設定が true または false に設定されている場合に viewer. CloudFront does に送信するレスポンス CloudFront に追加します false。

セキュリティヘッダー

セキュリティヘッダーの設定を使用して、レスポンスヘッダーポリシーでいくつかのセキュリティ関連の HTTP レスポンスヘッダーを追加および設定できます。

このリストは、レスポンスヘッダーポリシーの設定および有効な値の指定方法を示しています。これらの各ヘッダー、および実際の HTTP レスポンスでの使用方法の詳細については、MDN ウェブドキュメントへのリンクを参照してください。

Content-Security-Policy

がContent-Security-Policyレスポンスヘッダーの値 CloudFront として使用するコンテンツセキュリティポリシーディレクティブを指定します。

このヘッダーと有効なポリシーディレクティブの詳細については、MDN Web ドキュメントにある「[Content-Security-Policy](#)」を参照してください。

Note

Content-Security-Policy ヘッダー値は 1,783 文字に限定されます。

リファラーポリシー

がReferrer-Policyレスポンスヘッダーの値 CloudFront として使用するリファラポリシーディレクティブを指定します。この設定の有効値は、no-referrer、no-referrer-when-downgrade、origin、origin-when-cross-origin、same-origin、strict-origin、strict-origin-when-cross-origin、unsafe-url です。

このヘッダーおよびこれらのディレクティブの詳細については、MDN Web ドキュメントにある「[Referrer-Policy](#)」を参照してください。

Strict-Transport-Security

がStrict-Transport-Securityレスポンスヘッダーの値 CloudFront として使用するディレクティブと設定を指定します。この設定では、以下を個別に指定します。

- このヘッダーの max-ageディレクティブの値として CloudFront を使用する秒数
- のブール設定 (true または false) preload。このヘッダーの値に preloadディレクティブ CloudFront を含めるかどうかを決定します。
- のブール設定 (true または false) includeSubDomains。このヘッダーの値に includeSubDomainsディレクティブを含めるかどうか CloudFrontを決定します。

このヘッダーおよびこれらのディレクティブの詳細については、MDN Web ドキュメントにある「[Strict-Transport-Security](#)」を参照してください。

X-Content-Security-Policy

これは、 が X-Content-Type-Optionsヘッダーをレスポンス CloudFront に追加するかどうかを決定するブール設定 (true または false) です。この設定が の場合true、は X-Content-

Type-Options: nosniffヘッダーをレスポンス CloudFront に追加します。そう CloudFront しないと、はこのヘッダーを追加しません。

このヘッダーの詳細については、MDN Web ドキュメントにある「[X-Content-Type-Options](#)」を参照してください。

X-Frame-Options

がX-Frame-Optionsレスポンスヘッダーの値 CloudFront として使用する ディレクティブを指定します。この設定の有効値は、DENY または SAMEORIGIN です。

このヘッダーおよびこれらのディレクティブの詳細については、MDN Web ドキュメントにある「[X-Frame-Options](#)」を参照してください。

X-XSS-Protection

がX-XSS-Protectionレスポンスヘッダーの値 CloudFront として使用するディレクティブと設定を指定します。この設定では、以下を個別に指定します。

- X-XSS-Protection (XSS フィルタリングを無効にする) または 0 (XSS フィルタリングを有効にする) についての 1 の設定設定
- のブール設定 (true または false) block。このヘッダーの値に mode=blockディレクティブ CloudFront を含めるかどうかを決定します。
- レポート URI。このヘッダーの値に report=*reporting URI*ディレクティブ CloudFront を含めるかどうかを決定します。

block のために true を指定することができます。または、レポートの URI を指定することもできますが、両方一緒に指定することはできません。このヘッダーおよびこれらのディレクティブの詳細については、MDN Web ドキュメントにある「[X-XSS-Protection](#)」を参照してください。

オリジンのオーバーライド

これらの各セキュリティヘッダー設定には、オリジンからのレスポンスにそのヘッダーが含まれている場合 CloudFront の動作を決定するブール値設定 (true または false) が含まれています。

この設定が に設定trueされ、オリジンレスポンスに ヘッダーが含まれている場合、 はビューワーに送信するレスポンスにポリシーの ヘッダー CloudFront を追加します。オリジンから受け取ったヘッダーは無視されます。

この設定が に設定falseされ、オリジンレスポンスに ヘッダーが含まれている場合、 はビューワーに送信するレスポンスにオリジンから受信したヘッダー CloudFront を含めます。

オリジンレスポンスにヘッダーが含まれていない場合、は、ビューワーに送信するレスポンスにポリシーのヘッダー CloudFront を追加します。この設定が true または に設定されていると、この設定が CloudFront 実行されます false。

カスタムヘッダー

カスタムヘッダー設定を使用して、レスポンスヘッダーポリシーでカスタム HTTP ヘッダーを追加および設定できます。は、ビューワーに返されるすべてのレスポンスにこれらのヘッダー CloudFront を追加します。カスタムヘッダーごとにヘッダーの値も指定しますが、値の指定はオプションです。これは、CloudFront が値なしでレスポンスヘッダーを追加できるためです。

また各カスタムヘッダーには、独自のオリジンのオーバーライドの設定があります。

- この設定が に設定 true され、オリジンレスポンスにポリシー内のカスタムヘッダーが含まれている場合、はビューワーに送信するレスポンスにポリシー内のカスタムヘッダー CloudFront を追加します。オリジンから受け取ったヘッダーは無視されます。
- この設定が false で、オリジンレスポンスにポリシー内のカスタムヘッダーが含まれている場合、はビューワーに送信するレスポンスに、オリジンから受信したカスタムヘッダー CloudFront を含めます。
- オリジンレスポンスにポリシー内のカスタムヘッダーが含まれていない場合、は、この設定が true または に設定されているときに、ビューワー CloudFront does に送信するレスポンスにポリシー内のカスタムヘッダー CloudFront を追加します false。

ヘッダーを削除

オブジェクトが の CloudFront キャッシュから提供されるか、オリジンから送信されるかにかかわらず、がビューワー CloudFront に送信するすべてのレスポンスにヘッダーが含まれないように、オリジンから CloudFront 受信するレスポンスから削除するヘッダーを指定できます。は、ビューワーに送信するすべてのレスポンスからヘッダー CloudFront を削除します。例えば、X-Powered-By やなどのブラウザでは使用されていないヘッダーを削除して Vary、がビューワーに送信するレスポンスからこれらのヘッダー CloudFront を削除できます。

レスポンスヘッダーポリシーを使用して削除するヘッダーを指定すると、は最初にヘッダー CloudFront を削除し、レスポンスヘッダーポリシーの他のセクション (CORS ヘッダー、セキュリティヘッダー、カスタムヘッダーなど) で指定されているヘッダーを追加します。削除するヘッダーを指定し、ポリシーの別のセクションにも同じヘッダーを追加すると、はビューワーに送信するレスポンスにヘッダー CloudFront を含めます。

Note

レスポンスヘッダーポリシーを使用して、オリジンから CloudFront 受信した ヘッダー Server と Date ヘッダーを削除し、これらのヘッダー (オリジンから受信したもの) をがビューワー CloudFront に送信するレスポンスに含めないようにすることができます。ただし、その場合、はビューワーに送信するレスポンスに独自のバージョンのこれらのヘッダー CloudFront を追加します。が追加する Server CloudFront ヘッダーの場合、ヘッダーの値はです CloudFront。

削除できないヘッダー

以下のヘッダーは、レスポンスヘッダーポリシーを使用して削除することはできません。これらのヘッダーをレスポンスヘッダーポリシーの [Remove headers] (ヘッダーを削除) セクション (API の ResponseHeadersPolicyRemoveHeadersConfig) で指定すると、エラーが表示されます。

- Connection
- Content-Encoding
- Content-Length
- Expect
- Host
- Keep-Alive
- Proxy-Authenticate
- Proxy-Authorization
- Proxy-Connection
- Trailer
- Transfer-Encoding
- Upgrade
- Via
- Warning
- X-Accel-Buffering
- X-Accel-Charset
- X-Accel-Limit-Rate
- X-Accel-Redirect

- X-Amz-Cf-.*
- X-Amzn-Auth
- X-Amzn-Cf-Billing
- X-Amzn-Cf-Id
- X-Amzn-Cf-Xff
- X-Amzn-ErrorType
- X-Amzn-Fle-Profile
- X-Amzn-Header-Count
- X-Amzn-Header-Order
- X-Amzn-Lambda-Integration-Tag
- X-Amzn-RequestId
- X-Cache
- X-Edge-.*
- X-Forwarded-Proto
- X-Real-IP

Server-Timing ヘッダー

Server-Timing ヘッダー設定を使用して、 から送信された HTTP レスポンスの Server-Timing ヘッダーを有効にします CloudFront。このヘッダーを使用して、 CloudFront とオリジンの動作とパフォーマンスに関するインサイトを得るのに役立つメトリクスを表示できます。例えば、 キャッシュヒットを処理したキャッシュレイヤーを確認できます。または、 キャッシュミスがある場合に、 オリジンからの最初のバイトレイテンシーを確認できます。Server-Timing ヘッダーのメトリクスは、 問題のトラブルシューティングや、 CloudFront またはオリジン設定の効率のテストに役立ちます。

で Server-Timing ヘッダーを使用する方法の詳細については CloudFront、 以下のトピックを参照してください。

Server-Timing ヘッダーを有効にするには、 [レスポンスヘッダーポリシーを作成 \(または編集\) します](#)。

トピック

- [サンプリングレートと Pragma リクエストヘッダー](#)

- [オリジンからの Server-Timing ヘッダー](#)
- [Server-Timing ヘッダーのメトリクス](#)
- [Server-Timing ヘッダーの例](#)

サンプリングレートと Pragma リクエストヘッダー

レスポンスヘッダーポリシーで Server-Timing ヘッダーを有効にするときは、サンプリングレートも指定します。サンプリングレートは、Server-Timingヘッダー CloudFront を追加するレスポンスの割合を指定する 0~100 (両端を含む) の数値です。サンプリングレートを 100 に設定すると、は、レスポンスServer-Timingヘッダーポリシーがアタッチされているキャッシュ動作に一致するすべてのリクエストの HTTP レスポンスに ヘッダー CloudFront を追加します。50 に設定すると、はキャッシュ動作に一致するリクエストのレスポンスの 50% に ヘッダー CloudFront を追加します。サンプリングレートは、0~100 の任意の数値 (小数点以下 4 桁まで) に設定できます。

サンプリングレートが 100 未満の数値に設定されている場合、どのレスポンスがServer-Timingヘッダー CloudFront を追加するかを制御できず、パーセンテージのみ制御できます。ただし、HTTP リクエストで値を server-timing に設定して Pragma ヘッダーを追加し、そのリクエストへのレスポンスで Server-Timing ヘッダーを受け取ることができます。これは、サンプリングレートの設定とは無関係に機能します。サンプリングレートがゼロ (0) に設定されている場合でも、リクエストに Server-Timingヘッダーが含まれている場合、は Pragma: server-timingヘッダーをレスポンス CloudFront に追加します。

オリジンからの Server-Timing ヘッダー

キャッシュミスが発生してリクエストをオリジン CloudFront に転送すると、オリジンはへのレスポンスに Server-Timingヘッダーを含める場合があります CloudFront。この場合、はオリジンから受信した Server-Timingヘッダーに[メトリクス](#) CloudFront を追加します。がビューワー CloudFront に送信するレスポンスには、オリジンから取得した値とが CloudFront 追加したメトリクスを含む単一のServer-Timingヘッダーが含まれています。オリジンからのヘッダー値は、最後にあるか、がヘッダー CloudFront に追加する 2 つのメトリクスセットの間にある場合があります。

キャッシュヒットがある場合、がビューワー CloudFront に送信するレスポンスには、Server-Timingヘッダー値に CloudFront メトリクスのみを含む単一のヘッダーが含まれます (オリジンからの値は含まれません)。

Server-Timing ヘッダーのメトリクス

が Server-Timingヘッダーを HTTP レスポンス CloudFront に追加すると、ヘッダーの値には、CloudFront およびオリジンの動作とパフォーマンスに関するインサイトを得るのに役立つ 1 つ以上

のメトリクスが含まれます。次のリストには、すべてのメトリクスと想定される値が含まれています。Server-Timing ヘッダーには、を介したリクエストとレスポンスの性質に応じて、これらのメトリクスの一部のみが含まれます CloudFront。

これらのメトリクスのいくつかは、名前のみ (値なし) の Server-Timing ヘッダーに含まれます。その他は名前と価値です。メトリクスに値がある場合、名前と値はセミコロン (;) で区切られます。ヘッダーに複数のメトリクスが含まれている場合、メトリクスはカンマ (,) で区切られます。

cdn-cache-hit

CloudFront は、オリジンにリクエストを行わずにキャッシュからのレスポンスを提供しました。

cdn-cache-refresh

CloudFront は、キャッシュされたオブジェクトがまだ有効であることを確認するリクエストをオリジンに送信した後、キャッシュからのレスポンスを提供しました。この場合、はオリジンから完全なオブジェクトを取得 CloudFront できませんでした。

cdn-cache-miss

CloudFront はキャッシュからのレスポンスを提供しませんでした。この場合、CloudFront はレスポンスを返す前にオリジンから完全な オブジェクトを要求しました。

cdn-pop

リクエストを処理した CloudFront Point of Presence (POP) を表す値が含まれます。

cdn-rid

リクエストの CloudFront 一意の識別子を持つ値が含まれます。このリクエスト識別子 (RID) は、AWS Support に関する問題のトラブルシューティングに使用できます。

cdn-hit-layer

このメトリクスは、がオリジンにリクエストを行わずにキャッシュからのレスポンス CloudFront を提供するとき存在します。次のいずれかの値が含まれます。

- EDGE - POP ロケーションからキャッシュされたレスポンス CloudFront を提供しました。
- REC – [リージョン別エッジキャッシュ \(REC\) ロケーションからのキャッシュ](#)されたレスポンス CloudFront を提供しました。
- Origin Shield – [Origin Shield](#) として動作する REC からのキャッシュされたレスポンス CloudFront を提供しました。

cdn-upstream-layer

がオリジンから完全なオブジェクトを CloudFront リクエストすると、このメトリクスが存在し、次のいずれかの値が含まれます。

- EDGE – POP ロケーションがリクエストをオリジンに直接送信しました。
- REC – REC ロケーションがリクエストをオリジンに直接送信しました。
- オリジンシールド – [オリジンシールド](#)として機能している REC が、リクエストをオリジンに直接送信しました。

cdn-upstream-dns

オリジンの DNS レコードの取得にかかったミリ秒数を示す値が含まれます。値がゼロ (0) の場合は、 がキャッシュされた DNS 結果 CloudFront を使用したか、既存の接続を再利用したことを示します。

cdn-upstream-connect

オリジン DNS リクエストが完了してから、オリジンへの TCP 接続 (および該当する場合は TLS 接続) が完了するまでのミリ秒数を示す値が含まれます。値がゼロ (0) の場合は、 が既存の接続を CloudFront 再利用したことを示します。

cdn-upstream-fbl

オリジン HTTP リクエストが完了してから、オリジンからのレスポンスで最初のバイトを受信するまでのミリ秒数 (最初のバイトレイテンシー) を示す値が含まれます。

cdn-downstream-fbl

エッジロケーションがリクエストの受信を終了してから、レスポンスの最初のバイトをビューワーに送信するまでのミリ秒を示す値が含まれます。

Server-Timing ヘッダーの例

Server-Timing ヘッダー設定が有効になってい CloudFront の場合にビューワーが受信する可能性のある Server-Timing ヘッダーの例を次に示します。

Example – キャッシュミス

次の例は、リクエストされたオブジェクトが CloudFront キャッシュにない場合にビューワーが受け取る可能性のある Server-Timing ヘッダーを示しています。

```
Server-Timing: cdn-upstream-layer;desc="EDGE",cdn-upstream-dns;dur=0,cdn-upstream-connect;dur=114,cdn-upstream-fbl;dur=177,cdn-cache-miss,cdn-pop;desc="PHX50-C2",cdn-rid;desc="yNPsyYn7skvTzwWkq3Wcc8Nj_foxUjQUe9H1ifslzWhb0w7aLbFvGg==",cdn-downstream-fbl;dur=436
```

この Server-Timing ヘッダーは以下を示します。

- オリジンリクエストが、Point of Presence (POP) CloudFront ロケーション () から送信された `cdn-upstream-layer;desc="EDGE"`。
- CloudFront は、オリジン () のキャッシュされた DNS 結果を使用しました `cdn-upstream-dns;dur=0`。
- がオリジン () CloudFront への TCP (および該当する場合は TLS) 接続を完了するまでに 114 ミリ秒かかりました `cdn-upstream-connect;dur=114`。
- リクエスト () の完了後、 がオリジンからレスポンスの最初のバイトを受信する CloudFront までに 177 ミリ秒かかりました `cdn-upstream-fbl;dur=177`。
- リクエストされたオブジェクトは CloudFront のキャッシュ () にありませんでした `cdn-cache-miss`。
- リクエストは、コード PHX50-C2 (`cdn-pop;desc="PHX50-C2"`) によって識別されるエッジロケーションで受信されました。
- このリクエストの CloudFront 一意の ID は `yNPsyYn7skvTzwWkq3Wcc8Nj_foxUjQUe9H1ifslzWhb0w7aLbFvGg==` () でした `cdn-rid;desc="yNPsyYn7skvTzwWkq3Wcc8Nj_foxUjQUe9H1ifslzWhb0w7aLbFvGg=="`。
- ビューワーリクエスト () を受信した後、レスポンスの最初のバイトがビューワーに送信 CloudFront されるまでに 436 ミリ秒かかりました `cdn-downstream-fbl;dur=436`。

Example – キャッシュヒット

次の例は、リクエストされたオブジェクトが の CloudFront キャッシュにあるときにビューワーが受信できる Server-Timing ヘッダーを示しています。

```
Server-Timing: cdn-cache-hit,cdn-pop;desc="SEA19-C1",cdn-rid;desc="nQBz4aJU2kP9iC3KHEq7vFxfMozu-VYBwGzkw9di0peVc7xsrLKj-g==",cdn-hit-layer;desc="REC",cdn-downstream-fbl;dur=137
```

この Server-Timing ヘッダーは以下を示します。

- リクエストされたオブジェクトはキャッシュ (cdn-cache-hit) 内にありました。
- リクエストは、コード SEA19-C1 (cdn-pop;desc="SEA19-C1") によって識別されるエッジロケーションで受信されました。
- このリクエストの CloudFront 一意の ID は nQBz4aJU2kP9iC3KHEq7vFxfMoZu-VYBwGzkW9di0peVc7xsrLKj-g== () でしたcdn-rid;desc="nQBz4aJU2kP9iC3KHEq7vFxfMoZu-VYBwGzkW9di0peVc7xsrLKj-g=="。
- リクエストされたオブジェクトは、リージョン別エッジキャッシュ (REC) ロケーション (cdn-hit-layer;desc="REC") にキャッシュされました。
- ビューワーリクエスト () を受信した後、レスポンスの最初のバイトがビューワーに送信 CloudFront されるまでに 137 ミリ秒かかりましたcdn-downstream-fb1;dur=137。

が配信する CloudFront コンテンツの追加、削除、または置き換え

このセクションでは、ビューワーに提供するコンテンツに CloudFront がアクセスできることを確認する方法、ウェブサイトまたはアプリケーションでオブジェクトを指定する方法、コンテンツを削除または置き換える方法について説明します。

トピック

- [が CloudFront 配信するコンテンツの追加とアクセス](#)
- [CloudFront ディストリビューションを使用した既存のコンテンツの更新](#)
- [コンテンツを削除 CloudFront して配信しない](#)
- [のファイルの URL 形式のカスタマイズ CloudFront](#)
- [デフォルトのルートオブジェクトの指定](#)
- [ファイルの無効化](#)
- [圧縮ファイルの供給](#)
- [カスタムエラーレスポンスの生成](#)

が CloudFront 配信するコンテンツの追加とアクセス

コンテンツ (オブジェクト) CloudFront を配信する場合は、ディストリビューションに指定したオリジンの 1 つにファイルを追加し、ファイル CloudFront へのリンクを公開します。CloudFront エッジロケーションは、エッジロケーションがビューワーリクエストを受信するまで、オリジンから新しいファイルを取得しません。詳細については、「[がコンテンツを CloudFront 配信する方法](#)」を参照してください。

CloudFront 配信するファイルを追加するときは、ディストリビューションで指定された Amazon S3 バケットのいずれか、またはカスタムオリジンの場合は、指定されたドメインのディレクトリにそのファイルを追加してください。さらに、該当するキャッシュ動作のパスパターンが正しいオリジンにリクエストを送信していることを確認します。

たとえば、キャッシュ動作のパスパターンが *.html であるとします。そのオリジンにリクエストを転送するように他のキャッシュ動作を設定していない場合、転送されるのは *.html ファイル CloudFront のみです。例えば、このシナリオでは、.jpg ファイルを含むキャッシュ動作を作成していないため、オリジンにアップロードした .jpg ファイルは配布 CloudFront されません。

CloudFront サーバーは、提供するオブジェクトの MIME タイプを決定しません。ファイルをオリジンにアップロードする場合、ファイルの Content-Type ヘッダーフィールドを設定することをお勧めします。

CloudFront デイストリビューションを使用した既存のコンテンツの更新

配信するように設定された既存のコンテンツを更新するには、次の 2 CloudFront つの方法があります。

- 同じ名前を使用してファイルを更新する
- ファイル名にバージョン識別子を使用して更新する

CloudFront が提供するコンテンツの管理をより詳細に制御できるように、ファイル名またはフォルダ名にバージョン識別子を使用することをお勧めします。

バージョン付きのファイル名を使用した既存ファイルの更新

CloudFront デイストリビューション内の既存のファイルを更新する場合は、何らかのバージョン識別子をファイル名またはディレクトリ名に含めることをお勧めします。これにより、コンテンツをより適切に制御できます。この識別名には、日付タイムスタンプ、連番など、同じオブジェクトの 2 つのバージョンを区別する方法を使用できます。

たとえば、グラフィックファイルに image.jpg ではなく image_1.jpg という名前を付けることができます。ファイルの新しいバージョンを供給する場合は、新しいファイルに image_2.jpg という名前を付けて、image_2.jpg を指すようにお使いのウェブアプリケーションまたはウェブサイトのリンクを更新します。また、すべてのグラフィックを images_v1 ディレクトリに配置することもできます。1 つ以上のグラフィックの新しいバージョンを供給する場合は、新しい images_v2 ディレクトリを作成し、そのディレクトリを指すようにリンクを更新します。バージョンングでは、新しいバージョンのオブジェクトを提供し CloudFront 始める前にオブジェクトの有効期限が切れるのを待つ必要がなく、オブジェクトの無効化に対して料金を支払う必要もありません。

ファイルにバージョンを設定した場合も、有効期限切れ日付を設定することをお勧めします。詳細については、「[コンテンツがキャッシュに保持される期間 \(有効期限\) の管理](#)」を参照してください。

Note

バージョン付きのファイル名またはディレクトリ名の指定は、Amazon S3 オブジェクトのバージョンングとは関係がありません。

同じファイル名を使用した既存コンテンツの更新

デイス CloudFront トリビューション内の既存のファイルを更新し、同じファイル名を使用することはできますが、新しいファイルや更新されたファイルをオリジンに配置するのではなく、ファイルがリクエストされた場合にのみ、ファイルをエッジロケーションに CloudFront 配布することはお勧めしません。オリジン内の既存のファイルを同じ名前の新しいバージョンで更新した場合、以下の両方のイベントが発生するまで、エッジロケーションはオリジンから新しいバージョンを取得しません。

- キャッシュ内にあるファイルの旧バージョンが有効期限切れになった。詳細については、「[コンテンツがキャッシュに保持される期間 \(有効期限\) の管理](#)」を参照してください。
- エッジロケーションにファイルのユーザーリクエストが存在する。

ファイルの置き換え時に同じ名前を使用する場合、新しいファイルの提供 CloudFront を開始するタイミングを制御することはできません。デフォルトでは、エッジロケーションのファイルを 24 時間 CloudFront キャッシュします。(詳しくは、[コンテンツがキャッシュに保持される期間 \(有効期限\) の管理](#) を参照してください)。たとえば、ウェブサイト全体のすべてのファイルを置き換えると仮定します。

- 人気のないページのファイルは、どのエッジロケーションにも存在しない可能性があります。これらのファイルの新しいバージョンについては、次のリクエストで供給が開始されます。
- ページによっては、そのファイルが一部のエッジロケーションに存在し、他のエッジロケーションに存在しない場合があります。したがって、供給元のエッジロケーションによって、エンドユーザーに表示されるバージョンが異なります。
- 最も人気のあるページの新しいバージョンのファイルは、最大 24 時間提供されない場合があります。は、ファイルを新しいバージョンに置き換える直前にこれらのページのファイルを取得している CloudFront 可能性があるためです。

コンテンツを削除 CloudFront して配信しない

CloudFront デイストリビューションに含めたくないファイルをオリジンのファイルから削除することができます。ただし、ファイルの有効期限が切れるまで CloudFront、エッジキャッシュのコンテンツは引き続きビューワーに表示されます。

すぐにファイルを削除する場合は、次のいずれかを実行する必要があります。

- ファイルを無効化します。詳細については、「[ファイルの無効化](#)」を参照してください。
- ファイルバージョニングを使用します。バージョニングを使用する場合、デイス CloudFront トリビューションで使用できる名前は異なるバージョンになり、ビューワーに返されるファイルを変更します。詳細については、「[バージョン付きのファイル名を使用した既存ファイルの更新](#)」を参照してください。

のファイルの URL 形式のカスタマイズ CloudFront

ビューワー CloudFront に提供するオブジェクト (コンテンツ) を使用してオリジンを設定したら、が配信できるように CloudFront、正しい URLs を使用してウェブサイトまたはアプリケーションコードでそれらのオブジェクトを参照する必要があります。

ウェブページやウェブアプリケーションのオブジェクトの URL で使用するドメイン名には、次のいずれかを指定できます。

- デイストリビューションの作成時に CloudFront が自動的に割り当てる `d111111abcdef8.cloudfront.net`などのドメイン名
- `example.com` など、独自のドメイン名

たとえば、`image.jpg` というファイルを返すために、次の URL のいずれかを使用します。

```
https://d111111abcdef8.cloudfront.net/images/image.jpg
```

```
https://example.com/images/image.jpg
```

Amazon S3 バケット、またはカスタムオリジンのどちらにコンテンツを保存しても、独自のウェブサーバーと同様、同じ URL 形式を使用します。

Note

URL 形式は、ディストリビューションの [Origin Path] に指定した値によってある程度異なります。この値は CloudFront、オブジェクトの最上位ディレクトリパスを指定します。ウェブディストリビューションを作成する際のオリジンパスの設定の詳細については、「[オリジンのパス](#)」を参照してください。

URL 形式の詳細については、次のセクションを参照してください。

独自のドメイン名を使用する (example.com)

ディストリビューションの作成時に CloudFront 割り当てるデフォルトのドメイン名を使用する代わりに、など、より簡単に操作できる [代替ドメイン名を追加できます](#) example.com。で独自のドメイン名を設定することで CloudFront、ディストリビューション内のオブジェクトに次のような URL を使用できます。

```
https://example.com/images/image.jpg
```

ビューワーとの間で HTTPS を使用する予定がある場合は CloudFront、「」を参照してください [代替ドメイン名と HTTPS の使用](#)。

URL に末尾のスラッシュ (/) を使用する

CloudFront ディストリビューション内のディレクトリURLs を指定する場合は、常に末尾にスラッシュを使用するか、または末尾にスラッシュを使用しないかを選択します。たとえば、すべての URL に対して次のいずれか 1 つの形式のみを選択します。

```
https://d111111abcdef8.cloudfront.net/images/
```

```
https://d111111abcdef8.cloudfront.net/images
```

それが重要なのはなぜか。

どちらの形式も CloudFront オブジェクトへのリンクとして機能しますが、一貫性を保つことで、後でディレクトリを無効にする際の問題を防ぐことができます。CloudFront は、末尾のスラッシュを含め、定義されたとおりに URLs を保存します。したがって、形式に一貫性がない場合は、ディレクトリ CloudFront を削除するように、スラッシュの有無にかかわらずディレクトリ URLs を無効にする必要があります。

両方の URL 形式を無効にするのは不便で、追加コストが発生することがあります。これは、両方のタイプの URL をカバーするために無効化を二重に実行する場合、1 か月の無料の無効化回数の上限に達する可能性があるためです。そして、もしそのような事態になれば、CloudFront の各ディレクトリ URL にはただ 1 つの形式しか存在しない場合でも、すべての無効化に対して支払いをする必要があります。

制限されたコンテンツの署名付き URL の作成

アクセスを制限するコンテンツがある場合は、署名付き URL を作成できます。たとえば、認証されたユーザーのみにコンテンツを配信する場合は、指定された期間のみ、または指定された IP アドレスからのみ有効な URL を作成できます。詳細については、「[署名付き URL と署名付き Cookie を使用したプライベートコンテンツの提供](#)」を参照してください。

デフォルトのルートオブジェクトの指定

ディストリビューション内のオブジェクトをリクエストするのではなく、ユーザーがディストリビューションのルート URL をリクエストしたときに特定のオブジェクト (デフォルトのルートオブジェクト) を返す CloudFront ようにを設定できます。デフォルトのルートオブジェクトを指定すると、ディストリビューションのコンテンツが公開されなくなります。

トピック

- [デフォルトのルートオブジェクトを指定する方法](#)
- [デフォルトのルートオブジェクトの仕組み](#)
- [ルートオブジェクトを定義しない場合の CloudFront の動作](#)

デフォルトのルートオブジェクトを指定する方法

ディストリビューションのコンテンツが公開されたり、エラーが返されたりすることを回避するには、以下のステップを実行して、ディストリビューションのデフォルトのルートオブジェクトを指定します。

ディストリビューションのデフォルトルートオブジェクトを指定するには

1. デフォルトルートオブジェクトを、ディストリビューションが指しているオリジンにアップロードします。

ファイルには、CloudFront でサポートされるあらゆるタイプを使用できます。ファイル名に対する制約のリストについては、「」の DefaultRootObject 要素の説明を参照してください [DistributionConfig](#)。

Note

デフォルトのルートオブジェクトのファイル名が長すぎるか、無効な文字が含まれている場合、はエラー CloudFront を返します HTTP 400 Bad Request - InvalidDefaultRootObject。さらに、はコードを 10 秒間 CloudFront キャッシュし (デフォルトで)、その結果をアクセスログに書き込みます。

- オブジェクトのアクセス許可が CloudFront 少なくとも read アクセス権を付与していることを確認します。

Amazon S3 のアクセス権限の詳細については、Amazon Simple Storage Service ユーザーガイドの「[Amazon S3 での Identity and Access Management](#)」を参照してください。

- CloudFront コンソールまたは CloudFront API を使用して、デフォルトのルートオブジェクトを参照するようにディストリビューションを更新します。

CloudFront コンソールを使用してデフォルトのルートオブジェクトを指定するには：

- にサインイン AWS Management Console し、 で CloudFront コンソールを開きます <https://console.aws.amazon.com/cloudfront/v4/home>。
- 上部ペインにあるディストリビューションのリストで、更新するディストリビューションを選択します。
- [Settings] (設定) ペインで、[General] (一般) タブの [Edit] (編集) を選択します。
- [Edit settings] (設定の編集) ダイアログボックスの [Default root object] (デフォルトルートオブジェクト) フィールドに、デフォルトルートオブジェクトのファイル名を入力します。

オブジェクト名のみを入力します (例: index.html)。オブジェクト名の前に / を追加しないでください。

- [変更の保存] をクリックします。

CloudFront API を使用して設定を更新するには、ディストリビューションの DefaultRootObject 要素の値を指定します。CloudFront API を使用してデフォルトの

ルートオブジェクトを指定する方法については、「Amazon CloudFront API リファレンス」の [UpdateDistribution](#)「」を参照してください。

4. ルート URL を要求することで、デフォルトルートオブジェクトが有効になっていることを確認します。ブラウザにデフォルトルートオブジェクトが表示されない場合、以下のステップを実行します。
 - a. CloudFront コンソールでディストリビューションのステータスを表示して、ディストリビューションが完全にデプロイされていることを確認します。
 - b. ステップ 2 と 3 を繰り返し、適切な許可を付与したこと、およびデフォルトルートオブジェクトを指定するようにディストリビューションの構成を適切に更新したことを確認します。

デフォルトのルートオブジェクトの仕組み

次のリクエストはオブジェクト `image.jpg` を指すとします。

```
https://d111111abcdef8.cloudfront.net/image.jpg
```

これに対して、最初の例のように、次のリクエストは特定のオブジェクトではなく、同じディストリビューションのルート URL を指します。

```
https://d111111abcdef8.cloudfront.net/
```

デフォルトルートオブジェクトを定義した場合、ディストリビューションのルートを呼び出すエンドユーザーリクエストはデフォルトルートオブジェクトを返します。たとえば、ファイル `index.html` をデフォルトルートオブジェクトとして指定したと仮定します。

```
https://d111111abcdef8.cloudfront.net/
```

戻り値:

```
https://d111111abcdef8.cloudfront.net/index.html
```

Note

CloudFront は、末尾にスラッシュ (`https://d111111abcdef8.cloudfront.net///`) が複数ある URL が と等しいかどうかを判断しません `https://d111111abcdef8.cloudfront.net/`。オリジンサーバーがその比較を行います。

デフォルトルートオブジェクトを定義しても、ディストリビューションのサブディレクトリに対するエンドユーザーリクエストはデフォルトルートオブジェクトを返しません。例えば、`index.html`がデフォルトのルートオブジェクトであり、ディストリビューションの `install` ディレクトリ CloudFrontに対するエンドユーザーリクエスト CloudFront を受け取ったとします。

```
https://d1111111abcdef8.cloudfront.net/install/
```

CloudFront は、 のコピーが `install` ディレクトリ `index.html` に表示されていても、デフォルトのルートオブジェクトを返しません。

が CloudFront サポートするすべての HTTP メソッドを許可するようにディストリビューションを設定すると、デフォルトのルートオブジェクトがすべてのメソッドに適用されます。例えば、デフォルトのルートオブジェクトが `index.php` で、リクエストをドメインのルート (`https://example.com`) POST に送信するようにアプリケーションを作成する場合、CloudFront はリクエストを `https://example.com/index.php` に送信します。

CloudFront デフォルトのルートオブジェクトの動作は、Amazon S3 インデックスドキュメントの動作とは異なります。Amazon S3 バケットをウェブサイトとして設定し、インデックスドキュメントを指定した場合、ユーザーがバケット内のサブディレクトリを要求しても、Amazon S3 はインデックスドキュメントを返します。(インデックスドキュメントのコピーがすべてのサブディレクトリに含まれる必要があります)。Amazon S3 バケットをウェブサイトとして設定する方法とインデックスドキュメントの詳細については、Amazon Simple Storage Service ユーザーガイドの「[Amazon S3 でのウェブサイトのホスティング](#)」を参照してください。

Important

デフォルトのルートオブジェクトはデイス CloudFront トリビューションにのみ適用されます。オリジンのセキュリティを依然として管理する必要があります。例えば、Amazon S3 オリジンを使用する場合、Amazon S3 バケットの ACL も依然として適切に設定する必要があります。バケットに対する必要なアクセスレベルを確保する必要があります。

ルートオブジェクトを定義しない場合の CloudFront の動作

デフォルトルートオブジェクトを定義しない場合、ディストリビューションのルートの要求はオリジンサーバーに渡されます。Amazon S3 オリジンを使用する場合、以下のいずれかが返される場合があります。

- Amazon S3 バケットの内容のリスト — 次のいずれかの条件の下では、を使用してディストリビューション CloudFront にアクセスするすべてのユーザーに対し、オリジンの内容が表示されません。
 - バケットが適切に構成されていない。
 - ディストリビューションに関連付けられているバケット、およびバケット内のオブジェクトに対する Amazon S3 アクセス許可で、すべてのユーザーにアクセスが付与されている。
 - エンドユーザーがオリジンのルート URL を使用してオリジンにアクセスしている。
- オリジンのプライベートコンテンツのリスト – オリジンをプライベートディストリビューションとして設定する場合 (自分と CloudFront アクセス権のみ)、ディストリビューションに関連付けられた Amazon S3 バケットの内容は、を介してディストリビューションにアクセスするための認証情報を持つすべてのユーザーに表示されます CloudFront。この場合、ユーザーは、オリジンのルート URL を使用してコンテンツにアクセスできません。プライベートコンテンツの配信の詳細については、「[the section called “署名付き URL と署名付き Cookie を使用したコンテンツの制限”](#)」を参照してください。
- Error 403 Forbidden— CloudFront ディストリビューションに関連付けられた Amazon S3 バケットのアクセス許可、またはそのバケット内のオブジェクトのアクセス許可がすべてのユーザーへのアクセス CloudFront とすべてのユーザーへのアクセスを拒否した場合、このエラーを返します。

ファイルの無効化

有効期限が切れる前に CloudFront エッジキャッシュからファイルを削除する必要がある場合は、次のいずれかを実行できます。

- エッジキャッシュからファイルを無効にします。ビューワーが次にファイルをリクエストしたときに、CloudFront はオリジンに戻ってファイルの最新バージョンをフェッチします。
- ファイルのバージョニングを使用して、異なる名前を持つ異なるバージョンのファイルを供給します。詳しくは、「[バージョン付きのファイル名を使用した既存ファイルの更新](#)」を参照してください。

ファイルを無効にするには、個々のファイルのパスまたは * ワイルドカードで終わるパスのいずれかを指定します。これは次の例に示すように、1 つまたは複数のファイルに適用できます。

- /images/image1.jpg
- /images/image*

- `/images/*`

Note

ファイルの無効化に AWS Command Line Interface (AWS CLI) を使用し、* ワイルドカードが含まれるパスを指定する場合は、パスを引用符 (") で囲む必要があります。

例: `aws cloudfront create-invalidation --distribution-id distribution_ID --paths "/"`

月ごとに、一定の数の無効化パスを無料で送信できます。月ごとに割り当てられた数より多くの無効化パスを送信する場合、送信する無効化パスごとに料金が発生します。無効化に関する料金の詳細については、「[ファイルの無効化に対する支払い](#)」を参照してください。

トピック

- [ファイルを無効化するか、バージョン付きファイル名を使用するかの選択](#)
- [無効にするファイルの決定](#)
- [無効にするファイルの指定](#)
- [コンソールを使用したファイルの無効化](#)
- [CloudFront API を使用したファイルの無効化](#)
- [同時無効化リクエストの最大制限](#)
- [ファイルの無効化に対する支払い](#)

ファイルを無効化するか、バージョン付きファイル名を使用するかの選択

ディストリビューションから供給されるファイルのバージョンを制御するには、ファイルを無効にするか、バージョン付きファイル名をファイルに設定します。ファイルを頻繁に更新する必要がある場合は、以下の理由で、ファイルのバージョンングを第一に使用することをお勧めします。

- バージョニングを使用すると、ローカルにキャッシュされている、または企業のキャッシュプロキシの背後にキャッシュされているバージョンをユーザーが保持している場合でも、リクエストがどのファイルを返すかを制御できます。ファイルを無効にした場合、キャッシュ内でオブジェクトが有効期限切れになるまで、ユーザーに旧バージョンが引き続き表示されることがあります。
- CloudFront アクセスログにはファイルの名前が含まれているため、バージョンングを使用すると、ファイルの変更結果の分析が容易になります。

- バージョニングは、さまざまなバージョンのファイルをさまざまなユーザーに供給する方法を提供します。
- バージョニングによって、ファイルのリビジョン間のロールフォワードとロールバックが簡素化されます。
- バージョニングのほうが、コストが安くなります。ファイルの新しいバージョン CloudFront をエッジロケーションに転送するにはに料金を支払う必要がありますが、ファイルの無効化には料金を支払う必要はありません。

ファイルのバージョニングの詳細については、[バージョン付きのファイル名を使用した既存ファイルの更新](#) を参照してください。

無効にするファイルの決定

ディレクトリ内のすべてのファイルや、名前が同じ文字で始まるすべてのファイルなど、複数のファイルが無効にする場合は、無効化パスの末尾に * ワイルドカードを含めることができます。* ワイルドカードの使用の詳細については、「[Invalidation paths](#)」を参照してください。

選択されたファイルが無効にする必要があり、ユーザーがオリジンのすべてのファイルに必ずしもアクセスしない場合は、ビューワーが CloudFront からどのファイルを要求したかを確認し、そのファイルのみを無効にできます。ビューワーがリクエストしたファイルを確認するには、CloudFront アクセスログ記録を有効にします。アクセスログの詳細については、「[標準ログ \(アクセスログ\) の設定および使用](#)」を参照してください。

無効にするファイルの指定

無効にするファイルの指定時には、以下の事項に注意してください。

大文字と小文字の区別

無効化パスでは大文字と小文字が区別されます。そのため、`/images/image.jpg` と `/images/Image.jpg` は 2 つの異なるファイルを指定したことになります。

Lambda 関数を使用した URI の変更

デイス CloudFront トリビューションがビューワーリクエストイベントで Lambda 関数をトリガーし、関数がリクエストされたファイルの URI を変更する場合は、両方の URIs を無効にして CloudFront エッジキャッシュからファイルを削除することをお勧めします。

- ビューワーリクエストの URI
- 関数による変更後の URI

例えば、Lambda 関数が以下のファイルの URI を変更して、

```
https://d1111111abcdef8.cloudfront.net/index.html
```

言語ディレクトリを含む URI にしたとします。

```
https://d1111111abcdef8.cloudfront.net/en/index.html
```

ファイルを無効にするには、次のパスを指定する必要があります。

- /index.html
- /en/index.html

詳しくは、「[Invalidation paths](#)」を参照してください。

デフォルトのルートオブジェクト

デフォルトルートオブジェクト (ファイル) を無効にする場合、他のファイルのパスを指定する場合と同じ方法でパスを指定します。詳細については、「[デフォルトのルートオブジェクトの仕組み](#)」を参照してください。

Cookie の転送

Cookie CloudFront をオリジンに転送するようにを設定した場合、CloudFront エッジキャッシュにはファイルの複数のバージョンが含まれている可能性があります。ファイルを無効にすると、CloudFront は、関係付けられた Cookie に関係なく、そのファイルのキャッシュされたあらゆるバージョンを無効にします。一部のバージョンを選択して無効にすることも、関連付けられた Cookie に基づいてその他のバージョンを選択して無効にすることもできません。詳しくは、「[Cookie に基づくコンテンツのキャッシュ](#)」を参照してください。

ヘッダーの転送

ヘッダーのリストをオリジンに転送し CloudFront、ヘッダーの値に基づいてキャッシュするようにを設定した場合、CloudFront エッジキャッシュにはファイルの複数のバージョンが含まれている可能性があります。ファイルを無効にすると、はヘッダー値に関係なく、ファイルのキャッシュされたすべてのバージョンを CloudFront 無効にします。ヘッダー値に基づいて一部のバージョンのみ選択して無効にすることはできません (すべてのヘッダー CloudFront をオリジンに転送するようにを設定した場合、CloudFront はファイルをキャッシュしません)。詳細については、「[リクエストヘッダーに基づくコンテンツのキャッシュ](#)」を参照してください。

クエリ文字列の転送

クエリ文字列 CloudFront をオリジンに転送するようにを設定した場合は、次の例に示すように、ファイルを無効にするときにクエリ文字列を含める必要があります。

- /images/image.jpg?parameter1=a
- /images/image.jpg?parameter1=b

クライアントリクエストに、同じファイルに対する 5 つの異なるクエリ文字列が含まれる場合、クエリ文字列ごとに 1 回ずつ、5 回を無効にするか、次の例に示すように個別の無効化パスに * ワイルドカードを使用できます。

```
/images/image.jpg*
```

無効化パスでのワイルドカードの使用の詳細については、「[Invalidation paths](#)」を参照してください。クエリ文字列の詳細については、「[クエリ文字列パラメータに基づくコンテンツのキャッシュ](#)」を参照してください。どのクエリ文字列が使用されているかを判断するには、CloudFront ログ記録を有効にします。詳細については、「[標準ログ \(アクセスログ\) の設定および使用](#)」を参照してください。

最大許容数

許可される無効化の最大許容値については、「[同時無効化リクエストの最大制限](#)」を参照してください。

Microsoft Smooth Streaming ファイル

対応するキャッシュ動作に対してスムーズストリーミングを有効にした場合は、Microsoft Smooth Streaming 形式のメディアファイルは無効にすることはできません。

パス内の ASCII 以外の文字または安全ではない文字

パスに ASCII 以外の文字が含まれるか、[RFC 1738](#) に規定された安全ではない文字が含まれる場合、その文字を URL エンコードします。パス内の他の文字を URL エンコードしないでください。そう CloudFront しないと、更新されたファイルの古いバージョンが無効になることはありません。

無効化パス

パスはディストリビューションを基準とする相対パスです。たとえば、https://d111111abcdef8.cloudfront.net/images/image2.jpg のファイルは無効にするには、次のように指定します。

```
/images/image2.jpg
```

Note

[CloudFront コンソール](#)では、`images/image2.jpg`のようにパスの先頭のスラッシュを省略できます。CloudFront API を直接使用する場合、無効化パスは先頭にスラッシュを付ける必要があります。

* ワイルドカードを使用して、同時に複数のファイルを無効にすることもできます。0 個以上の文字を置き換える * は、無効化パスの最後の文字である必要があります。ファイルの無効化に AWS Command Line Interface (AWS CLI) を使用し、* ワイルドカードが含まれるパスを指定する場合は、パスを引用符 (") で囲む必要があります (例: `"/*`)。

次に例をいくつか示します。

- ディレクトリ内のすべてのファイルを無効にするには:

```
/directory-path/*
```

- ディレクトリ、そのすべてのサブディレクトリ、およびそのディレクトリとサブディレクトリのすべてのファイルを無効にするには:

```
/directory-path*
```

- 同じ名前前でファイル名拡張子が異なるすべてのファイル (`logo.jpg`、`logo.png`、`logo.gif` など) を無効にするには:

```
/directory-path/file-name.*
```

- ファイル名拡張子にかかわらず、ディレクトリ内でファイル名が同じ文字で始まるすべてのファイル (HLS 形式の動画のすべてのファイルなど) を無効にするには:

```
/#####/initial-characters-in-file-name*
```

- クエリ文字列パラメータに基づいてキャッシュ CloudFront するようにを設定し、ファイルのすべてのバージョンを無効にする場合:

```
/directory-path/file-name.file-name-extension*
```

- ディストリビューション内のすべてのファイルを無効にするには:

```
/*
```

パスの最大長は 4000 文字です。パス内にワイルドカードを使用することはできません。パスの末尾にのみ使用できます。

Lambda 関数を使用して URI を変更する場合のファイルの無効化の詳細については、「[Changing the URI Using a Lambda Function](#)」を参照してください。

無効化パスを送信する料金は、無効にするファイルの数に関係なく同じです。つまり、1つのファイル (/images/logo.jpg) であっても、ディストリビューションに関連付けられたすべてのファイル (/*) であっても同じです。詳細については、「[Amazon CloudFront 料金表](#)」を参照してください。

無効化パスがディレクトリであり、ディレクトリの指定方法 (末尾のスラッシュ (/) を付けるかどうか) を標準化していない場合、末尾のスラッシュを付けたディレクトリと付けないディレクトリの両方を無効にすることをお勧めします (例: /images および /images/)。

書名付き URL

署名付き URL を使用している場合は、URL の疑問符 (?) の前の部分のみを含めてファイルを無効にします。

コンソールを使用したファイルの無効化

CloudFront コンソールを使用して、無効化の作成と実行、以前に送信した無効化のリストの表示、個々の無効化に関する詳細情報の表示を行うことができます。また、既存の無効化をコピーしたり、ファイルパスのリストを編集したり、編集された無効化を実行したりもできます。無効化をリストから削除することはできません。

- [ファイルの無効化](#)
- [既存の無効化のコピー、編集、および再実行](#)
- [無効化のキャンセル](#)
- [無効化のリスト表示](#)
- [無効化に関する情報の表示](#)

ファイルの無効化

CloudFront コンソールを使用してファイルを無効にするには、次の手順を実行します。

ファイルを無効化するには

1. にサインインAWS Management Consoleし、 で CloudFront コンソールを開きます<https://console.aws.amazon.com/cloudfront/v4/home>。

2. 無効にするファイルのディストリビューションを選択します。
3. [Distribution Settings] を選択します。
4. [Invalidations] タブを選択します。
5. [Create Invalidation] を選択します。
6. 無効にするファイルについて、1 行ごとに 1 つの無効化パスを入力します。無効パスの指定の詳細については、「[無効にするファイルの指定](#)」を参照してください。

⚠ Important

ファイルパスを慎重に指定します。無効化リクエストは開始後にキャンセルすることはできません。

7. [Invalidate] を選択します。

既存の無効化のコピー、編集、および再実行

以前に作成した無効化をコピーし、無効化パスのリストを更新して、更新した無効化を実行することができます。既存の無効化をコピーし、無効化パスを更新して、更新した無効化を実行せずに保存することはできません。

⚠ Important

まだ進行中の無効化をコピーする場合は、無効化パスのリストを更新し、更新された無効化を実行しても、コピーした無効化は停止または削除 CloudFront されません。いずれかの無効化パスが元の とコピーに表示される場合、CloudFront はファイルを 2 回無効化しようとしています。両方の無効化は、その月の無料の無効化の最大数にカウントされます。無料で行うことができる無効化の最大数に既に達している場合は、各ファイルの両方の無効化に対して料金が発生します。詳しくは、「[同時無効化リクエストの最大制限](#)」を参照してください。

既存の無効化のコピー、編集、および再実行を行うには

1. にサインインAWS Management Consoleし、 で CloudFront コンソールを開きます<https://console.aws.amazon.com/cloudfront/v4/home>。
2. コピーする無効化が含まれるディストリビューションを選択します。
3. [Distribution Settings] を選択します。

4. [Invalidations] タブを選択します。
5. コピーする無効化を選択します。

コピーする無効化が不明な場合は、無効化を選択し、[Details] を選択すると、その無効化の詳細情報が表示されます。

6. [コピー] を選択します。
7. 必要に応じて、無効化パスのリストを更新します。
8. [Invalidate] を選択します。

無効化のキャンセル

に無効化リクエストを送信すると CloudFront、 は数秒以内にすべてのエッジロケーションにリクエストを CloudFront 転送し、各エッジロケーションは直ちに無効化の処理を開始します。そのため、無効化を送信後にキャンセルすることはできません。

無効化のリスト表示

CloudFront コンソールを使用して、ディストリビューションに対して作成して実行した過去 100 個の無効化のリストを表示できます。100 個を超える無効化のリストを取得する場合は、ListInvalidations API アクションを使用します。詳細については、「Amazon CloudFront API リファレンス」の [ListInvalidations](#) 「」を参照してください。

無効化のリストを表示するには

1. にサインインAWS Management Consoleし、 で CloudFront コンソールを開きます <https://console.aws.amazon.com/cloudfront/v4/home>。
2. 無効化リストの表示の対象となるディストリビューションを選択します。
3. [Distribution Settings] を選択します。
4. [Invalidations] タブを選択します。

Note

無効化をリストから削除することはできません。

無効化に関する情報の表示

ディストリビューション ID、無効化 ID、無効化のステータス、無効化が作成された日時、無効化パスの完全リストを含め、無効化に関する詳細情報を表示できます。

無効化に関する情報を表示するには

1. にサインインAWS Management Consoleし、 で CloudFront コンソールを開きます<https://console.aws.amazon.com/cloudfront/v4/home>。
2. 詳細情報の表示の対象となる無効化が含まれるディストリビューションを選択します。
3. [Distribution Settings] を選択します。
4. [Invalidations] タブを選択します。
5. 該当する無効化を選択します。
6. [Details] を選択します。

CloudFront API を使用したファイルの無効化

オブジェクトの無効化と CloudFront API を使用した無効化に関する情報の表示については、Amazon CloudFront API リファレンスの以下のトピックを参照してください。

- ファイルの無効化: [CreateInvalidation](#)
- 無効化のリストを取得する: [ListInvalidations](#)
- 特定の無効化に関する情報の取得: [GetInvalidation](#)

同時無効化リクエストの最大制限

ファイルを個別に無効にする場合は、進行中のディストリビューションごとに最大 3,000 個のファイルまで、一度に無効化リクエストを作成できます。これは、最大 3,000 個のファイルに対する 1 つの無効化リクエスト、1 つのファイルに対する最大 3,000 個のリクエスト、または 3,000 個のファイルを超えないその他の任意の組み合わせとすることができます。たとえば、それぞれ 100 個のファイルを無効にする 30 個の無効化リクエストを送信できます。30 個の無効化リクエストがすべてまだ実行中である限り、それ以上の無効化リクエストを送信することはできません。最大値を超えると、 はエラーメッセージ CloudFront を返します。

* ワイルドカードを使用している場合、最大 15 個の無効化パスのリクエストを一度に作成できます。また、進行中のディストリビューションごとに最大 3,000 個の個別のファイルを同時に作成す

ることができます。ワイルドカードの無効化リクエストの最大制限は、ファイルの個別の無効化の最大制限とは無関係です。

ファイルの無効化に対する支払い

1 か月に送信した無効化パスのうち、最初の 1,000 件は無料です。1 か月に 1,000 件を超えると、無効化パス 1 件ごとに支払いが発生します。無効なパスは 1 つのファイル (/images/logo.jpg など) に対して、または複数のファイル (/images/*) に対して発生する場合があります。* ワイルドカードを含むパスは、によって何千ものファイルが無効になった場合でも CloudFront、1 つのパスとしてカウントされます。

1 か月あたり 1,000 個の無料の無効化パスの上限は、1 つの AWS アカウントで作成するすべてのディストリビューションの無効化パスの合計数に対して適用されます。たとえば、AWS アカウント john@example.com を使用して 3 個のディストリビューションを作成し、ある月に、各ディストリビューションに 600 個の無効化パス (合計で 1,800 個の無効化パス) を送信しました。この場合、AWS は、その月に 800 個の無効化パスに対して料金を請求します。無効化料金の詳細については、[「Amazon CloudFront 料金表」](#)を参照してください。無効化パスの詳細については、[「Invalidation paths」](#)を参照してください。

圧縮ファイルの供給

ビューワー (ウェブブラウザまたはその他のクライアント) がサポートされているときに、を使用して特定のタイプのオブジェクト (ファイル) CloudFront を自動的に圧縮し、圧縮されたオブジェクトを提供できます。ビューワーが Accept-Encoding HTTP ヘッダーを含む圧縮オブジェクトのサポートの可否を示します。

CloudFront では、Gzip および Brotli 圧縮形式を使用してオブジェクトを圧縮できます。ビューワーが両方の形式をサポートし、両方に達したキャッシュサーバーに存在する場合、は Brotli を CloudFront 優先します。キャッシュサーバーに圧縮形式が 1 つだけ存在する場合、はそれ CloudFront を返します。

Note

ウェブブラウザ Chrome および Firefox では、HTTPS を使用してリクエストを送信する場合のみ、Brotli 圧縮がサポートされます。これらのブラウザでは、HTTP リクエストで Brotli がサポートされません。

リクエストされたオブジェクトを圧縮するとオブジェクトが小さくなるため、ダウンロード時間を短縮できます。場合によっては、元のサイズの 4 分の 1 未満になることがあります。特に JavaScript と CSS ファイルの場合、ダウンロードが高速になると、ユーザーによるウェブページのレンダリングが速くなります。さらに、CloudFront データ転送のコストは供給されたデータの総量に基づいているため、圧縮オブジェクトを供給する方が、圧縮されていないオブジェクトを供給するよりもコストが安くなる可能性があります。

一部のカスタムオリジンでは、オブジェクトを圧縮することもできます。オリジンは、圧縮 CloudFront されないオブジェクトを圧縮できる場合があります (「」を参照 [CloudFront が圧縮するファイルタイプ](#))。オリジンが圧縮オブジェクトを に返す場合 CloudFront、 は、オブジェクトが Content-Encoding ヘッダーの存在に基づいて圧縮されていること CloudFront を検出し、オブジェクトを再度圧縮しません。

オブジェクトを圧縮 CloudFront するための の設定

オブジェクトを圧縮 CloudFront するように を設定するには、次のすべてを実行して、圧縮オブジェクトを供給するキャッシュ動作を更新します。

1. [オブジェクトを自動的に圧縮する] 設定がはいになっていることを確認します。(AWS CloudFormation または CloudFront API では、 `Compress` を に設定します `true`。)
2. [キャッシュポリシー](#) を使用してキャッシュ設定を指定し、Gzip と Brotli の設定がどちらも有効になっていることを確認します。(AWS CloudFormation または API で CloudFront、 `EnableAcceptEncodingGzip` と `EnableAcceptEncodingBrotli` を に設定します) `true`。
3. キャッシュポリシーの TTL 値が 0 より大きい値に設定されていることを確認します。TTL 値をゼロに設定すると、キャッシュは無効になり、オブジェクトは圧縮 CloudFront されません。

キャッシュ動作を更新するには、次のいずれかのツールを使用できます。

- [CloudFront コンソール](#)
- [AWS CloudFormation](#)
- [AWS SDK とコマンドラインツール](#)

CloudFront 圧縮の仕組み

オブジェクトを圧縮 CloudFront するように を設定する場合 (前のセクションを参照)、その仕組みは次のとおりです。

1. ビューワーがオブジェクトを要求する ビューワーにより Accept-Encoding HTTP ヘッダーが リクエストに含まれます。ヘッダー値には gzip、br、またはその両方が含まれます。これは、ビューワーが圧縮オブジェクトをサポートすることを示します。ビューワーが Gzip と Brotli の両方をサポートしている場合、 は Brotli CloudFront を優先します。

Note

ウェブブラウザ Chrome および Firefox では、HTTPS を使用してリクエストを送信する場合のみ、Brotli 圧縮がサポートされます。これらのブラウザでは、HTTP リクエストで Brotli がサポートされません。

2. エッジロケーションで、 はリクエストされたオブジェクトの圧縮されたコピーのキャッシュ CloudFront をチェックします。
3. 圧縮オブジェクトがすでにキャッシュにある場合、 CloudFront はそれをビューワーに送信し、残りのステップをスキップします。

圧縮されたオブジェクトがキャッシュにない場合は、オリジンへのリクエストを CloudFront 転送します。

Note

オブジェクトの非圧縮コピーがすでにキャッシュにある場合、リクエストをオリジンに転送せずにビューワーに送信 CloudFront できます。例えば、 CloudFront [これは以前に圧縮をスキップ](#)した場合に発生する可能性があります。この場合、 は非圧縮オブジェクトを CloudFront キャッシュし、オブジェクトの有効期限が切れるか、削除されるか、または無効になるまで、引き続き提供します。

4. オリジンが HTTP レスポンスに Content-Encodingヘッダーがあることを示す圧縮オブジェクトを返す場合、 は圧縮オブジェクトをビューワー CloudFront に送信し、キャッシュに追加して、残りの step. CloudFront doesn はオブジェクトを再度圧縮しません。

オリジンが非圧縮オブジェクトを に返す場合 CloudFront (HTTP レスポンスにContent-Encodingヘッダーがない場合)、オブジェクトが圧縮可能かどうか CloudFront を決定します。がオブジェクトを圧縮できるかどうかを決定する方法 CloudFrontの詳細については、次のセクションを参照してください。

5. オブジェクトが圧縮可能な場合は、 が CloudFront 圧縮してビューワーに送信し、キャッシュに追加します。(まれに、 [圧縮をスキップ](#)して、圧縮されていないオブジェクトをビューワーに送信する CloudFront 場合があります)。

CloudFront 圧縮に関する注意事項

次のリストでは、がオブジェクトを CloudFront 圧縮するタイミングについて詳しく説明します。

リクエストで HTTP 1.0 が使用される

へのリクエストが HTTP 1.0 CloudFront を使用している場合、は Accept-Encoding ヘッダー CloudFront を削除し、レスポンス内のオブジェクトを圧縮しません。

Accept-Encoding リクエストヘッダー

Accept-Encoding ヘッダーがビューワーリクエストにない場合、または値 br として gzip またはが含まれていない場合、CloudFront はレスポンスでオブジェクトを圧縮しません。Accept-Encoding ヘッダーになどの追加値が含まれている場合 deflate、はリクエストをオリジンに転送する前にそれら CloudFront を削除します。

CloudFront が [オブジェクトを圧縮するように設定されている場合](#)、キャッシュキーとオリジンリクエストに Accept-Encoding ヘッダーが自動的に含まれます。

動的コンテンツ

CloudFront は、常に動的コンテンツを圧縮するとは限りません。動的コンテンツのレスポンスが圧縮される場合もあれば、圧縮されない場合もあります。

オブジェクトを圧縮 CloudFront するようにを設定すると、コンテンツはすでにキャッシュされています

CloudFront は、オリジンからオブジェクトを取得するときにオブジェクトを圧縮します。オブジェクトを圧縮 CloudFront するようにを設定する場合、エッジロケーションにすでにキャッシュされているオブジェクトは圧縮 CloudFront されません。さらに、キャッシュされたオブジェクトがエッジロケーションで期限切れになり、オブジェクトに対する別のリクエストをオリジン CloudFront に転送する場合、オリジンが CloudFront HTTP ステータスコード 304 を返すと、はオブジェクトを圧縮しません。つまり、エッジロケーションには既に最新バージョンのオブジェクトがあることを意味します。エッジロケーションにすでにキャッシュされているオブジェクト CloudFront を圧縮する場合は、それらのオブジェクトを無効にする必要があります。詳細については、「[ファイルの無効化](#)」を参照してください。

オブジェクトが圧縮されるようにオリジンがすでに設定されている

オブジェクトを圧縮 CloudFront するようにを設定し、オリジンもオブジェクトを圧縮する場合、オリジンには、オブジェクトがすでに圧縮されている CloudFront ことを示す Content-Encoding ヘッダーが含まれている必要があります。オリジンからのレスポンスに Content-

Encodingヘッダーが含まれている場合、ヘッダーの値に関係なく、CloudFront はオブジェクトを圧縮しません。レスポンスをビューワー CloudFront に送信し、オブジェクトをエッジロケーションにキャッシュします。

CloudFront 圧縮するファイルタイプ

が CloudFront 圧縮するファイルタイプの詳細なリストについては、「」を参照してください。
[CloudFront が圧縮するファイルタイプ](#)。

CloudFront 圧縮するオブジェクトのサイズ

CloudFront は、サイズが 1,000 バイトから 10,000,000 バイトまでのオブジェクトを圧縮します。

Content-Length ヘッダー

オリジンはレスポンスに Content-Lengthヘッダーを含める必要があります。この CloudFront ヘッダーは、を使用して、オブジェクトのサイズが の圧縮範囲内にあるかどうかを判断します CloudFront。Content-Length ヘッダーが見つからない場合、無効な値が含まれている場合、または が圧縮する CloudFrontサイズ範囲外の値が含まれている CloudFront 場合、オブジェクトは圧縮されません。

レスポンスの HTTP ステータスコード

CloudFront は、レスポンスの HTTP ステータスコードが 200、 、 403または の場合にのみオブジェクトを圧縮します404。

レスポンスに本文がない

オリジンからの HTTP レスポンスに本文がない場合、圧縮 CloudFront する はありません。

ETag ヘッダー

CloudFront オブジェクトを圧縮するときに、HTTP レスポンスの ETagヘッダーが変更されることがあります。詳細については、「[the section called “ETag ヘッダーの変換”](#)」を参照してください。

CloudFront 圧縮をスキップする

CloudFront は、ベストエフォートベースでオブジェクトを圧縮します。まれに、圧縮をスキップ CloudFront します。CloudFront は、ホスト容量など、さまざまな要因に基づいてこの決定を行います。がオブジェクトの圧縮を CloudFront スキップすると、圧縮されていないオブジェクトがキャッシュされ、オブジェクトの有効期限が切れるか、削除されるか、無効になるまで、ビューワーに引き続き提供されます。

CloudFront が圧縮するファイルタイプ

オブジェクトを圧縮 CloudFront するように を設定すると、 はContent-Typeレスポンスヘッダーに次のいずれかの値を持つオブジェクト CloudFront のみを圧縮します。

- application/dash+xml
- application/eot
- application/font
- application/font-sfnt
- application/javascript
- application/json
- application/opentype
- application/otf
- application/pdf
- application/pkcs7-mime
- application/protobuf
- application/rss+xml
- application/truetype
- application/ttf
- application/vnd.apple.mpegurl
- application/vnd.mapbox-vector-tile
- application/vnd.ms-fontobject
- application/wasm
- application/xhtml+xml
- application/xml
- application/x-font-opentype
- application/x-font-truetype
- application/x-font-ttf
- application/x-httpd-cgi
- application/x-javascript

- application/x-mpegurl
- application/x-opentype
- application/x-otf
- application/x-perl
- application/x-ttf
- font/eot
- font/opentype
- font/otf
- font/ttf
- image/svg+xml
- text/css
- text/csv
- text/html
- text/javascript
- text/js
- text/plain
- text/richtext
- text/tab-separated-values
- text/xml
- text/x-component
- text/x-java-source
- text/x-script
- vnd.apple.mpegurl

ETag ヘッダーの変換

オリジンからの非圧縮オブジェクトに有効で強力な HTTP ETag ヘッダーが含まれ、オブジェクトが CloudFront 圧縮されると、CloudFront は強い ETag ヘッダー値を弱いに変換し ETag、ビューワーに弱い ETag 値を返します。ビューワーは、弱い ETag 値を格納し、それを使用して If-None-Match

HTTP ヘッダーで条件付きリクエストを送信できます。これにより CloudFront、ビューワーとオリジンはオブジェクトの圧縮バージョンと非圧縮バージョンをセマンティックに同等として扱うことができるため、不要なデータ転送が減少します。

有効な強い ETag ヘッダー値は、二重引用符 (") で始まります。強力な ETag 値を弱い値に変換するために、 は強力な ETag 値の先頭 W/ に文字 CloudFront を追加します。

オリジンのオブジェクトに弱い ETag ヘッダー値 (文字 で始まる値 W/) が含まれている場合、この値は変更 CloudFront されず、オリジンから受け取ったとおりにビューワーに返されます。

オリジンからのオブジェクトに無効な ETag ヘッダー値が含まれている場合 (値が " または で始まらない W/ 場合)、 は ETag ヘッダー CloudFront を削除し、ETag レスポンスヘッダーなしでオブジェクトをビューワーに返します。

詳細については、MDN ウェブドキュメントの以下のページを参照してください。

- [ディレクティブ](#) (ETag HTTP ヘッダー)
- [弱い検証](#) (HTTP 条件付きリクエスト)
- [If-None-Match HTTP ヘッダー](#)

カスタムエラーレスポンスの生成

使用しているオブジェクトが何らかの理由で使用できない場合、ウェブサーバー CloudFront は通常、関連する HTTP ステータスコードを に返 CloudFront してこれを示します。例えば、ビューワーが無効な URL をリクエストした場合、ウェブサーバーは HTTP 404 (Not Found) ステータスコードを に返し CloudFront、そのステータスコードをビューワーに CloudFront 返します。

必要に応じて、代わりにビューワーにカスタムエラーレスポンスを返す CloudFront ように を設定できます。また、エラーが発生したときの の CloudFront 応答を管理するためのオプションがいくつかあります。カスタムエラーメッセージのオプションを指定するには、デイス CloudFront トリビューションを更新してそれらの値を指定します。詳細については、「[エラーレスポンス動作を設定する](#)」を参照してください。

HTTP ステータスコードのカスタムエラーページを返 CloudFront すように を設定しても、カスタムエラーページが使用できない場合、 はカスタムエラーページを含むオリジンから CloudFront 受信したステータスコードをビューワーに CloudFront 返します。例えば、カスタムオリジンが 500 ステータスコードを返し、Amazon S3 バケットから 500 ステータスコードのカスタムエラーページを取得する CloudFront ように を設定したとします。ただし、誰かが誤ってバケットからカスタムエ

ラーページを削除しました。は、オブジェクトをリクエストしたビューワーに HTTP 404 ステータスコード (Not Found) CloudFront を返します。

がカスタムエラーページをビューワーに CloudFront 返す場合、リクエストされたオブジェクトの CloudFront 料金ではなく、カスタムエラーページの標準料金を支払います。CloudFront 料金の詳細については、[「Amazon CloudFront 料金表」](#)を参照してください。

トピック

- [エラーレスポンス動作を設定する](#)
- [特定の HTTP ステータスコードに対応するカスタムエラーページの作成](#)
- [オブジェクトとカスタムエラーページを別の場所に格納する](#)
- [によって返されるレスポンスコードの変更 CloudFront](#)
- [がエラーを CloudFront キャッシュする時間の制御](#)

エラーレスポンス動作を設定する

カスタムエラーレスポンスを設定するには、CloudFront コンソール、CloudFront API、または を使用しますAWS CloudFormation。これらの内どれにより設定を更新する場合でも、次に挙げるヒントと推奨事項を参考にしてください。

- カスタムエラーページは、 がアクセスできる場所に保存します CloudFront。これらのページの保存先は、Amazon S3 バケットにすることを推奨します。また、[ウェブサイトやアプリケーションなど、他のコンテンツと同じ場所には保存しないようにしてください](#)。ウェブサイトまたはアプリケーションと同じオリジンにカスタムエラーページを保存し、オリジンが 5xx エラーを返し始めた場合、オリジンサーバーが使用できないため、カスタムエラーページを取得 CloudFront できません。詳細については、「[オブジェクトとカスタムエラーページを別の場所に格納する](#)」を参照してください。
- にカスタムエラーページを取得するアクセス許可 CloudFront があることを確認します。カスタムエラーページが Amazon S3 に保存されている場合、そのページはパブリックにアクセス可能であるか、CloudFront [オリジンアクセスコントロール \(OAC\)](#) を設定する必要があります。カスタムエラーページをカスタムオリジンに格納する場合には、そのページはパブリックにアクセス可能である必要があります。
- (オプション) 必要に応じてオリジンを設定し、カスタムエラーページに Cache-Control または Expires ヘッダーを追加します。エラーキャッシュ最小 TTL 設定を使用して、 がカスタムエラーページを CloudFront キャッシュする期間を制御することもできます。詳細については、「[がエラーを CloudFront キャッシュする時間の制御](#)」を参照してください。

カスタムエラーレスポンスを設定する (CloudFront コンソール)

CloudFront コンソールでカスタムエラーレスポンスを設定するには、ディストリビューションが必要です。CloudFront コンソールからカスタムエラーレスポンスの構成を設定する際には、ディストリビューションが既に用意されている必要があります。ディストリビューションの作成方法については、「[シンプルな CloudFront ディストリビューションの開始方法](#)」を参照してください。

カスタムエラーレスポンスを設定するには (コンソール)

1. にサインインAWS Management Consoleし、の CloudFront コンソールでディストリビューションページを開きます<https://console.aws.amazon.com/cloudfront/v4/home#distributions>。
2. ディストリビューションの一覧で、更新するディストリビューションを選択します。
3. [エラーページ] タブを開き、[カスタムエラーレスポンスの作成] をクリックします。
4. 適切な値を入力します。詳細については、「[カスタムエラーページとエラーキャッシュ](#)」を参照してください
5. 必要な値を入力したら、[作成] をクリックします。

カスタムエラーレスポンスを設定する (CloudFront API または AWS CloudFormation)

CloudFront API または を使用してカスタムエラーレスポンスを設定するにはAWS CloudFormation、ディストリビューションで CustomErrorResponse タイプを使用します。詳細については、次を参照してください。

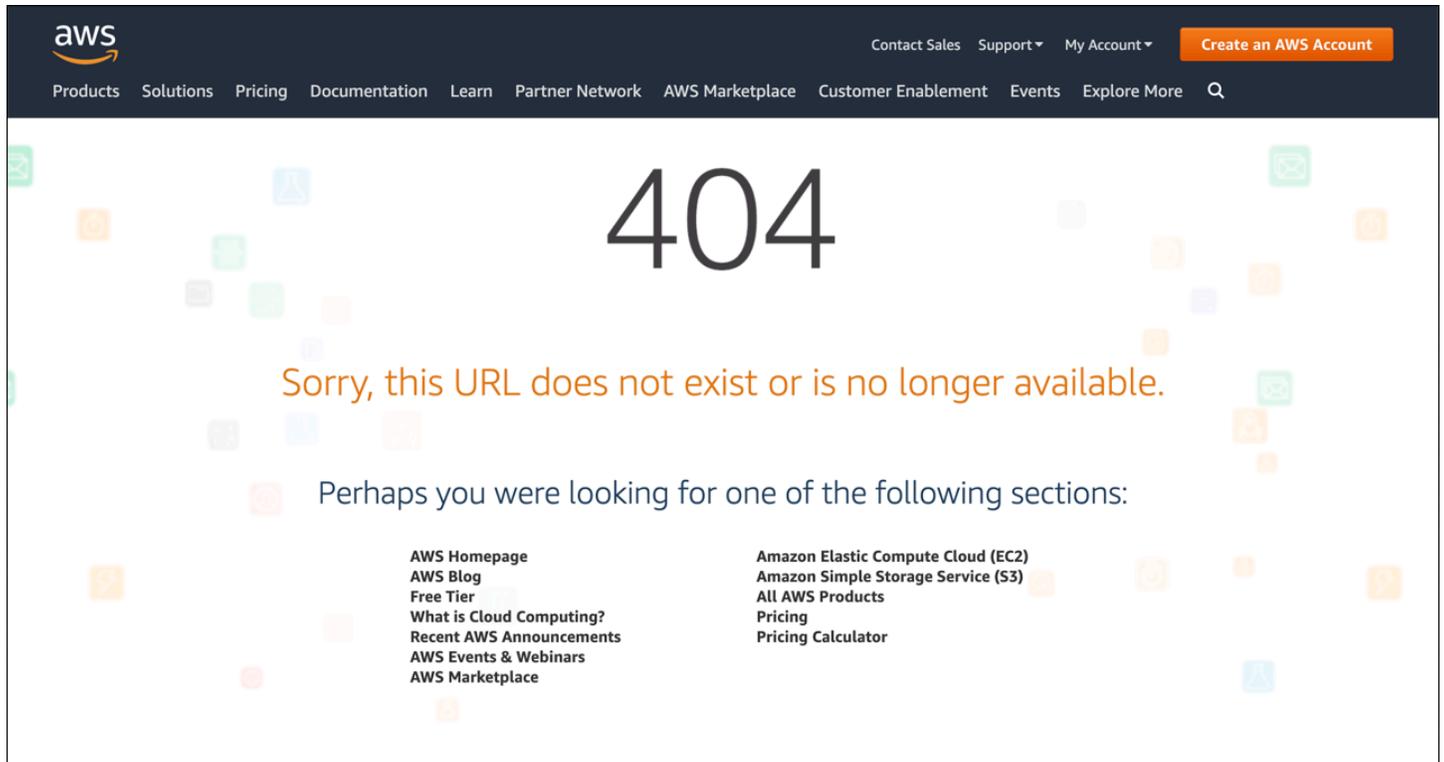
- 「AWS CloudFormation ユーザーガイド」の「[AWS::CloudFront::Distribution CustomErrorResponse](#)」
- [CustomErrorResponse](#) 「Amazon CloudFront API リファレンス」の「」

特定の HTTP ステータスコードに対応するカスタムエラーページの作成

デフォルトメッセージの代わりにカスタムエラーメッセージを表示する場合、例えば、ウェブサイトの他のページと同じ形式を使用するページなど、カスタムエラーメッセージを含むオブジェクト (HTML ファイルなど) をビューワー CloudFront に戻すことができます。

返すファイルと返すエラーを指定するには、ディストリビューションを更新してそれらの値を指定します。詳細については、「[エラーレスポンス動作を設定する](#)」を参照してください。

例として、カスタムエラーページは次のようなものになります。



サポートされている HTTP ステータスコードごとに異なるオブジェクトを指定することができます。または、サポートされているすべてのステータスコードに同じオブジェクトを使用することもできます。一部のステータスコードにカスタムエラーページを指定し、他のステータスコードには指定しないようにもできます。

使用しているオブジェクトは、さまざまな理由で使用できない CloudFront 場合があります。理由は、大きく 2 つに分類できます。以下にそれぞれを説明します。

- クライアントエラーは、リクエストに問題があることを示します。たとえば、指定した名前のオブジェクトが使用不能である場合や、Amazon S3 バケット内のオブジェクトを取得するために必要な権限がユーザーにない場合などです。クライアントエラーが発生すると、オリジンは 4xx 範囲の HTTP ステータスコードを に返します CloudFront。
- サーバーエラーの場合は、オリジンサーバーに問題があります。たとえば、HTTP サーバーが混雑していたり使用不能であったりする場合です。サーバーエラーが発生すると、オリジンサーバーは 5xx 範囲の HTTP ステータスコードを に返すか CloudFront、一定期間オリジンサーバーからレスポンスが届 CloudFront かず、504 ステータスコード (Gateway Timeout) を引き受けます。

がカスタムエラーページを返 CloudFront すことができる HTTP ステータスコードは次のとおりです。

- 400、403、404、405、414、416

Note

リクエストされた範囲は不適格であることを示す HTTP ステータスコード 416 (Requested Range Not Satisfiable) のカスタムエラーページを作成したり、オリジンが CloudFront にステータスコード 416 を返したときに CloudFront がビューワーに返す HTTP ステータスコードを変更したりできます (詳細については、[によって返されるレスポンスコードの変更 CloudFront](#) を参照してください)。ただし、ステータスコード 416 のレスポンスはキャッシュ CloudFront されないため、ステータスコード 416 のエラーキャッシュ最小 TTL の値を指定しても、CloudFront はそれを使用しません。

- 500、501、502、503、504

Note

場合によっては、HTTP 503 ステータスコードのカスタムエラーページを返 CloudFront さない CloudFront ように を設定しても、 が返さないことがあります。CloudFront エラーコードが Capacity Exceeded または の場合 Limit Exceeded、 はカスタムエラーページを使用せずに 503 ステータスコードをビューワーに CloudFront 返します。

がオリジンからのエラーレスポンス CloudFront を処理する方法の詳細については、「」を参照してください [がオリジンからの HTTP 4xx および 5xx ステータスコード CloudFront を処理してキャッシュする方法](#)。

オブジェクトとカスタムエラーページを別の場所に格納する

オブジェクトとカスタムエラーページを別の場所に保存する場合は、次の状況に該当するときに適用されるキャッシュ動作をディストリビューションに組み込む必要があります。

- [Path Pattern (パスパターン)] の値が、カスタムエラーメッセージのパスと一致している。たとえば、4xx エラーのカスタムエラーページを /4xx-errors というディレクトリの Amazon S3 バケットに保存したとします。このとき、パスパターンによってカスタムエラーページのリクエストがルーティングされる場所のキャッシュ動作を、ディストリビューションに組み込む必要があります (/4xx-errors/* など)。
- [Origin (オリジン)] の値は、カスタムエラーページが含まれているオリジンの [Origin ID (オリジン ID)] の値を指定しています。

詳細については、「[キャッシュ動作の設定](#)」を参照してください。

によって返されるレスポンスコードの変更 CloudFront

オリジンから受信した HTTP ステータスコードとは異なる HTTP ステータスコードをビューワーに返す CloudFront CloudFrontようにを設定できます。例えば、オリジンが 500 ステータスコードを返す場合 CloudFront、カスタムエラーページと 200 ステータスコード (OK) をビューワーに CloudFront 返すことができます。オリジンが返したステータスコードとは異なるステータスコードをビューワーに CloudFront 返す理由はさまざまです CloudFront。

- インターネットデバイス (一部のファイアウォールやコーポレートプロキシなど) の中には、HTTP 400 番台と 500 番台のステータスコードを遮断して、このレスポンスがビューワーに返信されないようするものがあります。このシナリオの場合、200 に置換することで、応答は遮断されなくなります。
- 異なるクライアントエラーやサーバーエラーの区別が重要でない場合は、`4xx` 400 または `5xx` ステータスコードに対して返す CloudFront 値 `500` として または `200` を指定できます。
- 200 ステータスコード (OK) と静的ウェブサイトを返すことにより、ウェブサイトが停止していることをユーザーが気づかないようにもできます。

[CloudFront 標準ログ](#)を有効にし、レスポンスの HTTP ステータスコードを変更するようにを設定 CloudFrontした場合、ログの `sc-status` 列の値には、指定したステータスコードが含まれます。ただし、`x-edge-result-type` 列の値は影響を受けません。この列には、オリジンから返された結果タイプが記述されます。例えば、オリジンが 404 (Not Found) を 200 に返すときに、ビューワー CloudFront にのステータスコードを返すようにを設定するとします CloudFront。オリジンが 404 ステータスコードでリクエストに応答すると、ログ内の `sc-status` 列の値は 200 になりますが、`x-edge-result-type` 列の値は `Error` になります。

カスタムエラーページとともに次の HTTP ステータスコードを返 CloudFront するようにを設定できます。

- 200
- 400、403、404、405、414、416
- 500、501、502、503、504

がエラーを CloudFront キャッシュする時間の制御

CloudFront は、デフォルトの期間である 10 秒間、エラーレスポンスをキャッシュします。CloudFront その後、オブジェクトに対する次のリクエストをオリジンに送信し、エラーの原因となった問題が解決され、リクエストされたオブジェクトが使用可能かどうかを確認します。

キャッシュする 4xx および 5xx ステータスコードごとに、エラーキャッシュ期間であるエラーキャッシュ最小 TTL を指定できます。CloudFront (詳細については、[キャッシュする CloudFront HTTP 4xx および 5xx ステータスコード](#) を参照してください)。期間を指定する場合は、以下の点に注意してください。

- 短いエラーキャッシュ期間を指定すると、期間を長く指定するよりもオリジンへのリクエストが多く CloudFront なります。この構成で 5xx エラーが発生すると、エラーの返信処理のために、オリジンで障害を起こした問題が悪化する可能性があります。
- オリジンがオブジェクトのエラーを返すと、 は、エラーキャッシュ期間が経過するまで、エラーレスポンスまたはカスタムエラーページを使用してオブジェクトのリクエスト CloudFront に対応します。エラーキャッシュ期間を長く指定すると、オブジェクトが再び利用可能になってから、 はリクエストにエラーレスポンスまたはカスタムエラーページで長期間応答し続ける CloudFront 可能性があります。

Note

リクエストされた範囲は不適格であることを示す HTTP ステータスコード 416 (Requested Range Not Satisfiable) のカスタムエラーページを作成したり、オリジンが CloudFront にステータスコード 416 を返したときに CloudFront がビューワーに返す HTTP ステータスコードを変更したりできます (詳細については、[によって返されるレスポンスコードの変更 CloudFront](#) を参照してください)。ただし、ステータスコード 416 CloudFront のレスポンスはキャッシュされないため、ステータスコード 416 のエラーキャッシュ最小 TTL の値を指定しても、CloudFront はそれを使用しません。

が個々のオブジェクトのエラーを CloudFront キャッシュする期間を制御する場合は、そのオブジェクトのエラーレスポンスに適切なヘッダーを追加するようにオリジンサーバーを設定できます。

オリジンが Cache-Control: max-age または Cache-Control: s-maxage デイレクティブ、または Expires ヘッダーを追加すると、 はヘッダーまたはエラー CloudFront キャッシュ最小 TTL の値のうち大きい方の値に対するエラーレスポンスをキャッシュします。

Note

Cache-Control: max-age および Cache-Control: s-maxage の値は、エラーページをフェッチするキャッシュ動作に設定されている [Maximum TTL] (最大 TTL) の値以下にする必要があります。

オリジンが他のCache-Controlディレクティブを追加するか、ヘッダーを追加しない場合、はエラーキャッシュ最小 TTL の値のエラーレスポンスをCloudFront キャッシュします。

オブジェクトに対する 4xx または 5xx ステータスコードの有効期限が、設定した待機時間よりも長く、さらにオブジェクトが使用可能状態に復帰した場合は、リクエストされたオブジェクトの URL を使ってそのステータスコードを無効化できます。オリジンが複数のオブジェクトに対してエラーレスポンスを返している場合は、各オブジェクトについて個別に無効化する必要があります。オブジェクトの無効化については、「[ファイルの無効化](#)」を参照してください。

AWS WAF 保護の使用

[AWS WAF](#) を使用してディストリビューションとオリジンサーバーを保護できます。AWS WAFは、ウェブアプリケーションと APIs を保護することで、ウェブアプリケーションと API を保護するのに役立つウェブアプリケーションファイアウォールです。

AWS WAF 保護を有効にするには、次のことができます。

- CloudFront コンソールでワンクリック保護を使用します。ワンクリック保護は、AWS WAF ウェブアクセスコントロールリスト (ウェブ ACL) を作成し、一般的なウェブ脅威からサーバーを保護するルールを設定し、ウェブ ACL を CloudFront ディストリビューションにアタッチします。このセクションのトピックでは、ワンクリック保護の使用を前提としています。
- AWS WAF コンソールまたは AWS WAF API を使用して作成した事前設定済みのウェブ ACL (アクセス制御リスト) を使用します。詳細については、「AWS WAF デベロッパーガイド」の [ACLs](#) および AWS WAF 「API リファレンス」の [AssociateWebACL](#) を参照してください。

以下の場合に AWS WAF を有効にすることができます。

- ディストリビューションを作成する
- [セキュリティ] ダッシュボードを使用して、既存のディストリビューションのセキュリティ設定を編集します。

ワンクリック保護を使用すると、AWS が推奨する一連の保護 CloudFront を適用します。

- Amazon の内部脅威インテリジェンスに基づく潜在的な脅威から IP アドレスをブロックします。
- [OWASP Top 10](#) で説明されているように、ウェブアプリケーションに見られる最も一般的な脆弱性から保護します。
- 悪意のある攻撃者がアプリケーションの脆弱性を発見するのを防ぎます。

Important

セキュリティダッシュボードで CloudFront セキュリティメトリクスを表示する AWS WAF 場合は、有効にする必要があります。AWS WAF を有効にしないと、セキュリティダッシュボードを使用して地理的 CloudFront 制限を有効化 AWS WAF または設定できません。ダッシュ

ボードの詳細については、このセクションの後半にある「[CloudFront セキュリティダッシュボードの使用](#)」を参照してください。

トピック

- [新しいディストリビューションで AWS WAF を有効にする](#)
- [既存のディストリビューションで AWS WAF を有効にする](#)
- [セキュリティ保護を無効にする](#)
- [レート制限の設定](#)
- [CloudFront セキュリティダッシュボードの使用](#)

新しいディストリビューションで AWS WAF を有効にする

以下の手順では、ディストリビューションの作成時に AWS WAF を有効にする方法と、新しいディストリビューションで既存の ACL を使用する方法について説明します。

新しいディストリビューションで AWS WAF を有効にするには

1. で CloudFront コンソールを開きます<https://console.aws.amazon.com/cloudfront/v4/home>。
2. ナビゲーションペインで [ディストリビューション] を選択し、[ディストリビューションを作成] を選択します。
3. 必要に応じて、「[ディストリビューションの作成](#)」の手順に従います。
4. [ウェブアプリケーションファイアウォール] セクションで、[編集]、[セキュリティ保護を有効にする] の順に選択します。追加のフィールドが表示されます。
5. 以下のフィールドに値を入力します。
 - モニタリングモードを使う: 保護の仕組みをテストするために最初にデータを収集する場合は、モニタリングモードを有効にします。モニタリングモードを有効にすると、保護が有効になっていてもリクエストはブロックされません。代わりに、モニタリングモードは、保護が有効な場合にブロックされるリクエストに関するデータを収集します。ブロックを開始する準備ができたなら、[セキュリティ] ページでブロックを有効にすることができます。
 - [その他の保護] セクションが表示される場合があります。有効にするオプションをすべて選択します。レート制限を有効にする場合の詳細については、「[the section called “レート制限の設定”](#)」を参照してください。

- [価格見積もり] セクション。セクションを開くと、1 か月あたりの異なるリクエスト数を入力するフィールドが表示され、新しい見積もりが表示されます。
6. 残りのディストリビューション設定を確認し、[ディストリビューションを作成する] または [設定を保存] (既存のディストリビューションを編集する場合) を選択します。

既存のウェブ ACL の使用

ウェブ ACL がある場合は、ワンクリック WAF が提供する保護の代わりに、そのウェブ ACL を使用できます。

既存の AWS WAF 設定を使用するには

1. で CloudFront コンソールを開きます <https://console.aws.amazon.com/cloudfront/v4/home>。
2. 次のいずれかを行います。
 - a. [ディストリビューションを作成] を選択し、「[ディストリビューションの作成](#)」の手順を実行して、このトピックに戻ります。
 - b. 既存の設定を選択し、[セキュリティ] タブを選択します。
3. [ウェブアプリケーションファイアウォール (WAF)] セクションで、[編集]、[セキュリティ保護を有効にする] の順に選択します。
4. [既存の WAF 設定を使用] を選択します。このオプションは、ウェブ ACL が設定されている場合にのみ表示されます。
5. [ウェブ ACL を選択] テーブルから既存のウェブ ACL を選択します。
6. 残りのディストリビューション設定を確認し、[ディストリビューションを作成] または [設定を保存] (既存のディストリビューションを編集する場合) を選択します。

既存のディストリビューションで AWS WAF を有効にする

CloudFront ディストリビューションを作成すると、によってセキュリティダッシュボードが作成されます。ディストリビューションを作成したら、ダッシュボードを使用して AWS WAF を有効にします。ダッシュボードのチャートとグラフは、AWS WAF を有効にするまで空白のままです。

1. で CloudFront コンソールを開きます <https://console.aws.amazon.com/cloudfront/v4/home>。
2. ナビゲーションペインで、[ディストリビューション] を選択し、変更するディストリビューションを選択します。

3. [Security] (セキュリティ) タブを選択します。
4. [ウェブアプリケーションファイアウォール] で、[編集]、[セキュリティ保護を有効にする] の順に選択します。
5. (オプション) [モニタリングモードを使う] を選択します。
6. [変更の保存] をクリックします。

モニタリングモードの詳細については、前のセクション「[新しいディストリビューションで AWS WAF を有効にする](#)」を参照してください。

既存のウェブ ACL の使用

ウェブ ACL を設定済みである場合、ワンクリック WAF が提供する保護の代わりに、これらのウェブ ACL を使用できます。

既存の AWS WAF 設定を使用するには

1. で CloudFront コンソールを開きます <https://console.aws.amazon.com/cloudfront/v4/home>。
2. 次のいずれかを行います。
 - a. [ディストリビューションを作成] を選択し、「[ディストリビューションの作成](#)」の手順を実行して、このトピックに戻ります。
 - b. [セキュリティ] を選択して既存のディストリビューションを設定します。
3. [ウェブアプリケーションファイアウォール (WAF)] セクションで、[セキュリティ保護を有効にする] を選択します。
4. [既存の WAF 設定を使用] を選択します。このオプションは、ウェブ ACL が設定されている場合にのみ表示されます。
5. [ウェブ ACL を選択] テーブルから既存のウェブ ACL を選択します。
6. 残りのディストリビューション設定を確認し、[ディストリビューションを作成] または [設定を保存] (既存のディストリビューションを編集する場合) を選択します。

セキュリティ保護を無効にする

でセキュリティ保護を無効にするには CloudFront

1. で CloudFront コンソールを開きます <https://console.aws.amazon.com/cloudfront/v4/home>。

2. ナビゲーションペインで、[ディストリビューション] を選択し、変更するディストリビューションを選択します。
3. [編集]、[セキュリティ] タブの順に選択します。
4. [ウェブアプリケーションファイアウォール] セクションで、[セキュリティ保護を有効にしないでください] を選択します。
5. [変更を保存] を選択します。

レート制限の設定

レート制限は、セキュリティ保護を設定する際に受け取る可能性のある推奨事項の 1 つです。

CloudFront では、常にモニタリングモードでレート制限を有効にします。モニタリングモードを有効にすると、はレート制限フィールドで設定したレートを超えたかどうか、頻度、および金額を示すメトリクスを CloudFront キャプチャします。

ディストリビューションを保存すると、はレート制限フィールドの数値に基づいてデータの収集 CloudFront を開始します。

レート制限の設定は、任意の CloudFront ディストリビューションのセキュリティタブの「セキュリティ - ウェブアプリケーションファイアウォール (WAF)」セクションで管理できます。[モニタリングモード] のメッセージを選択すると、収集されたデータに関する詳細を示すダイアログが表示されます。このダイアログでは、必要に応じてレート制限を変更できます。レートを微調整したら、ダイアログの [ブロックを有効にする] を選択してモニタリングモードを無効にできます。CloudFront は、指定されたレート制限を超えるリクエストのブロックを開始します。

CloudFront セキュリティダッシュボードの使用

CloudFront は、ディストリビューションごとにセキュリティダッシュボードを作成します。ダッシュボードは CloudFront コンソールで使用します。ダッシュボードを使用すると、CloudFront とを 1 か所で AWS WAF 一緒に使用して、ウェブアプリケーションの一般的なセキュリティ保護をモニタリングおよび管理できます。ダッシュボードは、以下のタスクとデータを提供します。

- セキュリティ設定：AWS WAF 保護を有効または無効にしたり、保護などのアプリ固有の WordPress 保護を確認したりできます。
- セキュリティトレンド: 許可およびブロックされたリクエスト、チャレンジおよび CAPTCHA リクエスト、上位の攻撃タイプ含まれます。

- **ボットリクエスト:** ボットからのトラフィック量、ボットの種類 (検証済みと未検証)、ボットの種類別の割り当て率 (検証済みと未検証) が時間の経過に伴ってどのように変化するかを確認できます。
- **リクエストログ:** ログデータは、セキュリティの傾向やボットのリクエストに関する質問に答えるのに役立ちます。クエリを作成しなくてもログを検索し、集計グラフを表示できるため、フィルタリングされた一連のログが主に HTTP メソッド、IP アドレス、URI パス、または国のサブセットによって処理されているかどうかを判断できます。グラフの値にカーソルを合わせると、IP アドレスや国をブロックできます。
- **地理的制限の管理**

Note

CloudFront セキュリティダッシュボードでセキュリティメトリクスを表示するAWS WAF場合は、[有効にする必要があります](#)。AWS WAF 有効にしないと、セキュリティダッシュボードを使用して地理的 CloudFront 制限を有効化AWS WAFまたは設定できます。

以下のセクションでは、ダッシュボードの使用方法について説明します。

トピック

- [AWS WAF の有効化](#)
- [トレンドデータの理解](#)
- [Bot Control の有効化](#)
- [ログの理解](#)
- [CloudFront 地理的制限の管理](#)
- [セキュリティダッシュボードの料金](#)

AWS WAF の有効化

セキュリティダッシュボードの上部にあるセクションを使用して、AWS WAF保護を有効または無効にします。には、設定に応じて、ディストリビューションに固有のセキュリティレコメンデーション CloudFront も表示されます。例えば、WordPress パスパターンを使用してキャッシュ動作を設定すると、WordPress 保護とレート制限が表示されます。

Note

CloudFront セキュリティダッシュボードでセキュリティメトリクスを表示するAWS WAF場合は、を有効にする必要があります。AWS WAF 有効にしないと、セキュリティダッシュボードを使用して地理的 CloudFront 制限を有効化AWS WAFまたは設定できます。

AWS WAF を有効にするには

1. で CloudFront コンソールを開きます<https://console.aws.amazon.com/cloudfront/v4/home>。
2. ナビゲーションペインで、[ディストリビューション] を選択し、変更するディストリビューションを選択します。
3. [Security] (セキュリティ) タブを選択します。
4. [ウェブアプリケーションファイアウォール] セクションで、[セキュリティ保護を有効にする] を選択します。
5. [変更を保存] を選択します。

AWS WAF 保護を無効にするには

- 上記の手順を繰り返します。ただし、[セキュリティ保護を無効にする] を選択します。

トレンドデータの理解

ダッシュボードの [指定した時間範囲のセキュリティトレンド] セクションには、一定期間のトラフィックの概要メトリクスが表示されます。ダッシュボードには、許可、ブロック、チャレンジ、CAPTCHA の各リクエストに関するデータが表示されます。トラフィックの比率と、その経時的な変化を確認できます。例えば、すべてのリクエストが 3% 増加し、許可されたリクエストが 14% 増加した場合、現在の期間でトラフィックを許可した割合が高かったことになります。

このセクションには、[リクエスト]、[上位の攻撃タイプ]、[上位の国] の 3 つの棒グラフが表示されます。[上位の国] グラフを使用して国をブロックできます。

グラフを使用するには

- グラフの上にある [日付範囲]、[ルールアクション]、[詳細度] の各コントロールを使用して、時間範囲を設定し、データをフィルタリングします。

- いずれかのバーにカーソルを合わせると、指定した期間のリクエスト、攻撃、または国のデータが表示されます。
- 国をブロックするには、そのバーにカーソルを合わせ、[国名をブロック] スライダーをオンの位置に移動します。

Note

以前に国をブロックするカスタムAWS WAFルールを CloudFront コンソールの外部で作成した場合は、ブロックオプションを使用できない場合があります。

Bot Control の有効化

[特定の時間帯のボットリクエスト] セクションには、ボットリクエストデータが表示されます。AWS WAF Bot Control を有効または無効にすることもできます。Bot Control を有効にすると料金が発生し、ダッシュボードに費用の見積もりが表示されます。

Bot Control を有効にすると、ダッシュボードのグラフにはボットの種類とカテゴリごとに転送されるトラフィックの量が表示されます。Bot Control を無効にすると、グラフにはリクエストサンプリングに基づくトラフィックの量が表示されます。

このセクションには、[ボットタイプ別のリクエスト] と [ボットカテゴリ別のリクエスト] の2つの棒グラフがあります。Bot Control を有効にしているかどうかに応じて、表示されるグラフは変わります。

Bot Control を有効にするには

1. で CloudFront コンソールを開きます <https://console.aws.amazon.com/cloudfront/v4/home>。
2. ナビゲーションペインで、[ディストリビューション] を選択し、変更するディストリビューションを選択します。
3. [Security] (セキュリティ) タブを選択します。
4. [特定の時間範囲のボットリクエスト] セクションまでスクロールし、[Bot Control を有効にする] を選択します。
5. Bot Control ダイアログボックスの設定 で、一般的なボットに対して Bot Control を有効にするチェックボックスをオンにします。
6. [変更の保存] をクリックします。

ボット保護を有効にすると、未確認の各ボットをボットカテゴリごとにどのように処理するかを設定できます。例えば、HTTP ライブラリボットを [モニタリングモード] に設定し、[チャレンジ] をリンクチェッカーに割り当てることができます。

既知の検索エンジンのクローラーなど、一般的で検証可能として AWS に知られているボットは、ここで設定したアクションの対象にはなりません。Bot Control は、ボットを検証済みとしてマークする前に、検証済みのボットが請求するソースに由来することを確認します。

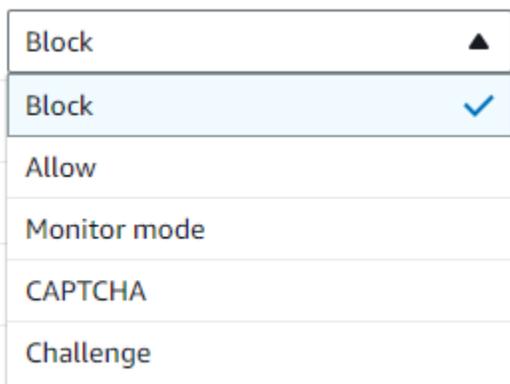
カテゴリの保護を設定するには

1. 前の手順のステップ 1 と 2 を繰り返して、[セキュリティ] ダッシュボードを起動します。
2. [ボットカテゴリ別のリクエスト] グラフで、[未確認のボットアクション] 列のいずれかの項目にカーソルを合わせ、編集アイコンを選択します。



3. 結果のリストを開き、以下のいずれかを選択します。

- ブロック
- 許可
- モニタリングモード
- CAPTCHA
- チャレンジ



4. リストの横にあるチェックマークを選択して、変更をコミットします。



グラフを使用するには

- [指定した時間範囲のセキュリティトレンド] セクションで、[日付範囲]、[ルールアクション]、[詳細度] の各コントロールを使用して、時間範囲を設定し、ポットデータをフィルタリングします。
- [ポットタイプ別のリクエスト] グラフで、いずれかのバーにカーソルを合わせると、ポットタイプ別のリクエスト数が表示されます。
- [ポットカテゴリ別のリクエスト] グラフで、いずれかのバーにカーソルを合わせると、ポットカテゴリ別のリクエスト数が表示されます。

ログの理解

ログデータは、特定のトラフィックパターンを切り分けるのに役立ちます。例えば、特定のトラフィックがどこから来ているのか、何をするのかをログで確認できます。

ログを有効にするには

1. [1 か月あたりのリクエスト数] ボックスに予想されるリクエスト量を入力して、ログを有効にした場合のコストを見積もります。
2. AWS WAF ログを有効にするチェックボックスを選択します。
3. [Enable (有効化)] を選択します。

CloudFront は CloudWatch ロググループを作成し、AWS WAF設定を更新してへのログ記録を開始します CloudWatch。最初に使用する場合は、ログデータが表示されるまでに数分かかります。グラフの [リクエスト] セクションに、各リクエストが一覧表示されます。各リクエストの下にある棒グラフは、HTTP メソッド、上位の URI パス、上位の IP アドレス、上位の国ごとにデータを集計します。グラフは、パターンを確認するのに役立ちます。例えば、1つの IP アドレスからの不釣り合いに多いリクエストや、以前はログに表示されていなかった国のデータを確認できます。国、ホストヘッダー、その他の属性に基づいてリクエストをフィルタリングすると、不要なトラフィックを見つけやすくなります。不要なトラフィックを特定したら、各リクエストやグラフ項目にカーソルを合わせ、IP アドレスまたは国をブロックします。

グラフを使用するには

- [指定した時間範囲のセキュリティトレンド] セクションの [日付範囲]、[ルールアクション]、[詳細度] の各コントロールを使用して、時間範囲を設定し、データをフィルタリングします。
- 任意のバーにカーソルを合わせると、指定した時間範囲の URI パス、IP アドレス、または国のデータが表示されます。

- IP アドレスまたは国をブロックするには、そのバーにカーソルを合わせ、[項目名をブロック] スライダーをオンの位置に移動します。

Note

以前に国または IP アドレスをブロックするカスタムAWS WAFルールを CloudFront コンソールの外部で作成した場合、ブロックオプションは使用できません。

CloudFront 地理的制限の管理

地理的制限はいつでも管理できます。

地理的制限を管理するには

1. [地理的制限] セクションまで下にスクロールします。
2. [編集] を選択します。
3. 許可された国のリストに国を追加するには [許可リスト] を選択し、ブロックされた国のリストに国を追加するには [ブロックリスト] を選択します。
4. 目的の国をリストに追加し、[変更を保存] を選択します。

CloudFront と AWS WAF は地理的制限機能を提供します。は地理的制限を無料で CloudFront 提供しますが、ダッシュボードにはブロックされた国のメトリクスは表示されません。一方、[セキュリティ] ダッシュボードの国のバーにカーソルを合わせて国をブロックすると、AWS WAF 地理的制限が適用されます。国をブロックするだけでなく、ダッシュボードには、ブロックしたリクエストのリクエストメトリクスも表示されます。

セキュリティダッシュボードの料金

Amazon へのAWS WAFログ記録を有効にすると CloudWatch、CloudFront セキュリティダッシュボードは CloudWatch ログからインサイトをクエリ、集約、表示します。セキュリティダッシュボードの使用には料金はありませんが、Amazon の CloudWatch 料金はダッシュボードからクエリされたログに適用されます。詳細については、[「Amazon の CloudWatch 料金」](#)を参照してください。

コンテンツへのセキュアなアクセスとアクセス制限の設定

CloudFront には、配信するコンテンツを保護するためのオプションがいくつか用意されています。以下は、CloudFront を使用してコンテンツへのアクセスを保護し、制限する方法です。

- HTTPS 接続を設定する
- 特定の地理的な場所にいるユーザーがコンテンツにアクセスできないようにする
- CloudFront 署名付き URLs または署名付き Cookie を使用してコンテンツにアクセスすることをユーザーに要求する
- 特定のコンテンツフィールドのフィールドレベルの暗号化を設定する
- AWS WAF を使用してコンテンツへのアクセスを管理する

トピック

- [で HTTPS を使用する CloudFront](#)
- [代替ドメイン名と HTTPS の使用](#)
- [署名付き URL と署名付き Cookie を使用したプライベートコンテンツの提供](#)
- [AWS オリジンへのアクセスの制限](#)
- [Application Load Balancers へのアクセスを制限する](#)
- [コンテンツの地理的ディストリビューションの制限](#)
- [フィールドレベル暗号化を使用した機密データの保護](#)

で HTTPS を使用する CloudFront

ビューワーが HTTPS を使用する CloudFront ように を設定して、 がビューワーと CloudFront 通信するときに接続が暗号化されるようにすることができます。 がオリジンと CloudFront 通信するときに接続が暗号化されるように、オリジンで HTTPS を使用する CloudFront ように を設定することもできます。

ビューワーとの通信とオリジンとの通信の両方で HTTPS を必須と CloudFront するように を設定した場合、 がリクエスト CloudFront を受信したときに次の処理が行われます。

1. ビューワーが HTTPS リクエストを に送信します CloudFront。ビューワーと の間には、SSL/TLS ネゴシエーションがいくつかあります CloudFront。最終的に、ビューワーはリクエストを暗号化形式で送信します。

2. CloudFront エッジロケーションにキャッシュされたレスポンスが含まれている場合、はレスポンスを CloudFront 暗号化してビューワーに返し、ビューワーはそれを復号します。
3. CloudFront エッジロケーションにキャッシュされたレスポンスが含まれていない場合、はオリジンとの SSL/TLS ネゴシエーション CloudFront を実行し、ネゴシエーションが完了すると、リクエストを暗号化された形式でオリジンに転送します。
4. オリジンはリクエストを復号し、処理 (レスポンスを生成) してレスポンスを暗号化し、レスポンスを に返します CloudFront。
5. CloudFront はレスポンスを復号し、再暗号化して、ビューワーに転送します。 CloudFront また、は、次にリクエストされたときに使用できるように、レスポンスをエッジロケーションにキャッシュします。
6. ビューワーは応答の暗号化を解除します。

このプロセスは、オリジンが Amazon S3 バケットであるか、HTTP/S サーバーなどのカスタムオリジンであるかにかかわらず MediaStore、基本的に同じように機能します。

Note

SSL 再ネゴシエーションタイプの攻撃を阻止するために、CloudFront はビューワーリクエストとオリジンリクエストの再ネゴシエーションをサポートしていません。

ビューワーと の間 CloudFront、および CloudFront とオリジンの間で HTTPS を必須にする方法については、以下のトピックを参照してください。

トピック

- [ビューワーと 間の通信に HTTPS を必須にする CloudFront](#)
- [CloudFront とカスタムオリジン間の通信に HTTPS を必須にする](#)
- [と Amazon S3 オリジン間の CloudFront通信に HTTPS を必須にする Amazon S3](#)
- [ビューワーと の間でサポートされているプロトコルと暗号 CloudFront](#)
- [とオリジン間でサポートされているプロトコル CloudFront と暗号](#)
- [HTTPS 接続料金](#)

ビューワーと 間の通信に HTTPS を必須にする CloudFront

ビューワーと 間の通信に HTTPS を要求するように、CloudFront デイストリビューション内の 1 つ以上のキャッシュ動作を設定できます CloudFront。また、HTTP と HTTPS の両方を許可するように 1 つ以上のキャッシュ動作を設定することもできます。これにより、では一部のオブジェクトに HTTPS CloudFront が必要ですが、他のオブジェクトには HTTPS を必要としません。設定手順はオブジェクト URL 内で使用しているドメイン名によって異なります。

- d111111abcdef8.cloudfront.net など、デイストリビューションに が CloudFront 割り当てたドメイン名を使用している場合は、1 つ以上のキャッシュ動作のビューワープロトコルポリシー設定を変更して HTTPS 通信を要求します。この設定で、CloudFront は SSL/TLS 証明書を提供します。

CloudFront コンソールを使用してビューワープロトコルポリシーの値を変更するには、このセクションの後半の手順を参照してください。

CloudFront API を使用して ViewerProtocolPolicy 要素の値を変更する方法については、「Amazon CloudFront API リファレンス」の [UpdateDistribution](#) 「」を参照してください。

- 独自のドメイン名 (example.com など) を使用している場合、CloudFront のいくつかの設定を変更する必要があります。また、AWS Certificate Manager (ACM) が提供する SSL/TLS 証明書を使用するか、サードパーティー認証機関からの証明書を ACM または IAM 証明書ストアにインポートする必要があります。詳細については、「[代替ドメイン名と HTTPS の使用](#)」を参照してください。

Note

ビューワーが取得するオブジェクトがオリジンから CloudFront 取得したときに CloudFront 暗号化されたことを確認する場合は、常に CloudFront とオリジンの間で HTTPS を使用します。CloudFront とオリジンの間で最近 HTTP から HTTPS に変更した場合は、CloudFront エッジロケーション内のオブジェクトを無効にすることをお勧めします。ビューワーが使用するプロトコル (HTTP または HTTPS) が、オブジェクトの取得に CloudFront が使用したプロトコルと一致するかどうかにかかわらず、ビューワーにオブジェクト CloudFront を返します。デイストリビューション内のオブジェクトの削除または置き換えの詳細については、「[が配信する CloudFront コンテンツの追加、削除、または置き換え](#)」を参照してください。

ビューワーと の間で 1 つ以上のキャッシュ動作 CloudFront に HTTPS を必須にするには、次の手順を実行します。

ビューワーとの間で HTTPS を要求する CloudFront ように を設定するには CloudFront

1. にサインインAWS Management Consoleし、 で CloudFront コンソールを開きます<https://console.aws.amazon.com/cloudfront/v4/home>。
2. CloudFront コンソールの上部のペインで、更新するディストリビューションの ID を選択します。
3. [Behaviors] タブで、更新するキャッシュ動作を選択した後、[Edit] を選択します。
4. [Viewer Protocol Policy] として次のいずれかの値を指定します。

Redirect HTTP to HTTPS

ビューワーは両方のプロトコルを使用できます。HTTP GETおよび HEADリクエストは、新しい HTTPS URL とともに HTTPS リクエストに自動的にリダイレクトされます。CloudFront は HTTP ステータスコード 301 (Moved Permanently) を返します。次に、ビューワーは HTTPS URL CloudFront を使用して にリクエストを再送信します。

Important

HTTP から HTTPS PUTへのキャッシュ動作と HTTP 1.1 以降のリクエストプロトコルバージョンを使用して POST、DELETE、OPTIONSまたは HTTP PATCH経由で送信した場合、 は HTTP ステータスコード 307 (Temporary Redirect) を使用してリクエストを HTTPS ロケーションに CloudFront リダイレクトします。これはリクエストが同じメソッドと本文ペイロードを使用して新しい場所に再度送信されることを保証するものです。

、POST、PUT、DELETEOPTIONS、または PATCHリクエストを HTTP 経由で HTTPS キャッシュ動作に HTTP 1.1 より前のリクエストプロトコルバージョンで送信すると、 は HTTP ステータスコード 403 (Forbidden) CloudFront を返します。

HTTPS リクエストにリダイレクトされる HTTP リクエストをビューワーが作成すると、CloudFront は両方のリクエストに課金します。HTTP リクエストの場合、料金はリクエストと がビューワーに CloudFront 返すヘッダーに対してのみ発生します。HTTPS リクエストの場合、リクエストの料金と、オリジンが返すヘッダーとオブジェクトの料金が課金されます。

HTTPS Only

ビューワーは、HTTPS を使用している場合にのみ、コンテンツにアクセスできます。

ビューワーが HTTPS リクエストではなく HTTP リクエストを送信すると、は HTTP ステータスコード 403 (Forbidden) を CloudFront 返し、オブジェクトは返しません。

5. [Yes, Edit (はい、編集します)] を選択します。
6. ビューワーと の間で HTTPS を要求する追加のキャッシュ動作ごとに、ステップ 3~5 を繰り返します CloudFront。
7. 本番環境で更新された情報を使用する前に、次を確認してください。
 - ビューワーに HTTPS の使用が必要とされるリクエストにのみ、各キャッシュ動作のパスパターンが適用されている。
 - キャッシュ動作は、CloudFront 評価する順序で一覧表示されます。詳細については、「[パスパターン](#)」を参照してください。
 - キャッシュ動作は、リクエストを正しいオリジンにルーティングします。

CloudFront とカスタムオリジン間の通信に HTTPS を必須にする

CloudFront とオリジン間の通信に HTTPS を要求できます。

Note

オリジンがウェブサイトエンドポイントとして設定されている Amazon S3 バケットである場合、Amazon S3 はウェブサイトエンドポイントの HTTPS をサポートしていないため、オリジンで HTTPS を使用する CloudFront ように を設定することはできません。

CloudFront とオリジン間で HTTPS を必須にするには、このトピックの手順に従って次の操作を行います。

1. ディストリビューションで、[Origin Protocol Policy] (オリジンプロトコルポリシー) 設定を変更します。
2. オリジンサーバーに SSL/TLS 証明書をインストールします (Amazon S3 オリジン、または特定のその他の AWS オリジンを使用する場合は必要ありません)。

トピック

- [設定を変更する CloudFront](#)
- [カスタムオリジンへの SSL/TLS 証明書のインストール](#)

設定を変更する CloudFront

次の手順では、HTTPS CloudFront を使用して Elastic Load Balancing ロードバランサー、Amazon EC2 インスタンス、または別のカスタムオリジンと通信するようにを設定する方法について説明します。CloudFront API を使用してディストリビューションを更新する方法については、「Amazon CloudFront API リファレンス」の[UpdateDistribution](#)「」を参照してください。

CloudFront とカスタムオリジン間で HTTPS を必須 CloudFront とするようにを設定するには

1. にサインインAWS Management Consoleし、 で CloudFront コンソールを開きます<https://console.aws.amazon.com/cloudfront/v4/home>。
2. CloudFront コンソールの上部のペインで、更新するディストリビューションの ID を選択します。
3. [Origins] タブで、更新するオリジンを選択し、[Edit] を選択します。
4. 次の設定を更新します。

オリジンプロトコルポリシー

ディストリビューションの該当するオリジンで、[Origin Protocol Policy] を変更します。

- HTTPS のみ — カスタムオリジンとの通信には HTTPS のみ CloudFront を使用します。
- 一致ビューワー — CloudFront ビューワーリクエストのプロトコルに応じて、HTTP または HTTPS を使用してカスタムオリジンと通信します。例えば、オリジンプロトコルポリシーで一致ビューワーを選択し、ビューワーが HTTPS を使用してにオブジェクトをリクエストする場合 CloudFront、CloudFront は HTTPS を使用してリクエストをオリジンに転送します。

[Viewer Protocol Policy] で [Redirect HTTP to HTTPS] または [HTTPS Only] を指定する場合は、[Match Viewer] のみを選択します。

CloudFront ビューワーが HTTP プロトコルと HTTPS プロトコルの両方を使用してリクエストを行った場合でも、はオブジェクトを 1 回だけキャッシュします。

オリジン SSL プロトコル

ディストリビューションの該当するオリジンで [Origin SSL Protocols] を選択します。SSLv3 プロトコルは安全性が低いため、オリジンが TLSv1 以降をサポートしていない場合にのみ SSLv3 を選択することをお勧めします。TLSv1 ハンドシェイクは SSLv3 と互換性がありますが、TLSv1.1 と TLSv1.2 にはありません。SSLv3 を選択すると、は SSLv3 ハンドシェイクリクエスト CloudFront のみを送信します。

5. [Yes, Edit (はい、編集します)] を選択します。
6. CloudFront とカスタムオリジンの間で HTTPS を要求する追加のオリジンごとに、ステップ 3~5 を繰り返します。
7. 本番環境で更新された情報を使用する前に、次を確認してください。
 - ビューワーに HTTPS の使用が必要とされるリクエストにのみ、各キャッシュ動作のパスパターンが適用されている。
 - キャッシュ動作は、CloudFront 評価する順序で一覧表示されます。詳細については、「[パスパターン](#)」を参照してください。
 - キャッシュ動作は、[Origin Protocol Policy] を変更したオリジンにリクエストをルーティングします。

カスタムオリジンへの SSL/TLS 証明書のインストール

SSL/TLS 証明書は、カスタムオリジンの次のソースから使用できます。

- オリジンが Elastic Load Balancing ロードバランサーである場合は、AWS Certificate Manager (ACM) で提供された証明書を使用できます。信頼されたサードパーティー認証機関が署名して ACM にインポートされた証明書を使用することもできます。
- Elastic Load Balancing ロードバランサー以外のオリジンの場合は、信頼されたサードパーティー認証機関 (CA) によって署名された証明書を使用する必要があります。例えば、Comodo、DigiCertSymantec などです。

オリジンから返される証明書には、次のいずれかのドメイン名が含まれている必要があります。

- オリジンのオリジンドメインフィールド (CloudFront API の DomainName フィールド) のドメイン名。
- キャッシュ動作が Host ヘッダーをオリジンに転送するように設定されている場合は、Host ヘッダーのドメイン名。

が HTTPS CloudFront を使用してオリジンと通信する場合、CloudFront は証明書が信頼できる認証機関によって発行されたことを確認します。は Mozilla と同じ認証機関 CloudFront をサポートします。最新のリストは、「[Mozilla に付属する CA 証明書一覧](#)」を参照してください。CloudFront とオリジン間の HTTPS 通信に自己署名証明書を使用することはできません。

Important

オリジンサーバーが期限切れの証明書、無効な証明書、または自己署名証明書を返した場合、またはオリジンサーバーが証明書チェーンを間違った順序で返した場合、は TCP 接続を CloudFront ドロップし、ビューワーに HTTP ステータスコード 502 (Bad Gateway) を返し、X-Cacheヘッダーを に設定しますError from cloudfront。また、中間証明書を含む証明書の完全なチェーンが存在しない場合、は TCP 接続を CloudFront ドロップします。

と Amazon S3 オリジン間の CloudFront通信に HTTPS を必須にする Amazon S3

オリジンが Amazon S3 バケットの場合、との通信に HTTPS を使用するオプションは、バケットの使用法 CloudFront によって異なります。Amazon S3 バケットがウェブサイトエンドポイントとして設定されている場合、HTTPS CloudFront を使用してオリジンと通信するように を設定することはできません。Amazon S3 はその設定で HTTPS 接続をサポートしていないためです。

オリジンが HTTPS 通信をサポートする Amazon S3 バケットの場合、CloudFront は常にビューワーがリクエストの送信に使用したプロトコルを使用してリクエストを S3 に転送します。[プロトコル \(カスタムオリジンのみ\)](#) 設定のデフォルト設定は [Match Viewer (ビューワーに合わせる)] で、変更できません。

CloudFront と Amazon S3 間の通信に HTTPS を要求する場合は、ビューワープロトコルポリシーの値を HTTP から HTTPS または HTTPS のみにリダイレクトするように変更する必要があります。このセクションの後半の手順では、CloudFront コンソールを使用してビューワープロトコルポリシー を変更する方法について説明します。CloudFront API を使用してディストリビューションの ViewerProtocolPolicy要素を更新する方法については、「Amazon CloudFront API リファレンス[UpdateDistribution](#)」の「」を参照してください。

HTTPS 通信をサポートする Amazon S3 バケットで HTTPS を使用する場合、Amazon S3 では SSL/TLS 証明書を使用できるため、この通信を使用する必要はありません。

Amazon S3 オリジン CloudFront に HTTPS を要求するようにを設定するには Amazon S3

1. にサインインAWS Management Consoleし、 で CloudFront コンソールを開きます<https://console.aws.amazon.com/cloudfront/v4/home>。
2. CloudFront コンソールの上部のペインで、更新するディストリビューションの ID を選択します。
3. [Behaviors] タブで、更新するキャッシュ動作を選択した後、[Edit] を選択します。
4. [Viewer Protocol Policy] として次のいずれかの値を指定します。

Redirect HTTP to HTTPS

ビューワーは両方のプロトコルを使用できますが、HTTP リクエストは自動的に HTTPS リクエストにリダイレクトされます。CloudFront は、新しい HTTPS URL とともに HTTP ステータスコード 301 (Moved Permanently) を返します。次に、ビューワーは HTTPS URL CloudFront を使用して にリクエストを再送信します。

Important

CloudFront は DELETE、OPTIONS、PATCH、POST または PUT リクエストを HTTP から HTTPS にリダイレクトしません。HTTPS にリダイレクトするようにキャッシュ動作を設定すると、 は HTTP ステータスコード 403 (禁止) でそのキャッシュ動作に対する OPTIONS、DELETE、PATCH、POST、または PUT リクエスト CloudFront に対応します。

HTTPS リクエストにリダイレクトされる HTTP リクエストをビューワーが作成すると、CloudFront は両方のリクエストに課金します。HTTP リクエストの場合、料金はリクエストと がビューワーに CloudFront 返すヘッダーに対してのみ発生します。HTTPS リクエストの場合、リクエストの料金と、オリジンが返すヘッダーとオブジェクトの料金が課金されます。

HTTPS Only

ビューワーは、HTTPS を使用している場合にのみ、コンテンツにアクセスできます。ビューワーが HTTPS リクエストではなく HTTP リクエストを送信すると、 は HTTP ステータスコード 403 (Forbidden) を CloudFront 返し、 オブジェクトは返しません。

5. [Yes, Edit (はい、編集します)] を選択します。

6. ビューワーとの間、および S3 の間の HTTPS を要求する追加のキャッシュ動作ごとに CloudFront、ステップ 3~5 CloudFront を繰り返します。
7. 本番環境で更新された情報を使用する前に、次を確認してください。
 - ビューワーに HTTPS の使用が必要とされるリクエストにのみ、各キャッシュ動作のパスパターンが適用されている。
 - キャッシュ動作は、CloudFront 評価する順序で一覧表示されます。詳細については、「[パスパターン](#)」を参照してください。
 - キャッシュ動作は、リクエストを正しいオリジンにルーティングします。

ビューワーとの間でサポートされているプロトコルと暗号 CloudFront

[ビューワーとデイス CloudFront トリビューション間で HTTPS が必要な場合は](#)、[セキュリティポリシー](#) を選択する必要があります。これにより、次の設定が決まります。

- がビューワーとの通信 CloudFront に使用する最低限の SSL/TLS プロトコル。
- がビューワーとの通信を暗号化するために CloudFront 使用できる暗号。

セキュリティポリシーを選択するには、[セキュリティポリシー](#) に該当する値を指定します。次の表に、各セキュリティポリシーに CloudFront が使用できるプロトコルと暗号を示します。

ビューワーは、との HTTPS 接続を確立するには、サポートされている暗号の少なくとも 1 つをサポートする必要があります CloudFront。ビューワーがサポートする暗号の中から、リストされた順序で暗号 CloudFront を選択します。「[OpenSSL、s2n、および RFC の暗号名](#)」も参照してください。

	セキュリティポリシー						
	SSLv3	TLSv1	TLSv1_2 6	TLSv1.1 016	TLSv1.2 018	TLSv1.2 019	TLSv1.2_2 021

サポートされている SSL/TLS プロトコル

TLSv1.3	◆	◆	◆	◆	◆	◆	◆
TLSv1.2	◆	◆	◆	◆	◆	◆	◆

	セキュリティポリシー						
	SSLv3	TLSv1	TLSv1.2_6	TLSv1.1_016	TLSv1.2_018	TLSv1.2_019	TLSv1.2_021
TLSv1.1	◆	◆	◆	◆			
TLSv1	◆	◆	◆				
SSLv3	◆						

サポートされている TLSv1.3 暗号

TLS_AES_128_GCM_SHA256	◆	◆	◆	◆	◆	◆	◆
TLS_AES_256_GCM_SHA384	◆	◆	◆	◆	◆	◆	◆
TLS_CHACHA20_POLY1305_SHA256	◆	◆	◆	◆	◆	◆	◆

サポートされる ECDSA 暗号

ECDHE-ECDSA-AES128-GCM-SHA256	◆	◆	◆	◆	◆	◆	◆
ECDHE-ECDSA-AES128-SHA256	◆	◆	◆	◆	◆	◆	
ECDHE-ECDSA-AES128-SHA	◆	◆	◆	◆			
ECDHE-ECDSA-AES256-GCM-SHA384	◆	◆	◆	◆	◆	◆	◆
ECDHE-ECDSA-CHACHA20-POLY1305	◆	◆	◆	◆	◆	◆	◆

	セキュリティポリシー						
	SSLv3	TLSv1	TLSv1.2_6	TLSv1.1_016	TLSv1.2_018	TLSv1.2_019	TLSv1.2_2021
ECDHE-ECDSA-AES256-SHA384	◆	◆	◆	◆	◆	◆	
ECDHE-ECDSA-AES256-SHA	◆	◆	◆	◆			
サポートされる RSA 暗号							
ECDHE-RSA-AES128-GCM-SHA256	◆	◆	◆	◆	◆	◆	◆
ECDHE-RSA-AES128-SHA256	◆	◆	◆	◆	◆	◆	
ECDHE-RSA-AES128-SHA	◆	◆	◆	◆			
ECDHE-RSA-AES256-GCM-SHA384	◆	◆	◆	◆	◆	◆	◆
ECDHE-RSA-CHACHA20-POLY1305	◆	◆	◆	◆	◆	◆	◆
ECDHE-RSA-AES256-SHA384	◆	◆	◆	◆	◆	◆	
ECDHE-RSA-AES256-SHA	◆	◆	◆	◆			
AES128-GCM-SHA256	◆	◆	◆	◆	◆		
AES256-GCM-SHA384	◆	◆	◆	◆	◆		
AES128-SHA256	◆	◆	◆	◆	◆		

	セキュリティポリシー						
	SSLv3	TLSv1	TLSv1.2_6	TLSv1.1_016	TLSv1.2_018	TLSv1.2_019	TLSv1.2_2021
AES256-SHA	◆	◆	◆	◆			
AES128-SHA	◆	◆	◆	◆			
DES-CBC3-SHA	◆	◆					
RC4-MD5	◆						

OpenSSL、s2n、および RFC の暗号名

OpenSSL と [s2n](#) では、TLS 標準で使用されている暗号名 ([RFC 2246](#)、[RFC 4346](#)、[RFC 5246](#)、[RFC 8446](#)) とは異なる暗号名が使用されます。以下の表では、各暗号化方式の OpenSSL 名と s2n 名から RFC 名までを示しています。

楕円曲線キー交換アルゴリズムを使用する暗号の場合、CloudFront は次の楕円曲線をサポートしません。

- prime256v1
- secp384r1
- X25519

OpenSSL および s2n の暗号名	RFC の暗号名
サポートされている TLSv1.3 暗号	
TLS_AES_128_GCM_SHA256	TLS_AES_128_GCM_SHA256
TLS_AES_256_GCM_SHA384	TLS_AES_256_GCM_SHA384
TLS_CHACHA20_POLY1305_SHA256	TLS_CHACHA20_POLY1305_SHA256
サポートされる ECDSA 暗号	

OpenSSL および s2n の暗号名	RFC の暗号名
ECDHE-ECDSA-AES128- GCM-SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
ECDHE-ECDSA-AES128-SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
ECDHE-ECDSA-AES128-SHA	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
ECDHE-ECDSA-AES256- GCM-SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
ECDHE-ECDSA-CHACHA20-POLY1305	TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256
ECDHE-ECDSA-AES256-SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
ECDHE-ECDSA-AES256-SHA	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
サポートされる RSA 暗号	
ECDHE-RSA-AES128- GCM-SHA256	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
ECDHE-RSA-AES128-SHA256	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
ECDHE-RSA-AES128-SHA	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
ECDHE-RSA-AES256- GCM-SHA384	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
ECDHE-RSA-CHACHA20-POLY1305	TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256

OpenSSL および s2n の暗号名	RFC の暗号名
ECDHE-RSA-AES256-SHA384	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
ECDHE-RSA-AES256-SHA	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
AES128-GCM-SHA256	TLS_RSA_WITH_AES_128_GCM_SHA256
AES256-GCM-SHA384	TLS_RSA_WITH_AES_256_GCM_SHA384
AES128-SHA256	TLS_RSA_WITH_AES_128_CBC_SHA256
AES256-SHA	TLS_RSA_WITH_AES_256_CBC_SHA
AES128-SHA	TLS_RSA_WITH_AES_128_CBC_SHA
DES-CBC3-SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA
RC4-MD5	TLS_RSA_WITH_RC4_128_MD5

ビューワーとの間でサポートされている署名スキーム CloudFront

CloudFront は、ビューワーと 間の接続について、次の署名スキームをサポートしますCloudFront。

- TLS_SIGNATURE_SCHEME_RSA_PSS_PSS_SHA256
- TLS_SIGNATURE_SCHEME_RSA_PSS_PSS_SHA384
- TLS_SIGNATURE_SCHEME_RSA_PSS_PSS_SHA512
- TLS_SIGNATURE_SCHEME_RSA_PSS_RSAE_SHA256
- TLS_SIGNATURE_SCHEME_RSA_PSS_RSAE_SHA384
- TLS_SIGNATURE_SCHEME_RSA_PSS_RSAE_SHA512
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA256
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA384
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA512
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA224
- TLS_SIGNATURE_SCHEME_ECDSA_SHA256

- TLS_SIGNATURE_SCHEME_ECDSA_SHA384
- TLS_SIGNATURE_SCHEME_ECDSA_SHA512
- TLS_SIGNATURE_SCHEME_ECDSA_SHA224
- TLS_SIGNATURE_SCHEME_ECDSA_SECP256R1_SHA256
- TLS_SIGNATURE_SCHEME_ECDSA_SECP384R1_SHA384
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA1
- TLS_SIGNATURE_SCHEME_ECDSA_SHA1

とオリジン間でサポートされているプロトコル CloudFront と暗号

[CloudFront とオリジンとの間で HTTPS を必須にする場合は、安全な接続を許可する SSL/TLS プロトコル](#)を決定し、次の表に示す ECDSA または RSA 暗号のいずれかを使用してオリジンに接続 CloudFront できます。オリジン CloudFront への HTTPS 接続を確立するには、オリジンがこれらの暗号のうち少なくとも 1 つをサポートしている必要があります。

OpenSSL と [s2n](#) では、TLS 標準で使用されている暗号名 ([RFC 2246](#)、[RFC 4346](#)、[RFC 5246](#)、[RFC 8446](#)) とは異なる暗号名が使用されます。次の表には、各暗号化方式の OpenSSL 名、s2n 名、および RFC 名などが示されています。

楕円曲線キー交換アルゴリズムを使用する暗号の場合、CloudFront は次の楕円曲線をサポートします。

- prime256v1
- secp384r1
- X25519

OpenSSL および s2n の暗号名	RFC の暗号名
サポートされる ECDSA 暗号	
ECDHE-ECDSA-AES256- GCM-SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
ECDHE-ECDSA-AES256-SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384

OpenSSL および s2n の暗号名	RFC の暗号名
ECDHE-ECDSA-AES256-SHA	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
ECDHE-ECDSA-AES128- GCM-SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
ECDHE-ECDSA-AES128-SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
ECDHE-ECDSA-AES128-SHA	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
サポートされる RSA 暗号	
ECDHE-RSA-AES256- GCM-SHA384	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
ECDHE-RSA-AES256-SHA384	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
ECDHE-RSA-AES256-SHA	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
ECDHE-RSA-AES128- GCM-SHA256	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
ECDHE-RSA-AES128-SHA256	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
ECDHE-RSA-AES128-SHA	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
AES256-SHA	TLS_RSA_WITH_AES_256_CBC_SHA
AES128-SHA	TLS_RSA_WITH_AES_128_CBC_SHA
DES-CBC3-SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA

OpenSSL および s2n の暗号名	RFC の暗号名
RC4-MD5	TLS_RSA_WITH_RC4_128_MD5

CloudFront とオリジン間でサポートされている署名スキーム

CloudFront は、CloudFront とオリジン間の接続について、次の署名スキームをサポートします。

- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA256
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA384
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA512
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA224
- TLS_SIGNATURE_SCHEME_ECDSA_SHA256
- TLS_SIGNATURE_SCHEME_ECDSA_SHA384
- TLS_SIGNATURE_SCHEME_ECDSA_SHA512
- TLS_SIGNATURE_SCHEME_ECDSA_SHA224
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA1
- TLS_SIGNATURE_SCHEME_ECDSA_SHA1

HTTPS 接続料金

HTTPS リクエストに対する追加料金が常に発生します。詳細については、[「Amazon の CloudFront 料金」](#)を参照してください。

代替ドメイン名と HTTPS の使用

ファイルの URL で独自ドメイン名を使用する場合 (例: `https://www.example.com/image.jpg`)、およびビューワーが HTTPS を使用する場合、次のトピックのステップを完了する必要があります。(URLs でデフォルトの CloudFront デイストリビューションドメイン名を使用する場合、例えばを使用する場合は`https://d1111111abcdef8.cloudfront.net/image.jpg`、代わりに次のトピックのガイダンスに従ってください: [ビューワーと間の通信に HTTPS を必須にする CloudFront。](#))

⚠ Important

ディストリビューションに証明書を追加すると、CloudFront はすぐに証明書をすべてのエッジロケーションに伝達します。新しいエッジロケーションが使用可能になると、CloudFront は証明書をそのロケーションにも伝達します。証明書を CloudFront 伝達するエッジロケーションを制限することはできません。

トピック

- [が HTTPS リクエスト CloudFront を処理する方法の選択](#)
- [で SSL/TLS 証明書を使用するための要件 CloudFront](#)
- [での SSL/TLS 証明書の使用に関するクォータ CloudFront \(ビューワーと間の HTTPS CloudFront のみ\)](#)
- [代替ドメイン名と HTTPS の設定](#)
- [SSL/TLS RSA 証明書内のパブリック \(公開\) キーのサイズの確認](#)
- [SSL/TLS 証明書のクォータの引き上げ](#)
- [SSL/TLS 証明書の更新](#)
- [カスタム SSL/TLS 証明書からデフォルト CloudFront 証明書に戻す](#)
- [独自 SSL/TLS 証明書を専用 IP アドレスから SNI に切り替える](#)

が HTTPS リクエスト CloudFront を処理する方法の選択

ビューワーに HTTPS を使用し、ファイルの代替ドメイン名を使用させる場合は、が HTTPS リクエスト CloudFront を処理する方法に関する以下のオプションのいずれかを選択します。

- [Server Name Indication \(SNI\) を使用する - 推奨](#)
- 各エッジロケーションの専用 IP アドレスを使用する

このセクションでは各オプションの仕組みについて説明します。

SNI を使用した HTTPS リクエストの処理 (ほとんどのクライアントで動作)

[Server Name Indication \(SNI\)](#) は、2010 年以降にリリースされたブラウザとクライアントでサポートされている TLS プロトコルを拡張したものです。SNI を使用して HTTPS リクエストを処理する CloudFront ように を設定した場合、 は代替ドメイン名を各エッジロケーションの IP アドレス

スに CloudFront 関連付けます。ビューワーがコンテンツに対して HTTPS リクエストを送信すると、DNS は、正しいエッジロケーションの IP アドレスにリクエストをルーティングします。ドメイン名の IP アドレスが SSL/TLS ハンドシェイクネゴシエーション中に決定されます。IP アドレスはディストリビューション専用にはなりません。

SSL/TLS ネゴシエーションは、HTTPS 接続を確立する処理の早い段階で実行されます。

CloudFront がリクエスト対象のドメインをすぐに判断できない場合、接続は切断されます。SNI をサポートするビューワーがコンテンツに対して HTTPS リクエストを送信すると、次のようになります。

1. ビューワーはリクエスト URL から自動的にドメイン名を取得し、リクエストヘッダーのフィールドに追加します。
2. がリクエスト CloudFront を受信すると、リクエストヘッダーでドメイン名を検索し、該当する SSL/TLS 証明書でリクエストに応答します。
3. ビューワーと が SSL/TLS ネゴシエーション CloudFront を実行します。
4. CloudFront は、リクエストされたコンテンツをビューワーに返します。

現在 SNI をサポートするブラウザの一覧については、Wikipedia の [Server Name Indication](#) の項目を参照してください。

SNI を使用したくても、ユーザーのブラウザの一部が SNI をサポートしていない場合は、選択肢がいくつかあります。

- SNI の代わりに専用 IP アドレスを使用して HTTPS リクエストを処理する CloudFront ように を設定します。詳細については、「[専用 IP アドレスを使用した HTTPS リクエストの処理 \(すべてのクライアントで動作\)](#)」を参照してください。
- カスタム証明書の代わりに CloudFront SSL/TLS 証明書を使用します。そのためには、ディストリビューションの CloudFront ドメイン名をファイルの URLs で使用する必要があります。例えば、`https://d1111111abcdef8.cloudfront.net/logo.png`。

デフォルトの CloudFront 証明書を使用する場合、ビューワーは SSL プロトコル TLSv1 以降をサポートする必要があります。CloudFront デフォルト CloudFront 証明書の SSLv3 はサポートされません。

また、CloudFront が使用している SSL/TLS 証明書をカスタム証明書からデフォルト CloudFront 証明書に変更する必要があります。

- ディストリビューションを使用してコンテンツを配信したことがない場合は、単に構成を変更できません。詳細については、「[ディストリビューションの更新](#)」を参照してください。
- ディストリビューションを使用してコンテンツを配信した場合は、新しいディストリビューションを作成し、ファイルの URLs を変更して、コンテンツが使用できない時間を短縮または削除する必要があります。詳細については、「[カスタム SSL/TLS 証明書からデフォルト CloudFront 証明書に戻す](#)」を参照してください。
- ユーザーが使用するブラウザを管理できる場合は、SNI をサポートするブラウザにアップグレードしてもらいます。
- HTTPS の代わりに HTTP を使用します。

専用 IP アドレスを使用した HTTPS リクエストの処理 (すべてのクライアントで動作)

Server Name Indication (SNI) は、リクエストをドメインと関連付ける 1 つの方法です。専用 IP アドレスを使用する方法もあります。2010 年以降にリリースされたブラウザまたはクライアントにアップグレードできないユーザーが存在する場合は、専用 IP アドレスを使用して HTTPS リクエストに対応できます。現在 SNI をサポートするブラウザの一覧については、Wikipedia の [Server Name Indication](#) の項目を参照してください。

Important

専用 IP アドレスを使用して HTTPS リクエストを処理する CloudFront ように を設定すると、追加の月額料金が発生します。課金は、ディストリビューションに SSL/TLS 証明書を関連付けて、ディストリビューションを有効にした時点から開始されます。CloudFront 料金の詳細については、「[Amazon 料金表 CloudFront](#)」を参照してください。また、[Using the Same Certificate for Multiple CloudFront Distributions](#) も参照してください。

専用 IP アドレスを使用して HTTPS リクエストを処理する CloudFront ように を設定すると、 は代替ドメイン名を各 CloudFront エッジロケーションの専用 IP アドレスに CloudFront 関連付けます。ビューワーがコンテンツに HTTPS リクエストを送信すると、次のようになります。

1. DNS は、該当するエッジロケーション内のディストリビューションの IP アドレスにリクエストをルーティングします。
2. CloudFront は IP アドレスを使用してディストリビューションを識別し、ビューワーに返す SSL/TLS 証明書を決定します。

3. ビューワーとは、SSL/TLS 証明書を使用して SSL/TLS ネゴシエーション CloudFront を実行します。
4. CloudFront は、リクエストされたコンテンツをビューワーに返します。

この方法は、ユーザーが使用するブラウザやその他のビューワーを問わず、あらゆる HTTPS リクエストで機能します。

3 つ以上の SSL/TLS 専用 IP 証明書を使用する許可をリクエストする

3 つ以上の SSL/TLS 専用 IP 証明書を に永続的に関連付けるアクセス許可が必要な場合は CloudFront、次の手順を実行します。HTTPS リクエストの詳細については、「[が HTTPS リクエスト CloudFront を処理する方法の選択](#)」を参照してください。

Note

この手順は、CloudFront デイストリビューションで 3 つ以上の専用 IP 証明書を使用するためのものです。デフォルト値は 2 です。デイストリビューションには SSL 証明書を複数バインドできないのでご注意ください。

一度に 1 つの SSL/TLS 証明書のみを CloudFront デイストリビューションに関連付けることができます。この数は、すべての CloudFront デイストリビューションで使用できる専用 IP SSL 証明書の合計数です。

CloudFront デイストリビューションに 3 つ以上の証明書を使用するための許可をリクエストする

1. [サポートセンター](#)にアクセスし、サポートケースを作成します。
2. 使用するためのアクセス権限が必要な証明書の数と状況をリクエストに記載してください。できる限り早くアカウントを更新します。
3. 次の手順に進みます。

で SSL/TLS 証明書を使用するための要件 CloudFront

このトピックでは、SSL/TLS 証明書の要件について説明します。これらの要件は、特に注記がなければ、以下の両方の証明書に適用されます。

- ビューワーと の間で HTTPS を使用するための証明書 CloudFront
- CloudFront とオリジンの間で HTTPS を使用するための証明書

トピック

- [証明書が発行者](#)
- [AWS Certificate Manager の場合は AWS リージョン](#)
- [証明書の形式](#)
- [中間証明書](#)
- [キーのタイプ](#)
- [プライベートキー](#)
- [アクセス許可](#)
- [証明書キーのサイズ](#)
- [サポートされている証明書のタイプ](#)
- [証明書の有効期限切れと更新](#)
- [CloudFront デイストリビューションと証明書のドメイン名](#)
- [SSL/TLS プロトコルの最小バージョン](#)
- [サポートされる HTTP バージョン](#)

証明書の発行者

[AWS Certificate Manager \(ACM\)](#) で発行された証明書を使用することをお勧めします。ACM からの証明書の取得の詳細については、[AWS Certificate Manager ユーザーガイド](#)を参照してください。で ACM 証明書を使用するには CloudFront、米国東部 (バージニア北部) リージョン () で証明書をリクエスト (またはインポート) していることを確認してください。us-east-1

CloudFront は Mozilla と同じ認証機関 (CAs) をサポートしているため、ACM を使用しない場合は、[Mozilla に付属する CA 証明書リストの CA によって発行された証明書](#)を使用してください。証明書の取得とインストール方法については、使用している HTTP サーバーソフトウェアのドキュメントおよび CA のドキュメントを参照してください。

AWS Certificate Manager の場合は AWS リージョン

AWS Certificate Manager (ACM) の証明書を使用してビューワーと の間で HTTPS を必須にするには CloudFront、米国東部 (バージニア北部) リージョン () で証明書をリクエスト (またはインポート) していることを確認してください。us-east-1

CloudFront とオリジンの間で HTTPS を必須にする場合、Elastic Load Balancing のロードバランサーをオリジンとして使用している場合は、任意の で証明書をリクエストまたはインポートできます AWS リージョン。

証明書の形式

公開証明書は X.509 PEM 形式で作成されている必要があります。これは、AWS Certificate Manager を使用する場合のデフォルトの形式です。

中間証明書

サードパーティー認定権限 (CA) を使用している場合、.pem ファイルには、ドメインの証明書の署名者である CA の証明書から始めて、証明書チェーン内のすべての中間証明書を含めます。通常は、適切なチェーン順で中間証明書とルート証明書を並べたファイルが CA のウェブサイトに用意されています。

Important

ルート証明書、信頼パス内に存在しない中間証明書、CA の公開キー証明書は含めないでください。

例を示します。

```
-----BEGIN CERTIFICATE-----  
Intermediate certificate 2  
-----END CERTIFICATE-----  
-----BEGIN CERTIFICATE-----  
Intermediate certificate 1  
-----END CERTIFICATE-----
```

キーのタイプ

CloudFront は、RSA および ECDSA パブリック/プライベートキーペアをサポートします。

CloudFront は、RSA 証明書と ECDSA 証明書を使用して、ビューワーとオリジンの両方への HTTPS 接続をサポートします。[AWS Certificate Manager \(ACM\)](#) を使用すると、RSA または ECDSA 証明書をリクエストおよびインポートし、デイス CloudFront トリビューションに関連付けることができます。

HTTPS 接続でネゴシエート CloudFront できる、でサポートされている RSA および ECDSA 暗号のリストについては、[the section called “ビューワーと の間でサポートされているプロトコルと暗号 CloudFront”](#)「」および「」を参照してください[the section called “とオリジン間でサポートされているプロトコル CloudFront と暗号”](#)。

プライベートキー

サードパーティー認証機関 (CA) の証明書を使用している場合、以下の点に注意してください。

- プライベートキーが証明書のパブリックキーと一致している。
- プライベートキーは、PEM 形式でなければなりません。
- プライベートキーはパスワードで暗号化できません。

AWS Certificate Manager (ACM) が証明書を提供した場合、ACM はプライベートキーをリリースしません。プライベートキーは、ACM に統合された AWS のサービスによる使用のために、ACM に保存されます。

アクセス許可

SSL/TLS 証明書を使用およびインポートするアクセス許可が必要です。AWS Certificate Manager (ACM) を使用している場合は、AWS Identity and Access Management アクセス許可を使用して証明書へのアクセスを制限することをお勧めします。詳細については、『AWS Certificate Manager ユーザーガイド』の「[Identity and Access Management](#)」を参照してください。

証明書キーのサイズ

が CloudFront サポートする証明書キーのサイズは、キーと証明書のタイプによって異なります。

RSA 証明書の場合:

CloudFront は、1024 ビット、2048 ビット、3072 ビット、4096 ビットの RSA キーをサポートします。で使用する RSA 証明書の最大キー長 CloudFront は 4096 ビットです。

ACM は、最大 2048 ビットのキーで RSA 証明書を発行することに注意してください。3072 ビットまたは 4096 ビットの RSA 証明書を使用するには、証明書を外部から取得して ACM にインポートする必要があります。その後、で使えるようになります CloudFront。

RSA キーのサイズを確認する方法については、「[SSL/TLS RSA 証明書内のパブリック \(公開\) キーのサイズの確認](#)」を参照してください。

ECDSA 証明書の場合:

CloudFront は 256 ビットキーをサポートします。ACM で ECDSA 証明書を使用してビューワーとの間で HTTPS を必須にするには CloudFront、素数 256v1 楕円曲線を使用します。

サポートされている証明書のタイプ

CloudFront は、信頼できる認証機関によって発行されたすべてのタイプの証明書をサポートします。

証明書の有効期限切れと更新

サードパーティー認定権限 (CA) から取得した証明書を使用する場合、証明書の有効期限切れをモニタリングし、AWS Certificate Manager (ACM) にインポートする、または AWS Identity and Access Management 証明書ストアにアップロードする証明書を有効期限前に更新する必要があります。

ACM が提供する証明書を使用している場合、証明書の更新は ACM によって管理されます。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。

CloudFront デистриビューションと証明書のドメイン名

カスタムオリジンを使用する場合、オリジンの SSL/TLS 証明書の共通名フィールドにドメイン名が含まれ、さらにサブジェクト代替名フィールドにもドメイン名がいくつか含まれることがあります。(は証明書ドメイン名でワイルドカード文字 CloudFront をサポートします)。

証明書のドメイン名のうち 1 つは、オリジンドメイン名に指定したドメイン名と一致する必要があります。ドメイン名が一致しない場合、は HTTP ステータスコードをビューワー 502 (Bad Gateway) に CloudFront 返します。

Important

デистриビューションに代替ドメイン名を追加すると、は、その代替ドメイン名がアタッチした証明書の対象 CloudFront になっていることを確認します。証明書は、証明書のサブジェクト代替名 (SAN) フィールドの代替ドメイン名をカバーする必要があります。つまり、SAN フィールドは、代替ドメイン名と完全に一致するか、追加する代替ドメイン名の同じレベルにワイルドカードが含まれている必要があります。

詳細については、「[代替ドメイン名を使用するための要件](#)」を参照してください。

SSL/TLS プロトコルの最小バージョン

専用 IP アドレスを使用している場合は、セキュリティポリシー CloudFront を選択して、ビューワーと間の接続に SSL/TLS プロトコルの最小バージョンを設定します。

詳細については、トピック「[デистриビューションを作成または更新する場合に指定する値](#)」の「[セキュリティポリシー](#)」を参照してください。

サポートされる HTTP バージョン

複数の証明書を複数の CloudFront ディストリビューションに関連付ける場合、証明書に関連付けられているすべてのディストリビューションは、に対して同じオプションを使用する必要があります [サポートされる HTTP バージョン](#)。このオプションは、CloudFront ディストリビューションを作成または更新するときに指定します。

での SSL/TLS 証明書の使用に関するクォータ CloudFront (ビューワーと間の HTTPS CloudFront のみ)

CloudFront で SSL/TLS 証明書を使用する際には、次のクォータに注意してください。これらのクォータは、AWS Certificate Manager (ACM) を使用してプロビジョニングした SSL/TLS 証明書、ACM にインポートした証明書、またはビューワーと間の HTTPS 通信のために IAM 証明書ストアにアップロードした証明書にのみ適用されます CloudFront。

CloudFront ディストリビューションあたりの証明書の最大数

各 CloudFront ディストリビューションに関連付けることができる SSL/TLS 証明書は最大 1 個です。

ACM へのインポートまたは IAM 証明書ストアへのアップロードが可能な証明書の最大数

サードパーティー認証機関から SSL/TLS 証明書を取得した場合、次のいずれかの場所に証明書を保存する必要があります。

- AWS Certificate Manager - ACM 証明書数に対する現在のクォータについては、AWS Certificate Manager ユーザーガイドの「[クォータ](#)」を参照してください。一覧表示されているクォータは、ACM を使用してプロビジョニングした証明書や ACM にインポートした証明書を含ま合計です。
- IAM 証明書ストア - AWS アカウントの IAM 証明書ストアにアップロードできる証明書の数の現在のクォータ (以前は制限と呼ばれていました) については、IAM ユーザーガイドの「[IAM および STS の制限](#)」を参照してください。[AWS Management Console](#) では、より高いクォータをリクエストできます。

AWS アカウントごとの証明書の最大数 (専用 IP アドレスのみ)

専用 IP アドレスを使用して HTTPS リクエストを供給する場合は、以下の点に注意してください。

- デフォルトでは、は AWS アカウントで 2 つの証明書を使用するアクセス許可 CloudFront を付与します。1 つは日常的な使用用、もう 1 つは複数のディストリビューションの証明書を更新する必要がある場合に使用します。

- AWSアカウントに2つ以上のカスタム SSL/TLS 証明書が必要な場合は、[サポートセンター](#)にアクセスしてケースを作成します。使用するためのアクセス許可が必要な証明書の数と状況をリクエストに記載してください。できる限り早くアカウントを更新します。

異なるアカウントを使用して作成されたディストリビューションに同じ証明書を使用する AWS

サードパーティー CA を使用しており、異なる AWSアカウントを使用して作成された複数の CloudFront ディストリビューションで同じ証明書を使用する場合は、ACM に証明書をインポートするか、AWSアカウントごとに1回 IAM 証明書ストアにアップロードする必要があります。

ACM が提供する証明書を使用している場合は、別のAWSアカウントで作成された証明書を使用する CloudFront ように を設定することはできません。

CloudFront と他の AWSのサービスに同じ証明書を使用する

Comodo、DigiCert、Symantec などの信頼された認証機関から証明書を購入した場合は、CloudFront と他の AWSサービスに同じ証明書を使用できます。ACM に証明書をインポートする場合、一度インポートするだけで複数の AWS サービスで使用できます。

ACM が提供した証明書を使用している場合、証明書は ACM に格納されています。

複数の CloudFront ディストリビューションに同じ証明書を使用する

HTTPS リクエストに対応するために使用している一部またはすべての CloudFront ディストリビューションで同じ証明書を使用できます。次の点に注意してください。

- 専用 IP アドレスを使用したリクエストの処理と SNI を使用したリクエストの処理の両方に同じ証明書を使用できます。
- 各ディストリビューションに1個のみ証明書を関連付けることができます。
- 各ディストリビューションには、証明書の共通名フィールドまたはサブジェクト代替名フィールドにも表示される1つ以上の代替ドメイン名を含める必要があります。
- 専用 IP アドレスを使用して HTTPS リクエストを処理していて、同じAWSアカウントを使用してすべてのディストリビューションを作成した場合は、すべてのディストリビューションに同じ証明書を使用することでコストを大幅に削減できます。はディストリビューションごとではなく、証明書ごとに CloudFront 課金します。

たとえば、同じ AWS アカウントを使用して3つのディストリビューションを作成し、3つのディストリビューションすべてに対して同じ証明書を使用するとします。専用 IP アドレスの使用に対する1つの料金のみが発生します。

ただし、専用 IP アドレスを使用して HTTPS リクエストを処理し、同じ証明書を使用して異なる AWS アカウントに CloudFront ディストリビューションを作成する場合、各アカウントには専用 IP アドレスの使用料金が請求されます。たとえば、3 つの異なる AWS アカウントを使用して 3 つのディストリビューションを作成し、3 つのすべてのディストリビューションに同じ証明書を使用する場合、専用 IP アドレスを使用するための料金全額が各アカウントに課金されます。

代替ドメイン名と HTTPS の設定

ファイルの URLs で代替ドメイン名を使用し、ビューワーとの間で HTTPS を使用するには CloudFront、該当する手順を実行します。

トピック

- [SSL/TLS 証明書を取得する](#)
- [SSL/TLS 証明書をインポートする](#)
- [CloudFront ディストリビューションの更新](#)

SSL/TLS 証明書を取得する

SSL/TLS 証明書を取得します (まだ取得していない場合)。詳細については、該当するドキュメントを参照してください。

- AWS Certificate Manager (ACM) が提供する証明書を使用するには、[AWS Certificate Manager ユーザーガイド](#)を参照してください。その後、[CloudFront ディストリビューションの更新](#)に進みます。

Note

ACM を使用して AWS が管理するリソースに SSL/TLS 証明書のプロビジョニング、管理、デプロイを行うことをお勧めします。米国東部 (バージニア北部) リージョンで ACM 証明書をリクエストする必要があります。

- サードパーティー認証機関 (CA) から証明書を取得するには、認証機関から提供されたドキュメントを参照してください。証明書を取得した後、次の手順に進んでください。

SSL/TLS 証明書をインポートする

サードパーティー認証機関から証明書を取得した場合、証明書を ACM にインポートするか、IAM 証明書ストアにアップロードします。

ACM (推奨)

ACM では、ACM コンソールから、またはプログラムによってサードパーティー証明書をインポートできます。ACM への証明書のインポートについては、AWS Certificate Manager ユーザーガイドの「[AWS Certificate Manager への証明書のインポート](#)」を参照してください。米国東部 (バージニア北部) リージョンで証明書をインポートする必要があります。

IAM 証明書ストア

(非推奨) 次の AWS CLI コマンドを使用してサードパーティー証明書を IAM 証明書ストアにアップロードします。

```
aws iam upload-server-certificate \  
  --server-certificate-name CertificateName \  
  --certificate-body file://public_key_certificate_file \  
  --private-key file://privatekey.pem \  
  --certificate-chain file://certificate_chain_file \  
  --path /cloudfront/path/
```

次の点に注意してください。

- AWS アカウント – デイス CloudFront トリビューションの作成に使用したのと同じAWSアカウントを使用して、IAM 証明書ストアに証明書をアップロードする必要があります。
- --path パラメータ – 証明書を IAM にアップロードする場合、--path パラメータ (証明書のパス) の値が /cloudfront/ で始まる必要があります (/cloudfront/production/、/cloudfront/test/ など)。パスは / で終わる必要があります。
- 既存の証明書 – --server-certificate-name および --path パラメータには、既存の証明書に関連付けられている値とは異なる値を指定する必要があります。
- CloudFront コンソールの使用 – 例えばAWS CLI、で --server-certificate-nameパラメータに指定した値はmyServerCertificate、コンソールの SSL 証明書リスト CloudFront に表示されます。
- CloudFront API の使用 – がAWS CLI返す英数字の文字列をメモします。例えば、で返すAS1A2M3P4L5E67SIIXR3J。これは、IAMCertificateId エlementに指定する値です。CLI から返される IAM ARN を書き留めておく必要はありません。

AWS CLI の詳細については、[AWS Command Line Interface ユーザーガイド](#)および [AWS CLI コマンドリファレンス](#)を参照してください。

CloudFront デистриビューションの更新

デистриビューションの設定を更新するには、以下の手順を実行します。

代替ドメイン名の CloudFront デистриビューションを設定するには

1. にサインインAWS Management Consoleし、 で CloudFront コンソールを開きます<https://console.aws.amazon.com/cloudfront/v4/home>。
2. 更新するデистриビューションの ID を選択します。
3. [General] タブで、[Edit] を選択します。
4. 以下の値を更新します。

代替ドメイン名 (CNAME)

該当する代替ドメイン名を追加します。ドメイン名をコンマで区切るか、新しい行にドメイン名を 1 つずつ入力します。

SSL 証明書

[Custom SSL Certificate (独自 SSL 証明書)] を選択して、リストから証明書を選択します。

100 個までの証明書がここに一覧表示されます。100 個を超える証明書があり、追加する証明書が表示されない場合には、フィールドに証明書の ARN を入力して選択できます。

IAM 証明書ストアに証明書をアップロードしたが、一覧に表示されず、フィールドに入力しても選択できない場合には、「[SSL/TLS 証明書をインポートする](#)」の手順を参照して、証明書を正しくアップロードしたかを確認します。

Important

SSL/TLS 証明書を CloudFront デистриビューションに関連付けた後、すべてのデистриビューションから証明書を削除し、デистриビューションのステータスが **デプロイ済み** に変わるまで、ACM または IAM 証明書ストアから証明書を削除しないでください。

Clients Supported (サポートされるクライアント)

適用可能なオプションを選択します。

- すべてのクライアント: 専用 IP アドレスを使用して HTTPS コンテンツ CloudFront を提供します。このオプションを選択した場合、有効になっているディストリビューションに SSL/TLS 証明書を関連付けると、追加料金がかかります。詳細については、[「Amazon の CloudFront 料金」](#) を参照してください。
- [Only Clients that Support Server Name Indication (SNI)]: 古いブラウザなど SNI をサポートしないクライアントは、別の方法を使用してコンテンツにアクセスする必要があります。

詳細については、[「が HTTPS リクエスト CloudFront を処理する方法の選択」](#) を参照してください。

5. [Yes, Edit (はい、編集します)] を選択します。
6. ビューワーとの間で HTTPS を要求する CloudFront ように を設定します CloudFront.
 - a. [Behaviors (動作)] タブで、更新するキャッシュ動作を選択して、[Edit (編集)] を選択します。
 - b. [Viewer Protocol Policy] として次のいずれかの値を指定します。

Redirect HTTP to HTTPS

ビューワーは両方のプロトコルを使用できますが、HTTP リクエストは自動的に HTTPS リクエストにリダイレクトされます。CloudFront は新しい HTTPS URL とともに HTTP ステータスコード 301 (Moved Permanently) を返します。次に、ビューワーは HTTPS URL CloudFront を使用して にリクエストを再送信します。

Important

CloudFront は、DELETE、OPTIONS、PATCH、POST または PUT リクエストを HTTP から HTTPS にリダイレクトしません。HTTPS にリダイレクトするようにキャッシュ動作を設定すると、は HTTP ステータスコードでそのキャッシュ動作に対する HTTP DELETE、OPTIONS、PATCH、POST、または PUT リクエスト CloudFront に対応します 403 (Forbidden)。

ビューワーが HTTPS リクエストにリダイレクトされる HTTP リクエストを行うと、は両方のリクエストに対して CloudFront 課金します。HTTP リクエストの場合、リクエストの料金と、CloudFront がビューワーに返すヘッダーの料金のみが課金されます。HTTPS リクエストの場合、リクエストの料金と、ファイルが返すヘッダーとオブジェクトの料金が課金されます。

HTTPS Only

ビューワーは、HTTPS を使用している場合にのみ、コンテンツにアクセスできます。ビューワーが HTTPS リクエストではなく HTTP リクエストを送信する403 (Forbidden)と、は HTTP ステータスコードを CloudFront 返し、ファイルは返しません。

- c. [Yes, Edit (はい、編集します)] を選択します。
 - d. ビューワーと CloudFront との間で HTTPS を必須にする追加のキャッシュ動作ごとに、ステップ a から c を繰り返します。
7. 本番環境で更新された情報を使用する前に、次を確認してください。
- ビューワーに HTTPS の使用が必要とされるリクエストにのみ、各キャッシュ動作のパスパターンが適用されている。
 - CloudFront が評価する順番にキャッシュ動作がリストされている。詳細については、「[パスパターン](#)」を参照してください。
 - キャッシュ動作は、リクエストを正しいオリジンにルーティングします。

SSL/TLS RSA 証明書内のパブリック (公開) キーのサイズの確認

CloudFront 代替ドメイン名と HTTPS を使用する場合、SSL/TLS RSA 証明書のパブリックキーの最大サイズは 4096 ビットです。(これはキーサイズであり、パブリックキー内の文字数ではありません)。証明書AWS Certificate Managerに を使用する場合、ACM はより大きな RSA キーをサポートしますが、でより大きなキーを使用することはできません CloudFront。

RSA パブリック (公開) キーのサイズを確認するには、次の OpenSSL コマンドを実行できます。

```
openssl x509 -in path and filename of SSL/TLS certificate -text -noout
```

各パラメータの意味は次のとおりです。

- `-in` は、SSL/TLS RSA 証明書のパスとファイル名を指定します。
- `-text` によって、OpenSSL において、RSA パブリック (公開) キーの長さがビット単位で表示されます。
- `-noout`: OpenSSL において、パブリックキーが非表示になります。

出力例:

```
Public-Key: (2048 bit)
```

SSL/TLS 証明書のクォータの引き上げ

[AWS Certificate Manager](#) にインポートしたり、[AWS Identity and Access Management](#) にアップロードしたりできる SSL/TLS 証明書の数にはクォータがあります。専用 IP アドレスを使用して HTTPS リクエストを処理する CloudFront ように を設定する場合、AWS アカウントで使用できる SSL/TLS 証明書の数にもクォータがあります。ただし、これらのクォータ数を増やすようにリクエストできます。

トピック

- [ACM にインポートできる証明書数](#)
- [IAM にアップロードできる証明書数](#)
- [専用 IP アドレスで使用できる証明書](#)

ACM にインポートできる証明書数

ACM にインポートできる証明書の数のクォータについては、AWS Certificate Manager ユーザーガイドの「[クォータ](#)」を参照してください。

クォータの引き上げをリクエストするには、サポートセンターコンソールで[ケースを作成してください](#)。次の値を指定します。

- [Service Limit Increase] のデフォルト値を受け入れます。
- [Limit type] で、[Certificate Manager] を選択します。
- [Region] で、AWS 証明書をインポートするリージョンを指定します。
- [Limit] で、[(ACM) Number of ACM Certificates] を選択します。

次にフォームの残りに入力して送信します。

IAM にアップロードできる証明書数

IAM にアップロードできる証明書の数のクォータについては、IAM ユーザーガイドの「[IAM と STS の制限](#)」を参照してください。

クォータの引き上げをリクエストするには、サポートセンターコンソールで[ケースを作成してください](#)。次の値を指定します。

- [Service Limit Increase] のデフォルト値を受け入れます。
- [Limit type] で、[Certificate Manager] を選択します。
- [Region] で、AWS 証明書をインポートするリージョンを指定します。
- [Limit] で、[Server Certificate Limit (IAM) (サーバー証明書の制限 (IAM))] を選択します。

次にフォームの残りに入力して送信します。

専用 IP アドレスで使用できる証明書

専用 IP アドレスを使用して HTTPS リクエストを処理するときに各 AWS アカウントに使用できる SSL 証明書の数のクォータについては、「[SSL 証明書のクォータ](#)」を参照してください。

クォータの引き上げをリクエストするには、サポートセンターコンソールで[ケースを作成してください](#)。次の値を指定します。

- [Service Limit Increase] のデフォルト値を受け入れます。
- 制限タイプ で、CloudFront ディストリビューション を選択します。
- [Limit Type] で、[Dedicated IP SSL Certificate Limit per Account (アカウントごとの専用 IP SSL 証明書制限)] を選択します。

次にフォームの残りに入力して送信します。

SSL/TLS 証明書の更新

AWS Certificate Manager (ACM) が提供する証明書を使用する場合、SSL/TLS 証明書を更新する必要はありません。証明書の更新は、ACM によって行われます。詳細については、AWS Certificate Manager ユーザーガイドの「[マネージド更新](#)」を参照してください。

Note

ACM では、サードパーティー認証機関から入手して ACM にインポートした証明書の更新が管理されません。

サードパーティー認証機関を利用しており、証明書を ACM にインポートした場合 (推奨) または IAM 証明書ストアにアップロードした場合、証明書の交換が必要になることがあります。たとえば、証明書の有効期限が近づいている場合、証明書を交換する必要があります。

Important

専用 IP アドレスを使用して HTTPS リクエストを処理する CloudFront ように を設定した場合、証明書のローテーション中に 1 つ以上の追加証明書の使用に対して、按分計算された追加料金が発生する可能性があります。ディストリビューションの更新を迅速に行って、追加料金を最低限にすることをお勧めします。

証明書をローテーションするには、以下の手順を実行します。ビューワーは、プロセスが完了した後だけでなく、証明書を更新している間もコンテンツにアクセスし続けることができます。

SSL/TLS 証明書をローテーションするには

1. [SSL/TLS 証明書のクォータの引き上げ](#)、追加の SSL 証明書を使用するためにアクセス権限が必要かどうかを調べます。その場合、アクセス権限をリクエストし、ステップ 2 を続ける前にアクセス権限が付与されるまで待ってください。
2. 新しい証明書を ACM にインポートするか、IAM にアップロードします。詳細については、「Amazon CloudFront デベロッパーガイド」の「[SSL/TLS 証明書のインポート](#)」を参照してください。
3. ディストリビューションを一度に 1 つずつ、新しい証明書を使用できるように更新します。詳細については、「[Amazon デベロッパーガイド](#)」の CloudFront 「[ディストリビューションの一覧表示、表示、更新](#)」を参照してください。 CloudFront
4. (オプション) すべての CloudFront ディストリビューションを更新したら、ACM または IAM から古い証明書を削除できます。

⚠ Important

すべてのディストリビューションから SSL/TLS 証明書を削除し、更新されたディストリビューションのステータスが [Deployed] に変わるまで、SSL/TLS 証明書を削除しないでください。

カスタム SSL/TLS 証明書からデフォルト CloudFront 証明書に戻す

ビューワーとの間で HTTPS を使用する CloudFront ように を設定し CloudFront、カスタム SSL/TLS 証明書を使用する CloudFront ように を設定した場合は、デフォルトの CloudFront SSL/TLS 証明書を使用するように設定を変更できます。プロセスは、コンテンツの配信にディストリビューションを使用しているかどうかによって異なります。

- ディストリビューションを使用してコンテンツを配信したことがない場合は、単に構成を変更できます。詳細については、「[ディストリビューションの更新](#)」を参照してください。
- ディストリビューションを使用してコンテンツを配信したことがある場合は、新しい CloudFront ディストリビューションを作成し、コンテンツが使用できない時間を減らすかゼロにするために、ファイルの URL を変更する必要があります。そのためには、以下の手順を実行します。

デフォルトの CloudFront 証明書に戻すには

1. 必要な設定で新しい CloudFront ディストリビューションを作成します。[SSL Certificate (SSL 証明書)] には、[Default CloudFront Certificate (*.cloudfront.net) (デフォルト CloudFront 証明書 (*.cloudfront.net))] を選択します。

詳細については、「[ディストリビューションを作成するためのステップ \(概要\)](#)」を参照してください。

2. を使用して配信するファイルの場合は CloudFront、アプリケーションの URLs を更新して、 が新しいディストリビューションに CloudFront 割り当てたドメイン名を使用します。例えば、`https://www.example.com/images/logo.png` を `https://d1111111abcdef8.cloudfront.net/images/logo.png` に変更します。
3. カスタム SSL/TLS 証明書に関連付けられているディストリビューションを削除するか、ディストリビューションを更新して SSL 証明書の値をデフォルト CloudFront 証明書 (*.cloudfront.net) に変更します。詳細については、「[ディストリビューションの更新](#)」を参照してください。

⚠ Important

このステップが完了するまで、AWS は独自 SSL/TLS 証明書を使用する料金を課金し続けます。

4. (オプション) カスタム SSL/TLS 証明書を削除します。
 - a. AWS CLI コマンド `list-server-certificates` を実行して、削除する証明書の証明書 ID を取得します。詳細については、AWS CLI コマンドリファレンスの [list-server-certificates](#) を参照してください。
 - b. AWS CLI コマンド `delete-signing-certificate` を実行して、証明書を削除します。詳細については、AWS CLI コマンドリファレンスの [delete-signing-certificate](#) を参照してください。

独自 SSL/TLS 証明書を専用 IP アドレスから SNI に切り替える

専用 IP アドレスでカスタム SSL/TLS 証明書を使用する CloudFront ように を設定した場合は、代わりに SNI でカスタム SSL/TLS 証明書を使用するように切り替えて、専用 IP アドレスに関連付けられている料金を排除できます。以下の手順でその方法を説明します。

⚠ Important

CloudFront この設定に対するこの更新は、SNI をサポートするビューワーには影響しません。ビューワーは、変更前と変更後、および変更が CloudFront エッジロケーションに反映されている間、コンテンツにアクセスできます。SNI をサポートしていないビューワーは、変更後、コンテンツにアクセスできなくなります。詳細については、「[が HTTPS リクエスト CloudFront を処理する方法の選択](#)」を参照してください。

独自 SSL/TLS 証明書を専用 IP アドレスから SNI に切り替えるには

1. にサインインAWS Management Consoleし、 で CloudFront コンソールを開きます<https://console.aws.amazon.com/cloudfront/v4/home>。
2. 表示または更新するディストリビューションの ID を選択します。
3. [Distribution Settings] を選択します。
4. [General] タブで、[Edit] を選択します。

5. [Custom SSL Client Support (独自 SSL クライアントサポート)] の設定を [Only Clients that Support Server Name Indication (SNI) (Server Name Indication (SNI) をサポートするクライアントのみ)] に変更します。
6. [Yes, Edit (はい、編集します)] を選択します。

署名付き URL と署名付き Cookie を使用したプライベートコンテンツの提供

インターネット経由でコンテンツを配信する多くの企業が、選ばれたユーザー (料金を支払っているユーザーなど) のドキュメント、ビジネスデータ、メディアストリーム、またはコンテンツに対して、アクセスを制限する必要があると考えています。を使用してこのプライベートコンテンツを安全に配信するには CloudFront、次の操作を行います。

- 特別な CloudFront 署名URLs または署名付き Cookie を使用してプライベートコンテンツにアクセスすることをユーザーに要求します。
- オリジンサーバー (Amazon S3 CloudFront URLs やプライベート HTTP サーバーなど) 上のコンテンツに直接アクセスする URLs ではなく、URL を使用してコンテンツにアクセスすることをユーザーに要求します。CloudFront URLsにする必要はありませんが、ユーザーが署名付き URLs または署名付き Cookie で指定した制限を回避できないようにすることをお勧めします。

トピック

- [プライベートコンテンツ提供の概要](#)
- [プライベートコンテンツ提供のためのタスクリスト](#)
- [署名付き URL と署名付き Cookie を作成できる署名者の指定](#)
- [署名付き URL と署名付き Cookie の選択](#)
- [署名付き URL の使用](#)
- [署名付き Cookie の使用](#)
- [Linux コマンドおよび OpenSSL を使用した Base64 エンコードおよび暗号化](#)
- [署名付き URL の署名を作成するためのコード例](#)

プライベートコンテンツ提供の概要

プライベートコンテンツへのユーザーアクセスは 2 つの方法を使用して制御可能です。

- [CloudFront キャッシュ内のファイルへのアクセスを制限します。](#)
- 次のいずれかを実行して、オリジン内のファイルへのアクセスを制限します。
 - [Amazon S3 バケットのオリジンアクセスコントロール \(OAC\) のセットアップ](#)
 - [プライベート HTTP サーバー \(カスタムオリジン\) のカスタムヘッダーの設定](#)

キャッシュ内の CloudFront ファイルへのアクセスの制限

ユーザーが署名URLs または署名付き Cookie を使用してファイルにアクセスするように要求 CloudFront するようにを設定できます。次に、署名付き URL を作成して認証されたユーザーに配信するか、認証されたユーザーの署名付き Cookie を設定する Set-Cookie ヘッダーを送信するアプリケーションを開発します (少数のファイルへの長期的なアクセスを数人のユーザーに付与するために、署名付き URL を手動で作成することもできます)。

ファイルへのアクセスを制御するための署名付き URL または署名付き Cookie を作成するときに、次の制限を指定できます。

- 最終日時。この日時以降、URL が有効ではなくなります。
- (オプション) URL が有効になる日時。
- (オプション) コンテンツへのアクセスに使用可能なコンピュータの IP アドレスまたはアドレス範囲。

署名付き URL または署名付き Cookie では、パブリックとプライベートのキーペアのプライベートキーを使用して、一部がハッシュ化され、署名が行われます。誰かが署名付き URL または署名付き Cookie を使用してファイルにアクセスすると、は URL または Cookie の署名付き部分と署名なし部分 CloudFront を比較します。一致しない場合は、ファイルを提供し CloudFront ません。

URLs または Cookie の署名には RSA-SHA1 CloudFront を使用する必要があります。他のアルゴリズムは使用できません。

Amazon S3 バケット内のファイルへのアクセス制限

オプションで、Amazon S3 バケット内のコンテンツを保護して、ユーザーが指定された CloudFront ディストリビューションを介してアクセスできても、Amazon S3 URL を使用して直接アクセスできないようにすることができます。URLs これにより、アクセスを制限するコンテンツを取得するために、誰かが Amazon S3 URL をバイパス CloudFront して使用できなくなります。署名付き URL を使用するためにこの手順を実行する必要はありませんが、推奨します。

URL を介してコンテンツにアクセスすることをユーザーに要求するには CloudFront URLs、次のタスクを実行します。

- S3 バケット内のファイルを読み取るためのオ CloudFront リジンアクセスコントロール許可を付与します。
- オリジンアクセスコントロールを作成し、CloudFront デイストリビューションに関連付けます。
- Amazon S3 URL を使用してファイルを読み取るためのアクセス許可を、他のすべてのユーザーから削除します。

詳しくは、「[the section called “Amazon S3 オリジンへのアクセスの制限”](#)」を参照してください。

カスタムオリジン上のファイルへのアクセス制限

カスタムオリジンを使用する場合は、カスタムヘッダーをオプションで設定して、アクセスを制限できます。CloudFront がカスタムオリジンからファイルを取得するには、標準の HTTP (または HTTPS) リクエスト CloudFront を使用してファイルにアクセスする必要があります。ただし、カスタムヘッダーを使用すると、コンテンツへのアクセスをさらに制限して、ユーザーが直接ではなく CloudFront を介してのみアクセスできるようにすることができます。署名付き URL を使用するためにこの手順を実行する必要はありませんが、推奨します。

ユーザーが を介してコンテンツにアクセスできるようにするには CloudFront、デイストリビューションで次の設定を変更します。

オリジンのカスタムヘッダー

カスタムヘッダー CloudFront をオリジンに転送するように を設定します。[オリジンリクエスト CloudFront にカスタムヘッダーを追加するための の設定](#) を参照してください。

ビューワープロトコルポリシー

ビューワーが CloudFront にアクセスするのに HTTPS を使用しなければならないようにデイストリビューションを設定します。[ビューワープロトコルポリシー](#) を参照してください。

オリジンプロトコルポリシー

ビューワーと同じプロトコルを使用してリクエストをオリジンに転送 CloudFront するよう に要求するようにデイストリビューションを設定します。[プロトコル \(カスタムオリジンのみ\)](#) を参照してください。

これらの変更を行ったら、送信 CloudFront するように設定したカスタムヘッダーを含むリクエストのみを受け入れるように、カスタムオリジンでアプリケーションを更新します。

ビューワープロトコルポリシーとオリジンプロトコルポリシーの組み合わせにより、カスタムヘッダーが転送中に暗号化されます。ただし、がオリジン CloudFront に転送するカスタムヘッダーをローテーションするには、定期的に次の操作を行うことをお勧めします。

1. デイス CloudFront トリビューションを更新して、カスタムオリジンへの新しいヘッダーの転送を開始します。
2. リクエストが から送信されていることの確認として、新しいヘッダーを受け入れるようにアプリケーションを更新します CloudFront。
3. 置き換えるヘッダーがリクエストに含まれなくなったら、リクエストが から送信されたことを確認するために、古いヘッダーを受け入れないようにアプリケーションを更新します CloudFront。

プライベートコンテンツ提供のためのタスクリスト

プライベートコンテンツを配信 CloudFront するように を設定するには、次のタスクを実行します。

1. (オプションですが推奨) ユーザーは を介してのみコンテンツにアクセスする必要があります CloudFront。使用方法は、Amazon S3 を使用するかカスタムオリジンを使用するかによって異なります。
 - Amazon S3 – 「[the section called “Amazon S3 オリジンへのアクセスの制限”](#)」を参照してください。
 - カスタムオリジン – 「[カスタムオリジン上のファイルへのアクセス制限](#)」を参照してください。

カスタムオリジンには、Amazon EC2、ウェブサイトエンドポイントとして設定されている Amazon S3 バケット、Elastic Load Balancing、独自の HTTP ウェブサーバーなどがあります。

2. 署名付き URL または署名付き Cookie の作成に使用する信頼されたキーグループまたは信頼された署名者を指定します。信頼されたキーグループを使用することをお勧めします。詳細については、「[署名付き URL と署名付き Cookie を作成できる署名者の指定](#)」を参照してください。
3. 署名付き URL または署名付き Cookie を設定する Set-Cookie ヘッダーを使用した、承認されたユーザーからのリクエストに対応するアプリケーションを記述します。以下のいずれかのトピックの手順に従ってください。

- [署名付き URL の使用](#)

- [署名付き Cookie の使用](#)

使用方法が明確でない場合は、「[署名付き URL と署名付き Cookie の選択](#)」を参照してください。

署名付き URL と署名付き Cookie を作成できる署名者の指定

トピック

- [信頼されたキーグループ \(推奨\) と AWS アカウントのいずれかの選択](#)
- [署名者のキーペアの作成](#)
- [プライベートキーの形式の変更 \(.NET および Java のみ\)](#)
- [ディストリビューションへの署名者の追加](#)
- [キーペアの更新](#)

署名付き URL または署名付き Cookie を作成するには、署名者が必要です。署名者は、で作成した信頼できるキーグループ CloudFront、または CloudFront キーペアを含む AWS アカウントのいずれかです。署名付き URL と署名付き Cookie が有効な信頼されたキーグループを使用することをお勧めします。詳細については、「[信頼されたキーグループ \(推奨\) と AWS アカウントのいずれかの選択](#)」を参照してください。

署名者には 2 つの目的があります。

- 署名者をディストリビューションに追加するとすぐに、は、ビューワーが署名付き URLs または署名付き Cookie を使用してファイルにアクセスするように要求し CloudFront 始めます。
- 署名付き URL または署名付き Cookie を作成するときは、署名者のキーペアのプライベートキーを使用して URL または Cookie に署名します。制限付きファイルをリクエストすると、は URL または Cookie 内の署名を署名なし URL または Cookie CloudFront と比較し、改ざんされていないことを確認します。CloudFront また、は URL または Cookie が有効であること、例えば有効期限が切れていないことを確認します。

署名者を指定するときは、署名者をキャッシュ動作に追加することにより、署名付き URL または署名付き Cookie を必要とするファイルも間接的に指定します。ディストリビューションのキャッシュ動作が 1 つしかない場合、ビューワーはディストリビューション内のファイルへのアクセスに、署名付き URL または署名付き Cookie の使用を求められます。複数のキャッシュ動作を作成して、署名者を一部のキャッシュ動作に追加し、それ以外のキャッシュ動作に追加しなかった場合、ビュー

ワールドワイドウェブの一部のファイルへのアクセスに、署名付き URL または署名付き Cookie の使用を求められ、その他のファイルへのアクセスには求められません。

署名付き URLs または署名付き Cookie の作成を許可する署名者 (プライベートキー) を指定し、署名者を CloudFront ディストリビューションに追加するには、次のタスクを実行します。

1. 署名者として、信頼されたキーグループを使用するか AWS アカウントを使用するかを決定します。信頼されたキーグループを使用することをお勧めします。詳細については、「[信頼されたキーグループ \(推奨\) と AWS アカウントのいずれかの選択](#)」を参照してください。
2. ステップ 1 で選択した署名者に対して、パブリックとプライベートのキーペアを作成します。詳細については、「[署名者のキーペアの作成](#)」を参照してください。
3. 署名付き URL または署名付き Cookie の作成に .NET または Java を使用する場合は、プライベートキーの形式を変更します。詳細については、「[プライベートキーの形式の変更 \(.NET および Java のみ\)](#)」を参照してください。
4. 署名付き URL または署名付き Cookie を作成するディストリビューションで、署名者を指定します。詳細については、「[ディストリビューションへの署名者の追加](#)」を参照してください。

信頼されたキーグループ (推奨) と AWS アカウントのいずれかの選択

署名付き URL または署名付き Cookie を使用するには、署名者が必要です。署名者は、で作成した信頼できるキーグループ CloudFront、または CloudFront キーペアを含む AWS アカウントのいずれかです。以下の理由から、信頼されたキーグループを使用することをお勧めします。

- CloudFront キーグループでは、署名付き URLs と CloudFront 署名付き Cookie のパブリックキーを管理するために、AWS アカウントのルートユーザーを使用する必要はありません。[AWS のベストプラクティス](#)では、必要ないときはルートユーザーを使用しないことをお勧めします。
- CloudFront キーグループを使用すると、CloudFront API を使用してパブリックキー、キーグループ、および信頼された署名者を管理できます。API を使用して、キーの作成とキーの更新を自動化できます。AWS ルートユーザーを使用する場合は、AWS Management Console を使用して CloudFront キーペアを管理する必要があるため、プロセスを自動化することはできません。
- CloudFront API を使用してキーグループを管理できるため、AWS Identity and Access Management (IAM) アクセス許可ポリシーを使用して、さまざまなユーザーが実行できる操作を制限することもできます。例えば、ユーザーにパブリックキーのアップロードを許可し、削除を禁止することができます。または、ユーザーにパブリックキーの削除を許可するが、許可するのは、多要素認証の使用、特定のネットワークからのリクエストの送信、特定の日時範囲内でのリクエストの送信など、特定の条件が満たされた場合に限ることもできます。

- CloudFront キーグループを使用すると、より多くのパブリックキーを CloudFront ディストリビューションに関連付けることができ、パブリックキーの使用方法和管理方法をより柔軟にできます。デフォルトでは、最大 4 つのキーグループを 1 つのディストリビューションに関連付けることができ、キーグループには最大 5 つのパブリックキーを含めることができます。

AWS アカウントのルートユーザーを使用して CloudFront キーペアを管理する場合、AWS アカウントあたり最大 2 つのアクティブな CloudFront キーペアのみを持つことができます。

署名者のキーペアの作成

CloudFront 署名付き URLs または署名付き Cookie の作成に使用する各署名者には、パブリックキーとプライベートキーのペアが必要です。署名者はプライベートキーを使用して URL または Cookie に署名し、パブリックキー CloudFront を使用して署名を検証します。

キーペアの作成方法は、信頼できるキーグループを署名者として使用するか (推奨)、CloudFront キーペアを使用するかによって異なります。詳細については、次のセクションを参照してください。作成するキーペアは、以下の要件を満たしている必要があります。

- SSH-2RSA キーペアである必要があります。
- base64 エンコードされた PEM 形式である必要があります。
- 2048 ビットのキーペアである必要があります。

アプリケーションを保護するために、キーペアを定期的に更新することをお勧めします。詳細については、「[キーペアの更新](#)」を参照してください。

信頼されたキーグループのキーペアを作成する (推奨)

信頼されたキーグループのキーペアを作成するには、以下の手順を実行します。

1. パブリックとプライベートのキーペアを作成します。
2. パブリックキーを にアップロードします CloudFront。
3. パブリックキーを CloudFront キーグループに追加します。

詳細については、次の手順を参照してください。

キーペアを作成するには

Note

以下の手順では、キーペアを作成する方法の一例として OpenSSL を使用します。RSA キーペアを作成する方法は他にも多数あります。

1. 以下のコマンド例では、OpenSSL を使用して 2,048 ビット長の RSA キーペアを生成し、`private_key.pem` という名前のファイルに保存します。

```
openssl genrsa -out private_key.pem 2048
```

2. 生成されるファイルには、パブリックキーとプライベートキーの両方が含まれます。以下のコマンド例では、`private_key.pem` という名前のファイルからパブリックキーを抽出します。

```
openssl rsa -pubout -in private_key.pem -out public_key.pem
```

後で、以下の手順でパブリックキー (`public_key.pem` ファイル内) をアップロードします。

パブリックキーを にアップロードするには CloudFront

1. にサインインAWS Management Consoleし、 で CloudFront コンソールを開きます<https://console.aws.amazon.com/cloudfront/v4/home>。
2. ナビゲーションメニューで、[Public keys (パブリックキー)] を選択します。
3. [Add public key (パブリックキーを追加)] を選択します。
4. [Add public key (パブリックキーの追加)] ウィンドウで、以下の手順を実行します。
 - a. [Key name (キー名)] に、パブリックキーを識別するための名前を入力します。
 - b. [Key value (キー値)] に、パブリックキーを貼り付けます。前の手順のステップに従った場合、パブリックキーは `public_key.pem` という名前のファイルにあります。パブリックキーの内容をコピーして貼り付けるには、以下の手順を実行します。
 - macOS または Linux コマンドラインで `cat` コマンドを次のように使用します。

```
cat public_key.pem
```

そのコマンドの出力をコピーして、[Key value (キー値)] フィールドに貼り付けます。

- メモ帳 (Windows の場合) や TextEdit (macOS の場合) などのプレーンテキストエディタを使用して public_key.pem ファイルを開きます。ファイルの内容をコピーし、[Key value (キー値)] フィールドに貼り付けます。
- c. (オプション) [Comment (コメント)] に、パブリックキーを説明するコメントを追加します。

完了したら、[Add (追加)] を選択します。

5. パブリックキー ID を記録します。この ID は、後で署名付き URL または署名付き Cookie を作成するときに、Key-Pair-Id フィールドの値として使用します。

パブリックキーをキーグループに追加するには

1. で CloudFront コンソールを開きます <https://console.aws.amazon.com/cloudfront/v4/home>。
2. ナビゲーションメニューで、[Key groups (キーグループ)] を選択します。
3. [Add key group (キーグループの追加)] を選択します。
4. [Create key group (キーグループの作成)] ページで、以下の手順を実行します。
 - a. [Key group name (キーグループ名)] に、キーグループを識別するための名前を入力します。
 - b. (オプション) [Comment (コメント)] に、キーグループを説明するコメントを入力します。
 - c. [Public keys (パブリックキー)] で、キーグループに追加するパブリックキーを選択してから、[Add (追加)] を選択します。キーグループに追加するパブリックキーごとに、このステップを繰り返します。
5. [Create key group (キーグループの作成)] を選択します。
6. キーグループ名を記録します。後でこれを使用して、キーグループを CloudFront ディストリビューションのキャッシュ動作に関連付けます。(CloudFront API では、キーグループ ID を使用してキーグループをキャッシュ動作に関連付けます)。

CloudFront キーペアを作成する (推奨されません。AWSアカウントのルートユーザーが必要です)

⚠ Important

ここでの手順に従う代わりに、信頼されたキーグループのパブリックキーを作成することをお勧めします。署名付き URL および署名付き Cookie のパブリックキーを作成するための推奨される方法については、「[信頼されたキーグループのキーペアを作成する \(推奨\)](#)」を参照してください。

CloudFront キーペアは、次の方法で作成できます。

- AWS Management Console でキーペアを作成し、プライベートキーをダウンロードします。後述の手順を参照してください。
- OpenSSL などのアプリケーションを使用して RSA キーペアを作成し、パブリックキーを AWS Management Console にアップロードします。RSA キーペアの作成の詳細については、「[信頼されたキーグループのキーペアを作成する \(推奨\)](#)」を参照してください。

で CloudFront キーペアを作成するには AWS Management Console

1. AWS アカウントの root ユーザーの認証情報を使用して、AWS Management Console にサインインします。

⚠ Important

IAM ユーザーは CloudFront キーペアを作成できません。キーペアを作成するには、root ユーザーの認証情報を使用してサインインする必要があります。

2. アカウント名を選択してから、[My Security Credentials (セキュリティ認証情報)] を選択します。
3. CloudFront キーペア を選択します。
4. 複数のキーペアが有効になっていないことを確認します。既に 2 つのキーペアが有効になっていると、キーペアを作成できません。
5. [Create New Key Pair (新しいキーペアの作成)] を選択します。

Note

独自のキーペアを作成し、パブリックキーペアをアップロードすることもできます。CloudFront キーペアは 1024、2048、または 4096 ビットキーをサポートします。

6. [Create Key Pair (キーペアの作成)] ダイアログボックスで、[Download Private Key File (プライベートキーファイルのダウンロード)] を選択し、ファイルをコンピュータに保存します。

Important

CloudFront キーペアのプライベートキーを安全な場所に保存し、必要な管理者のみが読み取ることができるようにファイルに対するアクセス許可を設定します。別のユーザーがこのプライベートキーを取得すると、そのユーザーは有効な署名付き URL および署名付き Cookie を生成し、コンテンツをダウンロードできます。プライベートキーは再取得できないため、紛失または削除した場合は、新しい CloudFront キーペアを作成する必要があります。

7. キーペアのキーペア ID を記録しておきます (AWS Management Console ではアクセスキー ID と呼ばれます)。この情報は、署名付き URL または署名付き Cookie を作成するときに使用します。

プライベートキーの形式の変更 (.NET および Java のみ)

.NET または Java を使用して署名付き URL または署名付き Cookie を作成する場合、キーペアのプライベートキーをデフォルトの PEM 形式のまま使用して署名を作成することはできません。代わりに、以下の手順を実行します。

- .NET Framework – .NET Framework で使用する XML 形式にプライベートキーを変換します。いくつかのツールを利用できます。
- Java – DER 形式にプライベートキーを変換します。そのための 1 つの方法は、以下の OpenSSL コマンドを使用することです。以下のコマンドで、`private_key.pem` は PEM 形式のプライベートキーを含むファイルの名前であり、`private_key.der` はコマンドの実行後に DER 形式のプライベートキーを含むファイルの名前です。

```
openssl pkcs8 -topk8 -nocrypt -in private_key.pem -inform PEM -out private_key.der -outform DER
```

エンコーダーが正常に機能するように、Bouncy Castle の Java 用暗号 API の JAR をプロジェクトに追加してから Bouncy Castle プロバイダーを追加します。

ディストリビューションへの署名者の追加

署名者は、ディストリビューションの署名URLs と署名付き Cookie を作成できる信頼できるキーグループ (推奨) または CloudFront キーペアです。署名URLs または署名付き Cookie を CloudFront ディストリビューションで使用するには、署名者を指定する必要があります。

署名者はキャッシュ動作に関連付けられています。これにより、同じディストリビューション内で、一部のファイルに署名付き URL または署名付き Cookie を要求し、その他のファイルには要求しないということが可能になります。ディストリビューションでは、対応するキャッシュ動作に関連付けられているファイルにのみ、署名付き URL または Cookie が必要です。

同様に、署名者は、対応するキャッシュ動作に関連付けられているファイルの URL または Cookie にのみ署名できます。例えば、1つのキャッシュ動作に対して1つの署名者があり、別のキャッシュ動作に対して別の署名者がある場合、どちらの署名者も、もう一方のキャッシュ動作に関連付けられたファイルに対して署名付き URL または署名付き Cookie を作成できません。

Important

ディストリビューションに署名者を追加する前に、以下の手順を実行します。

- キャッシュ動作のパスパターンとキャッシュ動作のシーケンスを慎重に定義して、ユーザーにコンテンツへの意図しないアクセスを許可したり、すべてのユーザーを対象としたコンテンツへのアクセスを禁止したりしないようにします。

たとえば、リクエストが、2つのキャッシュ動作のパスパターンに一致したと仮定します。最初のキャッシュ動作は署名付き URL または署名付き Cookie を要求しませんが、2番目のキャッシュ動作はこれらを要求します。ユーザーは署名付き URL または署名付き Cookie を使用せずにファイルにアクセスできます。これは、CloudFront が、最初の一致に関連付けられたキャッシュ動作を処理するためです。

パスパターンの詳細については、「[パスパターン](#)」を参照してください。

- コンテンツの配信に既に使用しているディストリビューションの場合は、署名者を追加する前に、署名付き URL と署名付き Cookie の生成を開始する準備ができていることを確認

してください。署名者を追加すると、は有効な署名付き URL または署名付き Cookie を含まないリクエスト CloudFront を拒否します。

CloudFront コンソールまたは API を使用して、ディストリビューションに署名 CloudFront 者を追加できます。

トピック

- [CloudFront コンソールを使用したディストリビューションへの署名者の追加](#)
- [CloudFront API を使用したディストリビューションへの署名者の追加](#)

CloudFront コンソールを使用したディストリビューションへの署名者の追加

以下の手順では、信頼されたキーグループを署名者として追加する方法を示します。信頼された署名者として AWS アカウントを追加することもできますが、お勧めしません。

コンソールを使用してディストリビューションに署名者を追加するには

1. 信頼された署名者として使用するキーグループのキーグループ ID を記録します。詳細については、「[信頼されたキーグループのキーペアを作成する \(推奨\)](#)」を参照してください。
2. で CloudFront コンソールを開きます <https://console.aws.amazon.com/cloudfront/v4/home>。
3. 署名付き URL または署名付き Cookie でファイルを保護するディストリビューションを選択します。

Note

新しいディストリビューションに署名者を追加するには、ディストリビューションを作成するときにステップ 6 で説明したのと同じ設定を指定します。

4. [Behaviors] タブを選択します。
5. 署名付き URL または署名付き Cookie で保護するファイルとパスパターンが一致するキャッシュ動作を選択し、[Edit (編集)] を選択します。
6. [Edit Behavior (動作の編集)] ページで、以下の手順を実行します。
 - a. [Restrict Viewer Access (Use Signed URLs or Signed Cookies) (ビューワのアクセスを制限 (署名付き URL または署名付き Cookie の使用))] で、[はい] を選択します。

- b. [Trusted Key Groups or Trusted Signer (信頼されたキーグループまたは信頼された署名者)] で、[Trusted Key Groups (信頼されたキーグループ)] を選択します。
 - c. [Trusted Key Groups (信頼されたキーグループ)] で、追加するキーグループを選択し、[Add (追加)] を選択します。複数のキーグループを追加する場合は、この手順を繰り返します。
7. [Yes, Edit (はい、編集する)] を選択して、キャッシュ動作を更新します。

CloudFront API を使用したディストリビューションへの署名者の追加

CloudFront API を使用して、信頼できるキーグループを署名者として追加できます。署名者を既存のディストリビューションまたは新しいディストリビューションに追加できます。どちらの場合も、TrustedKeyGroups 要素の値を指定します。

信頼された署名者として AWS アカウントを追加することもできますが、お勧めしません。

「Amazon CloudFront API リファレンス」の以下のトピックを参照してください。

- 既存のディストリビューションを更新する – [UpdateDistribution](#)
- 新しいディストリビューションを作成する – [CreateDistribution](#)

キーペアの更新

署名付き URL と署名付き Cookie のキーペアを定期的に更新 (変更) することをお勧めします。有効期限がまだ切れていない URL または Cookie を無効にすることなく、署名付き URL または署名付き Cookie の作成に使用しているキーペアを更新するには、以下のタスクを実行します。

1. 新しいキーペアを作成し、パブリックキーをキーグループに追加します。詳細については、「[信頼されたキーグループのキーペアを作成する \(推奨\)](#)」を参照してください。
2. 前の手順で新しいキーグループを作成した場合は、[キーグループを署名者としてディストリビューションに追加](#)します。

Important

キーグループから既存のパブリックキーを削除したり、ディストリビューションからキーグループを削除したりしないでください。新規追加のみを行ってください。

3. 新しいキーペアのプライベートキーを使用して署名を作成するようにアプリケーションを更新します。新しいプライベートキーで署名された URL または Cookie が機能することを確認します。

4. 以前のプライベートキーを使用して署名付き URL または Cookie の有効期限切れ日時が経過するまで待ちます。次に、古いパブリックキーをキーグループから削除します。ステップ 2 で新しいキーグループを作成した場合は、ディストリビューションから古いキーグループを削除します。

署名付き URL と署名付き Cookie の選択

CloudFront 署名URLs と署名付き Cookie は同じ基本機能を提供します。これにより、コンテンツにアクセスできるユーザーを制御できます。を通じてプライベートコンテンツを供給する場合に CloudFront、署名URLs と署名付き Cookie のどちらを使用するかを決定しようとする場合は、次の点を考慮してください。

次のような場合は、署名付き URL を使用します。

- 個別のファイル (アプリケーションのインストールダウンロード) へのアクセスを制限する場合。
- ユーザーが Cookie をサポートしていないクライアント (カスタム HTTP クライアントなど) を使用している場合。

次のような場合は、署名付き Cookie を使用します。

- 複数の制限されたファイル (HLS 形式の動画のすべてのファイルやウェブサイトの購読者の領域にあるすべてのファイルなど) へのアクセスを提供する場合。
- 現在の URL を変更したくない場合。

現在署名付き URL を使用していない場合で、署名なし URL に次のクエリ文字列パラメータ含まれる場合、署名付き URL と署名付き Cookie のいずれも使用できません。

- Expires
- Policy
- Signature
- Key-Pair-Id

CloudFront は、これらのクエリ文字列パラメータのいずれかを含む URLs が署名URLsであると仮定するため、署名付き Cookie は検索されません。

署名付き URL と署名付き Cookie の両方の使用

署名付き URL は署名付き Cookie よりも優先されます。署名URLs と署名付き Cookie の両方を使用して同じファイルへのアクセスを制御し、ビューワーが署名付き URL を使用してファイルをリクエストする場合、は署名付き URL のみに基づいてファイルをビューワーに返すかどうか CloudFront を決定します。

署名付き URL の使用

トピック

- [署名付き URL の既定ポリシーとカスタムポリシーの選択](#)
- [署名付き URL の仕組み](#)
- [署名付き URL の有効期間の選択](#)
- [は、署名付き URL の有効期限日時をいつ CloudFront 確認しますか？](#)
- [サンプルコードおよびサードパーティーツール](#)
- [既定ポリシーを使用する署名付き URL の作成](#)
- [カスタムポリシーを使用する署名付き URL の作成](#)

署名付き URL には、有効期限切れ日時など、追加の情報も含まれており、コンテンツへのアクセスをより詳細に制御できます。この追加情報は、既定ポリシーまたはカスタムポリシーに基づくポリシーステートメントに含まれます。既定ポリシーとカスタムポリシーの違いを以下の 2 つのセクションで説明します。

Note

既定ポリシーを使用して一部の署名付き URL を作成し、同じディストリビューションで、カスタムポリシーを使用して一部の署名付き URL を作成することができます。

署名付き URL の既定ポリシーとカスタムポリシーの選択

署名付き URL を作成する場合、URL の有効期間など、署名付き URL で制限を指定する JSON 形式のポリシーステートメントを作成します。既定ポリシーまたはカスタムポリシーのいずれかを使用できます。既定ポリシーとカスタムポリシーの比較を以下に示します。

説明	既定ポリシー	カスタムポリシー
ポリシーステートメントを複数のファイル用に再利用できる。ポリシーステートメントを再利用するには、Resource オブジェクトでワイルドカード文字を使用する必要があります。(詳しくは、「 カスタムポリシーを使用する署名付き URL のポリシーステートメントで指定する値 」を参照してください)。	いいえ	はい
ユーザーがコンテンツへのアクセスを開始できる日時を指定できる。	いいえ	はい (オプション)
ユーザーがコンテンツにアクセスできなくなる日時を指定できる。	はい	はい
コンテンツにアクセスできるユーザーの IP アドレスまたは IP アドレス範囲を指定できる。	いいえ	はい (オプション)
署名付き URL にポリシーの base64 エンコードされたバージョンが含まれているため、URL が長くなる。	いいえ	はい

既定ポリシーを使用して署名付き URL を作成する方法については、「[既定ポリシーを使用する署名付き URL の作成](#)」を参照してください。

カスタムポリシーを使用して署名付き URL を作成する方法については、「[カスタムポリシーを使用する署名付き URL の作成](#)」を参照してください。

署名付き URL の仕組み

ここでは、署名URLs 用に CloudFront と Amazon S3 を設定する方法と、ユーザーが署名付き URL を使用してファイルをリクエストしたときの CloudFront 応答の概要を示します。

- CloudFront デイストリビューションで、1 つ以上の信頼されたキーグループを指定します。このグループには、CloudFront が URL 署名の検証に使用できるパブリックキーが含まれています。対応するプライベートキーを使用して URL に署名します。

詳細については、「[署名付き URL と署名付き Cookie を作成できる署名者の指定](#)」を参照してください。

2. アプリケーションを開発して、ユーザーがコンテンツへのアクセス許可を持つかどうかを決定し、アプリケーションでアクセスを制限するファイルまたは部分用に署名付き URL を作成するかを決定します。詳細については、以下のトピックを参照してください。

- [既定ポリシーを使用する署名付き URL の作成](#)
- [カスタムポリシーを使用する署名付き URL の作成](#)

3. 署名付き URL を必要とするファイルをユーザーが要求します。
4. アプリケーションは、ファイルにアクセスするための資格がユーザーにあることを検証します。たとえば、ユーザーがサインインしていること、コンテンツへのアクセス料を支払っていること、他のいくつかのアクセス要件を満たしていることを検証します。
5. アプリケーションは署名付き URL を作成してエンドユーザーに返します。
6. 署名付き URL を使用すると、ユーザーはコンテンツのダウンロードやストリーミングができます。

このステップは自動で実行されます。ユーザーは、コンテンツにアクセスする以外に、何も行う必要はありません。たとえば、ユーザーがウェブブラウザでコンテンツにアクセスすると、アプリケーションは署名付き URL をブラウザに返します。ブラウザはすぐに署名付き URL を使用して、ユーザーからの介入なしに CloudFront エッジキャッシュ内のファイルにアクセスします。

7. CloudFront はパブリックキーを使用して署名を検証し、URL が改ざんされていないことを確認します。署名が無効である場合、リクエストは拒否されます。

署名が有効な場合、は URL のポリシーステートメント CloudFront を調べ (既定ポリシーを使用している場合はポリシーステートメントを作成します)、リクエストがまだ有効であることを確認します。例えば、URL に開始日時と終了日時を指定した場合、は、アクセスを許可する期間中にユーザーがコンテンツにアクセスしようとしている CloudFront ことを確認します。

リクエストがポリシーステートメントの要件を満たしている場合、は標準オペレーション CloudFront を実行します。は、ファイルがエッジキャッシュに既に存在するかどうかを判断し、必要に応じてリクエストをオリジンに転送して、そのファイルをユーザーに返します。

Note

署名なし URL にクエリ文字列パラメータが含まれている場合は、署名する URL にそれらのパラメータを含めてください。URL への署名後にその URL にクエリ文字列を追加すると、HTTP 403 ステータスが返されます。

署名付き URL の有効期間の選択

短期間 (おそらく数分) のみ有効な署名付き URL を使用してプライベートコンテンツを配信できます。このような短期間有効な署名付き URLs は、映画のレンタルや音楽のダウンロード on-the-fly をオンデマンドで顧客に配信するなど、特定の目的でコンテンツをユーザーに配信するのに適していません。署名付き URL を短期間だけ有効にする場合、開発したアプリケーションを使用して、署名付き URL を自動生成することが必要になる場合があります。ユーザーがファイルのダウンロードを開始するか、メディアファイルの再生を開始すると、CloudFront は URL の有効期限を現在の時刻と比較し、URL がまだ有効かどうかを判断します。

これよりも有効期間の長い (おそらく数年間の) 署名付き URL を使用して、プライベートコンテンツを配信できます。有効期間の長い署名付き URL は、プライベートコンテンツを既知のユーザーに配信する場合に役立ちます。たとえば、事業計画を投資家に配信したり、教育資料を従業員に配信したりする場合に役立ちます。これらの長期的な署名付き URL を生成するアプリケーションを開発できます。

は、署名付き URL の有効期限日時をいつ CloudFront 確認しますか？

CloudFront は、HTTP リクエスト時に署名付き URL の有効期限日時を確認します。有効期限切れ時刻の直前にクライアントが大きなファイルのダウンロードを開始した場合、ダウンロード中に有効期限切れ時刻が経過してもダウンロードは完了します。TCP 接続が中断し、有効期限切れ時刻が経過した後にクライアントがダウンロードを再開した場合、ダウンロードは失敗します。

クライアントが、ファイルを断片的に取得するレンジ GET を使用した場合、有効期限切れ時刻が経過した後に実行された GET リクエストは失敗します。レンジ GET の詳細については、「[ガオブジェクトの部分リクエスト CloudFront を処理する方法 \(範囲 GETs\)](#)」を参照してください。

サンプルコードおよびサードパーティーツール

署名付き URL の、ハッシュ化および署名されたパートを作成するサンプルコードについては、以下のトピックを参照してください。

- [Perl を使用して URL 署名を作成する](#)
- [PHP を使用して URL 署名を作成する](#)
- [C# と .NET Framework を使用して URL 署名を作成する](#)
- [Java を使用して URL 署名を作成する](#)

既定ポリシーを使用する署名付き URL の作成

既定ポリシーを使用して署名付き URL を作成するには、以下の手順を実行します。

既定ポリシーを使用して署名付き URL を作成するには

1. .NET または Java を使用して署名付き URL を作成しており、キーペアのプライベートキーをデフォルトの .pem 形式から .NET または Java 対応の形式に変更していない場合は、それを変換します。詳細については、「[プライベートキーの形式の変更 \(.NET および Java のみ\)](#)」を参照してください。
2. 以下の値を指定の順序で連結し、パート間の空白文字 (タブと改行文字を含む) を削除します。アプリケーションコード内の文字列にエスケープ文字を含めることが必要になる場合があります。すべての値は文字列型です。各パートの番号 (1) は以下の 2 つの例の番号に対応します。

1

URL

ベース URL は、署名付き CloudFront URL を使用していない場合、ファイルへのアクセスに使用する URL です。これには、独自のクエリ文字列パラメータも含まれます。URLs デイストリビューション用の URL 形式の詳細については、「[ファイルの URL 形式のカスタマイズ CloudFront](#)」を参照してください。

- 次の CloudFront URL は、デイストリビューション内のイメージファイル用です (CloudFront ドメイン名を使用)。image.jpg は images ディレクトリにあります。URL 内のファイルへのパスは、HTTP サーバーまたは Amazon S3 バケットのファイルへのパスに一致する必要があります。

```
https://d111111abcdef8.cloudfront.net/images/image.jpg
```

- 次の CloudFront URL にはクエリ文字列が含まれています。

```
https://d111111abcdef8.cloudfront.net/images/image.jpg?size=large
```

- 以下の CloudFront URLs は、デイストリビューション内のイメージファイル用です。両方の URL で代替ドメイン名が使用されており、2 番目の URL にはクエリ文字列が含まれています。

```
https://www.example.com/images/image.jpg
```

```
https://www.example.com/images/image.jpg?color=red
```

- 次の CloudFront URL は、代替ドメイン名と HTTPS プロトコルを使用するディストリビューション内のイメージファイル用です。

```
https://www.example.com/images/image.jpg
```

2

?

? は、クエリ文字列パラメータがベース URL の後に続いていることを示します。独自のクエリ文字列パラメータがない場合も ? を含めます。

3

```
##### (####) &
```

この値はオプションです。独自のクエリ文字列パラメータ、たとえば次のクエリ文字列パラメータを追加すると仮定します。

```
color=red&size=medium
```

この場合、?

2

を参照) の後、かつ Expires パラメータの前にこのパラメータを追加します。特定のまれな状況では、Key-Pair-Id の後にクエリ文字列パラメータを配置する必要があります。

⚠ Important

パラメータに Expires、Signature、または Key-Pair-Id という名前を付けることはできません。

独自のパラメータを追加する場合は、最後のパラメータを含め、各パラメータの後に & を付け加えます。

4

```
Expires=Unix #### (###) ##### (UTC) ###
```

URL によるファイルへのアクセスの許可を停止する日付と時刻。

有効期限切れ日時を Unix 時間形式 (秒単位) および協定世界時 (UTC) で指定します。たとえば、UTC の 2013 年 1 月 1 日午前 10 時 00 分は、Unix 時間形式の 1357034400 に変換されます。エポック時間を使用するには、日付に 2147483647 (2038 年 1 月 19 日 03:14:07 UTC) より前の 32 ビット整数を使用します。UTC の詳細については、RFC 3339, Date and Time on the Internet: Timestamps (<https://tools.ietf.org/html/rfc3339>) を参照してください。

5

&Signature=#####

ハッシュ化され、署名された base64 エンコードバージョンの JSON ポリシーステートメント。詳細については、「[既定ポリシーを使用する署名付き URL の署名の作成](#)」を参照してください。

6

&Key-Pair-Id=#####CloudFront ##### ID

CloudFront パブリックキーの ID。例: K2JCMDEHXQW5F。パブリックキー ID は、署名付き URL の検証に使用する CloudFront パブリックキーをに伝えます。CloudFront は、署名内の情報をポリシーステートメント内の情報と比較し、URL が改ざんされていないことを確認します。

このパブリックキーは、ディストリビューションの信頼された署名者であるキーグループに属している必要があります。詳細については、「[署名付き URL と署名付き Cookie を作成できる署名者の指定](#)」を参照してください。

署名付き URL の例:

1

d111111abcdef8.cloudfront.net/image.jpg

2

?

3

color=red&size=medium&

4

Expires=1357034400

5

&Signature=nitfHRCrtziw02HwPfWw~yYDhUF5EwRunQA-j19DzZrvDh6hQ731Dx~-

https

ar3UocvvRQVw6EkC~GdpGQyy0SKQim-

TxAnW7d8F5Kkai9HVx0FIu-5jcQb0UEmatEXAMPLE3ReXySpLSMj0yCd3ZAB4UcBCAqEijkytL6f3fVY

6

&Key-Pair-Id=K2JJCJMDEHXQW5F

既定ポリシーを使用する署名付き URL の署名の作成

既定ポリシーを使用する署名付き URL の署名を作成するには、以下の手順を実行します。

1. ポリシーステートメントを作成します。[既定ポリシーを使用する署名付き URL のポリシーステートメントの作成](#) を参照してください。
2. ポリシーステートメントに署名して、署名を作成します。[既定ポリシーを使用する署名付き URL の署名の作成](#) を参照してください。

既定ポリシーを使用する署名付き URL のポリシーステートメントの作成

既定ポリシーを使用して署名付き URL を作成する場合、Signature パラメータは、ポリシーステートメントのハッシュ化および署名されたバージョンです。カスタムポリシーを使用する署名付き URL とは異なり、既定ポリシーを使用する署名付き URL では、URL にポリシーステートメントを含めません。ポリシーステートメントを作成するには、以下の手順を実行します。

既定ポリシーを使用する署名付き URL のポリシーステートメントを作成するには

1. 以下の JSON 形式および UTF-8 文字エンコードを使用してポリシーステートメントを構築します。すべての句読点および他のリテラル値を、指定されたとおりに正確に含めます。Resource および DateLessThan パラメータの詳細については、「[既定ポリシーを使用する署名付き URL のポリシーステートメントで指定する値](#)」を参照してください。

```
{
  "Statement": [
    {
      "Resource": "base URL or stream name",
      "Condition": {
        "DateLessThan": {
          "AWS:EpochTime": ending date and time in Unix time format and
          UTC
        }
      }
    }
  ]
}
```

```
}
```

2. ポリシーステートメントからすべての空白文字 (タブと改行文字を含む) を削除します。アプリケーションコード内の文字列にエスケープ文字を含めることが必要になる場合があります。

既定ポリシーを使用する署名付き URL のポリシーステートメントで指定する値

既定ポリシーのポリシーステートメントを作成する場合、以下の値を指定します。

リソース

Note

Resource の日付形式は 1 つだけ指定できます。

クエリ文字列が含まれているが、CloudFront Expires、Signature および Key-Pair-Id パラメータを除く基本 URL。例：

```
https://d1111111abcdef8.cloudfront.net/images/horizon.jpg?  
size=large&license=yes
```

次の点に注意してください。

- プロトコル – 値は `http://` または `https://` で始まっている必要があります。
- クエリ文字列パラメータ – クエリ文字列パラメータがない場合は、疑問符を省略します。
- 代替ドメイン名 – URL で代替ドメイン名 (CNAME) を指定する場合は、ウェブページまたはアプリケーション内のファイルを参照するときに代替ドメイン名を指定する必要があります。オブジェクトの Amazon S3 URL を指定しないでください。

DateLessThan

URL の有効期限切れ日時。Unix 時間形式 (秒単位) および協定世界時 (UTC) で指定します。たとえば、UTC の 2013 年 1 月 1 日午前 10 時 00 分は、Unix 時間形式の 1357034400 に変換されます。

この値は、署名付き URL 内の Expires クエリ文字列パラメータの値と一致する必要があります。値を引用符で囲まないでください。

詳細については、「[は、署名付き URL の有効期限日時をいつ CloudFront 確認しますか？](#)」を参照してください。

既定ポリシーを使用する署名付き URL のポリシーステートメントの例

署名付き URL 内の既定ポリシーの以下のポリシーステートメントの例を使用すると、ユーザーは、UTC の 2013 年 1 月 1 日午前 10 時 00 分までファイル `https://d111111abcdef8.cloudfront.net/horizon.jpg` にアクセスできます。

```
{
  "Statement": [
    {
      "Resource": "https://d111111abcdef8.cloudfront.net/horizon.jpg?size=large&license=yes",
      "Condition": {
        "DateLessThan": {
          "AWS:EpochTime": 1357034400
        }
      }
    }
  ]
}
```

既定ポリシーを使用する署名付き URL の署名の作成

署名付き URL の Signature パラメータの値を作成するには、「[既定ポリシーを使用する署名付き URL のポリシーステートメントの作成](#)」で作成したポリシーステートメントをハッシュ化して署名します。

ポリシーステートメントのハッシュ化、署名、およびエンコードを行う方法の詳細および例については、以下の各資料を参照してください。

- [Linux コマンドおよび OpenSSL を使用した Base64 エンコードおよび暗号化](#)
- [署名付き URL の署名を作成するためのコード例](#)

オプション 1: 既定ポリシーを使用して署名を作成するには

1. 「[既定ポリシーを使用する署名付き URL のポリシーステートメントを作成するには](#)」の手順で作成したポリシーステートメントを、SHA-1 ハッシュ関数と RSA を使用してハッシュ化し、署名します。空白を含まないバージョンのポリシーステートメントを使用します。

ハッシュ関数に必要なプライベートキーには、対応するパブリックキーがディストリビューション内のアクティブな信頼されたキーグループにあるものを使用してください。

Note

ポリシーステートメントをハッシュ化および署名するための方法は、プログラミング言語およびプラットフォームによって異なります。サンプルコードについては、「[署名付き URL の署名を作成するためのコード例](#)」を参照してください。

2. ハッシュ化および署名された文字列から、空白文字 (タブや改行文字を含む) を削除します。
3. MIME base64 エンコーディングを使用して文字列を Base64 エンコードします。詳細については、RFC 2045, MIME (Multipurpose Internet Mail Extensions) Part One: Format of Internet Message Bodies の [Section 6.8, Base64 Content-Transfer-Encoding](#) を参照してください。
4. URL クエリ文字列内の無効な文字を有効な文字で置き換えます。次の表に無効な文字と有効な文字を示します。

無効な文字 (置換元)	有効な文字 (置換先)
+	- (ハイフン)
=	_ (下線)
/	~ (チルダ)

5. 結果の値を署名付き URL の &Signature= の後に付加し、「[既定ポリシーを使用して署名付き URL を作成するには](#)」に戻って、署名付き URL の各パートの連結を終了します。

カスタムポリシーを使用する署名付き URL の作成

トピック

- [カスタムポリシーを使用する署名付き URL のポリシーステートメントの作成](#)
- [カスタムポリシーを使用する署名付き URL のポリシーステートメントの例](#)
- [カスタムポリシーを使用する署名付き URL の署名の作成](#)

カスタムポリシーを使用して署名付き URL を作成するには、以下の手順を実行します。

カスタムポリシーを使用して署名付き URL を作成するには

1. .NET または Java を使用して署名付き URL を作成しており、キーペアのプライベートキーをデフォルトの .pem 形式から .NET または Java 対応の形式に変更していない場合は、それを変換します。詳細については、「[プライベートキーの形式の変更 \(.NET および Java のみ\)](#)」を参照してください。
2. 以下の値を指定の順序で連結し、パート間の空白文字 (タブと改行文字を含む) を削除します。アプリケーションコード内の文字列にエスケープ文字を含めることが必要になる場合があります。すべての値は文字列型です。各パートの番号 (**1**) は以下の 2 つの例の番号に対応します。

1

URL

ベース URL は、署名付き CloudFront URL を使用していない場合、ファイルへのアクセスに使用する URL です。これには、独自のクエリ文字列パラメータも含まれます。URLs デイストリビューション用の URL 形式の詳細については、「[このファイルの URL 形式のカスタマイズ CloudFront](#)」を参照してください。

以下の例は、デイストリビューションで指定する値を示しています。

- 次の CloudFront URL は、デイストリビューション内のイメージファイル用です (CloudFront ドメイン名を使用)。image.jpg は images ディレクトリにあります。URL 内のファイルへのパスは、HTTP サーバーまたは Amazon S3 バケットのファイルへのパスに一致する必要があります。

```
https://d111111abcdef8.cloudfront.net/images/image.jpg
```

- 次の CloudFront URL にはクエリ文字列が含まれています。

```
https://d111111abcdef8.cloudfront.net/images/image.jpg?size=large
```

- 以下の CloudFront URLs は、デイストリビューション内のイメージファイル用です。両方の URL で代替ドメイン名が使用されており、2 番目の URL にはクエリ文字列が含まれています。

```
https://www.example.com/images/image.jpg
```

```
https://www.example.com/images/image.jpg?color=red
```

- 次の CloudFront URL は、代替ドメイン名と HTTPS プロトコルを使用するディストリビューション内のイメージファイル用です。

```
https://www.example.com/images/image.jpg
```

2

?

? は、クエリ文字列パラメータがベース URL の後に続いていることを示します。独自のクエリ文字列パラメータがない場合も ? を含めます。

3

(####) &

この値はオプションです。独自のクエリ文字列パラメータ、たとえば次のクエリ文字列パラメータを追加すると仮定します。

```
color=red&size=medium
```

この場合、?

(2)

を参照) の後、かつ Policy パラメータの前にこのパラメータを追加します。特定のまれな状況では、Key-Pair-Id の後にクエリ文字列パラメータを配置する必要があります。

⚠ Important

パラメータに Policy、Signature、または Key-Pair-Id という名前を付けることはできません。

独自のパラメータを追加する場合は、最後のパラメータを含め、各パラメータの後に & を付け加えます。

4

Policy=##### base64 #####

空白文字が削除され、base64 エンコードされた、JSON 形式のポリシーステートメント。詳細については、「[カスタムポリシーを使用する署名付き URL のポリシーステートメントの作成](#)」を参照してください。


```
j19DzZrvDh6hQ731Dx~ -ar3UocvvRQVw6EkC~GdpGQyy0SKQim-  
TxAnW7d8F5Kkai9HVx0FIu-5jcQb0UEmat  
EXAMPLE3ReXySpLSMj0yCd3ZAB4UcBCAqEijkYtL6f3fVYNGQI6
```

6**&Key-**

```
Pair-Id=K2JJCJMDEHXQW5F
```

カスタムポリシーを使用する署名付き URL のポリシーステートメントの作成

カスタムポリシーを使用する署名付き URL のポリシーステートメントを作成するには、以下の手順を実行します。

さまざまな方法でファイルへのアクセスを制御するポリシーステートメントの例については、「[the section called “カスタムポリシーを使用する署名付き URL のポリシーステートメントの例”](#)」を参照してください。

カスタムポリシーを使用する署名付き URL のポリシーステートメントを作成するには

1. 以下の JSON 形式を使用してポリシーステートメントを構築します。小なり記号 (<) と大なり記号 (>)、およびそれらの中の説明は、独自の値に置き換えます。詳細については、「[the section called “カスタムポリシーを使用する署名付き URL のポリシーステートメントで指定する値”](#)」を参照してください。

```
{  
  "Statement": [  
    {  
      "Resource": "<Optional but recommended: URL of the file>",  
      "Condition": {  
        "DateLessThan": {  
          "AWS:EpochTime": <Required: ending date and time in Unix time  
format and UTC>  
        },  
        "DateGreaterThan": {  
          "AWS:EpochTime": <Optional: beginning date and time in Unix time  
format and UTC>  
        },  
        "IpAddress": {  
          "AWS:SourceIp": "<Optional: IP address>"  
        }  
      }  
    }  
  ]  
}
```

```
}

```

次の点に注意してください。

- ポリシーには、1つのステートメントのみを含めることができます。
 - UTF-8 文字エンコードを使用します。
 - すべての句読点およびパラメータ名を、指定されたとおりに正確に含めます。パラメータ名の省略形は受け付けられません。
 - Condition セクションのパラメータの順序は問題ではありません。
 - Resource、DateLessThan、DateGreaterThan、および IPAddress の値については、「[the section called “カスタムポリシーを使用する署名付き URL のポリシーステートメントで指定する値”](#)」を参照してください。
2. ポリシーステートメントからすべての空白文字 (タブと改行文字を含む) を削除します。アプリケーションコード内の文字列にエスケープ文字を含める必要がある場合があります。
 3. MIME base64 エンコーディングを使用してポリシーステートメントを Base64 エンコードします。詳細については、RFC 2045, MIME (Multipurpose Internet Mail Extensions) Part One: Format of Internet Message Bodies の [Section 6.8, Base64 Content-Transfer-Encoding](#) を参照してください。
 4. URL クエリ文字列内の無効な文字を有効な文字で置き換えます。次の表に無効な文字と有効な文字を示します。

無効な文字 (置換元)	有効な文字 (置換先)
+	- (ハイフン)
=	_ (下線)
/	~ (チルダ)

5. 結果の値を署名付き URL の Policy= の後に付加します。
6. ポリシーステートメントのハッシュ化、署名、および base64 エンコードを行って、署名付き URL の署名を作成します。詳細については、「[the section called “カスタムポリシーを使用する署名付き URL の署名の作成”](#)」を参照してください。

カスタムポリシーを使用する署名付き URL のポリシーステートメントで指定する値

カスタムポリシーのポリシーステートメントを作成する場合、以下の値を指定します。

リソース

クエリ文字列を含む URL。ただし、CloudFront Policy、Signature、および Key-Pair-Id パラメータは除きます。例:

```
https://d1111111abcdef8.cloudfront.net/images/horizon.jpg\?  
size=large&license=yes
```

Resource の URL の値は 1 つだけ指定できます。

Important

ポリシーで、Resource パラメータを省略できますが、その場合、署名付き URL を持つすべてのユーザーが、署名付き URL の作成に使用するキーペアに関連付けられたあらゆるディストリビューションのすべてのファイルにアクセスできることになります。

次の点に注意してください。

- プロトコル – 値は http://、https://、または *:// で始まっている必要があります。
- クエリ文字列パラメータ – URL にクエリ文字列パラメータがある場合は、バックスラッシュ文字 (\) を使用してクエリ文字列の最初の疑問符 (?) をエスケープします。例:

```
https://d1111111abcdef8.cloudfront.net/images/horizon.jpg\?  
size=large&license=yes
```

- ワイルドカード文字 – ポリシーの URL にはワイルドカード文字を使用できます。次のワイルドカード文字がサポートされています。
 - アスタリスク (*) は、0 個以上の文字に一致します
 - 疑問符 (?) は、1 つの文字に一致します

がポリシー内の URL を HTTP リクエスト内の URL と CloudFront 一致させる場合、ポリシー内の URL は、プロトコル、ドメイン、パス、クエリ文字列の 4 つのセクションに分割されません。

```
[protocol]://[domain]/[path]\?[query string]
```

ポリシーの URL にワイルドカード文字を使用する場合、ワイルドカードマッチングはワイルドカードを含むセクションの境界内でのみ適用されます。例えば、次のようなポリシーの URL を考えてみます。

```
https://www.example.com/hello*world
```

この例では、アスタリスクワイルドカード (*) はパスセクション内でのみ適用されるため、URL `https://www.example.com/helloworld` や `https://www.example.com/hello-world` には一致しますが、URL `https://www.example.net/hello?world` には一致しません。

ワイルドカードマッチングのセクションの境界には、次の例外が適用されます。

- パスセクションの末尾にアスタリスクがある場合、クエリ文字列セクションにアスタリスクが付いていることを意味します。たとえば、`http://example.com/hello*` と `http://example.com/hello*\?*` は同じです。
- ドメインセクションの末尾にアスタリスクがある場合、パスセクションとクエリ文字列セクションの両方にアスタリスクが付いていることを意味します。たとえば、`http://example.com*` と `http://example.com/*\?*` は同じです。
- ポリシーの URL では、プロトコルセクションを省略し、ドメインセクションをアスタリスクで始めることができます。その場合、プロトコルセクションは暗黙的にアスタリスクに設定されます。例えば、ポリシーの URL `*example.com` は `*://*example.com/` と同等です。
- アスタリスク ("Resource": "*") 単独の場合は、どの URL にも一致します。

例えば、ポリシーの値 `https://d111111abcdef8.cloudfront.net/*game_download.zip*` は、次の URL すべてに一致します。

- `https://d111111abcdef8.cloudfront.net/game_download.zip`
- `https://d111111abcdef8.cloudfront.net/example_game_download.zip?license=yes`
- `https://d111111abcdef8.cloudfront.net/test_game_download.zip?license=temp`
- 代替ドメイン名 – ポリシーの URL で代替ドメイン名 (CNAME) を指定する場合、HTTP リクエストはウェブページまたはアプリケーション内の代替ドメイン名を使用する必要があります。ポリシーのファイルの Amazon S3 URL を指定しないでください。

DateLessThan

URL の有効期限切れ日時。Unix 時間形式 (秒単位) および協定世界時 (UTC) で指定します。ポリシーでは、値を引用符で囲まないでください。UTC の詳細については、「[Date and Time on the Internet: Timestamps](#)」を参照してください。

たとえば、UTC の 2023 年 1 月 31 日午前 10 時 00 分は、Unix 時間形式の 1675159200 に変換されます。

これは、Condition セクションで唯一必須のパラメータです。では、ユーザーがプライベートコンテンツに永続的にアクセスできないように、この値 CloudFront が必要です。

詳細については、「[the section called “は、署名付き URL の有効期限日時をいつ CloudFront 確認しますか?”](#)」を参照してください。

DateGreaterThan (オプション)

オプションの URL 開始日時。Unix 時間形式 (秒単位) および協定世界時 (UTC) で指定します。ユーザーは、指定された日時以前のファイルにアクセスすることはできません。値を引用符で囲まないでください。

IpAddress (オプション)

HTTP リクエストを実行するクライアントの IP アドレス。次の点に注意してください。

- ファイルへのアクセスをすべての IP アドレスに許可するには、IpAddress パラメータを省略します。
- IP アドレスまたは IP アドレス範囲を 1 つ指定できます。2 つの別々の範囲のどちらかにクライアントの IP アドレスが入っている場合にアクセスを許可するようなポリシーを使用することはできません。
- 1 つの IP アドレスからのアクセスを許可するには、以下のように指定します。

```
"IPv4 IP #####/32"
```

- IP アドレス範囲は標準の IPv4 CIDR 形式 (192.0.2.0/24 など) で指定する必要があります。詳細については、「[Classless Inter-domain Routing \(CIDR\): The Internet Address Assignment and Aggregation Plan](#)」を参照してください。

Important

IPv6 形式の IP アドレス (例: 2001:0db8:85a3::8a2e:0370:7334) はサポートされていません。

IpAddress を含むカスタムポリシーを使用する場合、ディストリビューションで IPv6 は有効にしません。一部のコンテンツへのアクセスを IP アドレスによって制限し、他のコンテンツで IPv6 リクエストをサポートする場合、2 つのディストリビューションを作成します。詳細については、トピック「[the section called “指定する値”](#)」の「[the section called “IPv6 を有効にする”](#)」を参照してください。

カスタムポリシーを使用する署名付き URL のポリシーステートメントの例

以下のポリシーステートメントの例は、特定のファイル、ディレクトリ内のすべてのファイル、またはキーペア ID に関連付けられたすべてのファイルへのアクセスを制御する方法を示しています。また、この例は、個々の IP アドレスまたは IP アドレス範囲からのアクセスを制御する方法、および指定された日時以降にユーザーが署名付き URL を使用することを禁止する方法も示しています。

これらの例のいずれかをコピーして貼り付ける場合は、すべての空白文字 (タブと改行文字を含む) を削除し、値を独自の値に置き換えて、右中かっこ (}) の後に改行文字を含めます。

詳細については、「[the section called “カスタムポリシーを使用する署名付き URL のポリシーステートメントで指定する値”](#)」を参照してください。

トピック

- [ポリシーステートメントの例: IP アドレス範囲から 1 つのファイルにアクセスする](#)
- [ポリシーステートメントの例: IP アドレス範囲からディレクトリ内のすべてのファイルにアクセスする](#)
- [ポリシーステートメントの例: キーペア ID に関連付けられたすべてのファイルに 1 つの IP アドレスからアクセスする](#)

ポリシーステートメントの例: IP アドレス範囲から 1 つのファイルにアクセスする

次の署名付き URL 内のカスタムポリシーの例では、UTC の 2023 年 1 月 31 日午前 10 時 00 分まで、範囲 192.0.2.0/24 の IP アドレスから、ユーザーがファイル `https://d111111abcdef8.cloudfront.net/game_download.zip` にアクセスできることを指定しています。

```
{
  "Statement": [
    {
```

```
    "Resource": "https://d111111abcdef8.cloudfront.net/game_download.zip",
    "Condition": {
      "IpAddress": {
        "AWS:SourceIp": "192.0.2.0/24"
      },
      "DateLessThan": {
        "AWS:EpochTime": 1675159200
      }
    }
  }
]
```

ポリシーステートメントの例: IP アドレス範囲からディレクトリ内のすべてのファイルにアクセスする

以下のカスタムポリシーの例では、Resource パラメータのアスタリスクワイルドカード文字 (*) が示すとおり、training ディレクトリ内のあらゆるファイルを対象とする署名付き URL を作成できます。UTC の 2023 年 1 月 31 日午前 10 時 00 分まで、範囲 192.0.2.0/24 の IP アドレスから、ユーザーはファイルにアクセスできます。

```
{
  "Statement": [
    {
      "Resource": "https://d111111abcdef8.cloudfront.net/training/*",
      "Condition": {
        "IpAddress": {
          "AWS:SourceIp": "192.0.2.0/24"
        },
        "DateLessThan": {
          "AWS:EpochTime": 1675159200
        }
      }
    }
  ]
}
```

このポリシーを使用する各署名付き URL には、次のように、特定のファイルを識別する URL があります。

<https://d111111abcdef8.cloudfront.net/training/orientation.pdf>

ポリシーステートメントの例: キーペア ID に関連付けられたすべてのファイルに 1 つの IP アドレスからアクセスする

以下のカスタムポリシーの例では、Resource パラメータのアスタリスクワイルドカード文字 (*) が示すとおり、あらゆるディストリビューションに関連付けられたあらゆるファイルを対象とする署名付き URL を作成できます。署名付き URL には、http:// ではなく https:// プロトコルを使用する必要があります。ユーザーは IP アドレス 192.0.2.10/32 を使用する必要があります。(CIDR 表記の値 192.0.2.10/32 は 1 つの IP アドレス 192.0.2.10 を参照します)。ファイルは、UTC の 2023 年 1 月 31 日午前 10 時 00 分から UTC の 2023 年 2 月 2 日午前 10 時 00 分まで使用できます。

```
{
  "Statement": [
    {
      "Resource": "https://*",
      "Condition": {
        "IpAddress": {
          "AWS:SourceIp": "192.0.2.10/32"
        },
        "DateGreaterThan": {
          "AWS:EpochTime": 1675159200
        },
        "DateLessThan": {
          "AWS:EpochTime": 1675332000
        }
      }
    }
  ]
}
```

このポリシーを使用する各署名付き URL には、特定の CloudFront ディストリビューション内の特定のファイルを識別する URL があります。次に例を示します。

<https://d111111abcdef8.cloudfront.net/training/orientation.pdf>

署名付き URL にはキーペア ID も含まれます。キーペア ID は、URL に指定されたディストリビューション (d111111abcdef8.cloudfront.net) 内の信頼されたキーグループに関連付けられる必要があります。

カスタムポリシーを使用する署名付き URL の署名の作成

カスタムポリシーを使用する署名付き URL の署名は、ハッシュ化、署名、および base64 エンコードが行われたバージョンのポリシーステートメントです。カスタムポリシーの署名を作成するには、以下の手順を実行します。

ポリシーステートメントのハッシュ化、署名、およびエンコードを行う方法の詳細および例については、以下の各資料を参照してください。

- [Linux コマンドおよび OpenSSL を使用した Base64 エンコードおよび暗号化](#)
- [署名付き URL の署名を作成するためのコード例](#)

オプション 1: カスタムポリシーを使用して署名を作成するには

1. 「[カスタムポリシーを使用する署名付き URL のポリシーステートメントを作成するには](#)」の手順で作成した JSON ポリシーステートメントを、SHA-1 ハッシュ関数と RSA を使用してハッシュ化し、署名します。空白文字が含まれていないが、まだ base64 エンコードされていないバージョンのポリシーステートメントを使用します。

ハッシュ関数に必要なプライベートキーには、対応するパブリックキーがディストリビューション内のアクティブな信頼されたキーグループにあるものを使用してください。

Note

ポリシーステートメントをハッシュ化および署名するための方法は、プログラミング言語およびプラットフォームによって異なります。サンプルコードについては、「[署名付き URL の署名を作成するためのコード例](#)」を参照してください。

2. ハッシュ化および署名された文字列から、空白文字 (タブや改行文字を含む) を削除します。
3. MIME base64 エンコーディングを使用して文字列を Base64 エンコードします。詳細については、RFC 2045, MIME (Multipurpose Internet Mail Extensions) Part One: Format of Internet Message Bodies の [Section 6.8, Base64 Content-Transfer-Encoding](#) を参照してください。
4. URL クエリ文字列内の無効な文字を有効な文字で置き換えます。次の表に無効な文字と有効な文字を示します。

無効な文字 (置換元)	有効な文字 (置換先)
+	- (ハイフン)
=	_ (下線)
/	~ (チルダ)

5. 結果の値を署名付き URL の &Signature= の後に付加し、「[カスタムポリシーを使用して署名付き URL を作成するには](#)」に戻って、署名付き URL の各部分の連結を終了します。

署名付き Cookie の使用

CloudFront 署名付き Cookie を使用すると、現在の URLs を変更しない場合や、ウェブサイトのサブスクライバーの領域にあるすべてのファイルなど、複数の制限付きファイルへのアクセスを提供する場合に、コンテンツにアクセスできるユーザーを制御できます。このトピックでは、署名付き Cookie を使用する際の考慮事項と、既定ポリシーとカスタムポリシーを使用するように署名付き Cookie を設定する方法について説明します。

トピック

- [署名付き Cookie の既定ポリシーとカスタムポリシーの選択](#)
- [署名付き Cookie の仕組み](#)
- [署名付き Cookie の悪用の防止](#)
- [は署名付き Cookie の有効期限日時をいつ CloudFront 確認しますか？](#)
- [サンプルコードおよびサードパーティーツール](#)
- [既定ポリシーを使用する署名付き Cookie の設定](#)
- [カスタムポリシーを使用する署名付き Cookie の設定](#)

署名付き Cookie の既定ポリシーとカスタムポリシーの選択

署名付き Cookie を作成する場合、Cookie の有効期間など、署名付き Cookie で制限を指定する JSON 形式のポリシーステートメントを作成します。既定ポリシーまたはカスタムポリシーを使用できます。次の表では、既定ポリシーとカスタムポリシーを比較しています。

説明	既定ポリシー	カスタムポリシー
ポリシーステートメントを複数のファイル用に再利用できる。ポリシーステートメントを再利用するには、Resource オブジェクトでワイルドカード文字を使用する必要があります。(詳しくは、「 署名付き Cookie のカスタムポリシーのポリシーステートメントで指定する値 」を参照してください)。	いいえ	はい
ユーザーがコンテンツへのアクセスを開始できる日時を指定できる	いいえ	はい (オプション)
ユーザーがコンテンツにアクセスできなくなる日時を指定できる	はい	はい
コンテンツにアクセスできるユーザーの IP アドレスまたは IP アドレス範囲を指定できる	いいえ	はい (オプション)

既定ポリシーを使用して署名付き Cookie を作成する方法については、「[既定ポリシーを使用する署名付き Cookie の設定](#)」を参照してください。

カスタムポリシーを使用して署名付き Cookie を作成する方法については、「[カスタムポリシーを使用する署名付き Cookie の設定](#)」を参照してください。

署名付き Cookie の仕組み

署名付き Cookie CloudFront の の設定方法と、ユーザーが署名付き Cookie を含むリクエストを送信したときの の CloudFront 応答の概要を次に示します。

- CloudFront デイストリビューションで、1 つ以上の信頼されたキーグループを指定します。このグループには、CloudFront が URL 署名の検証に使用できるパブリックキーが含まれています。対応するプライベートキーを使用して URL に署名します。

詳細については、「[署名付き URL と署名付き Cookie を作成できる署名者の指定](#)」を参照してください。

- ユーザーがコンテンツにアクセスできるかどうかを判断し、アクセスできる場合は、3 つの Set-Cookie ヘッダーをビューワーに送信するアプリケーションを開発します (各 Set-Cookie ヘッダーには名前と値のペアを 1 つだけ含めることができ、CloudFront 署名付き

Cookie には 3 つの名前と値のペアが必要です)。ビューワーがプライベートコンテンツをリクエストする前に、ビューワーに Set-Cookie ヘッダーを送信する必要があります。Cookie の有効期限を短く設定した場合、ユーザーがアクセスを続行できるように、以降のリクエストに対してさらに 3 つの Set-Cookie ヘッダーを送信することもできます。

通常、CloudFront デイストリビューションには少なくとも 2 つのキャッシュ動作があります。1 つは認証を必要としない動作で、もう 1 つは認証を必要としない動作です。サイトのセキュリティで保護された部分のエラーページには、ログインページへのリダイレクトまたはリンクが含まれます。

Cookie に基づいてファイルをキャッシュするようにデイストリビューションを設定する場合、署名 CloudFront 付き Cookie の属性に基づいて個別のファイルをキャッシュしないでください。

3. ユーザーがウェブサイトにサインインし、コンテンツに対して支払いをするか、またはその他のアクセスの要件を満たします。
4. アプリケーションは、レスポンスで Set-Cookie ヘッダーを返し、ビューワーは名前と値のペアを格納します。
5. ユーザーがファイルをリクエストします。

ユーザーのブラウザまたはその他のビューワーは、ステップ 4 の名前と値のペアを取得し、リクエストの Cookie ヘッダーに追加します。これが署名付き Cookie です。

6. CloudFront はパブリックキーを使用して、署名付き Cookie の署名を検証し、Cookie が改ざんされていないことを確認します。署名が無効である場合、リクエストは拒否されます。

Cookie 内の署名が有効な場合、は Cookie 内のポリシーステートメント CloudFront を確認し (既定ポリシーを使用している場合はポリシーステートメントを作成します)、リクエストがまだ有効であることを確認します。例えば、Cookie の開始日時と終了日時を指定した場合、は、アクセスを許可する期間中にユーザーがコンテンツにアクセスしようとしている CloudFront ことを確認します。

リクエストがポリシーステートメントの要件を満たしている場合、は制限されていないコンテンツと同様にコンテンツ CloudFront を提供し、ファイルがエッジキャッシュにすでに存在するかどうかを判断し、必要に応じてリクエストをオリジンに転送して、そのファイルをユーザーに返します。

署名付き Cookie の悪用の防止

Set-Cookie ヘッダーで Domain パラメータを指定する場合、同一ルートドメイン名を使用するユーザーによる潜在的なアクセスを低減できる、最も厳密な値を指定します。たとえば、apex.example.com は、特に example.com を制御しない場合は、example.com よりも優先されます。これによって、ユーザーが example.com のコンテンツにアクセスすることを防止できます。

この種類の攻撃を防ぐには、以下の作業を行います。

- Expires ヘッダーでセッション Cookie が作成されるように、Max-Age および Set-Cookie Cookie 属性を除外します。セッション Cookie は、ユーザーがブラウザを閉じたときに自動的に削除されるため、ユーザーがコンテンツに不正アクセスする可能性が低くなります。
- ビューワーがリクエストに Cookie を含める場合に Cookie が暗号化されるように、Secure 属性を含めます。
- 可能な場合、カスタムポリシーを使用してビューワーの IP アドレスを含めます。
- CloudFront-Expires 属性では、ユーザーがコンテンツにアクセスできるようにする期間に基づいて、最短で適切な有効期限の時刻を指定します。

は署名付き Cookie の有効期限日時をいつ CloudFront 確認しますか？

署名付き Cookie がまだ有効かどうかを判断するために、は HTTP リクエスト時に Cookie の有効期限日時 CloudFront を確認します。有効期限切れ時刻の直前にクライアントが大きなファイルのダウンロードを開始した場合、ダウンロード中に有効期限切れ時刻が経過してもダウンロードは完了しません。TCP 接続が中断し、有効期限切れ時刻が経過した後にクライアントがダウンロードを再開した場合、ダウンロードは失敗します。

クライアントが、ファイルを断片的に取得するレンジ GET を使用した場合、有効期限切れ時刻が経過した後に実行された GET リクエストは失敗します。レンジ GET の詳細については、「[ガオブジェクトの部分リクエスト CloudFront を処理する方法 \(範囲 GETs\)](#)」を参照してください。

サンプルコードおよびサードパーティーツール

プライベートコンテンツ用のサンプルコードは、署名付き URL の署名を作成する方法のみを示しています。ただし、署名付き Cookie の署名を作成するプロセスは非常によく似ており、サンプルコードの多くの部分は関連しています。詳細については、以下のトピックを参照してください。

- [Perl を使用して URL 署名を作成する](#)

- [PHP を使用して URL 署名を作成する](#)
- [C# と .NET Framework を使用して URL 署名を作成する](#)
- [Java を使用して URL 署名を作成する](#)

既定ポリシーを使用する署名付き Cookie の設定

既定ポリシーを使用して署名付き Cookie を設定するには、以下のステップを実行します。署名を作成するには、「[既定ポリシーを使用する署名付き Cookie の署名の作成](#)」を参照してください。

既定ポリシーを使用して署名付き Cookie を設定するには

1. .NET または Java を使用して署名付き Cookie を作成しており、キーペアのプライベートキーをデフォルトの .pem 形式から .NET または Java 対応の形式に変更していない場合は、それを変換します。詳細については、「[プライベートキーの形式の変更 \(.NET および Java のみ\)](#)」を参照してください。
2. 承認されたビューワーに 3 つの Set-Cookie ヘッダーを送信するアプリケーションをプログラムします。各 Set-Cookie ヘッダーには名前と値のペアを 1 つだけ含めることができ、CloudFront 署名付き Cookie では 3 つの名前と値のペアが必要であるため、3 つの Set-Cookie ヘッダーが必要です。名前と値のペアは、CloudFront-Expires、CloudFront-Signature、および CloudFront-Key-Pair-Id です。アクセスを制御するファイルに対してユーザーが最初のリクエストを行う前に、値がビューワーに存在している必要があります。

Note

一般的に、Expires 属性と Max-Age 属性を除外することをお勧めします。これらの属性を除外すると、ユーザーがブラウザを閉じたときに、ブラウザで Cookie が削除されるため、ユーザーがコンテンツに不正アクセスする可能性が低くなります。詳細については、「[署名付き Cookie の悪用の防止](#)」を参照してください。

Cookie の属性の名前では、大文字と小文字が区別されます。

改行は、属性を判読しやすくするためにのみ含まれています。

```
Set-Cookie:  
CloudFront-Expires=date and time in Unix time format (in seconds) and Coordinated  
Universal Time (UTC);  
Domain=optional domain name;
```

```
Path=/optional directory path;  
Secure;  
HttpOnly  
  
Set-Cookie:  
CloudFront-Signature=hashed and signed version of the policy statement;  
Domain=optional domain name;  
Path=/optional directory path;  
Secure;  
HttpOnly  
  
Set-Cookie:  
CloudFront-Key-Pair-Id=public key ID for the CloudFront public key whose  
corresponding private key you're using to generate the signature;  
Domain=optional domain name;  
Path=/optional directory path;  
Secure;  
HttpOnly
```

(オプション) Domain

リクエストされたファイルのドメイン名。Domain 属性を指定しない場合、デフォルト値は URL のドメイン名で、指定されたドメイン名にのみ適用され、サブドメインには適用されません。Domain 属性を指定する場合、サブドメインにも適用されます。ドメイン名の先頭のドット (例えば、Domain=.example.com) はオプションです。さらに、Domain 属性を指定する場合は、URL のドメイン名と Domain 属性の値が一致している必要があります。

がディストリビューションに CloudFront 割り当てたドメイン名を指定できます。例えば、d111111abcdef8.cloudfront.net です。ただし、ドメイン名に *.cloudfront.net を指定することはできません。

URL で代替ドメイン名 (example.com など) を使用する場合は、Domain 属性を指定するかどうかにかかわらず、代替ドメイン名をディストリビューションに追加する必要があります。詳細については、トピック「[ディストリビューションを作成または更新する場合に指定する値](#)」の「[代替ドメイン名 \(CNAME\)](#)」を参照してください。

(オプション) Path

リクエストされたファイルのパス。Path 属性を指定しない場合、デフォルト値は URL のパスです。

Secure

リクエストを送信する前に、ビューワーが Cookie を暗号化することを要求します。Cookie 属性が man-in-the-middle 攻撃から保護されるように、HTTPS 接続で Set-Cookie ヘッダーを送信することをお勧めします。

HttpOnly

ビューワーが HTTP または HTTPS リクエストでのみ Cookie を送信することを要求します。

CloudFront-Expires

有効期限切れ日時を Unix 時間形式 (秒単位) および協定世界時 (UTC) で指定します。たとえば、UTC の 2013 年 1 月 1 日午前 10 時 00 分は、Unix 時間形式の 1357034400 に変換されます。エポック時間を使用するには、日付に 2147483647 (2038 年 1 月 19 日 03:14:07 UTC) より前の 32 ビット整数を使用します。UTC の詳細については、RFC 3339, Date and Time on the Internet: Timestamps (<https://tools.ietf.org/html/rfc3339>) を参照してください。

CloudFront-Signature

ハッシュ化され、署名された base64 エンコードバージョンの JSON ポリシーステートメント。詳細については、「[既定ポリシーを使用する署名付き Cookie の署名の作成](#)」を参照してください。

CloudFront-Key-Pair-Id

CloudFront パブリックキーの ID。例: K2JCJMDEHXQW5F。パブリックキー ID は、署名付き URL の検証に使用する CloudFront パブリックキーをに伝えます。CloudFront は、署名内の情報をポリシーステートメント内の情報と比較し、URL が改ざんされていないことを確認します。

このパブリックキーは、ディストリビューションの信頼された署名者であるキーグループに属している必要があります。詳細については、「[署名付き URL と署名付き Cookie を作成できる署名者の指定](#)」を参照してください。

以下は、ファイルの URL のディストリビューションに関連付けられたドメイン名を使用する場合の、1 つの署名付き Cookie の Set-Cookie ヘッダーの例です。

```
Set-Cookie: CloudFront-Expires=1426500000; Domain=d1111111abcdef8.cloudfront.net; Path=/images/*; Secure; HttpOnly
```

```
Set-Cookie: CloudFront-Signature=yXrSIgyQoeE4FBI4eMKF6ho~CA8_  
Domain=d111111abcdef8.cloudfront.net; Path=/images/*; Secure; HttpOnly  
Set-Cookie: CloudFront-Key-Pair-Id=K2JJCJMDEHXQW5F;  
Domain=d111111abcdef8.cloudfront.net; Path=/images/*; Secure; HttpOnly
```

以下は、ファイルの URL に代替ドメイン名 example.org を使用している場合の、1 つの署名付き Cookie の Set-Cookie ヘッダーの例です。

```
Set-Cookie: CloudFront-Expires=1426500000; Domain=example.org; Path=/images/*; Secure;  
HttpOnly  
Set-Cookie: CloudFront-Signature=yXrSIgyQoeE4FBI4eMKF6ho~CA8_; Domain=example.org;  
Path=/images/*; Secure; HttpOnly  
Set-Cookie: CloudFront-Key-Pair-Id=K2JJCJMDEHXQW5F; Domain=example.org; Path=/images/*;  
Secure; HttpOnly
```

URL で代替ドメイン名 (example.com など) を使用する場合は、Domain 属性を指定するかどうかにかかわらず、代替ドメイン名をディストリビューションに追加する必要があります。詳細については、トピック「[ディストリビューションを作成または更新する場合に指定する値](#)」の「[代替ドメイン名 \(CNAME\)](#)」を参照してください。

既定ポリシーを使用する署名付き Cookie の署名の作成

既定ポリシーを使用する署名付き Cookie の署名を作成するには、以下を実行します。

1. ポリシーステートメントを作成します。[既定ポリシーを使用する署名付き Cookie のポリシーステートメントの作成](#) を参照してください。
2. ポリシーステートメントに署名して、署名を作成します。[既定ポリシーを使用する署名付き Cookie の署名を作成するためのポリシーステートメントの署名](#) を参照してください。

既定ポリシーを使用する署名付き Cookie のポリシーステートメントの作成

既定ポリシーを使用する署名付き Cookie を設定した場合、CloudFront-Signature 属性は、ポリシーステートメントのハッシュ化および署名されたバージョンです。カスタムポリシーを使用する署名付き Cookie とは異なり、既定ポリシーを使用する署名付き Cookie では、Set-Cookie ヘッダーにポリシーステートメントを含めません。ポリシーステートメントを作成するには、以下の手順を実行します。

既定ポリシーを使用する署名付き Cookie のポリシーステートメントを作成するには

1. 以下の JSON 形式および UTF-8 文字エンコードを使用してポリシーステートメントを構築します。すべての句読点および他のリテラル値を、指定されたとおりに正確に含めます。Resource および DateLessThan パラメータの詳細については、「[署名付き Cookie の既定ポリシーのポリシーステートメントで指定する値](#)」を参照してください。

```
{
  "Statement": [
    {
      "Resource": "base URL or stream name",
      "Condition": {
        "DateLessThan": {
          "AWS:EpochTime": ending date and time in Unix time format and
          UTC
        }
      }
    }
  ]
}
```

2. ポリシーステートメントからすべての空白文字 (タブと改行文字を含む) を削除します。アプリケーションコード内の文字列にエスケープ文字を含めることが必要になる場合があります。

署名付き Cookie の既定ポリシーのポリシーステートメントで指定する値

既定ポリシーのポリシーステートメントを作成する場合、以下の値を指定します。

リソース

クエリ文字列 (存在する場合) を含むベース URL。以下に例を示します。

```
https://d1111111abcdef8.cloudfront.net/images/horizon.jpg?
size=large&license=yes
```

Resource の日付形式は 1 つだけ指定できます。

次の点に注意してください。

- プロトコル - 値は http:// または https:// で始まっている必要があります。
- クエリ文字列パラメータ - クエリ文字列パラメータがない場合は、疑問符を省略します。

- 代替ドメイン名 – URL で代替ドメイン名 (CNAME) を指定する場合は、ウェブページまたはアプリケーション内のファイルを参照するときに代替ドメイン名を指定する必要があります。ファイルの Amazon S3 URL を指定しないでください。

DateLessThan

URL の有効期限切れ日時。Unix 時間形式 (秒単位) および協定世界時 (UTC) で指定します。値を引用符で囲まないでください。

たとえば、UTC の 2015 年 3 月 16 日午前 10 時 00 分は、UNIX 時間形式の 1426500000 に変換されます。

この値は、CloudFront-Expires ヘッダーの Set-Cookie 属性の値と一致する必要があります。値を引用符で囲まないでください。

詳細については、「[は署名付き Cookie の有効期限日時をいつ CloudFront 確認しますか？](#)」を参照してください。

既定ポリシーのポリシーステートメントの例

署名付き Cookie 内で以下のポリシーステートメントの例を使用すると、ユーザーは、UTC の 2015 年 3 月 16 日午前 10 時 00 分までファイル `https://d111111abcdef8.cloudfront.net/horizon.jpg` にアクセスできます。

```
{
  "Statement": [
    {
      "Resource": "https://d111111abcdef8.cloudfront.net/horizon.jpg?
size=large&license=yes",
      "Condition": {
        "DateLessThan": {
          "AWS:EpochTime": 1426500000
        }
      }
    }
  ]
}
```

既定ポリシーを使用する署名付き Cookie の署名を作成するためのポリシーステートメントの署名

CloudFront-Signature ヘッダーの Set-Cookie 属性の値を作成するには、「[既定ポリシーを使用する署名付き Cookie のポリシーステートメントを作成するには](#)」で作成したポリシーステートメントをハッシュ化して署名します。

ポリシーステートメントのハッシュ化、署名、およびエンコードを行う方法の詳細および例については、以下のトピックを参照してください。

- [Linux コマンドおよび OpenSSL を使用した Base64 エンコードおよび暗号化](#)
- [署名付き URL の署名を作成するためのコード例](#)

既定ポリシーを使用して署名付き Cookie の署名を作成するには

1. 「[既定ポリシーを使用する署名付き Cookie のポリシーステートメントを作成するには](#)」の手順で作成したポリシーステートメントを、SHA-1 ハッシュ関数と RSA を使用してハッシュ化し、署名します。空白を含まないバージョンのポリシーステートメントを使用します。

ハッシュ関数に必要なプライベートキーには、対応するパブリックキーがディストリビューション内のアクティブな信頼されたキーグループにあるものを使用してください。

Note

ポリシーステートメントをハッシュ化および署名するための方法は、プログラミング言語およびプラットフォームによって異なります。サンプルコードについては、「[署名付き URL の署名を作成するためのコード例](#)」を参照してください。

2. ハッシュ化および署名された文字列から、空白文字 (タブや改行文字を含む) を削除します。
3. MIME base64 エンコーディングを使用して文字列を Base64 エンコードします。詳細については、RFC 2045, MIME (Multipurpose Internet Mail Extensions) Part One: Format of Internet Message Bodies の [Section 6.8, Base64 Content-Transfer-Encoding](#) を参照してください。
4. URL クエリ文字列内の無効な文字を有効な文字で置き換えます。次の表に無効な文字と有効な文字を示します。

無効な文字 (置換元)	有効な文字 (置換先)
+	-(ハイフン)

無効な文字 (置換元)	有効な文字 (置換先)
=	_ (下線)
/	~ (チルダ)

5. 結果の値を、Set-Cookie の名前と値のペアの CloudFront-Signature ヘッダーに含めます。次に、「[既定ポリシーを使用して署名付き Cookie を設定するには](#)」に戻り、Set-Cookie の CloudFront-Key-Pair-Id ヘッダーを追加します。

カスタムポリシーを使用する署名付き Cookie の設定

トピック

- [カスタムポリシーを使用する署名付き Cookie のポリシーステートメントの作成](#)
- [カスタムポリシーを使用する署名付き Cookie のポリシーステートメントの例](#)
- [カスタムポリシーを使用する署名付き Cookie の署名の作成](#)

カスタムポリシーを使用する署名付き Cookie を設定するには、以下の手順を実行します。

カスタムポリシーを使用して署名付き Cookie を設定するには

1. .NET または Java を使用して署名付き URL を作成しており、キーペアのプライベートキーをデフォルトの .pem 形式から .NET または Java 対応の形式に変更していない場合は、それを変換します。詳細については、「[プライベートキーの形式の変更 \(.NET および Java のみ\)](#)」を参照してください。
2. 承認されたビューワーに 3 つの Set-Cookie ヘッダーを送信するアプリケーションをプログラムします。各 Set-Cookie ヘッダーには名前と値のペアを 1 つだけ含めることができ、CloudFront 署名付き Cookie には 3 つの名前と値のペアが必要なため、3 つの Set-Cookie ヘッダーが必要です。名前と値のペアは、CloudFront-Policy、CloudFront-Signature、および CloudFront-Key-Pair-Id です。アクセスを制御するファイルに対してユーザーが最初のリクエストを行う前に、値がビューワーに存在している必要があります。

Note

一般的に、Expires 属性と Max-Age 属性を除外することをお勧めします。これにより、ユーザーがブラウザを閉じたときに、ブラウザで Cookie が削除されるため、ユー

ザーがコンテンツに不正アクセスする可能性が低くなります。詳細については、「[署名付き Cookie の悪用の防止](#)」を参照してください。

Cookie の属性の名前では、大文字と小文字が区別されます。

改行は、属性を判読しやすくするためにのみ含まれています。

```
Set-Cookie:  
CloudFront-Policy=base64 encoded version of the policy statement;  
Domain=optional domain name;  
Path=/optional directory path;  
Secure;  
HttpOnly  
  
Set-Cookie:  
CloudFront-Signature=hashed and signed version of the policy statement;  
Domain=optional domain name;  
Path=/optional directory path;  
Secure;  
HttpOnly  
  
Set-Cookie:  
CloudFront-Key-Pair-Id=public key ID for the CloudFront public key whose  
corresponding private key you're using to generate the signature;  
Domain=optional domain name;  
Path=/optional directory path;  
Secure;  
HttpOnly
```

(オプション) Domain

リクエストされたファイルのドメイン名。Domain 属性を指定しない場合、デフォルト値は URL のドメイン名で、指定されたドメイン名にのみ適用され、サブドメインには適用されません。Domain 属性を指定する場合、サブドメインにも適用されます。ドメイン名の先頭のドット (例えば、Domain=.example.com) はオプションです。さらに、Domain 属性を指定する場合は、URL のドメイン名と Domain 属性の値が一致している必要があります。

がディストリビューションに CloudFront 割り当てたドメイン名を指定できます。例えば、d1111111abcdef8.cloudfront.net です。ただし、ドメイン名に *.cloudfront.net を指定することはできません。

URL で代替ドメイン名 (example.com など) を使用する場合は、Domain 属性を指定するかどうかにかかわらず、代替ドメイン名をディストリビューションに追加する必要があります。詳細については、トピック「[ディストリビューションを作成または更新する場合に指定する値](#)」の「[代替ドメイン名 \(CNAME\)](#)」を参照してください。

(オプション) Path

リクエストされたファイルのパス。Path 属性を指定しない場合、デフォルト値は URL のパスです。

Secure

リクエストを送信する前に、ビューワーが Cookie を暗号化することを要求します。Cookie 属性が man-in-the-middle 攻撃から保護されるように、HTTPS 接続で Set-Cookie ヘッダーを送信することをお勧めします。

HttpOnly

ビューワーが HTTP または HTTPS リクエストでのみ Cookie を送信することを要求します。

CloudFront-Policy

空白文字が削除され、base64 エンコードされた、JSON 形式のポリシーステートメント。詳細については、「[カスタムポリシーを使用する署名付き Cookie の署名の作成](#)」を参照してください。

ポリシーステートメントは、署名付き Cookie によってユーザーに許可されるアクセスをコントロールします。これには、ユーザーがアクセスできるファイル、有効期限切れ日時、URL が有効になる日時 (オプション)、ファイルへのアクセスが許可されている IP アドレスまたは IP アドレス範囲 (オプション) が含まれます。

CloudFront-Signature

ハッシュ化され、署名された base64 エンコードバージョンの JSON ポリシーステートメント。詳細については、「[カスタムポリシーを使用する署名付き Cookie の署名の作成](#)」を参照してください。

CloudFront-Key-Pair-Id

CloudFront パブリックキーの ID。例: K2JJCJMDEHXQW5F。パブリックキー ID は、署名付き URL の検証に使用する CloudFront パブリックキーを に伝えます。CloudFront は、署名内の情報をポリシーステートメント内の情報と比較し、URL が改ざんされていないことを確認します。

このパブリックキーは、ディストリビューションの信頼された署名者であるキーグループに属している必要があります。詳細については、「[署名付き URL と署名付き Cookie を作成できる署名者の指定](#)」を参照してください。

ファイルの URL のディストリビューションに関連付けられたドメイン名を使用する場合の、1 つの署名付き Cookie の Set-Cookie ヘッダーの例

```
Set-Cookie: CloudFront-  
Policy=eyJTdGF0ZW11bnQiO1t7I1Jlc291cmNlIjoiaHR0cDovL2QxMTEyMTFhYmNkZWY4LmNsb3VhZnJvbnQubmV0L2dh  
Domain=d111111abcdef8.cloudfront.net; Path=/; Secure; HttpOnly  
Set-Cookie: CloudFront-Signature=dtKhpJ3aUYxqDIwepczPiDb9NXQ_  
Domain=d111111abcdef8.cloudfront.net; Path=/; Secure; HttpOnly  
Set-Cookie: CloudFront-Key-Pair-Id=K2JJCJMDEHXQW5F;  
Domain=d111111abcdef8.cloudfront.net; Path=/; Secure; HttpOnly
```

ファイルの URL に代替ドメイン名 example.org を使用している場合の、1 つの署名付き Cookie の Set-Cookie ヘッダーの例:

```
Set-Cookie: CloudFront-  
Policy=eyJTdGF0ZW11bnQiO1t7I1Jlc291cmNlIjoiaHR0cDovL2QxMTEyMTFhYmNkZWY4LmNsb3VhZnJvbnQubmV0L2dh  
Domain=example.org; Path=/; Secure; HttpOnly  
Set-Cookie: CloudFront-Signature=dtKhpJ3aUYxqDIwepczPiDb9NXQ_; Domain=example.org;  
Path=/; Secure; HttpOnly  
Set-Cookie: CloudFront-Key-Pair-Id=K2JJCJMDEHXQW5F; Domain=example.org; Path=/; Secure;  
HttpOnly
```

URL で代替ドメイン名 (example.com など) を使用する場合は、Domain 属性を指定するかどうかにかかわらず、代替ドメイン名をディストリビューションに追加する必要があります。詳細については、トピック「[ディストリビューションを作成または更新する場合に指定する値](#)」の「[代替ドメイン名 \(CNAME\)](#)」を参照してください。

カスタムポリシーを使用する署名付き Cookie のポリシーステートメントの作成

カスタムポリシーのポリシーステートメントを作成するには、以下の手順を実行します。さまざまな方法でファイルへのアクセスを制御するポリシーステートメントのいくつかの例については、「[カスタムポリシーを使用する署名付き Cookie のポリシーステートメントの例](#)」を参照してください。

カスタムポリシーを使用する署名付き Cookie のポリシーステートメントを作成するには

1. 以下の JSON 形式を使用してポリシーステートメントを構築します。

```
{
  "Statement": [
    {
      "Resource": "URL of the file",
      "Condition": {
        "DateLessThan": {
          "AWS:EpochTime": required ending date and time in Unix time
          format and UTC
        },
        "DateGreaterThan": {
          "AWS:EpochTime": optional beginning date and time in Unix time
          format and UTC
        },
        "IpAddress": {
          "AWS:SourceIp": "optional IP address"
        }
      }
    }
  ]
}
```

次の点に注意してください。

- 1つのステートメントのみを含めることができます。
- UTF-8 文字エンコードを使用します。
- すべての句読点およびパラメータ名を、指定されたとおりに正確に含めます。パラメータ名の省略形は受け付けられません。
- Condition セクションのパラメータの順序は問題ではありません。

- Resource、DateLessThan、DateGreaterThan、および IPAddress の値については、「[署名付き Cookie のカスタムポリシーのポリシーステートメントで指定する値](#)」を参照してください。
2. ポリシーステートメントからすべての空白文字 (タブと改行文字を含む) を削除します。アプリケーションコード内の文字列にエスケープ文字を含めることが必要になる場合があります。
 3. MIME base64 エンコーディングを使用してポリシーステートメントを Base64 エンコードします。詳細については、RFC 2045, MIME (Multipurpose Internet Mail Extensions) Part One: Format of Internet Message Bodies の [Section 6.8, Base64 Content-Transfer-Encoding](#) を参照してください。
 4. URL クエリ文字列内の無効な文字を有効な文字で置き換えます。次の表に無効な文字と有効な文字を示します。

無効な文字 (置換元)	有効な文字 (置換先)
+	- (ハイフン)
=	_ (下線)
/	~ (チルダ)

5. 結果の値を、Set-Cookie ヘッダーの CloudFront-Policy= の後に含めます。
6. ポリシーステートメントのハッシュ化、署名、および base64 エンコードを行って、Set-Cookie 用に CloudFront-Signature ヘッダーの署名を作成します。詳細については、「[カスタムポリシーを使用する署名付き Cookie の署名の作成](#)」を参照してください。

署名付き Cookie のカスタムポリシーのポリシーステートメントで指定する値

カスタムポリシーのポリシーステートメントを作成する場合、以下の値を指定します。

リソース

クエリ文字列 (存在する場合) を含むベース URL。

```
https://d1111111abcdef8.cloudfront.net/images/horizon.jpg?
size=large&license=yes
```

⚠ Important

Resource パラメータを省略した場合、ユーザーは、署名付き URL の作成に使用するキーペアに関連付けられたあらゆるディストリビューションに関連付けられるすべてのファイルにアクセスできます。

Resource の日付形式は 1 つだけ指定できます。

次の点に注意してください。

- プロトコル – 値は `http://` または `https://` で始まっている必要があります。
- クエリ文字列パラメータ – クエリ文字列パラメータがない場合は、疑問符を省略します。
- ワイルドカード – 0 個以上の文字に一致するワイルドカード文字 (*)、または 1 つの文字に一致するワイルドカード文字 (?) を使用できます。文字列のどこにでも含めることができます。次に例を示します。この値は、

```
https://d111111abcdef8.cloudfront.net/*game_download.zip*
```

たとえば、次のファイルを含みます。

- `https://d111111abcdef8.cloudfront.net/game_download.zip`
- `https://d111111abcdef8.cloudfront.net/example_game_download.zip?license=yes`
- `https://d111111abcdef8.cloudfront.net/test_game_download.zip?license=temp`
- 代替ドメイン名 – URL で代替ドメイン名 (CNAME) を指定する場合は、ウェブページまたはアプリケーション内のファイルを参照するときに代替ドメイン名を指定する必要があります。ファイルの Amazon S3 URL を指定しないでください。

DateLessThan

URL の有効期限切れ日時。Unix 時間形式 (秒単位) および協定世界時 (UTC) で指定します。値を引用符で囲まないでください。

たとえば、UTC の 2015 年 3 月 16 日午前 10 時 00 分は、UNIX 時間形式の 1426500000 に変換されます。

詳細については、「[は署名付き Cookie の有効期限日時をいつ CloudFront 確認しますか？](#)」を参照してください。

DateGreaterThan (オプション)

オプションの URL 開始日時。Unix 時間形式 (秒単位) および協定世界時 (UTC) で指定します。ユーザーは、指定された日時以前のファイルにアクセスすることはできません。値を引用符で囲まないでください。

IpAddress (オプション)

GET リクエストを実行するクライアントの IP アドレス。次の点に注意してください。

- ファイルへのアクセスをすべての IP アドレスに許可するには、IpAddress パラメータを省略します。
- IP アドレスまたは IP アドレス範囲を 1 つ指定できます。たとえば、2 つの別々の範囲のどちらかにクライアントの IP アドレスが入っている場合にアクセスを許可するようなポリシーを設定することはできません。
- 1 つの IP アドレスからのアクセスを許可するには、以下のように指定します。

```
"IPv4 IP ####/32"
```

- IP アドレス範囲は標準の IPv4 CIDR 形式 (192.0.2.0/24 など) で指定する必要があります。詳細については、RFC 4632, Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan (<https://tools.ietf.org/html/rfc4632>) を参照してください。

Important

IPv6 形式の IP アドレス (例: 2001:0db8:85a3::8a2e:0370:7334) はサポートされていません。

IpAddress を含むカスタムポリシーを使用する場合、ディストリビューションで IPv6 は有効にしません。一部のコンテンツへのアクセスを IP アドレスによって制限し、他のコンテンツで IPv6 リクエストをサポートする場合、2 つのディストリビューションを作成します。詳細については、トピック「[ディストリビューションを作成または更新する場合に指定する値](#)」の「[IPv6 を有効にする](#)」を参照してください。

カスタムポリシーを使用する署名付き Cookie のポリシーステートメントの例

以下のポリシーステートメントの例は、特定のファイル、ディレクトリ内のすべてのファイル、またはキーペア ID に関連付けられたすべてのファイルへのアクセスを制御する方法を示しています。ま

た、この例は、個々の IP アドレスまたは IP アドレス範囲からのアクセスを制御する方法、および指定された日時以降にユーザーが署名付き Cookie を使用することを禁止する方法も示しています。

この例のいずれかをコピーして貼り付ける場合は、すべての空白文字 (タブと改行文字を含む) を削除し、適用可能な値を独自の値で置き換えて、右中かっこ (}) の後に改行文字を含めます。

詳細については、「[署名付き Cookie のカスタムポリシーのポリシーステートメントで指定する値](#)」を参照してください。

トピック

- [ポリシーステートメントの例: IP アドレス範囲から 1 つのファイルにアクセスする](#)
- [ポリシーステートメントの例: IP アドレス範囲からディレクトリ内のすべてのファイルにアクセスする](#)
- [ポリシーステートメントの例: キーペア ID に関連付けられたすべてのファイルに 1 つの IP アドレスからアクセスする](#)

ポリシーステートメントの例: IP アドレス範囲から 1 つのファイルにアクセスする

次の署名付き Cookie 内のカスタムポリシーの例では、UTC の 2023 年 1 月 1 日午前 10 時 00 分まで、範囲 192.0.2.0/24 の IP アドレスから、ユーザーがファイル `https://d111111abcdef8.cloudfront.net/game_download.zip` にアクセスできることを指定しています。

```
{
  "Statement": [
    {
      "Resource": "https://d111111abcdef8.cloudfront.net/game_download.zip",
      "Condition": {
        "IpAddress": {
          "AWS:SourceIp": "192.0.2.0/24"
        },
        "DateLessThan": {
          "AWS:EpochTime": 1357034400
        }
      }
    }
  ]
}
```

ポリシーステートメントの例: IP アドレス範囲からディレクトリ内のすべてのファイルにアクセスする

以下のカスタムポリシーの例では、Resource パラメータの * ワイルドカード文字が示すとおり、training ディレクトリ内のあらゆるファイルを対象とする署名付き Cookie を作成できます。UTC の 2013 年 1 月 1 日午前 10 時 00 分まで、範囲 192.0.2.0/24 の IP アドレスから、ユーザーはファイルにアクセスできます。

```
{
  "Statement": [
    {
      "Resource": "https://d111111abcdef8.cloudfront.net/training/*",
      "Condition": {
        "IpAddress": {
          "AWS:SourceIp": "192.0.2.0/24"
        },
        "DateLessThan": {
          "AWS:EpochTime": 1357034400
        }
      }
    }
  ]
}
```

このポリシーを使用する各署名付き Cookie には、たとえば次のように、特定のファイルを識別するベース URL が含まれます。

<https://d111111abcdef8.cloudfront.net/training/orientation.pdf>

ポリシーステートメントの例: キーペア ID に関連付けられたすべてのファイルに 1 つの IP アドレスからアクセスする

以下のカスタムポリシーの例では、Resource パラメータの * ワイルドカード文字が示すとおり、あらゆるディストリビューションに関連付けられたあらゆるファイルを対象とする署名付き Cookie を設定できます。ユーザーは IP アドレス 192.0.2.10/32 を使用する必要があります。(CIDR 表記の値 192.0.2.10/32 は 1 つの IP アドレス 192.0.2.10 を参照します)。ファイルは、UTC の 2013 年 1 月 1 日午前 10 時 00 分から UTC の 2013 年 1 月 2 日午前 10 時 00 分まで使用できます。

```
{
  "Statement": [
    {
      "Resource": "https://*",
```

```
    "Condition": {
      "IpAddress": {
        "AWS:SourceIp": "192.0.2.10/32"
      },
      "DateGreaterThan": {
        "AWS:EpochTime": 1357034400
      },
      "DateLessThan": {
        "AWS:EpochTime": 1357120800
      }
    }
  }
]
```

このポリシーを使用する各署名付き Cookie には、特定の CloudFront デイストリビューション内の特定のファイルを識別するベース URL が含まれています。次に例を示します。

<https://d1111111abcdef8.cloudfront.net/training/orientation.pdf>

署名付き Cookie にはキーペア ID も含まれます。キーペア ID は、ベース URL に指定されたデイストリビューション (d1111111abcdef8.cloudfront.net) 内の信頼されたキーグループに関連付けられる必要があります。

カスタムポリシーを使用する署名付き Cookie の署名の作成

カスタムポリシーを使用する署名付き Cookie の署名は、ハッシュ化、署名、および base64 エンコードが行われたバージョンのポリシーステートメントです。

ポリシーステートメントのハッシュ化、署名、およびエンコードを行う方法の詳細および例については、以下の各資料を参照してください。

- [Linux コマンドおよび OpenSSL を使用した Base64 エンコードおよび暗号化](#)
- [署名付き URL の署名を作成するためのコード例](#)

カスタムポリシーを使用して署名付き Cookie の署名を作成するには

1. 「[カスタムポリシーを使用する署名付き URL のポリシーステートメントを作成するには](#)」の手順で作成した JSON ポリシーステートメントを、SHA-1 ハッシュ関数と RSA を使用してハッシュ化し、署名します。空白文字が含まれていないが、まだ base64 エンコードされていないバージョンのポリシーステートメントを使用します。

ハッシュ関数に必要なプライベートキーには、対応するパブリックキーがディストリビューション内のアクティブな信頼されたキーグループにあるものを使用してください。

Note

ポリシーステートメントをハッシュ化および署名するための方法は、プログラミング言語およびプラットフォームによって異なります。サンプルコードについては、「[署名付き URL の署名を作成するためのコード例](#)」を参照してください。

- ハッシュ化および署名された文字列から、空白文字 (タブや改行文字を含む) を削除します。
- MIME base64 エンコーディングを使用して文字列を Base64 エンコードします。詳細については、RFC 2045, MIME (Multipurpose Internet Mail Extensions) Part One: Format of Internet Message Bodies の [Section 6.8, Base64 Content-Transfer-Encoding](#) を参照してください。
- URL クエリ文字列内の無効な文字を有効な文字で置き換えます。次の表に無効な文字と有効な文字を示します。

無効な文字 (置換元)	有効な文字 (置換先)
+	- (ハイフン)
=	_ (下線)
/	~ (チルダ)

- 結果の値を、Set-Cookie の名前と値のペアの CloudFront-Signature= ヘッダーに含めて、「[カスタムポリシーを使用して署名付き Cookie を設定するには](#)」に戻り、Set-Cookie の CloudFront-Key-Pair-Id ヘッダーを追加します。

Linux コマンドおよび OpenSSL を使用した Base64 エンコードおよび暗号化

次の Linux コマンドラインのコマンドおよび OpenSSL を使用して、ポリシーステートメントをハッシュ化して署名します。次に、署名を base64 エンコードし、URL クエリ文字列パラメータでの無効な文字を有効な文字置き換えます。

OpenSSL の詳細については、「<https://www.openssl.org>」にアクセスしてください。

```
❶  
cat policy |  
❷  
tr -d "\n" | tr -d " \t\n\r" |  
❸  
openssl sha1 -sign private_key.pem |  
❹  
openssl base64 -A |  
❺  
tr -- '+=/' '-_~'
```

各パラメータの意味は次のとおりです。

❶
cat が policy ファイルを読み取ります。

❷
tr -d "\n" | tr -d " \t\n\r" によって追加された空白や改行文字が cat で削除されます。

❸
OpenSSL は、SHA-1 を使用してファイルをハッシュ化し、RSA とプライベートキーファイル を使用してファイルに署名します。private_key.pem

❹
OpenSSL は、ハッシュ化および署名されたポリーステートメントを base64 エンコードします。

❺
tr は、URL クエリ文字列パラメータの無効な文字を有効な文字で置き換えます。

いくつかのプログラミング言語での署名の作成方法を示すコード例については、「[署名付き URL の署名を作成するためのコード例](#)」を参照してください。

署名付き URL の署名を作成するためのコード例

このセクションには、署名付き URL の署名の作成方法を示す、ダウンロード可能なアプリケーションの例が含まれます。例は、Perl、PHP、C#、および Java で使用できます。任意の例を使用して、署名付き URL を作成できます。Perl スクリプトは Linux および macOS プラットフォームで実行さ

れます。PHP の例は、PHP が実行されているあらゆるサーバーで動作します。C# の例では、.NET Framework が使用されます。

JavaScript (Node.js) のコード例については、AWSデベロッパーブログの「[Creating Amazon CloudFront Signed URLs in Node.js](#)」を参照してください。

Python のコード例については、AWS SDK for [Python \(Boto3\) API リファレンスの「Amazon の署名付き URL の生成 CloudFront](#)」と、Boto3 GitHub リポジトリの「[このサンプルコード](#)」を参照してください。

トピック

- [Perl を使用して URL 署名を作成する](#)
- [PHP を使用して URL 署名を作成する](#)
- [C# と .NET Framework を使用して URL 署名を作成する](#)
- [Java を使用して URL 署名を作成する](#)

Perl を使用して URL 署名を作成する

このセクションには、プライベートコンテンツの署名を作成するために使用できる Linux/Mac プラットフォームの Perl スクリプトが含まれます。署名を作成するには、CloudFront URL、署名者のプライベートキーへのパス、キー ID、および URL の有効期限を指定するコマンドライン引数を使用してスクリプトを実行します。このツールでは、署名付き URL のデコードを行うこともできます。

Note

URL 署名の作成は、署名付き URL を使用してプライベートコンテンツを供給するためのプロセスの 1 パートにすぎません。end-to-end プロセスの詳細については、「[署名付き URL の使用](#)」を参照してください。

トピック

- [署名付き URL を作成する Perl スクリプトのソース](#)

署名付き URL を作成する Perl スクリプトのソース

次の Perl ソースコードを使用して、の署名付き URL を作成できます CloudFront。コード内のコメントに、コマンドラインスイッチおよびこのツールの各種機能の情報が含まれています。

```
#!/usr/bin/perl -w

# Copyright 2008 Amazon Technologies, Inc. Licensed under the Apache License, Version
  2.0 (the "License");
# you may not use this file except in compliance with the License. You may obtain a
  copy of the License at:
#
# https://aws.amazon.com/apache2.0
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
  KIND, either express or implied.
# See the License for the specific language governing permissions and limitations under
  the License.

=head1 cfsign.pl

cfsign.pl - A tool to generate and verify Amazon CloudFront signed URLs

=head1 SYNOPSIS

This script uses an existing RSA key pair to sign and verify Amazon CloudFront signed
URLs

View the script source for details as to which CPAN packages are required beforehand.

For help, try:

cfsign.pl --help

URL signing examples:

cfsign.pl --action encode --url https://images.my-website.com/gallery1.zip --policy
  sample_policy.json --private-key privkey.pem --key-pair-id mykey

cfsign.pl --action encode --url https://images.my-website.com/gallery1.zip --expires
  1257439868 --private-key privkey.pem --key-pair-id mykey

URL decode example:

cfsign.pl --action decode --url "http://mydist.cloudfront.net/?Signature=AG0-
  PgXkYo99MkJFHvjfGXjG1QDEXeaDb4Qtzmy85wqyJjK7eKojQWa4BCRcow__&Policy=eyJTdGF0ZW11bnQiO1t7I1Jlc29
  Pair-Id=mykey"
```

To generate an RSA key pair, you can use `openssl` and the following commands:

```
# Generate a 2048 bit key pair
openssl genrsa -out private-key.pem 2048
openssl rsa -in private-key.pem -pubout -out public-key.pem
```

=head1 OPTIONS

=over 8

=item B<--help>

Print a help message and exits.

=item B<--action> [action]

The action to execute. action can be one of:

- encode - Generate a signed URL (using a canned policy or a user policy)
- decode - Decode a signed URL

=item B<--url>

The URL to en/decode

=item B<--stream>

The stream to en/decode

=item B<--private-key>

The path to your private key.

=item B<--key-pair-id>

The key pair identifier.

=item B<--policy>

The CloudFront policy document.

=item B<--expires>

The Unix epoch time when the URL is to expire. If both this option and the `--policy` option are specified, `--policy` will be used. Otherwise, this option alone will use a canned policy.

```
=back
```

```
=cut
```

```
use strict;
```

```
use warnings;
```

```
# you might need to use CPAN to get these modules.
```

```
# run perl -MCPAN -e "install <module>" to get them.
```

```
# The openssl command line will also need to be in your $PATH.
```

```
use File::Temp qw/tempfile/;
```

```
use File::Slurp;
```

```
use Getopt::Long;
```

```
use IPC::Open2;
```

```
use MIME::Base64 qw(encode_base64 decode_base64);
```

```
use Pod::Usage;
```

```
use URI;
```

```
my $CANNED_POLICY
```

```
    = '{"Statement":[{"Resource":"<RESOURCE>","Condition":{"DateLessThan":  
{"AWS:EpochTime":<EXPIRES>}}}]}';
```

```
my $POLICY_PARAM      = "Policy";
```

```
my $EXPIRES_PARAM    = "Expires";
```

```
my $SIGNATURE_PARAM  = "Signature";
```

```
my $KEY_PAIR_ID_PARAM = "Key-Pair-Id";
```

```
my $verbose = 0;
```

```
my $policy_filename = "";
```

```
my $expires_epoch = 0;
```

```
my $action = "";
```

```
my $help = 0;
```

```
my $key_pair_id = "";
```

```
my $url = "";
```

```
my $stream = "";
```

```
my $private_key_filename = "";
```

```
my $result = GetOptions("action=s"      => \$action,  
                        "policy=s"     => \$policy_filename,
```

```
        "expires=i"      => \$expires_epoch,
        "private-key=s"  => \$private_key_filename,
        "key-pair-id=s"  => \$key_pair_id,
        "verbose"       => \$verbose,
        "help"          => \$help,
        "url=s"         => \$url,
        "stream=s"      => \$stream,
    );

if ($help or !$result) {
    pod2usage(1);
    exit;
}

if ($url eq "" and $stream eq "") {
    print STDERR "Must include a stream or a URL to encode or decode with the --stream
or --url option\n";
    exit;
}

if ($url ne "" and $stream ne "") {
    print STDERR "Only one of --url and --stream may be specified\n";
    exit;
}

if ($url ne "" and !is_url_valid($url)) {
    exit;
}

if ($stream ne "") {
    exit unless is_stream_valid($stream);

    # The signing mechanism is identical, so from here on just pretend we're
    # dealing with a URL
    $url = $stream;
}

if ($action eq "encode") {
    # The encode action will generate a private content URL given a base URL,
    # a policy file (or an expires timestamp) and a key pair id parameter
    my $private_key;
    my $public_key;
    my $public_key_file;
```

```
my $policy;
if ($policy_filename eq "") {
    if ($expires_epoch == 0) {
        print STDERR "Must include policy filename with --policy argument or an
expires" .
                "time using --expires\n";
    }

    $policy = $CANNED_POLICY;
    $policy =~ s/<EXPIRES>/$expires_epoch/g;
    $policy =~ s/<RESOURCE>/$url/g;
} else {
    if (! -e $policy_filename) {
        print STDERR "Policy file $policy_filename does not exist\n";
        exit;
    }
    $expires_epoch = 0; # ignore if set
    $policy = read_file($policy_filename);
}

if ($private_key_filename eq "") {
    print STDERR "You must specific the path to your private key file with --
private-key\n";
    exit;
}

if (! -e $private_key_filename) {
    print STDERR "Private key file $private_key_filename does not exist\n";
    exit;
}

if ($key_pair_id eq "") {
    print STDERR "You must specify a key pair id with --key-pair-id\n";
    exit;
}

my $encoded_policy = url_safe_base64_encode($policy);
my $signature = rsa_sha1_sign($policy, $private_key_filename);
my $encoded_signature = url_safe_base64_encode($signature);

my $generated_url = create_url($url, $encoded_policy, $encoded_signature,
$key_pair_id, $expires_epoch);
```

```
if ($stream ne "") {
    print "Encoded stream (for use within a swf):\n" . $generated_url . "\n";
    print "Encoded and escaped stream (for use on a webpage):\n" .
escape_url_for_webpage($generated_url) . "\n";
} else {
    print "Encoded URL:\n" . $generated_url . "\n";
}
} elsif ($action eq "decode") {
    my $decoded = decode_url($url);
    if (!$decoded) {
        print STDERR "Improperly formed URL\n";
        exit;
    }

    print_decoded_url($decoded);
} else {
    # No action specified, print help.  But only if this is run as a program (caller
will be empty)
    pod2usage(1) unless caller();
}

# Decode a private content URL into its component parts
sub decode_url {
    my $url = shift;

    if ($url =~ /(.*?)\?(.*)/) {
        my $base_url = $1;
        my $params = $2;

        my @unparsed_params = split(/&/, $params);
        my %params = ();
        foreach my $param (@unparsed_params) {
            my ($key, $val) = split(/=/, $param);
            $params{$key} = $val;
        }

        my $encoded_signature = "";
        if (exists $params{$SIGNATURE_PARAM}) {
            $encoded_signature = $params{"Signature"};
        } else {
            print STDERR "Missing Signature URL parameter\n";
            return 0;
        }
    }
}
```

```
my $encoded_policy = "";
if (exists $params{$POLICY_PARAM}) {
    $encoded_policy = $params{$POLICY_PARAM};
} else {
    if (!exists $params{$EXPIRES_PARAM}) {
        print STDERR "Either the Policy or Expires URL parameter needs to be
specified\n";
        return 0;
    }

    my $expires = $params{$EXPIRES_PARAM};

    my $policy = $CANNED_POLICY;
    $policy =~ s/<EXPIRES>/$expires/g;

    my $url_without_cf_params = $url;
    $url_without_cf_params =~ s/$SIGNATURE_PARAM=[^&]*&?//g;
    $url_without_cf_params =~ s/$POLICY_PARAM=[^&]*&?//g;
    $url_without_cf_params =~ s/$EXPIRES_PARAM=[^&]*&?//g;
    $url_without_cf_params =~ s/$KEY_PAIR_ID_PARAM=[^&]*&?//g;

    if ($url_without_cf_params =~ /(.*)\?$/) {
        $url_without_cf_params = $1;
    }

    $policy =~ s/<RESOURCE>/$url_without_cf_params/g;

    $encoded_policy = url_safe_base64_encode($policy);
}

my $key = "";
if (exists $params{$KEY_PAIR_ID_PARAM}) {
    $key = $params{$KEY_PAIR_ID_PARAM};
} else {
    print STDERR "Missing $KEY_PAIR_ID_PARAM parameter\n";
    return 0;
}

my $policy = url_safe_base64_decode($encoded_policy);

my %ret = ();
$ret{"base_url"} = $base_url;
$ret{"policy"} = $policy;
$ret{"key"} = $key;
```

```
        return \%ret;
    } else {
        return 0;
    }
}

# Print a decoded URL out
sub print_decoded_url {
    my $decoded = shift;

    print "Base URL: \n" . $decoded->{"base_url"} . "\n";
    print "Policy: \n" . $decoded->{"policy"} . "\n";
    print "Key: \n" . $decoded->{"key"} . "\n";
}

# Encode a string with base 64 encoding and replace some invalid URL characters
sub url_safe_base64_encode {
    my ($value) = @_;

    my $result = encode_base64($value);
    $result =~ tr|+="/|-_~|;

    return $result;
}

# Decode a string with base 64 encoding. URL-decode the string first
# followed by reversing any special character ("+="/) translation.
sub url_safe_base64_decode {
    my ($value) = @_;

    $value =~ s/%([0-9A-Fa-f]{2})/chr(hex($1))/eg;
    $value =~ tr|_~|+="/;

    my $result = decode_base64($value);

    return $result;
}

# Create a private content URL
sub create_url {
    my ($path, $policy, $signature, $key_pair_id, $expires) = @_;

    my $result;
```

```

    my $separator = $path =~ /\?/ ? '&' : '?';
    if ($expires) {
        $result = "$path$separator$EXPIRES_PARAM=$expires&$SIGNATURE_PARAM=$signature&
$KEY_PAIR_ID_PARAM=$key_pair_id";
    } else {
        $result = "$path$separator$POLICY_PARAM=$policy&$SIGNATURE_PARAM=$signature&
$KEY_PAIR_ID_PARAM=$key_pair_id";
    }
    $result =~ s/\n//g;

    return $result;
}

# Sign a document with given private key file.
# The first argument is the document to sign
# The second argument is the name of the private key file
sub rsa_sha1_sign {
    my ($to_sign, $pvkFile) = @_;
    print "openssl sha1 -sign $pvkFile $to_sign\n";

    return write_to_program($pvkFile, $to_sign);
}

# Helper function to write data to a program
sub write_to_program {
    my ($keyfile, $data) = @_;
    unlink "temp_policy.dat" if (-e "temp_policy.dat");
    unlink "temp_sign.dat" if (-e "temp_sign.dat");

    write_file("temp_policy.dat", $data);

    system("openssl dgst -sha1 -sign \"\$keyfile\" -out temp_sign.dat temp_policy.dat");

    my $output = read_file("temp_sign.dat");

    return $output;
}

# Read a file into a string and return the string
sub read_file {
    my ($file) = @_;

    open(INFILE, "<$file") or die("Failed to open $file: $!");
    my $str = join('', <INFILE>);

```

```
    close INFILE;

    return $str;
}

sub is_url_valid {
    my ($url) = @_;

    # HTTP distributions start with http[s]:// and are the correct thing to sign
    if ($url =~ /^https?:\/\/) {
        return 1;
    } else {
        print STDERR "CloudFront requires absolute URLs for HTTP distributions\n";
        return 0;
    }
}

sub is_stream_valid {
    my ($stream) = @_;

    if ($stream =~ /^rtmp:\/\// or $stream =~ /^\/?cfx\/st/) {
        print STDERR "Streaming distributions require that only the stream name is
signed.\n";
        print STDERR "The stream name is everything after, but not including, cfx/st/
\n";
        return 0;
    } else {
        return 1;
    }
}

# flash requires that the query parameters in the stream name are url
# encoded when passed in through javascript, etc. This sub handles the minimal
# required url encoding.
sub escape_url_for_webpage {
    my ($url) = @_;

    $url =~ s/\?/%3F/g;
    $url =~ s/=/%3D/g;
    $url =~ s/&/%26/g;

    return $url;
}
```

```
1;
```

PHP を使用して URL 署名を作成する

PHP を実行するウェブサーバーは、この PHP サンプルコードを使用して、プライベート CloudFront ディストリビューションのポリシーステートメントと署名を作成できます。完全な例では、CloudFront ストリーミングを使用してビデオストリームを再生する署名付き URL リンクを持つ機能するウェブページを作成します。完全な例は、<https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/samples/demo-php.zip> でダウンロードできます。

AWS SDK for PHP の `UrlSigner` クラスを使用して署名付き URL を作成することもできます。詳細については、AWS SDK for PHP API リファレンスの「[Class UrlSigner](#)」を参照してください。

Note

URL 署名の作成は、署名付き URL を使用してプライベートコンテンツを供給するためのプロセスの 1 パートにすぎません。プロセス全体の詳細については、「[署名付き URL の使用](#)」を参照してください。

トピック

- [サンプル: RSA SHA-1 署名](#)
- [サンプル: 既定ポリシーの作成](#)
- [サンプル: カスタムポリシーの作成](#)
- [完全なサンプルコード](#)

サンプル: RSA SHA-1 署名

以下のサンプルコードでは、関数 `rsa_sha1_sign` によってポリシーステートメントのハッシュと署名を行います。必要な引数は、ポリシーステートメントと、ディストリビューションの信頼されたキーグループにあるパブリックキーに対応するプライベートキーです。次に、`url_safe_base64_encode` 関数で、URL で使用可能なバージョンの署名を作成します。

```
function rsa_sha1_sign($policy, $private_key_filename) {  
    $signature = "";  
  
    // load the private key  
    $fp = fopen($private_key_filename, "r");
```

```
$priv_key = fread($fp, 8192);
fclose($fp);
$pkeyid = openssl_get_privatekey($priv_key);

// compute signature
openssl_sign($policy, $signature, $pkeyid);

// free the key from memory
openssl_free_key($pkeyid);

return $signature;
}

function url_safe_base64_encode($value) {
    $encoded = base64_encode($value);
    // replace unsafe characters +, = and / with
    // the safe characters -, _ and ~
    return str_replace(
        array('+', '=', '/'),
        array('-', '_', '~'),
        $encoded);
}
```

サンプル: 既定ポリシーの作成

以下のサンプルコードでは、署名用の既定ポリシーステートメントを作成します。既定ポリシーの詳細については、「[既定ポリシーを使用する署名付き URL の作成](#)」を参照してください。

Note

`$expires` 変数は、文字列ではなく整数である必要がある日時スタンプです。

```
function get_canned_policy_stream_name($video_path, $private_key_filename,
    $key_pair_id, $expires) {
    // this policy is well known by CloudFront, but you still need to sign it,
    // since it contains your parameters
    $canned_policy = '{"Statement":[{"Resource":"' . $video_path . '","Condition":
{"DateLessThan":{"AWS:EpochTime":' . $expires . '}}]}';

    // sign the canned policy
    $signature = rsa_sha1_sign($canned_policy, $private_key_filename);
```

```
// make the signature safe to be included in a url
$encoded_signature = url_safe_base64_encode($signature);

// combine the above into a stream name
$stream_name = create_stream_name($video_path, null, $encoded_signature,
$key_pair_id, $expires);
// url-encode the query string characters to work around a flash player bug
return encode_query_params($stream_name);
}
```

サンプル: カスタムポリシーの作成

以下のサンプルコードでは、署名用のカスタムポリシーステートメントを作成します。カスタムポリシーの詳細については、「[カスタムポリシーを使用する署名付き URL の作成](#)」を参照してください。

```
function get_custom_policy_stream_name($video_path, $private_key_filename,
$key_pair_id, $policy) {
    // sign the policy
    $signature = rsa_sha1_sign($policy, $private_key_filename);
    // make the signature safe to be included in a url
    $encoded_signature = url_safe_base64_encode($signature);

    // combine the above into a stream name
    $stream_name = create_stream_name($video_path, $encoded_policy, $encoded_signature,
$key_pair_id, null);
    // url-encode the query string characters to work around a flash player bug
    return encode_query_params($stream_name);
}
```

完全なサンプルコード

次のサンプルコードは、PHP で CloudFront 署名URLs を作成する完全なデモンストレーションを提供します。この完全な例は、<https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/samples/demo-php.zip> でダウンロードできます。

```
<?php

function rsa_sha1_sign($policy, $private_key_filename) {
    $signature = "";

    // load the private key
```

```
$fp = fopen($private_key_filename, "r");
$priv_key = fread($fp, 8192);
fclose($fp);
$pkeyid = openssl_get_privatekey($priv_key);

// compute signature
openssl_sign($policy, $signature, $pkeyid);

// free the key from memory
openssl_free_key($pkeyid);

return $signature;
}

function url_safe_base64_encode($value) {
    $encoded = base64_encode($value);
    // replace unsafe characters +, = and / with the safe characters -, _ and ~
    return str_replace(
        array('+', '=', '/'),
        array('-', '_', '~'),
        $encoded);
}

function create_stream_name($stream, $policy, $signature, $key_pair_id, $expires) {
    $result = $stream;
    // if the stream already contains query parameters, attach the new query parameters
    // to the end
    // otherwise, add the query parameters
    $separator = strpos($stream, '?') == FALSE ? '?' : '&';
    // the presence of an expires time means we're using a canned policy
    if($expires) {
        $result .= $path . $separator . "Expires=" . $expires . "&Signature=" .
        $signature . "&Key-Pair-Id=" . $key_pair_id;
    }
    // not using a canned policy, include the policy itself in the stream name
    else {
        $result .= $path . $separator . "Policy=" . $policy . "&Signature=" .
        $signature . "&Key-Pair-Id=" . $key_pair_id;
    }

    // new lines would break us, so remove them
    return str_replace('\n', '', $result);
}
```

```
function encode_query_params($stream_name) {
    // Adobe Flash Player has trouble with query parameters being passed into it,
    // so replace the bad characters with their URL-encoded forms
    return str_replace(
        array('?', '=', '&'),
        array('%3F', '%3D', '%26'),
        $stream_name);
}

function get_canned_policy_stream_name($video_path, $private_key_filename,
    $key_pair_id, $expires) {
    // this policy is well known by CloudFront, but you still need to sign it, since it
    // contains your parameters
    $canned_policy = '{"Statement":[{"Resource":"' . $video_path . '","Condition":
{"DateLessThan":{"AWS:EpochTime":' . $expires . '}}]}]';
    // the policy contains characters that cannot be part of a URL, so we base64 encode
    // it
    $encoded_policy = url_safe_base64_encode($canned_policy);
    // sign the original policy, not the encoded version
    $signature = rsa_sha1_sign($canned_policy, $private_key_filename);
    // make the signature safe to be included in a URL
    $encoded_signature = url_safe_base64_encode($signature);

    // combine the above into a stream name
    $stream_name = create_stream_name($video_path, null, $encoded_signature,
    $key_pair_id, $expires);
    // URL-encode the query string characters to support Flash Player
    return encode_query_params($stream_name);
}

function get_custom_policy_stream_name($video_path, $private_key_filename,
    $key_pair_id, $policy) {
    // the policy contains characters that cannot be part of a URL, so we base64 encode
    // it
    $encoded_policy = url_safe_base64_encode($policy);
    // sign the original policy, not the encoded version
    $signature = rsa_sha1_sign($policy, $private_key_filename);
    // make the signature safe to be included in a URL
    $encoded_signature = url_safe_base64_encode($signature);

    // combine the above into a stream name
    $stream_name = create_stream_name($video_path, $encoded_policy, $encoded_signature,
    $key_pair_id, null);
    // URL-encode the query string characters to support Flash Player
```

```
    return encode_query_params($stream_name);
}

// Path to your private key. Be very careful that this file is not accessible
// from the web!

$private_key_filename = '/home/test/secure/example-priv-key.pem';
$key_pair_id = 'K2JCJMDEHXQW5F';

$video_path = 'example.mp4';

$expires = time() + 300; // 5 min from now
$canned_policy_stream_name = get_canned_policy_stream_name($video_path,
    $private_key_filename, $key_pair_id, $expires);

$client_ip = $_SERVER['REMOTE_ADDR'];
$policy =
'{' .
    '"Statement":[' .
        '{' .
            '"Resource":' . $video_path . ',' .
            '"Condition":{' .
                '"IpAddress":{"AWS:SourceIp":"' . $client_ip . '/32"}', .
                '"DateLessThan":{"AWS:EpochTime":"' . $expires . '}' .
            '}' .
        ']' .
    '}' .
';

$custom_policy_stream_name = get_custom_policy_stream_name($video_path,
    $private_key_filename, $key_pair_id, $policy);

?>

<html>

<head>
    <title>CloudFront</title>
<script type='text/javascript' src='https://example.cloudfront.net/player/
swfobject.js'></script>
</head>

<body>
    <h1>Amazon CloudFront</h1>
```

```
<h2>Canned Policy</h2>
<h3>Expires at <? = gmdate('Y-m-d H:i:s T', $expires) ?></h3>
<br />

<div id='canned'>The canned policy video will be here</div>

<h2>Custom Policy</h2>
<h3>Expires at <? = gmdate('Y-m-d H:i:s T', $expires) ?> only viewable by IP <? =
$client_ip ?></h3>
<div id='custom'>The custom policy video will be here</div>

<!-- ***** Have to update the player.swf path to a real JWPlayer instance.
The fake one means that external people cannot watch the video right now -->
<script type='text/javascript'>
var so_canned = new SWFObject('https://files.example.com/
player.swf', 'mpl', '640', '360', '9');
so_canned.addParam('allowfullscreen', 'true');
so_canned.addParam('allowscriptaccess', 'always');
so_canned.addParam('wmode', 'opaque');
so_canned.addVariable('file', '<? = $canned_policy_stream_name ?>');
so_canned.addVariable('streamer', 'rtmp://example.cloudfront.net/cfx/st');
so_canned.write('canned');

var so_custom = new SWFObject('https://files.example.com/
player.swf', 'mpl', '640', '360', '9');
so_custom.addParam('allowfullscreen', 'true');
so_custom.addParam('allowscriptaccess', 'always');
so_custom.addParam('wmode', 'opaque');
so_custom.addVariable('file', '<? = $custom_policy_stream_name ?>');
so_custom.addVariable('streamer', 'rtmp://example.cloudfront.net/cfx/st');
so_custom.write('custom');
</script>
</body>
</html>
```

次の資料も参照してください:

- [Perl を使用して URL 署名を作成する](#)
- [C# と .NET Framework を使用して URL 署名を作成する](#)
- [Java を使用して URL 署名を作成する](#)

C# と .NET Framework を使用して URL 署名を作成する

このセクションの C# の例では、既定ポリシーステートメントとカスタムポリシーステートメントを使用して CloudFront プライベートディストリビューションの署名を作成する方法を示すサンプルアプリケーションを実装しています。これらの例には、.NET アプリケーションに便利な [AWS SDK for .NET](#) に基づくユーティリティ関数が含まれています。

AWS SDK for .NET を使用して署名付き URL と署名付き Cookie を作成することもできます。[AWS SDK for .NET API リファレンス](#)で、以下のトピックを参照してください。

- 署名付き URLs – Amazon.CloudFront > AmazonCloudFrontUrlSigner
- 署名付き Cookie – Amazon.CloudFront > AmazonCloudFrontCookieSigner

Note

URL 署名の作成は、署名付き URL を使用してプライベートコンテンツを供給するためのプロセスの 1 パートにすぎません。プロセス全体の詳細については、「[署名付き URL の使用](#)」を参照してください。

コードをダウンロードするには、[C# による署名コード](#)にアクセスしてください。

.NET Framework で RSA キーを使用するには、AWS 提供の .pem ファイルを .NET Framework が使用する XML 形式に変換する必要があります。

変換後、RSA プライベートキーファイルの形式は以下のようになります。

Example XML .NET Framework 形式の RSA プライベートキー

```
<RSAKeyValue>
  <Modulus>
    w05IvYCP5UcoCKDo1dcspoMehWBZcyfs9QEzGi60e5y+ewGr1oW+vB2GPB
    ANBiVPcUHTFWhwaIBd3oglmF0lGQ1jP/j0fmXHUK2kUUnLnJp+o0BL2NiuFtqcW6h/L51IpD8Yq+NRHg
    Ty4zDsyr2880MvXv88yEFURckqEXAMPLE=
  </Modulus>
  <Exponent>AQAB</Exponent>
  <P>
    5bmKDaTz
    npENGvqz4Cea8XPH+sxt+2VaAwYnsarVUoSBeVt8WL1oVuZGG9IZYmH5KteXEu7fZveYd9UEXAMPLE==
  </P>
  <Q>
```

```

1v9l/WN1a1N3r0K4VGoCokx7kR2SyTMSbZgF9IWJN0ugR/WZw7HTnjip03c9dy1Ms9pUKwUF4
6d7049EXAMPLE==
</Q>
<DP>
RgrSKuLWXMyBH+/l1Dx/I4tXuAJIrl1Pyo+Vmi0c7b5NzHptkSHEPFR9s1
0K0VqjknclqCJ3Ig860MEtEXAMPLE==
</DP>
<DQ>
pjPjvSFw+RoaTu0pgCA/jwW/FGyfn6iim1RFbkT4
z49DZb2IM885f3vf35eLTaEYRYUHqgZtChNEV0TEXAMPLE==
</DQ>
<InverseQ>
nkv0JTg5QtGNgWb9i
cVtzrL/1pFE0HbJXwEJdU99N+7sMK+1066DL/HSBUCD63qD4USpnf0myc24in0EXAMPLE==</InverseQ>
<D>
Bc7mp7XYHynuPZxChjWNJZIQ+A73gm0ASDv6At7F8Vi9r0xU1Qe/v0AQS3ycN8Q1yR4XMbzMLYk
3yjsxFDXo4ZKQt0GzLGteCU2srANiLv26/imXA8FVidZftTAtLviWQZBVPTeYIA69ATUYPEq0a5u5wjGy
U0ij90WyuEXAMPLE=
</D>
</RSAKeyValue>

```

以下の C# コードは、以下を実行して、既定ポリシーを使用する署名付き URL を作成します。

- ポリシーステートメントを作成する。
- SHA1 を使用してポリシーステートメントをハッシュ化し、RSA とプライベートキー (信頼されたキーグループにあるパブリックキーに対応するもの) を使用して、結果に署名します。
- ハッシュ化および署名されたポリシーステートメントを base64 エンコードし、特殊文字を置き換えて文字列を URL リクエストパラメータとして使用できるようにする。
- 値を連結する。

完全な実装については、[C# による署名コード](#)の例を参照してください。

Example C# における既定ポリシーの署名方法

```

public static string ToUrlSafeBase64String(byte[] bytes)
{
    return System.Convert.ToBase64String(bytes)
        .Replace('+', '-')
        .Replace('=', '~')
        .Replace('/', '~');
}

```

```
public static string CreateCannedPrivateURL(string urlString,
    string durationUnits, string durationNumber, string pathToPolicyStmnt,
    string pathToPrivateKey, string privateKeyId)
{
    // args[] 0-thisMethod, 1-resourceUrl, 2-seconds-minutes-hours-days
    // to expiration, 3-numberOfPreviousUnits, 4-pathToPolicyStmnt,
    // 5-pathToPrivateKey, 6-PrivateKeyId

    TimeSpan timeSpanInterval = GetDuration(durationUnits, durationNumber);

    // Create the policy statement.
    string strPolicy = CreatePolicyStatement(pathToPolicyStmnt,
        urlString,
        DateTime.Now,
        DateTime.Now.Add(timeSpanInterval),
        "0.0.0.0/0");
    if ("Error!" == strPolicy) return "Invalid time frame." +
        "Start time cannot be greater than end time.";

    // Copy the expiration time defined by policy statement.
    string strExpiration = CopyExpirationTimeFromPolicy(strPolicy);

    // Read the policy into a byte buffer.
    byte[] bufferPolicy = Encoding.ASCII.GetBytes(strPolicy);

    // Initialize the SHA1CryptoServiceProvider object and hash the policy data.
    using (SHA1CryptoServiceProvider
        cryptoSHA1 = new SHA1CryptoServiceProvider())
    {
        bufferPolicy = cryptoSHA1.ComputeHash(bufferPolicy);

        // Initialize the RSACryptoServiceProvider object.
        RSACryptoServiceProvider providerRSA = new RSACryptoServiceProvider();
        XmlDocument xmlPrivateKey = new XmlDocument();

        // Load PrivateKey.xml, which you created by converting your
        // .pem file to the XML format that the .NET framework uses.
        // Several tools are available.
        xmlPrivateKey.Load(pathToPrivateKey);

        // Format the RSACryptoServiceProvider providerRSA and
        // create the signature.
        providerRSA.FromXmlString(xmlPrivateKey.InnerXml);
    }
}
```

```
RSAPKCS1SignatureFormatter rsaFormatter =
    new RSAPKCS1SignatureFormatter(providerRSA);
rsaFormatter.SetHashAlgorithm("SHA1");
byte[] signedPolicyHash = rsaFormatter.CreateSignature(bufferPolicy);

// Convert the signed policy to URL-safe base64 encoding and
// replace unsafe characters + = / with the safe characters - _ ~
string strSignedPolicy = ToUrlSafeBase64String(signedPolicyHash);

// Concatenate the URL, the timestamp, the signature,
// and the key pair ID to form the signed URL.
return urlString +
    "?Expires=" +
    strExpiration +
    "&Signature=" +
    strSignedPolicy +
    "&Key-Pair-Id=" +
    privateKeyId;
}
}
```

以下の C# コードは、以下の手順を実行して、カスタムポリシーを使用する署名付き URL を作成します。

1. ポリシーステートメントを作成する。
2. ポリシーステートメントを Base64 エンコードし、特殊文字を置き換えて文字列を URL リクエストパラメータとして使用できるようにする。
3. SHA1 を使用してポリシーステートメントをハッシュ化し、RSA とプライベートキー (信頼されたキーグループにあるパブリックキーに対応するもの) を使用して、結果を暗号化します。
4. ハッシュ化されたポリシーステートメントを base64 エンコードし、特殊文字を置き換えて文字列を URL リクエストパラメータとして使用できるようにする。
5. 値を連結する。

完全な実装については、[C# による署名コード](#)の例を参照してください。

Example C# におけるカスタムポリシーの署名方法

```
public static string ToUrlSafeBase64String(byte[] bytes)
{
    return System.Convert.ToBase64String(bytes)
}
```

```
        .Replace('+', '-')
        .Replace('=', '_')
        .Replace('/', '~');
    }

    public static string CreateCustomPrivateURL(string urlString,
        string durationUnits, string durationNumber, string startIntervalFromNow,
        string ipaddress, string pathToPolicyStmnt, string pathToPrivateKey,
        string PrivateKeyId)
    {
        // args[] 0-thisMethod, 1-resourceUrl, 2-seconds-minutes-hours-days
        // to expiration, 3-numberOfPreviousUnits, 4-starttimeFromNow,
        // 5-ip_address, 6-pathToPolicyStmnt, 7-pathToPrivateKey, 8-privateKeyId

        TimeSpan timeSpanInterval = GetDuration(durationUnits, durationNumber);
        TimeSpan timeSpanToStart = GetDurationByUnits(durationUnits,
            startIntervalFromNow);
        if (null == timeSpanToStart)
            return "Invalid duration units." +
                "Valid options: seconds, minutes, hours, or days";

        string strPolicy = CreatePolicyStatement(
            pathToPolicyStmnt, urlString, DateTime.Now.Add(timeSpanToStart),
            DateTime.Now.Add(timeSpanInterval), ipaddress);

        // Read the policy into a byte buffer.
        byte[] bufferPolicy = Encoding.ASCII.GetBytes(strPolicy);

        // Convert the policy statement to URL-safe base64 encoding and
        // replace unsafe characters + = / with the safe characters - _ ~

        string urlSafePolicy = ToUrlSafeBase64String(bufferPolicy);

        // Initialize the SHA1CryptoServiceProvider object and hash the policy data.
        byte[] bufferPolicyHash;
        using (SHA1CryptoServiceProvider cryptoSHA1 =
            new SHA1CryptoServiceProvider())
        {
            bufferPolicyHash = cryptoSHA1.ComputeHash(bufferPolicy);

            // Initialize the RSACryptoServiceProvider object.
            RSACryptoServiceProvider providerRSA = new RSACryptoServiceProvider();
            XmlDocument xmlPrivateKey = new XmlDocument();
```

```
// Load PrivateKey.xml, which you created by converting your
// .pem file to the XML format that the .NET framework uses.
// Several tools are available.
xmlPrivateKey.Load("PrivateKey.xml");

// Format the RSACryptoServiceProvider providerRSA
// and create the signature.
providerRSA.FromXmlString(xmlPrivateKey.InnerXml);
RSAPKCS1SignatureFormatter RSAFormatter =
    new RSAPKCS1SignatureFormatter(providerRSA);
RSAFormatter.SetHashAlgorithm("SHA1");
byte[] signedHash = RSAFormatter.CreateSignature(bufferPolicyHash);

// Convert the signed policy to URL-safe base64 encoding and
// replace unsafe characters + = / with the safe characters - _ ~
string strSignedPolicy = ToUrlSafeBase64String(signedHash);

return urlString +
    "?Policy=" +
    urlSafePolicy +
    "&Signature=" +
    strSignedPolicy +
    "&Key-Pair-Id=" +
    PrivateKeyId;
}
}
```

Example 署名生成のためのユーティリティメソッド

以下のメソッドは、ファイルからポリシーステートメントを取得し、署名生成の期間を解析します。

```
public static string CreatePolicyStatement(string policyStmnt,
    string resourceUrl,
    DateTime startTime,
    DateTime endTime,
    string ipAddress)
{
    // Create the policy statement.
    FileStream streamPolicy = new FileStream(policyStmnt, FileMode.Open,
    FileAccess.Read);
    using (StreamReader reader = new StreamReader(streamPolicy))
    {
        string strPolicy = reader.ReadToEnd();
    }
}
```

```
TimeSpan startTimeSpanFromNow = (startTime - DateTime.Now);
TimeSpan endTimeSpanFromNow = (endTime - DateTime.Now);
TimeSpan intervalStart =
    (DateTime.UtcNow.Add(startTimeSpanFromNow)) -
    new DateTime(1970, 1, 1, 0, 0, 0, DateTimeKind.Utc);
TimeSpan intervalEnd =
    (DateTime.UtcNow.Add(endTimeSpanFromNow)) -
    new DateTime(1970, 1, 1, 0, 0, 0, DateTimeKind.Utc);

int startTimestamp = (int)intervalStart.TotalSeconds; // START_TIME
int endTimestamp = (int)intervalEnd.TotalSeconds; // END_TIME

if (startTimestamp > endTimestamp)
    return "Error!";

// Replace variables in the policy statement.
strPolicy = strPolicy.Replace("RESOURCE", resourceUrl);
strPolicy = strPolicy.Replace("START_TIME", startTimestamp.ToString());
strPolicy = strPolicy.Replace("END_TIME", endTimestamp.ToString());
strPolicy = strPolicy.Replace("IP_ADDRESS", ipAddress);
strPolicy = strPolicy.Replace("EXPIRES", endTimestamp.ToString());
return strPolicy;
}
}

public static TimeSpan GetDuration(string units, string numUnits)
{
    TimeSpan timeSpanInterval = new TimeSpan();
    switch (units)
    {
        case "seconds":
            timeSpanInterval = new TimeSpan(0, 0, 0, int.Parse(numUnits));
            break;
        case "minutes":
            timeSpanInterval = new TimeSpan(0, 0, int.Parse(numUnits), 0);
            break;
        case "hours":
            timeSpanInterval = new TimeSpan(0, int.Parse(numUnits), 0, 0);
            break;
        case "days":
            timeSpanInterval = new TimeSpan(int.Parse(numUnits), 0, 0, 0);
            break;
        default:
```

```
        Console.WriteLine("Invalid time units;" +
            "use seconds, minutes, hours, or days");
        break;
    }
    return TimeSpanInterval;
}

private static TimeSpan GetDurationByUnits(string durationUnits,
    string startIntervalFromNow)
{
    switch (durationUnits)
    {
        case "seconds":
            return new TimeSpan(0, 0, int.Parse(startIntervalFromNow));
        case "minutes":
            return new TimeSpan(0, int.Parse(startIntervalFromNow), 0);
        case "hours":
            return new TimeSpan(int.Parse(startIntervalFromNow), 0, 0);
        case "days":
            return new TimeSpan(int.Parse(startIntervalFromNow), 0, 0, 0);
        default:
            return new TimeSpan(0, 0, 0, 0);
    }
}

public static string CopyExpirationTimeFromPolicy(string policyStatement)
{
    int startExpiration = policyStatement.IndexOf("EpochTime");
    string strExpirationRough = policyStatement.Substring(startExpiration +
        "EpochTime".Length);
    char[] digits = { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9' };

    List<char> listDigits = new List<char>(digits);
    StringBuilder buildExpiration = new StringBuilder(20);

    foreach (char c in strExpirationRough)
    {
        if (listDigits.Contains(c))
            buildExpiration.Append(c);
    }
    return buildExpiration.ToString();
}
```

以下の資料も参照してください。

- [Perl を使用して URL 署名を作成する](#)
- [PHP を使用して URL 署名を作成する](#)
- [Java を使用して URL 署名を作成する](#)

Java を使用して URL 署名を作成する

次のコード例に加えて、[AWS SDK for Java \(バージョン 1\) の CloudFrontUrlSigner ユーティリティクラス](#)を使用して [CloudFront 署名URLs](#) を作成できます。

その他の例については、[URLs と Cookie を作成するAWS](#)」を参照してください。AWS

Note

署名付き URL の作成は、[でプライベートコンテンツを提供する CloudFront](#)プロセスの一部にすぎません。プロセス全体の詳細については、「[署名付き URL の使用](#)」を参照してください。

次の例は、CloudFront 署名付き URL を作成する方法を示しています。プライベートキーを使用するには、プライベートキーを PEM 形式から、Java 実装用の DER 形式に変換する必要があります。

Example Java のポリシーおよび署名暗号化メソッド

```
// Signed URLs for a private distribution
// Note that Java only supports SSL certificates in DER format,
// so you will need to convert your PEM-formatted file to DER format.
// To do this, you can use openssl:
// openssl pkcs8 -topk8 -nocrypt -in origin.pem -inform PEM -out new.der
// -outform DER
// So the encoder works correctly, you should also add the bouncy castle jar
// to your project and then add the provider.

Security.addProvider(new org.bouncycastle.jce.provider.BouncyCastleProvider());

String distributionDomain = "a1b2c3d4e5f6g7.cloudfront.net";
String privateKeyFilePath = "/path/to/rsa-private-key.der";
String s3objectKey = "s3/object/key.txt";
String policyResourcePath = "https://" + distributionDomain + "/" + s3objectKey;
```

```
// Convert your DER file into a byte array.

byte[] derPrivateKey = ServiceUtils.readInputStreamToBytes(new
    FileInputStream(privateKeyFilePath));

// Generate a "canned" signed URL to allow access to a
// specific distribution and file

String signedUrlCanned = CloudFrontService.signUrlCanned(
    "https://" + distributionDomain + "/" + s3objectKey, // Resource URL or Path
    keyPairId, // Certificate identifier,
    // an active trusted signer for the distribution
    derPrivateKey, // DER Private key data
    ServiceUtils.parseIso8601Date("2011-11-14T22:20:00.000Z") // DateLessThan
);
System.out.println(signedUrlCanned);

// Build a policy document to define custom restrictions for a signed URL.

String policy = CloudFrontService.buildPolicyForSignedUrl(
    // Resource path (optional, can include '*' and '?' wildcards)
    policyResourcePath,
    // DateLessThan
    ServiceUtils.parseIso8601Date("2011-11-14T22:20:00.000Z"),
    // CIDR IP address restriction (optional, 0.0.0.0/0 means everyone)
    "0.0.0.0/0",
    // DateGreaterThan (optional)
    ServiceUtils.parseIso8601Date("2011-10-16T06:31:56.000Z")
);

// Generate a signed URL using a custom policy document.

String signedUrl = CloudFrontService.signUrl(
    // Resource URL or Path
    "https://" + distributionDomain + "/" + s3objectKey,
    // Certificate identifier, an active trusted signer for the distribution
    keyPairId,
    // DER Private key data
    derPrivateKey,
    // Access control policy
    policy
);
System.out.println(signedUrl);
```

以下も参照してください。

- [Perl を使用して URL 署名を作成する](#)
- [PHP を使用して URL 署名を作成する](#)
- [C# と .NET Framework を使用して URL 署名を作成する](#)

AWS オリジンへのアクセスの制限

CloudFront および オAWSリジンは、次の利点を提供する方法で設定できます。

- AWS へのアクセスを制限して、パブリックにアクセスできないようにします。
- ビューワー (ユーザー) が、指定された CloudFront デистриビューションを介してのみAWSオリジン内のコンテンツにアクセスできるようにします。これにより、バケットから直接コンテンツにアクセスしたり、意図しない CloudFront デистриビューションを介してコンテンツにアクセスしたりすることが防止されます。

これを行うには、認証されたリクエスト CloudFront をAWSオリジンに送信するようにを設定し、からの認証されたリクエストへのアクセスのみを許可するようにAWSオリジンを設定します CloudFront。互換性のある AWS オリジンのタイプについては、以下のトピックを参照してください。

トピック

- [オリジンへのアクセス MediaStoreの制限](#)
- [Amazon S3 オリジンへのアクセスの制限](#)

オリジンへのアクセス MediaStoreの制限

CloudFront は、オリジンへのアクセスを制限するためのオリジンアクセスコントロール (OAC) を提供します。AWS Elemental MediaStore

トピック

- [新しいオリジンアクセスコントロールの作成](#)
- [オリジンアクセスコントロールの詳細設定](#)

新しいオリジンアクセスコントロールの作成

で新しいオリジンアクセスコントロールを設定するには、以下のトピックで説明するステップを実行します CloudFront。

トピック

- [前提条件](#)
- [オリジンアクセスコントロールに MediaStore オリジンへのアクセス許可を付与する](#)
- [オリジンアクセスコントロールの作成](#)

前提条件

オリジンアクセスコントロールを作成して設定する前に、MediaStore オリジンを含む CloudFront デистриビューションが必要です。

オリジンアクセスコントロールに MediaStore オリジンへのアクセス許可を付与する

オリジンアクセスコントロールを作成したり、CloudFront デистриビューションで設定したりする前に、OAC に MediaStore オリジンへのアクセス許可があることを確認してください。これは、CloudFront デистриビューションを作成した後、デистриビューション設定のオリジンに MediaStore OAC を追加する前に行います。

オ MediaStore リジンへのアクセス許可を OAC に付与するには、MediaStore コンテナポリシーを使用して、CloudFront サービスプリンシパル (cloudfront.amazonaws.com) にオリジンへのアクセスを許可します。ポリシーの Condition 要素を使用して、リクエストが MediaStore オリジンを含む CloudFront デистриビューションに代わっている場合にのみ、が MediaStore コンテナ CloudFront にアクセスできるようにします。

以下は、CloudFront OAC が MediaStore オリジンにアクセスできるようにする MediaStore コンテナポリシーの例です。

Example MediaStore CloudFront OAC への読み取り専用アクセスを許可する コンテナポリシー

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCloudFrontServicePrincipalReadOnly",
      "Effect": "Allow",
      "Principal": {
```

```

        "Service": "cloudfront.amazonaws.com"
    },
    "Action": [
        "mediastore:GetObject"
    ],
    "Resource": "arn:aws:mediastore:<region>:<AWS #####
ID>:container/<container name>/*",
    "Condition": {
        "StringEquals": {
            "AWS:SourceArn": "arn:aws:cloudfront::<AWS #####
ID>:distribution/<CloudFront distribution ID>"
        },
        "Bool": {
            "aws:SecureTransport": "true"
        }
    }
}
]
}

```

Example MediaStore OAC への CloudFront 読み取りおよび書き込みアクセスを許可する コンテナポリシー

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCloudFrontServicePrincipalReadWrite",
            "Effect": "Allow",
            "Principal": {
                "Service": "cloudfront.amazonaws.com"
            },
            "Action": [
                "mediastore:GetObject",
                "mediastore:PutObject"
            ],
            "Resource": "arn:aws:mediastore:<region>:<AWS #####
ID>:container/<container name>/*",
            "Condition": {
                "StringEquals": {
                    "AWS:SourceArn": "arn:aws:cloudfront::<AWS #####
ID>:distribution/<CloudFront distribution ID>"
                }
            },

```

```
        "Bool": {
            "aws:SecureTransport": "true"
        }
    }
}
]
```

Note

書き込みアクセスを許可するには、許可された HTTP メソッドを設定して、ディストリビューションの動作設定PUTに CloudFrontを含める必要があります。

オリジンアクセスコントロールの作成

OAC を作成するには、AWS Management Console、AWS CLI、または CloudFront API AWS CloudFormationを使用します。

Console

オリジンアクセスコントロールを作成するには

1. にサインインAWS Management Consoleし、 で CloudFront コンソールを開きます<https://console.aws.amazon.com/cloudfront/v4/home>。
2. ナビゲーションペインで、[オリジンアクセス] を選択します。
3. [コントロール設定を作成] を選択します。
4. [コントロール設定を作成] フォームで、以下の操作を行います。
 - a. [詳細] ペインに、[名前] と (オプションで) オリジンアクセスコントロールの [説明] を入力します。
 - b. [設定] ペインでは、デフォルト設定である [Sign requests (recommended)] (署名リクエスト (推奨)) をそのまま使用することをお勧めします。詳細については、「[the section called “オリジンアクセスコントロールの詳細設定”](#)」を参照してください。
5. オリジンタイプのドロップダウン MediaStore から選択します。
6. [作成] を選択します。

OAC を作成したら、[名前] を書き留めておきます。次の手順では、この操作を行う必要があります。

ディストリビューションのオリジンに MediaStore オリジンアクセスコントロールを追加するには

1. で CloudFront コンソールを開きます<https://console.aws.amazon.com/cloudfront/v4/home>。
2. OAC を追加する MediaStore オリジンを持つディストリビューションを選択し、オリジンタブを選択します。
3. OAC を追加する MediaStore オリジンを選択し、編集を選択します。
4. オリジンの [Protocol] (プロトコル) として [HTTPS only] (HTTPS のみ) を選択します。
5. [Origin access control] (オリジンアクセスコントロール) ドロップダウンメニューから、使用する OAC を選択します。
6. [変更の保存] をクリックします。

ディストリビューションは、すべての CloudFront エッジロケーションへのデプロイを開始します。エッジロケーションは、新しい設定を受け取ると、バケットオリジンに送信するすべてのリクエストに署名します MediaStore。

CloudFormation

AWS CloudFormation を使用してオリジンアクセスコントロール (OAC) を作成するには、`AWS::CloudFront::OriginAccessControl` リソースタイプを使用します。以下の例は、オリジンアクセスコントロールを作成するための AWS CloudFormation テンプレート構文を YAML 形式で示しています。

```
Type: AWS::CloudFront::OriginAccessControl
Properties:
  OriginAccessControlConfig:
    Description: An optional description for the origin access control
    Name: ExampleOAC
    OriginAccessControlOriginType: mediastore
    SigningBehavior: always
    SigningProtocol: sigv4
```

詳細については、「ユーザーガイド」の[AWS::CloudFront::OriginAccess「コントロール」](#)を参照してください。AWS CloudFormation

CLI

AWS Command Line Interface (AWS CLI) を使用してオリジンアクセスコントロールを作成するには、`aws cloudfront create-origin-access-control` コマンドを使用します。コマンドの入力パラ

メータは、コマンドライン入力として個別に指定せずに、入力ファイルを使用して指定できます。

オリジンアクセスコントロール (入力ファイルを含む CLI) を作成するには

1. 次のコマンドを使用して、`origin-access-control.yaml` という名前のファイルを作成します。このファイルには、`create-origin-access-control` コマンドのすべての入力パラメータが含まれます。

```
aws cloudfront create-origin-access-control --generate-cli-skeleton yaml-input >
origin-access-control.yaml
```

2. 先ほど作成した `origin-access-control.yaml` ファイルを開きます。ファイルを編集して OAC の名前、説明 (オプション) を追加し、`SigningBehavior` を `always` に変更します。その後、ファイルを保存します。

その他の OAC の設定については、「[the section called “オリジンアクセスコントロールの詳細設定”](#)」を参照してください。

3. 次のコマンドを使用して、`origin-access-control.yaml` ファイルの入力パラメータを使用し、オリジンアクセスコントロールを作成します。

```
aws cloudfront create-origin-access-control --cli-input-yaml file://origin-
access-control.yaml
```

コマンド出力の `Id` 値を書き留めます。OAC を CloudFront ディストリビューションの MediaStore オリジンに追加するには、OAC が必要です。

OAC を既存のディストリビューションの MediaStore オリジンにアタッチするには (入力ファイルを含む CLI)

1. 次のコマンドを使用して、OAC を追加するディストリビューションのディストリビューション設定 CloudFront を保存します。ディストリビューションには MediaStore オリジンが必要です。

```
aws cloudfront get-distribution-config --id <CloudFront distribution ID> --output yaml > dist-config.yaml
```

2. 先ほど作成した `dist-config.yaml` という名前のファイルを開きます。ファイルを編集し、以下の変更を加えます。
 - Origins オブジェクトで、`OriginAccessControlId` という名前のフィールドに OAC の ID を追加します。
 - `OriginAccessIdentity` という名前のフィールドから値を削除します (存在する場合)。
 - `Etag` フィールドの名前を `IfMatch` に変更します。ただし、フィールドの値は変更しないでください。

完了したら、ファイルを保存します。

3. オリジンアクセスコントロールを使用するようにディストリビューションを更新するには、次のコマンドを使用します。

```
aws cloudfront update-distribution --id <CloudFront distribution ID> --cli-input-yaml file://dist-config.yaml
```

ディストリビューションは、すべての CloudFront エッジロケーションへのデプロイを開始します。エッジロケーションは、新しい設定を受け取ると、オリジンに送信するすべてのリクエストに署名します MediaStore。

API

CloudFront API を使用してオリジンアクセスコントロールを作成するには、[CreateOriginAccessControl](#) を使用します。この API コールで指定するフィールドの詳細については、AWS SDK やその他の API クライアントの API リファレンスドキュメントを参照してください。

オリジンアクセスコントロールを作成したら、次のいずれかの API コールを使用して、ディストリビューションの MediaStore オリジンにアタッチできます。

- 既存のディストリビューションにアタッチするには、[UpdateDistribution](#) を使用します。
- 新しいディストリビューションにアタッチするには、[CreateDistribution](#) を使用します。

これらの API コールの両方について、`OriginAccessControlId` フィールドのオリジン内にオリジンアクセスコントロール ID を指定します。これらの API コールで指定するその他フィー

ルドの詳細については、「[ディストリビューションを作成または更新する場合に指定する値](#)」と、AWS SDK またはその他 API クライアントの API リファレンスドキュメントを参照してください。

オリジンアクセスコントロールの詳細設定

CloudFront オリジンアクセスコントロール機能には、特定のユースケースのみを対象とした詳細設定が含まれています。詳細設定が特に必要でない限り、推奨設定を使用してください。

オリジンアクセスコントロールには、[署名動作] (コンソール)、または `SigningBehavior` (API、CLI、および AWS CloudFormation) という設定が含まれています この設定では、次のオプションを使用できます。

オリジンリクエストに常に署名する (推奨設定)

コンソールの [Sign requests (recommended)] (署名リクエスト (推奨))、または API、CLI、および AWS CloudFormation の `always` という設定を使用することをお勧めします。この設定では、CloudFront はオ MediaStore リジンに送信するすべてのリクエストに常に署名します。

オリジンリクエストに署名しない

この設定は、コンソールでは [リクエストに署名しない]、または API、CLI、および AWS CloudFormation では `never` です。この設定を使用して、このオリジンアクセスコントロールを使用するすべてのディストリビューションのすべてのオリジンに対して、オリジンアクセスコントロールをオフにします。これにより、オリジンアクセスコントロールを使用するすべてのオリジンとディストリビューションからコントロールを 1 つずつ削除する場合に比べて、時間と労力を節約できます。この設定 CloudFront では、はオ MediaStore リジンに送信するリクエストに署名しません。

Warning

この設定を使用するには、MediaStore オリジンがパブリックにアクセス可能である必要があります。パブリックにアクセスできない MediaStore オリジンでこの設定を使用すると、CloudFront はオリジンにアクセスできません。MediaStore オリジンはエラーを CloudFront に返し、CloudFront それらのエラーをビューワーに渡します。詳細については、[HTTPS 経由のパブリック読み取りアクセス](#)の MediaStore コンテナポリシーの例を参照してください。

ビューワー (クライアント) の **Authorization** ヘッダーを上書きしない

この設定は、コンソールでは [認証ヘッダーを上書きしない] で、API、CLI、および AWS CloudFormation では no-override です。この設定は、対応するビューワーリクエスト CloudFront に Authorizationヘッダーが含まれていない場合にのみ、オリジンリクエストに署名する場合に使用します。この設定では、ビューワーリクエストが存在する場合、はビューワーリクエストから Authorizationヘッダーを CloudFront 渡しますが、ビューワーリクエストに Authorizationヘッダーが含まれていない場合、オリジンリクエストに署名 (独自の Authorizationヘッダーを追加) します。

Warning

ビューワーリクエストから Authorizationヘッダーを渡すには、このオリジンアクセスコントロールに関連付けられた MediaStore オリジンを使用するすべてのキャッシュ動作の[キャッシュポリシー](#)に Authorizationヘッダーを追加する必要があります。

Amazon S3 オリジンへのアクセスの制限

CloudFront では、認証されたリクエストを Amazon S3 オリジンに送信する 2 つの方法として、オリジンアクセスコントロール (OAC) とオリジンアクセスアイデンティティ (OAI) があります。OAC は以下をサポートしているため、OAC の使用をお勧めします。

- すべての AWS リージョンのすべての Amazon S3 バケット (2022 年 12 月以降に開始されたオプトインリージョンを含む)
- [AWS KMS による Amazon S3 サーバー側の暗号化 \(SSE-KMS\)](#)
- Amazon S3 に対する動的なリクエスト (PUT と DELETE)

オリジンアクセスアイデンティティ (OAI) は、上記のリストのシナリオでは機能しないか、追加の回避策が必要です。以下のトピックでは、Amazon S3 オリジンでオリジンアクセスコントロール (OAC) を使用する方法について説明します。オリジンアクセスアイデンティティ (OAI) からオリジンアクセスコントロール (OAI) への移行方法については、「[the section called “オリジンアクセスアイデンティティ \(OAI\) からオリジンアクセスコントロール \(OAC\) への移行”](#)」を参照してください。

メモ

- Amazon S3 バケットオリジンで CloudFront OAC を使用する場合は、Amazon S3 オブジェクト所有権を、新しい Amazon S3 バケットのデフォルトであるバケット所有者強制に設定する必要があります。ACLs が必要な場合は、バケット所有者の優先設定を使用して、経由でアップロードされたオブジェクトの制御を維持します CloudFront。
- オリジンが [ウェブサイトエンドポイント](#) として設定された Amazon S3 バケットである場合は、[カスタムオリジン CloudFront](#) として設定する必要があります。つまり、OAC (または OAI) を使用することはできません。OAC は Lambda@Edge を使用したオリジンリダイレクトをサポートしていません。

トピック

- [the section called “新しいオリジンアクセスコントロールの作成”](#)
- [the section called “オリジンアクセスアイデンティティ \(OAI\) からオリジンアクセスコントロール \(OAC\) への移行”](#)
- [the section called “オリジンアクセスコントロールの詳細設定”](#)

新しいオリジンアクセスコントロールの作成

で新しいオリジンアクセスコントロールを設定するには、以下のトピックで説明するステップを実行します CloudFront。

トピック

- [前提条件](#)
- [S3 バケットへのアクセス許可をオリジンアクセスコントロールに付与する](#)
- [オリジンアクセスコントロールの作成](#)

前提条件

オリジンアクセスコントロール (OAC) を作成して設定する前に、CloudFront Amazon S3 バケットオリジンを持つディストリビューションが必要です。このオリジンは、[ウェブサイトエンドポイント](#)として設定されたバケットではなく、通常の S3 バケットである必要があります。S3 バケットオリジンを使用した CloudFront ディストリビューションの設定の詳細については、「」を参照してください [the section called “簡単なディストリビューションの使用を開始する”](#)。

Note

OAC を使用して S3 バケットオリジンを保護する場合、特定の設定に関係なく、CloudFront と Amazon S3 間の通信は常に HTTPS 経由で行われます。

S3 バケットへのアクセス許可をオリジンアクセスコントロールに付与する

オリジンアクセスコントロール (OAC) を作成したり、CloudFront デイストリビューションで設定したりする前に、OAC に S3 バケットオリジンへのアクセス許可があることを確認してください。これは、CloudFront デイストリビューションを作成した後、デイストリビューション設定の S3 オリジンに OAC を追加する前に行います。

S3 バケットへのアクセス許可を OAC に付与するには、S3 [バケットポリシー](#) を使用して、サービスプリンシパル (cloudfront.amazonaws.com) に CloudFront バケットへのアクセスを許可します。ポリシーの Condition 要素を使用して、リクエストが S3 オリジンを含む CloudFront デイストリビューションに代わっている場合にのみ、[バケット CloudFront にアクセスできるようにします](#)。

バケットポリシーの追加または変更の詳細については、Amazon S3 ユーザーガイドの「[Amazon S3 コンソールを使用したバケットポリシーの追加](#)」を参照してください。

以下は、CloudFront OAC が S3 オリジンにアクセスできるようにする S3 バケットポリシーの例です。

Example CloudFront OAC への読み取り専用アクセスを許可する S3 バケットポリシー

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCloudFrontServicePrincipalReadOnly",
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudfront.amazonaws.com"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::<S3 bucket name>/*",
      "Condition": {
        "StringEquals": {
          "AWS:SourceArn": "arn:aws:cloudfront::<AWS #####
ID>:distribution/<CloudFront distribution ID>"
        }
      }
    }
  ]
}
```

```
    }  
  }  
}
```

Example CloudFront OAC への読み取りおよび書き込みアクセスを許可する S3 バケットポリシー

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "AllowCloudFrontServicePrincipalReadWrite",  
      "Effect": "Allow",  
      "Principal": {  
        "Service": "cloudfront.amazonaws.com"  
      },  
      "Action": [  
        "s3:GetObject",  
        "s3:PutObject"  
      ],  
      "Resource": "arn:aws:s3:::<S3 bucket name>/*",  
      "Condition": {  
        "StringEquals": {  
          "AWS:SourceArn": "arn:aws:cloudfront::<AWS #####  
ID>:distribution/<CloudFront distribution ID>"  
        }  
      }  
    }  
  ]  
}
```

SSE-KMS

S3 バケットオリジン内のオブジェクトが [AWS Key Management Service \(SSE-KMS\) でのサーバー側の暗号化](#) を使用して暗号化されている場合は、OAC に AWS KMS キーを使用するアクセス許可があることを確認する必要があります。KMS キーを使用するアクセス許可を OAC に付与するには、[KMS キーポリシー](#) にステートメントを追加します。キーポリシーを変更する方法については、AWS Key Management Service デベロッパーガイドの「[キーポリシーの変更](#)」を参照してください。

次の例は、OAC による KMS キーの使用を許可する KMS キーポリシーステートメントを示しています。

Example CloudFront OAC が SSE-KMS の KMS キーにアクセスすることを許可する KMS キーポリシーステートメント

```
{
  "Sid": "AllowCloudFrontServicePrincipalSSE-KMS",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::<AWS ##### ID>:root",
    "Service": "cloudfront.amazonaws.com"
  },
  "Action": [
    "kms:Decrypt",
    "kms:Encrypt",
    "kms:GenerateDataKey*"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "AWS:SourceArn": "arn:aws:cloudfront::<AWS ##### ID>:distribution/<CloudFront distribution ID>"
    }
  }
}
```

オリジンアクセスコントロールの作成

オリジンアクセスコントロール (OAC) を作成するには、AWS Management Console、AWS CloudFormation、AWS CLI、または CloudFront API を使用できます。

Console

オリジンアクセスコントロールを作成するには

1. にサインインAWS Management Consoleし、 で CloudFront コンソールを開きます<https://console.aws.amazon.com/cloudfront/v4/home>。
2. ナビゲーションペインで、[オリジンアクセス] を選択します。
3. [コントロール設定を作成] を選択します。
4. [コントロール設定を作成] フォームで、以下の操作を行います。
 - a. [詳細] ペインに、[名前] と (オプションで) オリジンアクセスコントロールの [説明] を入力します。

- b. [設定] ペインでは、デフォルト設定である [Sign requests (recommended)] (署名リクエスト (推奨)) をそのまま使用することをお勧めします。詳細については、「[the section called “オリジンアクセスコントロールの詳細設定”](#)」を参照してください。
5. [Origin type] (オリジンタイプ) ドロップダウンから [S3] を選択します。
6. [作成] を選択します。

OAC を作成したら、[名前] を書き留めておきます。次の手順では、この操作を行う必要があります。

ディストリビューションの S3 オリジンにオリジンアクセスコントロールを追加するには

1. で CloudFront コンソールを開きます<https://console.aws.amazon.com/cloudfront/v4/home>。
2. OAC を追加する S3 オリジンのあるディストリビューションを選択し、[オリジン] タブを選択します。
3. OAC を追加する S3 オリジンを選択し、[編集] を選択します。
4. オリジンアクセスで、オリジンアクセスコントロール設定 (推奨) を選択します。
5. [Origin access control] (オリジンアクセスコントロール) ドロップダウンメニューから、使用する OAC を選択します。
6. [変更の保存] をクリックします。

ディストリビューションは、すべての CloudFront エッジロケーションへのデプロイを開始します。エッジロケーションは、新しい設定を受け取ると、S3 バケットオリジンに送信するすべてのリクエストに署名します。

CloudFormation

AWS CloudFormation を使用してオリジンアクセスコントロール (OAC) を作成するには、AWS::::OriginAccessControl リソースタイプを使用します。以下の例は、オリジンアクセスコントロールを作成するための AWS CloudFormation テンプレート構文を YAML 形式で示しています。

```
Type: AWS::::OriginAccessControl
Properties:
  OriginAccessControlConfig:
    Description: An optional description for the origin access control
    Name: ExampleOAC
    OriginAccessControlOriginType: s3
```

```
SigningBehavior: always
SigningProtocol: sigv4
```

詳細については、「ユーザーガイド」の[AWS::CloudFront::OriginAccess「コントロール」](#)を参照してください。AWS CloudFormation

CLI

AWS Command Line Interface (AWS CLI) を使用してオリジンアクセスコントロールを作成するには、`aws cloudfront create-origin-access-control` コマンドを使用します。コマンドの入力パラメータは、コマンドライン入力として個別に指定せずに、入力ファイルを使用して指定できます。

オリジンアクセスコントロール (入力ファイルを含む CLI) を作成するには

1. 次のコマンドを使用して、`origin-access-control.yaml` という名前のファイルを作成します。このファイルには、`create-origin-access-control` コマンドのすべての入力パラメータが含まれます。

```
aws cloudfront create-origin-access-control --generate-cli-skeleton yaml-input >
origin-access-control.yaml
```

2. 先ほど作成した `origin-access-control.yaml` ファイルを開きます。ファイルを編集して OAC の名前、説明 (オプション) を追加し、`SigningBehavior` を `always` に変更します。その後、ファイルを保存します。

その他の OAC の設定については、「[the section called “オリジンアクセスコントロールの詳細設定”](#)」を参照してください。

3. 次のコマンドを使用して、`origin-access-control.yaml` ファイルの入力パラメータを使用し、オリジンアクセスコントロールを作成します。

```
aws cloudfront create-origin-access-control --cli-input-yaml file://origin-
access-control.yaml
```

コマンド出力の `Id` 値を書き留めます。OAC を CloudFront ディストリビューションの S3 バケットオリジンに追加するには、この OAC が必要です。

OAC を既存のディストリビューションの S3 バケットオリジンにアタッチするには (入力ファイルを含む CLI)

1. 次のコマンドを使用して、OAC を追加するディストリビューションのディストリビューション設定 CloudFrontを保存します。ディストリビューションには S3 バケットオリジンが必要です。

```
aws cloudfront get-distribution-config --id <CloudFront distribution ID> --output yaml > dist-config.yaml
```

2. 先ほど作成した dist-config.yaml という名前のファイルを開きます。ファイルを編集し、以下の変更を加えます。
 - Origins オブジェクトで、OriginAccessControlId という名前のフィールドに OAC の ID を追加します。
 - OriginAccessIdentity という名前のフィールドから値を削除します (存在する場合)。
 - ETag フィールドの名前を IfMatch に変更します。ただし、フィールドの値は変更しないでください。

完了したら、ファイルを保存します。

3. オリジンアクセスコントロールを使用するようにディストリビューションを更新するには、次のコマンドを使用します。

```
aws cloudfront update-distribution --id <CloudFront distribution ID> --cli-input-yaml file:///dist-config.yaml
```

ディストリビューションは、すべての CloudFront エッジロケーションへのデプロイを開始します。エッジロケーションは、新しい設定を受け取ると、S3 バケットオリジンに送信するすべてのリクエストに署名します。

API

CloudFront API を使用してオリジンアクセスコントロールを作成するには、[CreateOriginAccessControl](#) を使用します。この API コールで指定するフィールドの詳細については、AWS SDK やその他の API クライアントの API リファレンスドキュメントを参照してください。

オリジンアクセスコントロールを作成したら、次の API コールのいずれかを使用して、それをディストリビューションの S3 バケットオリジンにアタッチできます。

- 既存のディストリビューションにアタッチするには、を使用します [UpdateDistribution](#)。
- 新しいディストリビューションにアタッチするには、を使用します [CreateDistribution](#)。

これらの API コールの両方について、OriginAccessControlId フィールドのオリジン内にオリジンアクセスコントロール ID を指定します。これらの API コールで指定する他のフィールドの詳細については、「[ディストリビューションを作成または更新する場合に指定する値](#)」と、AWS SDK またはその他 API クライアントの API リファレンスドキュメントを参照してください。

オリジンアクセスアイデンティティ (OAI) からオリジンアクセスコントロール (OAC) への移行

レガシーのオリジンアクセスアイデンティティ (OAI) からオリジンアクセスコントロール (OAC) に移行するには、まず S3 バケットオリジンを更新して、OAI と OAC の両方がバケットのコンテンツにアクセスできるようにします。これにより、移行中に がバケットにアクセスできなくなる CloudFront ことがなくなります。OAI と OAC の両方が S3 バケットにアクセスできるようにするには、[バケットポリシー](#)を更新し、プリンシパルの種類ごとに 1 つずつ、2 つのステートメントを含めます。

次の S3 バケットポリシー例では、OAI と OAC の両方に S3 オリジンへのアクセスを許可しています。

Example OAI および OAC への読み取り専用アクセスを許可する S3 バケットポリシー

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCloudFrontServicePrincipalReadOnly",
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudfront.amazonaws.com"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::<S3 bucket name>/*",
      "Condition": {
```

```

        "StringEquals": {
            "AWS:SourceArn": "arn:aws:cloudfront::<AWS #####
ID>:distribution/<CloudFront distribution ID>"
        }
    },
    {
        "Sid": "AllowLegacyOAIReadOnly",
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::cloudfront:user/CloudFront Origin Access
Identity <origin access identity ID>"
        },
        "Action": "s3:GetObject",
        "Resource": "arn:aws:s3:::<S3 bucket name>/*"
    }
]
}

```

S3 オリジンのバケットポリシーを更新して OAI と OAC の両方へのアクセスを許可したら、OAI の代わりに OAC を使用するようにディストリビューション設定を更新できます。詳細については、「[the section called “新しいオリジンアクセスコントロールの作成”](#)」を参照してください。

ディストリビューションが完全にデプロイされたら、OAI へのアクセスを許可するバケットポリシー内のステートメントを削除できます。詳しくは、「[the section called “S3 バケットへのアクセス許可をオリジンアクセスコントロールに付与する”](#)」を参照してください。

オリジンアクセスコントロールの詳細設定

CloudFront オリジンアクセスコントロール機能には、特定のユースケースのみを対象とした詳細設定が含まれています。詳細設定が特に必要でない限り、推奨設定を使用してください。

オリジンアクセスコントロールには、[署名動作] (コンソール)、または SigningBehavior (API、CLI、および AWS CloudFormation) という設定が含まれています。この設定では、次のオプションを使用できます。

オリジンリクエストに常に署名する (推奨設定)

コンソールの [Sign requests (recommended)] (署名リクエスト (推奨))、または API、CLI、および AWS CloudFormation の `always` という設定を使用することをお勧めします。この設定では、CloudFront は S3 バケットオリジンに送信するすべてのリクエストに常に署名します。

オリジンリクエストに署名しない

この設定は、コンソールでは [リクエストに署名しない]、または API、CLI、および AWS CloudFormation では `never` です。この設定を使用して、このオリジンアクセスコントロールを使用するすべてのディストリビューションのすべてのオリジンに対して、オリジンアクセスコントロールをオフにします。これにより、オリジンアクセスコントロールを使用するすべてのオリジンとディストリビューションからコントロールを 1 つずつ削除する場合に比べて、時間と労力を節約できます。この設定 CloudFront では、は S3 バケットオリジンに送信するリクエストに署名しません。

Warning

この設定を使用するには、S3 バケットオリジンがパブリックにアクセス可能である必要があります。パブリックにアクセスできない S3 バケットオリジンでこの設定を使用する場合は、オリジンにアクセス CloudFront できません。S3 バケットオリジンは にエラーを返し CloudFront、 CloudFront それらのエラーをビューワーに渡します。

ビューワー (クライアント) の **Authorization** ヘッダーを上書きしない

この設定は、コンソールでは [認証ヘッダーを上書きしない] で、API、CLI、および AWS CloudFormation では `no-override` です。この設定は、対応するビューワーリクエスト CloudFront に `Authorization` ヘッダーが含まれていない場合にのみ、オリジンリクエストに署名する場合に使用します。この設定では、ビューワーリクエストが存在する場合、はビューワーリクエストから `Authorization` ヘッダーを CloudFront 渡しますが、ビューワーリクエストに `Authorization` ヘッダーが含まれていない場合、オリジンリクエストに署名 (独自の `Authorization` ヘッダーを追加) します。

Warning

ビューワーリクエストから `Authorization` ヘッダーを渡すには、このオリジンアクセスコントロールに関連付けられた S3 バケットオリジンを使用するすべてのキャッシュ動作に対して、`Authorization` ヘッダーを [キャッシュポリシー](#) に必ず追加する必要があります。

オリジンアクセスアイデンティティの使用 (レガシー、非推奨)

オリジンアクセスアイデンティティの概要

CloudFront オリジンアクセスアイデンティティ (OAI) は、オリジンアクセスコントロール (OAC) と同様の機能を提供しますが、すべてのシナリオで機能するわけではありません。これが、代わりに OAC の使用をお勧めする理由です。具体的には、OAI は以下をサポートしていません。

- オプトインリージョンを含む、すべての AWS リージョンの Amazon S3 バケット
- [AWS KMS による Amazon S3 サーバー側の暗号化 \(SSE-KMS\)](#)
- Amazon S3 に対する動的なリクエスト (PUT、POST、または DELETE)
- 2022 年 12 月以降に開始された新しい AWS リージョン

OAI から OAC への移行に関する詳細については、「[the section called “オリジンアクセスアイデンティティ \(OAI\) からオリジンアクセスコントロール \(OAC\) への移行”](#)」を参照してください。

オリジンアクセスアイデンティティに Amazon S3 バケット内のファイルの読み取りアクセス許可を付与する

CloudFront コンソールを使用して OAI を作成するか、ディストリビューションに追加すると、Amazon S3 バケットポリシーを自動的に更新して、バケットへのアクセス許可を OAI に付与できます。あるいは、このバケットポリシーの手動での作成または更新を選択することができます。どちらの方法を使用する場合でも、アクセス許可を確認して次のことを確認する必要があります。

- CloudFront OAI は、を通じてリクエストしているビューワーに代わってバケット内のファイルにアクセスできます CloudFront。
- ビューワーは、Amazon S3 URLs を使用して の外部にあるファイルにアクセスすることはできません CloudFront。

Important

が CloudFront サポートするすべての HTTP メソッドを受け入れて転送 CloudFront するように を設定する場合は、必要なアクセス許可を CloudFront OAI に付与してください。例えば、DELETEメソッドを使用するリクエストを受け入れて転送 CloudFront するように を設定した場合、ビューワーが目的のファイルのみを削除できるように、DELETEリクエストを適切に処理するようにバケットポリシーを設定します。

Amazon S3 バケットポリシーの使用

次の方法でバケットポリシーを作成または更新することで、Amazon S3 バケット内のファイルへのアクセス権を CloudFront OAI に付与できます。

- [Amazon S3 コンソール](#)で、Amazon S3 バケットの [アクセス許可] タブを使用する。
- Amazon S3 API [PutBucketPolicy](#)で を使用します。Amazon S3
- [CloudFront コンソールを使用する](#)。CloudFront コンソールでオリジン設定に OAI を追加する場合、はい、バケットポリシーを更新して、ユーザーに代わってバケットポリシーを更新する CloudFront ように に指示できます。

バケットポリシーを手動で更新する場合は、次の点を確認してください。

- ポリシーで正しい OAI を Principal として指定します。
- ビューワーに代わってオブジェクトにアクセスするために必要なアクセス許可を OAI に付与します。

詳細については、次のセクションを参照してください。

バケットポリシーで OAI を Principal として指定

Amazon S3 バケットポリシーで Principal として OAI を指定するには、OAI の ID を含む OAI の Amazon リソースネーム (ARN) を使用します。例:

```
"Principal": {
  "AWS": "arn:aws:iam::cloudfront:user/CloudFront Origin Access Identity <origin
access identity ID>"
}
```

CloudFront コンソールの「セキュリティ」、「オリジンアクセス」、「アイデンティティ (レガシー)」で OAI ID を見つけます。または、API [ListCloudFrontOriginAccessIdentities](#)で CloudFront を使用します。

OAI にアクセス許可を付与

Amazon S3 バケット内のオブジェクトにアクセスするためのアクセス許可を OAI に付与するには、特定の Amazon S3 API オペレーションに関連するポリシーでアクションを使用します。例えば、s3:GetObject アクションは、OAI がバケット内のオブジェクトを読み取ることを許可しま

す。詳細については、次のセクションの例を参照するか、Amazon Simple Storage Service ユーザーガイドの「[Amazon S3 のアクション](#)」を参照してください。

Amazon S3 バケットポリシーの例

次の例は、CloudFront OAI が S3 バケットにアクセスできるようにする Amazon S3 バケットポリシーを示しています。

CloudFront コンソールの「セキュリティ」、「オリジンアクセス」、「アイデンティティ (レガシー)」で OAI ID を見つけます。または、API [ListCloudFrontOriginAccessIdentities](#) で CloudFront を使用します。

Example OAI に読み取りアクセスを許可する Amazon S3 バケットポリシー

次の例では、指定されたバケット (s3:GetObject) 内のオブジェクトの読み取りを OAI に許可します。

```
{
  "Version": "2012-10-17",
  "Id": "PolicyForCloudFrontPrivateContent",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::cloudfront:user/CloudFront Origin Access
Identity <origin access identity ID>"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::<S3 bucket name>/*"
    }
  ]
}
```

Example OAI に読み取りおよび書き込みアクセスを許可する Amazon S3 バケットポリシー

次の例では、指定されたバケット (s3:GetObject と s3:PutObject) 内のオブジェクトの読み取りおよび書き込みを OAI に許可します。これにより、ビューワーは を介して Amazon S3 バケットにファイルをアップロードできます CloudFront。

```
{
  "Version": "2012-10-17",
  "Id": "PolicyForCloudFrontPrivateContent",
```

```
"Statement": [
  {
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::cloudfront:user/CloudFront Origin Access
Identity <origin access identity ID>"
    },
    "Action": [
      "s3:GetObject",
      "s3:PutObject"
    ],
    "Resource": "arn:aws:s3:::<S3 bucket name>/*"
  }
]
```

Amazon S3 オブジェクト ACL の使用 (非推奨)

Important

[Amazon S3 バケットポリシーを使用して](#) OAI へのアクセスを S3 バケットに与えることを推奨します。このセクションで説明したようにアクセスコントロールリスト (ACL) を使用できませんが、お勧めしません。

Amazon S3 では所有権が強化されたバケットに [S3 オブジェクトの所有権](#) を設定することを推奨しており、すなわち、バケットとその中のオブジェクトについて ACL が無効になっているということです。この設定を [オブジェクトの所有権] に適用する場合は、バケットポリシーを使用して OAI へのアクセスを与える必要があります (前のセクションを参照)。次のセクションは、ACL を必要とするレガシーユースケースのみを対象としています。

CloudFront OAI に Amazon S3 バケット内のファイルへのアクセスを許可するには、次の方法でファイルの ACL を作成または更新します。

- [Amazon S3 コンソール](#) で、Amazon S3 オブジェクトの [アクセス許可] タブを使用する。
- Amazon S3 API [PutObjectAcl](#) で を使用します。Amazon S3

ACL を使用して OAI へのアクセスを許可する場合、Amazon S3 の正規ユーザー ID を使用して OAI を指定する必要があります。CloudFront コンソールでは、セキュリティ、オリジンアクセス、アイデンティティ (レガシー) の下にこの ID があります。CloudFront API を使用してい

る場合は、OAI を作成したときに返された S3CanonicalUserId要素の値を使用するか、API [ListCloudFrontOriginAccessIdentities](#) を呼び出します CloudFront。

署名バージョン 4 の認証のみをサポートする Amazon S3 リージョンでのオリジンアクセスアイデンティティの使用

新しい Amazon S3 リージョンでは、リクエストの認証用に署名バージョン 4 を使用する必要があります。(各 Amazon S3 リージョンでサポートされている署名バージョンについては、「AWS 全般のリファレンス」の「[Amazon Simple Storage Service エンドポイントおよびクォータ](#)」を参照してください)。オリジンアクセスアイデンティティを使用しており、バケットが、署名バージョン 4 が必要なリージョンの 1 つにある場合、以下の点に注意してください。

- DELETE、GET、HEAD、OPTIONS、および PATCH リクエストは条件なしでサポートされます。
- POST リクエストはサポートされません。

Application Load Balancers へのアクセスを制限する

Elastic Load Balancing の Application Load Balancer によって提供されるウェブアプリケーションやその他のコンテンツについては、オブジェクトを CloudFront キャッシュしてユーザー (閲覧者) に直接提供し、Application Load Balancer の負荷を軽減します。CloudFront は、レイテンシーを減らし、分散型サービス拒否 (DDoS) 攻撃を吸収するのにも役立ちます。ただし、ユーザーが Application Load Balancer を直接バイパス CloudFront してアクセスできる場合、これらの利点はありません。ただし、ユーザーが Application Load Balancer に直接アクセスできないように Amazon CloudFront と Application Load Balancer を設定できます。これにより、ユーザーは を介してのみ Application Load Balancer にアクセスでき CloudFront、 を使用する利点が得られます CloudFront。

ユーザーが Application Load Balancer に直接アクセスできないようにし、経由でのみアクセスを許可するには CloudFront、以下の大まかなステップを実行します。

1. Application Load Balancer に送信するリクエストにカスタム HTTP ヘッダーを追加する CloudFront ように を設定します。
2. カスタム HTTP ヘッダーを含むリクエストだけを転送するように、Application Load Balancer を設定します。
3. (オプション) このソリューションのセキュリティを向上させるためには、HTTPS が必要です。

詳細については、以下のトピックを参照してください。これらのステップを完了すると、ユーザーは を介してのみ Application Load Balancer にアクセスできます CloudFront。

トピック

- [リクエスト CloudFront にカスタム HTTP ヘッダーを追加するための の設定](#)
- [特定のヘッダーを含むリクエストだけを転送するように Application Load Balancer を構成する](#)
- [\(オプション \) このソリューションのセキュリティ向上](#)

リクエスト CloudFront にカスタム HTTP ヘッダーを追加するための の設定

オリジン (この場合は Application Load Balancer) に送信するリクエストにカスタム HTTP ヘッダーを追加する CloudFront ように を設定できます。

Important

このユースケースは、カスタムヘッダー名と値の機密性維持を信頼しています。ヘッダー名と値が機密でない場合、他の HTTP クライアントは、Application Load Balancer に直接送信するリクエストにヘッダー名や値を含める可能性があります。これにより、Application Load Balancer は、リクエストが から送信されたかのように動作 CloudFront する可能性があります。これを防ぐためには、カスタムヘッダー名と値を機密にしておきます。

コンソール、AWS CloudFormation または CloudFront API を使用して CloudFront、オリジンリクエストにカスタム HTTP ヘッダーを追加する CloudFront ように を設定できます。

カスタム HTTP ヘッダーを追加するには (CloudFront コンソール)

CloudFront コンソールで、オリジン設定 のオリジンカスタムヘッダー設定を使用します。次の例に示すように、ヘッダー名とその値を入力します。

Note

この例のヘッダー名と値は、デモンストレーションのためだけに使用されます。製作では、ランダムに生成された値を使用します。ヘッダー名と値は、ユーザー名やパスワードなどの安全な資格情報として扱います。

Origin Custom Headers	Header Name	Value	
	<input type="text" value="X-Custom-Header"/>	<input type="text" value="random-value-1234567890"/>	

オリジンカスタムヘッダー設定は、既存の CloudFront ディストリビューションのオリジンを作成または編集するとき、および新しいディストリビューションを作成するとき編集できます。詳細については、「[ディストリビューションの更新](#)」および「[ディストリビューションの作成](#)」を参照してください。

カスタム HTTP ヘッダー (AWS CloudFormation) の追加

AWS CloudFormation テンプレートで、次の例に示すように、OriginCustomHeaders プロパティを使用します。

Note

この例のヘッダー名と値は、デモンストレーションのためだけに使用されます。製作では、ランダムに生成された値を使用します。ヘッダー名と値は、ユーザー名やパスワードなどの安全な資格情報として扱います。

```
AWSTemplateFormatVersion: '2010-09-09'
Resources:
  TestDistribution:
    Type: 'AWS::CloudFront::Distribution'
    Properties:
      DistributionConfig:
        Origins:
          - DomainName: app-load-balancer.example.com
            Id: Example-ALB
            CustomOriginConfig:
              OriginProtocolPolicy: https-only
              OriginSSLProtocols:
                - TLSv1.2
            OriginCustomHeaders:
              - HeaderName: X-Custom-Header
                HeaderValue: random-value-1234567890
        Enabled: 'true'
      DefaultCacheBehavior:
        TargetOriginId: Example-ALB
        ViewerProtocolPolicy: allow-all
```

```
CachePolicyId: 658327ea-f89d-4fab-a63d-7e88639e58f6
PriceClass: PriceClass_All
ViewerCertificate:
  CloudFrontDefaultCertificate: 'true'
```

詳細については、「AWS CloudFormationユーザーガイド」の「[オリジンとOriginCustomHeaderプロパティ](#)」を参照してください。

カスタム HTTP ヘッダーを追加するには (CloudFront API)

CloudFront API では、内で CustomHeaders オブジェクトを使用しますOrigin。詳細については、「Amazon API リファレンス[UpdateDistribution](#)」の[CreateDistribution](#)「」と「」、および SDK やその他の API クライアントのドキュメントを参照してください。 CloudFront

オリジンカスタムヘッダーとして指定できないヘッダー名がいくつかあります。詳細については、「[がオリジンリクエストに追加 CloudFront できないカスタムヘッダー](#)」を参照してください。

特定のヘッダーを含むリクエストだけを転送するように Application Load Balancer を構成する

Application Load Balancer に送信するリクエストにカスタム HTTP ヘッダーを追加する CloudFront ように を設定した後 ([前のセクション](#)「」を参照)、このカスタムヘッダーを含むリクエストのみを転送するようにロードバランサーを設定できます。これを行うためには、新しいルールを追加し、ロードバランサーのリスナーでデフォルトルールを変更します。

前提条件

次の手順を使用するためには、最低 1 つのリスナーがある Application Load Balancer が必要です。まだ作成していない場合は、[Application Load Balancer ユーザーガイド](#)の「Application Load Balancer の作成」を参照してください。

次の手順では、HTTPS リスナーを変更します。同じプロセスを使用して HTTP リスナーを変更できます。

Application Load Balancer リスナーのルールの更新

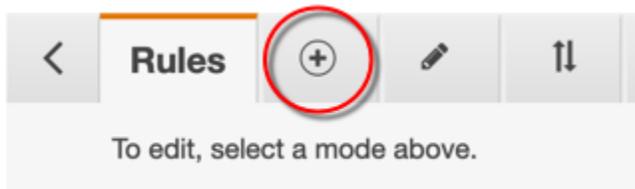
1. Amazon EC2 コンソールで [\[ロードバランサー\] ページを開きます](#)。
2. CloudFront デイストリビューションのオリジンであるロードバランサーを選択し、リスナータブを選択します。

3. 変更しているリスナーに、[ルールを表示/編集] を選択します。

Add listener Edit Delete

<input type="checkbox"/>	Listener ID	Security policy	SSL Certificate	Rules
<input type="checkbox"/>	HTTP : 80 arn...ae7dc34c19caf856 ▾	N/A	N/A	Default: returnin View/edit rules
<input type="checkbox"/>	HTTPS : 443 arn...e1f05424a9a62da1 ▾	ELBSecurityPolicy-TLS-1-2-Ext-2018-06	Default: b858ae2b-e0a3-4420-9538-4d7fe0e49b19 (ACM) View/edit certificates	Default: forward View/edit rules

4. アイコンを選択してルールを追加します。



5. [Insert Rule] を選択します。

< Rules + ✎ ⇅ - example-app | HTTPS:443 ▾

Click a location for your new rule. Each rule must include one action of type forward, redirect, fixed response.

example-app | HTTPS:443 (1 rules)

▶ Rule limits for condition values, wildcards, and total rules.

[+ Insert Rule](#)

last	HTTPS 443: default action <i>This rule cannot be moved or deleted</i>	IF ✓ Requests otherwise not routed	THEN Forward to example-app : 1 (100%) Group-level stickiness: Off
------	--	---------------------------------------	---

6. 新しいルールでは、次の操作を実行します。

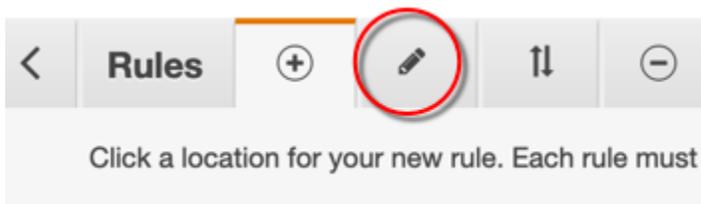
- [条件の追加] を選択してから、[Http ヘッダー] を選択します。でオリジンカスタムヘッダーとして追加した HTTP ヘッダー名と値を指定します CloudFront。
- [アクションの追加] を選択してから、[転送先] を選択します。リクエストを転送するターゲットグループを選択します。
- [保存] を選択して新しいルールを作成します。

Click a location for your new rule. Each rule must include one action of type forward, redirect, fixed response. Cancel Save

↑ Insert Rule ↓

RULE ID	IF (all match)	THEN
1 A rule ID (ARN) is generated when you save your rule.	<p>Http header...</p> <p>X-Custom-Header</p> <p>is random-value-1234567890 ✕</p> <p>or Value ✕</p> <p>✓</p> <p>+ Add condition</p>	<p>1. Forward to...</p> <p>Target group : Weight (0-999)</p> <p>example-app 1 ✕</p> <p>Traffic distribution 100%</p> <p>Select a target group 0 ✕</p> <p>▶ Group-level stickiness</p> <p>✓</p> <p>+ Add action</p>

7. アイコンを選択してルールを編集します。



8. デフォルトルールの編集アイコンを選択します。

Rules (+) [edit icon] [sort icon] (-)

Select the rule to edit. Each rule must include one action of type forward, redirect, fixed response.

example-app | **HTTPS:443** (2 rules)

▶ Rule limits for condition values, wildcards, and total rules.

1 [edit icon] arn...de3a0 ▾

IF
✓ Http header X-Custom-Header is random-value-1234567890

[edit icon] last **HTTPS 443: default action**
This rule cannot be moved or deleted

IF
✓ Requests otherwise not routed

9. デフォルトルールでは、次の操作を行います。

a. デフォルトのアクションを削除します。

Edit Rule

RULE ID	IF (all match)	THEN
last arn...2ef04 ▾	✓ Requests otherwise not routed	1. Forward to example-app: 1 (100%) Group-level stickiness: Off <div style="text-align: right;">[trash icon]</div>

+ Add action ▾

b. [アクションの追加] を選択してから、[固定応答を返す] を選択します。

c. [応答コード] に、**403**を入力します。

d. [応答本文] に、**Access denied**を入力します。

e. 「更新」を選択して、デフォルトルールを更新します。

Select the rule to edit. Each rule must include one action of type forward, redirect, fixed response.

Cancel Update

Edit Rule

RULE ID	IF (all match)	THEN
last arn...2ef04 ▾	<input checked="" type="checkbox"/> Requests otherwise not routed	<div style="border: 1px solid gray; padding: 5px;"> <p>1. Return fixed response... 🗑️</p> <p>Response code (2xx,4xx,5xx)</p> <input type="text" value="403"/> <p>Content-Type (optional)</p> <input type="text" value="text/plain"/> <p>Response body (optional)</p> <input style="width: 100%;" type="text" value="Access denied"/> </div>

これらのステップを完了すると、次の図が示すように、ロードバランサーリスナーに 2 つのルールがあります。最初のルールは、HTTP ヘッダーを含むリクエスト (からのリクエスト) を転送します CloudFront。2 番目のルールは、他のすべてのリクエスト (からではないリクエスト) に固定レスポンスを送信します CloudFront。

Rules
example-app | HTTPS:443 ▾

To edit, select a mode above.

example-app | **HTTPS:443** (2 rules)

▶ Rule limits for condition values, wildcards, and total rules.

1	arn...de3a0 ▾	IF <input checked="" type="checkbox"/> Http header X-Custom-Header is random-value-1234567890	THEN Forward to example-app: 1 (100%) Group-level stickiness: Off
last	HTTPS 443: default action <i>This rule cannot be moved or deleted</i>	IF <input checked="" type="checkbox"/> Requests otherwise not routed	THEN Return fixed response 403 (more...)

デイス CloudFront トリビューションと Application Load Balancer にリクエストを送信することで、ソリューションが機能することを確認できます。ウェブアプリケーションまたはコンテンツを CloudFront 返すリクエスト、および Application Load Balancer に直接送信されたリクエストは、プレーンテキストメッセージを含む 403 レスポンスを返します Access denied。

(オプション) このソリューションのセキュリティ向上

このソリューションのセキュリティを向上させるために、Application Load Balancer にリクエストを送信するときに常に HTTPS を使用するようにデイス CloudFront トリビューションを設定できます。このソリューションは、カスタムヘッダー名と値を機密に保つ場合に限り機能します。HTTPS を使用すると、盗聴者がヘッダー名と値を検出するのを防ぐことに役立ちます。また、ヘッダー名と値を定期的に交換することをお勧めします。

オリジンリクエストに HTTPS を使用する

オリジンリクエストに HTTPS を使用する CloudFront ように を設定するには、オリジンプロトコルポリシー設定を HTTPS のみに設定します。この設定は、CloudFront コンソール、AWS CloudFormation、および CloudFront API で使用できます。詳細については、「[プロトコル \(カスタムオリジンのみ\)](#)」を参照してください。

オリジンリクエストに HTTPS を使用する CloudFront ように を設定する場合は、Application Load Balancer に HTTPS リスナーがあることを確認する必要があります ([前のセクション](#) を参照)。これには、Application Load Balancer にルーティングされるドメイン名と一致する SSL/TLS 証明書が必要です。詳細については、Application Load Balancer [ユーザーガイドの「HTTPS リスナーを作成する」](#)を参照してください。

ウェブアプリケーションのエンドユーザー (ビューワー、またはクライアントとも呼ばれます) が HTTPS を使用できる場合は、エンドユーザーからの HTTPS 接続を優先 (または要求) CloudFront するように を設定することもできます。これを行うためには、[ビューワープロトコルポリシー] 設定を使用します。エンドユーザを HTTP から HTTPS にリダイレクトする、または HTTP を使用するリクエストを拒否するように設定できます。この設定は、CloudFront コンソール、AWS CloudFormation、および CloudFront API で使用できます。詳細については、「[ビューワープロトコルポリシー](#)」を参照してください。

ヘッダー名と値を交換する

HTTPS の使用に加え、ヘッダー名と値を定期的に交換することをお勧めします。これを行うためのハイレベルな手順は次のとおりです。

1. Application Load Balancer に送信するリクエストに追加のカスタム HTTP ヘッダーを追加する CloudFront ように を設定します。
2. Application Load Balancer リスナールールを更新して、この追加のカスタム HTTP ヘッダーを含むリクエストを転送します。

3. Application Load Balancer に送信するリクエストへの元のカスタム HTTP ヘッダーの追加 CloudFront を停止するように を設定します。
4. Application Load Balancer リスナールールを更新して、元のカスタム HTTP ヘッダーを含むリクエストの転送を停止します。

これらの手順を実行する方法の詳細については、前述のセクションを参照してください。

コンテンツの地理的ディストリビューションの制限

地理的制限 は、地理的ブロッキング とも呼ばれ、特定の地理的場所のユーザーが Amazon CloudFront ディストリビューションを通じて配信するコンテンツにアクセスできないようにすることができます。地理的制限を使用するには、次の 2 つの方法があります。

- CloudFront 地理的制限機能を使用します。ディストリビューションに関連するすべてのファイルへのアクセスを制限し、国レベルでアクセスを制限する場合は、この方法を使用します。
- サードパーティーの位置情報サービスを使用する。ディストリビューションに関連するファイルのサブセットへのアクセスを制限する場合や、国レベルより詳細なレベルでアクセスを制限する場合は、この方法を使用します。

トピック

- [CloudFront 地理的制限の使用](#)
- [サードパーティーの位置情報サービスの使用](#)

CloudFront 地理的制限の使用

ユーザーがコンテンツをリクエストすると、CloudFront は通常、ユーザーがいる場所に関係なく、リクエストされたコンテンツを提供します。特定の国のユーザーがコンテンツにアクセスできないようにする必要がある場合は、CloudFront 地理的制限機能を使用して次のいずれかを実行できます。

- 承認された国の許可リストに含まれているいずれかの国にユーザーがいる場合のみ、コンテンツへのアクセス許可を付与する。
- ユーザーが拒否リストにある禁止国にいる場合、コンテンツへのアクセスを禁止する。

例えば、コンテンツの配信が許可されていない国からリクエストが送信された場合は、CloudFront 地理的制限を使用してリクエストをブロックできます。

Note

CloudFront は、サードパーティーのデータベースを使用してユーザーの場所を決定します。IP アドレスと国とのマッピングの正確さは、リージョンによって異なります。最近のテストによれば、全体的な正確性は 99.8% です。CloudFront がユーザーの場所を特定できない場合、はユーザーがリクエストしたコンテンツ CloudFront を提供します。

地理的制限は次のような仕組みになっています。

1. 仮に、コンテンツをリヒテンシュタインでのみ配信する権限を持っているとしましょう。デイス CloudFront トリビューションを更新して、リヒテンシュタインのみを含む許可リストを追加します。(または、リヒテンシュタイン以外のすべての国を含む拒否リストを追加することもできます)。
2. Monaco のユーザーはコンテンツをリクエストし、DNS はリクエストをイタリアのミラノの CloudFront エッジロケーションにルーティングします。
3. ミラノのエッジロケーションはお客さまのディストリビューションを検索し、モナコ王国のユーザーはコンテンツをダウンロードするアクセス許可がないと判断します。
4. CloudFront は HTTP ステータスコードをユーザーに返403 (Forbidden)します。

オプションで、カスタムエラーメッセージをユーザーに返す CloudFront ように を設定できます。また、リクエストされたファイルのエラーレスポンスをキャッシュ CloudFront する期間を指定できます。デフォルト値は 10 秒です。詳細については、「[特定の HTTP ステータスコードに対応するカスタムエラーページの作成](#)」を参照してください。

地理的制限はディストリビューション全体に適用されます。コンテンツの一部に 1 つの制限を適用し、コンテンツの別の部分に別の制限を適用する (または制限を適用しない) 必要がある場合は、別の CloudFront ディストリビューションを作成するか、[サードパーティーの位置情報サービスを使用する必要があります](#)。

CloudFront [標準ログ](#) (アクセスログ) を有効にすると、sc-status (HTTP ステータスコード) の値がであるログエントリを検索して、CloudFront 拒否されたリクエストを特定できます403。ただし、標準ログのみを使用して、ユーザーの場所に基づいて拒否された CloudFront リクエストと、別の理由でファイルにアクセスするアクセス許可がユーザーになかったために CloudFront 拒否されたリクエストを区別することはできません。Digital Element や などのサードパーティーの位置情報サービスを使用している場合は MaxMind、アクセスログの c-ip (クライアント IP) 列の IP アドレス

に基づいてリクエストの場所を特定できます。CloudFront 標準ログの詳細については、「」を参照してください [標準ログ \(アクセスログ\) の設定および使用](#)。

次の手順では、CloudFront コンソールを使用して既存のディストリビューションに地理的制限を追加する方法について説明します。コンソールを使用してディストリビューションを作成する方法の詳細については、「[ディストリビューションの作成](#)」を参照してください。

CloudFront ウェブディストリビューションに地理的制限を追加するには (コンソール)

1. にサインインAWS Management Consoleし、 で CloudFront コンソールを開きます <https://console.aws.amazon.com/cloudfront/v4/home>。
2. ナビゲーションペインで、[ディストリビューション] を選択し、更新するディストリビューションを選択します。
3. [セキュリティ] タブを選択し、[地理的制限] を選択します。
4. [編集] を選択します。
5. 許可した国のリストを作成する場合は、許可リスト、またはブロックした国のリストを作成する場合は、ブロックリストを作成します。
6. 目的の国をリストに追加し、[Save changes] (変更の保存) を選択します。

サードパーティーの位置情報サービスの使用

CloudFront 地理的制限機能を使用すると、特定のウェブディストリビューションで配信するすべてのファイルについて、コンテンツのディストリビューションを国レベルで制御できます。制限が国の境界に従わない地理的制限のユースケースがある場合、または特定のディストリビューションで処理しているファイルの一部のみにアクセスを制限する場合は、CloudFront サードパーティーの位置情報サービスと組み合わせることができます。これにより、国だけではなく、都市、郵便番号、または緯度/経度に基づいてコンテンツの制御を提供します。

サードパーティーの位置情報サービスを使用する場合は、CloudFront 署名URLsを使用することをお勧めします。この URL では、URL が有効でなくなるまで、有効期限の日時を指定できます。さらに、オリジン CloudFront [アクセスコントロールを使用して、ユーザーがオリジン](#)から直接コンテンツにアクセスできないようにできるため、オリジンとして Amazon S3 バケットを使用することをお勧めします。署名付き URL とオリジンアクセスコントロールの詳細については、「[署名付き URL と署名付き Cookie を使用したプライベートコンテンツの提供](#)」を参照してください。

以下のステップは、サードパーティーの位置情報サービスを使用してファイルへのアクセスを制御する方法を説明しています。

サードパーティーの位置情報サービスを使用してディストリビューション内のファイルへのアクセスを制限するには CloudFront

1. 位置情報サービスのアカウントを取得します。
2. コンテンツを Amazon S3 バケットにアップロードします。
3. プライベートコンテンツを提供するように Amazon CloudFront と Amazon S3 を設定します。Amazon S3 詳細については、「[署名付き URL と署名付き Cookie を使用したプライベートコンテンツの提供](#)」を参照してください。
4. 以下の処理を行うようにウェブアプリケーションを記述します。
 - 各ユーザーリクエストの IP アドレスを位置情報サービスに送信します。
 - 位置情報サービスからの戻り値を評価して、ユーザーがコンテンツを CloudFront 配信する場所にいるかどうかを判断します。
 - コンテンツをユーザーの場所に配信する場合は、CloudFront コンテンツの署名付き URL を生成します。コンテンツをその場所に配信しない場合、HTTP ステータスコード 403 (Forbidden) をユーザーに返します。または、カスタムエラーメッセージを返す CloudFront ように を設定することもできます。詳細については、「[the section called “特定の HTTP ステータスコードに対応するカスタムエラーページの作成”](#)」を参照してください。

詳細については、使用する位置情報サービスのドキュメントを参照してください。

ウェブサーバー変数を使用すると、ウェブサイトを訪れたユーザーの IP アドレスを取得できます。次の点に注意してください。

- ウェブサーバーがインターネットにロードバランサー経由で接続されていない場合、ウェブサーバー変数を使用してリモート IP アドレスを取得できます。ただし、この IP アドレスが必ずしもユーザーの IP アドレスであるとは限りません。ユーザーのインターネットへの接続方法によっては、プロキシサーバーの IP アドレスである可能性もあります。
- ウェブサーバーがインターネットにロードバランサー経由で接続されている場合、ウェブサーバー変数には、ユーザーの IP アドレスではなく、ロードバランサーの IP アドレスが含まれる可能性があります。この構成では、X-Forwarded-For HTTP ヘッダーに含まれる最後の IP アドレスを使用することをお勧めします。通常、このヘッダーには複数の IP アドレスが含まれており、そのほとんどはプロキシまたはロードバランサーの IP アドレスです。ユーザーの地理的な場所に関連付けられている可能性が最も高い IP アドレスは、リストの最後にある IP アドレスです。

ウェブサーバーがロードバランサーに接続されていない場合は、IP アドレスのスプーフィングを回避するために、X-Forwarded-For ヘッダーではなくウェブサーバー変数を使用することをお勧めします。

フィールドレベル暗号化を使用した機密データの保護

Amazon では CloudFront、HTTPS を使用してオリジンサーバーへの安全な end-to-end 接続を強制できます。フィールドレベル暗号化では、セキュリティのレイヤーが追加されます。これにより、システムの処理中に特定のデータに特定のアプリケーションのみがアクセスできるように、そのデータを保護できます。

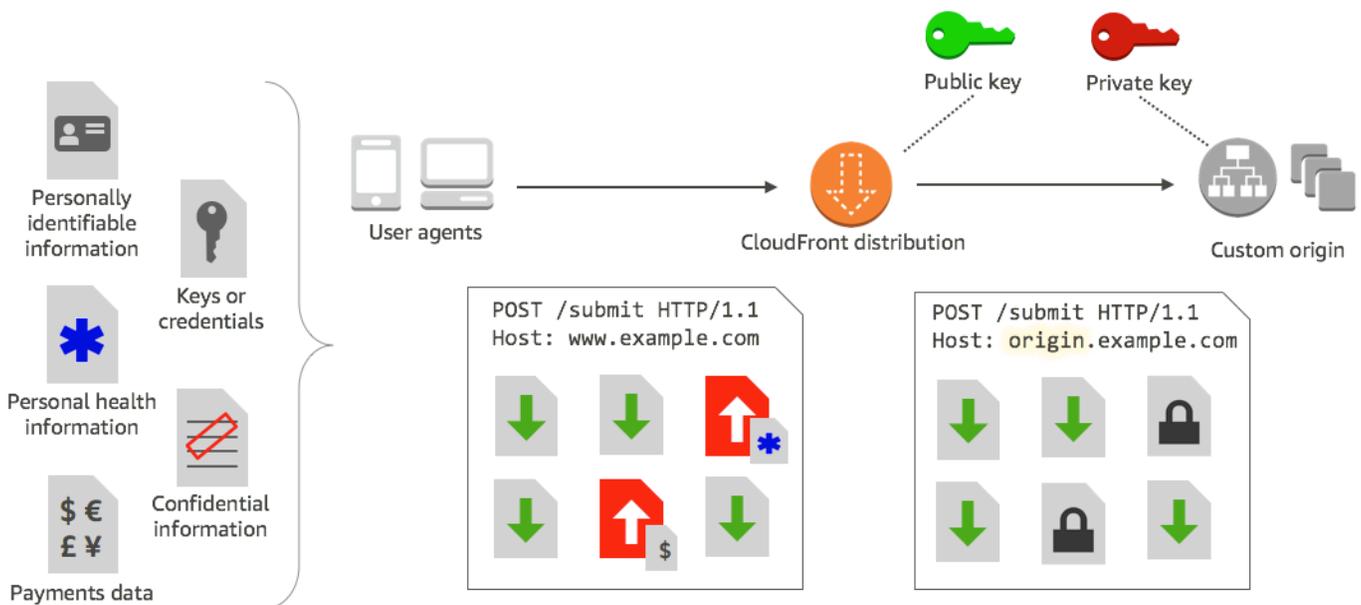
フィールドレベル暗号化により、ユーザーが機密情報をウェブサーバーに安全にアップロードできるようになります。ユーザーから提供される機密情報は、ユーザー近くのエッジで暗号化され、アプリケーションスタック全体で暗号化された状態が維持されます。この暗号化により、データを必要としており、復号するための認証情報を持つアプリケーションだけが暗号化できるようになります。

フィールドレベル暗号化を使用するには、CloudFront デイストリビューションを設定するときに、暗号化する POST リクエストのフィールドセットと、それらを暗号化するために使用するパブリックキーを指定します。リクエストの最大 10 個のデータフィールドを暗号化できます (フィールドレベル暗号化を使用して、リクエストのすべてのデータをまとめて暗号化することはできません。暗号化する個々のフィールドを指定する必要があります)。

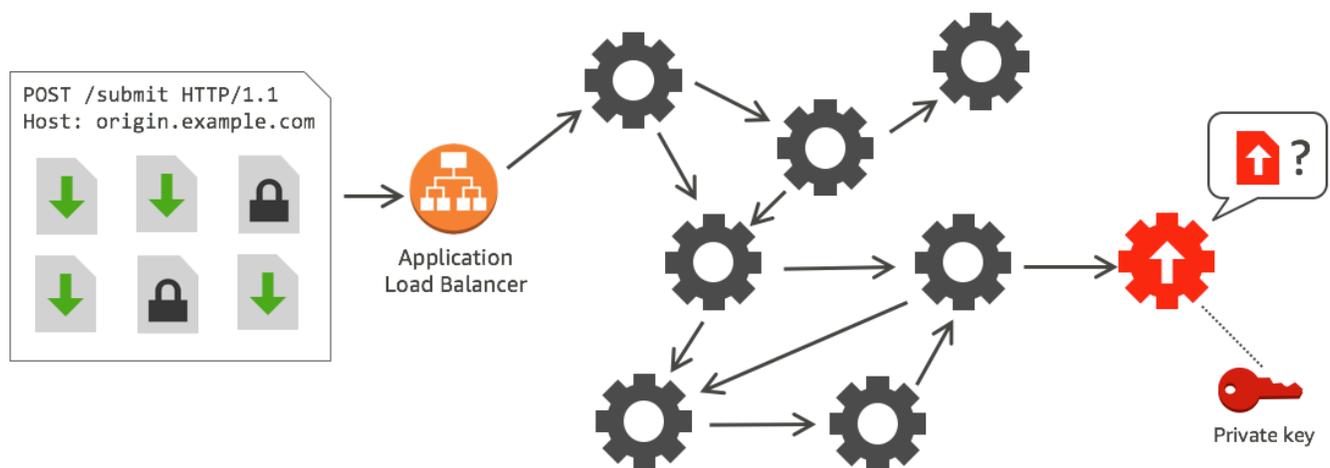
フィールドレベル暗号化を含む HTTPS リクエストがオリジンに転送され、リクエストがオリジンアプリケーションまたはサブシステム全体にルーティングされても、機密データは暗号化されたままです。それにより、機密データの漏洩や偶発的な損失のリスクが軽減されます。クレジット番号へのアクセスを必要とする支払い処理システムなど、ビジネス上の理由で機密データにアクセスする必要のあるコンポーネントは、適切なプライベートキーを使用してデータを復号化し、そのデータにアクセスできます。

Note

フィールドレベル暗号化を使用するには、オリジンがチャンクエンコードをサポートしている必要があります。



CloudFront フィールドレベル暗号化は、非対称暗号化を使用します。これはパブリックキー暗号化とも呼ばれます。にパブリックキーを指定すると CloudFront、指定したすべての機密データが自動的に暗号化されます。に提供するキー CloudFront は、暗号化された値の復号には使用できません。プライベートキーのみが復号できます。



トピック

- [フィールドレベル暗号化の概要](#)
- [フィールドレベル暗号化の設定](#)
- [オリジンでのデータフィールドの復号化](#)

フィールドレベル暗号化の概要

以下に示しているのは、フィールドレベル暗号化の設定手順の概要です。この手順の詳細については、「[フィールドレベル暗号化の設定](#)」を参照してください。

1. パブリックキーとプライベートキーのペアを取得する。CloudFront でフィールドレベル暗号化を設定する前に、パブリックキーを取得して追加する必要があります。
2. フィールドレベル暗号化のプロファイルを作成する。で作成するフィールドレベル暗号化プロファイルは CloudFront、暗号化するフィールドを定義します。
3. フィールドレベル暗号化の設定を作成する。設定では、特定のデータフィールドの暗号化に使用するプロファイル (リクエストのコンテンツタイプまたはクエリ引数に基づく) を指定します。また、さまざまなシナリオで必要なリクエスト転送動作オプションを選択することもできます。例えば、リクエスト URL のクエリ引数で指定されたプロファイル名が に存在しない場合の動作を設定できます CloudFront。
4. キャッシュ動作にリンクする。設定をディストリビューションのキャッシュ動作にリンクして、CloudFront がいつデータを暗号化するかを指定します。

フィールドレベル暗号化の設定

フィールドレベル暗号化を使用するには、以下の手順に従います。フィールドレベル暗号化のクォータ (以前は制限と呼ばれていました) の詳細については、「[クォータ](#)」を参照してください。

- [ステップ 1: RSA キーペアを作成する](#)
- [ステップ 2: パブリックキーを に追加する CloudFront](#)
- [ステップ 3: フィールドレベル暗号化のプロファイルを作成する](#)
- [ステップ 4: 設定を作成する](#)
- [ステップ 5: キャッシュ動作に設定を追加する](#)

ステップ 1: RSA キーペアを作成する

開始するには、パブリックキーとプライベートキーを含む RSA キーペアを作成する必要があります。パブリックキーを使用すると、CloudFront はデータを暗号化でき、プライベートキーを使用すると、オリジンのコンポーネントは暗号化されたフィールドを復号できます。OpenSSL または別のツールを使用してキーペアを作成できます。キーのサイズは 2,048 ビットであることが必要です。

たとえば、OpenSSL を使用する場合、次のコマンドを使用して 2048 ビット長のキーペアを生成し、`private_key.pem` ファイルに保存できます。

```
openssl genrsa -out private_key.pem 2048
```

生成されるファイルには、パブリックキーとプライベートキーの両方が含まれます。そのファイルからパブリックキーを抽出するには、次のコマンドを実行します。

```
openssl rsa -pubout -in private_key.pem -out public_key.pem
```

公開キーファイル (`public_key.pem`) には、次のステップで貼り付けるエンコードされたキー値が含まれています。

ステップ 2: パブリックキーを に追加する CloudFront

RSA キーペアを取得したら、パブリックキーを に追加します CloudFront。

パブリックキーを に追加するには CloudFront (コンソール)

1. にサインインAWS Management Consoleし、 で CloudFront コンソールを開きます<https://console.aws.amazon.com/cloudfront/v4/home>。
2. ナビゲーションペインで、[Public key (パブリックキー)] を選択します。
3. [Add public key (パブリックキーを追加)] を選択します。
4. [Key name (キー名)] にキーの一意の名前を入力します。キー名にはスペースを使用できません。英数字、アンダースコア (`_`)、ハイフン (`-`) のみを使用できます。最大長は 128 文字です。
5. [Key value (キー値)] に、-----BEGIN PUBLIC KEY----- および -----END PUBLIC KEY----- 行を含む、パブリックキーのエンコードされたキー値を貼り付けます。
6. [Comment (コメント)] で、オプションのコメントを追加します。たとえば、パブリックキーの有効期限を含めることができます。
7. [追加] を選択します。

手順のステップを繰り返す CloudFront ことで、 で使用するキーをさらに追加できます。

ステップ 3: フィールドレベル暗号化のプロファイルを作成する

に少なくとも 1 つのパブリックキーを追加したら CloudFront、暗号化CloudFront するフィールドを に指示するプロファイルを作成します。

フィールドレベル暗号化のプロファイルを作成するには (コンソール)

1. ナビゲーションペインで、[Field-level encryption (フィールドレベル暗号化)] を選択します。
2. [Create profile (プロファイルの作成)] を選択します。
3. 以下のフィールドに入力します。

Profile name

プロファイルの一意的な名前を入力します。キー名にはスペースを使用できません。英数字、アンダースコア (_)、ハイフン (-) のみを使用できます。最大長は 128 文字です。

Public key name

ドロップダウンリストで、ステップ 2 CloudFront で追加したパブリックキーの名前を選択します。CloudFront はキーを使用して、このプロファイルで指定したフィールドを暗号化します。

Provider name

キーを識別するのに役立つフレーズ (キーペアを取得したプロバイダーなど) を入力します。この情報は、プライベートキーと共に、アプリケーションがデータフィールドを復号化するときに必要になります。プロバイダー名にはスペースを使用できません。英数字、コロン (:)、アンダースコア (_)、ハイフン (-) のみを使用できます。最大長は 128 文字です。

Field name pattern to match

CloudFront で暗号化するデータフィールドの名前、またはリクエスト内のデータフィールド名を識別するパターンを入力します。+ オプションを選択して、このキーで暗号化するすべてのフィールドを追加します。

フィールド名パターンでは、`.*` のようなデータフィールドの名前全体を入力するか `DateOfBirth`、名前の最初の部分だけに `CreditCard*` のようなワイルドカード文字 (*) を付けることができます。フィールド名パターンには、オプションのワイルドカード文字 (*)に加えて、英数字、角かっこ ([および])、ピリオド (.)、アンダースコア (_)、ハイフン (-) のみを含める必要があります。

異なるフィールド名パターンに重複する文字を使用しないでください。たとえば、`ABC*` というフィールド名パターンがある場合、`AB*` という別のフィールド名パターンを追加することはできません。また、フィールド名は大文字と小文字が区別され、最大長は 128 文字です。

コメント

(オプション) このプロファイルに関するコメントを入力します。最大長は 128 文字です。

4. フィールドに入力した後、[Create profile (プロファイルの作成)] を選択します。
5. さらにプロファイルを追加する場合は、[Add profile (プロファイルの追加)] を選択します。

ステップ 4: 設定を作成する

1 つ以上のフィールドレベル暗号化プロファイルを作成したら、暗号化するデータを含むリクエストのコンテンツタイプ、暗号化に使用するプロファイル、および暗号化 CloudFront の処理方法を指定するその他のオプションを指定する設定を作成します。

例えば、CloudFront がデータを暗号化できない場合、次のシナリオで がリクエストをブロックするか、オリジンに転送するか CloudFront を指定できます。

- リクエストのコンテンツタイプが設定にない場合 – コンテンツタイプを設定に追加していない場合は、 がそのコンテンツタイプのリクエストをデータフィールドを暗号化せずにオリジンに転送 CloudFront するか、リクエストをブロックしてエラーを返すかを指定できます。

Note

コンテンツタイプを設定に追加しても、そのタイプで使用するプロファイルを指定していない場合、CloudFront は常にそのコンテンツタイプのリクエストをオリジンに転送します。

- クエリ引数で指定されたプロファイル名が不明な場合 – ディストリビューションに存在しないプロファイル名で `fle-profile` クエリ引数を指定すると、CloudFront がデータフィールドを暗号化せずにリクエストをオリジンに送信するか、リクエストをブロックしてエラーを返すかを指定できます。

設定で、URL でクエリ引数としてプロファイルを提供するかどうかを指定して、そのクエリのコンテンツタイプにマッピングしたプロファイルを上書きすることもできます。デフォルトでは、コンテンツタイプを指定した場合はコンテンツタイプにマッピングしたプロファイル CloudFront を使用します。これにより、デフォルトで使用されるプロファイルを提供できますが、特定のリクエストに使用される別のプロファイルを指定することもできます。

たとえば、使用するクエリ引数プロファイルとして (設定で) **SampleProfile** を指定できます。その後、`https://d1234.cloudfront.net?fle-profile=SampleProfile` の代わりに URL を

使用して `https://d1234.cloudfront.net`、リクエストのコンテンツタイプ `SampleProfile` に設定したプロファイルの代わりに、このリクエストに CloudFront を使用できます。

1 つのアカウント用に最大 10 個の設定を作成し、いずれかの設定をそのアカウントの任意のディストリビューションのキャッシュ動作に関連付けることができます。

フィールドレベル暗号化の設定を作成するには (コンソール)

1. [Field-level encryption (フィールドレベル暗号化)] ページで、[Create configuration (設定の作成)] を選択します。

注意: プロファイルを 1 つ以上作成していない場合、設定を作成するオプションは表示されません。

2. 以下のフィールドに入力して、使用するプロファイルを指定します (一部のフィールドは変更できません)。

Content type (変更不可)

コンテンツタイプは `application/x-www-form-urlencoded` に設定されており、変更することはできません。

Default profile ID (オプション)

ドロップダウンリストから、[Content type (コンテンツタイプ)] フィールドでコンテンツタイプにマッピングするプロファイルを選択します。

Content format (変更不可)

コンテンツ形式は `URLencoded` に設定されており、変更することはできません。

3. 以下のオプションの CloudFront デフォルト動作を変更する場合は、適切なチェックボックスをオンにします。

Forward request to origin when request's content type is not configured

リクエストのコンテンツタイプに使用するプロファイルを指定していないときに、リクエストをオリジンに転送させる場合は、このチェックボックスをオンにします。

Override the profile for a content type with a provided query argument

クエリ引数で指定したプロファイルによって、コンテンツタイプ用に指定したプロファイルを上書きする場合は、このチェックボックスをオンにします。

- このチェックボックスをオンにして、クエリ引数によってデフォルトプロファイルが上書きされるようにした場合は、設定で以下の追加フィールドを入力する必要があります。クエリで使用するこれらのクエリ引数マッピングは最大 5 つ作成できます。

Query argument

file-profile クエリ引数の URL に含める値を入力します。CloudFront は、このクエリ引数の値に関連付けられたプロファイル ID (次のフィールドで指定) をこのクエリのフィールドレベル暗号化に使用します。

最大長は 128 文字です。値にはスペースを含めることはできません。英数字、ダッシュ (-)、ピリオド (.)、アンダースコア (_)、アスタリスク (*)、プラス記号 (+)、パーセント (%) のみを使用する必要があります。

Profile ID

ドロップダウンリストから、[Query argument (クエリ引数)] に入力した値に関連付けるプロファイルを選択します。

Forward request to origin when the profile specified in a query argument does not exist

クエリ引数で指定したプロファイルが CloudFront で定義されていないときに、リクエストをオリジンに転送させる場合は、このチェックボックスをオンにします。

ステップ 5: キャッシュ動作に設定を追加する

フィールドレベル暗号化を使用するには、設定 ID をディストリビューション用の値として追加して、ディストリビューションのキャッシュ動作に設定をリンクします。

Important

フィールドレベルの暗号化設定をキャッシュ動作にリンクするには、常に HTTPS を使用し、ビューワからの HTTP POST および PUT リクエストを受け入れるようにディストリビューションを設定する必要があります。つまり、以下が満たされている必要があります。

- キャッシュ動作の [ビューワプロトコルポリシーは] が [HTTP を HTTPS にリダイレクト] または [HTTPS のみ] に設定されている必要があります。(AWS CloudFormation または CloudFront API では、を `redirect-to-https` または `https-only` に設定 ViewerProtocolPolicy する必要があります)

- キャッシュ動作の [Allowed HTTP Methods] (許可された HTTP メソッド) が、[GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE] に設定されている必要があります。(AWS CloudFormation または CloudFront API では、 を GET、HEAD、OPTIONS、PUT、POST、PATCH、 に設定 AllowedMethods する必要があります DELETE。これらは任意の順序で指定できます)。
- オリジン設定の [オリジンプロトコルポリシー] が [ビューワーに合わせる] または [HTTPS のみ] に設定されている必要があります。(AWS CloudFormation または CloudFront API では、 を match-viewer または に設定 OriginProtocolPolicy する必要があります) https-only。

詳細については、「[ディストリビューションを作成または更新する場合に指定する値](#)」を参照してください。

オリジンでのデータフィールドの復号化

CloudFront は、 を使用してデータフィールドを暗号化します [AWS Encryption SDK](#)。データはアプリケーションスタック全体で暗号化されたままであり、復号化するための認証情報のあるアプリケーションによってのみアクセスできます。

暗号化の後、暗号テキストは base64 でエンコードされます。アプリケーションがオリジンでテキストを復号化するときには、まず暗号テキストをデコードしてから、AWS Encryption SDK を使用してデータを復号化する必要があります。

以下のサンプルコードでは、アプリケーションがオリジンでデータを復号化する方法を示しています。次の点に注意してください。

- わかりやすい例になるように、このサンプルは作業ディレクトリのファイルからパブリックキーとプライベートキー (DER 形式) を読み込みます。実際には、オフラインのハードウェアセキュリティモジュールなどの安全なオフラインの場所にプライベートキーを保存し、開発チームにパブリックキーを配布します。
- CloudFront はデータを暗号化する際に特定の情報を使用します。データを復号するには、オリジンで同じパラメータセットを使用する必要があります。の初期化時に が使用するパラメータ CloudFront MasterKey には、次のものがあります。
- PROVIDER_NAME: フィールドレベル暗号化のプロファイルを作成したときにこの値を指定しました。同じ値をここで使用します。

- KEY_NAME: パブリックキーの名前を にアップロードしたときに作成し CloudFront、プロファイルでキー名を指定しました。同じ値をここで使用します。
- ALGORITHM: 暗号化のアルゴリズムRSA/ECB/OAEPWithSHA-256AndMGF1Paddingとして CloudFront を使用するため、データを復号するには同じアルゴリズムを使用する必要があります。
- 暗号テキストを入力として以下のサンプルプログラムを実行すると、復号化されたデータがコンソールに出力されます。詳細については、[Encryption SDK の「Java サンプルコードAWS」](#)を参照してください。

サンプルコード

```
import java.nio.file.Files;
import java.nio.file.Paths;
import java.security.KeyFactory;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;

import org.apache.commons.codec.binary.Base64;

import com.amazonaws.encryptionsdk.AwsCrypto;
import com.amazonaws.encryptionsdk.CryptoResult;
import com.amazonaws.encryptionsdk.jce.JceMasterKey;

/**
 * Sample example of decrypting data that has been encrypted by CloudFront field-level
 * encryption.
 */
public class DecryptExample {

    private static final String PRIVATE_KEY_FILENAME = "private_key.der";
    private static final String PUBLIC_KEY_FILENAME = "public_key.der";
    private static PublicKey publicKey;
    private static PrivateKey privateKey;

    // CloudFront uses the following values to encrypt data, and your origin must use
    // same values to decrypt it.
```

```
// In your own code, for PROVIDER_NAME, use the provider name that you specified
when you created your field-level
// encryption profile. This sample uses 'DEMO' for the value.
private static final String PROVIDER_NAME = "DEMO";
// In your own code, use the key name that you specified when you added your public
key to CloudFront. This sample
// uses 'DEMOKEY' for the key name.
private static final String KEY_NAME = "DEMOKEY";
// CloudFront uses this algorithm when encrypting data.
private static final String ALGORITHM = "RSA/ECB/OAEPWithSHA-256AndMGF1Padding";

public static void main(final String[] args) throws Exception {

    final String dataToDecrypt = args[0];

    // This sample uses files to get public and private keys.
    // In practice, you should distribute the public key and save the private key
in secure storage.
    populateKeyPair();

    System.out.println(decrypt(debase64(dataToDecrypt)));
}

private static String decrypt(final byte[] bytesToDecrypt) throws Exception {
    // You can decrypt the stream only by using the private key.

    // 1. Instantiate the SDK
    final AwsCrypto crypto = new AwsCrypto();

    // 2. Instantiate a JCE master key
    final JceMasterKey masterKey = JceMasterKey.getInstance(
        publicKey,
        privateKey,
        PROVIDER_NAME,
        KEY_NAME,
        ALGORITHM);

    // 3. Decrypt the data
    final CryptoResult <byte[], ? > result = crypto.decryptData(masterKey,
bytesToDecrypt);
    return new String(result.getResult());
}

// Function to decode base64 cipher text.
```

```
private static byte[] debase64(final String value) {
    return Base64.decodeBase64(value.getBytes());
}

private static void populateKeyPair() throws Exception {
    final byte[] PublicKeyBytes =
Files.readAllBytes(Paths.get(PUBLIC_KEY_FILENAME));
    final byte[] privateKeyBytes =
Files.readAllBytes(Paths.get(PRIVATE_KEY_FILENAME));
    publicKey = KeyFactory.getInstance("RSA").generatePublic(new
X509EncodedKeySpec(PublicKeyBytes));
    privateKey = KeyFactory.getInstance("RSA").generatePrivate(new
PKCS8EncodedKeySpec(privateKeyBytes));
}
}
```

キャッシュの最適化と可用性

このセクションでは、オブジェクトのキャッシュを設定および管理してパフォーマンスを向上させ、ビジネス要件を満たす方法を説明します。

配信するコンテンツの追加と削除については、CloudFront「」を参照してください [が配信する CloudFront コンテンツの追加、削除、または置き換え](#)。

トピック

- [CloudFront エッジロケーションでのキャッシュの仕組み](#)
- [CloudFront キャッシュから直接提供されるリクエストの割合を増やす \(キャッシュヒット率\)](#)
- [Amazon CloudFront Origin Shield の使用](#)
- [CloudFront オリジンフェイルオーバーによる高可用性の最適化](#)
- [コンテンツがキャッシュに保持される期間 \(有効期限\) の管理](#)
- [クエリ文字列パラメータに基づくコンテンツのキャッシュ](#)
- [Cookie に基づくコンテンツのキャッシュ](#)
- [リクエストヘッダーに基づくコンテンツのキャッシュ](#)

CloudFront エッジロケーションでのキャッシュの仕組み

を使用する目的の 1 つは、オリジンサーバーが直接応答する必要があるリクエストの数を減らす CloudFront ことです。CloudFront キャッシュを使用すると、ユーザーに近いエッジロケーションから CloudFront より多くのオブジェクトが供給されます。これにより、オリジンサーバーの負荷が軽減され、レイテンシーが減少します。

エッジキャッシュから が処理 CloudFront できるリクエストが多いほど、オブジェクトの最新バージョンまたは一意のバージョンを取得するためにオリジンに転送する必要があるビューワーリクエスト CloudFront が少なくなります。オリジンへのリクエストをできるだけ少なく CloudFront するように最適化するには、CloudFront Origin Shield の使用を検討してください。詳細については、「[Amazon CloudFront Origin Shield の使用](#)」を参照してください。

すべてのリクエストに対する CloudFront キャッシュから直接提供されるリクエストの割合は、キャッシュヒット率と呼ばれます。CloudFront コンソールで、ヒット、ミス、エラーであるビューワーリクエストの割合を表示できます。詳細については、「[CloudFront キャッシュ統計レポート](#)」を参照してください。

キャッシュヒット率に影響を与えるいくつかの要因があります。「」のガイダンスに従って、キャッシュヒット率を向上させるように CloudFront デистриビューション設定を調整できます。[CloudFront キャッシュから直接提供されるリクエストの割合を増やす \(キャッシュヒット率\)](#)。

CloudFront キャッシュから直接提供されるリクエストの割合を増やす (キャッシュヒット率)

コンテンツのためにオリジンサーバーにアクセスする代わりに、CloudFront キャッシュから直接提供されるビューワーリクエストの割合を増やすことで、パフォーマンスを向上させることができます。これは、キャッシュヒット率の向上として知られています。

以下のセクションでは、キャッシュヒット率を改善する方法について説明します。

トピック

- [がオブジェクトを CloudFront キャッシュする期間を指定する](#)
- [Origin Shield の使用](#)
- [クエリ文字列パラメータに基づくキャッシュ](#)
- [Cookie 値に基づくキャッシュ](#)
- [リクエストヘッダーに基づくキャッシュ](#)
- [圧縮が不要な場合に Accept-Encoding ヘッダーを削除する](#)
- [HTTP を使用したメディアコンテンツの提供](#)

がオブジェクトを CloudFront キャッシュする期間を指定する

キャッシュヒット率を向上させるには、[Cache-Control max-age](#) ディレクティブをオブジェクトに追加し、max-age に対して最も長い実用的な値を指定するようにオリジンを設定します。キャッシュ期間が短いほど、オブジェクトが変更されているかどうかを判断し、最新バージョンを取得するための CloudFront リクエストをオリジンに送信する頻度が高くなります。stale-while-revalidate および stale-if-error ディレクティブで max-age を補足すると、特定の条件下でのキャッシュヒット率をさらに向上させることができます。詳細については、「[コンテンツがキャッシュに保持される期間 \(有効期限\) の管理](#)」を参照してください。

Origin Shield の使用

CloudFront Origin Shield は、オリジンの前に追加のキャッシュレイヤーを提供するため、CloudFront デистриビューションのキャッシュヒット率を向上させるのに役立ちます。Origin

Shield を使用すると、のすべての CloudFront キャッシュレイヤーからオリジンへのすべてのリクエストは 1 つの場所から送信されます。は Origin Shield からの 1 つのオリジンリクエストを使用して各オブジェクトを取得 CloudFront でき、キャッシュの CloudFront 他のすべてのレイヤー (エッジロケーションと [リージョン別エッジキャッシュ](#)) は Origin Shield からオブジェクトを取得できます。

詳細については、「[Amazon CloudFront Origin Shield の使用](#)」を参照してください。

クエリ文字列パラメータに基づくキャッシュ

クエリ文字列パラメータに基づいてキャッシュ CloudFront するように を設定する場合、次の操作を行うとキャッシュを改善できます。

- オリジンが一意のオブジェクトを返すクエリ文字列パラメータのみを転送する CloudFront ように設定します。
- 同じパラメータのすべてのインスタンスで大文字と小文字の区別を統一する。例えば、あるリクエストに が含まれ、parameter1=A別のリクエストに が含まれている場合parameter1=a、リクエストに が含まれparameter1=A、リクエストに が含まれている場合、オリジンへの CloudFront 個別のリクエスト。CloudFront オブジェクトが同一であってもparameter1=a、はオリジンから返された対応するオブジェクトを個別にキャッシュします。A または のみを使用するとa、オリジンへのリクエストが少なく CloudFront なります。
- パラメータの順序を統一する。大文字と小文字が区別されることと同じように、あるオブジェクトに対するリクエストに parameter1=a¶meter2=b というクエリ文字列が含まれており、同じオブジェクトに対する別のリクエストに parameter2=b¶meter1=a が含まれている場合、オブジェクトは同一であっても、CloudFront は両方のリクエストをオリジンに転送し、対応するオブジェクトを個別にキャッシュします。パラメータの順序を統一すると、CloudFront がオリジンに転送するリクエストが少なくなります。

詳細については、「[クエリ文字列パラメータに基づくコンテンツのキャッシュ](#)」を参照してください。がオリジン CloudFront に転送するクエリ文字列を確認する場合は、CloudFront ログファイルの cs-uri-query列の値を参照してください。詳細については、「[標準ログ \(アクセスログ\) の設定および使用](#)」を参照してください。

Cookie 値に基づくキャッシュ

Cookie 値に基づいてキャッシュ CloudFront するように を設定する場合、次の操作を行うとキャッシュを改善できます。

- すべての Cookie を転送する代わりに、指定された Cookie のみを転送 CloudFront するように設定します。オリジンに転送 CloudFront するように設定した Cookie CloudFront については、Cookie の名前と値の組み合わせごとに が転送されます。その場合、オリジンが返すオブジェクトがすべて同一であっても、別々にキャッシュされます。

例えば、ビューワーがリクエストごとに 2 つの Cookie を含み、各 Cookie に 3 つの可能な値があり、すべての Cookie 値の組み合わせが可能であるとします。CloudFront では、オブジェクトごとにオリジンに対して最大 6 つの異なるリクエストを行うことができます。オリジンが Cookie の 1 つだけに基いて異なるバージョンのオブジェクトを返す場合、CloudFront は必要以上のリクエストをオリジンに転送し、オブジェクトの複数の同一バージョンを不必要にキャッシュします。

- 静的コンテンツと動的コンテンツに対してそれぞれ異なるキャッシュ動作を作成し、動的コンテンツの場合にのみ Cookie をオリジンに転送するように CloudFront を設定する。

例えば、ディストリビューションに対してキャッシュ動作が 1 つしかなく、.js ファイルなどの動的コンテンツと、ほとんど変更されない .css ファイルの両方にディストリビューションを使用しているとします。CloudFront は、Cookie 値に基づいて個別のバージョンの .css ファイルをキャッシュするため、各 CloudFront エッジロケーションは新しい Cookie 値または Cookie 値の組み合わせごとにリクエストをオリジンに転送します。

パスパターンが *.css で、Cookie 値に基づいて CloudFront がキャッシュしないキャッシュ動作を作成する場合、は、エッジロケーションが特定の .css ファイルに対して受信する最初のリクエストと .css、ファイルの有効期限が切れた後の最初のリクエストに対してのみ、.css ファイルのリクエストをオリジン CloudFront に転送します。

- Cookie 値がユーザーごとに一意である (ユーザー ID など) 動的コンテンツと、より少ない数の一意の値に基づいて変化する動的コンテンツに対してそれぞれ異なるキャッシュ動作を作成する (可能な場合)。

詳細については、「[Cookie に基づくコンテンツのキャッシュ](#)」を参照してください。がオリジン CloudFront に転送する Cookie を確認する場合は、CloudFront ログファイルの cs(Cookie) 列の値を参照してください。詳細については、「[標準ログ \(アクセスログ\) の設定および使用](#)」を参照してください。

リクエストヘッダーに基づくキャッシュ

リクエストヘッダーに基づいてキャッシュ CloudFront するように を設定する場合、次の操作を行うとキャッシュを改善できます。

- すべてのヘッダーに基づいて転送およびキャッシュするのではなく、指定されたヘッダーのみに基づいて転送およびキャッシュ CloudFront するように を設定します。指定したヘッダーについて、CloudFront はヘッダー名と値のすべての組み合わせを転送します。その場合、オリジンが返すオブジェクトがすべて同一であっても、別々にキャッシュされます。

Note

CloudFront は、次のトピックで指定されたヘッダーを常にオリジンに転送します。

- ガリクエスト CloudFront を処理して Amazon S3 オリジンサーバーに転送する方法 > [CloudFront 削除または更新する HTTP リクエストヘッダー](#)
- ガリクエスト CloudFront を処理してカスタムオリジンサーバーに転送する方法 > [HTTP リクエストヘッダーと CloudFront 動作 \(カスタムオリジンと Amazon S3 オリジン\)](#)

リクエストヘッダーに基づいてキャッシュ CloudFront するように を設定する場合、 が CloudFront 転送するヘッダーは変更されず、 がヘッダー値に基づいてオブジェクトを CloudFront キャッシュする場合に限ります。

- 多数の一意の値を持つリクエストヘッダーに基づいてキャッシュすることを可能な限り回避する。

例えば、ユーザーのデバイスに基づいてさまざまなサイズのイメージを供給する場合は、大量の値を持つ User-Agentヘッダーに基づいてキャッシュ CloudFront するように を設定しないでください。代わりに、CloudFront デバイスタイプのヘッダー、CloudFront-Is-Desktop-Viewer、CloudFront-Is-SmartTV-Viewerおよび に基づいてキャッシュ CloudFront するように CloudFront-Is-Mobile-Viewerを設定しますCloudFront-Is-Tablet-Viewer。さらに、タブレットとデスクトップに同じバージョンのイメージを返す場合は、CloudFront-Is-Tablet-Viewer ヘッダーではなく CloudFront-Is-Desktop-Viewer ヘッダーのみを転送します。

詳細については、「[リクエストヘッダーに基づくコンテンツのキャッシュ](#)」を参照してください。

圧縮が不要な場合に **Accept-Encoding** ヘッダーを削除する

オリジンが圧縮をサポートしていないため、サポート CloudFront していないか、コンテンツが圧縮可能でないため、ディストリビューション内のキャッシュ動作を Custom Origin Header を設定するオリジンに関連付けることで、キャッシュヒット率を高めることができます。

- ヘッダー名: Accept-Encoding

- ヘッダー値: (空白のままにする)

この設定を使用すると、はキャッシュキーから Accept-Encodingヘッダー CloudFront を削除し、オリジンリクエストにヘッダーを含めません。この設定は、そのオリジンからのディストリビューションと CloudFront 連携するすべてのコンテンツに適用されます。

HTTP を使用したメディアコンテンツの提供

ビデオオンデマンド (VOD) およびビデオコンテンツのストリーミングの最適化の詳細については、「[によるビデオオンデマンドおよびライブストリーミングビデオ CloudFront](#)」を参照してください。

Amazon CloudFront Origin Shield の使用

CloudFront Origin Shield は、オリジンの負荷を最小限に抑え、可用性を向上させ、運用コストを削減するのに役立つ CloudFront キャッシュインフラストラクチャの追加レイヤーです。CloudFront Origin Shield を使用すると、次の利点が得られます。

キャッシュヒット率の向上

Origin Shield は、オリジンの前に追加のキャッシュレイヤーを提供するため、CloudFront ディストリビューションのキャッシュヒット率を向上させるのに役立ちます。Origin Shield を使用すると、のすべての CloudFront キャッシュレイヤーからオリジンへのすべてのリクエストが Origin Shield を通過し、キャッシュヒットの可能性が高まります。CloudFront は Origin Shield からオリジンへの単一のオリジンリクエストで各オブジェクトを取得でき、CloudFront キャッシュの他のすべてのレイヤー (エッジロケーションと [リージョン別エッジキャッシュ](#)) は Origin Shield からオブジェクトを取得できます。

オリジンの負荷を軽減

Origin Shield を使用すると、オリジンに送信される同じオブジェクトへの [同時リクエスト](#) の数をさらに減らすことができます。Origin Shield のキャッシュにないコンテンツへのリクエストは、同じオブジェクトに対する他のリクエストと統合され、オリジンに送信されるリクエストは 1 件のみになります。オリジンで処理するリクエスト数を減らすと、ピーク負荷や予期しないトラフィックの急増時にオリジンの可用性を維持でき、パッケージ化、画像変換、データ転送アウト (DTO) などの just-in-time コストを削減できます。

ネットワークパフォーマンスの向上

[オリジンに対するレイテンシーが最も低い](#) AWS リージョンで Origin Shield を有効にすることで、最良のネットワークパフォーマンスが得られます。AWS リージョンのオリジンの場合、CloudFront ネットワークトラフィックはオリジンまで高スループット CloudFront ネットワーク上に留まります。他のオリジンの場合AWS、CloudFront ネットワークトラフィックはオリジンへの接続レイテンシーが低い Origin Shield に到達するまで CloudFront ネットワーク上に残ります。

Origin Shield の使用には追加料金が発生します。詳細については、「[のCloudFront 料金](#)」を参照してください。

トピック

- [Origin Shield のユースケース](#)
- [Origin Shield の AWS リージョンの選択](#)
- [Origin Shield の有効化](#)
- [Origin Shield のコストの見積もり](#)
- [Origin Shield の高可用性](#)
- [Origin Shield が他の CloudFront 機能とやり取りする方法](#)

Origin Shield のユースケース

CloudFront Origin Shield は、次のような多くのユースケースに役立ちます。

- 異なる地理的リージョンにビューワーが分散している場合
- ライブストリーミングまたは on-the-fly画像処理の just-in-time パッケージを提供するオリジン
- オンプレミスのオリジンに、容量または帯域幅の制約がある場合
- ワークロードが複数のコンテンツ配信ネットワーク (CDN) を使用する場合

Origin Shield は、動的コンテンツがオリジンにプロキシ化される場合、コンテンツのキャッシュ可能性が低い安倍、コンテンツのリクエスト頻度の低い場合など、上記以外の状況には適さないことがあります。

以下のセクションでは、以下のユースケースにおける Origin Shield の利点について説明します。

ユースケース

- [異なる地理的リージョンにビューワーが分散している場合](#)
- [複数の CDN を使用する場合](#)

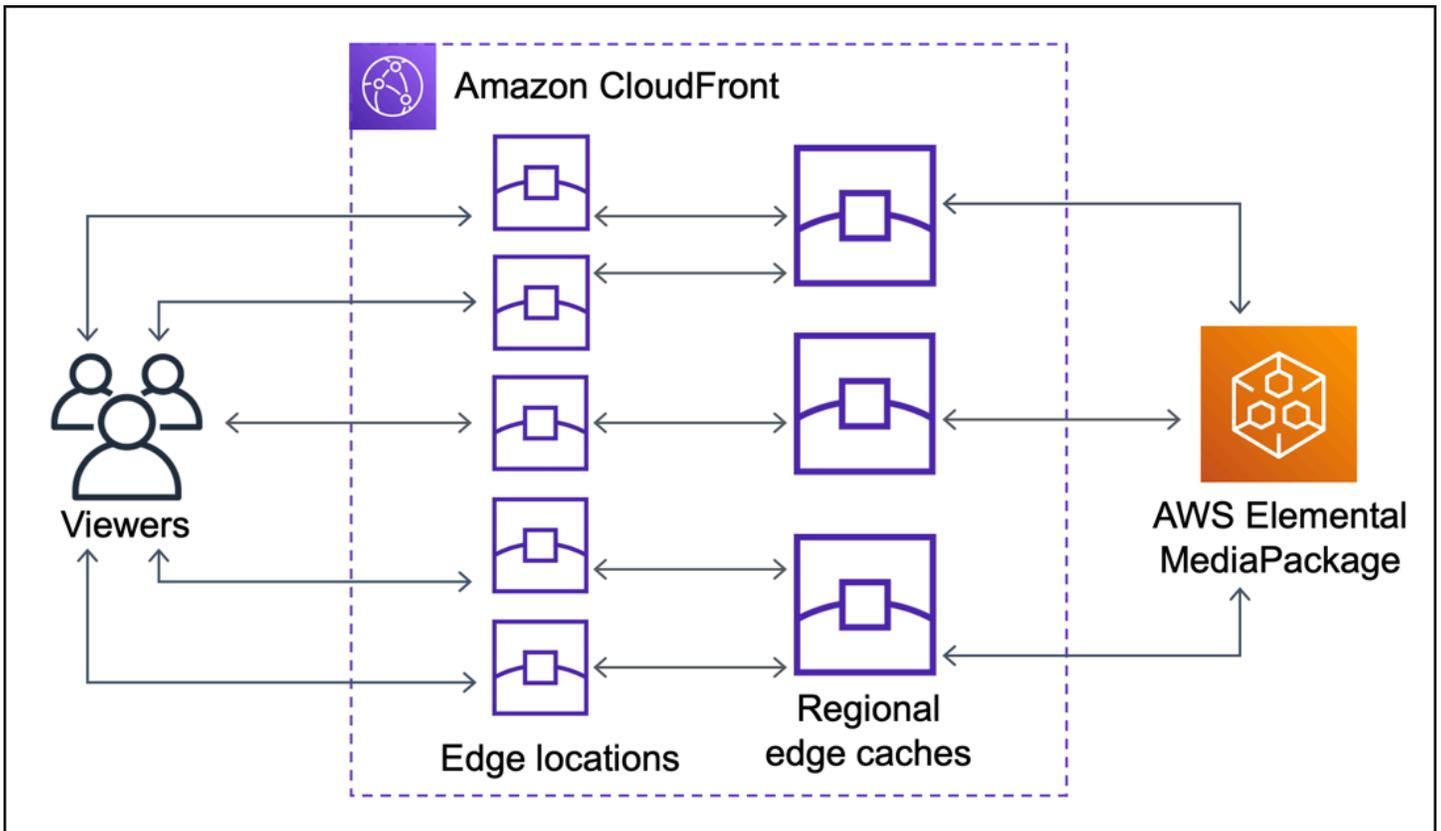
異なる地理的リージョンにビューワーが分散している場合

Amazon では CloudFront、キャッシュから処理 CloudFront できるリクエストはオリジンに送信されないため、本質的にオリジンの負荷が軽減されます。エッジロケーションのグローバル CloudFront ネットワークに加えて、[リージョン別エッジキャッシュ](#)は中間層のキャッシュレイヤーとして機能し、地理的に近いリージョンのビューワーにキャッシュヒットを提供し、オリジンリクエストを統合します。https://aws.amazon.com/cloudfront/features/#Amazon_CloudFront_Infrastructureビューワーリクエストは、まず近くの CloudFront エッジロケーションにルーティングされ、オブジェクトがそのロケーションにキャッシュされていない場合、リクエストはリージョン別エッジキャッシュに送信されます。

ビューワーのリージョンが地理的に異なる場合、リクエストは異なるリージョン別エッジキャッシュを介してルーティングされ、それぞれから同じコンテンツに対するリクエストがオリジンに送信される可能性があります。Origin Shield を使用すると、リージョン別エッジキャッシュとオリジンの間にキャッシュのレイヤーが追加されます。すべてのリージョン別エッジキャッシュからのリクエストはすべて、Origin Shield を通過し、オリジンの負荷をさらに軽減します。次の図にその概念を示します。次の図で、オリジンは AWS Elemental MediaPackage です。

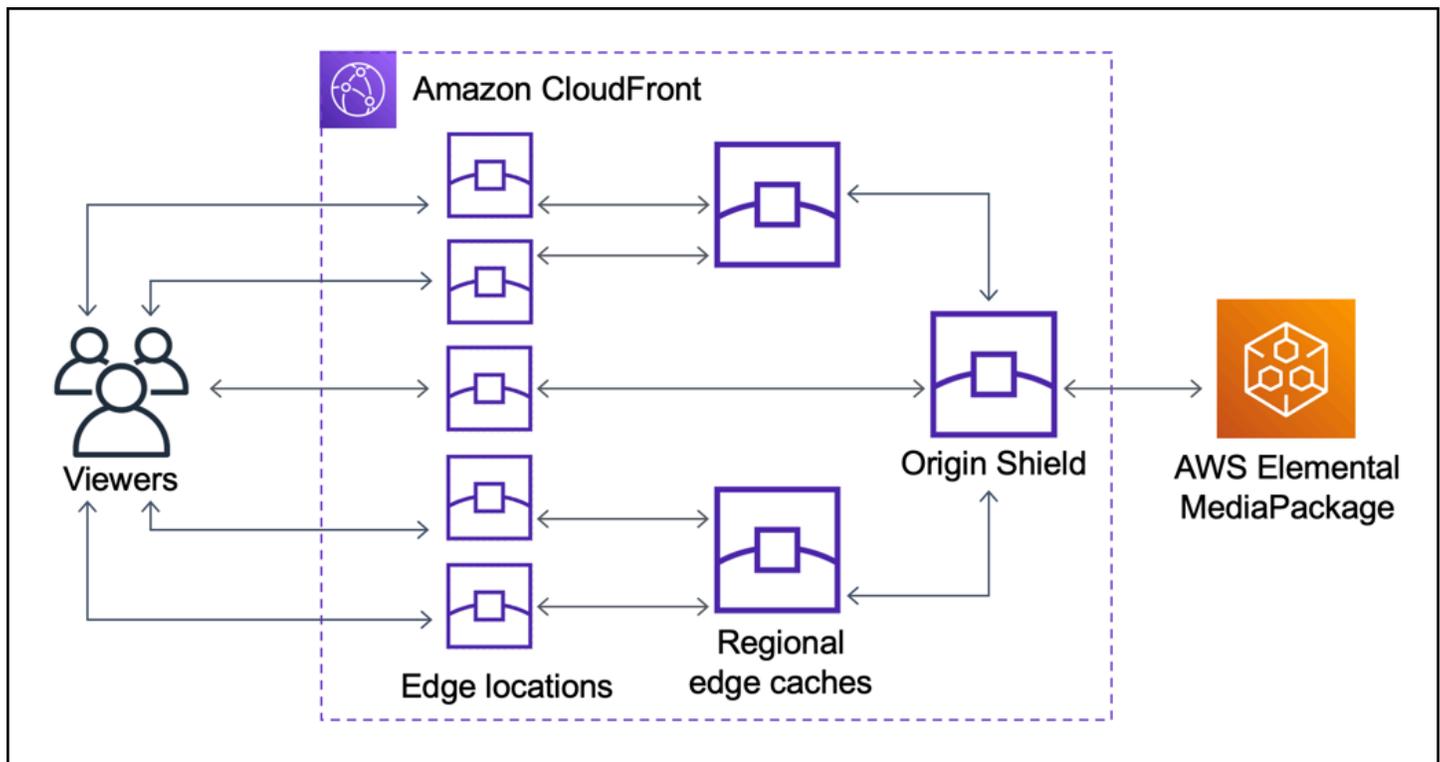
Origin Shield を使用しない場合

Origin Shield を使用しない場合は、次の図に示されているように、同じコンテンツに対するリクエストをオリジンが重複して受け取る可能性があります。



Origin Shield を使用する場合

Origin Shield を使用すると、次の図に示されているように、オリジンの負荷を軽減できます。



複数の CDN を使用する場合

ライブビデオイベントや人気のあるオンデマンドコンテンツを配信するには、複数のコンテンツ配信ネットワーク (CDN) を使用することがあります。複数の CDN の使用には利点もありますが、同じコンテンツに対して多数の重複リクエストをオリジンが受け取る可能性があります。それぞれのリクエストは、異なる CDN から送信されることも、同じ CDN 内の異なる場所から送信されることもあります。これらの冗長リクエストは、オリジンの可用性に悪影響を及ぼしたり、just-in-time パッケージ化やインターネットへのデータ転送 (DTO) などのプロセスの運用コストが増加したりする可能性があります。

Origin Shield と他の CDN のオリジンとして CloudFront デイストリビューションを使用すると、次の利点が得られます。

- オリジンで受信される冗長リクエストが少なくなるため、複数の CDN を使用した場合の悪影響を軽減できます。
- CDN 全体で共通の [キャッシュキー](#) を使用し、オリジン向けの機能を一元管理できます。
- ネットワークパフォーマンスの向上。他の CDN からのネットワークトラフィックは近くの CloudFront エッジロケーションで終了し、ローカルキャッシュからヒットが発生する可能性があります。リクエストされたオブジェクトがエッジロケーションキャッシュにない場合、オリジンへのリクエストは Origin Shield に到達するまで CloudFront ネットワーク上に留まり、オリジンへ

の高スループットと低レイテンシーを実現します。リクエストされたオブジェクトが Origin Shield のキャッシュ内にある場合、オリジンへのリクエストは完全に回避されます。

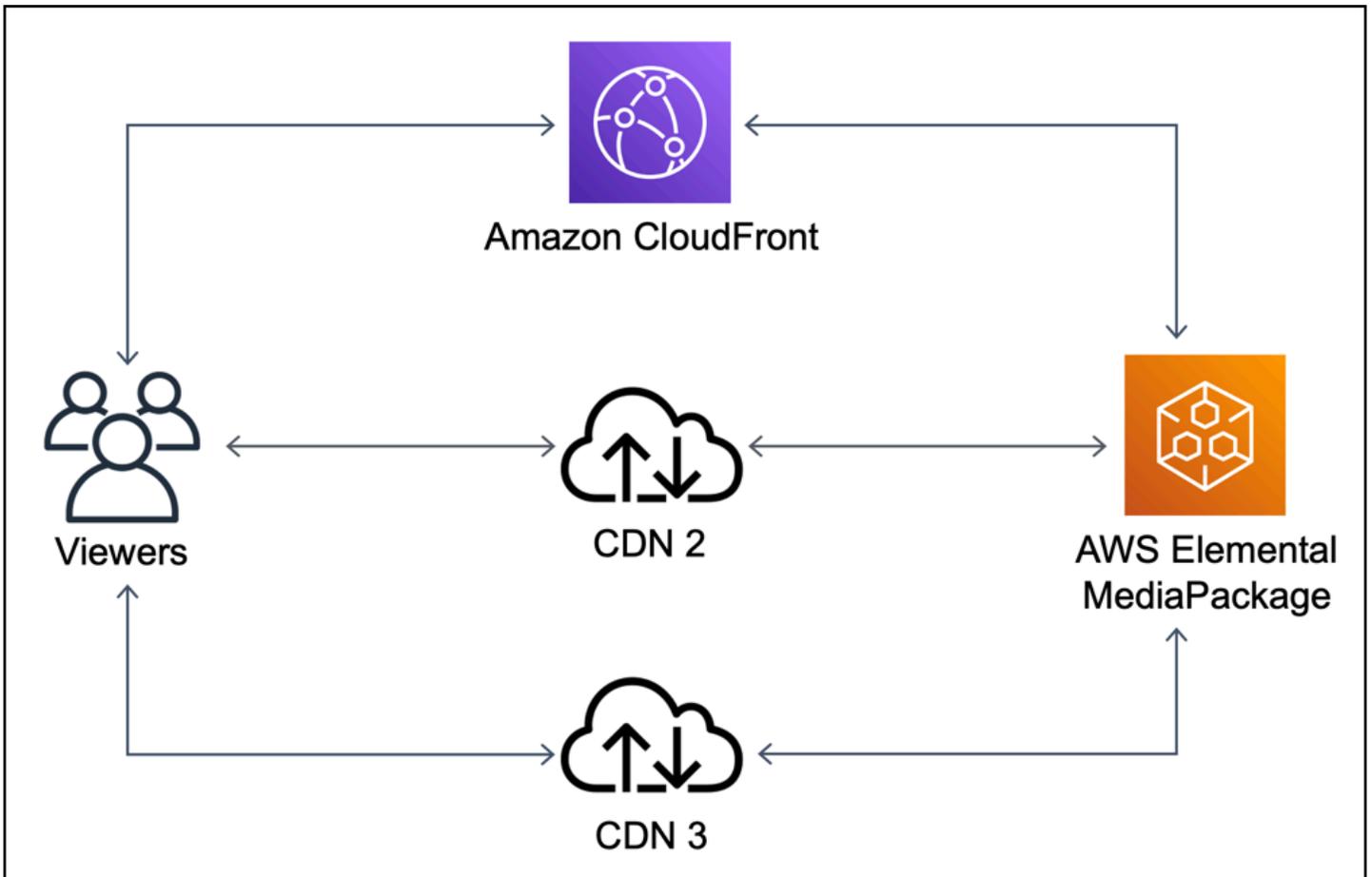
⚠ Important

マルチ CDN アーキテクチャでオリジンシールドを使用し、割引料金を利用する場合は、[お問い合わせ](#) または AWS 営業担当者を通じて詳細をご確認ください。追加料金が適用される場合があります。

以下の図は、複数の CDN で人気のあるライブビデオイベントを提供する場合に、この設定がオリジンへの負荷を最小限に抑える方法を示しています。次の図で、オリジンは AWS Elemental MediaPackage です。

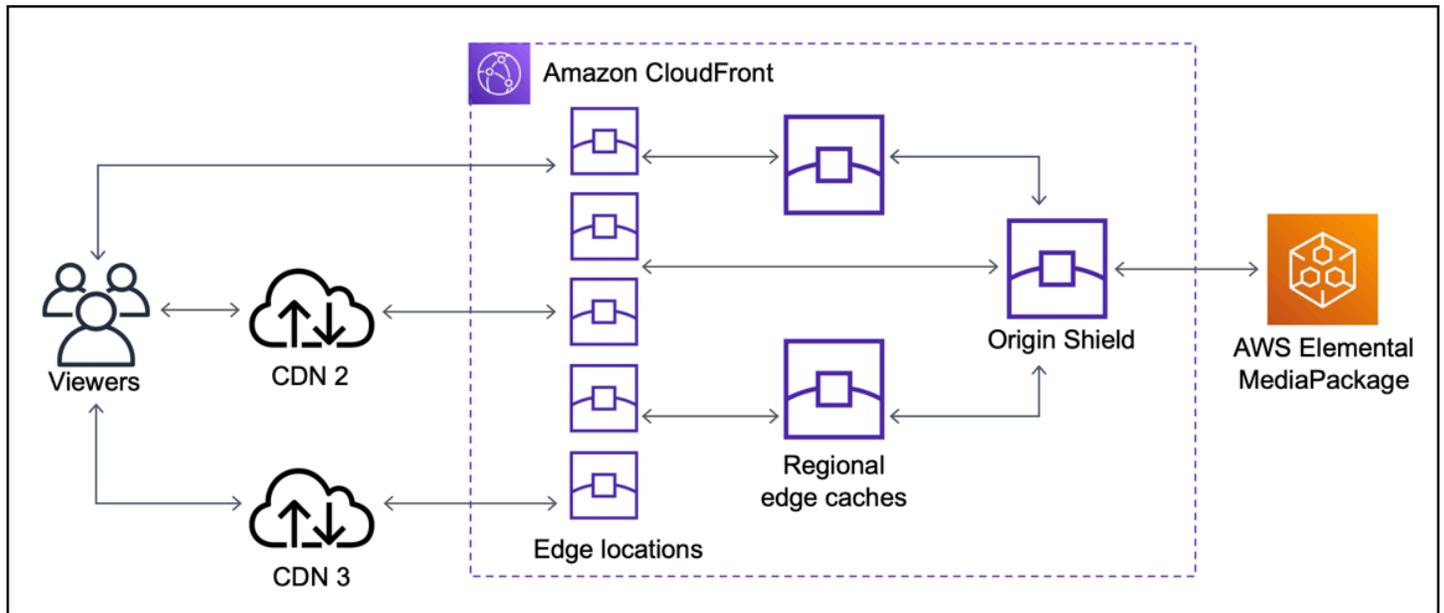
Origin Shield を使用しない場合 (複数の CDN を使用)

Origin Shield を使用しない場合は、次の図に示されているように、同じコンテンツに対する多数の重複リクエストを (それぞれ異なる CDN から) オリジンが受け取る可能性があります。



Origin Shield を使用する場合 (複数の CDN を使用)

Origin Shield を他の CDNsのオリジン CloudFront として使用すると、次の図に示すように、オリジンの負荷を軽減できます。



Origin Shield の AWS リージョンの選択

Amazon CloudFront は、[にリージョン別エッジキャッシュ](#) CloudFront があるAWSリージョンで Origin Shield を提供しています。Origin Shield を有効にするときは、Origin Shield の AWS リージョンを選択します。オリジンに対するレイテンシーが最も低い AWS リージョンを選択するようにしてください。Origin Shieldは、AWS リージョン内にあるオリジンにも、AWS 外のオリジンにも使用できます。

AWS リージョン内のオリジン

オリジンが AWSリージョンにある場合は、まず、オリジンが が次のリージョンで Origin Shield. CloudFront offers Origin Shield CloudFront を提供するAWSリージョンにあるかどうかを確認します。

- 米国東部 (オハイオ) – us-east-2
- 米国東部 (バージニア北部) – us-east-1
- 米国西部 (オレゴン) – us-west-2
- アジアパシフィック (ムンバイ) – ap-south-1
- アジアパシフィック (ソウル) – ap-northeast-2
- アジアパシフィック (シンガポール) – ap-southeast-1
- アジアパシフィック (シドニー) – ap-southeast-2
- アジアパシフィック (東京) – ap-northeast-1

- 欧州 (フランクフルト) – eu-central-1
- 欧州 (アイルランド) – eu-west-1
- 欧州 (ロンドン) – eu-west-2
- 南米 (サンパウロ) – sa-east-1

オリジンが Origin Shield CloudFront を提供する AWSリージョンにある場合

オリジンが Origin Shield CloudFront を提供する AWSリージョンにある場合 (上記のリストを参照)、オリジンと同じリージョンで Origin Shield を有効にします。

オリジンが CloudFront Origin Shield を提供する AWSリージョンにない場合

オリジンが Origin Shield CloudFront を提供する AWSリージョンにない場合は、次の表を参照して、Origin Shield を有効にするリージョンを決定します。

オリジンの場所	Origin Shield を有効にするリージョン
米国西部 (北カリフォルニア) – us-west-1	米国西部 (オレゴン) – us-west-2
アフリカ (ケープタウン) – af-south-1	欧州 (アイルランド) – eu-west-1
アジアパシフィック (香港) – ap-east-1	アジアパシフィック (シンガポール) – ap-southeast-1
カナダ (中部) – ca-central-1	米国東部 (バージニア北部) – us-east-1
欧州 (ミラノ) – eu-south-1	欧州 (フランクフルト) – eu-central-1
欧州 (パリ) – eu-west-3	欧州 (ロンドン) – eu-west-2
欧州 (ストックホルム) – eu-north-1	欧州 (ロンドン) – eu-west-2
中東 (バーレーン) – me-south-1	アジアパシフィック (ムンバイ) – ap-south-1

オリジンが 外にある場合AWS

Origin Shield は、オンプレミスのオリジン、または AWS リージョン外のオリジンにも使用できます。その場合は、オリジンに対するレイテンシーが最も低い AWS リージョンで Origin Shield を有

効にします。オリジンに対するレイテンシーが最も低い AWS リージョンがわからない場合は、以下の提案を参考にして判断することができます。

- 上記の表を参照し、オリジンの地理的位置に基づいて、オリジンに対するレイテンシーが最も低いと思われる AWS リージョンを見計らいます。
- オリジンに地理的に近いいくつかの異なる AWS リージョンで Amazon EC2 インスタンスを起動し、ping を使用してテストを数回実行して、これらのリージョンとオリジン間の典型的なネットワークレイテンシーを測定できます。

Origin Shield の有効化

Origin Shield を有効にすると、キャッシュヒット率の向上、オリジンへの負荷の軽減、パフォーマンスの強化を図ることができます。Origin Shield を有効にするには、CloudFront デистриビューションのオリジン設定を変更します。Origin Shield は、オリジンのプロパティです。CloudFront デистриビューション内のオリジンごとに、そのオリジンに最適なパフォーマンスを提供する AWS リージョンで Origin Shield を個別に有効にできます。

Origin Shield は AWS CloudFormation、CloudFront コンソール、または CloudFront API を使用して有効にできます。

Console

既存のオリジンに対して Origin Shield を有効にするには (コンソール)

1. にサインイン AWS Management Console し、 で CloudFront コンソールを開きます <https://console.aws.amazon.com/cloudfront/v4/home>。
2. 更新するオリジンがあるデистриビューションを選択します。
3. [Origin and Origin Groups (オリジンおよびオリジングループ)] タブを選択します。
4. 更新するオリジンを選択し、[編集] を選択します。
5. [Origin Shield を有効にする] で、[はい] を選択します。
6. [Origin Shield Region] (Origin Shield のリージョン) には、Origin Shield を有効にする AWS リージョンを選択します。リージョンの選択については、「[Origin Shield の AWS リージョンの選択](#)」を参照してください。
7. ページの最下部で [はい、編集します] を選択します。

ディストリビューションのステータスが [デプロイ済み] であれば、Origin Shield の準備が完了しています。これには数分かかります。

新しいオリジンの Origin Shield を有効にするには (コンソール)

1. にサインインAWS Management Consoleし、 で CloudFront コンソールを開きます<https://console.aws.amazon.com/cloudfront/v4/home>。
2. 既存のディストリビューションに新しいオリジンを作成するには、次の操作を行います。
 1. オリジンを作成するディストリビューションを選択します。
 2. [オリジンの作成] を選択し、ステップ 3 に進みます。

新しいディストリビューションに新しいオリジンを作成するには、次の操作を行います。

1. [Create Distribution] を選択します。
2. [ウェブ] セクションで、[今すぐ始める] を選択します。[オリジン設定] セクションで、次の操作をステップ 3 から行います。
3. [Origin Shield を有効にする] で、[はい] を選択します。
4. [Origin Shield Region] (Origin Shield のリージョン) には、Origin Shield を有効にする AWS リージョンを選択します。リージョンの選択については、「[Origin Shield の AWS リージョンの選択](#)」を参照してください。

新しいディストリビューションを作成する場合は、そのページの他の設定を使用して、ディストリビューションの設定を続けて行います。詳細については、「[ディストリビューションを作成または更新する場合に指定する値](#)」を参照してください。

5. [作成] (既存のディストリビューションに新しいオリジンを作成する場合) を選択するか、[ディストリビューションの作成] (新しいディストリビューションに新しいオリジンを作成する場合) を選択して、変更内容を保存します。

ディストリビューションのステータスが [デプロイ済み] であれば、Origin Shield の準備が完了しています。これには数分かかります。

AWS CloudFormation

AWS CloudFormation で Origin Shield を有効にするに

は、AWS::CloudFront::Distribution リソースの Origin プロパティタイプに OriginShield プロパティを使用します。OriginShield プロパティは、既存の Origin に追加することも、新しい Origin を作成するときに含めることができます。

次の例は、米国西部 (オレゴン) リージョン (OriginShield) で us-west-2 を有効にするための構文を YAML 形式で示しています。リージョンの選択については、「[the section called “Origin Shield の AWS リージョンの選択”](#)」を参照してください。この例では、Origin リソース全体ではなく、AWS::CloudFront::Distribution プロパティタイプのみを示しています。

```
Origins:
- DomainName: 3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com
  Id: Example-EMP-3ae97e9482b0d011
  OriginShield:
    Enabled: true
    OriginShieldRegion: us-west-2
  CustomOriginConfig:
    OriginProtocolPolicy: match-viewer
    OriginSSLProtocols: TLSv1
```

詳細については、「AWS CloudFormationユーザーガイド」の「リソースとプロパティのリファレンス」セクションの[AWS::CloudFront::Distribution 「オリジン」](#)を参照してください。

API

AWS SDKs または AWS Command Line Interface (AWS CLI) を使用して CloudFront API で Origin Shield を有効にするには、OriginShieldタイプを使用します。OriginShield は、Origin の DistributionConfig 内で指定します。OriginShield タイプの詳細については、「Amazon CloudFront API リファレンス」の次の情報を参照してください。

- [OriginShield](#) (タイプ)
- [Origin](#) (タイプ)
- [DistributionConfig](#) (タイプ)
- [UpdateDistribution](#) (オペレーション)
- [CreateDistribution](#) (オペレーション)

これらのタイプと操作を使用するための具体的な構文は、使用する SDK、CLI、API クライアントによって異なります。詳細については、SDK、CLI、またはクライアントのリファレンスドキュメントを参照してください。

Origin Shield のコストの見積もり

Origin Shield の料金は、増分レイヤーとして Origin Shield に送信されるリクエストの数に基づいて計算されます。

オリジンにプロキシ化される動的 (キャッシュ不可能な) リクエストの場合、Origin Shield は常に増分レイヤーとなります。動的リクエストでは、HTTP メソッド PUT、POST、PATCH、および DELETE を使用します。

動的リクエストに対する Origin Shield の料金を見積もるには、次の式を使用します。

動的リクエストの総数 x 10,000 リクエストあたりの Origin Shield 料金 / 10,000

キャッシュ可能なリクエスト (HTTP メソッド GET、HEAD、OPTIONS) の場合、Origin Shield は増分レイヤーになることがあります。Origin Shield を有効にするときは、Origin Shield の AWS リージョンを選択します。Origin Shield と同じリージョン内の [リージョン別エッジキャッシュ](#) に本来送信されるリクエストの場合、Origin Shield は増分レイヤーになりません。このようなリクエストに対して Origin Shield の料金は発生しません。Origin Shield とは異なるリージョンのリージョン別エッジキャッシュに送信されてから Origin Shield に送信されるリクエストの場合、Origin Shield は増分レイヤーになります。このようなリクエストに対しては、Origin Shield の料金が発生します。

キャッシュ可能なリクエストに対する Origin Shield の料金を見積もるには、次の式を使用します。

キャッシュ可能なリクエストの総数 x (1 - キャッシュヒット率) x 異なるリージョンのリージョン別エッジキャッシュから Origin Shield に送信されるリクエストの割合 x 10,000 リクエストあたりの Origin Shield 料金 / 10,000

Origin Shield の 10,000 リクエストあたりの料金の詳細については、[CloudFront 「の料金」](#) を参照してください。

Origin Shield の高可用性

Origin Shield は、Amazon CloudFrontの [リージョン別エッジキャッシュ](#) を利用します。これらのエッジキャッシュはそれぞれ、少なくとも3つの [アベイラビリティゾーン](#) と Amazon EC2 Auto Scaling インスタンスのフリートを使用して、AWS リージョン内に構築されます。CloudFront 口ケーションから Origin Shield への接続では、リクエストごとにアクティブなエラー追跡も使用され、プライマリ Origin Shield 口ケーションが使用できない場合は、セカンダリ Origin Shield 口ケーションにリクエストを自動的にルーティングします。

Origin Shield が他の CloudFront 機能とやり取りする方法

以下のセクションでは、Origin Shield が他の CloudFront 機能とやり取りする方法について説明します。

Origin Shield と CloudFront ログ記録

Origin Shield がいつリクエストを処理したかを確認するには、以下のいずれかを有効にする必要があります。

- [CloudFront 標準ログ \(アクセスログ\)](#)。標準ログは無料で提供されます。
- [CloudFront リアルタイムログ](#)。リアルタイムログの使用には追加料金が発生します。「[Amazon CloudFront 料金表](#)」を参照してください。

Origin Shield からのキャッシュヒットは、CloudFront ログ `OriginShieldHit` の `x-edge-detailed-result-type` フィールドに として表示されます。Origin Shield は、Amazon CloudFront の [リージョン別エッジキャッシュを活用します](#)。リクエストが CloudFront エッジロケーションから Origin Shield として動作するリージョン別エッジキャッシュにルーティングされた場合、リクエストは としてではなく、 としてログ Hit に として報告されます `OriginShieldHit`。

Origin Shield とオリジングループ

Origin Shield は [CloudFront オリジングループ](#) と互換性があります。Origin Shield はオリジンのプロパティであるため、オリジンがオリジングループの一部であっても、リクエストは常に各オリジンの Origin Shield を通過します。特定のリクエストについて、はプライマリオリジンの Origin Shield を介してオリジングループ内のプライマリオリジンにリクエストを CloudFront ルーティングします。そのリクエストが失敗した場合 (オリジングループのフェイルオーバー基準に基づく)、はセカンダリオリジンの Origin Shield を介してセカンダリオリジンにリクエストを CloudFront ルーティングします。

Origin Shield と Lambda@Edge

Origin Shield は [Lambda@Edge](#) 関数の機能性には影響しませんが、これらの関数が実行される AWS リージョンに影響を及ぼす可能性があります。Lambda@Edge と共に Origin Shield を使用する場合、[オリジン向けのトリガー](#) (オリジンリクエストとオリジンレスポンス) は、Origin Shield が有効になっている AWS リージョンで実行されます。ビューワー向けトリガーは影響を受けません。

CloudFront オリジンフェイルオーバーによる高可用性の最適化

高可用性を必要とするシナリオでは、オリジンフェイルオーバー CloudFront を使用して を設定できます。開始するには、プライマリとセカンダリの 2 つのオリジンを持つオリジングループを作成します。プライマリオリジンが使用できない場合、または障害を示す特定の HTTP レスポンスステータスコードを返す場合、CloudFront はセカンダリオリジンに自動的に切り替わります。

オリジンフェイルオーバーを設定するには、少なくとも 2 つのオリジンを持つディストリビューションが必要です。次に、1 つをプライマリとして設定した 2 つのオリジンを含むディストリビューションのオリジングループを作成します。最後に、オリジングループを使用するようにキャッシュ動作を作成または更新します。

オリジングループを設定して特定のオリジンフェイルオーバーオプションを設定する手順については、「[オリジングループの作成](#)」を参照してください。

キャッシュ動作のオリジンフェイルオーバーを設定すると、 はビューワーリクエストに対して次の CloudFront 操作を行います。

- キャッシュヒットが発生すると、 はリクエストされたオブジェクト CloudFront を返します。
- キャッシュミスがある場合、 はオリジングループ内のプライマリオリジンにリクエストを CloudFront ルーティングします。
- プライマリオリジンが HTTP 2xx や 3xx ステータスコードなど、フェイルオーバー用に設定されていないステータスコードを返すと、 はリクエストされたオブジェクトをビューワーに CloudFront 保存します。
- 次のいずれかに該当する場合:
 - プライマリオリジンは、フェイルオーバー用に設定した HTTP ステータスコードを返す
 - CloudFront がプライマリオリジンへの接続に失敗する
 - プライマリオリジンからの応答に時間がかかりすぎる (タイムアウト)

次に、 はオリジングループのセカンダリオリジンにリクエストを CloudFront ルーティングします。

Note

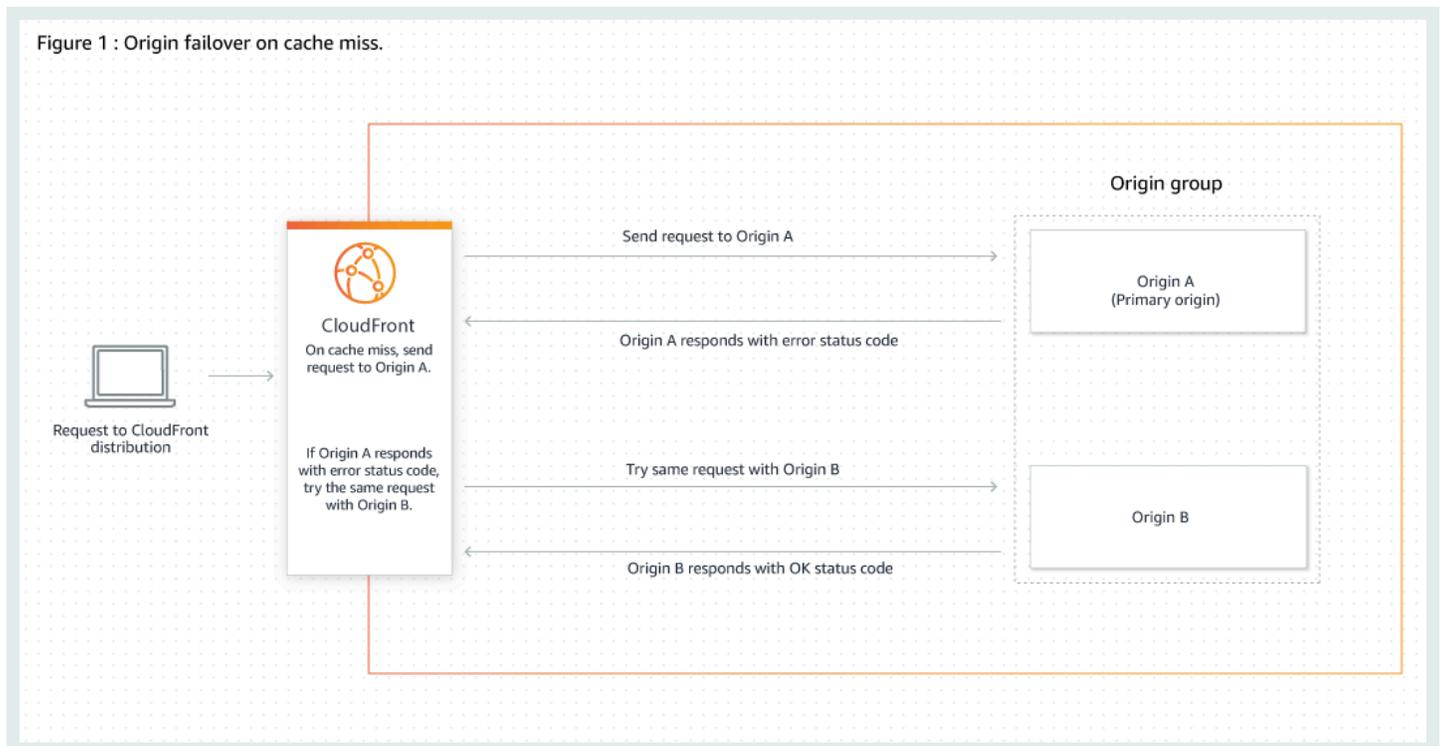
ビデオコンテンツのストリーミングなど、ユースケースによっては、セカンダリオリジンにすばやくフェイルオーバー CloudFront したい場合があります。がセカンダリオリジンに

CloudFront フェイルオーバーする速度を調整するには、「」を参照してください[オリジンのタイムアウトと試行の制御](#)。

CloudFront は、前のリクエストがセカンダリオリジンにフェイルオーバーされた場合でも、すべての受信リクエストをプライマリオリジンにルーティングします。CloudFront は、プライマリオリジンへのリクエストが失敗した場合にのみ、セカンダリオリジンにリクエストを送信します。

CloudFront ビューワーリクエストの HTTP メソッドが GET、HEAD、または の場合にのみ、はセカンダリオリジンにフェイルオーバーしますOPTIONS。ビューワーが別の HTTP メソッド (、 などPUT) を送信しても POST. CloudFront does はフェイルオーバーしません。

以下の図は、オリジンフェイルオーバーのしくみを示しています。



トピック

- [オリジングループの作成](#)
- [オリジンのタイムアウトと試行の制御](#)
- [Lambda@Edge 関数でのオリジンフェイルオーバーの使用](#)
- [オリジンフェイルオーバーでのカスタムエラーページの使用](#)

オリジングループの作成

オリジングループを作成するには

1. にサインインAWS Management Consoleし、 で CloudFront コンソールを開きます<https://console.aws.amazon.com/cloudfront/v4/home>。
2. オリジングループを作成するディストリビューションを選択します。
3. [オリジン] タブを選択します。
4. ディストリビューションに複数のオリジンがあることを確認します。そうでない場合は、2 番目のオリジンを追加します。
5. [Origin groups] (オリジンのグループ) ペインの [Origins] (オリジン) タブで、[Create Origin group] (オリジングループの作成) を選択します。
6. オリジングループのオリジンを選択します。オリジンを追加したら、矢印を使用して優先度 (つまり、どのオリジンがプライマリで、どのオリジンがセカンダリであるか) を設定します。
7. オリジングループの名前を入力します。
8. フェイルオーバー基準として使用する HTTP ステータスコードを選択します。次のステータスコードを任意に組み合わせて選択できます。400、403、404、416、500、502、503、または 504。が指定したステータスコードのいずれかを含むレスポンス CloudFront を受信すると、セカンダリオリジンにフェイルオーバーします。

Note

CloudFront ビューワーリクエストの HTTP メソッドが GET、HEAD、または の場合にのみ、 はセカンダリオリジンにフェイルオーバーしますOPTIONS。ビューワーが別の HTTP メソッド (、 などPUT) を送信しても POST. CloudFront does はフェイルオーバーしません。

9. [Create origin group] (オリジングループの作成) を選択します。

ディストリビューションのオリジングループを指定する方法については、「[名前](#)」を参照してください。

オリジンのタイムアウトと試行の制御

デフォルトでは、 はセカンダリオリジンにフェイルオーバーする前に、オリジングループのプライマリオリジンへの接続 CloudFront を 30 秒間試行します (それぞれ 10 秒間の 3 回の接続試行)。ビ

デオコンテンツのストリーミングなど、ユースケースによっては、セカンダリオリジンにフェイルオーバーをより迅速に CloudFront したい場合があります。次の設定を調整して、セカンダリオリジンへの CloudFront フェイルオーバーの速度に影響を与えることができます。オリジンがセカンダリオリジン、またはオリジングループの一部ではないオリジンである場合、これらの設定はビューワーに HTTP 504 レスポンスを CloudFront 返す速度に影響します。

よりすばやくフェイルオーバーするには、接続タイムアウトを短くするか、接続試行回数を減らすか、またはその両方を行います。カスタムオリジン (静的ウェブサイトホスティングで設定されている Amazon S3 バケットオリジンを含む) の場合、オリジン応答タイムアウトを調整することもできます。

オリジン接続タイムアウト

オリジン接続タイムアウト設定は、オリジンへの接続を確立しようとしたときに CloudFront 待機する時間に影響します。デフォルトでは、は接続を確立するまで 10 秒 CloudFront 待機しますが、1~10 秒 (両端を含む) を指定できます。詳細については、「[接続タイムアウト](#)」を参照してください。

オリジン接続の試行

オリジン接続の試行回数の設定は、がオリジンへの接続を CloudFront 試行する回数に影響します。デフォルトでは、は接続を 3 回 CloudFront 試行しますが、1~3 (両端を含む) を指定できます。詳細については、「[接続の試行](#)」を参照してください。

カスタムオリジン (静的ウェブサイトホスティングで設定された Amazon S3 バケットを含む) の場合、この設定は、オリジン応答タイムアウトの場合にがオリジンから応答を取得 CloudFront しようとする回数にも影響します。

オリジン応答タイムアウト

Note

これは、カスタムオリジンにのみ適用されます。

オリジン応答タイムアウト設定は、がオリジンから応答を受信する (または完全な応答を受信する) までの CloudFront 待機時間に影響します。デフォルトでは、は 30 秒間 CloudFront 待機しますが、1~60 秒 (両端を含む) を指定できます。詳細については、「[応答タイムアウト \(カスタムオリジンのみ\)](#)」を参照してください。

これらの設定を変更する方法

[CloudFront コンソール](#)でこれらの設定を変更するには

- 新しいオリジンまたは新しいディストリビューションの場合、リソースの作成時にこれらの値を指定します。
- 既存のディストリビューションの既存のオリジンについては、オリジンを編集するときこれらの値を指定します。

詳細については、「[ディストリビューションを作成または更新する場合に指定する値](#)」を参照してください。

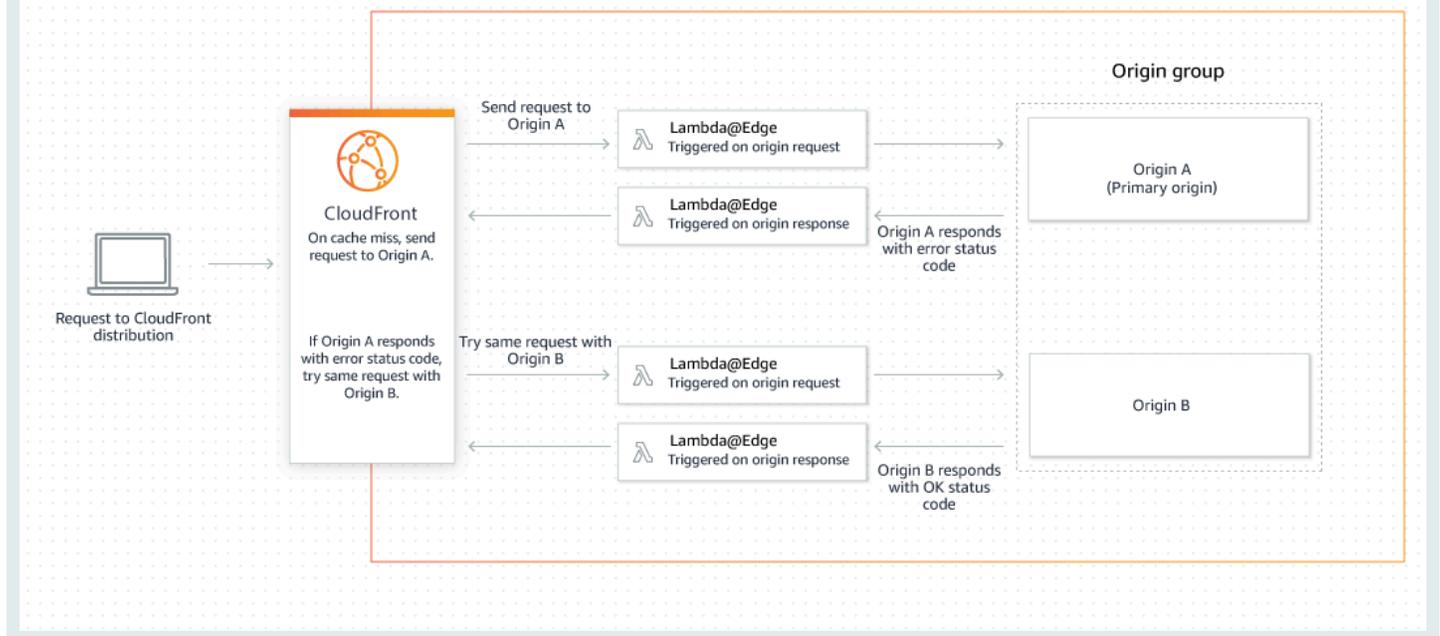
Lambda@Edge 関数でのオリジンフェイルオーバーの使用

Lambda@Edge 関数は、オリジングループで設定した CloudFront ディストリビューションで使用できます。Lambda 関数を使用するには、キャッシュ動作を作成するときにオリジングループの[オリジンリクエストまたはオリジンレスポンストリガー](#)で指定します。オリジングループで Lambda@Edge 関数を使用すると、1つのビューワーリクエストに対して、この関数を2回トリガーできます。たとえば、次のシナリオが考えられます。

1. オリジンリクエストトリガーを使用して Lambda@Edge 関数を作成します。
2. Lambda 関数は、プライマリオリジンにリクエスト CloudFront を送信すると (キャッシュミス時に) 1回トリガーされます。
3. プライマリオリジンは、フェイルオーバー用に設定された HTTP ステータスコードで応答します。
4. Lambda 関数は、セカンダリオリジンに同じリクエスト CloudFront を送信すると再びトリガーされます。

次の図は、オリジンリクエストまたはレスポンストリガーに Lambda@Edge 関数を含める場合、オリジンフェイルオーバーがどのように機能するかを示しています。

Figure 2 : Origin failover with Lambda@Edge functions triggered on origin request and response events.



Lambda@Edge トリガーの使用の詳細については、「[the section called “トリガーの追加”](#)」を参照してください。

オリジンフェイルオーバーでのカスタムエラーページの使用

オリジンフェイルオーバー用に設定されていないオリジンでそれらを使用する方法と同様に、オリジングループでカスタムエラーページを使用できます。

オリジンフェイルオーバーを使用する場合、プライマリオリジンまたはセカンダリオリジン (またはその両方) のカスタムエラーページを返す CloudFront ように を設定できます。

- プライマリオリジンのカスタムエラーページを返す – プライマリオリジンがフェイルオーバー用に設定されていない HTTP ステータスコードを返した場合、 はカスタムエラーページをビューワーに CloudFront 返します。
- セカンダリオリジンのカスタムエラーページを返 CloudFront す – がセカンダリオリジンから失敗ステータスコード CloudFront を受信した場合、 はカスタムエラーページを返します。

でカスタムエラーページを使用する方法の詳細については CloudFront、「[」を参照してください](#)[カスタムエラーレスポンスの生成](#)。

コンテンツがキャッシュに保持される期間 (有効期限) の管理

が別のリクエストをオリジン CloudFront に転送するまでのファイルを CloudFront キャッシュに保持する期間を制御できます。この期間を短くすると、動的なコンテンツを供給できます。この期間を長くすると、ユーザー側のパフォーマンスは向上します。ファイルがエッジキャッシュから直接返される可能性が高くなるためです。期間を長くすると、オリジンの負荷も軽減されます。

通常、は、指定したキャッシュ期間が経過するまで、つまりファイルの有効期限が切れるまで、エッジロケーションからファイル CloudFront を保存します。有効期限が切れると、エッジロケーションが次にファイルのリクエストを受け取ったときに、キャッシュにファイルの最新バージョンが含まれていることを確認するリクエストをオリジン CloudFront に転送します。オリジンからのレスポンスは、ファイルが変更されたかどうかによって異なります。

- CloudFront キャッシュに最新バージョンがすでに存在する場合、オリジンはステータスコードを返します 304 Not Modified。
- CloudFront キャッシュに最新バージョンがない場合、オリジンはステータスコード 200 OK とファイルの最新バージョンを返します。

エッジロケーションのファイルが頻繁にリクエストされない場合は、有効期限が切れる前にファイルを削除して、最近リクエストされたファイル用のスペースを確保 CloudFront できます。

デフォルトでは、各ファイルは 24 時間後に自動的に有効期限切れになりますが、2 つの方法でこのデフォルトの動作を変更できます。

- 同じパスパターンに一致するすべてのファイルのキャッシュ期間を変更するには、キャッシュ動作の最小 TTL、最大 TTL、およびデフォルト TTL CloudFront の設定を変更できます。個々の設定については、「[the section called “指定する値”](#)」の「[最小 TTL](#)」、「[最大 TTL](#)」、「[デフォルト TTL](#)」を参照してください。
- 個々のファイルのキャッシュ保持期間を変更するには、ファイルに Cache-Control または max-age ディレクティブが付いた s-maxage を追加するか、Expires ヘッダーフィールドを追加するようにオリジンを設定します。詳細については、「[ヘッダーを使用した個々のオブジェクトのキャッシュ保持期間の制御](#)」を参照してください。

[Minimum TTL (最小 TTL)]、[Default TTL (デフォルト TTL)]、[Maximum TTL (最大 TTL)] が max-age ディレクティブ、s-maxage ディレクティブ、Expires ヘッダーフィールドとどのように連動するかの詳細については、「[the section called “がオブジェクトを CloudFront キャッシュする時間を指定する”](#)」を参照してください。

また、オリジンに別のリクエストを転送して、リクエストされたオブジェクトの取得を CloudFront 再試行するまでのエラー (など404 Not Found) のCloudFront キャッシュへの保持期間を制御することもできます。詳細については、「[the section called “がオリジンからの HTTP 4xx および 5xx ステータスコード CloudFront を処理してキャッシュする方法”](#)」を参照してください。

トピック

- [ヘッダーを使用した個々のオブジェクトのキャッシュ保持期間の制御](#)
- [古い \(期限切れの\) コンテンツの提供](#)
- [がオブジェクトを CloudFront キャッシュする時間を指定する](#)
- [Amazon S3 コンソールを使用したオブジェクトへのヘッダーの追加](#)

ヘッダーを使用した個々のオブジェクトのキャッシュ保持期間の制御

Cache-Control および Expires ヘッダーを使用して、オブジェクトをキャッシュに保持する期間を制御できます。[Minimum TTL]、[Default TTL]、[Maximum TTL] の設定もキャッシュ保持期間に影響を与えますが、ここでは、ヘッダーがキャッシュ保持期間に与える影響について概要を示します。

- Cache-Control max-age ディレクティブを使用すると、 がオリジンサーバーからオブジェクトを再度 CloudFront 取得するまでに、オブジェクトをキャッシュに保持する期間 (秒単位) を指定できます。が CloudFront サポートする最小有効期限は 0 秒です。最大値は 100 (年) です。値は次の形式で指定します。

```
Cache-Control: max-age=#
```

例えば、次のディレクティブは、関連付けられたオブジェクトを 3600 秒 (1 時間) キャッシュに保持 CloudFront するように に指示します。

```
Cache-Control: max-age=3600
```

オブジェクトをブラウザキャッシュに保持する期間とは異なる期間 CloudFront エッジキャッシュに保持する場合は、Cache-Control max-age および Cache-Control s-maxage ディレクティブを一緒に使用できます。詳細については、「[がオブジェクトを CloudFront キャッシュする時間を指定する](#)」を参照してください。

- Expires ヘッダーフィールドでは、「[RFC 2616、ハイパーテキスト転送プロトコル — HTTP/1.1 セクション 3.3.1、完全な日付](#)」に規定された形式を使用して、有効期限切れ日時を指定できます。

```
Sat, 27 Jun 2015 23:59:59 GMT
```

オブジェクトのキャッシュを制御するには、Cache-Control max-age ヘッダーフィールドではなく、Expires ディレクティブを使用することをお勧めします。Cache-Control max-age と Expires の両方の値を指定した場合、CloudFront は Cache-Control max-age の値のみを使用します。

詳細については、「[ガオブジェクトを CloudFront キャッシュする時間を指定する](#)」を参照してください。

ビューワーからの GET リクエストで HTTP Cache-Control または Pragma ヘッダーフィールドを使用して、オブジェクトのオリジンサーバー CloudFront に強制的に戻ることはできません。は、ビューワーリクエストのこれらのヘッダーフィールド CloudFront を無視します。

Cache-Control および Expires ヘッダーフィールドの詳細については、「RFC 2616、ハイパーテキスト転送プロトコル — HTTP/1.1」の以下のセクションを参照してください。

- [セクション 14.9 キャッシュ制御](#)
- [セクション 14.21 期限](#)

古い (期限切れの) コンテンツの提供

CloudFront は、Stale-While-Revalidate および Stale-If-Error キャッシュ制御ディレクティブをサポートします。

- stale-while-revalidate ディレクティブにより、CloudFront はキャッシュから古いコンテンツを供給できますが、オリジンから新しいバージョンを非同期的に取得します。これにより、ユーザーはバックグラウンドフェッチを待たずに CloudFront のエッジロケーションからすぐにレスポンスを受信し、今後のリクエストに備えて新しいコンテンツがバックグラウンドでロードされるため、レイテンシーが向上します。

次の例では、はレスポンスを 1 時間 CloudFront キャッシュします (max-age=3600)。この期間以降にリクエストが行われた場合、は古いコンテンツ CloudFront を提供し、同時にキャッシュされたコンテンツを再検証して更新するリクエストをオリジンに送信します。コンテンツが再検証される間、古いコンテンツは最大 10 分間 (stale-while-revalidate=600) 提供されます。

```
Cache-Control: max-age=3600, stale-while-revalidate=600
```

- stale-if-error ディレクティブは CloudFront、オリジンに到達できない場合、または 500 ~ 600 のエラーコードを返す場合、がキャッシュから古いコンテンツを配信できるようにします。

これにより、ビューワーはオリジンが停止しているときでもコンテンツにアクセスできるようになります。

次の例では、はレスポンスを 1 時間 CloudFront キャッシュします (max-age=3600)。オリジンがダウンしている場合、またはこの期間が経過した後にエラーを返した場合、CloudFront は、最大 24 時間 () 古いコンテンツを引き続き提供しますstale-if-error=86400。

```
Cache-Control: max-age=3600, stale-if-error=86400
```

Note

stale-if-error と の両方の[カスタムエラーレスポンス](#)が設定されている場合、CloudFront は、指定されたstale-if-error期間内にエラーが発生した場合に、古いコンテンツを提供しようとします。古いコンテンツが使用できない場合、またはコンテンツが stale-if-error 期間を超えている場合、は対応するエラーステータスコードに設定されたカスタムエラーレスポンス CloudFront を保存します。

両方一緒に使用する

stale-while-revalidate および stale-if-error は独立したキャッシュ制御ディレクティブで、これらを一緒に使用することでレイテンシーを減らしたり、オリジンが応答または回復するためのバッファを追加したりできます。

次の例では、はレスポンスを 1 時間 CloudFront キャッシュします (max-age=3600)。この期間を過ぎてリクエストが行われた場合、は、コンテンツが再検証されている間、最大 10 分間 (stale-while-revalidate=600) 古いコンテンツCloudFront を提供します。がコンテンツを再検証しようとしたときにオリジンサーバーがエラーを返 CloudFront した場合、CloudFront は最大 24 時間 () 古いコンテンツを引き続き提供しますstale-if-error=86400。

```
Cache-Control: max-age=3600, stale-while-revalidate=600, stale-if-error=86400
```

Tip

キャッシュによって、パフォーマンスと鮮度が保たれます。stale-while-revalidate や stale-if-error などのディレクティブを使用すると、パフォーマンスとユーザーエクスペリエンスが向上しますが、コンテンツをどれだけ新鮮にするかの希望に合った設定に

してください。古いコンテンツディレクティブは、コンテンツを更新する必要があるが、最新バージョンであることが重要でない場合に最適です。さらに、コンテンツが変更されないか、ほとんど変更されない場合、stale-while-revalidate は不要なネットワークリクエストを追加する可能性があります。代わりに、キャッシュ期間を長く設定することを検討してください。

がオブジェクトを CloudFront キャッシュする時間を指定する

オリジンに別のリクエストを送信する前に、がオブジェクトをキャッシュに CloudFront 保持する時間を制御するには、次の操作を行います。

- CloudFront デイストリビューションのキャッシュ動作で、最小、最大、およびデフォルトの TTL 値を設定します。これらの値は、キャッシュ動作にアタッチされた[キャッシュポリシー](#) (推奨) の中、またはレガシーキャッシュ設定の中で設定できます。
- オリジンからの応答に Cache-Control または Expires ヘッダーを含めます。これらのヘッダーは、別のリクエストを に送信する前に、ブラウザがオブジェクトをブラウザキャッシュに保持する期間を決定するのにも役立ちます CloudFront。

次の表では、オリジンから送信された Cache-Control ヘッダーと Expires ヘッダーがキャッシュ動作の TTL 設定とどのように関係し、キャッシュに影響を与えるのかを説明しています。

オリジンヘッダー	最小 TTL = 0	最小 TTL > 0
オリジンが Cache-Control: max-age ディレクティブをオブジェクトに追加	<p>CloudFront キャッシュ</p> <p>CloudFront は、Cache-Control: max-age ディレクティブの値または CloudFront 最大 TTL の値のうち、小さい方の値に対応する期間、オブジェクトをキャッシュに保持します。</p> <p>ブラウザキャッシュ</p>	<p>CloudFront キャッシュ</p> <p>CloudFront キャッシュは、CloudFront 最小 TTL、最大 TTL、および Cache-Control max-age ディレクティブの値によって異なります。</p> <ul style="list-style-type: none"> • 最小 TTL < max-age < 最大 TTL の場合、は Cache-Control: max-age

オリジンヘッダー	最小 TTL = 0	最小 TTL > 0
	<p>ブラウザは、Cache-Control: max-age ディレクティブの値に対応する期間、オブジェクトをキャッシュに保持します。</p>	<p>ディレクティブの値に対応する期間、オブジェクトを CloudFront キャッシュします。</p> <ul style="list-style-type: none">• max-age < 最小 TTL の場合 CloudFront、は CloudFront 最小 TTL の値に対応する期間、オブジェクトをキャッシュします。• max-age > 最大 TTL の場合 CloudFront、は CloudFront 最大 TTL の値に対してオブジェクトをキャッシュします。 <p>ブラウザキャッシュ</p> <p>ブラウザは、Cache-Control: max-age ディレクティブの値に対応する期間、オブジェクトをキャッシュに保持します。</p>

オリジンヘッダー	最小 TTL = 0	最小 TTL > 0
<p>オリジンが Cache-Control: max-age ディレクティブをオブジェクトに追加しない</p>	<p>CloudFront キャッシュ</p> <p>CloudFront は、CloudFront デフォルト TTL の値に対してオブジェクトをキャッシュします。</p> <p>ブラウザキャッシュ</p> <p>ブラウザによって異なります。</p>	<p>CloudFront キャッシュ</p> <p>CloudFront は、CloudFront 最小 TTL またはデフォルト TTL の値のうち大きい方の値に対応する期間、オブジェクトをキャッシュに保持します。</p> <p>ブラウザキャッシュ</p> <p>ブラウザによって異なります。</p>

オリジンヘッダー	最小 TTL = 0	最小 TTL > 0
<p>オリジンが Cache-Control: max-age および Cache-Control: s-maxage デイレクティブをオブジェクトに追加</p>	<p>CloudFront キャッシュ</p> <p>CloudFront は、Cache-Control: s-maxage デイレクティブの値または CloudFront 最大 TTL の値のうち、小さい方の値に対応する期間、オブジェクトをキャッシュに保持します。</p> <p>ブラウザキャッシュ</p> <p>ブラウザは、Cache-Control max-age デイレクティブの値に対応する期間、オブジェクトをキャッシュに保持します。</p>	<p>CloudFront キャッシュ</p> <p>CloudFront キャッシュは、CloudFront 最小 TTL、最大 TTL、および Cache-Control: s-maxage デイレクティブの値によって異なります。</p> <ul style="list-style-type: none"> • 最小 TTL < s-maxage < 最大 TTL の場合、は Cache-Control: s-maxage デイレクティブの値に対してオブジェクトを CloudFront キャッシュします。 • s-maxage < 最小 TTL の場合 CloudFront、は CloudFront 最小 TTL の値に対応する期間、オブジェクトをキャッシュします。 • s-maxage > 最大 TTL の場合 CloudFront、は CloudFront 最大 TTL の値に対してオブジェクトをキャッシュします。 <p>ブラウザキャッシュ</p> <p>ブラウザは、Cache-Control: max-age デイレク</p>

オリジンヘッダー	最小 TTL = 0	最小 TTL > 0
		タイプの値に対応する期間、オブジェクトをキャッシュに保持します。

オリジンヘッダー	最小 TTL = 0	最小 TTL > 0
<p>オリジンが Expires ヘッダーをオブジェクトに追加</p>	<p>CloudFront キャッシュ</p> <p>CloudFront は、Expires ヘッダーまたは CloudFront 最大 TTL の値のいずれか早い方の日付までオブジェクトをキャッシュに保持します。</p> <p>ブラウザキャッシュ</p> <p>ブラウザは、Expires ヘッダーにある日付まで、オブジェクトをキャッシュに保持します。</p>	<p>CloudFront キャッシュ</p> <p>CloudFront キャッシュは、CloudFront 最小 TTL、最大 TTL、ヘッダーの値によって異なります Expires。</p> <ul style="list-style-type: none"> • 最小 TTL < Expires < 最大 TTL の場合、は Expires ヘッダーの日付と時刻までオブジェクトを CloudFront キャッシュします。 • Expires < 最小 TTL の場合 CloudFront、は CloudFront 最小 TTL の値に対応する期間、オブジェクトをキャッシュします。 • Expires > 最大 TTL の場合 CloudFront、は CloudFront 最大 TTL の値に対してオブジェクトをキャッシュします。 <p>ブラウザキャッシュ</p> <p>ブラウザは、Expires ヘッダーの日付けおよび時刻まで、オブジェクトをキャッシュに保持します。</p>

オリジンヘッダー	最小 TTL = 0	最小 TTL > 0
<p>オリジンが Cache-Control: no-cache、no-store、および (または) private ディレクティブをオブジェクトに追加</p>	<p>CloudFront および ブラウザはヘッダーを優先します。</p>	<p>CloudFront キャッシュ</p> <p>CloudFront は、CloudFront 最小 TTL の値に対応する期間、オブジェクトをキャッシュに保持します。この表の下にある注意点をお読みください。</p> <p>ブラウザキャッシュ</p> <p>ブラウザはヘッダーを優先します。</p>

Warning

が Cache-Control: no-cache、no-store および/または private ディレクティブを含むオブジェクトをオリジンから CloudFront 取得し、後で同じオブジェクトに対する別のビューワーリクエスト CloudFront を取得した場合、CloudFront はビューワーリクエストを満たすためにオリジンに接続しようとします。

オリジンに到達可能な場合、はオリジンからオブジェクト CloudFront を取得し、ビューワーに返します。

オリジンに到達できず、最小または最大の TTL 値が 0 より大きい場合、CloudFront は以前にオリジンから取得したオブジェクトを提供します。

この動作を回避するには、Cache-Control: stale-if-error=0 ディレクティブに、オリジンから返されたオブジェクトを含めます。これにより CloudFront、オリジンから以前に取得したオブジェクトを返すのではなく、オリジンに到達できない場合に、は今後のリクエストに応じてエラーを返します。

CloudFront コンソールを使用してディストリビューションの設定を変更する方法については、「」を参照してください[ディストリビューションの更新](#)。CloudFront API を使用してディストリビューションの設定を変更する方法については、「」を参照してください[UpdateDistribution](#)。

Amazon S3 コンソールを使用したオブジェクトへのヘッダーの追加

Amazon S3 コンソールを使用して Amazon S3 オブジェクトに **Cache-Control** または **Expires** ヘッダーフィールドを追加するには

1. AWS Management Console にサインインし、Amazon S3 コンソール (<https://console.aws.amazon.com/s3/>) を開きます。
2. バケットの一覧で、ヘッダーを追加するファイルを含むバケットの名前を選択します。
3. ヘッダーを追加するファイルまたはフォルダの名前の横にあるチェックボックスをオンにします。フォルダにヘッダーを追加すると、そのフォルダ内のすべてのファイルに影響します。
4. [Actions] (アクション) を選択し、[Edit metadata] (メタデータの編集) を選択します。
5. [Add metadata] (メタデータを追加) パネルで、次の操作を行います。
 - a. [Add metadata] (メタデータの追加) を選択します。
 - b. [Type] (タイプ) で、[System defined] (システム定義) を選択します。
 - c. [Key] (キー) で、追加するヘッダーの名前 ([Cache-Control] または [Expires]) を選択します。
 - d. [Value] (値) で、ヘッダー値を入力します。例えば、Cache-Control ヘッダーの場合は、max-age=86400 と入力します。Expires で、有効期限の日時を Wed, 30 Jun 2021 09:28:00 GMT のように入力できます。
6. ページの最下部で [Edit metadata] (メタデータの編集) を選択します。

クエリ文字列パラメータに基づくコンテンツのキャッシュ

ウェブアプリケーションによっては、クエリ文字列を使用してオリジンに情報を送信します。クエリ文字列はウェブリクエストの一部で、? 文字の後に追加されます。この文字列には & 文字で区切られたパラメータを 1 つ以上含めることができます。次の例では、クエリ文字列には 2 つのパラメータ (*color=red* と *size=large*) が含まれています。

`https://d1111111abcdef8.cloudfront.net/images/image.jpg?color=red&size=large`

ディストリビューションでは、クエリ文字列 CloudFront をオリジンに転送するか、すべてのパラメータまたは選択したパラメータに基づいてコンテンツをキャッシュするかを選択できます。これが役立つ場合があるのはなぜですか。次の例を考えます。

たとえば、ウェブサイトが 5 種類の言語で使用でき、ディレクトリ構造とファイル名はウェブサイトの 5 つのバージョンすべてで共通だとします。ユーザーがウェブサイトを表示すると、ユーザー

が選択した言語に基づいて言語クエリ文字列パラメータ CloudFront を含めるように転送されるリクエスト。クエリ文字列 CloudFront をオリジンに転送し、言語パラメータに基づいてキャッシュするようにを設定できます。選択された言語に対応する特定バージョンのページを返すようウェブサーバーを設定した場合、CloudFront は、それぞれの言語によるクエリ文字列パラメータに基づく各言語のバージョンを個別にキャッシュします。

この例では、ウェブサイトのメインページが の場合main.html、次の 5 つのリクエストにより、言語クエリ文字列パラメータの値ごとに 1 回、 が main.html5 回 CloudFront キャッシュされます。

- <https://d111111abcdef8.cloudfront.net/main.html?language=de>
- <https://d111111abcdef8.cloudfront.net/main.html?language=en>
- <https://d111111abcdef8.cloudfront.net/main.html?language=es>
- <https://d111111abcdef8.cloudfront.net/main.html?language=fr>
- <https://d111111abcdef8.cloudfront.net/main.html?language=jp>

次の点に注意してください。

- 一部の HTTP サーバーはクエリ文字列パラメータを処理しません。このため、パラメータ値に基づくオブジェクトの別バージョンを返しません。これらのオリジンでは、クエリ文字列パラメータをオリジンに転送 CloudFront するようにを設定すると、オリジンがすべてのパラメータ値 CloudFront に対して同じバージョンのオブジェクトを に返した場合でも、パラメータ値に基づいてキャッシュが CloudFront 分離されます。
- 上記の例で説明したようにクエリ文字列パラメータを言語で使用するには、クエリ文字列パラメータ間の区切り文字として & 文字を使用する必要があります。別の区切り記号を使用すると、CloudFront がキャッシュのベースとして使用するよう指定したパラメータと、パラメータがクエリ文字列に表示される順序によっては、予期しない結果が発生する可能性があります。

次の例は、別の区切り文字を使用し、colorパラメータのみに基づいてキャッシュ CloudFront するようにを設定した場合の動作を示しています。

- 次のリクエストでは、 は colorパラメータの値に基づいてコンテンツを CloudFront キャッシュしますが、値は *red#size=large* と CloudFront 解釈します。

```
https://d111111abcdef8.cloudfront.net/images/  
image.jpg?color=red;size=large
```

- 次のリクエストでは、 はコンテンツを CloudFront キャッシュしますが、クエリ文字列パラメータに基づくキャッシュは行いません。これは、colorパラメータに基づいてキャッシュ

CloudFront するように を設定しましたが、次の文字列を large の値を持つsizeパラメータのみを含むものとして CloudFront 解釈するためです。color=red。

```
https://d1111111abcdef8.cloudfront.net/images/  
image.jpg?size=large;color=red
```

次のいずれかを実行する CloudFront ように を設定できます。

- クエリ文字列をオリジンにまったく転送しない。クエリ文字列を転送しない場合、CloudFront はクエリ文字列パラメータに基づくキャッシュを実行しません。
- クエリ文字列をオリジンに転送し、クエリ文字列内のすべてのパラメータに基づいてキャッシュする。
- クエリ文字列をオリジンに転送し、クエリ文字列内の指定したパラメータに基づいてキャッシュする。

詳細については、「[the section called “キャッシュの最適化”](#)」を参照してください。

トピック

- [クエリ文字列の転送とキャッシュのためのコンソールおよび API の設定](#)
- [キャッシュの最適化](#)
- [クエリ文字列パラメータと CloudFront 標準ログ \(アクセスログ\)](#)

クエリ文字列の転送とキャッシュのためのコンソールおよび API の設定

CloudFront コンソールでクエリ文字列の転送とキャッシュを設定するには、で次の設定を参照してください[the section called “指定する値”](#)。

- [the section called “クエリ文字列の転送とキャッシュ”](#)
- [the section called “クエリ文字列の許可リスト”](#)

CloudFront API を使用してクエリ文字列の転送とキャッシュを設定するには、「Amazon CloudFront API リファレンス」の[DistributionConfig](#)「」と[DistributionConfigWithTags](#)「」の次の設定を参照してください。

- QueryString

- QueryStringCacheKeys

キャッシュの最適化

クエリ文字列パラメータに基づいてキャッシュ CloudFront するように を設定する場合、次の手順を実行して、 がオリジン CloudFront に転送するリクエストの数を減らすことができます。CloudFront エッジロケーションがオブジェクトを提供する場合、オブジェクトはユーザーに近い場所から提供されるため、オリジンサーバーの負荷が軽減され、レイテンシーが短縮されます。

オリジンが返すオブジェクトのバージョンが変わるパラメータだけにに基づいてキャッシュする

ウェブアプリケーションが に転送するクエリ文字列パラメータごとに CloudFront、CloudFront はパラメータ値ごとにリクエストをオリジンに転送し、パラメータ値ごとにオブジェクトの個別のバージョンをキャッシュします。これは、オリジンがパラメータの値に関係なく常に同じオブジェクトを返す場合も当てはまります。パラメータが複数ある場合、リクエスト数とオブジェクトの数は乗算されます。たとえば、あるオブジェクトのリクエストにパラメータが 2 つ含まれていて、それぞれのパラメータに異なる値が 3 つある場合、CloudFront がキャッシュするオブジェクトのバージョンは 6 種類になります。この場合、このセクションの別の推奨事項を参照することをお勧めします。

オリジンが返すバージョンが異なるクエリ文字列パラメータのみに基づいてキャッシュ

CloudFront するように を設定し、各パラメータに基づいてキャッシュする利点を慎重に検討することをお勧めします。たとえば、ある通販ウェブサイトを運営していて、ジャケットの写真が色違いで 6 つあり、ジャケットのサイズは 10 種類だとします。また、ジャケットの写真は色違いだけが表示され、サイズ違いの分までは表示されていないものとします。キャッシュを最適化するには、サイズパラメータではなく色パラメータのみに基づいてキャッシュ CloudFront するように を設定する必要があります。これにより、 がキャッシュからのリクエストを処理 CloudFront できる可能性が高くなり、パフォーマンスが向上し、オリジンの負荷が軽減されます。

パラメータの順序を常に統一する

クエリ文字列では、パラメータの順序が重要になります。次の例は、パラメータの順序だけが異なる同じクエリ文字列です。これにより、CloudFront は image.jpg に対する 2 つの異なるリクエストをオリジンに転送し、2 つの異なるバージョンのオブジェクトをキャッシュします。

- `https://d111111abcdef8.cloudfront.net/images/image.jpg?color=red&size=large`
- `https://d111111abcdef8.cloudfront.net/images/image.jpg?size=large&color=red`

このため、パラメータ名は、常に同じ順序 (アルファベット順など) にすることをお勧めします。

パラメータ名とパラメータ値の大文字と小文字を常に統一する

CloudFront は、クエリ文字列パラメータに基づいてキャッシュするとき、パラメータ名と値の大文字と小文字を考慮します。次の例は、パラメータ名とパラメータ値の大文字と小文字だけが異なる、同じクエリ文字列です。これにより、CloudFront は image.jpg に対する 4 つの異なるリクエストをオリジンに転送し、4 つの異なるバージョンのオブジェクトをキャッシュします。

- `https://d1111111abcdef8.cloudfront.net/images/image.jpg?color=red`
- `https://d1111111abcdef8.cloudfront.net/images/image.jpg?color=Red`
- `https://d1111111abcdef8.cloudfront.net/images/image.jpg?Color=red`
- `https://d1111111abcdef8.cloudfront.net/images/image.jpg?Color=Red`

このため、パラメータ名とパラメータ値の大文字と小文字を統一する (すべて小文字など) ことをお勧めします。

署名付き URL と競合するパラメータ名を使わない

署名URLs を使用してコンテンツへのアクセスを制限している場合 (信頼された署名者をディストリビューションに追加した場合)、残りの URL をオリジンに転送する前に、次のクエリ文字列パラメータ CloudFront を削除します。

- Expires
- Key-Pair-Id
- Policy
- Signature

署名URLs を使用していて、クエリ文字列をオリジンに転送 CloudFront するようにを設定する場合、独自のクエリ文字列パラメータに Expires、Key-Pair-Id、Policy、または という名前を付けることはできません Signature。

クエリ文字列パラメータと CloudFront 標準ログ (アクセスログ)

ログ記録を有効にすると、はクエリ文字列パラメータを含む完全な URL CloudFront を記録します。これは、クエリ文字列をオリジンに転送 CloudFront するようにを設定しているかどうかに関係なく当てはまります。CloudFront ログ記録の詳細については、「」を参照してください [the section called “標準ログ \(アクセスログ\) の使用”](#)。

Cookie に基づくコンテンツのキャッシュ

デフォルトでは、CloudFront はリクエストとレスポンスを処理するとき、またはエッジロケーションでオブジェクトをキャッシュするときに Cookie を考慮しません。がヘッダーの内容を除いて同一の 2 つのリクエスト CloudFront を受信すると Cookie、デフォルトでは はリクエストを同一として CloudFront 処理し、両方のリクエストに対して同じオブジェクトを返します。

ビューワーリクエストの一部またはすべての Cookie をオリジン CloudFront に転送し、転送する Cookie 値に基づいてオブジェクトの個別のバージョンをキャッシュするように を設定できます。これを行うと、 は、転送するように設定されているものにかかわらず、ビューワーリクエストの一部またはすべての Cookie CloudFront を使用して、キャッシュ内のオブジェクトを一意に識別します。

たとえば、locations.html に対するリクエストに country Cookie が含まれており、その値が uk または fr であるとします。Cookie の値に基づいてオブジェクトをキャッシュ CloudFront するように country を設定すると、CloudFront は をオリジン locations.html にリクエストし、Cookie country とその値を含めます。オリジンは を返し locations.html、Cookie の値がであるリクエストでは 1 country 回 uk、値がであるリクエストでは 1 回、オブジェクトを CloudFront キャッシュします fr。

Important

Amazon S3 および一部の HTTP サーバーは Cookie を処理しません。Cookie を処理しない、または Cookie に基づいて応答を変更しないオリジンに Cookie を転送する CloudFront ように を設定しないでください。これにより、CloudFront が同じオブジェクトのオリジンにより多くのリクエストを転送する可能性があり、パフォーマンスが低下し、オリジンの負荷が増大します。前の例を考慮すると、オリジンが Cookie country を処理しないか、Cookie の値 CloudFront に関係なく常に同じバージョンの country locations.html を に返す場合は、その Cookie を転送する CloudFront ように を設定しないでください。

逆に、カスタムオリジンが特定の Cookie に依存している場合や、Cookie に基づいて異なるレスポンスを送信する場合は、その Cookie をオリジンに転送 CloudFront するように を設定してください。それ以外の場合は、リクエストをオリジンに転送する前に Cookie CloudFront を削除します。

Cookie 転送を設定するには、デистриビューションのキャッシュ動作を更新します。キャッシュ動作の詳細については、「[キャッシュ動作の設定](#)」の、特に「[cookie の転送](#)」および「[許可リスト Cookie](#)」セクションを参照してください。

各キャッシュ動作を設定して、次のいずれかを実行できます。

- すべての Cookie をオリジンに転送する – CloudFront リクエストをオリジンに転送するときにビューワーによって送信されるすべての Cookie が含まれます。オリジンがレスポンスを返すと、はビューワーリクエストの Cookie 名と値を使用してレスポンスを CloudFront キャッシュします。オリジンレスポンスに Set-Cookie ヘッダーが含まれている場合、はリクエストされたオブジェクトをビューワーに CloudFront 返します。はオリジンから返されたオブジェクトを含む Set-Cookie ヘッダー CloudFront もキャッシュし、すべてのキャッシュヒットでそれらの Set-Cookie ヘッダーをビューワーに送信します。
- 指定した一連の Cookie を転送する – CloudFront は、リクエストをオリジンに転送する前に、ビューワーが許可リストにない Cookie をすべて削除します。は、ビューワーリクエストにリストされている Cookie の名前と値を使用してレスポンスを CloudFront キャッシュします。オリジンレスポンスに Set-Cookie ヘッダーが含まれている場合、はリクエストされたオブジェクトをビューワーに CloudFront 返します。はオリジンから返されたオブジェクトを含む Set-Cookie ヘッダー CloudFront もキャッシュし、すべてのキャッシュヒットでそれらの Set-Cookie ヘッダーをビューワーに送信します。

Cookie 名でワイルドカードを指定する方法の詳細については、「[許可リスト Cookie](#)」を参照してください。

キャッシュ動作ごとに転送できる Cookie 名の数に関する現在のクォータについて、またはクォータの引き上げをリクエストするには、「[クエリ文字列のクォータ \(従来のキャッシュ設定\)](#)」を参照してください。

- Cookie をオリジンに転送しない - CloudFront ビューワーによって送信された Cookie に基づいてオブジェクトをキャッシュしません。さらに、CloudFront はリクエストをオリジンに転送する前に Cookie を削除し、レスポンスをビューワーに返す前にレスポンスから Set-Cookie ヘッダーを削除します。

転送する Cookie を指定するときには、以下に注意してください。

アクセスログ

がリクエストをログに記録し CloudFront、Cookie をログに記録するようにを設定する場合、Cookie をオリジンに転送 CloudFront しないように設定した場合や、特定の Cookie のみを転送する CloudFront ようにを設定した場合も、はすべての Cookie とすべての Cookie 属性を CloudFront ログに記録します。CloudFront ログ記録の詳細については、「」を参照してください [標準ログ \(アクセスログ\) の設定および使用](#)。

大文字と小文字の区別

Cookie の名前と値は、大文字と小文字を区別します。例えば、CloudFront がすべての Cookie を転送するように設定されていて、同じオブジェクトに対する 2 つのビューワーリクエストに、大文字と小文字を除いて同一の Cookie がある場合、はオブジェクトを 2 回 CloudFront キャッシュします。

CloudFront Cookie をソートする

CloudFront が Cookie (すべてまたはサブセット) を転送するように設定されている場合、はリクエストをオリジンに転送する前に、Cookie 名で自然な順序で Cookie を CloudFront ソートします。

If-Modified-Since および If-None-Match

If-Modified-Since Cookie (すべてまたはサブセット) を転送するように が設定されている場合 CloudFront、 および If-None-Match条件付きリクエストはサポートされません。

標準の名前と値のペア形式が必要

CloudFront は、値が [標準の名前と値のペア形式に準拠している場合にのみ](#) Cookie ヘッダーを転送します。次に例を示します。 "Cookie: cookie1=value1; cookie2=value2"

Set-Cookie ヘッダーのキャッシュの無効化

CloudFront が Cookie をオリジンに転送するように設定されている場合 (すべてまたは特定の Cookie)、オリジンレスポンスで受信した Set-Cookie ヘッダーもキャッシュします。は、これらの Set-Cookie ヘッダーを元のビューワーへのレスポンスに CloudFront 含め、CloudFront キャッシュから提供される後続のレスポンスにも含めます。

オリジンで Cookie を受け取りたいが、オリジンのレスポンスで Set-Cookie ヘッダーを CloudFront キャッシュしたくない場合は、フィールド名 Set-Cookie として を指定する no-cache デイレクティブで Cache-Control ヘッダーを追加するようにオリジンを設定します。例: Cache-Control: no-cache="Set-Cookie"。詳細については、「Hypertext Transfer Protocol (HTTP/1.1): Caching」標準の「[Response Cache-Control Directives](#)」を参照してください。

Cookie 名の全体の長さ

特定の Cookie CloudFront をオリジンに転送するように を設定した場合、転送 CloudFront するように設定するすべての Cookie 名の合計バイト数は、512 から転送する Cookie の数を引いた値を超えることはできません。例えば、10 個の Cookie CloudFront をオリジンに転送するように を設定した場合、10 個の Cookie の名前の合計長は 502 バイト (512 - 10) を超えることはできません。

すべての Cookie CloudFront をオリジンに転送するようにを設定した場合、Cookie 名の長さは関係ありません。

CloudFront コンソールを使用してディストリビューションを更新し、が Cookie をオリジン CloudFront に転送する方法については、「」を参照してください[ディストリビューションの更新](#)。CloudFront API を使用してディストリビューションを更新する方法については、「Amazon CloudFront API リファレンス」の[UpdateDistribution](#)「」を参照してください。

リクエストヘッダーに基づくコンテンツのキャッシュ

CloudFront では、ヘッダー CloudFront をオリジンに転送し、ビューワーリクエストのヘッダー値に基づいて、指定されたオブジェクトの個別のバージョンをキャッシュするかどうかを選択できます。こうすることで、ユーザーが使っているデバイスの種類やビューワーの場所、ビューワーで使われている言語など、さまざまな条件に基づいてコンテンツの異なるバージョンを配信できます。

トピック

- [ヘッダーとディストリビューションの概要](#)
- [キャッシュ条件に使用するヘッダーを選択する](#)
- [CORS 設定を優先 CloudFront するようにを設定する](#)
- [デバイスタイプに基づいてキャッシュを設定する](#)
- [ビューワーの言語に基づいてキャッシュを設定する](#)
- [ビューワーの場所に基づいてキャッシュを設定する](#)
- [リクエストのプロトコルに基づいてキャッシュを設定する](#)
- [圧縮ファイルのキャッシュの設定](#)
- [ヘッダーに基づくキャッシュがパフォーマンスに及ぼす影響](#)
- [ヘッダーとヘッダー値の大文字小文字がキャッシュに及ぼす影響](#)
- [がビューワーに CloudFront 返すヘッダー](#)

ヘッダーとディストリビューションの概要

デフォルトでは、CloudFront は、エッジロケーションでオブジェクトをキャッシュするときにヘッダーを考慮しません。オリジンが 2 つのオブジェクトを返し、リクエストヘッダーの値によってのみ異なる場合、CloudFront はオブジェクトの 1 つのバージョンのみをキャッシュします。

ヘッダー CloudFront をオリジンに転送するようにを設定できます。これにより、は 1 つ以上のリクエストヘッダーの値に基づいてオブジェクトの複数のバージョンを CloudFront キャッシュします。特定のヘッダーの値に基づいてオブジェクトをキャッシュするように CloudFront を設定するには、ディストリビューションのキャッシュ動作の設定を指定します。詳細については、「[選択されたリクエストヘッダーに基づいたキャッシュ](#)」を参照してください。

たとえば、logo.jpg のヘッダーオブジェクトがカスタム Product ヘッダーを含み、その値が Acme または Apex であるとします。Product ヘッダーの値に基づいてオブジェクトをキャッシュ CloudFront するようにを設定すると、CloudFront はをオリジン logo.jpg にリクエストし、Product ヘッダーの値がであるリクエストには 1 CloudFront logo.jpg 回、値がであるリクエストには 1 回、ヘッダー値 Acme と Product ヘッダー値を含めます Apex。

ディストリビューションの各キャッシュ動作を以下のいずれかを実行するように設定できます。

- すべてのヘッダーをオリジンに転送する

Note

レガシーキャッシュ設定の場合 – すべてのヘッダー CloudFront をオリジンに転送するようにを設定した場合、このキャッシュ動作に関連付けられたオブジェクトをキャッシュ CloudFront しないでください。その代わりに、すべてのリクエストをオリジンに送信します。

- 指定したヘッダーのリストを転送します。は、指定したすべてのヘッダーの値に基づいてオブジェクトを CloudFront キャッシュします。CloudFront また、はデフォルトで転送するヘッダーも転送しますが、指定したヘッダーのみに基づいてオブジェクトをキャッシュします。
- デフォルトのヘッダーのみを転送する。この設定では、CloudFront はリクエストヘッダーの値に基づいてオブジェクトをキャッシュしません。

キャッシュ動作ごとに転送できるヘッダーの数に関する現在のクォータについて、またはクォータの引き上げをリクエストするには、「[ヘッダーのクォータ](#)」を参照してください。

CloudFront コンソールを使用してディストリビューションを更新し、ヘッダーをオリジン CloudFront に転送する方法については、「」を参照してください [ディストリビューションの更新](#)。CloudFront API を使用して既存のディストリビューションを更新する方法については、「Amazon CloudFront API リファレンス」の「[ディストリビューションの更新](#)」を参照してください。

キャッシュ条件に使用するヘッダーを選択する

オリジンに転送でき、キャッシュ CloudFront に基づくヘッダーは、オリジンが Amazon S3 バケットであるかカスタムオリジンであるかによって異なります。

- Amazon S3 – 多数の特定のヘッダーに基づいてオブジェクトを CloudFront 転送およびキャッシュするようにを設定できます (次の例外のリストを参照)。ただし、Cross-Origin Resource Sharing (CORS) を実装するか、オリジン側イベントで Lambda@Edge を使用してコンテンツをパーソナライズする必要がない限り、Amazon S3 オリジンを使用してヘッダーを転送しないようにすることをお勧めします。
- CORS を設定するには、[が Cross-Origin Resource Sharing \(CORS\) が有効になっているウェブサイトのコンテンツを CloudFront 配信できるようにするヘッダーを転送する必要があります。](#)詳細については、「[CORS 設定を優先 CloudFront するようにを設定する](#)」を参照してください。
- Amazon S3 オリジンに転送するヘッダーを使用してコンテンツをパーソナライズするには、Lambda@Edge 関数を記述して追加し、オリジン向けイベントによってトリガーされるように CloudFront デイストリビューションに関連付けます。ヘッダーの操作によるコンテンツのパーソナライズの詳細については、「[国またはデバイスタイプヘッダー別のコンテンツのパーソナライズ - 例](#)」を参照してください。

使用していないヘッダーを転送してコンテンツをパーソナライズしないようにすることをお勧めします。追加のヘッダーを転送すると、キャッシュヒット率が低下する可能性があるためです。つまり、エッジキャッシュからのリクエストを、すべてのリクエストに占める割合だけ処理 CloudFront することはできません。

- カスタムオリジン – 以下を除くすべてのリクエストヘッダーの値に基づいてキャッシュ CloudFront するようにを設定できます。
 - Connection
 - Cookie - Cookie に基づいて転送しキャッシュする場合は、デイストリビューションの別の設定を使用します。詳細については、「[Cookie に基づくコンテンツのキャッシュ](#)」を参照してください。
 - Host (for Amazon S3 origins)
 - Proxy-Authorization
 - TE
 - Upgrade

ヘッダーDateと User-Agentヘッダーの値に基づいてオブジェクトをキャッシュ CloudFront するように を設定できますが、お勧めしません。これらのヘッダーには可能な値が多数あり、その値に基づいてキャッシュすると、CloudFront がオリジンに転送するリクエストの数が大幅に増加します。

HTTP リクエストヘッダーの完全なリストと、ヘッダー CloudFront の処理方法については、「」を参照してください[HTTP リクエストヘッダーと CloudFront 動作 \(カスタムオリジンと Amazon S3 オリジン\)](#)。

CORS 設定を優先 CloudFront するように を設定する

Cross-Origin Resource Sharing (CORS) を Amazon S3 バケットまたはカスタムオリジンで有効にしている場合、その CORS 設定を優先させるために、転送する特定のヘッダーを選択する必要があります。転送する必要があるヘッダーは、オリジン (Amazon S3 またはカスタム)、および OPTIONS レスポンスをキャッシュするかどうかによって異なります。

Amazon S3

- OPTIONS レスポンスをキャッシュする場合は、次の操作を行います。
 - OPTIONS レスポンスのキャッシュを有効にする、デフォルトのキャッシュ動作設定のオプションを選択します。
 - ヘッダー Origin、Access-Control-Request-Headers CloudFront を転送するように を設定しますAccess-Control-Request-Method。
- OPTIONS レスポンスをキャッシュしない場合は、オリジンに必要な他のヘッダー (たとえば、Access-Control-Request-Headers や Access-Control-Request-Method など) と一緒に Origin ヘッダーを転送するように CloudFront を設定します。

カスタムオリジン - オリジンが必要とする他のヘッダーと共に、Origin ヘッダーを転送します。

CORS に基づいてレスポンスをキャッシュ CloudFront するように を設定するには、キャッシュポリシーを使用してヘッダーを転送する CloudFront ように を設定する必要があります。詳細については、「[ポリシーの使用](#)」を参照してください。

CORS と Amazon S3 の詳細については、Amazon Simple Storage Service ユーザーガイドの「[Cross-Origin Resource Sharing \(CORS\) の使用](#)」を参照してください。

デバイスタイプに基づいてキャッシュを設定する

ユーザーがコンテンツの表示に使用しているデバイスに基づいてオブジェクトの異なるバージョンを CloudFront キャッシュする場合は、該当するヘッダーをカスタムオリジン CloudFront に転送するようにを設定します。

- CloudFront-Is-Desktop-Viewer
- CloudFront-Is-Mobile-Viewer
- CloudFront-Is-SmartTV-Viewer
- CloudFront-Is-Tablet-Viewer

User-Agent ヘッダーの値に基づいて、はリクエストをオリジンに転送 false する前に、これらのヘッダーの値を true または CloudFront に設定します。デバイスが複数のカテゴリに属する場合は、複数の値が true になることがあります。例えば、一部のタブレットデバイスでは、CloudFront-Is-Mobile-Viewer と CloudFront の両方 CloudFront-Is-Tablet-Viewer をに設定することができます true。

ビューワーの言語に基づいてキャッシュを設定する

リクエストで指定された言語に基づいてオブジェクトの異なるバージョンを CloudFront キャッシュする場合は、Accept-Language ヘッダーをオリジン CloudFront に転送するようにを設定します。

ビューワーの場所に基づいてキャッシュを設定する

リクエスト元の国に基づいて異なるバージョンのオブジェクトを CloudFront キャッシュする場合は、CloudFront-Viewer-Country ヘッダーをオリジン CloudFront に転送するようにを設定します。は、リクエスト元の IP アドレスを 2 文字の国コード CloudFront に自動的に変換します。国コード easy-to-use のリストについては、コード別および国名別にソートできます。Wikipedia の「[ISO 3166-1 alpha-2](#)」のエントリを参照してください。

リクエストのプロトコルに基づいてキャッシュを設定する

リクエストのプロトコル HTTP または HTTPS に基づいてオブジェクトの異なるバージョンを CloudFront キャッシュする場合は、CloudFront-Forwarded-Proto ヘッダーをオリジン CloudFront に転送するようにを設定します。

圧縮ファイルのキャッシュの設定

オリジンが Brotli 圧縮をサポートしている場合は、Accept-Encoding ヘッダーに基づいてキャッシュできます。オリジンがヘッダーに基づいて異なるコンテンツを配信する場合のみ、Accept-Encoding に基づいてキャッシュを設定する必要があります。

ヘッダーに基づくキャッシュがパフォーマンスに及ぼす影響

1 つ以上のヘッダーに基づいて CloudFront キャッシュするようにを設定し、ヘッダーに複数の可能な値がある場合、CloudFront 同じオブジェクトに対するオリジンサーバーへのリクエストが増えます。このためパフォーマンスが低下し、オリジンサーバーの負荷が増加します。オリジンサーバーが特定のヘッダーの値に関係なく同じオブジェクトを返す場合は、そのヘッダーに基づいてキャッシュ CloudFront するようにを設定しないことをお勧めします。

複数のヘッダーを転送する CloudFront ようにを設定した場合、ビューワーリクエストのヘッダーの順序は、値が同じである限り、キャッシュには影響しません。例えば、あるリクエストにヘッダー A:1、B:2 が含まれ、別のリクエストに B:2、A:1 が含まれている場合、CloudFront はオブジェクトのコピーを 1 つだけキャッシュします。

ヘッダーとヘッダー値の大文字小文字がキャッシュに及ぼす影響

がヘッダー値に基づいて CloudFront キャッシュする場合、ヘッダー名の大文字と小文字は考慮されませんが、ヘッダー値の大文字と小文字は考慮されます。

- ビューワーリクエストに Product:Acmeと の両方が含まれている場合product:Acme、 はオブジェクトを 1 回だけ CloudFront キャッシュします。両者の違いはヘッダー名の大文字小文字だけで、これはキャッシュ動作に影響しません。
- ビューワーリクエストに Product:Acmeと の両方が含まれている場合Product:acme、 はオブジェクトを 2 回 CloudFront キャッシュします。これは、値が一部のリクエストAcmeでは、他のリクエストacmeでは であるためです。

がビューワーに CloudFront 返すヘッダー

ヘッダーを転送してキャッシュ CloudFront するようにを設定しても、ビューワーに返されるヘッダー CloudFrontには影響しません。は、いくつかの例外を除いて、オリジンから取得した CloudFront すべてのヘッダーを返します。詳細については、該当するトピックを参照してください。

- Amazon S3 のオリジン - 「[削除または更新する CloudFront HTTP レスポンスヘッダー](#)」を参照してください。
- カスタムオリジン - 「[CloudFront を削除または置き換える HTTP レスポンスヘッダー](#)」を参照してください。

トラブルシューティング

コンテンツを配信 CloudFront するように Amazon を設定するとき、または Lambda@Edge を使用するときには発生する可能性のある一般的な問題をトラブルシューティングし、考えられる解決策を見つけます。

トピック

- [トラブルシューティング: ディストリビューション](#)
- [オリジンからのエラーレスポンスのトラブルシューティング](#)
- [負荷テスト CloudFront](#)

トラブルシューティング: ディストリビューション

この情報を使用して、証明書エラー、アクセス拒否の問題、または Amazon CloudFront ディストリビューションでウェブサイトやアプリケーションを設定するときには発生する可能性のあるその他の一般的な問題の診断や修復に役立ててください。

トピック

- [CloudFront 代替ドメイン名を追加しようとする、がInvalidViewerCertificateエラーを返します。](#)
- [ディストリビューション内のファイルを表示できません](#)
- [エラーメッセージ: Certificate: <certificate-id> is used by CloudFront](#)

CloudFront 代替ドメイン名を追加しようとする、がInvalidViewerCertificateエラーを返します。

ディストリビューションに代替ドメイン名 (CNAME) を追加しようとしたときに、がInvalidViewerCertificateエラーを CloudFront 返す場合は、以下の情報を確認して問題のトラブルシューティングに役立ててください。このエラーは、代替ドメイン名を正常に追加できる前に次のいずれかの問題を解決する必要があることを示している可能性があります。

以下のエラーは、が代替ドメイン名を追加する認可 CloudFront をチェックする順序でリストされています。これは、が CloudFront 返すエラーに基づいて、どの検証チェックが正常に完了したかを知ることができるため、問題のトラブルシューティングに役立ちます。

ディストリビューションにアタッチされた証明書はありません。

代替ドメイン名 (CNAME) を追加するには、信頼された、有効な証明書をディストリビューションにアタッチする必要があります。前提条件を確認し、これを満たす有効な証明書を取得してディストリビューションにアタッチしてから、操作をやり直してください。詳細については、「[代替ドメイン名を使用するための要件](#)」を参照してください。

アタッチした証明書の証明書チェーンに多すぎる証明書があります。

証明書チェーンには最大 5 つの証明書のみを含むことができます。チェーンの証明書の数を減らしてから、操作をやり直してください。

証明書チェーンには 1 つまたは複数の現在の日付には有効ではない証明書が含まれています。

追加した証明書の証明書チェーンに、証明書がまだ有効ではないあるいは有効期限切れの 1 つ以上の証明書が含まれています。証明書チェーンの証明書の [Not Valid Before] (有効期限開始日) フィールドおよび [Not Valid After] (失効日) フィールドをチェックし、リストした日付に基づいてすべての証明書が有効であることを確認します。

アタッチした証明書は、信頼される認証機関 (CA) によって署名されていません。

代替ドメイン名を検証 CloudFront するために アタッチする証明書は、自己署名証明書にすることはできません。信頼される認証機関によって署名される必要があります。詳細については、「[代替ドメイン名を使用するための要件](#)」を参照してください。

アタッチした証明書が、正しくフォーマットされない

証明書に含まれているドメイン名と IP アドレス形式、および証明書自体の形式は、証明書の標準に従っている必要があります。

CloudFront 内部エラーが発生しました。

CloudFront は内部の問題によってブロックされ、証明書を検証できませんでした。このシナリオでは、は HTTP 500 ステータスコードを CloudFront 返し、証明書をアタッチする際に内部 CloudFront の問題があることを示します。数分間待機してから、証明書に代替ドメイン名の追加を再試行します。

アタッチした証明書が、追加しようとしている代替ドメイン名を対象としていません。

追加する代替ドメイン名ごとに、CloudFront では、ドメイン名を対象とする信頼された認証機関 (CA) から有効な SSL/TLS 証明書をアタッチして、その証明書を使用する権限を検証する必要があります。証明書を更新して、追加しようとしている CNAME を対象とするドメイン名を含めてください。ワイルドカードがあるドメイン名を使用する詳細と例については、「[代替ドメイン名を使用するための要件](#)」を参照してください。

ディストリビューション内のファイルを表示できません

CloudFront ディストリビューション内のファイルを表示できない場合は、いくつかの一般的なソリューションについて以下のトピックを参照してください。

CloudFront と Amazon S3 の両方にサインアップしましたか？

Amazon S3 オリジン CloudFront で Amazon を使用するには、CloudFront と Amazon S3 の両方に個別にサインアップする必要があります。Amazon S3 CloudFront および Amazon S3 へのサインアップの詳細については、「」を参照してください[設定](#)。

Amazon S3 バケットとオブジェクトのアクセス許可は正しく設定されていますか？

Amazon S3 オリジン CloudFront で を使用している場合、コンテンツの元のバージョンは S3 バケットに保存されます。Amazon S3 CloudFront で を使用する最も簡単な方法は、すべてのオブジェクトを Amazon S3 でパブリックに読み取り可能にすることです。そのためには、Amazon S3 にアップロードするオブジェクトごとに公開特権を明示的に有効にする必要があります。

コンテンツがパブリックに読み取り可能でない場合は、 が CloudFront コンテンツにアクセスできるように CloudFront オリジンアクセスコントロール (OAC) を作成する必要があります。CloudFront オリジンアクセスコントロールの詳細については、「」を参照してください[the section called “Amazon S3 オリジンへのアクセスの制限”](#)。

オブジェクトのプロパティとバケットのプロパティはそれぞれ独立しています。権限は Amazon S3 のオブジェクトそれぞれに対して明示的に付与する必要があります。オブジェクトのプロパティはバケットから取得できないので、別途設定する必要があります。

代替ドメイン名 (CNAME) が正しく設定されていますか？

お使いのドメイン名に対応する CNAME レコードがすでに存在する場合は、そのレコードを更新または変更して、レコードがディストリビューションのドメイン名を指すようにしてください。

CNAME レコードが、Amazon S3 バケットではなく、ディストリビューションのドメイン名を指していることも確認してください。DNS システムの CNAME レコードがディストリビューションのドメイン名を指していることを確認することができます。そのためには、dig などの DNS ツールを使用します。

以下は、images.example.com というドメイン名に対する dig リクエストと、レスポンスのうち関連する部分のサンプルです。ANSWER SECTION の下で、CNAME が含まれる行を探します。CNAME

の右側にある値が CloudFront デイストリビューションのドメイン名である場合、ドメイン名の CNAME レコードが正しく設定されます。もしそれが Amazon S3 のオリジンサーバーのバケットや他のドメイン名になっている場合は、その CNAME レコードは正しく設定されていません。

```
[prompt]> dig images.example.com

; <<> DiG 9.3.3rc2 <<> images.example.com
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15917
;; flags: qr rd ra; QUERY: 1, ANSWER: 9, AUTHORITY: 2, ADDITIONAL: 0
;; QUESTION SECTION:
;images.example.com.      IN      A
;; ANSWER SECTION:
images.example.com. 10800 IN CNAME d111111abcdef8.cloudfront.net.
...
...
```

CNAME の詳細については、「[代替ドメイン名 \(CNAME\) を追加することによるカスタム URL の使用](#)」を参照してください。

デイストリビューションの正しい URL CloudFront を参照していますか？

参照する URL が Amazon S3 バケットやカスタムオリジンではなく、CloudFront デイストリビューションのドメイン名 (または CNAME) を使用していることを確認してください。Amazon S3

カスタムオリジンに関するトラブルシューティングでサポートが必要ですか？

カスタムオリジンのトラブルシューティングで AWS のサポートが必要な場合は、おそらくリクエストからの X-Amz-Cf-Id ヘッダーエントリの調査が必要になります。現在ヘッダーエントリのログを記録していない場合は、将来に備えて記録することをお勧めします。詳しくは、「[the section called “Amazon EC2 \(または他のカスタムオリジン\) の使用”](#)」を参照してください。詳細については、[AWS サポートセンター](#)までお問い合わせください。

エラーメッセージ: Certificate: <certificate-id> is used by CloudFront

問題: IAM 証明書ストアから SSL/TLS 証明書を削除しようとする時、「Certificate: <certificate-id> is used by」というメッセージが表示されます CloudFront。

解決策: すべての CloudFront デイストリビューションをデフォルトの CloudFront 証明書またはカスタム SSL/TLS 証明書に関連付ける必要があります。SSL/TLS 証明書を削除する前に、SSL/TLS

証明書を更新するか (現行の独自 SSL/TLS 証明書を別の独自 SSL/TLS 証明書に置き換える)、または使用する証明書を独自 SSL/TLS 証明書からデフォルトの CloudFront 証明書に戻してください。この問題を解決するには、次のいずれかの手順のステップを実行します。

- [SSL/TLS 証明書の更新](#)
- [カスタム SSL/TLS 証明書からデフォルト CloudFront 証明書に戻す](#)

オリジンからのエラーレスポンスのトラブルシューティング

がオリジンからオブジェクトを CloudFront リクエストし、オリジンが HTTP 4xx または 5xx ステータスコードを返す場合、CloudFront とオリジン間の通信に問題があります。以下のトピックでは、これらの HTTP ステータスコードの一般的な原因と、考えられる解決策について説明します。

トピック

- [HTTP 400 ステータスコード \(Bad Request\)](#)
- [HTTP 502 ステータスコード \(Bad Gateway\)](#)
- [HTTP 502 ステータスコード \(Lambda validation error\)](#)
- [HTTP 502 ステータスコード \(DNS error\)](#)
- [HTTP 503 ステータスコード \(関数実行エラー\)](#)
- [HTTP 503 ステータスコード \(Lambda limit exceeded\)](#)
- [HTTP 503 ステータスコード \(Service Unavailable\)](#)
- [HTTP 504 ステータスコード \(Gateway Timeout\)](#)

HTTP 400 ステータスコード (Bad Request)

CloudFront デイストリビューションは、HTTP ステータスコード 400 Bad Request のエラーレスポンスと、次のようなメッセージを送信する場合があります。

The authorization header is malformed; the region '*<AWS Region>*' is wrong; expecting '*<AWS Region>*'

次に例を示します。

認証ヘッダーの形式が正しくありません。リージョン 'us-east-1' が間違っています。'us-west-2' を予期しています。

この問題は、次のシナリオで発生する可能性があります。

1. CloudFront デイストリビューションのオリジンは Amazon S3 バケットです。
2. S3 バケットを 1 つの AWS リージョンから別のリージョンに移動した。つまり、S3 バケットを削除し、後ほど同じバケット名で新しい S3 バケットを作成したが、元の S3 バケットがあった AWS リージョンとはちがうリージョンで作成したということです。

このエラーを修正するには、バケットの現在のAWSリージョンで S3 バケットが見つかるように、`CloudFront` トリビューションを更新します。

CloudFront デイストリビューションを更新するには

1. `にサインイン` AWS Management Console し、`で CloudFront コンソールを開きます` <https://console.aws.amazon.com/cloudfront/v4/home>。
2. このエラーを生成するデイストリビューションを選択します。
3. [Origin and Origin Groups (オリジンおよびオリジングループ)] を選択します。
4. 移動した S3 バケットのオリジンを見つけます。このオリジンの横にあるチェックボックスをオンにして、[編集] を選択します。
5. [Yes, Edit (はい、編集します)] を選択します。[Yes, Edit (はい、編集します)] を選択する前に、設定を変更する必要はありません。

これらのステップを完了すると、`は` デイストリビューション `CloudFront` を再デプロイします。デイストリビューションのデプロイ中に、最終更新日時列の下にデプロイステータスが表示されます。デプロイが完了してからしばらくすると、`AuthorizationHeaderMalformed` エラーレスポンスの受信を停止する必要があります。

HTTP 502 ステータスコード (Bad Gateway)

HTTP 502 ステータスコード (Bad Gateway) は、`CloudFront` がオリジンサーバーに接続できなかったため、リクエストされたオブジェクトを提供できなかったことを示します。

トピック

- [CloudFront とカスタムオリジンサーバー間の SSL/TLS ネゴシエーションの失敗](#)
- [サポートされている暗号化/プロトコルではオリジンが応答しません](#)
- [オリジンの SSL/TLS 証明書が期限切れ、無効、自己署名になっている、または間違った順番の証明書チェーンになっている](#)

- [オリジンがオリジン設定のポート指定に応答しません](#)

CloudFront とカスタムオリジンサーバー間の SSL/TLS ネゴシエーションの失敗

カスタムオリジンを使用し、CloudFront とオリジンの間で HTTPS を必須 CloudFront にするようにを設定した場合、問題はドメイン名の不一致である可能性があります。オリジンにインストールした SSL/TLS 証明書では、[Common Name (共通名)] フィールドにドメイン名が含まれ、[Subject Alternative Names (サブジェクトの代替名)] フィールドにもドメイン名がいくつか含まれることがあります。(証明書ドメイン名でワイルドカード文字 CloudFront をサポートします)。証明書のドメイン名の 1 つは、次の値の 1 つまたは両方と一致する必要があります。

- ディストリビューションの該当するオリジンの [Origin Domain Name] に指定した値。
- Host ヘッダーをオリジンに転送 CloudFront するようにを設定した場合の Host ヘッダーの値。Host ヘッダーのオリジンへの転送の詳細については、「[リクエストヘッダーに基づくコンテンツのキャッシュ](#)」を参照してください。

ドメイン名が一致しない場合、SSL/TLS ハンドシェイクは失敗し、HTTP ステータスコード 502 (Bad Gateway) CloudFront が返され、X-Cacheヘッダーが に設定されず Error from cloudfront。

証明書のドメイン名がディストリビューションまたは Host ヘッダーの [Origin Domain Name] と一致するかどうかを確認するには、オンライン SSL チェッカーまたは OpenSSL を使用できます。ドメイン名が一致しない場合、2 つのオプションがあります。

- 該当するドメイン名を含む新しい SSL/TLS 証明書を取得します。

AWS Certificate Manager (ACM) を使用する場合は、AWS Certificate Manager ユーザーガイドの「[パブリック証明書をリクエストする](#)」を参照して、新しい証明書をリクエストしてください。

- ディストリビューション設定を変更して、オリジンへの接続に SSL を使用し CloudFront ないようにします。

オンライン SSL チェッカー

SSL テスト ツールを見つけるには、インターネットで「online ssl checker」を検索します。通常、ドメイン名を指定すると、ツールから SSL/TLS 証明書に関するさまざまな情報が返されます。証明書の [Common Names] フィールドまたは [Subject Alternative Names] フィールドにドメイン名が含まれていることを確認します。

OpenSSL

からの HTTP 502 エラーのトラブルシューティングに役立つように CloudFront、OpenSSL を使用してオリジンサーバーへの SSL/TLS 接続を試みることができます。OpenSSL が接続できない場合、オリジンサーバーの SSL/TLS 設定に問題がある可能性があります。OpenSSL が接続を確立できる場合、証明書の共通名 (Subject CN フィールド) やサブジェクト代替名 (Subject Alternative Name フィールド) など、オリジンサーバーの証明書に関する情報を返します。

次の OpenSSL コマンドを使用して、オリジンサーバーへの接続をテストします (##### を example.com などのオリジンサーバーのドメイン名に置き換えます)。

```
openssl s_client -connect origin domain name:443
```

次のことが当てはまるとします。

- オリジンサーバーは、複数の SSL/TLS 証明書を持つ複数のドメイン名をサポートしている
- Host ヘッダーをオリジンに転送するようにディストリビューションが設定されている

この場合、次の例のように OpenSSL コマンドに `-servername` オプションを追加します (`CNAME` をディストリビューションで設定した CNAME に置き換えます)。

```
openssl s_client -connect origin domain name:443 -servername CNAME
```

サポートされている暗号化/プロトコルではオリジンが応答しません

CloudFront は、暗号とプロトコルを使用してオリジンサーバーに接続します。が CloudFront サポートする暗号とプロトコルのリストについては、「」を参照してください[the section called “とオリジン間でサポートされているプロトコル CloudFront と暗号”](#)。オリジンが SSL/TLS 交換でこれらの暗号またはプロトコルのいずれかで応答しない場合、は接続に CloudFront 失敗します。[SSL Labs](#) などのオンラインツールを使って、オリジンが暗号とプロトコルをサポートすることを確認できます。[Host Name] フィールドでオリジンのドメイン名を入力し、[Submit] を選択します。テスト結果の [Common names] フィールドと [Alternative names] フィールドを見て、オリジンのドメイン名と一致しているかどうかを確認します。テスト完了後、テスト結果の [Protocols] または [Cipher Suites] セクションでオリジンがサポートする暗号とプロトコルを確認してください。それらを「[the section called “とオリジン間でサポートされているプロトコル CloudFront と暗号”](#)」のリストと比較します。

オリジンの SSL/TLS 証明書が期限切れ、無効、自己署名になっている、または間違っ
た順番の証明書チェーンになっている

オリジンサーバーが以下を返す場合、 は TCP 接続を CloudFront ドロップし、HTTP ステータスコード 502 (Bad Gateway) を返し、 X-Cacheヘッダーを に設定しますError from cloudfront。

- 証明書が期限切れです
- 証明書が無効です
- 証明書が自己署名です
- 間違っした順番の証明書チェーンです

Note

中間証明書を含む証明書の完全なチェーンが存在しない場合、 は TCP 接続を CloudFront ドロップします。

カスタムオリジンサーバーで SSL/TLS 証明書をインストールする方法の詳細については、「[the section called “カスタムオリジンに対して HTTPS を必須にする”](#)」を参照してください。

オリジンがオリジン設定のポート指定に応答しません

CloudFront デイストリビューションでオリジンを作成するときに、HTTP および HTTPS トラフィック用に を使用してオリジンに接続するポート CloudFrontを設定できます。デフォルトでは TCP 80/443です。これらのポートは変更可能です。オリジンが何らかの理由でこれらのポートのトラフィックを拒否している場合、またはバックエンドサーバーがポートで応答していない場合、CloudFront は接続に失敗します。

これらの問題におけるトラブルシューティングには、インフラストラクチャで稼働するファイアウォールを確認し、サポートする IP 範囲がブロックされていないかを確認します。詳細については、Amazon Web Services 全般のリファレンスの [AWS IP アドレスの範囲](#)をご参照ください。ウェブサーバーがオリジンで稼働中であるかどうかも確認してください。

HTTP 502 ステータスコード (Lambda validation error)

Lambda@Edge を使用している場合、HTTP 502 ステータスコードは、Lambda 関数のレスポンスの形式が正しくないか、レスポンスに無効なコンテンツが含まれていたことを示している可能性があります

まず、Lambda@Edge エラーのトラブルシューティングの詳細については、「[Lambda@Edge 関数のテストとデバッグ](#)」を参照してください。

HTTP 502 ステータスコード (DNS error)

エラーコードを含む HTTP 502 NonS3OriginDnsError エラーは、がオリジンに接続 CloudFront できない DNS 設定の問題があることを示しています。からこのエラーが発生した場合は CloudFront、オリジンの DNS 設定が正しく機能していることを確認します。

は、有効期限が切れているか、キャッシュにないオブジェクトのリクエスト CloudFront を受け取ると、オリジンにオブジェクトの取得をリクエストします。オリジンへのリクエストを正常に行うために、はオリジンドメイン名に対して DNS 解決 CloudFront を実行します。ドメインの DNS サービスに問題が発生した場合は、ドメイン名を解決して IP アドレスを取得 CloudFront できないため、HTTP 502 エラー () が発生します NonS3OriginDnsError。この問題を解決するには、DNS プロバイダーにお問い合わせください。Amazon Route 53 を使用している場合は、「[Route 53 DNS サービスを使用している自分のウェブサイトアクセスできないのはなぜですか?](#)」を参照してください。

この問題の詳しいトラブルシューティングを行うには、オリジンのルートドメインまたは zone apex (example.com など) の[権威ネームサーバー](#)が正しく機能していることを確認します。[dig](#) や [nslookup](#) などのツールにより、次のコマンドを使用して apex オリジンのネームサーバーを検索できます。

```
dig OriginAPEXDomainName NS +short
```

```
nslookup -query=NS OriginAPEXDomainName
```

ネームサーバーの名前がある場合、次のコマンドを使用して、それらに対してオリジンのドメイン名のクエリを実行し、各サーバーが応答して答えを返すことを確認します。

```
dig OriginDomainName @NameServer
```

```
nslookup OriginDomainName NameServer
```

Important

パブリックインターネットに接続されているコンピュータを使用して、この DNS トラブルシューティングを実行していることを確認してください。はインターネット上のパブリック

DNS を使用してオリジンドメイン名を CloudFront 解決するため、同様のコンテキストでトラブルシューティングすることが重要です。

オリジンがサブドメインであり、このサブドメインの DNS 権限がルートドメインとは異なるネームサーバーに委任されている場合は、ネームサーバー (NS) および Start of Authority (SOA) レコードが、このサブドメインに対して正しく設定されていることを確認してください。これらのレコードは、前述の例と同様のコマンドを使用して確認できます。

DNS の詳細については、Amazon Route 53 ドキュメントの「[ドメインネームシステム \(DNS\) の概念](#)」を参照してください。

HTTP 503 ステータスコード (関数実行エラー)

Lambda@Edge または CloudFront Functions を使用している場合、HTTP 503 ステータスコードは、関数が実行エラーを返したことを示している可能性があります。

Lambda@Edge エラーのトラブルシューティングの詳細については、「[Lambda@Edge 関数のテストとデバッグ](#)」を参照してください。

CloudFront 関数のトラブルシューティングの詳細については、「」を参照してください [関数のテスト](#)。

HTTP 503 ステータスコード (Lambda limit exceeded)

Lambda@Edge を使用している場合、HTTP 503 ステータスコードは、Lambda サービスがエラーを返したことを示している可能性があります。このエラーは、次のいずれかの条件によって発生する可能性があります。

- 関数の実行数が、AWS リージョンでの実行をスロットリングするために Lambda が設定したクォータ (以前は制限と呼ばれていたもの) の 1 つを超過している (同時実行数または呼び出し頻度)。
- 関数が Lambda 関数のタイムアウトクォータを超過している。

AWS Lambda クォータの詳細については、AWS Lambda デベロッパーガイドの「[Lambda クォータ](#)」を参照してください。Lambda@Edge エラーのトラブルシューティングの詳細については、「[the section called “テストとデバッグ”](#)」を参照してください。

HTTP 503 ステータスコード (Service Unavailable)

通常、HTTP 503 ステータスコード (Service Unavailable) は、オリジンサーバーのパフォーマンスの問題を示します。まれに、エッジロケーションのリソース制約により、が CloudFront 一時的にリクエストを満たせないことを示します。

トピック

- [オリジンサーバーにリクエスト率をサポートする十分な容量がない](#)
- [CloudFront エッジロケーションのリソース制約が原因で がエラーの原因となった](#)

オリジンサーバーにリクエスト率をサポートする十分な容量がない

CloudFront は、オリジンサーバーが受信リクエストに圧倒されると、このエラーを生成します。CloudFront その後、 はエラーをユーザーに中継します。この問題を解決するには、以下の手順をお試しください。

- Amazon S3 をオリジンサーバーとして使用している場合は、キー命名規則のベストプラクティスに従って Amazon S3 のパフォーマンスを最適化します。詳細については、Amazon Simple Storage Service ユーザーガイドの「[Amazon S3 のパフォーマンスの最適化](#)」を参照してください。
- Elastic Load Balancing をオリジンサーバーとして使用している場合は、「[Classic Load Balancer 使用中に返された 503 エラーをトラブルシューティングするにはどうすればいいですか](#)」を参照してください。
- カスタムオリジンを使用している場合は、アプリケーションログを調べて、オリジンにメモリ、CPU、ディスクサイズなどのリソースが十分にあることを確認します。Amazon EC2 をバックエンドとして使用している場合は、受信されるリクエストを満たす適切なリソースがインスタンスタイプにあることを確認します。詳細については、Linux インスタンス用 Amazon EC2 ユーザーガイドの「[インスタンスタイプ](#)」を参照してください。

CloudFront エッジロケーションのリソース制約が原因で がエラーの原因となった

このエラーは、次に利用可能なエッジロケーションにリクエストをルーティング CloudFront できず、リクエストを満たすことができないというまれな状況で発生します。このエラーは、CloudFront デイストリビューションで負荷テストを実行するときによく発生します。これを回避するには、「[the section called “負荷テスト CloudFront”](#)」のガイドラインに従って 503 (キャパシティー超過) エラーが発生しないようにします。

本稼働環境でこのエラーが発生した場合は、[AWS サポート](#)にお問い合わせください。

HTTP 504 ステータスコード (Gateway Timeout)

HTTP 504 ステータスコード (Gateway Timeout) は、 がリクエストをオリジン CloudFront に転送したとき (リクエストされたオブジェクトがエッジキャッシュになかったため)、次のいずれかが発生したことを示します。

- オリジンが HTTP 504 ステータスコードを に返しました CloudFront。
- リクエストの期限切れまでにオリジンが応答しなかった。

CloudFront は、ファイアウォールまたはセキュリティグループによってトラフィックがオリジンにブロックされている場合、またはインターネットでオリジンにアクセスできない場合、HTTP 504 ステータスコードを返します。最初に、これらの問題を確認します。次に、アクセスに問題がない場合は、アプリケーションの遅延とサーバーのタイムアウトを調べると、問題の特定と修正に役立ちます。

トピック

- [CloudFront トラフィックを許可するようにオリジンサーバーのファイアウォールを設定する](#)
- [CloudFront トラフィックを許可するようにオリジンサーバーのセキュリティグループを設定する](#)
- [インターネットでカスタムオリジンサーバーをアクセス可能にする](#)
- [オリジンサーバーでアプリケーションからの遅延したレスポンスを見つけて修正する](#)

CloudFront トラフィックを許可するようにオリジンサーバーのファイアウォールを設定する

オリジンサーバーのファイアウォールが CloudFront トラフィックをブロックしている場合、 は HTTP 504 ステータスコードを CloudFront 返すため、他の問題をチェックする前に、 が問題でないことを確認することをお勧めします。

これがファイアウォールの問題であるかどうかを判断するために使用する方法は、オリジンサーバーが使用しているシステムによって異なります。

- Linux サーバーで IPTable ファイアウォールを使用している場合は、IPTables を操作するのに役立つツールと情報を検索できます。

- Windows サーバーで Windows ファイアウォールを使用している場合は、Microsoft ドキュメントの「[Add or Edit Firewall Rule](#)」(ファイアウォール規則を追加または編集する)を参照してください。

オリジンサーバーのファイアウォール設定を評価するときは、[公開されている IP アドレス範囲](#)に基づいて、CloudFront エッジロケーションからのトラフィックをブロックするファイアウォールまたはセキュリティルールを探します。

CloudFront IP アドレス範囲がオリジンサーバーへの接続を許可されている場合は、必ずサーバーのセキュリティルールを更新して変更を組み込んでください。Amazon SNS トピックにサブスクライブして、IP アドレス範囲ファイルが更新されたときに通知を受け取ることができます。通知を受け取ったら、コードを使用してファイルを取得し、解析して、ローカル環境を調整することができます。詳細については、AWS ニュースブログの「[Subscribe to AWS Public IP Address Changes via Amazon SNS](#)」を参照してください。

CloudFront トラフィックを許可するようにオリジンサーバーのセキュリティグループを設定する

オリジンが Elastic Load Balancing を使用している場合は、[ELB セキュリティグループ](#)を確認し、セキュリティグループがからのインバウンドトラフィックを許可していることを確認します CloudFront。

AWS Lambda を使用して、からのインバウンドトラフィックを許可するようにセキュリティグループを自動的に更新することもできます CloudFront。

インターネットでカスタムオリジンサーバーをアクセス可能にする

インターネットで公開されていないために CloudFront がカスタムオリジンサーバーにアクセスできない場合、は HTTP 504 エラー CloudFront を返します。

CloudFront エッジロケーションは、インターネット経由でオリジンサーバーに接続します。カスタムオリジンがプライベートネットワーク上にある場合、そのオリジンに到達 CloudFront することはできません。このため、[内部 Classic Load Balancer](#) を含むプライベートサーバーをのオリジンサーバーとして使用することはできません CloudFront。

インターネットトラフィックがオリジンサーバーに接続できることを確認するには、次のコマンドを実行します (OriginDomainName はサーバーのドメイン名です)。

HTTPS トラフィックの場合:

- nc -zv OriginDomainName 443
- telnet OriginDomainName 443

HTTP トラフィックの場合:

- nc -zv OriginDomainName 80
- telnet OriginDomainName 80

オリジンサーバーでアプリケーションからの遅延したレスポンスを見つけて修正する

サーバーのタイムアウトは、多くの場合、アプリケーションの応答に非常に長い時間がかかっているか、タイムアウト値の設定が低すぎる場合に発生します。

HTTP 504 エラーを回避するための簡単な修正は、ディストリビューションの CloudFront タイムアウト値を高く設定するだけで済みます。ただし、アプリケーションとオリジンサーバーのパフォーマンスとレイテンシーの問題があれば、最初にその問題に対応することをお勧めします。次に、HTTP 504 エラーを回避してユーザーに良好な応答性を提供する、適切なタイムアウト値を設定できます。

パフォーマンスの問題を見つけて修正するための手順について、以下に概要を示します。

1. ウェブアプリケーションの一般的な高負荷のレイテンシー (応答性) を測定します。
2. 必要に応じて CPU やメモリなどのリソースを追加します。データベースクエリを高負荷シナリオに対応するようにチューニングするなど、問題に対応する他のステップを実行します。
3. 必要に応じて、CloudFront ディストリビューションのタイムアウト値を調整します。

各ステップの詳細を以下に示します。

一般的な高負荷のレイテンシーの測定

1 台以上のバックエンドウェブアプリケーションサーバーで長いレイテンシーが発生しているかどうか調べるには、各サーバーで次の Linux curl コマンドを実行します。

```
curl -w "Connect time: %{time_connect} Time to first byte:
%{time_starttransfer} Total time: %{time_total} \n" -o /dev/null https://
www.example.com/yourobject
```

Note

サーバーで Windows を実行する場合は、Windows 用の curl を検索およびダウンロードして、類似したコマンドを実行できます。

サーバーで実行するアプリケーションのレイテンシーを測定および評価する場合は、次の点に留意します。

- レイテンシーの値は、各アプリケーションに対して相対的です。ただし、先頭バイトまでの時間は、秒単位またはそれ以上ではなく、ミリ秒単位が合理的です。
- 通常の負荷でアプリケーションのレイテンシーを測定して問題がなくても、高負荷がかかった場合に、ビューワーにタイムアウトが発生する可能性があることに注意してください。需要が高い場合、サーバーでレスポンスの遅延が発生するか、まったく応答しないことがあります。高負荷に伴うレイテンシーの問題を避けるために、CPU、メモリ、ディスクの読み取りと書き込みなど、サーバーのリソースをチェックし、サーバーが高負荷に合わせてスケールできることを確認します。

次の Linux コマンドを実行して、Apache プロセスによって使用されているメモリを確認できます。

```
watch -n 1 "echo -n 'Apache Processes: ' && ps -C apache2 --no-headers | wc -l && free -m"
```

- サーバーでの高い CPU 使用率により、アプリケーションのパフォーマンスが大幅に低下する場合があります。バックエンドサーバーに Amazon EC2 インスタンスを使用する場合は、サーバーのメトリクスを確認して CloudWatch CPU 使用率を確認します。詳細については、「[Amazon ユーザーガイド CloudWatch](#)」を参照してください。または、独自のサーバーを使用している場合は、CPU 使用率を確認する方法について、サーバーのヘルプドキュメントを参照してください。
- リクエストの量が多くてデータベースクエリが遅くなるなど、高負荷に伴って発生する可能性がある他の問題を確認します。

リソースの追加およびサーバーとデータベースのチューニング

アプリケーションとサーバーの応答性を評価したら、一般的なトラフィックと高負荷の状況に対する十分なリソースがあることを確認します。

- 独自のサーバーがある場合は、評価に基づいて、ビューワーリクエストを処理する十分な CPU、メモリ、およびディスクスペースがあることを確認します。
- Amazon EC2 インスタンスをバックエンドサーバーとして使用している場合は、受信されるリクエストを満たす適切なリソースがインスタンスタイプにあることを確認します。詳細については、Amazon EC2 ユーザーガイドの「[インスタンスタイプ](#)」を参照してください。

さらに、タイムアウトを避けるために次のチューニングステップを検討します。

- curl コマンドによって返される先頭バイトまでの時間の値が高いと思われる場合は、アプリケーションのパフォーマンスを向上させるステップを実行します。アプリケーションの応答性の向上は、タイムアウトエラーを減らすうえで有効です。
- データベースクエリをチューニングし、パフォーマンスを低下させることなく高いリクエストボリュームを処理できるようにします。
- バックエンドサーバーで[キープアライブ \(持続的\)](#) 接続を設定します。このオプションは、それ以降のリクエストまたはユーザーに対して接続を再確立する必要があるときに発生するレイテンシーを回避するために有効です。
- ELB をオリジンとして使用している場合、レイテンシーを減らす方法については、ナレッジセンター記事「[ELB Classic Load Balancer 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](#)」の推奨事項を参照してください。

必要に応じて、CloudFront タイムアウト値を調整します。

アプリケーションの低いパフォーマンス、オリジンサーバーの容量、およびその他の問題について評価、対応したが、それでもビューワーに HTTP 504 エラーが発生する場合は、オリジン応答タイムアウトに対してディストリビューションで指定されている時間の変更を検討します。詳細については、「[the section called “応答タイムアウト \(カスタムオリジンのみ\)”](#)」を参照してください。

負荷テスト CloudFront

は、DNS CloudFront を使用して地理的に分散したエッジロケーション間および各エッジロケーション内の負荷を分散 CloudFront するため、従来の負荷テスト方法は ではなく機能しません。クライアントが にコンテンツをリクエストすると CloudFront、クライアントは一連の IP アドレスを含む DNS レスポンスを受け取ります。DNS が返す IP アドレスの 1 つだけにリクエストを送信してテストする場合、実際のトラフィックパターンを正確に表さない 1 つの CloudFront エッジロケーションにあるリソースの小さなサブセットのみをテストします。リクエストされるデータの量によっては、

この方法でテストすると、サーバーのその小さなサブセットのパフォーマンスが過負荷になり、パフォーマンスが低下する可能性があります CloudFront。

CloudFront は、複数の地理的リージョンで異なるクライアント IP アドレスと異なる DNS リゾルバーを持つビューワー向けにスケールするように設計されています。CloudFront パフォーマンスを正確に評価する負荷テストを実行するには、次のすべてを実行することをお勧めします。

- 複数の地理的リージョンからクライアントのリクエストを送信します。
- 各クライアントで独立した DNS リクエストを実行するようにテストを構成します。これで、各クライアントは DNS から一連の異なる IP アドレスを受け取ります。
- リクエストを行うクライアントごとに、DNS から返される一連の IP アドレスにクライアントリクエストを分散します。これにより、ロードが CloudFront エッジロケーション内の複数のサーバーに分散されます。

負荷テスト では、次の制限に注意してください CloudFront。

- Lambda @Edge [ビューアリクエストまたはビューアレスポンストリガー](#)があるキャッシュヒエラルキーでは、ロードテストはできません。
- ロードテストは、[Origin Shield](#)が有効であるオリジンではできません。

リクエストとレスポンスの動作

以下のセクションでは、`GET` ビューワーリクエスト CloudFront を処理して Amazon S3 またはカスタムオリジンに転送する方法と、`4xx` および `5xx` HTTP ステータスコード CloudFront を処理してキャッシュする方法など、オリジンからのレスポンス CloudFront を処理する方法について説明します。

トピック

- [Amazon S3 オリジンに対するリクエストとレスポンスの動作](#)
- [「カスタムオリジンのリクエストとレスポンスの動作」](#)
- [オリジングループに対するリクエストとレスポンスの動作](#)
- [オリジンリクエストへのカスタムヘッダーの追加](#)
- [オブジェクトの部分リクエスト CloudFront を処理する方法 \(範囲 GETs\)](#)
- [オリジンからの HTTP 3xx ステータスコード CloudFront を処理する方法](#)
- [オリジンからの HTTP 4xx および 5xx ステータスコード CloudFront を処理してキャッシュする方法](#)

Amazon S3 オリジンに対するリクエストとレスポンスの動作

トピック

- [GET HTTP および HTTPS リクエスト CloudFront を処理する方法](#)
- [GET リクエスト CloudFront を処理して Amazon S3 オリジンに転送する方法](#)
- [GET Amazon S3 オリジンからのレスポンス CloudFront を処理する方法](#)

GET HTTP および HTTPS リクエスト CloudFront を処理する方法

Amazon S3 オリジンの場合、`GET` はデフォルトでデイス CloudFront トリビューション内のオブジェクトに対する HTTP プロトコルと HTTPS プロトコルの両方のリクエスト CloudFront を受け入れます。CloudFront 次に、`GET` は、リクエストが行われたのと同じプロトコルを使用して Amazon S3 バケットにリクエストを転送します。

カスタムオリジンの場合、デイス トリビューションを作成するときに、`GET` オリジン CloudFront にアクセスする方法を指定できます。HTTP のみ、またはビューワーが使用するプロトコルと一致させる

ことができます。がカスタムオリジンの HTTP および HTTPS リクエスト CloudFront を処理する方法の詳細については、「」を参照してください[プロトコル](#)。

エンドユーザーが HTTPS を使用してのみオブジェクトにアクセスできるようにディストリビューションを制限する方法については、「[で HTTPS を使用する CloudFront](#)」を参照してください。

Note

HTTPS リクエストの料金は HTTP リクエストの料金よりも高くなります。請求料率の詳細については、[CloudFront 料金表体系](#)を参照してください。

がリクエスト CloudFront を処理して Amazon S3 オリジンに転送する方法

このトピックには、がビューワーリクエスト CloudFront を処理し、リクエストを Amazon S3 オリジンに転送する方法に関する情報が含まれています。

トピック

- [キャッシュ期間および最小 TTL](#)
- [クライアント IP アドレス](#)
- [条件付きの GET](#)
- [cookie](#)
- [クロスオリジンリソース共有 \(CORS\)](#)
- [本文が含まれている GET リクエスト](#)
- [HTTP メソッド](#)
- [CloudFront 削除または更新する HTTP リクエストヘッダー](#)
- [リクエストの最大長と URL の最大長](#)
- [OCSP Stapling](#)
- [プロトコル](#)
- [クエリ文字列](#)
- [オリジン接続のタイムアウトと試行](#)
- [オリジン応答タイムアウト](#)
- [同一オブジェクトへの同時リクエスト \(リクエストを折りたたむ\)](#)

キャッシュ期間および最小 TTL

がオリジンに別のリクエスト CloudFront を転送する前にオブジェクトが CloudFront キャッシュに保持される期間を制御するには、次の操作を行います。

- Cache-Control または Expires ヘッダーフィールドを各オブジェクトに追加するようにオリジンを構成します。
- CloudFront キャッシュ動作の最小 TTL の値を指定します。
- デフォルト値の 24 時間を使用します。

詳細については、「[コンテンツがキャッシュに保持される期間 \(有効期限\) の管理](#)」を参照してください。

クライアント IP アドレス

ビューワーが にリクエストを送信 CloudFront し、 X-Forwarded-For リクエストヘッダーを含まない場合、 は TCP 接続からビューワーの IP アドレス CloudFront を取得し、 IP アドレスを含む X-Forwarded-For ヘッダーを追加して、リクエストをオリジンに転送します。例えば、 が TCP 接続 192.0.2.2 から IP アドレス CloudFront を取得する場合、次のヘッダーをオリジンに転送します。

```
X-Forwarded-For: 192.0.2.2
```

ビューワーが にリクエストを送信 CloudFront し、 X-Forwarded-For リクエストヘッダーが含まれている場合、 は TCP 接続からビューワーの IP アドレス CloudFront を取得し、 X-Forwarded-For ヘッダーの末尾に追加して、リクエストをオリジンに転送します。例えば、ビューワーリクエストに が含まれ X-Forwarded-For: 192.0.2.4,192.0.2.3、 TCP 接続 192.0.2.2 から IP アドレス CloudFront を取得する場合、次のヘッダーをオリジンに転送します。

```
X-Forwarded-For: 192.0.2.4,192.0.2.3,192.0.2.2
```

Note

X-Forwarded-For ヘッダーには、IPv4 アドレス (192.0.2.44 など) および IPv6 アドレス (2001:0db8:85a3::8a2e:0370:7334 など) が含まれます。

条件付きの GET

は、エッジキャッシュから有効期限切れになっているオブジェクトのリクエスト CloudFront を受け取ると、リクエストを Amazon S3 オリジンに転送してオブジェクトの最新バージョンを取得するか、CloudFront エッジキャッシュに最新バージョンがすでに存在することを Amazon S3 に確認します。Amazon S3 が最初にオブジェクトを に送信したとき CloudFront、レスポンスに ETag 値と LastModified 値が含まれていました。が Amazon S3 CloudFront に転送する新しいリクエストでは、 は次のいずれかまたは両方 CloudFront を追加します。

- オブジェクトの有効期限切れバージョンの If-Match 値が含まれる If-None-Match または ETag ヘッダー。
- オブジェクトの有効期限切れバージョンの If-Modified-Since 値が含まれる LastModified ヘッダー。

Amazon S3 はこの情報を使用して、オブジェクトが更新されたかどうか、つまりオブジェクト全体を に返すか、HTTP 304 ステータスコードのみ (変更なし) CloudFront を返すかを決定します。

cookie

Amazon S3 は Cookie を処理しません。Cookie を Amazon S3 オリジンに転送するようにキャッシュ動作を設定した場合、CloudFront は Cookie を転送しますが、Amazon S3 はそれらを無視します。同じオブジェクトに対する今後すべてのリクエストは、Cookie を変更するかどうかに関わらず、キャッシュ内の既存のオブジェクトから処理されます。

クロスオリジンリソース共有 (CORS)

Amazon S3 の Cross-Origin Resource Sharing 設定を優先 CloudFront する場合は、選択したヘッダー CloudFront を Amazon S3 に転送するように を設定します。詳細については、「[リクエストヘッダーに基づくコンテンツのキャッシュ](#)」を参照してください。

本文が含まれている GET リクエスト

ビューワーGETリクエストに本文が含まれている場合、 は HTTP ステータスコード 403 (禁止) をビューワーに CloudFront 返します。

HTTP メソッド

サポートしているすべての HTTP メソッドを処理する CloudFront ように を設定した場合、CloudFront はビューワーからの次のリクエストを受け入れ、Amazon S3 オリジンに転送します。

- DELETE
- GET
- HEAD
- OPTIONS
- PATCH
- POST
- PUT

CloudFront は、GET および HEAD リクエストへのレスポンスを常にキャッシュします。また、OPTIONS requests. CloudFront does へのレスポンスをキャッシュ CloudFront するようにを設定することもできます。他のメソッドを使用するリクエストへのレスポンスはキャッシュしません。

マルチパートアップロードを使用して Amazon S3 バケットにオブジェクトを追加する場合は、オ CloudFront リジンアクセスコントロール (OAC) をディストリビューションに追加し、OAC に必要なアクセス許可を付与する必要があります。詳細については、「[the section called “Amazon S3 オリジンへのアクセスの制限”](#)」を参照してください。

Important

がサポートする CloudFront すべての HTTP メソッドを受け入れて Amazon S3 に転送 CloudFront するようにを設定する場合は、オ CloudFront リジンアクセスコントロール (OAC) を作成して Amazon S3 コンテンツへのアクセスを制限し、OAC に必要なアクセス許可を付与する必要があります。例えば、を使用するためにこれらのメソッドを受け入れて転送 CloudFront するようにを設定する場合 PUT、ビューワーが不要なリソースを削除できないように、DELETE リクエストを適切に処理するように Amazon S3 バケットポリシーを設定する必要があります。詳細については、「[the section called “Amazon S3 オリジンへのアクセスの制限”](#)」を参照してください。

Amazon S3 がサポートする操作の詳細については、「[Amazon S3 ドキュメント](#)」を参照してください。

CloudFront 削除または更新する HTTP リクエストヘッダー

CloudFront は、Amazon S3 オリジンにリクエストを転送する前に、一部のヘッダーを削除または更新します。ほとんどのヘッダーで、この動作はカスタムオリジンの場合と同じです。HTTP リクエ

トヘッダーの完全なリストと、ヘッダー CloudFront の処理方法については、「」を参照してください[HTTP リクエストヘッダーと CloudFront 動作 \(カスタムオリジンと Amazon S3 オリジン\)](#)。

リクエストの最大長と URL の最大長

パス、クエリ文字列 (ある場合)、ヘッダーを含め、リクエストの最大長は 20480 バイトです。

CloudFront はリクエストから URL を作成します。この URL の最大長は 8192 文字です。

リクエストまたは URL がこれらの最大値を超えると、は HTTP ステータスコード 413、リクエスト エンティティが大きすぎるをビューワーに CloudFront 返し、ビューワーへの TCP 接続を終了します。

OCSP Stapling

ビューワーがオブジェクトの HTTPS リクエストを送信すると、CloudFront またはビューワーは、ドメインの SSL 証明書が取り消されていないことを認証機関 (CA) に確認する必要があります。OCSP Stapling は、CloudFront が証明書を検証し、CA からのレスポンスをキャッシュできるようにすることで、証明書の検証を高速化するため、クライアントは CA で証明書を直接検証する必要はありません。

同ドメイン内のオブジェクトに対する多数の HTTPS リクエストを が CloudFront 受信すると、OCSP stapling のパフォーマンス向上がより顕著になります。CloudFront エッジロケーション内の各サーバーは、個別の検証リクエストを送信する必要があります。が同じドメインに対して多数の HTTPS リクエスト CloudFront を受信すると、エッジロケーション内のすべてのサーバーは、SSL ハンドシェイクのパケットに「ステーブル」できるという CA からの応答をすぐに受け取ります。証明書が有効であることがビューワーに確認されると、リクエストされたオブジェクトを提供 CloudFront できます。デイストリビューションが CloudFront エッジロケーションで大量のトラフィックを受信しない場合、新しいリクエストは CA で証明書をまだ検証していないサーバーに誘導される可能性が高くなります。その場合、ビューワーは個別に検証ステップを実行し、CloudFront サーバーはオブジェクトを提供します。この CloudFront サーバーも CA に検証リクエストを送信するため、同じドメイン名が含まれるリクエストを次に受信したときには、CA からの検証応答が既に存在しているということになります。

プロトコル

CloudFront は、ビューワーリクエストのプロトコルに基づいて、HTTP または HTTPS リクエストをオリジンサーバーに転送します。

⚠ Important

Amazon S3 バケットがウェブサイトエンドポイントとして設定されている場合、HTTPS を使用してオリジンと通信するようにを設定 CloudFrontすることはできません。Amazon S3 は、その設定で HTTPS 接続をサポートしていないためです。

クエリ文字列

がクエリ文字列パラメータを Amazon S3 オリジン CloudFront に転送するかどうかを設定できます。詳細については、「[クエリ文字列パラメータに基づくコンテンツのキャッシュ](#)」を参照してください。

オリジン接続のタイムアウトと試行

オリジン接続タイムアウトは、オリジンへの接続を確立しようとしたときに が CloudFront 待機する秒数です。

オリジン接続の試行回数は、 がオリジンへの接続を CloudFront 試行する回数です。

これらの設定は、セカンダリオリジンにフェイルオーバーするか (オリジングループの場合)、ビューワーにエラーレスポンスを返すまでに、オリジンへの接続を CloudFront 試行する時間を決定します。デフォルトでは、 はセカンダリオリジンに接続しようとしたり、エラーレスポンスを返したりする前に、30 秒 (それぞれ 10 秒間の試行が 3 回) CloudFront 待機します。接続タイムアウトを短くするか、試行回数を減らすか、その両方を行うことで、この時間を短縮できます。

詳細については、「[オリジンのタイムアウトと試行の制御](#)」を参照してください。

オリジン応答タイムアウト

オリジン応答タイムアウト (オリジンの読み取りタイムアウトまたはオリジンリクエストタイムアウトとも呼ばれます) は、次の両方に適用されます。

- リクエストをオリジンに転送した後、 がレスポンスを CloudFront 待機する秒単位の時間。
- がオリジンからレスポンスの packets を受信した後、次の packets を受信するまでに CloudFront 待機する秒単位の時間。

CloudFront 動作は、ビューワーリクエストの HTTP メソッドによって異なります。

- GET および HEAD リクエスト – オリジンが 30 秒以内に応答しない場合、または 30 秒間応答しない場合、は接続を CloudFront ドロップします。指定された数の [オリジン接続の試行](#) が 1 回を超える場合、CloudFront は完全なレスポンスの取得を再試行します。は、オリジン接続 CloudFront の試行回数設定の値によって決まるように、最大 3 回試行します。オリジンが最後の試行中に応答しない場合、CloudFront 同じオリジンのコンテンツに対する別のリクエストを受信するまで、は再試行しません。
- DELETE、OPTIONS、PATCH、PUT および POST リクエスト – オリジンが 30 秒以内に応答しない場合、は接続を CloudFront ドロップし、オリジンへの接続を再試行しません。クライアントは、必要に応じてリクエストを再送信できます。

Amazon S3 オリジン (静的ウェブサイトホスティングで設定されていない S3 バケット) の応答タイムアウトを変更することはできません。

同一オブジェクトへの同時リクエスト (リクエストを折りたたむ)

CloudFront エッジロケーションがオブジェクトのリクエストを受け取り、オブジェクトがキャッシュにないか、キャッシュされたオブジェクトの有効期限が切れている場合、CloudFront はすぐにリクエストをオリジンに送信します。ただし、同じオブジェクトに対する同時リクエストがある場合、つまり、同じオブジェクトに対する追加のリクエスト (同じキャッシュキーを持つ) が、が最初のリクエストに対するレスポンス CloudFront を受信する前にエッジロケーションに到着した場合、は追加のリクエストをオリジンに転送する前に CloudFront 一時停止します。この短い一時停止は、オリジンの負荷を軽減するために役立ちます。は、一時停止中に受信したすべてのリクエストに、元のリクエストからのレスポンス CloudFront を送信します。これはリクエストの折りたたみと呼ばれます。CloudFront ログでは、最初のリクエストは `x-edge-result-type` フィールド `Miss` として識別され、折りたたまれたリクエストは `Hit` として識別されます。CloudFront ログの詳細については、「」を参照してください [the section called “CloudFront およびエッジ関数のログ記録”](#)。

CloudFront は、[キャッシュキー](#) を共有するリクエストのみを折りたたみます。例えば、リクエストヘッダー、Cookie、またはクエリ文字列に基づいてキャッシュ CloudFront するようにを設定しているため、追加のリクエストが同じキャッシュキーを共有しない場合、オリジンに対して一意のキャッシュキーを持つすべてのリクエスト CloudFront が転送されます。

すべてのリクエストが折りたたまれないようにするには、マネージドキャッシュポリシーを使用できます。これにより `CachingDisabled`、キャッシュも防止されます。詳細については、「[管理キャッシュポリシーの使用](#)」を参照してください。

特定のオブジェクトでのリクエストの折りたたみを防ぐ場合は、キャッシュ動作の最小 TTL を 0 に設定し、さらに `Cache-Control: private`、`Cache-Control: no-store`、`Cache-Control:`

no-cache、Cache-Control: max-age=0、または Cache-Control: s-maxage=0 を送信するようにオリジンを設定できます。これらの設定により、オリジンの負荷が増加し、最初のリクエストへの応答を CloudFront 待機している間に一時停止される同時リクエストのレイテンシーが増加します。

が Amazon S3 オリジンからのレスポンス CloudFront を処理する方法

このトピックでは、が Amazon S3 オリジンからのレスポンス CloudFront を処理する方法について説明します。

トピック

- [取り消されたリクエスト](#)
- [削除または更新する CloudFront HTTP レスポンスヘッダー](#)
- [キャッシュ可能なファイルの最大サイズ](#)
- [リダイレクト](#)

取り消されたリクエスト

オブジェクトがエッジキャッシュになく、ビューワーがオリジンからオブジェクト CloudFront を取得した後、リクエストされたオブジェクトを配信する前にセッションを終了する (ブラウザを閉じるなど) 場合、CloudFront はエッジロケーションにオブジェクトをキャッシュしません。

削除または更新する CloudFront HTTP レスポンスヘッダー

CloudFront は、Amazon S3 オリジンからビューワーにレスポンスを転送する前に、次のヘッダーフィールドを削除または更新します。

- X-Amz-Id-2
- X-Amz-Request-Id
- Set-Cookie – Cookie を転送する CloudFront ように を設定すると、Set-Cookieヘッダーフィールドがクライアントに転送されます。詳細については、「[Cookie に基づくコンテンツのキャッシュ](#)」を参照してください。
- Trailer
- Transfer-Encoding – Amazon S3 オリジンがこのヘッダーフィールドを返す場合、はビューワーにレスポンスを返す chunked 前に値を CloudFront に設定します。
- Upgrade

- Via - ビューワーへのレスポンスで、値を次のように CloudFront 設定します。

Via: *http-version alphanumeric-string*.cloudfront.net (CloudFront)

たとえば、クライアントが HTTP/1.1 を介してリクエストを行った場合、値は次のようになります。

Via: 1.1 1026589cc7887e7a0dc7827b4example.cloudfront.net (CloudFront)

キャッシュ可能なファイルの最大サイズ

がキャッシュに CloudFront 保存するレスポンス本文の最大サイズは 50 GB です。これには、Content-Length ヘッダーの値を指定しないチャンク転送レスポンスが含まれます。

CloudFront を使用して、範囲リクエストを使用して各 50 GB 以下のパートのオブジェクトをリクエストすることで、このサイズより大きいオブジェクトをキャッシュできます。CloudFront は、各パートが 50 GB 以下のため、これらのパートをキャッシュします。ビューワーによって、オブジェクトのすべてのパートを取得した後、元の大きなオブジェクトが再構築されます。詳しくは、「[範囲リクエストを使用して大きなオブジェクトをキャッシュする](#)」を参照してください。

リダイレクト

すべてのリクエストを別のホスト名にリダイレクトするように Amazon S3 バケットを構成できます。別のホスト名には、別の Amazon S3 バケットまたは HTTP サーバーを使用できます。すべてのリクエストをリダイレクトするようにバケットを設定し、バケットがディストリビューションの CloudFront オリジンである場合は、ディストリビューションのドメイン名 (d1111111abcdef8.cloudfront.net など) または CloudFront ディストリビューションに関連付けられた代替ドメイン名 (CNAME) を使用して、すべてのリクエストをディストリビューションにリダイレクトするようにバケットを設定することをお勧めします example.com。それ以外の場合、ビューワーリクエストはをバイパスし CloudFront、オブジェクトは新しいオリジンから直接提供されます。

Note

代替ドメイン名にリクエストをリダイレクトする場合は、CNAME レコードを追加してドメインの DNS サービスを更新する必要があります。詳細については、「[代替ドメイン名 \(CNAME\) を追加することによるカスタム URL の使用](#)」を参照してください。

すべてのリクエストをリダイレクトするようにバケットを構成した場合の動作を以下に示します。

1. ビューワー (ブラウザなど) が オブジェクトをリクエストします CloudFront。
2. CloudFront は、ディストリビューションのオリジンである Amazon S3 バケットにリクエストを転送します。
3. Amazon S3 は、HTTP ステータスコード 301 (Moved Permanently) と新しい場所を返します。
4. CloudFront はリダイレクトステータスコードと新しい場所をキャッシュし、その値を viewer. CloudFront does に返します。新しい場所からオブジェクトを取得するためにリダイレクトに従わないでください。
5. ビューワーはオブジェクトに対して別のリクエストを送信しますが、今回はビューワーが から取得した新しい場所を指定します CloudFront。
 - Amazon S3 バケットが、ディストリビューションのドメイン名または代替ドメイン名を使用して、すべてのリクエストをディストリビューションに CloudFrontリダイレクトしている場合、 は新しい場所にある Amazon S3 バケットまたは HTTP サーバーからオブジェクトを CloudFront リクエストします。新しい場所がオブジェクトを返すと、 はそれをビューワーに CloudFront 返し、エッジロケーションにキャッシュします。
 - Amazon S3 バケットがリクエストを別の場所にリダイレクトしている場合、2 番目のリクエストは をバイパスします CloudFront。新しい場所の Amazon S3 バケットまたは HTTP サーバーはオブジェクトをビューワーに直接返すため、オブジェクトが CloudFront エッジキャッシュにキャッシュされることはありません。

「カスタムオリジンのリクエストとレスポンスの動作」

トピック

- [がリクエスト CloudFront を処理してカスタムオリジンに転送する方法](#)
- [がカスタムオリジンからのレスポンス CloudFront を処理する方法](#)

がリクエスト CloudFront を処理してカスタムオリジンに転送する方法

このトピックでは、 がビューワーリクエスト CloudFront を処理し、カスタムオリジンにリクエストを転送する方法について説明します。

トピック

- [認証](#)
- [キャッシュ期間および最小 TTL](#)

- [クライアント IP アドレス](#)
- [クライアント側の SSL 認証](#)
- [圧縮](#)
- [条件付きリクエスト](#)
- [cookie](#)
- [クロスオリジンリソース共有 \(CORS\)](#)
- [暗号化](#)
- [本文を含む GET リクエスト](#)
- [HTTP メソッド](#)
- [HTTP リクエストヘッダーと CloudFront 動作 \(カスタムオリジンと Amazon S3 オリジン\)](#)
- [HTTP バージョン](#)
- [リクエストの最大長と URL の最大長](#)
- [OCSP Stapling](#)
- [持続的接続](#)
- [プロトコル](#)
- [クエリ文字列](#)
- [オリジン接続のタイムアウトと試行](#)
- [オリジン応答タイムアウト](#)
- [同一オブジェクトへの同時リクエスト \(リクエストを折りたたむ\)](#)
- [User-Agent ヘッダー](#)

認証

DELETE、GET、HEAD、POST、および PUT リクエストの場合、ヘッダーをオリジンに転送するように PATCH を設定すると、クライアント認証をリクエストするようにオリジンサーバーを設定できます。 [CloudFront Authorization](#)

OPTIONS リクエストの場合、次の CloudFront 設定を使用する場合にのみ、クライアント認証をリクエストするようにオリジンサーバーを設定できます。

- [Authorization ヘッダーをオリジン CloudFront に転送するように](#) を設定します。
- OPTIONS リクエストへの応答をキャッシュ CloudFront しないように を設定します。

HTTP または HTTPS を使用してオリジンにリクエストを転送する CloudFront ように を設定できます。詳細については、「」を参照してください [で HTTPS を使用する CloudFront](#)。

キャッシュ期間および最小 TTL

がオリジンに別のリクエスト CloudFront を転送する前にオブジェクトが CloudFront キャッシュに保持される期間を制御するには、次の操作を行います。

- Cache-Control または Expires ヘッダーフィールドを各オブジェクトに追加するようにオリジンを構成します。
- CloudFront キャッシュ動作の最小 TTL の値を指定します。
- デフォルト値の 24 時間を使用します。

詳細については、「[コンテンツがキャッシュに保持される期間 \(有効期限\) の管理](#)」を参照してください。

クライアント IP アドレス

ビューワーが にリクエストを送信 CloudFront し、 X-Forwarded-For リクエストヘッダーを含まない場合、 は TCP 接続からビューワーの IP アドレス CloudFront を取得し、IP アドレスを含む X-Forwarded-For ヘッダーを追加して、リクエストをオリジンに転送します。例えば、 が TCP 接続 192.0.2.2 から IP アドレス CloudFront を取得する場合、次のヘッダーをオリジンに転送します。

```
X-Forwarded-For: 192.0.2.2
```

ビューワーが にリクエストを送信 CloudFront し、 X-Forwarded-For リクエストヘッダーが含まれている場合、 は TCP 接続からビューワーの IP アドレス CloudFront を取得し、X-Forwarded-For ヘッダーの末尾に追加して、リクエストをオリジンに転送します。例えば、ビューワーリクエストに が含まれ X-Forwarded-For: 192.0.2.4,192.0.2.3、TCP 接続 192.0.2.2 から IP アドレス CloudFront を取得する場合、次のヘッダーをオリジンに転送します。

```
X-Forwarded-For: 192.0.2.4,192.0.2.3,192.0.2.2
```

ロードバランサー (Elastic Load Balancing を含む)、ウェブアプリケーションファイアウォール、リバースプロキシ、侵入防止システム、API Gateway などの一部のアプリケーションは、リクエストを X-Forwarded-For ヘッダーの最後に転送した CloudFront エッジサーバーの IP アドレスを追加します。例えば、ELB に転送するリクエスト X-Forwarded-For: 192.0.2.2 に CloudFront が含ま

れ、CloudFront エッジサーバーの IP アドレスが 192.0.2.199 の場合、EC2 インスタンスが受信するリクエストには次のヘッダーが含まれます。

```
X-Forwarded-For: 192.0.2.2,192.0.2.199
```

Note

X-Forwarded-For ヘッダーには、IPv4 アドレス (192.0.2.44 など) および IPv6 アドレス (2001:0db8:85a3::8a2e:0370:7334 など) が含まれます。

また、X-Forwarded-Forヘッダーは、現在のサーバー () へのパス上のすべてのノードによって変更される可能性があることに注意してくださいCloudFront。詳細については、[RFC 7239](#) のセクション 8.1 を参照してください。CloudFront エッジコンピューティング関数を使用してヘッダーを変更することもできます。

クライアント側の SSL 認証

CloudFront は、クライアント側の SSL 証明書によるクライアント認証をサポートしていません。オリジンがクライアント側の証明書をリクエストした場合、CloudFront はリクエストを削除します。

圧縮

詳しくは、「[圧縮ファイルの供給](#)」を参照してください。

条件付きリクエスト

は、エッジキャッシュから有効期限切れになったオブジェクトのリクエスト CloudFront を受け取ると、リクエストをオリジンに転送してオブジェクトの最新バージョンを取得するか、CloudFront エッジキャッシュに最新バージョンがすでに存在することをオリジンに確認します。通常、オリジンが最後にオブジェクトを に送信したとき CloudFront、レスポンスに ETag値、LastModified値、またはその両方の値が含まれていました。がオリジン CloudFront に転送する新しいリクエストでは、 は次のいずれかまたは両方 CloudFront を追加します。

- オブジェクトの有効期限切れバージョンの If-Match 値が含まれる If-None-Match または ETag ヘッダー。
- オブジェクトの有効期限切れバージョンの If-Modified-Since 値が含まれる LastModified ヘッダー。

オリジンはこの情報を使用して、オブジェクトが更新されたかどうか、つまりオブジェクト全体を返すか、HTTP 304 ステータスコードのみ (変更なし) CloudFront を返すかを決定します。

cookie

Cookie CloudFront をオリジンに転送するようにを設定できます。詳細については、「[Cookie に基づくコンテンツのキャッシュ](#)」を参照してください。

クロスオリジンリソース共有 (CORS)

クロスオリジンリソース共有設定を優先 CloudFront する場合は、Originヘッダーをオリジン CloudFront に転送するようにを設定します。詳細については、「[リクエストヘッダーに基づくコンテンツのキャッシュ](#)」を参照してください。

暗号化

ビューワーが HTTPS を使用して リクエストを送信 CloudFront し、ビューワーが使用するプロトコルを使用してカスタムオリジンにリクエストを転送する CloudFront ように要求できます。詳細については、次のディストリビューション設定を参照してください。

- [ビューワープロトコルポリシー](#)
- [プロトコル \(カスタムオリジンのみ\)](#)

CloudFront は、SSLv3, TLSv1TLSv1.1、および TLSv1.1,TLSv1.2 プロトコルを使用して、HTTPS リクエストをオリジンサーバーに転送します。カスタムオリジンでは、オリジンと通信する際に CloudFront が使用する SSL プロトコルを選択できます。

- CloudFront コンソールを使用している場合は、オリジン SSL プロトコルチェックボックスを使用してプロトコルを選択します。詳細については、「[ディストリビューションの作成](#)」を参照してください。
- CloudFront API を使用している場合は、OriginSslProtocols要素を使用してプロトコルを指定します。詳細については、「Amazon API リファレンス[DistributionConfig](#)」の[OriginSslProtocols](#)「」と「」を参照してください。 CloudFront

オリジンが Amazon S3 バケットの場合、CloudFront は常に TLSv1.2.

⚠ Important

SSL と TLS のその他のバージョンはサポートされていません。

で HTTPS を使用方法の詳細については CloudFront、「」を参照してください [で HTTPS を使用する CloudFront](#)。ビューワーと 間の HTTPS 通信 CloudFront、および CloudFront とオリジン間の HTTPS 通信で が CloudFront サポートする暗号のリストについては、「」を参照してください [ビューワーと の間でサポートされているプロトコルと暗号 CloudFront](#)。

本文を含む GET リクエスト

ビューワーGETリクエストに本文が含まれている場合、 は HTTP ステータスコード 403 (禁止) をビューワーに CloudFront 返します。

HTTP メソッド

サポートしているすべての HTTP メソッドを処理する CloudFront ように を設定した場合、CloudFront はビューワーからの次のリクエストを受け入れ、カスタムオリジンに転送します。

- DELETE
- GET
- HEAD
- OPTIONS
- PATCH
- POST
- PUT

CloudFront は、GET および HEAD リクエストへのレスポンスを常にキャッシュします。また、OPTIONS requests. CloudFront does へのレスポンスをキャッシュ CloudFront するように を設定することもできます。他のメソッドを使用するリクエストへのレスポンスはキャッシュしません。

カスタムオリジンが上記のメソッドを処理するかどうかを構成する方法の詳細については、オリジンのドキュメントを参照してください。

⚠ Important

が CloudFront サポートするすべての HTTP メソッド CloudFront を受け入れてオリジンに転送するようにを設定する場合は、すべてのメソッドを処理するようにオリジンサーバーを設定します。例えば、を使用するためにこれらのメソッドを受け入れて転送 CloudFront するようにを設定する場合 POST、ビューワーが不要なリソースを削除できないように、DELETE リクエストを適切に処理するようにオリジンサーバーを設定する必要があります。詳細については、HTTP サーバーのドキュメントを参照してください。

HTTP リクエストヘッダーと CloudFront 動作 (カスタムオリジンと Amazon S3 オリジン)

次の表は、カスタムオリジンと Amazon S3 オリジンの両方に転送できる HTTP リクエストヘッダーを示しています (例外も注記されています)。この表には、各ヘッダーについて以下に関する情報も含まれています。

- CloudFront ヘッダーをオリジンに転送 CloudFront するようにを設定しない場合の動作。これにより、CloudFront はヘッダー値に基づいてオブジェクトをキャッシュします。
- そのヘッダーのヘッダー値に基づいてオブジェクトをキャッシュ CloudFront するようにを設定できるかどうか。

ヘッダー Date と User-Agent ヘッダーの値に基づいてオブジェクトをキャッシュ CloudFront するようにを設定できますが、お勧めしません。これらのヘッダーには可能な値が多数あり、その値に基づいてキャッシュすると、がオリジン CloudFront に転送するリクエストの数が大幅に増加します。

ヘッダー値に基づくキャッシュの詳細については、「[リクエストヘッダーに基づくコンテンツのキャッシュ](#)」を参照してください。

[Header] (ヘッダー)	ヘッダー値に基づいてキャッシュ CloudFront するようにを設定しない場合の動作	ヘッダー値に基づくキャッシュがサポートされている
-----------------	---	--------------------------

[Header] (ヘッダー)	ヘッダー値に基づいてキャッシュ CloudFront するように を設定しない場合の動作	ヘッダー値に基づくキャッシュがサポートされている
他の定義されたヘッダー	レガシーキャッシュ設定 — CloudFront ヘッダーをオリジンに転送します。	はい
Accept	CloudFront は ヘッダーを削除します。	はい
Accept-Charset	CloudFront は ヘッダーを削除します。	はい
Accept-Encoding	値に gzip または が含まれている場合 br、オリジンに正規化された Accept-Encoding ヘッダーを CloudFront 転送します。 詳細については、「 圧縮のサポート 」および「 圧縮ファイルの供給 」を参照してください。	はい
Accept-Language	CloudFront は ヘッダーを削除します。	はい

[Header] (ヘッダー)	ヘッダー値に基づいてキャッシュ CloudFront するように を設定しない場合の動作	ヘッダー値に基づくキャッシュがサポートされている
Authorization	<ul style="list-style-type: none"> GET および HEAD リクエスト – CloudFront リクエストをオリジンに転送する前に、Authorization ヘッダーフィールドを削除します。 OPTIONS リクエスト – CloudFront リクエストへのレスポンスをキャッシュするように を設定した場合、は OPTIONS リクエストをオリジンに転送する前に CloudFront Authorization ヘッダーフィールドを削除します。 <p>CloudFront OPTIONS リクエストへの応答をキャッシュ CloudFront するように を設定しない場合、は Authorization ヘッダーフィールドをオリジンに転送します。</p> <ul style="list-style-type: none"> DELETE、PATCH、POST および PUT リクエスト – CloudFront リクエストをオリジンに転送する前に、ヘッダーフィールドを削除しません。 	はい
Cache-Control	CloudFront は ヘッダーをオリジンに転送します。	いいえ
CloudFront-Forwarded-Proto	<p>CloudFront は、リクエストをオリジンに転送する前に ヘッダーを追加しません。</p> <p>詳細については、「リクエストのプロトコルに基づいてキャッシュを設定する」を参照してください。</p>	はい

[Header] (ヘッダー)	ヘッダー値に基づいてキャッシュ CloudFront するように を設定しない場合の動作	ヘッダー値に基づくキャッシュがサポートされている
CloudFront-Is-Desktop-Viewer	CloudFront は、リクエストをオリジンに転送する前に ヘッダーを追加しません。 詳細については、「 デバイスタイプに基づいてキャッシュを設定する 」を参照してください。	はい
CloudFront-Is-Mobile-Viewer	CloudFront は、リクエストをオリジンに転送する前に ヘッダーを追加しません。 詳細については、「 デバイスタイプに基づいてキャッシュを設定する 」を参照してください。	はい
CloudFront-Is-Tablet-Viewer	CloudFront は、リクエストをオリジンに転送する前に ヘッダーを追加しません。 詳細については、「 デバイスタイプに基づいてキャッシュを設定する 」を参照してください。	はい
CloudFront-Viewer-Country	CloudFront は、リクエストをオリジンに転送する前に ヘッダーを追加しません。	はい
Connection	CloudFront は、リクエストをオリジンに転送Connection: Keep-Alive する前に、このヘッダーを に置き換えます。	いいえ
Content-Length	CloudFront は ヘッダーをオリジンに転送します。	いいえ
Content-MD5	CloudFront は ヘッダーをオリジンに転送します。	はい

[Header] (ヘッダー)	ヘッダー値に基づいてキャッシュ CloudFront するように を設定しない場合の動作	ヘッダー値に基づくキャッシュがサポートされている
Content-Type	CloudFront は ヘッダーをオリジンに転送します。	はい
Cookie	Cookie を転送する CloudFront ように を設定すると、Cookieヘッダーフィールドがオリジンに転送されます。指定しない場合、 は Cookieヘッダーフィールド CloudFront を削除します。詳細については、「 Cookie に基づくコンテンツのキャッシュ 」を参照してください。	いいえ
Date	CloudFront は ヘッダーをオリジンに転送します。	はい、ただし推奨されません
Expect	CloudFront は ヘッダーを削除します。	はい
From	CloudFront は ヘッダーをオリジンに転送します。	はい
Host	CloudFront は、リクエストされたオブジェクトに関連付けられているオリジンのドメイン名に値を設定します。 Amazon S3 または MediaStore オリジンのホストヘッダーに基づいてキャッシュすることはできません。	はい (カスタム) いいえ (S3 および MediaStore)
If-Match	CloudFront は ヘッダーをオリジンに転送します。	はい

[Header] (ヘッダー)	ヘッダー値に基づいてキャッシュ CloudFront するように を設定しない場合の動作	ヘッダー値に基づくキャッシュがサポートされている
If-Modified-Since	CloudFront は ヘッダーをオリジンに転送します。	はい
If-None-Match	CloudFront は ヘッダーをオリジンに転送します。	はい
If-Range	CloudFront は ヘッダーをオリジンに転送します。	はい
If-Unmodified-Since	CloudFront は ヘッダーをオリジンに転送します。	はい
Max-Forwards	CloudFront は ヘッダーをオリジンに転送します。	いいえ
Origin	CloudFront は ヘッダーをオリジンに転送します。	はい
Pragma	CloudFront は ヘッダーをオリジンに転送します。	いいえ
Proxy-Authenticate	CloudFront は ヘッダーを削除します。	いいえ
Proxy-Authorization	CloudFront は ヘッダーを削除します。	いいえ
Proxy-Connection	CloudFront は ヘッダーを削除します。	いいえ
Range	CloudFront は ヘッダーをオリジンに転送します。詳細については、「 ガオブジェクトの部分リクエスト CloudFront を処理する方法 (範囲 GETs) 」を参照してください。	はい (デフォルト)

[Header] (ヘッダー)	ヘッダー値に基づいてキャッシュ CloudFront するように を設定しない場合の動作	ヘッダー値に基づくキャッシュがサポートされている
Referer	CloudFront は ヘッダーを削除します。	はい
Request-Range	CloudFront は ヘッダーをオリジンに転送します。	いいえ
TE	CloudFront は ヘッダーを削除します。	いいえ
Trailer	CloudFront は ヘッダーを削除します。	いいえ
Transfer-Encoding	CloudFront は ヘッダーをオリジンに転送します。	いいえ
Upgrade	CloudFront は、WebSocket 接続を確立しない限り、ヘッダーを削除します。	いいえ (WebSocket 接続を除く)
User-Agent	CloudFront は、このヘッダーフィールドの値を置き換えますAmazon CloudFront 。ユーザーが使用しているデバイスに基づいてコンテンツを CloudFront キャッシュする場合は、「」を参照してください デバイスタイプに基づいてキャッシュを設定する 。	はい、ただし推奨されません
Via	CloudFront は ヘッダーをオリジンに転送します。	はい
Warning	CloudFront は ヘッダーをオリジンに転送します。	はい

[Header] (ヘッダー)	ヘッダー値に基づいてキャッシュ CloudFront するように を設定しない場合の動作	ヘッダー値に基づくキャッシュがサポートされている
X-Amz-Cf-Id	CloudFront は、リクエストをオリジンに転送する前に、ビューワーリクエストにヘッダーを追加します。ヘッダー値には、リクエストを一意に識別する暗号化された文字列が含まれます。	いいえ
X-Edge-*	CloudFront はすべてのX-Edge-*ヘッダーを削除します。	いいえ
X-Forwarded-For	CloudFront はヘッダーをオリジンに転送します。詳細については、「 クライアント IP アドレス 」を参照してください。	はい
X-Forwarded-Proto	CloudFront はヘッダーを削除します。	いいえ
X-HTTP-Method-Override	CloudFront はヘッダーを削除します。	はい
X-Real-IP	CloudFront はヘッダーを削除します。	いいえ

HTTP バージョン

CloudFront は、HTTP/1.1 を使用してカスタムオリジンにリクエストを転送します。

リクエストの最大長と URL の最大長

パス、クエリ文字列 (ある場合)、ヘッダーを含め、リクエストの最大長は 20480 バイトです。

CloudFront はリクエストから URL を作成します。この URL の最大長は 8192 文字です。

リクエストまたは URL がこれらの最大値を超えると、は HTTP ステータスコード 413、リクエストエンティティが大きすぎるをビューワーに CloudFront 返し、ビューワーへの TCP 接続を終了します。

OCSP Stapling

ビューワーがオブジェクトの HTTPS リクエストを送信すると、CloudFront または ビューワーは、ドメインの SSL 証明書が取り消されていないことを認証機関 (CA) に確認する必要があります。OCSP Stapling は、CloudFront が証明書を検証し、CA からのレスポンスをキャッシュできるようにすることで、証明書の検証を高速化するため、クライアントは CA で証明書を直接検証する必要はありません。

が同じドメイン内のオブジェクトに対する多数の HTTPS リクエスト CloudFront を受信すると、OCSP ステープリングのパフォーマンス向上がより顕著になります。CloudFront エッジロケーション内の各サーバーは、別々の検証リクエストを送信する必要があります。が同じドメインに対して多数の HTTPS リクエスト CloudFront を受信すると、エッジロケーション内のすべてのサーバーは、SSL ハンドシェイクのパケットに「ステープル」できるという CA からの応答をすぐに受け取ります。証明書が有効であることがビューワーに確認されると、リクエストされたオブジェクトを提供 CloudFront できます。CloudFront エッジロケーション内でディストリビューションが十分なトラフィックを確保できない場合、新しいリクエストは、CA に対して証明書がまだ検証されていないサーバーに誘導される可能性が高くなります。その場合、ビューワーは個別に検証ステップを実行し、CloudFront サーバーはオブジェクトを提供します。その CloudFront サーバーも CA に検証リクエストを送信するため、同じドメイン名を含むリクエストを次回受信すると、CA から検証レスポンスが返されます。

持続的接続

がオリジンからレスポンス CloudFront を取得すると、その期間中に別のリクエストが到着した場合に、数秒間接続を維持しようとしています。持続的接続を維持すると、TCP 接続の再構築に必要な時間と後続のリクエストに対する別の TLS ハンドシェイクの実行に必要な時間を節約できます。

持続的接続の期間を設定する方法など、詳細については、「[キープアライブタイムアウト \(カスタムオリジンのみ\)](#)」セクションの「[ディストリビューションを作成または更新する場合に指定する値](#)」を参照してください。

プロトコル

CloudFront は、以下に基づいて HTTP または HTTPS リクエストをオリジンサーバーに転送します。

- ビューワーが に送信するリクエストのプロトコル。HTTP または HTTPS CloudFrontのいずれかです。
- コンソールのオリジンプロトコルポリシーフィールド CloudFrontの値、または CloudFront API を使用している場合は、DistributionConfig複合型の OriginProtocolPolicy要素。CloudFront コンソールでは、オプションは HTTP のみ、HTTPS のみ、一致ビューワー です。

[HTTP Only] または [HTTPS Only] を指定すると、CloudFront では、ビューワーリクエストのプロトコルに関係なく、指定されたプロトコルのみを使用してリクエストがオリジンサーバーに転送されます。

一致ビューワー を指定した場合、ビューワーリクエストのプロトコルを使用してオリジンサーバーにリクエストを CloudFront 転送します。ビューワーが HTTP プロトコルと HTTPS プロトコルの両方を使用してリクエストを行った場合でも、 はオブジェクトを 1 回だけ CloudFront キャッシュすることに注意してください。

Important

が HTTPS プロトコルを使用してリクエストをオリジン CloudFront に転送し、オリジンサーバーが無効な証明書または自己署名証明書を返す場合、 は TCP 接続を CloudFront ドロップします。

CloudFront コンソールを使用してディストリビューションを更新する方法については、「」を参照してください [ディストリビューションの更新](#)。CloudFront API を使用してディストリビューションを更新する方法については、「Amazon CloudFront API リファレンス [UpdateDistribution](#)」の「」を参照してください。

クエリ文字列

がクエリ文字列パラメータをオリジン CloudFront に転送するかどうかを設定できます。詳細については、「[クエリ文字列パラメータに基づくコンテンツのキャッシュ](#)」を参照してください。

オリジン接続のタイムアウトと試行

オリジン接続タイムアウトは、オリジンへの接続を確立しようとしたときに が CloudFront 待機する秒数です。

オリジン接続の試行回数は、 がオリジンへの接続を CloudFront 試行する回数です。

これらの設定は、セカンダリオリジンにフェイルオーバーするか (オリジングループの場合)、ビューワーにエラーレスポンスを返す前に、オリジンへの接続を CloudFront 試行する時間を決定します。デフォルトでは、はセカンダリオリジンに接続しようとしたり、エラーレスポンスを返したりする前に、30 秒 (それぞれ 10 秒間の試行が 3 回) CloudFront 待機します。接続タイムアウトを短くするか、試行回数を減らすか、その両方を行うことで、この時間を短縮できます。

詳細については、「[オリジンのタイムアウトと試行の制御](#)」を参照してください。

オリジン応答タイムアウト

オリジン応答タイムアウト (オリジンの読み取りタイムアウトまたはオリジンリクエストタイムアウトとも呼ばれます) は、次の両方に適用されます。

- リクエストをオリジンに転送した後、レスポンスを CloudFront 待機する秒単位の時間。
- がオリジンからレスポンスの packets を受信した後、次の packets を受信するまでに CloudFront 待機する秒単位の時間。

CloudFront 動作は、ビューワーリクエストの HTTP メソッドによって異なります。

- GET および HEAD リクエスト – 応答タイムアウトの期間内にオリジンが応答しない場合、または応答を停止した場合、は接続を CloudFront ドロップします。指定された数の[オリジン接続の試行](#)が 1 回を超える場合、CloudFront は完全なレスポンスの取得を再試行します。は、オリジン接続 CloudFront の試行回数設定の値によって決まるように、最大 3 回試行します。オリジンが最後の試行中に応答しない場合、CloudFront 同じオリジンのコンテンツに対する別のリクエストを受信するまで、は再試行しません。
- DELETE、OPTIONS、PATCH、PUT および POST リクエスト – オリジンが 30 秒以内に応答しない場合、は接続を CloudFront ドロップし、オリジンへの接続を再試行しません。クライアントは、必要に応じてリクエストを再送信できます。

オリジン応答タイムアウトを設定する方法など、詳細については、「[応答タイムアウト \(カスタムオリジンのみ\)](#)」を参照してください。

同一オブジェクトへの同時リクエスト (リクエストを折りたたむ)

CloudFront エッジロケーションがオブジェクトのリクエストを受け取り、オブジェクトがキャッシュにないか、キャッシュされたオブジェクトの有効期限が切れている場合、CloudFront はすぐにリクエストをオリジンに送信します。ただし、同じオブジェクトに対する同時リクエストがある場合、つまり、同じオブジェクトに対する追加のリクエスト (同じキャッシュキーを持つ) が、が最初

のリクエストに対するレスポンス CloudFront を受信する前にエッジロケーションに到着した場合、は追加のリクエストをオリジンに転送する前に CloudFront 一時停止します。この短い一時停止は、オリジンの負荷を軽減するために役立ちます。は、一時停止中に受信したすべてのリクエストに、元のリクエストからのレスポンス CloudFront を送信します。これはリクエストの折りたたみと呼ばれます。CloudFront ログでは、最初のリクエストは `x-edge-result-type` フィールド `Miss` として識別され、折りたたまれたリクエストは `Hit` として識別されます。CloudFront ログの詳細については、「」を参照してください [the section called “CloudFront およびエッジ関数のログ記録”](#)。

CloudFront は、[キャッシュキー](#) を共有するリクエストのみを折りたたみます。例えば、リクエストヘッダー、Cookie、またはクエリ文字列に基づいてキャッシュ CloudFront するようにを設定しているため、追加のリクエストが同じキャッシュキーを共有しない場合、オリジンに対して一意のキャッシュキーを持つすべてのリクエスト CloudFront が転送されます。

すべてのリクエストが折りたたまれないようにするには、マネージドキャッシュポリシーを使用できます。これにより `CachingDisabled`、キャッシュも防止されます。詳細については、「[管理キャッシュポリシーの使用](#)」を参照してください。

特定のオブジェクトでのリクエストの折りたたみを防ぐ場合は、キャッシュ動作の最小 TTL を 0 に設定し、さらに `Cache-Control: private`、`Cache-Control: no-store`、`Cache-Control: no-cache`、`Cache-Control: max-age=0`、または `Cache-Control: s-maxage=0` を送信するようにオリジンを設定できます。これらの設定により、オリジンの負荷が増加し、が最初のリクエストへの応答を CloudFront 待っている間に一時停止される同時リクエストのレイテンシーが増加します。

User-Agent ヘッダー

ユーザーがコンテンツの表示に使用しているデバイスに基づいてオブジェクトの異なるバージョンを CloudFront キャッシュする場合は、以下のヘッダーの 1 つ以上 CloudFront をカスタムオリジンに転送するようにを設定することをお勧めします。

- `CloudFront-Is-Desktop-Viewer`
- `CloudFront-Is-Mobile-Viewer`
- `CloudFront-Is-SmartTV-Viewer`
- `CloudFront-Is-Tablet-Viewer`

User-Agent ヘッダーの値に基づいて、はリクエストをオリジンに転送 `false` する前に、これらのヘッダーの値を `true` または CloudFront に設定します。デバイスが複数のカテゴリに属する場合は、複数の値が `true` になることがあります。例えば、一部のタブレットデバイスでは、

CloudFront-Is-Mobile-Viewerと CloudFront の両方CloudFront-Is-Tablet-Viewerを に設定することができますtrue。リクエストヘッダーに基づいてキャッシュ CloudFront するようにを設定する方法の詳細については、「」を参照してください[リクエストヘッダーに基づくコンテンツのキャッシュ](#)。

User-Agent ヘッダーの値に基づいてオブジェクトをキャッシュ CloudFront するようにを設定できますが、お勧めしません。User-Agent ヘッダーには可能な値が多数あり、これらの値に基づいてキャッシュすると、 はオリジン CloudFront に転送するリクエストが大幅に多くなります。

User-Agent ヘッダーの値に基づいてオブジェクトをキャッシュ CloudFront するようにを設定しない場合、 はリクエストをオリジンに転送する前に、次の値を持つUser-Agentヘッダー CloudFront を追加します。

User-Agent = Amazon CloudFront

CloudFront は、ビューワーからのリクエストに ヘッダーが含まれているかどうかにかかわらず、このUser-Agentヘッダーを追加します。ビューワーからのリクエストに User-Agentヘッダーが含まれている場合、 はそのヘッダー CloudFront を削除します。

がカスタムオリジンからのレスポンス CloudFront を処理する方法

このトピックでは、 がカスタムオリジンからのレスポンス CloudFront を処理する方法について説明します。

トピック

- [100 Continue 件のレスポンス](#)
- [キャッシュ](#)
- [取り消されたリクエスト](#)
- [コンテンツネゴシエーション](#)
- [cookie](#)
- [中断された TCP 接続](#)
- [CloudFront を削除または置き換える HTTP レスポンスヘッダー](#)
- [キャッシュ可能なファイルの最大サイズ](#)
- [使用できないオリジン](#)
- [リダイレクト](#)
- [Transfer-Encoding ヘッダー](#)

100 Continue 件のレスポンス

オリジンは、複数の 100-Continue レスポンスを送信することはできません CloudFront。最初の 100-Continue レスポンスの後に、は HTTP 200 OK レスポンス CloudFront を想定します。オリジンが最初のレスポンスの後に別の 100-Continue レスポンスを送信すると、CloudFront エラーが返されます。

キャッシュ

- オリジンサーバーが Date および Last-Modified ヘッダーフィールドに有効かつ正確な値を設定していることを確認します。
- CloudFront 通常、はオリジンからのレスポンスの Cache-Control: no-cache ヘッダーを優先します。例外については、「[同一オブジェクトへの同時リクエスト \(リクエストを折りたたむ\)](#)」を参照してください。

取り消されたリクエスト

オブジェクトがエッジキャッシュになく、ビューワーがオリジンからオブジェクト CloudFront を取得した後、リクエストされたオブジェクトを配信する前にセッションを終了する (ブラウザを閉じるなど) 場合、CloudFront はエッジロケーションにオブジェクトをキャッシュしません。

コンテンツネゴシエーション

オリジンがレスポンス Vary: * に戻り、対応するキャッシュ動作の最小 TTL の値が 0 の場合、はオブジェクトを CloudFront キャッシュしますが、オブジェクトに対する後続のすべてのリクエストをオリジンに転送して、キャッシュにオブジェクトの最新バージョンが含まれていることを確認します。CloudFront には、If-None-Match や などの条件付きヘッダーは含まれません If-Modified-Since。その結果、オリジンはすべてのリクエスト CloudFront に応答して オブジェクトを に返します。

オリジンがレスポンス Vary: * で を返し、対応するキャッシュ動作の最小 TTL の値が他の値である場合、は で説明されているように Vary ヘッダー CloudFront を処理します [CloudFront を削除または置き換える HTTP レスポンスヘッダー](#)。

cookie

キャッシュ動作の Cookie を有効にし、オリジンがオブジェクトとともに Cookie を返す場合、はオブジェクトと Cookie の両方を CloudFront キャッシュします。これにより、オブジェクトのキャッ

シユ可能性が低下します。詳細については、「[Cookie に基づくコンテンツのキャッシュ](#)」を参照してください。

中断された TCP 接続

オリジンがオブジェクトを に返している間に CloudFront とオリジン間の TCP 接続が切断された場合 CloudFront、CloudFront オリジンがレスポンスに Content-Lengthヘッダーを含めたかどうかによって動作が異なります。

- Content-Length ヘッダー – CloudFront オリジンからオブジェクトを取得すると、オブジェクトをビューワーに返します。ただし、Content-Length ヘッダーの値がオブジェクトのサイズに一致しない場合、CloudFront はオブジェクトをキャッシュしません。
- Transfer-Encoding: Chunked – オリジンからオブジェクトを取得すると、オブジェクトをビューワーに CloudFront 返します。ただし、チャンクレスポンスが完了しない場合、CloudFront はオブジェクトをキャッシュしません。
- Content-Length ヘッダーなし – オブジェクトをビューワーに CloudFront 返してキャッシュしますが、オブジェクトが完了しない可能性があります。Content-Length ヘッダーがないと、TCP 接続が誤って切断されたか、または目的で切断されたかを特定 CloudFront できません。

オブジェクトの一部がキャッシュ CloudFront されないように Content-Lengthヘッダーを追加するように HTTP サーバーを設定することをお勧めします。

CloudFront を削除または置き換える HTTP レスポンスヘッダー

CloudFront は、オリジンからビューワーにレスポンスを転送する前に、次のヘッダーフィールドを削除または更新します。

- Set-Cookie – Cookie を転送する CloudFront ように を設定すると、Set-Cookieヘッダーフィールドがクライアントに転送されます。詳細については、「[Cookie に基づくコンテンツのキャッシュ](#)」を参照してください。
- Trailer
- Transfer-Encoding – オリジンがこのヘッダーフィールドを返す場合、 はレスポンスをビューワーに返すchunked前に値を CloudFront に設定します。
- Upgrade
- Vary – 次の点に注意してください。
 - デバイス固有のヘッダーのいずれかをオリジン (CloudFront-Is-Desktop-Viewer、CloudFront-Is-Mobile-Viewer、CloudFront-Is-Tablet-Viewer)

CloudFront に転送するようにを設定し CloudFront-Is-SmartTV-Viewer、オリジンが Vary:User-Agent に戻るように設定した場合 CloudFront、CloudFront はビューワー Vary:User-Agent に戻ります。詳細については、「[デバイスタイプに基づいてキャッシュを設定する](#)」を参照してください。

- Vary ヘッダー Cookie に Accept-Encoding または を含めるようにオリジンを設定すると、はビューワーへのレスポンスに値 CloudFront を含めます。
- ヘッダーをオリジンに転送 CloudFront するようにを設定し、Vary ヘッダー (など Vary:Accept-Charset, Accept-Language) CloudFront の にヘッダー名を返すようにオリジンを設定すると、はそれらの値を持つ Vary ヘッダーをビューワーに CloudFront 返します。
- が Vary ヘッダー* の の値 CloudFront を処理する方法については、「」を参照してください [コンテンツネゴシエーション](#)。
- Vary ヘッダーに他の値を含めるようにオリジンを設定すると、はビューワーにレスポンスを返す前に値 CloudFront を削除します。
- Via - ビューワーへのレスポンスで、値を次のように CloudFront 設定します。

Via: *http-version alphanumeric-string*.cloudfront.net (CloudFront)

たとえば、クライアントが HTTP/1.1 を介してリクエストを行った場合、値は次のようになります。

Via: 1.1 1026589cc7887e7a0dc7827b4example.cloudfront.net (CloudFront)

キャッシュ可能なファイルの最大サイズ

がキャッシュに CloudFront 保存するレスポンス本文の最大サイズは 50 GB です。これには、Content-Length ヘッダーの値を指定しないチャンク転送レスポンスが含まれます。

CloudFront を使用して、範囲リクエストを使用して各 50 GB 以下のパートのオブジェクトをリクエストすることで、このサイズより大きいオブジェクトをキャッシュできます。CloudFront は、各パートが 50 GB 以下のため、これらのパートをキャッシュします。ビューワーによって、オブジェクトのすべてのパートを取得した後、元の大きなオブジェクトが再構築されます。詳しくは、「[範囲リクエストを使用して大きなオブジェクトをキャッシュする](#)」を参照してください。

使用できないオリジン

オリジンサーバーが使用できず、エッジキャッシュにあるが有効期限が切れているオブジェクト (例えば、Cache-Control max-age ディレクティブで指定された期間が経過したため) のリクエスト

CloudFront を受け取った場合、CloudFront はオブジェクトの有効期限が切れたバージョンを提供するか、カスタムエラーページを提供します。カスタムエラーページを設定したときの CloudFront の動作の詳細については、「[カスタムエラーページを設定した場合に がエラー CloudFront を処理する方法](#)」を参照してください。

要求頻度の低いオブジェクトが削除され、エッジキャッシュで使用できなくなる場合があります。は、削除されたオブジェクトを提供 CloudFront できません。

リダイレクト

オリジンサーバーでオブジェクトの場所を変更した場合、リクエストを新しい場所へリダイレクトするようにウェブサーバーを構成できます。リダイレクトを設定すると、ビューワーがオブジェクトのリクエストを初めて送信したときに、CloudFront はリクエストをオリジンに送信し、オリジンはリダイレクトで応答します (例: 302 Moved Temporarily)。はリダイレクトを CloudFront キャッシュしてビューワーに返します。CloudFront はリダイレクトに従いません。

リクエストを以下のどちらかの場所へリダイレクトするようにウェブサーバーを構成できます。

- オリジンサーバーのオブジェクトの新しい URL。ビューワーが新しい URL へのリダイレクトに従うと、ビューワーはバイパス CloudFront してオリジンに直接移動します。つまり、オリジンにあるオブジェクトの新しい URL へリクエストをリダイレクトしないことをお勧めします。
- オブジェクトの新しい CloudFront URL。ビューワーが新しい CloudFront URL を含むリクエストを送信すると、はオリジンの新しい場所からオブジェクト CloudFront を取得し、エッジロケーションにキャッシュして、ビューワーにオブジェクトを返します。オブジェクトに対する以降のリクエストはエッジロケーションによって処理されます。これにより、オリジンのオブジェクトを要求するビューワーに関連するレイテンシーと負荷が回避されます。ただし、オブジェクトに対する新しいすべてのリクエストに、CloudFront への 2 つのリクエストに対する料金がかかります。

Transfer-Encoding ヘッダー

CloudFront は、Transfer-Encoding ヘッダー chunked の値のみをサポートします。オリジンが返した場合 Transfer-Encoding: chunked、はエッジロケーションでオブジェクトが受信されるとオブジェクトをクライアントに CloudFront 返し、後続のリクエストではオブジェクトをチャンク形式でキャッシュします。

ビューワーが Range GET リクエストを行い、オリジンが返した場合 Transfer-Encoding: chunked、はリクエストされた範囲ではなくオブジェクト全体をビューワーに CloudFront 返します。

レスポンスのコンテンツ長を事前に決定できない場合は、チャンクエンコーディングを使用することをお勧めします。詳細については、「[中断された TCP 接続](#)」を参照してください。

オリジングループに対するリクエストとレスポンスの動作

オリジングループへのリクエストは、オリジングループとして設定されていないオリジンへのリクエストと同じように機能します。ただし、オリジンフェイルオーバーがある場合を除きます。他のオリジンと同様に、CloudFront がリクエストを受信し、コンテンツがすでにエッジロケーションにキャッシュされている場合、コンテンツはキャッシュからビューワーに配信されます。キャッシュミスがあり、オリジンがオリジングループである場合、ビューワーリクエストはオリジングループ内のプライマリオリジンに転送されます。

プライマリオリジンのリクエストとレスポンスの動作は、オリジングループにないオリジンの場合と同じです。詳細については、「[Amazon S3 オリジンに対するリクエストとレスポンスの動作](#)」および「[カスタムオリジンのリクエストとレスポンスの動作](#)」を参照してください。

以下にプライマリオリジンが特定の HTTP ステータスコードを返す場合のオリジンフェイルオーバーの動作を示します。

- HTTP 2xx ステータスコード (成功): ファイルを CloudFront キャッシュしてビューワーに返します。
- HTTP 3xx ステータスコード (リダイレクト): CloudFront ステータスコードをビューワーに返します。
- HTTP 4xx または 5xx ステータスコード (クライアント/サーバーエラー): 返されたステータスコードがフェイルオーバー用に設定されている場合、はオリジングループのセカンダリオリジンに同じリクエスト CloudFront を送信します。
- HTTP 4xx または 5xx ステータスコード (クライアント/サーバーエラー): 返されたステータスコードがフェイルオーバー用に設定されていない場合、はエラーをビューワーに CloudFront 返します。

CloudFront は、ビューワーリクエストの HTTP メソッドが GET、HEAD または OPTIONS の場合にのみセカンダリオリジンにフェイルオーバーします。ビューワーが別の HTTP メソッド (、など PUT) を送信しても POST フェイルオーバー CloudFront しません。

がセカンダリオリジンにリクエスト CloudFront を送信する場合、レスポンスの動作は、CloudFront オリジングループにないオリジンの場合と同じです。

オリジングループの詳細については、「[CloudFront オリジンフェイルオーバーによる高可用性の最適化](#)」を参照してください。

オリジンリクエストへのカスタムヘッダーの追加

オリジンに送信するリクエストにカスタムヘッダーを追加する CloudFront ように を設定できます。これらのカスタムヘッダーを使用すると、一般的なビューワーリクエストでは得られない情報をオリジンから送信および収集できます。これらのヘッダーは、オリジンごとにカスタマイズすることもできます。は、カスタムオリジンと Amazon S3 オリジンの両方でカスタムヘッダー CloudFront をサポートします。

トピック

- [オリジンのカスタムヘッダーのユースケース](#)
- [オリジンリクエスト CloudFront にカスタムヘッダーを追加するための の設定](#)
- [がオリジンリクエストに追加 CloudFront できないカスタムヘッダー](#)
- [Authorization ヘッダーを転送する CloudFront ための の設定](#)

オリジンのカスタムヘッダーのユースケース

カスタムヘッダーは、次のようなさまざまな用途に使用できます。

からのリクエストの識別 CloudFront

オリジンが から受け取るリクエストを特定できます CloudFront。これは、ユーザーが をバイパスしているかどうかを知りたい場合 CloudFront、または複数の CDN を使用していて、各 CDN から送信されるリクエストに関する情報が必要な場合に役立ちます。

Note

Amazon S3 オリジンを使用して、[Amazon S3 サーバーアクセスログ](#)を有効にした場合、ログにヘッダー情報は含まれません。

特定のディストリビューションから送信されたリクエストの判断

同じオリジンを使用するように複数の CloudFront ディストリビューションを設定する場合は、ディストリビューションごとに異なるカスタムヘッダーを追加できます。その後、オリジンから

のログを使用して、どのリクエストがどの CloudFront デистриビューションから来たのかを判断できます。

Cross-Origin Resource Sharing (CORS) の有効化

一部のビューワーが Cross-Origin Resource Sharing (CORS) をサポートしていない場合は、オリジンに送信するリクエストに `Origin` ヘッダーを常に追加 CloudFront するようにを設定できます。次に、リクエストごとに `Access-Control-Allow-Origin` ヘッダーを返すようにオリジンを設定できます。また、[CORS 設定を優先 CloudFront するようにを設定](#)する必要があります。

コンテンツへのアクセス制御

カスタムヘッダーを使用して、コンテンツへのアクセスを制御できます。によって追加されたカスタムヘッダーが含まれている場合にのみリクエストに応答するようにオリジンを設定することで CloudFront、ユーザーがオリジンでコンテンツを直接バイパス CloudFront してアクセスすることを防止できます。詳細については、「[カスタムオリジン上のファイルへのアクセス制限](#)」を参照してください。

オリジンリクエスト CloudFront にカスタムヘッダーを追加するための の設定

オリジンに送信されるリクエストにカスタムヘッダーを追加するようデистриビューションを設定するには、次のいずれかの方法を使用してオリジン設定を更新します。

- CloudFront コンソール – デистриビューションを作成または更新するときは、オリジンカスタムヘッダー設定でヘッダー名と値を指定します。詳細については、[デистриビューションの作成](#)または[デистриビューションの更新](#)を参照してください。
- CloudFront API – カスタムヘッダーを追加するオリジンごとに、内の `CustomHeaders` フィールドでヘッダー名と値を指定します `Origin`。詳細については、[CreateDistribution](#) または [UpdateDistribution](#) を参照してください。

指定したヘッダー名と値がビューワーリクエストにまだ存在しない場合、はそれらをオリジンリクエスト CloudFront に追加します。ヘッダーが存在する場合、はリクエストをオリジンに転送する前にヘッダー値を CloudFront 上書きします。

オリジンカスタムヘッダーに適用されるクォータについては、「[ヘッダーのクォータ](#)」を参照してください。

がオリジンリクエストに追加 CloudFront できないカスタムヘッダー

オリジンに送信するリクエストに次のヘッダーを追加する CloudFront ように を設定することはできません。

- Cache-Control
- Connection
- Content-Length
- Cookie
- Host
- If-Match
- If-Modified-Since
- If-None-Match
- If-Range
- If-Unmodified-Since
- Max-Forwards
- Pragma
- Proxy-Authorization
- Proxy-Connection
- Range
- Request-Range
- TE
- Trailer
- Transfer-Encoding
- Upgrade
- Via
- X-Amz- で始まるヘッダー
- X-Edge- で始まるヘッダー
- X-Real-IP

Authorization ヘッダーを転送する CloudFront ための の設定

がビューワーリクエストをオリジン CloudFront に転送すると、 は ヘッダーを含むビューワーAuthorizationヘッダーをデフォルトで CloudFront 削除します。オリジンがオリジンリクエストの Authorization ヘッダーを常に受け取るようにするには、次のオプションがあります。

- キャッシュポリシーを使用して、Authorization ヘッダーをキャッシュキーに追加します。キャッシュキー内のすべてのヘッダーは、オリジンリクエストに自動的に含まれます。詳細については、「[キャッシュキーの管理](#)」を参照してください。
- オリジンリクエストポリシーを使用し、すべてのビューワーヘッダーをオリジンに転送します。オリジンリクエストポリシーで Authorizationヘッダーを個別に転送することはできませんが、すべてのビューワーヘッダーを転送すると、ビューワーリクエストに Authorizationヘッダー CloudFront が含まれます。このユースケースでは、マネージドAllViewer と呼ばれるマネージドオリジンリクエストポリシーが CloudFront から提供されます。詳細については、「[管理オリジンリクエストポリシーの使用](#)」を参照してください。

がオブジェクトの部分リクエスト CloudFront を処理する方法 (範囲 GETs)

大きなオブジェクトの場合、ビューワー (ウェブブラウザまたはその他のクライアント) は、複数のGET リクエストを実行し、Range リクエストヘッダーを使用して、小さいパートでオブジェクトをダウンロードできます。Range GET リクエストとも呼ばれるこのバイト範囲のリクエストでは、部分的なダウンロードの効率、および部分的に失敗した転送からの回復の効率が向上します。

がRange GETリクエスト CloudFront を受信すると、リクエストを受信したエッジロケーションのキャッシュをチェックします。そのエッジロケーションのキャッシュにオブジェクト全体またはオブジェクトのリクエストされた部分がすでに含まれている場合、 は要求された範囲をキャッシュから CloudFront 即座に提供します。

キャッシュにリクエストされた範囲が含まれていない場合は、オリジンへのリクエストを CloudFront 転送します。(パフォーマンスを最適化するには、 でクライアントがリクエストした範囲よりも広い範囲をリクエスト CloudFront できます) Range GET。次に実行される処理は、オリジンが Range GET リクエストをサポートするかどうかによって異なります。

- オリジンが**Range GET**リクエストをサポートしている場合： リクエストされた範囲を返します。 はリクエストされた範囲 CloudFront に対応し、今後のリクエストに備えてキャッシュします。(Amazon S3 も多くの HTTP サーバーも Range GET リクエストをサポートしています)。

- オリジンが**Range GET**リクエストをサポートしていない場合：オブジェクト全体を返します。オブジェクト全体を送信し、今後のリクエストに備えてキャッシュしながら、現在のリクエスト CloudFront を処理します。は、オブジェクト全体をエッジキャッシュに CloudFront キャッシュした後、リクエストされた範囲を提供することで新しいRange GETリクエストに応答します。

いずれの場合も、オリジンから最初のバイトが到着するとすぐに、リクエストされた範囲またはオブジェクトをエンドユーザーに提供する CloudFront ようになります。

Note

ビューワーがRange GETリクエストを行い、オリジンが を返した場合Transfer-Encoding: chunked、 はリクエストされた範囲ではなくオブジェクト全体をビューワーに CloudFront 返します。

CloudFront は通常、 Rangeヘッダーの RFC 仕様に従います。ただし、Range ヘッダーが以下の要件に準拠しない場合、CloudFront は、指定された範囲とステータスコード 206 を返す代わりに、完全なオブジェクトと HTTP ステータスコード 200 を返します。

- 範囲は昇順に指定されている必要があります。たとえば、100-200,300-400 は有効で、300-400,100-200 は無効です。
- 範囲は重複できません。たとえば、100-200,150-250 は無効です。
- すべての範囲指定が有効である必要があります。たとえば、範囲の一部に負の値を指定することはできません。

Rangeリクエストヘッダーの詳細については、RFC 7233 の「[範囲リクエスト](#)」または MDN Web ドキュメントの「[範囲](#)」を参照してください。

範囲リクエストを使用して大きなオブジェクトをキャッシュする

キャッシュが有効になっている場合、50 CloudFront GB を超えるオブジェクトを取得またはキャッシュしません。オリジンがこのサイズより大きいことを (Content-Lengthレスポンスヘッダーで) 示すと、 はオリジンへの接続を CloudFront 閉じ、ビューワーにエラーを返します。(キャッシュを無効にすると、このサイズより大きいオブジェクトをオリジンから取得し、ビューワーに渡す CloudFront ことができます。ただし、オブジェクトはキャッシュ CloudFront されません)。

ただし、範囲リクエストでは、CloudFront を使用して、キャッシュ可能な最大ファイルサイズより大きいオブジェクトをキャッシュできます。たとえば、100 GB のオブジェクトを持つオリジンについて考えてみましょう。キャッシュを有効にすると、このサイズのオブジェクトを取得したりキャッシュ CloudFront したりしません。ただし、ビューワーは複数の範囲リクエストを送信することにより、それぞれサイズが 50 GB 未満のパートを使用して、このパートのオブジェクトを取得できます。例えば、ビューワーは、ヘッダー付きのリクエストを送信することで 20 GB のパートのオブジェクトがリクエストでき、Range: bytes=0-21474836480 最初のパート (他のヘッダー付きのリクエスト) を取得して、Range: bytes=21474836481-42949672960 その次のパートを取得、などといったようになります。ビューワーがすべてのパートを受信すると、それらを組み合わせて元の 100 GB のオブジェクトを構築できます。この場合、はオブジェクトの 20 GB の各部分を CloudFront キャッシュし、キャッシュから同じ部分に対する後続のリクエストに応答できます。

がオリジンからの HTTP 3xx ステータスコード CloudFront を処理する方法

が Amazon S3 バケットまたはカスタムオリジンサーバーからオブジェクトを CloudFront リクエストすると、オリジンが HTTP 3xx ステータスコードを返すことがあります。これは、次のいずれかを示しています。

- オブジェクトの URL が変更された (たとえば、ステータスコード 301、302、307、308)
- オブジェクトは、最後に CloudFront リクエストされてから変更されていません (ステータスコード 304)

CloudFront は、デイス CloudFront トリビューション内の設定と response. CloudFront caches 307 および 308 レスポンスのヘッダーに従って 3xx レスポンスをキャッシュします。これは、オリジンからのレスポンスに Cache-Control ヘッダーを含めた場合のみです。詳細については、「[コンテンツがキャッシュに保持される期間 \(有効期限\) の管理](#)」を参照してください。

オリジンがリダイレクトステータスコード (301 や 307 など) を返す CloudFront 場合、ビューワーへの 301 または 307 レスポンスに沿って redirect. CloudFront passes を実行しないでください。新しいリクエストを送信することでリダイレクトをフォローできます。

がオリジンからの HTTP 4xx および 5xx ステータスコード CloudFront を処理してキャッシュする方法

トピック

- [カスタムエラーページを設定した場合に がエラー CloudFront を処理する方法](#)
- [カスタムエラーページを設定していない場合に がエラー CloudFront を処理する方法](#)
- [キャッシュする CloudFront HTTP 4xx および 5xx ステータスコード](#)

が Amazon S3 バケットまたはカスタムオリジンサーバーからオブジェクトを CloudFront リクエストすると、オリジンが HTTP 4xx または 5xx ステータスコードを返すことがあります。これは、エラーが発生したことを示します。CloudFront 動作は以下によって異なります。

- カスタムエラーページが構成されているかどうか。
- オリジンからのエラーレスポンスをキャッシュ CloudFront する期間 (エラーキャッシュ最小 TTL) を設定しているかどうか。
- ステータスコード。
- 5xx ステータスコードの場合、リクエストされたオブジェクトが現在 CloudFront エッジキャッシュにあるかどうか。
- いくつかの 4xx ステータスコードでは、オリジンが Cache-Control max-age または Cache-Control s-maxage のヘッダーを返すかどうか。

CloudFront は、GET および HEAD リクエストへのレスポンスを常にキャッシュします。また、OPTIONS requests. CloudFront does へのレスポンスをキャッシュ CloudFront するようにを設定することもできます。他のメソッドを使用するリクエストへのレスポンスはキャッシュしません。

オリジンが応答しない場合、オリジンへの CloudFront リクエストはタイムアウトし、オリジンからの HTTP 5xx エラーと見なされます。これは、オリジンがそのエラーで応答しなかった場合でも発生します。このシナリオでは、はキャッシュされたコンテンツの提供 CloudFront を継続します。詳細については、「[使用できないオリジン](#)」を参照してください。

ログ記録を有効にしている場合、は HTTP ステータスコードに関係なく結果をログに CloudFront 書き込みます。

から返されるエラーメッセージに関連する機能とオプションの詳細については CloudFront、以下を参照してください。

- CloudFront コンソールのカスタムエラーページの設定については、「」を参照してください [カスタムエラーページとエラーキャッシュ](#)。
- CloudFront コンソールでのエラーキャッシュ最小 TTL の詳細については、「」を参照してください [Error caching minimum TTL \(seconds\) \(エラーキャッシュ最小 TTL \(秒\)\)](#)。

- が CloudFront キャッシュする HTTP ステータスコードのリストについては、「」を参照してください [キャッシュする CloudFront HTTP 4xx および 5xx ステータスコード](#)。

カスタムエラーページを設定した場合に がエラー CloudFront を処理する方法

カスタムエラーページを設定している場合、リクエストされたオブジェクトがエッジキャッシュにあるかどうかによって CloudFront 動作が異なります。

リクエストされたオブジェクトがエッジキャッシュにない場合

CloudFront 以下がすべて当てはまる場合、 は引き続きオリジンからリクエストされたオブジェクトを取得しようとします。

- ビューワーがオブジェクトを要求する
- オブジェクトがエッジキャッシュにない
- オリジンが HTTP 4xx または 5xx ステータスコードを返して、以下のいずれかに該当する:
 - オリジンがステータスコード 304 (変更なし) またはオブジェクトの更新バージョンを返す代わりに HTTP 5xx ステータスコードを返す
 - オリジンが、キャッシュコントロールヘッダーによって制限されておらず、以下のステータスコードのリストに含まれている HTTP 4xx ステータスコードを返す: [常にキャッシュする CloudFront HTTP 4xx および 5xx ステータスコード](#)。
 - オリジンが Cache-Control max-age ヘッダーまたは Cache-Control s-maxage ヘッダーなしで HTTP 4xx ステータスコードを返す。ステータスコードは、次のステータスコードの一覧に含まれる: コントロール [Cache-Control ヘッダーに基づいてキャッシュする CloudFront HTTP 4xx ステータスコード](#)。

CloudFront は以下を実行します。

1. ビューワーリクエストを受信した CloudFront エッジキャッシュで、 はディストリビューション設定 CloudFront をチェックし、オリジンが返したステータスコードに対応するカスタムエラーページのパスを取得します。
2. CloudFront は、カスタムエラーページのパスと一致するパスパターンを持つディストリビューション内の最初のキャッシュ動作を検索します。
3. CloudFront エッジロケーションは、キャッシュ動作で指定されたオリジンにカスタムエラーページのリクエストを送信します。

4. オリジンはカスタムエラーページをエッジロケーションに返します。
5. CloudFront は、リクエストを行ったビューワーにカスタムエラーページを返します。また、次の最大値までカスタムエラーページをキャッシュします。
 - エラーキャッシュ最小 TTL で指定された時間の長さ (デフォルトでは 10 秒)
 - Cache-Control max-age ヘッダー、または最初のリクエストがエラーを生成したときに発信元から返された Cache-Control s-maxage ヘッダーで指定された時間
6. キャッシュ時間 (ステップ 5 で決定) が経過すると、 はオリジンに別のリクエストを転送して、リクエストされたオブジェクトの取得を CloudFront 再試行します。 は、エラーキャッシュ最小 TTL で指定された間隔で再試行を CloudFront 続けます。

リクエストされたオブジェクトがエッジキャッシュにある場合

CloudFront 次のすべてが当てはまる場合、 はエッジキャッシュに現在存在するオブジェクトを提供し続けます。

- ビューワーがオブジェクトを要求する
- オブジェクトがエッジキャッシュに存在するが有効期限が切れている
- オリジンがステータスコード 304 (変更なし) またはオブジェクトの更新バージョンを返す代わりに HTTP 5xx ステータスコードを返す

CloudFront は以下を実行します。

1. オリジンが 5xx ステータスコードを返した場合、 は有効期限が切れていてもオブジェクト CloudFront を提供します。エラーキャッシュ最小 TTL の間、 はエッジキャッシュからオブジェクトを提供することでビューワーリクエストに応答し CloudFront 続けます。

オリジンが 4xx ステータスコードを返した場合、 CloudFront はビューワーに、リクエストされたオブジェクトではなく、ステータスコードを返します。

2. エラーキャッシュ最小 TTL が経過すると、 はオリジンに別のリクエストを転送して、リクエストされたオブジェクトの取得を CloudFront 再試行します。オブジェクトが頻繁にリクエストされない場合は、オリジンサーバーが 5xx レスポンスを返している間に、エッジキャッシュからオブジェクトを削除する CloudFront 可能性があります。オブジェクトが CloudFront エッジキャッシュに保持される期間については、「」を参照してください [コンテンツがキャッシュに保持される期間 \(有効期限\) の管理](#)。

カスタムエラーページを設定していない場合に がエラー CloudFront を処理する方法

カスタムエラーページを設定していない場合の動作 CloudFront は、リクエストされたオブジェクトがエッジキャッシュにあるかどうかによって異なります。

リクエストされたオブジェクトがエッジキャッシュにない場合

CloudFront 以下がすべて当てはまる場合、 は引き続きオリジンからリクエストされたオブジェクトを取得しようとします。

- ビューワーがオブジェクトを要求する
- オブジェクトがエッジキャッシュにない
- オリジンが HTTP 4xx または 5xx ステータスコードを返して、以下のいずれかに該当する:
 - オリジンがステータスコード 304 (変更なし) またはオブジェクトの更新バージョンを返す代わりに HTTP 5xx ステータスコードを返す
 - オリジンがキャッシュコントロールヘッダーによって制限されておらず、以下のステータスコードのリストに含まれている HTTP 4xx ステータスコードを返す: [常にキャッシュする CloudFront HTTP 4xx および 5xx ステータスコード](#)。
 - オリジンが Cache-Control max-age ヘッダーまたは Cache-Control s-maxage ヘッダーなしで HTTP 4xx ステータスコードを返す。ステータスコードは、次のステータスコードの一覧に含まれる: コントロール [Cache-Control ヘッダーに基づいてキャッシュする CloudFront HTTP 4xx ステータスコード](#)。

CloudFront は以下を実行します。

1. CloudFront は、4xx または 5xx ステータスコードをビューワーに返し、リクエストを受信したエッジキャッシュにステータスコードをキャッシュします。
 - エラーキャッシュ最小 TTL で指定された時間の長さ (デフォルトでは 10 秒)
 - Cache-Control max-age ヘッダー、または最初のリクエストがエラーを生成したときに発信元から返された Cache-Control s-maxage ヘッダーで指定された時間
2. キャッシュ時間 (ステップ 1 で決定) の間、キャッシュされた 4xx または 5xx ステータスコードを持つ同じオブジェクトに対する後続のビューワーリクエスト CloudFront に対応します。
3. キャッシュ時間 (ステップ 1 で決定) が経過すると、 はオリジンに別のリクエストを転送して、リクエストされたオブジェクトの取得を CloudFront 再試行します。 は、エラーキャッシュ最小 TTL で指定された間隔で再試行を CloudFront 続行します。

リクエストされたオブジェクトがエッジキャッシュにある場合

CloudFront 次のすべてが当てはまる場合、 はエッジキャッシュに現在存在するオブジェクトを提供し続けます。

- ビューワーがオブジェクトを要求する
- オブジェクトがエッジキャッシュに存在するが有効期限が切れている
- オリジンがステータスコード 304 (変更なし) またはオブジェクトの更新バージョンを返す代わりに HTTP 5xx ステータスコードを返す

CloudFront は以下を実行します。

1. オリジンが 5xx エラーコードを返した場合、 は有効期限が切れていてもオブジェクト CloudFront を提供します。エラーキャッシュの最小 TTL (デフォルトでは 10 秒) の間、エッジキャッシュからオブジェクトを提供することでビューワーリクエストに応答するように CloudFront 続行します。

オリジンが 4xx ステータスコードを返した場合、 CloudFront はビューワーに、リクエストされたオブジェクトではなく、ステータスコードを返します。

2. エラーキャッシュ最小 TTL が経過すると、 はオリジンに別のリクエストを転送して、リクエストされたオブジェクトの取得を CloudFront 再試行します。オブジェクトが頻繁にリクエストされない場合は、オリジンサーバーが 5xx レスポンスを返している間に、エッジキャッシュからオブジェクトを削除する CloudFront 可能性があります。オブジェクトが CloudFront エッジキャッシュに保持される期間については、「」を参照してください [コンテンツがキャッシュに保持される期間 \(有効期限\) の管理](#)。

キャッシュする CloudFront HTTP 4xx および 5xx ステータスコード

CloudFront は、返される特定のステータスコードと、オリジンがレスポンスで特定のヘッダーを返すかどうかに応じて、オリジンから返される HTTP 4xx および 5xx ステータスコードをキャッシュします。

常にキャッシュする CloudFront HTTP 4xx および 5xx ステータスコード

CloudFront は、オリジンから返される次の HTTP 4xx および 5xx ステータスコードを常にキャッシュします。HTTP ステータスコードのカスタムエラーページを設定している場合、 はカスタムエラーページを CloudFront キャッシュします。

404	Not Found
414	Request-URI Too Large
500	Internal Server Error
501	Not Implemented
502	Bad Gateway
503	Service Unavailable
504	Gateway Time-out

Cache-Control ヘッダーに基づいてキャッシュする CloudFront HTTP 4xx ステータスコード

CloudFront は、オリジンが Cache-Control max-age または Cache-Control s-maxage ヘッダーを返す場合にのみ、オリジンから返された次の HTTP 4xx ステータスコードをキャッシュします。これらの HTTP ステータスコードのいずれかに対してカスタムエラーページを設定しており、オリジンがキャッシュコントロールヘッダーの 1 つを返す場合、はカスタムエラーページを CloudFront キャッシュします。

400	Bad Request
403	Forbidden
405	Method Not Allowed
412	Precondition Failed
415	Unsupported Media Type

によるビデオオンデマンドおよびライブストリーミングビデオ CloudFront

を使用して CloudFront、任意の HTTP オリジンを使用してビデオオンデマンド (VOD) またはライブストリーミングビデオを配信できます。クラウドで動画ワークフローをセットアップする方法の 1 つは、[AWS Media Services](#) と CloudFront を併用することです。

トピック

- [ストリーミングビデオについて: オンデマンドおよびライブストリーミングビデオ](#)
- [によるビデオオンデマンド \(VOD\) の配信 CloudFront](#)
- [CloudFront と AWS メディアサービスによるライブストリーミングビデオの配信](#)

ストリーミングビデオについて: オンデマンドおよびライブストリーミングビデオ

がコンテンツを CloudFront に配信する前に、エンコーダーを使用して動画コンテンツをパッケージ化する必要があります。パッケージ化プロセスでは、オーディオ、ビデオ、キャプションのコンテンツを含むセグメントが作成されます。また、マニフェストファイルも生成します。マニフェストファイルは、どのセグメントをいつ再生するかについての特定の順序を説明します。パッケージの一般的な形式は MPEG DASH、Apple HLS、Microsoft Smooth Streaming、CMAF です。

ビデオオンデマンド (VOD) ストリーミング

オンデマンドビデオ (VOD) ストリーミングの場合、ビデオコンテンツはサーバーに保存され、ビューワーはいつでも視聴できます。ビューワーがストリーミングできるアセットを作成するには [AWS Elemental MediaConvert](#) などのエンコーダを使用してメディアファイルをフォーマットおよびパッケージ化します。

動画を適切な形式にパッケージ化したら、サーバーまたは Amazon S3 バケットに保存し、ビューワー CloudFront がリクエストしたときに配信できます。

ライブビデオストリーミング

ライブビデオストリーミングの場合、ビデオコンテンツは、ライブイベントが発生するとリアルタイムでストリーミングされるか、24 時間 365 日のライブチャンネルとして設定されます。ブロードキャストおよびストリーミング配信用のライブ出力を作成するには、AWS Elemental

MediaLive などのエンコーダを使用してビデオを圧縮し、デバイス表示用にフォーマットします。

ビデオがエンコードされたら、そのビデオを AWS Elemental MediaStore に保存するか、AWS Elemental MediaPackage を使用してさまざまな配信形式に変換できます。これらのオリジンのいずれかを使用して、コンテンツを配信する CloudFront ディストリビューションを設定します。これらのサービスと連携するディストリビューションを作成するための具体的な手順とガイダンスについては、「[AWS Elemental MediaStore をオリジンとして使用する動画の配信](#)」と「[AWS Elemental MediaPackage でフォーマットされたライブ動画の配信](#)」を参照してください。

Wowza と統合ストリーミングには、でビデオをストリーミングするために使用できるツールも用意されています。CloudFront。で Wowza を使用方法の詳細については CloudFront、[Wowza ドキュメントウェブサイトの「Wowza Streaming Engine ライセンスを CloudFront ライブ HTTP ストリーミングに持ち込む](#)」を参照してください。VOD ストリーミング CloudFront にで統合ストリーミングを使用する方法については、統合ストリーミングドキュメントウェブサイト [CloudFront](#) の「」を参照してください。

によるビデオオンデマンド (VOD) の配信 CloudFront

でビデオオンデマンド (VOD) ストリーミングを配信するには CloudFront、次のサービスを使用します。

- Amazon S3 を使用してコンテンツを元の形式で保存し、トランスコードされたビデオを保存します。
- ビデオをストリーミング形式に変換するエンコーダ (AWS Elemental MediaConvert など)。
- CloudFront トランスコードされたビデオを視聴者に配信する。Microsoft スムーズストリーミングについては、「[Microsoft Smooth Streaming のビデオオンデマンドの設定](#)」を参照してください。

で VOD ソリューションを作成するには CloudFront

1. コンテンツを Amazon S3 バケットにアップロードします。Amazon S3 の使用の詳細については、[Amazon Simple Storage Service ユーザーガイド](#)を参照してください。
2. MediaConvert ジョブを使用してコンテンツをトランスコードします。このジョブは、ビデオをビューワーが使用するプレーヤーが必要とする形式に変換します。ジョブを使用して、解像度とビットレートが異なるアセットを作成することもできます。これらのアセットは、視聴者の利用可能な帯域幅に応じて視聴品質を調整するアダプティブビットレート (ABR) ストリーミングに使用されます。はトランスコードされた動画を S3 バケットに MediaConvert 保存します。

- CloudFront デイストリビューションを使用して、変換されたコンテンツを配信します。ビューワーはいつでも、どのデバイスでもコンテンツを視聴できます。

Tip

AWS CloudFormation テンプレートを使用して VOD AWS ソリューションを関連するすべてのコンポーネントと共にデプロイする方法を検討することができます。テンプレートの使用手順については、AWS オンデマンドビデオガイドの「[Automated Deployment \(自動デプロイ\)](#)」を参照してください。

Microsoft Smooth Streaming のビデオオンデマンドの設定

CloudFront を使用して、Microsoft Smooth Streaming 形式にトランスコードしたビデオオンデマンド (VOD) コンテンツを配信するには、次のオプションがあります。

- Microsoft IIS を実行し、デイストリビューションのオリジンとしてスムーズストリーミングをサポートするウェブサーバーを指定します。
- CloudFront デイストリビューションのキャッシュ動作でスムーズストリーミングを有効にします。1つのデイストリビューションでは複数のキャッシュビヘイビアを使用できるため、1つのデイストリビューションをスムーズストリーミングメディアファイルとその他のコンテンツに使用できます。

Important

Microsoft IIS を実行しているウェブサーバーをオリジンとして指定する場合は、ディスク CloudFront トリビューションのキャッシュ動作でスムーズストリーミングを有効にしないでください。スムーズストリーミングをキャッシュ動作として有効にすると、Microsoft IIS サーバーをオリジンとして CloudFront 使用することはできません。

オリジンサーバーに対してスムーズストリーミングを有効にした場合 (つまり、Microsoft IIS サーバーがない場合) は、以下の点に注意してください。

- 他のコンテンツがキャッシュ動作の [Path Pattern (パスパターン)] の値と一致している場合、同じキャッシュ動作を使用してそのコンテンツも配信できます。

- CloudFront は、スムーズストリーミングメディアファイルの Amazon S3 バケットまたはカスタムオリジンのいずれかを使用できます。キャッシュ動作でスムーズストリーミングを有効にすると、Microsoft IIS Server をオリジンとして CloudFront 使用することはできません。
- スムーズストリーミング形式のメディアファイルを無効にすることはできません。有効期限が切れる前にファイルを更新する場合は、ファイルの名前を変更する必要があります。詳細については、「[が配信する CloudFront コンテンツの追加、削除、または置き換え](#)」を参照してください。

スムーズストリーミングクライアントの詳細については、Microsoft ドキュメントウェブサイトの「[スムーズストリーミング](#)」を参照してください。

Microsoft IIS ウェブサーバーがオリジンでないときに CloudFront を使用してスムーズストリーミングファイルを配信するには

1. スムーズストリーミングでフラグメント化された MP4 形式にメディアファイルを変換します。
2. 次のいずれかを行います。
 - CloudFront コンソールを使用している場合：ディストリビューションを作成または更新するときは、ディストリビューションのキャッシュ動作の 1 つ以上でスムーズストリーミングを有効にします。
 - CloudFront API を使用している場合：ディストリビューションのキャッシュ動作の 1 つ以上の DistributionConfig 複合型に SmoothStreaming 要素を追加します。
3. スムーズストリーミングファイルをオリジンにアップロードします。
4. `clientaccesspolicy.xml` または `crossdomainpolicy.xml` ファイルを作成し、それをディストリビューションのルートのアクセスできる場所に追加します (例: `https://d1111111abcdef8.cloudfront.net/clientaccesspolicy.xml`)。次に、ポリシーの例を示します。

```
<?xml version="1.0" encoding="utf-8"?>
<access-policy>
<cross-domain-access>
<policy>
<allow-from http-request-headers="*">
<domain uri="*" />
</allow-from>
<grant-to>
<resource path="/" include-subpaths="true" />
</grant-to>
</policy>
```

```
</cross-domain-access>  
</access-policy>
```

詳細については、Microsoft デベロッパーネットワークウェブサイトの「[Making a Service Available Across Domain Boundaries](#)」を参照してください。

- アプリケーション (メディアプレーヤーなど) 内のリンクの場合は、以下の標準形式でメディアファイルの URL を指定します。

```
https://d1111111abcdef8.cloudfront.net/video/presentation.ism/Manifest
```

CloudFront と AWS メディアサービスによるライブストリーミングビデオの配信

で AWS メディアサービスを使用して世界中の視聴者にライブコンテンツを CloudFront に配信するには、このセクションに含まれるガイダンスに従ってください。

[AWS Elemental MediaLive](#) では、ライブビデオストリームがリアルタイムでエンコードされます。大きなビデオストリームをエンコードするために、はビューワーに配信できる小さなバージョン (エンコード) に MediaLive で圧縮します。

ライブビデオストリームを圧縮した後、次の 2 つの主要なオプションのいずれかを使用して、コンテンツを準備および配信できます。

- 必要な形式にコンテンツを変換して配信する: [AWS Elemental MediaPackage](#) を使用して、ビデオコンテンツを 1 つの形式から複数の形式に変換し、さまざまなタイプのデバイス用にコンテンツをパッケージ化できます。コンテンツをパッケージ化するとき、追加の機能を実装し、デジタル著作権管理 (DRM) を追加して、コンテンツの不正使用を防ぐこともできます。CloudFront を使用してフォーマットされたコンテンツを配信する step-by-step 手順については、MediaPackage「」を参照してください [AWS Elemental MediaPackage でフォーマットされたライブ動画の配信](#)。
- スケーラブルなオリジンを使用してコンテンツを保存して配信する: ビューワーが使用するすべてのデバイスに必要な形式でコンテンツを MediaLive でエンコードした場合は、のような高度 [AWS Elemental MediaStore](#) にスケーラブルなオリジンを使用してコンテンツを提供します。CloudFront を使用して MediaStore コンテナに保存されているコンテンツを配信する step-by-step 手順については、「」を参照してください [AWS Elemental MediaStore をオリジンとして使用する動画の配信](#)。

これらのオプションのいずれかを使用してオリジンを設定したら、CloudFront を使用して、ビューワーにライブストリーミングビデオを配信できます。

Tip

可用性の高いリアルタイムの視聴エクスペリエンスを実現するサービスを自動的にデプロイする AWS ソリューションについて、情報を入手することができます。このソリューションを自動的にデプロイする手順については、「[ライブストリーミングの自動デプロイ](#)」を参照してください。

トピック

- [AWS Elemental MediaStore をオリジンとして使用する動画の配信](#)
- [AWS Elemental MediaPackage でフォーマットされたライブ動画の配信](#)

AWS Elemental MediaStore をオリジンとして使用する動画の配信

[AWS Elemental MediaStore](#) コンテナにビデオが保存されている場合は、コンテンツを配信する CloudFront ディストリビューションを作成できます。

開始するには、MediaStore コンテナ CloudFront へのアクセスを許可します。次に、CloudFront ディストリビューションを作成し、と連携するように設定します MediaStore。

AWS Elemental MediaStore コンテナからコンテンツを提供するには

1. [「Amazon が AWS Elemental MediaStore コンテナ にアクセス CloudFront することを許可する」](#)の手順に従って、以下の手順に戻りディストリビューションを作成します。
2. 次の設定でディストリビューションを作成します。

オリジンドメイン

MediaStore コンテナに割り当てられるデータエンドポイント。ドロップダウンリストから、ライブ動画の MediaStore コンテナを選択します。

オリジンのパス

オブジェクトが保存されている MediaStore コンテナ内のフォルダ構造。詳細については、「[the section called “オリジンのパス”](#)」を参照してください。

カスタムヘッダーを追加する

リクエスト CloudFront をオリジンに転送するときカスタムヘッダーを追加する場合は、ヘッダー名と値を追加します。

ビューワープロトコルポリシー

[Redirect HTTP to HTTPS (HTTP から HTTPS へのリダイレクト)] を選択します。詳細については、「[the section called “ビューワープロトコルポリシー”](#)」を参照してください。

キャッシュポリシーとオリジンリクエストポリシー

キャッシュポリシーでは、[Create policy] (ポリシーの作成) を選択し、キャッシュのニーズとセグメントの継続期間に適したキャッシュポリシーを作成します。ポリシーを作成したら、キャッシュポリシーのリストを更新し、先ほど作成したポリシーを選択します。

オリジンリクエストポリシー で、ドロップダウンリストから CORS-CustomOrigin を選択します。

その他の設定については、他の技術的要件またはビジネスのニーズに基づいて特定の値を設定できます。ディストリビューションのすべてのオプションのリストおよびその設定に関する情報については、「[the section called “指定する値”](#)」を参照してください。

3. アプリケーション (メディアプレーヤーなど) 内のリンクの場合は、を使用して配信する他のオブジェクトに使用すると同じ形式でメディアファイルの名前を指定します CloudFront。

AWS Elemental MediaPackage でフォーマットされたライブ動画の配信

AWS Elemental MediaPackage を使用してライブストリームを視聴用にフォーマットした場合は、CloudFront ディストリビューションを作成し、キャッシュ動作を設定して、ライブストリームを配信できます。以下のプロセスでは、を使用してライブ動画の[チャンネルを作成し、エンドポイントを追加](#)済みであることを前提としています MediaPackage。

の CloudFront ディストリビューション MediaPackage を手動で作成するには、次の手順に従います。

ステップ

- [ステップ 1: CloudFront ディストリビューションを作成して設定する](#)
- [ステップ 2: MediaPackage エンドポイントのドメインにオリジンを追加する](#)
- [ステップ 3: すべてのエンドポイントのキャッシュ動作を設定する](#)

- [ステップ 4: ヘッダーベースの MediaPackage CDN 認証を有効にする](#)
- [ステップ 5: CloudFront を使用してライブストリームチャンネルを提供する](#)

ステップ 1: CloudFront ディストリビューションを作成して設定する

で作成したライブビデオチャンネルの CloudFront ディストリビューションを設定するには、以下の手順を実行します MediaPackage。

ライブビデオチャンネルのディストリビューションを作成するには

1. にサインインAWS Management Consoleし、 で CloudFront コンソールを開きます<https://console.aws.amazon.com/cloudfront/v4/home>。
2. [ディストリビューションの作成] を選択します。
3. 以下のようなディストリビューション用の設定を選択します。

オリジンドメイン

MediaPackage ライブ動画チャンネルとエンドポイントがあるオリジン。テキストフィールドを選択し、ドロップダウンリストからライブ動画の MediaPackage オリジンドメインを選択します。1 つのドメインを複数のオリジンエンドポイントにマップできます。

別の AWS アカウントを使用してオリジンドメインを作成した場合は、オリジンの URL 値をフィールドに入力します。オリジンは HTTPS URL であることが必要です。

例えば、<https://3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com/out/v1/abc123/index.m3u8> などの HLS エンドポイントの場合、オリジンドメインは 3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com です。

詳細については、「[the section called “オリジンドメイン”](#)」を参照してください。

オリジンのパス

コンテンツが提供される MediaPackage エンドポイントへのパス。

[オリジンパス] フィールドは自動入力されません。正しいオリジンパスを手動で入力する必要があります。

オリジンパスの仕組みの詳細については、[the section called “オリジンのパス”](#) を参照してください。

⚠ Important

CloudFront デイストリビューション内のどこかにルーティングするには、ワイルドカードパス*が必要です。明示的なパスと一致しないリクエストが実際のオリジンにルーティングされないようにするには、そのワイルドカードパスに「ダミー」オリジンを作成します。

Example : 「ダミー」オリジンの作成

次の例で、エンドポイント abc123 および def456 は「実際の」オリジンにルーティングしますが、他のエンドポイントのビデオコンテンツのリクエストは、適切なサブドメインがないと `mediapackage.us-west-2.amazonaws.com` にルーティングされ、HTTP 404 エラーになります。

MediaPackage エンドポイント :

```
https://3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com/out/v1/abc123/index.m3u8
https://3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com/out/v1/def456/index.m3u8
```

CloudFront オリジン A:

```
Domain: 3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com
Path: None
```

CloudFront オリジン B:

```
Domain: mediapackage.us-west-2.amazonaws.com
Path: None
```

CloudFront キャッシュ動作 :

1. Path: `/out/v1/abc123/*` forward to Origin A
2. Path: `/out/v1/def456/*` forward to Origin A
3. Path: `*` forward to Origin B

その他のディストリビューション設定には、他の技術的要件またはビジネスニーズに基づいて値を指定します。ディストリビューションのすべてのオプションのリストおよびその設定に関する情報については、「[the section called “指定する値”](#)」を参照してください。

他のディストリビューションの設定を選択したら、[Create Distribution] (ディストリビューションの作成) を選択します。

4. 作成したディストリビューションを選択し、[Behaviors] (動作) タブを選択します。
5. デフォルトのキャッシュ動作を選択し、[Edit] (編集) をクリックします。オリジンに対して選択したチャンネルの正しいキャッシュ動作設定を指定します。後で他のオリジンを1つ以上追加し、それらのキャッシュ動作設定を編集します。
6. [CloudFront ディストリビューションページ](#) に移動します。
7. ディストリビューションの最終更新日時列の値がデプロイから、CloudFront がディストリビューションを作成したことを示す日時に変更されるまで待ちます。

ステップ 2: MediaPackage エンドポイントのドメインにオリジンを追加する

このステップを繰り返して、各 MediaPackage チャンネルエンドポイントをディストリビューションに追加します。ただし、「ダミー」オリジンを作成する必要があることに注意してください。

オリジンとして他のエンドポイントを追加するには

1. CloudFront コンソールで、チャンネル用に作成したディストリビューションを選択します。
2. [Origins] (オリジン)、[Create origin] (オリジンの作成) の順に選択します。
3. オリジンドメイン の場合、ドロップダウンリストからチャンネルのMediaPackage エンドポイントを選択します。
4. その他の設定には、他の技術的要件またはビジネスニーズに基づいて値を指定します。詳細については、「[the section called “オリジンの設定”](#)」を参照してください。
5. [Create Origin] (オリジンの作成) を選択します。

ステップ 3: すべてのエンドポイントのキャッシュ動作を設定する

エンドポイントごとに、キャッシュ動作を設定して、リクエストを正しくルーティングするパスパターンを追加する必要があります。指定するパスパターンは、配信するビデオの形式によって異なります。次の手順には、Apple HLS、CMAF、DASH、および Microsoft スムーズストリーミング形式で使用するパスパターン情報が含まれています。

通常、エンドポイントごとに2つのキャッシュ動作を設定します。

- ファイルのインデックスになる親マニフェスト
- セグメント (ビデオコンテンツのファイル)

エンドポイントのキャッシュ動作を作成するには

1. CloudFront コンソールで、チャンネル用に作成したディストリビューションを選択します。
2. [Behaviors] (動作)、[Create Behavior] (動作の作成) の順に選択します。
3. パスパターン では、パスプレフィックスとして特定の MediaPackage OriginEndpoint GUID を使用します。

パスパターン

`https://3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com/out/v1/abc123/index.m3u8` などの HLS エンドポイントの場合は、以下の2つのキャッシュビヘイビアを作成します。

- 親および子マニフェストの場合は、`/out/v1/abc123/*.m3u8` を使用します。
- コンテンツセグメントの場合は、`/out/v1/abc123/*.ts` を使用します。

`https://3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com/out/v1/abc123/index.m3u8` などの CMAF エンドポイントの場合は、以下の2つのキャッシュビヘイビアを作成します。

- 親および子マニフェストの場合は、`/out/v1/abc123/*.m3u8` を使用します。
- コンテンツセグメントの場合は、`/out/v1/abc123/*.mp4` を使用します。

`https://3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com/out/v1/abc123/index.mpd` などの DASH エンドポイントの場合は、以下の2つのキャッシュビヘイビアを作成します。

- 親マニフェストの場合は、`/out/v1/abc123/*.mpd` を使用します。
- コンテンツセグメントの場合は、`/out/v1/abc123/*.mp4` を使用します。

`https://3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com/out/v1/abc123/index.ism` などの Microsoft Smooth Streaming エンドポイントの場合は、マニフェストのみが使用されるため、1つのキャッシュビヘイビア (`out/v1/abc123/index.ism/*`) のみを作成します。

4. キャッシュ動作ごとに、以下の設定の値を指定します。

ビューワープロトコルポリシー

[Redirect HTTP to HTTPS (HTTP から HTTPS へのリダイレクト)] を選択します。

キャッシュポリシーとオリジンリクエストポリシー

キャッシュポリシーでは、[Create policy] (ポリシーの作成) を選択します。新しいキャッシュポリシーでは、次の設定を指定します。

最小 TTL

古いコンテンツを提供しないように 5 秒以下に設定します。

クエリ文字列

クエリ文字列([Cache key settings] (キャッシュキー設定)) で、[Include specified query strings] (指定したクエリ文字列を含める) を選択します。[Allow] (許可) では、次の値を入力し、[Add item] (アイテムを追加) を選択します。

- キャッシュの基準として使用するクエリ文字列パラメータ CloudFront として を追加します。MediaPackage レスポンスには、エンドポイントの変更時刻をキャプチャ? m=### するための タグが常に含まれます。コンテンツがすでにキャッシュされていて、このタグに別の値が含まれている場合、 はキャッシュされたバージョンを提供する代わりに、新しいマニフェストを CloudFront リクエストします。
- でタイムシフト表示機能を使用している場合は MediaPackage、マニフェストリクエスト (、 *.m3u8、) のキャッシュ動作の追加のクエリ文字列パラメータ startend として *.mpd と を指定します index.ism/*。これにより、マニフェストリクエストで指定した期間に固有のコンテンツが配信されます。コンテンツのタイムシフト表示および形式の開始/終了リクエストパラメータの詳細については、「AWS Elemental MediaPackage ユーザーガイド」の「[タイムシフト表示](#)」を参照してください。
- でマニフェストフィルタリング機能を使用している場合は MediaPackage、マニフェストリクエスト (、 *.m3u8、 *.mpd) のキャッシュ動作で使用するキャッシュポリシーの追加クエリ文字列パラメータ aws.manifestfilter として を指定します index.ism/*。これにより、aws.manifestfilter クエリ文字列を MediaPackage オリジンに転送するようにディストリビューションが設定されます。オリジンは、マニフェストフィルタリング機能が機能するために必要です。詳細については、「AWS Elemental MediaPackage ユーザーガイド」の「[マニフェストでのフィルタリング](#)」を参照してください。

- 低レイテンシー HLS (LL-HLS) を使用している場合は、マニフェストリクエスト (* .m3u8) のキャッシュビヘイビアで使用するキャッシュポリシーの追加クエリ文字列パラメータとして `_HLS_msn` および `_HLS_part` を指定します。これにより、`_HLS_msn` および `_HLS_part` クエリ文字列を MediaPackage オリジンに転送するようにディストリビューションが設定されます。これは、LL-HLS ブロックプレイリストリクエスト機能が機能するために必要です。
5. [作成] を選択します。
 6. キャッシュポリシーを作成したら、キャッシュ動作の作成ワークフローに戻ります。キャッシュポリシーのリストを更新し、先ほど作成したポリシーを選択します。
 7. [Create behavior] (動作の作成) を選択します。
 8. エンドポイントが Microsoft Smooth Streaming エンドポイント以外の場合、次の手順を繰り返して 2 つ目のキャッシュ動作を作成します。

ステップ 4: ヘッダーベースの MediaPackage CDN 認証を有効にする

エンドポイントとディストリビューション間で MediaPackage ヘッダーベースの MediaPackage CDN 認証を有効にすることをお勧めします。詳細については、「[ユーザーガイド](#)」の「[で CDN 認証を有効にする MediaPackage](#) AWS Elemental MediaPackage」を参照してください。

ステップ 5: CloudFront を使用してライブストリームチャンネルを提供する

ディストリビューションを作成し、オリジンを追加し、キャッシュ動作を作成し、ヘッダーベースの CDN 認証を有効にすると、キャッシュ動作に設定した設定に基づいて、ビューワーからの CloudFront. CloudFront routes リクエストを正しい MediaPackage エンドポイントに配信できます。

アプリケーション (メディアプレーヤーなど) 内のリンクの場合は、メディアファイルの URL を URL CloudFront URLs。詳細については、「[the section called “ファイルの URL のカスタマイズ”](#)」を参照してください。

関数を使用してエッジでカスタマイズ

Amazon では CloudFront、独自のコードを記述して、CloudFront デイストリビューションが HTTP リクエストとレスポンスを処理する方法をカスタマイズできます。このコードは、レイテンシーを最小限に抑えるためにビューワー (ユーザー) の近くで実行するため、サーバーやその他のインフラストラクチャを管理する必要はありません。を通過するリクエストとレスポンスの操作 CloudFront、基本的な認証と承認の実行、エッジでの HTTP レスポンスの生成などを行うコードを記述できます。

CloudFront デイストリビューションに記述してアタッチするコードは、エッジ関数 と呼ばれます。CloudFront には、エッジ関数の記述と管理を行う 2 つの方法が用意されています。

- CloudFront 関数 – CloudFront Functions を使用すると、で軽量の関数を記述 JavaScript し、レイテンシーの影響を受けやすい CDN のカスタマイズを大規模に行うことができます。CloudFront Functions ランタイム環境は、1 秒あたり数百万件のリクエストを処理するようにすぐにスケールでき、安全性も高くなります。CloudFront Functions は のネイティブ機能です。つまり CloudFront、内でコードを完全に構築、テスト、デプロイできます CloudFront。
- Lambda@Edge - Lambda@Edge は、複雑な関数のための強力で柔軟なコンピューティングを提供し、ビューワーにより近い場所に完全なアプリケーションロジックを実装する、安全性に優れた [AWS Lambda](#) の拡張機能です。Lambda@Edge 関数は、Node.js または Python runtime 環境で実行されます。1 つのAWSリージョンに発行しますが、関数を CloudFront デイストリビューションに関連付けると、Lambda@Edge はコードを世界中に自動的にレプリケートします。

CloudFront Functions と Lambda@Edge の選択

CloudFront 関数と Lambda@Edge はどちらも、イベントに応答して CloudFrontコードを実行する方法を提供します。しかし、両者には大きな違いがあります。この違いが分かれば、ユースケースに適したものを選択するのに役立ちます。次の表は、CloudFront 関数と Lambda@Edge の重要な違いの一部を示しています。

	CloudFront 関数	Lambda@Edge
プログラミング言語	JavaScript (ECMAScript 5.1 準拠)	Node.js と Python
イベントソース	• ビューワーリクエスト	• ビューワーリクエスト

	CloudFront 関数	Lambda@Edge
	<ul style="list-style-type: none"> ビューワーレスポンス 	<ul style="list-style-type: none"> ビューワーレスポンス オリジンリクエスト オリジンレスポンス
Scale	リクエスト数: 毎秒 10,000,000 件以上	リクエスト数: 1 リージョンあたり毎秒 10,000 件まで
関数の持続時間	サブミリ秒	<p>最大 5 秒 (ビューワーリクエスト、ビューワーレスポンス)</p> <p>最大 30 秒 (オリジンリクエスト、オリジンレスポンス)</p>
最大メモリ	2 MB	128 ~ 3,008 MB
関数コードと含まれるライブラリの最大サイズ	10 KB	<p>1 MB (ビューワーリクエスト、ビューワーレスポンス)</p> <p>50 MB (オリジンリクエスト、オリジンレスポンス)</p>
ネットワークアクセス	いいえ	はい
ファイルシステムへのアクセス	いいえ	はい
リクエスト本文へのアクセス	いいえ	はい

	CloudFront 関数	Lambda@Edge
位置情報とデバイスデータへのアクセス	はい	いいえ (ビューワーリクエスト) はい (オリジンリクエスト、オリジンレスポンス、ビューワーレスポンス)
内で完全に構築およびテストできる CloudFront	はい	いいえ
関数のログとメトリクス	はい	はい
料金表	無料利用枠あり。リクエストごとに課金。	無料利用枠なし。リクエストと機能期間ごとに課金。

CloudFront 関数は、次のような軽量で実行時間の短い関数に最適です。

- キャッシュキーの正規化: HTTP リクエスト属性 (ヘッダー、クエリ文字列、Cookie、さらには URL パス) を変換して、最適な [キャッシュキー](#) を作成できます。これにより、キャッシュのヒット率を向上させることができます。
- ヘッダー操作: リクエストまたはレスポンスで HTTP ヘッダーを挿入、変更、または削除できます。たとえば、すべてのリクエストに True-Client-IP ヘッダーを追加できます。
- URL リダイレクトまたは書き換え: リクエスト内の情報に基づいてビューワーを他のページにリダイレクトしたり、すべてのリクエストをあるパスから別のパスに書き換えたりできます。
- リクエストの承認: 承認ヘッダーやその他のリクエストメタデータを調べることで、JSON ウェブトークン (JWT) などのハッシュ化された承認トークンを検証できます。

CloudFront Functions の使用を開始するには、「」を参照してください [CloudFront Functions によるエッジでのカスタマイズ](#)。

Lambda@Edge は、次のシナリオに適しています。

- 完了までに数ミリ秒以上かかる関数。

- 調整可能な CPU またはメモリを必要とする機能。
- サードパーティライブラリに依存する関数 (AWS SDK を含む、AWS のその他サービスとの統合用)。
- 処理に外部サービスを使用するためにネットワークアクセスが必要となる関数。
- ファイルシステムへのアクセスまたは HTTP リクエストの本文へのアクセスを必要とする関数。

Lambda@Edge の使用を開始するには、「[Lambda@Edge を使用したエッジでのカスタマイズ](#)」を参照してください。

CloudFront Functions によるエッジでのカスタマイズ

CloudFront Functions を使用すると、で軽量な関数を記述 JavaScript して、レイテンシーの影響を受けやすい CDN のカスタマイズを大規模に実行できます。関数は、を通過するリクエストとレスポンスの操作 CloudFront、基本的な認証と承認の実行、エッジでの HTTP レスポンスの生成などを行うことができます。CloudFront Functions ランタイム環境は、1 秒あたり数百万件のリクエストを処理するようにすぐにスケールでき、安全性が高くなります。CloudFront Functions は のネイティブ機能です。つまり CloudFront、内でコードを完全に構築、テスト、デプロイできます CloudFront。

CloudFront 関数は、次のような軽量で実行時間の短い関数に最適です。

- キャッシュキーの正規化: HTTP リクエスト属性 (ヘッダー、クエリ文字列、Cookie、さらには URL パス) を変換して、最適な[キャッシュキー](#)を作成できます。これにより、キャッシュのヒット率を向上させることができます。
- ヘッダー操作: リクエストまたはレスポンスで HTTP ヘッダーを挿入、変更、または削除できます。たとえば、すべてのリクエストに True-Client-IP ヘッダーを追加できます。
- ステータスコードの変更と本文の生成 — ヘッダーを評価し、カスタマイズされたコンテンツでビューワーに返すことができます。
- URL リダイレクトまたは書き換え: リクエスト内の情報に基づいてビューワーを他のページにリダイレクトしたり、すべてのリクエストをあるパスから別のパスに書き換えたりできます。
- リクエストの承認: 承認ヘッダーやその他のリクエストメタデータを調べることで、JSON ウェブトークン (JWT) などのハッシュ化された承認トークンを検証できます。

CloudFront 関数を CloudFront ディストリビューションに関連付けると、は CloudFront エッジロケーションでリクエストとレスポンスを CloudFront インターセプトし、関数に渡します。次のイベントが発生したときに、CloudFront 関数を呼び出すことができます。

- がビューワーからリクエスト CloudFront を受信したとき (ビューワーリクエスト)
- がビューワーにレスポンスを CloudFront 返す前 (ビューワーレスポンス)

簡単な概要については、「」を参照してください[チュートリアル: Functions を使用したシンプルな CloudFront 関数の作成](#)。

CloudFront キー値ストアに保存されているキーと値のペアを使用するように関数を設定することで、関数に変数を含めることができます。CloudFront 関数にキーと値のペアを含める方法の概要については、「」を参照してください[the section called “チュートリアル: キー値のある関数”](#)。

関数コードの記述とサンプルコードの読み取りを開始するには、[関数コードの記述](#)「」および「」を参照してください[サンプルのコード](#)。

チュートリアル: Functions を使用したシンプルな CloudFront 関数の作成

このチュートリアルでは、CloudFront 関数の使用を開始する方法を示します。ビューワーを別の URL にリダイレクトし、カスタムレスポンスヘッダーを返すシンプルな関数を作成できます。

前提条件

CloudFront Functions を使用するには、CloudFront デイストリビューションが必要です。お持ちでない場合は、[シンプルな CloudFront デイストリビューションの開始方法](#) の手順に従います。

関数の作成

この手順では、CloudFront コンソールを使用して、ビューワーを別の URL にリダイレクトし、カスタムレスポンスヘッダーも返すシンプルな関数を作成する方法を示します。

CloudFront コンソールで関数を作成するには

1. にサインインAWS Management Consoleし、 で CloudFront コンソールを開きます<https://console.aws.amazon.com/cloudfront/v4/home>。
2. ナビゲーションペインで、関数を選択し、関数の作成を選択します。
3. 関数の作成 ページの 名前 に、 などの関数名を入力します *MyFunctionName*。
4. (オプション) 説明 に、 などの関数の説明を入力します **Simple test function**。
5. ランタイム では、選択した JavaScript デフォルトのバージョンを維持します。
6. [関数の作成] を選択します。

7. 次の関数コードをコピーします。この関数コードは、ビューワーを別の URL にリダイレクトするほか、カスタムレスポンスヘッダーも返します。

```
function handler(event) {
    // NOTE: This example function is for a viewer request event trigger.
    // Choose viewer request for event trigger when you associate this function
    with a distribution.
    var response = {
        statusCode: 302,
        statusDescription: 'Found',
        headers: {
            'cloudfront-functions': { value: 'generated-by-CloudFront-Functions' },
            'location': { value: 'https://aws.amazon.com/cloudfront/' }
        }
    };
    return response;
}
```

8. 関数コードでは、コードをコードエディタに貼り付けて、デフォルトのコードを置き換えます。
9. [変更の保存] をクリックします。
10. (オプション) 関数を公開する前にテストできます。このチュートリアルでは、関数をテストする方法については説明していません。詳細については、「[関数のテスト](#)」を参照してください。
11. 公開 タブを選択し、公開 関数 を選択します。関数をデイス CloudFront トリビューションに関連付ける前に、関数を公開する必要があります。
12. 次に、関数をデイス トリビューションまたはキャッシュ動作に関連付けることができます。*MyFunctionName* ページで、公開タブを選択します。

Warning

次のステップでは、テストに使用するデイス トリビューションまたはキャッシュ動作を選択します。このテスト関数を、本番環境で使用されるデイス トリビューションまたはキャッシュ動作に関連付けないでください。

13. [Add association] を選択します。
14. 関連付けダイアログボックスで、デイス トリビューションまたはキャッシュ動作を選択します。イベントタイプ の場合、デフォルト値のままにします。
15. [Add association] を選択します。

関連付けられているディストリビューションは、[関連付けられているディストリビューション] テーブルに表示されます。

16. 関連付けられたディストリビューションのデプロイが完了するまで数分待ちます。ディストリビューションのステータスを確認するには、関連ディストリビューションテーブルでディストリビューションを選択し、ディストリビューションの表示を選択します。

ディストリビューションのステータスが [Deployed] になったら、関数の動作を確認できます。

関数の検証

関数の動作が確認できたら、ウェブブラウザでディストリビューションのドメイン名 (例: `https://d111111abcdef8.cloudfront.net`) に移動します。関数はブラウザにリダイレクトを返すため、ブラウザは自動的に `https://aws.amazon.com/cloudfront/` に移動します。

`curl` などのツールを使用してディストリビューションのドメイン名にリクエストを送信すると、次の例で強調されているように、関数によって追加されたリダイレクト応答 (302 Found) とカスタムレスポンスヘッダーが表示されます。

```
curl -v https://d111111abcdef8.cloudfront.net/
> GET / HTTP/1.1
> Host: d111111abcdef8.cloudfront.net
> User-Agent: curl/7.64.1
> Accept: */*
>
< HTTP/1.1 302 Found
< Server: CloudFront
< Date: Tue, 16 Mar 2021 18:50:48 GMT
< Content-Length: 0
< Connection: keep-alive
< Location: https://aws.amazon.com/cloudfront/
< Cloudfront-Functions: generated-by-CloudFront-Functions
< X-Cache: FunctionGeneratedResponse from cloudfront
< Via: 1.1 3035b31bddaf14eded329f8d22cf188c.cloudfront.net (CloudFront)
< X-Amz-Cf-Pop: PHX50-C2
< X-Amz-Cf-Id: ULZdIz6j43uGB1Xyob_JctF9x7CCbwpNniiM1mNbmwzH1YWP9FsEHg==
```

チュートリアル: キー値を含む関数の作成

このチュートリアルでは、CloudFront 関数にキー値を含める方法を示します。キー値はキーと値のペアの一部です。関数コードには (キーと値のペアの) 名前を含めます。関数が実行されると、は名前を CloudFront 値に置き換えます。

キーと値のペアはキー値ストアに保存される変数です。関数で (ハードコードされた値の代わりに) キーを使用すると、関数の柔軟性が高まります。キーの値は、コードの変更をデプロイしなくても変更できます。キーと値のペアによって関数のサイズを小さくすることもできます。キーと値のペアとキー値ストアの詳細については、「[???](#)」を参照してください。

前提条件

CloudFront 関数に精通していることを前提としています。関数とキー値ストアの両方に慣れていない場合は、まずは [the section called “チュートリアル: 単純な関数”](#) のチュートリアルに従ってください。

キー値ストアをセットアップする

ステップ 1: キー値ストアを作成する

1. 関数に含めたいキーと値のペアを計画します。キーの名前を書き留めます。

関数で使用するキーと値のペアはすべて 1 つのキー値ストアに含める必要があることに注意してください。

2. 作業の順序を決めます。次の 2 通りの方法があります。
 - キー値ストアを作成し、そのストアにキーと値のペアを追加します。次に、関数を作成 (または変更) し、キー名を組み込みます。
 - または、関数を作成 (または変更) し、使用したいキー名を組み込みます。次にキー値ストアを作成し、キーと値のペアを追加します。

このチュートリアルでは、[関数のチュートリアル](#)の関数を拡張することを前提としています。また、最初にキー値ストアを作成することを前提としています。

3. にサインインAWS Management Consoleし、 で CloudFront コンソールを開きます <https://console.aws.amazon.com/cloudfront/v4/home>。
4. 左のナビゲーションバーで、[関数] を選択します。関数ページで、KeyValueStoresタブを選択します。
5. 次のように、フィールドの作成 KeyValueStore を選択し、入力します。

- ストアの名前とオプションで説明を入力する
 - このチュートリアルではキーと値のペアを手動で入力する方法を実証しているため、[S3 URI] は空白のままにしておく
6. [サーバーの作成] ボタンを選択します。新しいキー値ストアの詳細ページが表示されます。このページには、現在空欄の [キーと値のペア] セクションがあります。

ステップ 2: キーと値のペアをストアに追加する

1. [キーと値のペア] セクションで、[キーと値のペアを追加] タブを選択します。[ペアの追加] を選択して名前と値を入力します。
2. [ペアの追加] ボタンを選択して別のペアを追加します。
3. 完了したら、[変更内容を保存] を選択してストア内のすべてのペアを保存します。表示される確認ダイアログで [完了] を選択します。

これで、キーと値のペアのグループを含むストアができました。

関数内でセットアップする

ステップ 3: キー値ストアを関数に関連付ける

これで、キー値ストアが作成されました。また、キー値ストアのキー名を含む関数を作成または変更しました。これでキー値ストアと関数に関連付けることができます。その関連付けは関数内から作成します。

1. 左のナビゲーションバーで、[関数] を選択します。デフォルトでは、[関数] タブが一番上に表示されます。
2. 関連 KeyValueStore セクションで、既存の の関連付け KeyValueStore を選択します。キー値ストアを選択し、関連付け KeyValueStore ボタンを選択します。各関数に関連付けることができるキー値ストアは 1 つだけであることに注意してください。

ステップ 4: 関数コードをテストしてパブリッシュする

1. 以下を行う場合を含め、関数コードを変更するたびに必ずテストする必要があります。
 - キー値ストアを関数に関連付ける
 - 関数とそのキー値ストアを変更して、新しいキーと値のペアを含める
 - キーと値のペアの値を変更する

関数をテストする方法については、「[the section called “関数のテスト”](#)」を参照してください。DEVELOPMENT ステージで関数のテストを選択するようにしてください。

2. 関数を (新規または改訂されたキーと値のペアとともに) LIVE 環境で使用できるようになったら、関数をパブリッシュします。

公開すると、は DEVELOPMENT ステージから ライブステージに関数のバージョン CloudFront をコピーします。関数には新しいコードが含まれ、キー値ストアに関連付けられます。(ライブステージで関連付けを再度実行する必要はありません)。

関数をパブリッシュする方法については、「[the section called “関数の公開”](#)」を参照してください。

関数コードの記述

Amazon の CloudFront 関数を使用すると CloudFront、で軽量な関数を記述 JavaScript して、レイテンシーの影響を受けやすい CDN のカスタマイズを大規模に行うことができます。関数コードは、を通過するリクエストとレスポンスの操作 CloudFront、基本的な認証と認可の実行、エッジでの HTTP レスポンスの生成などを行うことができます。

以下のトピックは、関数の CloudFront 関数コードを作成するのに役立ちます。

トピック

- [関数の目的を決定する](#)
- [CloudFront Functions イベント構造](#)
- [JavaScript CloudFront Functions の ランタイム機能](#)
- [キー値ストアのヘルパーメソッド](#)
- [CloudFront Functions のコード例](#)

関数の目的を決定する

関数コードを記述する前に、関数の目的を決めます。CloudFront Functions のほとんどの関数には、次のいずれかの目的があります。詳細については、関数の目的に該当するトピックを参照してください。

関数の目的にかかわらず、handler はあらゆる関数のエントリポイントです。という単一の引数を取り event、によって関数に渡されます CloudFront。event は、HTTP リクエスト表記の JSON オ

プロジェクトです (関数が HTTP レスポンスを変更する場合は、レスポンス)。event オブジェクトの構造の詳細については、「[CloudFront Functions イベント構造](#)」を参照してください。

CloudFront Functions と Lambda@Edge に適用される制限の詳細については、「[」](#)を参照してください。[エッジ関数に対する制限](#)。

トピック

- [ビューワーリクエストイベントタイプの HTTP リクエストの変更](#)
- [ビューワーリクエストイベントタイプで HTTP レスポンスを生成する](#)
- [ビューワーレスポンスイベントタイプの HTTP レスポンスの変更](#)

ビューワーリクエストイベントタイプの HTTP リクエストの変更

関数は、ビューワー (クライアント) から が CloudFront 受信した HTTP リクエストを変更し、変更されたリクエストを に返 CloudFront して処理を継続できます。たとえば、関数コードで[キャッシュキー](#)を正規化したり、リクエストヘッダーを変更したりできます。

HTTP リクエストを変更する関数を作成する場合は、必ずビューワーリクエストイベントタイプを選択してください。つまり、リクエストされたオブジェクトが CloudFront キャッシュ内にあるかどうかを確認する前に、 がビューワーからリクエスト CloudFront を受信するたびに関数が実行されます。

次の擬似コードは、HTTP リクエストを変更する関数の構造を示しています。

```
function handler(event) {
    var request = event.request;

    // Modify the request object here.

    return request;
}
```

この関数は、変更されたrequestオブジェクトを に返します CloudFront。CloudFront は CloudFront キャッシュヒットをチェックし、必要に応じてリクエストをオリジンに送信することで、返されたリクエストの処理を続行します。

event、request オブジェクトの構造の詳細については、「[イベントの構造](#)」を参照してください。

ビューワーリクエストイベントタイプで HTTP レスポンスを生成する

関数は、エッジで HTTP レスポンスを生成し、キャッシュされたレスポンスや によるそれ以上の処理をチェックすることなく、ビューワー (クライアント) に直接返すことができます CloudFront。たとえば、関数コードは、リクエストを新しい URL にリダイレクトしたり、承認をチェックして、401 や 403 レスポンスを非承認リクエストに返す場合があります。

HTTP レスポンスを生成する関数を作成する場合は、必ずビューワーリクエストイベントタイプを選択してください。つまり、 がビューワーからリクエスト CloudFront を受信するたびに、CloudFront がリクエストの処理を実行する前に、関数が実行されます。

次の擬似コードは、HTTP 応答を生成する関数の構造を示しています。

```
function handler(event) {
    var request = event.request;

    var response = ...; // Create the response object here,
                        // using the request properties if needed.

    return response;
}
```

この関数は response オブジェクトを に返します。これは CloudFront CloudFront、 CloudFront キャッシュを確認したり、オリジンにリクエストを送信したりすることなく、すぐにビューワーに返します。

event、request、response オブジェクトの構造の詳細については、「[イベントの構造](#)」を参照してください。

ビューワーレスポンスイベントタイプの HTTP レスポンスの変更

関数は、レスポンスが CloudFront キャッシュまたはオリジンのいずれから送信されたかに関係なく、 がビューワー (クライアント) CloudFront に送信する前に HTTP レスポンスを変更できます。例えば、関数コードでレスポンスヘッダー、ステータスコード、本文のコンテンツを追加または変更する場合があります。

HTTP レスポンスを変更する関数を作成する場合は、必ずビューワーレスポンスイベントタイプを選択してください。つまり、関数は、レスポンスが CloudFront キャッシュまたはオリジンのいずれから送信されたかに関係なく、 がビューワーにレスポンスを CloudFront 返す前に実行されます。

次の擬似コードは、HTTP レスポンスを変更する関数の構造を示しています。

```
function handler(event) {
  var request = event.request;
  var response = event.response;

  // Modify the response object here,
  // using the request properties if needed.

  return response;
}
```

この関数は、変更されたresponseオブジェクトを に返し CloudFront、すぐに CloudFrontビューワに返します。

event、response オブジェクトの構造の詳細については、「[イベントの構造](#)」を参照してください。

CloudFront Functions の関数コードの記述の詳細については、[イベントの構造](#)「」、[JavaScript ランタイム機能](#)「」、および「[サンプルのコード](#)」を参照してください。

CloudFront Functions イベント構造

CloudFront 関数は、関数の実行時にeventオブジェクトを入力として関数コードに渡します。[関数をテスト](#)するときには、event オブジェクトを作成し、関数に渡します。関数をテストするために event オブジェクトを作成する場合は、context オブジェクト内のdistributionDomainName、distributionId、requestId フィールドを省略できます。ヘッダーの名前が小文字であることを確認してください。これは、CloudFront Functions が本番環境で関数に渡す event オブジェクトでは常に当てはまります。

次に、このイベントオブジェクトの構造の概要を示します。詳細については、次のトピックを参照してください。

```
{
  "version": "1.0",
  "context": {
    <context object>
  },
  "viewer": {
    <viewer object>
  },
  "request": {
    <request object>
  }
}
```

```
    },  
    "response": {  
      <response object>  
    }  
  }  
}
```

トピック

- [バージョンフィールド](#)
- [コンテキストオブジェクト](#)
- [ビューワーオブジェクト](#)
- [リクエストオブジェクト](#)
- [レスポンスオブジェクト](#)
- [ステータスコードと本文](#)
- [クエリ文字列、ヘッダー、または Cookie の構造](#)
- [レスポンスオブジェクトの例](#)
- [イベントオブジェクトの例](#)

バージョンフィールド

version フィールドには、CloudFront 関数イベントオブジェクトのバージョンを指定する文字列が含まれています。現在のバージョンは 1.0 です。

コンテキストオブジェクト

context オブジェクトには、イベントに関するコンテキスト情報が含まれます。次のフィールドが含まれています。

distributionDomainName

イベントに関連付けられているディストリビューションの CloudFront ドメイン名 (d1111111abcdef8.cloudfront.net など)。

distributionId

イベントに関連付けられたディストリビューションの ID (例: EDFDVBD6EXAMPLE)。

eventType

イベントタイプ (viewer-request または viewer-response)。

requestId

CloudFront リクエスト (および関連するレスポンス) を一意に識別する文字列。

ビューワーオブジェクト

viewer オブジェクトには、リクエストを送信したビューワー (クライアント) の IP アドレスを値とする ip フィールドが含まれています。ビューワーリクエストが HTTP プロキシまたはロードバランサーを通して来た場合、値はプロキシまたはロードバランサーの IP アドレスです。

リクエストオブジェクト

request オブジェクトには、ビューワーから CloudFront HTTP へのリクエストの表現が含まれています。関数に渡される event オブジェクトでは、request オブジェクトは、ビューワーから CloudFront 受信した実際のリクエストを表します。

関数コードが request オブジェクトを返す場合は CloudFront、同じ構造を使用する必要があります。

request オブジェクトには、以下のフィールドが含まれています。

method

リクエストの HTTP メソッド。関数コードが request を返す場合、このフィールドは変更できません。これは、request オブジェクト内唯一の読み取り専用フィールドです。

uri

リクエストされたオブジェクトの相対パス。関数が uri 値を変更する場合、次の点に注意してください。

- 新しい uri 値は、フォワードスラッシュ (/) で始まる必要があります。
- 関数で uri 値を変更すると、ビューワーがリクエストしているオブジェクトが変更されます。
- 関数で uri 値を変更しても、リクエストのキャッシュ動作やオリジンリクエストの送信先は変わりません。

querystring

リクエストのクエリ文字列を表すオブジェクト。リクエストにクエリ文字列が含まれていない場合でも、request オブジェクトには空の querystring オブジェクトが含まれています。

queryString オブジェクトには、リクエストのクエリ文字列パラメータ 1 つにつき 1 つのフィールドが含まれます。

headers

リクエストの HTTP ヘッダーを表すオブジェクト。リクエストに Cookie ヘッダーが含まれている場合、それらのヘッダーは headers オブジェクトの一部ではありません。Cookie は cookies オブジェクトで個別に表示されます。

headers オブジェクトには、リクエストのヘッダー 1 つにつき 1 つのフィールドが含まれます。ヘッダー名は、イベントオブジェクトでは小文字に変換されます。また、関数コードで追加する場合、ヘッダー名を小文字にする必要があります。CloudFront Functions がイベントオブジェクトを HTTP リクエストに変換し直すと、ヘッダー名の各単語の最初の文字が大文字になります。各単語はハイフン (-) で区切られます。例えば、関数コードが という名前のヘッダーを追加した場合 example-header-name、はこれを HTTP リクエスト Example-Header-Name の CloudFront に変換します。

cookies

リクエスト (Cookie ヘッダー) の Cookie を表すオブジェクト。

cookies オブジェクトには、リクエストの Cookie 1 つにつき 1 つのフィールドが含まれます。

クエリ文字列、ヘッダーおよび Cookie の構造の詳細については、「[クエリ文字列、ヘッダー、または Cookie の構造](#)」を参照してください。

event オブジェクトの例については、「[イベントオブジェクトの例](#)」を参照してください。

レスポンスオブジェクト

response オブジェクトには、-to CloudFront-viewer HTTP レスポンスの表現が含まれています。関数に渡される event オブジェクトでは、response オブジェクトはビューワーリクエストに対する CloudFront の実際のレスポンスを表します。

関数コードが response オブジェクトを返す場合は、これと同じ構造を使用する必要があります。

response オブジェクトには、以下のフィールドが含まれています。

statusCode

レスポンスの HTTP ステータスコード。この値は文字列ではなく整数です。

関数は `statusCode` を生成または変更できます。

statusDescription

レスポンスの HTTP ステータスの説明。関数コードがレスポンスを生成する場合、このフィールドはオプションです。

headers

レスポンスの HTTP ヘッダーを表すオブジェクト。レスポンスに `Set-Cookie` ヘッダーが含まれている場合、それらのヘッダーは `headers` オブジェクトの一部ではありません。Cookie は `cookies` オブジェクトで個別に表示されます。

`headers` オブジェクトには、レスポンス内のヘッダー 1 つにつき 1 つのフィールドが含まれます。ヘッダー名は、イベントオブジェクトでは小文字に変換されます。また、関数コードで追加する場合、ヘッダー名を小文字にする必要があります。CloudFront Functions がイベントオブジェクトを HTTP レスポンスに変換し直すと、ヘッダー名の各単語の最初の文字が大文字になります。各単語はハイフン (-) で区切られます。例えば、関数コードが `example-header-name` という名前のヘッダーを追加すると `example-header-name`、はこれを HTTP レスポンス `Example-Header-Name` の CloudFront に変換します。

cookies

レスポンス (`Set-Cookie` ヘッダー) で Cookie を表すオブジェクト。

`cookies` オブジェクトには、レスポンスの Cookie 1 つにつき 1 つのフィールドが含まれます。

body

`body` フィールドの追加はオプションで、関数で指定しない限り `response` オブジェクトには表示されません。関数は、キャッシュまたはオリジンによって CloudFront 返された元の本文にアクセスできません。ビューワーレスポンス関数で `body` フィールドを指定しない場合、CloudFront キャッシュまたはオリジンによって返された元の本文がビューワーに返されます。

カスタム本文をビューワーに返 CloudFront す場合は、`data` フィールドに本文の内容を指定し、`encoding` フィールドに本文エンコーディングを指定します。エンコーディングは、プレーンテキスト ("`encoding`": "`text`") または Base64 でエンコードされたコンテンツ ("`encoding`": "`base64`") として指定できます。

ショートカットとして、本文の内容を `body` フィールド ("`body`": "<specify the body content here>") で直接指定することもできます。この場合、`data` および `fieldsencoding`. CloudFront treat を省略すると、本文はプレーンテキストとして表示されます。

encoding

body コンテンツ (data フィールド) のエンコーディング。有効なエンコードは text と base64 のみです。

encoding base64 として を指定しても本文が有効な base64 ではない場合、 はエラー CloudFront を返します。

data

body コンテンツ。

変更されたステータスコードと本文の内容の詳細については、[ステータスコードと本文](#) を参照してください。

ヘッダーと Cookie の構造の詳細については、「[クエリ文字列、ヘッダー、または Cookie の構造](#)」を参照してください。

response オブジェクトの例については、「[レスポンスオブジェクトの例](#)」を参照してください。

ステータスコードと本文

CloudFront Functions を使用すると、ビューワーレスポンスステータスコードの更新、レスポンス本文全体の新しいコードへの置換、レスポンス本文の削除を行うことができます。キャッシュまたはオリジンからの CloudFront レスポンスの側面を評価した後にビューワーレスポンスを更新する一般的なシナリオには、次のようなものがあります。

- ステータスを変更して HTTP 200 ステータスコードを設定し、ビューワーに返す静的な本文コンテンツを作成する。
- HTTP 301 または 302 ステータスコードを設定して、ユーザーを別のウェブサイトへリダイレクトする。
- ビューワーレスポンスの本文を配信するか削除するかを決定します。

Note

オリジンが 400 以上の HTTP エラーを返した場合、CloudFront 関数は実行されません。詳細については、[すべてのエッジ機能に対する制限](#)を参照してください。

HTTP レスポンスを使用する場合、CloudFront Functions はレスポンス本文にアクセスできません。必要な値に設定することで本文コンテンツを置き換えたり、値を空に設定することで本文を削除したりできます。関数の本文フィールドを更新しない場合、CloudFront キャッシュまたはオリジンによって返された元の本文がビューワーに返されます。

Tip

CloudFront Functions を使用して本文を置き換える場合は、`content-encoding`、`content-type` または `content-length` などの対応するヘッダーを新しい本文のコンテンツに合わせるようにしてください。

例えば、CloudFront オリジンまたはキャッシュが返した `content-encoding: gzip` が、ビューワーレスポンス関数がプレーンテキストの本文を設定する場合、関数はそれに応じてヘッダー `content-encoding` と `content-type` ヘッダーも変更する必要があります。

400 以上の HTTP エラーを返すように CloudFront 関数が設定されている場合、ビューワーには同じステータスコードに対して指定した [カスタムエラーページ](#) が表示されません。

クエリ文字列、ヘッダー、または Cookie の構造

クエリ文字列、ヘッダー、Cookie は同じ構造を共有します。クエリ文字列は、リクエストに表示される場合があります。ヘッダーは、リクエストとレスポンスに表示されます。Cookie は、リクエストとレスポンスに表示されます。

クエリ文字列、ヘッダーおよび Cookie はすべて、親 `querystring`、`headers`、`cookies` オブジェクトで一意的なフィールドです。フィールド名は、クエリ文字列、ヘッダー、または Cookie の名前です。各フィールドには、クエリ文字列、ヘッダー、Cookie の値を持つ `value` プロパティが含まれます。

トピック

- [クエリ文字列値またはクエリ文字列オブジェクト](#)
- [ヘッダーに関する特別な考慮事項](#)
- [重複するクエリ文字列、ヘッダー、Cookie \(multiValue 配列\)](#)
- [Cookie 属性](#)

クエリ文字列値またはクエリ文字列オブジェクト

関数は、クエリ文字列オブジェクトに加えてクエリ文字列値を返すことができます。クエリ文字列値を使用して、クエリ文字列パラメータを任意のカスタム順序で配置できます。例えば、関数コードでクエリ文字列を変更するには、次のようなコードを使用します。

```
var request = event.request;
request.querystring =
  'ID=42&Exp=1619740800&TTL=1440&NoValue=&querymv=val1&querymv=val2,val3';
```

ヘッダーに関する特別な考慮事項

ヘッダーのみの場合、ヘッダー名がイベントオブジェクトで小文字に変換されます。また、関数コードで追加する場合は、ヘッダー名を小文字にする必要があります。CloudFront Functions がイベントオブジェクトを HTTP リクエストまたはレスポンスに変換し直すと、ヘッダー名の各単語の最初の文字が大文字になります。各単語はハイフン (-) で区切られます。例えば、関数コードがという名前のヘッダーを追加すると `example-header-name`、は HTTP リクエストまたはレスポンス `Example-Header-Name` でこれを CloudFront に変換します。

たとえば、HTTP リクエストで次の Host ヘッダーを考えてみましょう。

```
Host: video.example.com
```

このヘッダーは、`request` オブジェクトで次のように表されます。

```
"headers": {
  "host": {
    "value": "video.example.com"
  }
}
```

関数コードの Host ヘッダーにアクセスするには、次のようなコードを使用します。

```
var request = event.request;
var host = request.headers.host.value;
```

関数コードでヘッダーを追加または変更するには、次のようなコードを使用します (このコードは、`X-Custom-Header` 値で `example value` という名前のヘッダーを追加します)。

```
var request = event.request;
```

```
request.headers['x-custom-header'] = {value: 'example value'};
```

重複するクエリ文字列、ヘッダー、Cookie (**multiValue** 配列)

HTTP リクエストまたはレスポンスには、同じ名前のクエリ文字列、ヘッダー、Cookie が含まれることがあります。この場合、重複するクエリ文字列、ヘッダー、Cookie は `request` または `response` オブジェクトの 1 つのフィールドに折りたたまれていますが、このフィールドには `multiValue` という名前の追加のプロパティが含まれます。`multiValue` プロパティには、重複するクエリ文字列、ヘッダー、Cookie の各値を含む配列が含まれます。

たとえば、次の `Accept` ヘッダーを持つ HTTP リクエストを考えてみましょう。

```
Accept: application/json
Accept: application/xml
Accept: text/html
```

これらのヘッダーは、`request` オブジェクトで次のように表されます。

```
"headers": {
  "accept": {
    "value": "application/json",
    "multiValue": [
      {
        "value": "application/json"
      },
      {
        "value": "application/xml"
      },
      {
        "value": "text/html"
      }
    ]
  }
}
```

最初のヘッダー値 (この場合は `application/json`) は、`value`、`multiValue` プロパティの両方で繰り返されています。これにより、`multiValue` 配列をループしてすべての値にアクセスできます。

関数コードが `multiValue` 配列を持つクエリ文字列、ヘッダー、または Cookie を変更する場合、CloudFront Functions は次のルールを使用して変更を適用します。

1. multiValue 配列が存在し、変更がある場合は、その変更が適用されます。value プロパティの最初の要素は無視されます。
2. それ以外の場合は、value プロパティへの変更が適用され、それ以降の値 (存在する場合) は変更されません。

この multiValue プロパティは、前の例に示すように、HTTP リクエストまたはレスポンスに同じ名前の重複するクエリ文字列、ヘッダー、Cookie のいずれかが含まれている場合にのみ使用されます。ただし、1つのクエリ文字列、ヘッダー、または Cookie に複数の値がある場合、multiValue プロパティは使用されません。

たとえば、次の例のように、一つの Accept ヘッダーに3つの値が含まれるリクエストについて考えてみます。

```
Accept: application/json, application/xml, text/html
```

このヘッダーは、request オブジェクトで次のように表されます。

```
"headers": {
  "accept": {
    "value": "application/json, application/xml, text/html"
  }
}
```

Cookie 属性

HTTP レスポンスの Set-Cookie ヘッダーでは、ヘッダーに Cookie の名前と値のペア、および必要に応じてセミコロンで区切られた属性のセットが含まれます。次に例を示します。

```
Set-Cookie: cookie1=val1; Secure; Path=/; Domain=example.com; Expires=Wed, 05 Apr 2021
07:28:00 GMT"
```

response オブジェクトでは、これらの属性は Cookie フィールドの attributes プロパティで表されます。たとえば、前の Set-Cookie ヘッダーは次のように表されます。

```
"cookie1": {
  "value": "val1",
  "attributes": "Secure; Path=/; Domain=example.com; Expires=Wed, 05 Apr 2021
07:28:00 GMT"
```

```
}
```

レスポンスオブジェクトの例

次の例は、本文がビューワーレスポンス関数に置き換えられた response オブジェクト (ビューワーレスポンス関数の出力) を示しています。

```
{
  "response": {
    "statusCode": 200,
    "statusDescription": "OK",
    "headers": {
      "date": {
        "value": "Mon, 04 Apr 2021 18:57:56 GMT"
      },
      "server": {
        "value": "gunicorn/19.9.0"
      },
      "access-control-allow-origin": {
        "value": "*"
      },
      "access-control-allow-credentials": {
        "value": "true"
      },
      "content-type": {
        "value": "text/html"
      },
      "content-length": {
        "value": "86"
      }
    },
    "cookies": {
      "ID": {
        "value": "id1234",
        "attributes": "Expires=Wed, 05 Apr 2021 07:28:00 GMT"
      },
      "Cookie1": {
        "value": "val1",
        "attributes": "Secure; Path=/; Domain=example.com; Expires=Wed, 05 Apr 2021 07:28:00 GMT",
        "multiValue": [
          {
            "value": "val1",
```

```
        "attributes": "Secure; Path=/; Domain=example.com; Expires=Wed, 05 Apr 2021
07:28:00 GMT"
      },
      {
        "value": "val2",
        "attributes": "Path=/cat; Domain=example.com; Expires=Wed, 10 Jan 2021
07:28:00 GMT"
      }
    ]
  }
},

// Adding the body field is optional and it will not be present in the response
object
// unless you specify it in your function.
// Your function does not have access to the original body returned by the
CloudFront
// cache or origin.
// If you don't specify the body field in your viewer response function, the
original
// body returned by the CloudFront cache or origin is returned to viewer.

  "body": {
    "encoding": "text",
    "data": "<!DOCTYPE html><html><body><p>Here is your custom content.</p></body></
html>"
  }
}
}
```

イベントオブジェクトの例

以下は、完全な event オブジェクトの例です。

Note

event オブジェクトは関数への入力です。関数は、event オブジェクト全体ではなく、request または response オブジェクトだけを返します。

```
{
  "version": "1.0",
```

```
"context": {
  "distributionDomainName": "d111111abcdef8.cloudfront.net",
  "distributionId": "EDFDVBD6EXAMPLE",
  "eventType": "viewer-response",
  "requestId": "EXAMPLEentjQpEXAMPLE_SG5Z-EXAMPLEPmPfEXAMPLEu3EqEXAMPLE=="
},
"viewer": {"ip": "198.51.100.11"},
"request": {
  "method": "GET",
  "uri": "/media/index.mpd",
  "querystring": {
    "ID": {"value": "42"},
    "Exp": {"value": "1619740800"},
    "TTL": {"value": "1440"},
    "NoValue": {"value": ""},
    "querymv": {
      "value": "val1",
      "multiValue": [
        {"value": "val1"},
        {"value": "val2,val3"}
      ]
    }
  ]
},
"headers": {
  "host": {"value": "video.example.com"},
  "user-agent": {"value": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:83.0) Gecko/20100101 Firefox/83.0"},
  "accept": {
    "value": "application/json",
    "multiValue": [
      {"value": "application/json"},
      {"value": "application/xml"},
      {"value": "text/html"}
    ]
  ],
  "accept-language": {"value": "en-GB,en;q=0.5"},
  "accept-encoding": {"value": "gzip, deflate, br"},
  "origin": {"value": "https://website.example.com"},
  "referer": {"value": "https://website.example.com/videos/12345678?
action=play"},
  "cloudfront-viewer-country": {"value": "GB"}
},
"cookies": {
  "Cookie1": {"value": "value1"},
```

```
    "Cookie2": {"value": "value2"},
    "cookie_consent": {"value": "true"},
    "cookiemv": {
      "value": "value3",
      "multiValue": [
        {"value": "value3"},
        {"value": "value4"}
      ]
    }
  },
  "response": {
    "statusCode": 200,
    "statusDescription": "OK",
    "headers": {
      "date": {"value": "Mon, 04 Apr 2021 18:57:56 GMT"},
      "server": {"value": "unicorn/19.9.0"},
      "access-control-allow-origin": {"value": "*"},
      "access-control-allow-credentials": {"value": "true"},
      "content-type": {"value": "application/json"},
      "content-length": {"value": "701"}
    },
    "cookies": {
      "ID": {
        "value": "id1234",
        "attributes": "Expires=Wed, 05 Apr 2021 07:28:00 GMT"
      },
      "Cookie1": {
        "value": "val1",
        "attributes": "Secure; Path=/; Domain=example.com; Expires=Wed, 05 Apr
2021 07:28:00 GMT",
        "multiValue": [
          {
            "value": "val1",
            "attributes": "Secure; Path=/; Domain=example.com; Expires=Wed,
05 Apr 2021 07:28:00 GMT"
          },
          {
            "value": "val2",
            "attributes": "Path=/cat; Domain=example.com; Expires=Wed, 10
Jan 2021 07:28:00 GMT"
          }
        ]
      }
    }
  }
}
```

```
    }  
  }  
}
```

JavaScript CloudFront Functions の ランタイム機能

Amazon CloudFront Functions JavaScript ランタイム環境は [ECMAScript \(ES\) バージョン 5.1](#) に準拠しており、ES バージョン 6~12 の一部の機能をサポートしています。

最新の機能を利用するには、ランタイム 2.0 を使用することをお勧めします。ランタイム 2.0 は 1.0 と比較して以下の変更点があることに注意してください。

- バッファーマジュールメソッドが利用可能です。
- 以下の非標準の文字列プロトタイプメソッドは使用できません。
 - `String.prototype.bytesFrom()`
 - `String.prototype.fromBytes()`
 - `String.prototype.fromUTF8()`
 - `String.prototype.toBytes()`
 - `String.prototype.toUTF8()`
- 暗号モジュールには次の変更があります。
 - `hash.digest()` - エンコーディングが指定されていない場合、戻り型が `Buffer` に変更されました
 - `hmac.digest()` - エンコーディングが指定されていない場合、戻り型が `Buffer` に変更されました
- その他の新機能については、[JavaScript CloudFront Functions の ランタイム 2.0 機能](#) に記載されています。

トピック

- [JavaScript CloudFront Functions の ランタイム 1.0 機能](#)
- [JavaScript CloudFront Functions の ランタイム 2.0 機能](#)

JavaScript CloudFront Functions のランタイム 1.0 機能

CloudFront Functions JavaScript ランタイム環境は [ECMAScript \(ES\) バージョン 5.1](#) に準拠しており、ES バージョン 6~9 の一部の機能もサポートしています。また、ES 仕様に含まれない非標準メソッドも提供しています。次のトピックでは、サポートされるすべての言語機能の一覧を示します。

トピック

- [主要機能](#)
- [プリミティブオブジェクト](#)
- [ビルトインオブジェクト](#)
- [エラーのタイプ](#)
- [Globals](#)
- [ビルトインモジュール](#)
- [制限された機能](#)

主要機能

ES の次の主要機能がサポートされています。

Types]

すべての ES 5.1 タイプでサポートされています。これには、ブール値、数値、文字列、オブジェクト、配列、関数、関数コンストラクタ、正規表現が含まれます。

演算子

すべての ES 5.1 演算子でサポートされています。

ES 7 指数演算子 (**) がサポートされています。

ステートメント

Note

const および let ステートメントはサポートされていません。

次の ES 5.1 ステートメントがサポートされています。

- break

- `catch`
- `continue`
- `do-while`
- `else`
- `finally`
- `for`
- `for-in`
- `if`
- `return`
- `switch`
- `throw`
- `try`
- `var`
- `while`
- ラベル付きステートメント

リテラル

ES 6 テンプレートリテラル (複数行の文字列、式の補間、および入れ子テンプレート) がサポートされています。

関数

すべての ES 5.1 機能がサポートされています。

ES 6 のアロー関数、ES 6 のレストパラメータ (残余因数) 構文がサポートされています。

Unicode

ソーステキストおよび文字列リテラルには、Unicode でエンコードされた文字を含めることができます。6 文字の Unicode エスケープシーケンス (コードポイント、例: `\uXXXX`) もサポートされています。

Strict モード

関数は Strict モードで動作するため、関数コードに `use strict` ステートメントを追加する必要はありません。これは変更できません。

プリミティブオブジェクト

以下の ES プリミティブオブジェクトがサポートされています。

オブジェクト

オブジェクトについて以下の ES 5.1 メソッドがサポートされています。

- `create` (プロパティリストなし)
- `defineProperties`
- `defineProperty`
- `freeze`
- `getOwnPropertyDescriptor`
- `getOwnPropertyNames`
- `getPrototypeOf`
- `hasOwnProperty`
- `isExtensible`
- `isFrozen`
- `prototype.isPrototypeOf`
- `isSealed`
- `keys`
- `preventExtensions`
- `prototype.propertyIsEnumerable`
- `seal`
- `prototype.toString`
- `prototype.valueOf`

オブジェクトについて以下の ES 6 メソッドがサポートされています。

- `assign`
- `is`
- `prototype.setPrototypeOf`

オブジェクトについて以下の ES 8 メソッドがサポートされています。

- `entries`
- `values`

文字列

文字列について以下の ES 5.1 メソッドがサポートされています。

- `fromCharCode`
- `prototype.charAt`
- `prototype.concat`
- `prototype.indexOf`
- `prototype.lastIndexOf`
- `prototype.match`
- `prototype.replace`
- `prototype.search`
- `prototype.slice`
- `prototype.split`
- `prototype.substr`
- `prototype.substring`
- `prototype.toLowerCase`
- `prototype.trim`
- `prototype.toUpperCase`

文字列について以下の ES 6 メソッドがサポートされています。

- `fromCodePoint`
- `prototype.codePointAt`
- `prototype.endsWith`
- `prototype.includes`
- `prototype.repeat`
- `prototype.startsWith`

文字列について以下の ES 8 メソッドがサポートされています。

- `prototype.padStart`

- `prototype.padEnd`

文字列について以下の ES 9 メソッドがサポートされています。

- `prototype.trimStart`
- `prototype.trimEnd`

文字列について以下の非標準メソッドがサポートされています。

- `prototype.bytesFrom(array | string, encoding)`

オクテット列またはエンコードされた文字列からバイト文字列を作成します。文字列エンコーディングオプションは `hex`、`base64`、`base64url` です。

- `prototype.fromBytes(start[, end])`

バイト文字列から Unicode 文字列を作成します。各バイトは、対応する Unicode コードポイントで置き換えられます。

- `prototype.fromUTF8(start[, end])`

UTF-8 でエンコードされたバイト文字列から Unicode 文字列を作成します。エンコーディングが正しくない場合は、`null` が返されます。

- `prototype.toBytes(start[, end])`

Unicode 文字列からバイト文字列を作成します。すべての文字は `[0,255]` の範囲内にある必要があります。そうでない場合は、`null` が返されます。

- `prototype.toUTF8(start[, end])`

Unicode 文字列から UTF-8 でエンコードされたバイト文字列を作成します。

数値

番号に関するすべての ES 5.1 メソッドがサポートされています。

番号について以下の ES 6 メソッドがサポートされています。

- `isFinite`
- `isInteger`
- `isNaN`
- `isSafeInteger`
- `parseFloat`

- `parseInt`
- `prototype.toExponential`
- `prototype.toFixed`
- `prototype.toPrecision`
- `EPSILON`
- `MAX_SAFE_INTEGER`
- `MAX_VALUE`
- `MIN_SAFE_INTEGER`
- `MIN_VALUE`
- `NEGATIVE_INFINITY`
- `NaN`
- `POSITIVE_INFINITY`

ビルトインオブジェクト

ES の以下のビルトインオブジェクトがサポートされています。

Math

ES 5.1 のすべての Math メソッドがサポートされています。

Note

CloudFront Functions ランタイム環境では、この `Math.random()` 実装は、関数の実行時のタイムスタンプが `arc4random` シードされた OpenBSD を使用します。

以下の ES 6 Math メソッドがサポートされています。

- `acosh`
- `asinh`
- `atanh`
- `cbrt`
- `clz32`

- cosh
- expm1
- fround
- hypot
- imul
- log10
- log1p
- log2
- sign
- sinh
- tanh
- trunc
- E
- LN10
- LN2
- LOG10E
- LOG2E
- PI
- SQRT1_2
- SQRT2

日付

すべての ES 5.1 の Date 機能がサポートされています。

Note

セキュリティ上の理由から、Date は、単一の関数実行の有効期間中、常に同じ値 (関数の開始時間) を返します。詳細については、「[制限された機能](#)」を参照してください。

関数

apply、bind、call メソッドがサポートされています。

関数コンストラクタはサポートされていません。

正規表現

すべての ES 5.1 の正規表現機能がサポートされています。正規表現言語は Perl 互換です。ES 9 の名前付きキャプチャグループがサポートされています。

JSON

parse、stringify を含むすべての ES 5.1 JSON 機能がサポートされています。

配列

配列について以下の ES 5.1 メソッドがサポートされています。

- `isArray`
- `prototype.concat`
- `prototype.every`
- `prototype.filter`
- `prototype.forEach`
- `prototype.indexOf`
- `prototype.join`
- `prototype.lastIndexOf`
- `prototype.map`
- `prototype.pop`
- `prototype.push`
- `prototype.reduce`
- `prototype.reduceRight`
- `prototype.reverse`
- `prototype.shift`
- `prototype.slice`
- `prototype.some`
- `prototype.sort`
- `prototype.splice`
- `prototype.unshift`

配列について以下の ES 6 メソッドがサポートされています。

- `of`
- `prototype.copyWithIn`
- `prototype.fill`
- `prototype.find`
- `prototype.findIndex`

配列について以下の ES 7 メソッドがサポートされています。

- `prototype.includes`

型付き配列

以下の ES 6 型付き配列がサポートされています。

- `Int8Array`
- `Uint8Array`
- `Uint8ClampedArray`
- `Int16Array`
- `Uint16Array`
- `Int32Array`
- `Uint32Array`
- `Float32Array`
- `Float64Array`
- `prototype.copyWithIn`
- `prototype.fill`
- `prototype.join`
- `prototype.set`
- `prototype.slice`
- `prototype.subarray`
- `prototype.toString`

ArrayBuffer

ArrayBuffer について以下のメソッドがサポートされています。

- `prototype.isView`
- `prototype.slice`

promise

promise について以下のメソッドがサポートされています。

- `reject`
- `resolve`
- `prototype.catch`
- `prototype.finally`
- `prototype.then`

Crypto

暗号モジュールは、標準のハッシュおよびハッシュベースのメッセージ認証コード (HMAC) ヘルパーを提供します。 `require('crypto')` を使用してモジュールをロードできます。モジュールは、Node.js の相対物とまったく同じように動作する以下のメソッドを公開します。

- `createHash(algorithm)`
- `hash.update(data)`
- `hash.digest([encoding])`
- `createHmac(algorithm, secret key)`
- `hmac.update(data)`
- `hmac.digest([encoding])`

詳細については、「ビルトインモジュールセクション」の「[Crypto \(ハッシュと HMAC\)](#)」を参照してください。

コンソール

これはデバッグ用のヘルパーオブジェクトです。ログメッセージを記録するための `log()` メソッドのみサポートしています。

Note

CloudFront 関数は、などのカンマ構文をサポートしていません `console.log('a', 'b')`。代わりに、 `console.log('a' + ' ' + 'b')`形式を使用します。

エラーのタイプ

以下のエラーオブジェクトがサポートされています。

- Error
- EvalError
- InternalError
- MemoryError
- RangeError
- ReferenceError
- SyntaxError
- TypeError
- URIError

Globals

globalThis オブジェクトはサポートされています。

以下の ES 5.1 グローバル関数がサポートされています。

- decodeURI
- decodeURIComponent
- encodeURI
- encodeURIComponent
- isFinite
- isNaN
- parseFloat
- parseInt

以下のグローバル定数がサポートされています。

- NaN
- Infinity
- undefined

ビルトインモジュール

以下のビルトインモジュールがサポートされています。

モジュール

- [Crypto \(ハッシュと HMAC\)](#)
- [クエリ文字列](#)

Crypto (ハッシュと HMAC)

暗号モジュール (crypto) は、標準のハッシュおよびハッシュベースのメッセージ認証コード (HMAC) ヘルパーを提供します。require('crypto') を使用してモジュールをロードできます。このモジュールは、Node.js の相対物とまったく同じように動作する以下のメソッドを提供します。

ハッシュメソッド

`crypto.createHash(algorithm)`

ハッシュオブジェクトを作成して返します。このハッシュオブジェクトは、指定されたアルゴリズム (md5、sha1、sha256 のいずれか) を使用してハッシュダイジェストの生成に使用できません。

`hash.update(data)`

指定された *data* を使用してハッシュコンテンツを更新します。

`hash.digest([encoding])`

`hash.update()` を使用して渡されたすべてのデータのダイジェストを計算します。エンコードは hex、base64、base64url のいずれかを使用します。

HMACメソッド

`crypto.createHmac(algorithm, secret key)`

指定された *algorithm* と *secret key* を使用する HMAC オブジェクトを作成して返します。アルゴリズムは md5、sha1、sha256 のいずれかを使用します。

`hmac.update(data)`

指定された *data* を使用して HMAC コンテンツを更新します。

```
hmac.digest([encoding])
```

`hmac.update()` を使用して渡されたすべてのデータのダイジェストを計算します。エンコードは `hex`、`base64`、`base64url` のいずれかを使用します。

クエリ文字列

Note

[CloudFront Functions イベントオブジェクト](#) は、URL クエリ文字列を自動的に解析します。つまり、ほとんどの場合、このモジュールを使用する必要はありません。

クエリ文字列モジュール (`querystring`) は、URL クエリ文字列を解析および書式設定するためのメソッドを提供します。`require('querystring')` を使用してモジュールをロードできます。このモジュールは、以下のメソッドを提供します。

```
querystring.escape(string)
```

URL は `string` をエンコードし、エスケープしたクエリ文字列を返します。このメソッドは `querystring.stringify()` で使用するため、直接使用しないでください。

```
querystring.parse(string [, separator [, equal [, options]])
```

クエリ文字列 (`string`) を解析し、オブジェクトを返します。

`separator` パラメータは、クエリ文字列のキーと値のペアを区切る substring です。デフォルトでは、`&` です。

`equal` パラメータは、クエリ文字列のキーと値を区切る substring です。デフォルトでは、`=` です。

`options` パラメータは、以下のキーを持つオブジェクトです。

`decodeURIComponent` *function*

クエリ文字列のパーセントエンコーディングされた文字を decode する関数です。デフォルトでは、`querystring.unescape()` です。

`maxKeys` *number*

解析するキーの最大数。デフォルトでは、`1000` です。キーカウントの制限を解除するには、`0` の値を使用します。

デフォルトでは、クエリ文字列のパーセントエンコーディングされた文字は、UTF-8 エンコーディングを使用していると見なされます。無効な UTF-8 シーケンスは、U+FFFD 置換文字に置き換えられます。

たとえば、次のクエリ文字列の場合:

```
'name=value&abc=xyz&abc=123'
```

`querystring.parse()` の戻り値は次のとおりです。

```
{
  name: 'value',
  abc: ['xyz', '123']
}
```

`querystring.decode()` は `querystring.parse()` のエイリアスです。

`querystring.stringify(object[, separator[, equal[, options]])`

`object` をシリアル化し、クエリ文字列を返します。

`separator` パラメータは、クエリ文字列のキーと値のペアを区切る substring です。デフォルトでは、& です。

`equal` パラメータは、クエリ文字列のキーと値を区切る substring です。デフォルトでは、= です。

`options` パラメータは、以下のキーを持つオブジェクトです。

`encodeURIComponent` *function*

URL-unsafe 文字をクエリ文字列のパーセントエンコーディングに変換するために使用される関数です。デフォルトでは、`querystring.escape()` です。

デフォルトでは、クエリ文字列でパーセントエンコーディングが必要な文字は UTF-8 としてエンコードされます。別のエンコーディングを使用するには、`encodeURIComponent` オプションを指定します。

以下のコードでの例:

```
querystring.stringify({ name: 'value', abc: ['xyz', '123'], anotherName: '' });
```

戻り値:

```
'name=value&abc=xyz&abc=123&anotherName='
```

`querystring.encode()` は `querystring.stringify()` のエイリアスです。

`querystring.unescape(string)`

指定された `string` 内の URL パーセントエンコーディングされた文字をデコードし、エスケープしていないクエリ文字列を返します。このメソッドは `querystring.parse()` で使用するため、直接使用しないでください。

制限された機能

以下の JavaScript 言語機能は、セキュリティ上の問題によりサポートされていないか、制限されています。

動的コード評価

動的コード評価はサポートされていません。 `eval()`、`Function` 両方のコンストラクタが試行された場合、エラーをスローします。たとえば、`const sum = new Function('a', 'b', 'return a + b')` はエラーをスローします。

タイマー

`setTimeout()`、`setImmediate()`、`clearTimeout()` 関数はサポートされていません。関数実行中に `defer` または `yield` する規定はありません。関数は同期的に実行しないと完了できません。

日付とタイムスタンプ

セキュリティ上の理由から、高解像度タイマーにはアクセスできません。現在の時刻を照会するすべての `Date` メソッドは、単一の関数実行の存続期間中は常に同じ値を返します。返されるタイムスタンプは、関数の実行を開始した時刻です。したがって、関数内で経過時間を測定することはできません。

ファイルシステムへのアクセス

ファイルシステムにはアクセスできません。たとえば、Node.js にあるようなファイルシステムアクセス用の `fs` モジュールはありません。

ネットワークアクセス

ネットワークコールはサポートされていません。たとえば、XHR、HTTP (S)、ソケットはサポートされていません。

JavaScript CloudFront Functions のランタイム 2.0 機能

CloudFront Functions JavaScript ランタイム環境は [ECMAScript \(ES\) バージョン 5.1](#) に準拠しており、ES バージョン 6~12 の一部の機能もサポートしています。また、ES 仕様に含まれない非標準メソッドも提供しています。次のトピックでは、このランタイムでサポートされるすべての機能を一覧表示します。

トピック

- [主要機能](#)
- [プリミティブオブジェクト](#)
- [ビルトインオブジェクト](#)
- [エラーのタイプ](#)
- [Globals](#)
- [ビルトインモジュール](#)
- [制限された機能](#)

主要機能

ES の次の主要機能がサポートされています。

Types]

すべての ES 5.1 タイプでサポートされています。これには、ブール値、数値、文字列、オブジェクト、配列、関数、正規表現が含まれます。

演算子

すべての ES 5.1 演算子でサポートされています。

ES 7 指数演算子 (**) がサポートされています。

ステートメント

次の ES 5.1 ステートメントがサポートされています。

- break
- catch
- continue

- do-while
- else
- finally
- for
- for-in
- if
- label
- return
- switch
- throw
- try
- var
- while

次の ES 6 ステートメントがサポートされています。

- async
- await
- const
- let

 Note

async、await、const、let は JavaScript ランタイム 2.0 で新しく追加されました。

リテラル

ES 6 テンプレートリテラル (複数行の文字列、式の補間、および入れ子テンプレート) がサポートされています。

関数

すべての ES 5.1 機能がサポートされています。

ES 6 のアロー関数、ES 6 のレストパラメータ (残余因数) 構文がサポートされています。

Unicode

ソーステキストおよび文字列リテラルには、Unicode でエンコードされた文字を含めることができます。6 文字の Unicode エスケープシーケンス (コードポイント、例: `\uXXXX`) もサポートされています。

Strict モード

関数は Strict モードで動作するため、関数コードに `use strict` ステートメントを追加する必要はありません。これは変更できません。

プリミティブオブジェクト

以下の ES プリミティブオブジェクトがサポートされています。

オブジェクト

オブジェクトについて以下の ES 5.1 メソッドがサポートされています。

- `Object.create()` (プロパティリストなし)
- `Object.defineProperties()`
- `Object.defineProperty()`
- `Object.freeze()`
- `Object.getOwnPropertyDescriptor()`
- `Object.getOwnPropertyDescriptors()`
- `Object.getOwnPropertyNames()`
- `Object.getPrototypeOf()`
- `Object.isExtensible()`
- `Object.isFrozen()`
- `Object.isSealed()`
- `Object.keys()`
- `Object.preventExtensions()`
- `Object.seal()`

オブジェクトについて以下の ES 6 メソッドがサポートされています。

- `Object.assign()`

オブジェクトについて以下の ES 8 メソッドがサポートされています。

- `Object.entries()`
- `Object.values()`

オブジェクトについて以下の ES 5.1 プロトタイプメソッドがサポートされています。

- `Object.prototype.hasOwnProperty()`
- `Object.prototype.isPrototypeOf()`
- `Object.prototype.propertyIsEnumerable()`
- `Object.prototype.toString()`
- `Object.prototype.valueOf()`

オブジェクトについて以下の ES 6 プロトタイプメソッドがサポートされています。

- `Object.prototype.is()`
- `Object.prototype.setPrototypeOf()`

文字列

文字列について以下の ES 5.1 メソッドがサポートされています。

- `String.fromCharCode()`

文字列について以下の ES 6 メソッドがサポートされています。

- `String.fromCodePoint()`

文字列について以下の ES 5.1 プロトタイプメソッドがサポートされています。

- `String.prototype.charAt()`
- `String.prototype.concat()`
- `String.prototype.indexOf()`
- `String.prototype.lastIndexOf()`
- `String.prototype.match()`
- `String.prototype.replace()`
- `String.prototype.search()`
- `String.prototype.slice()`
- `String.prototype.split()`
- `String.prototype.substr()`

- `String.prototype.substring()`
- `String.prototype.toLowerCase()`
- `String.prototype.trim()`
- `String.prototype.toUpperCase()`

文字列について以下の ES 6 プロトタイプメソッドがサポートされています。

- `String.prototype.codePointAt()`
- `String.prototype.endsWith()`
- `String.prototype.includes()`
- `String.prototype.repeat()`
- `String.prototype.startsWith()`

文字列について以下の ES 8 プロトタイプメソッドがサポートされています。

- `String.prototype.padStart()`
- `String.prototype.padEnd()`

文字列について以下の ES 9 プロトタイプメソッドがサポートされています。

- `String.prototype.trimStart()`
- `String.prototype.trimEnd()`

文字列について以下の ES 12 プロトタイプメソッドがサポートされています。

- `String.prototype.replaceAll()`

 Note

`String.prototype.replaceAll()` は JavaScript、ランタイム 2.0 で新しいです。

数

すべての ES 5 番号がサポートされています。

番号について以下の ES 6 プロパティがサポートされています。

- `Number.EPSILON`
- `Number.MAX_SAFE_INTEGER`

- `Number.MIN_SAFE_INTEGER`
- `Number.MAX_VALUE`
- `Number.MIN_VALUE`
- `Number.NaN`
- `Number.NEGATIVE_INFINITY`
- `Number.POSITIVE_INFINITY`

番号について以下の ES 6 メソッドがサポートされています。

- `Number.isFinite()`
- `Number.isInteger()`
- `Number.isNaN()`
- `Number.isSafeInteger()`
- `Number.parseInt()`
- `Number.parseFloat()`

番号について以下の ES 5.1 プロトタイプメソッドがサポートされています。

- `Number.prototype.toExponential()`
- `Number.prototype.toFixed()`
- `Number.prototype.toPrecision()`

ES 12 数字区切り文字がサポートされています。

 Note

ES 12 数値区切り文字は JavaScript、ランタイム 2.0 で新しく追加されました。

ビルトインオブジェクト

ES の以下のビルトインオブジェクトがサポートされています。

Math

ES 5.1 のすべての Math メソッドがサポートされています。

以下の ES 6 数学的プロパティがサポートされています。

- `Math.E`
- `Math.LN10`
- `Math.LN2`
- `Math.LOG10E`
- `Math.LOG2E`
- `Math.PI`
- `Math.SQRT1_2`
- `Math.SQRT2`

以下の ES 6 Math メソッドがサポートされています。

- `Math.abs()`
- `Math.acos()`
- `Math.acosh()`
- `Math.asin()`
- `Math.asinh()`
- `Math.atan()`
- `Math.atan2()`
- `Math.atanh()`
- `Math.cbrt()`
- `Math.ceil()`
- `Math.clz32()`
- `Math.cos()`
- `Math.cosh()`
- `Math.exp()`
- `Math.expm1()`
- `Math.floor()`
- `Math.fround()`
- `Math.hypot()`
- `Math.imul()`
- `Math.log()`

- `Math.log1p()`
- `Math.log2()`
- `Math.log10()`
- `Math.max()`
- `Math.min()`
- `Math.pow()`
- `Math.random()`
- `Math.round()`
- `Math.sign()`
- `Math.sinh()`
- `Math.sin()`
- `Math.sqrt()`
- `Math.tan()`
- `Math.tanh()`
- `Math.trunc()`

日付

すべての ES 5.1 の Date 機能がサポートされています。

Note

セキュリティ上の理由から、Date は、単一の関数実行の有効期間中、常に同じ値 (関数の開始時間) を返します。詳細については、「[制限された機能](#)」を参照してください。

関数

以下の ES 5.1 プロトタイプメソッドがサポートされています。

- `Function.prototype.apply()`
- `Function.prototype.bind()`
- `Function.prototype.call()`

関数コンストラクタはサポートされていません。

正規表現

すべての ES 5.1 の正規表現機能がサポートされています。正規表現言語は Perl 互換です。

以下の ES 5.1 プロトタイプアクセサプロパティがサポートされています。

- `RegExp.prototype.global`
- `RegExp.prototype.ignoreCase`
- `RegExp.prototype.multiline`
- `RegExp.prototype.source`
- `RegExp.prototype.sticky`
- `RegExp.prototype.flags`

Note

`RegExp.prototype.sticky` と `RegExp.prototype.flags` はランタイム 2.0 で JavaScript 新しく追加されました。

以下の ES 5.1 プロトタイプメソッドがサポートされています。

- `RegExp.prototype.exec()`
- `RegExp.prototype.test()`
- `RegExp.prototype.toString()`
- `RegExp.prototype[@@replace]()`
- `RegExp.prototype[@@split]()`

Note

`RegExp.prototype[@@split]()` は、ランタイム 2.0 で JavaScript 新しいです。

以下の ES 5.1 インスタンスプロパティがサポートされています。

- `lastIndex`

ES 9 の名前付きキャプチャグループがサポートされています。

JSON

以下の ES 5.1 メソッドがサポートされています。

- `JSON.parse()`
- `JSON.stringify()`

配列

配列について以下の ES 5.1 メソッドがサポートされています。

- `Array.isArray()`

配列について以下の ES 6 メソッドがサポートされています。

- `Array.of()`

以下の ES 5.1 プロトタイプメソッドがサポートされています。

- `Array.prototype.concat()`
- `Array.prototype.every()`
- `Array.prototype.filter()`
- `Array.prototype.forEach()`
- `Array.prototype.indexOf()`
- `Array.prototype.join()`
- `Array.prototype.lastIndexOf()`
- `Array.prototype.map()`
- `Array.prototype.pop()`
- `Array.prototype.push()`
- `Array.prototype.reduce()`
- `Array.prototype.reduceRight()`
- `Array.prototype.reverse()`
- `Array.prototype.shift()`
- `Array.prototype.slice()`
- `Array.prototype.some()`
- `Array.prototype.sort()`
- `Array.prototype.splice()`
- `Array.prototype.unshift()`

以下の ES 6 プロトタイプメソッドがサポートされています。

- `Array.prototype.copyWithin()`

- `Array.prototype.fill()`
- `Array.prototype.find()`
- `Array.prototype.findIndex()`

以下の ES 7 プロトタイプメソッドがサポートされています。

- `Array.prototype.includes()`

型付き配列

以下の ES 6 型付き配列コンストラクターがサポートされています。

- `Float32Array`
- `Float64Array`
- `Int8Array`
- `Int16Array`
- `Int32Array`
- `Uint8Array`
- `Uint8ClampedArray`
- `Uint16Array`
- `Uint32Array`

以下の ES 6 メソッドがサポートされています。

- `TypedArray.from()`
- `TypedArray.of()`

Note

`TypedArray.from()` と `TypedArray.of()` は JavaScript ランタイム 2.0 で新しく追加されました。

以下の ES 6 プロトタイプメソッドがサポートされています。

- `TypedArray.prototype.copyWithin()`
- `TypedArray.prototype.every()`
- `TypedArray.prototype.fill()`
- `TypedArray.prototype.filter()`
- `TypedArray.prototype.find()`

- `TypedArray.prototype.findIndex()`
- `TypedArray.prototype.forEach()`
- `TypedArray.prototype.includes()`
- `TypedArray.prototype.indexOf()`
- `TypedArray.prototype.join()`
- `TypedArray.prototype.lastIndexOf()`
- `TypedArray.prototype.map()`
- `TypedArray.prototype.reduce()`
- `TypedArray.prototype.reduceRight()`
- `TypedArray.prototype.reverse()`
- `TypedArray.prototype.some()`
- `TypedArray.prototype.set()`
- `TypedArray.prototype.slice()`
- `TypedArray.prototype.sort()`
- `TypedArray.prototype.subarray()`
- `TypedArray.prototype.toString()`

 Note

`TypedArray.prototype.every()`、`TypedArray.prototype.fill()`、`TypedArray.prototype`、`TypedArray.prototype.some()`は JavaScript ランタイム 2.0 で新しく追加されました。

ArrayBuffer

では、次の ES 6 メソッド `ArrayBuffer` がサポートされています。

- `isView()`

では、次の ES 6 プロトタイプメソッド `ArrayBuffer` がサポートされています。

- `ArrayBuffer.prototype.slice()`

promise

Promise について以下の ES 6 メソッドがサポートされています。

- `Promise.all()`

- `Promise.allSettled()`
- `Promise.any()`
- `Promise.reject()`
- `Promise.resolve()`
- `Promise.race()`

 Note

`Promise.all()`、`Promise.allSettled()`、`Promise.any()`、`Promise.race()`は JavaScript ランタイム 2.0 で新しく追加されました。

Promise について以下の ES 6 プロトタイプメソッドがサポートされています。

- `Promise.prototype.catch()`
- `Promise.prototype.finally()`
- `Promise.prototype.then()`

DataView

以下の ES 6 プロトタイプメソッドがサポートされています。

- `DataView.prototype.getFloat32()`
- `DataView.prototype.getFloat64()`
- `DataView.prototype.getInt16()`
- `DataView.prototype.getInt32()`
- `DataView.prototype.getInt8()`
- `DataView.prototype.getUint16()`
- `DataView.prototype.getUint32()`
- `DataView.prototype.getUint8()`
- `DataView.prototype.setFloat32()`
- `DataView.prototype.setFloat64()`
- `DataView.prototype.setInt16()`
- `DataView.prototype.setInt32()`
- `DataView.prototype.setInt8()`
- `DataView.prototype.setUint16()`

- `DataView.prototype.setUint32()`
- `DataView.prototype.setUint8()`

 Note

すべての DataView ES 6 プロトタイプメソッドは、ランタイム 2.0 で JavaScript 新しく追加されました。

記号

以下の ES 6 メソッドがサポートされています。

- `Symbol.for()`
- `Symbol.keyfor()`

 Note

すべての Symbol ES 6 メソッドは JavaScript、ランタイム 2.0 で新しく追加されました。

テキストデコーダー

以下のプロトタイプメソッドがサポートされています。

- `TextDecoder.prototype.decode()`

以下のプロトタイプアクセサープロパティがサポートされています。

- `TextDecoder.prototype.encoding`
- `TextDecoder.prototype.fatal`
- `TextDecoder.prototype.ignoreBOM`

テキストエンコーダー

以下のプロトタイプメソッドがサポートされています。

- `TextEncoder.prototype.encode()`
- `TextEncoder.prototype.encodeInto()`

エラーのタイプ

以下のエラーオブジェクトがサポートされています。

- Error
- EvalError
- InternalError
- RangeError
- ReferenceError
- SyntaxError
- TypeError
- URIError

Globals

globalThis オブジェクトはサポートされています。

以下の ES 5.1 グローバル関数がサポートされています。

- decodeURI()
- decodeURIComponent()
- encodeURI()
- encodeURIComponent()
- isFinite()
- isNaN()
- parseFloat()
- parseInt()

以下の ES 6 グローバル関数がサポートされています。

- atob()
- btoa()

Note

atob() と btoa() は JavaScript ランタイム 2.0 で新しく追加されました。

以下のグローバル定数がサポートされています。

- NaN
- Infinity
- undefined
- arguments

ビルトインモジュール

以下のビルトインモジュールがサポートされています。

モジュール

- [バッファ](#)
- [クエリ文字列](#)
- [Crypto](#)

バッファ

このモジュールは、以下のメソッドを提供します。

- `Buffer.alloc(size[, fill[, encoding]])`

`Buffer` を割り当てます。

- `size`: バッファサイズ。整数を入力します。
- `fill`: オプション。文字列、`Buffer`、`Uint8Array` または整数を入力します。デフォルトは `0` です。
- `encoding`: オプション。`fill` が文字列である場合は、`utf8`、`hex`、`base64`、`base64url` のいずれかを入力します。デフォルトは `utf8` です。
- `Buffer.allocUnsafe(size)`

初期化されていない `Buffer` を割り当てます。

- `size`: 整数を入力します。
- `Buffer.byteLength(value[, encoding])`

値の長さをバイト単位で返します。

- `value`: 文字列、`Buffer TypedArray`、`DataView`、または `Arraybuffer`。

- `encoding`: オプション。 `value` が文字列である場合は、 `utf8`、 `hex`、 `base64`、 `base64url` のいずれかを入力します。デフォルトは `utf8` です。
- `Buffer.compare(buffer1, buffer2)`

2つの `Buffer` を比較すると、配列をソートしやすくなります。両者が同じ場合は `0`、`buffer1` が先に来る場合は `-1`、`buffer2` が先に来る場合は `1` を返します。

- `buffer1`: `Buffer` を入力します。
- `buffer2`: 別の `Buffer` 値を入力します。
- `Buffer.concat(list[, totalLength])`

複数の `Buffer` を連結します。ない場合は `0` を返します。 `totalLength` までの値を返します。

- `list`: `Buffer` のリストを入力します。これは `totalLength` に切り捨てられることに注意してください。
- `totalLength`: オプション。符号なし整数を入力します。空欄の場合はリスト内の `Buffer` インスタンス総数を使用します。
- `Buffer.from(array)`

配列から `Buffer` を作成します。

- `array`: `0` から `255` までのバイト配列を入力します。
- `Buffer.from(arrayBuffer, byteOffset[, length])`

オフセット `byteOffset` から始めて長さが `length` のビューを `arrayBuffer` から作成します。

- `arrayBuffer`: `Buffer` 配列を入力します。
- `byteOffset`: 整数を入力します。
- `length`: オプション。整数を入力します。
- `Buffer.from(buffer)`

`Buffer` のコピーを作成します。

- `buffer`: `Buffer` を入力します。
- `Buffer.from(object[, offsetOrEncoding[, length]])`

オブジェクトから `Buffer` を作成します。 `valueOf()` がオブジェクトと等しくない場合は `Buffer.from(object.valueOf(), offsetOrEncoding, length)` を返します。

- `object`: オブジェクトを入力します。
- `offsetOrEncoding`: オプション。整数またはエンコーディング文字列を入力します。

- `length`: オプション。整数を入力します。
- `Buffer.from(string[, encoding])`

文字列から `Buffer` を作成します。

- `string`: 文字列を入力します。
- `encoding`: オプション。utf8、hex、base64、base64url のいずれかを入力します。デフォルトは utf8 です。
- `Buffer.isBuffer(object)`

`object` がバッファかどうかをチェックします。true または false を返します。

- `object`: オブジェクトを入力します。
- `Buffer.isEncoding(encoding)`

`encoding` がサポートされているかをチェックします。true または false を返します。

- `encoding`: オプション。utf8、hex、base64、base64url のいずれかを入力します。デフォルトは utf8 です。

このモジュールは、以下のバッファプロトタイプメソッドを提供します。

- `Buffer.prototype.compare(target[, targetStart[, targetEnd[, sourceStart[, sourceEnd]]]])`

ターゲットと `Buffer` を比較します。両者が同じ場合は 0、`buffer` が先に来る場合は 1、`target` が先に来る場合は -1 を返します。

- `target`: `Buffer` を入力します。
- `targetStart`: オプション。整数を入力します。デフォルトは 0 です。
- `targetEnd`: オプション。整数を入力します。デフォルトは `target` の長さです。
- `sourceStart`: オプション。整数を入力します。デフォルトは 0 です。
- `sourceEnd`: オプション。整数を入力します。デフォルトは `Buffer` の長さです。
- `Buffer.prototype.copy(target[, targetStart[, sourceStart[, sourceEnd]]])`

バッファを `target` にコピーします。

- `target`: `Buffer` または `Uint8Array` を入力します。
- `targetStart`: オプション。整数を入力します。デフォルトは 0 です。
- `sourceStart`: オプション。整数を入力します。デフォルトは 0 です。

- `sourceEnd`: オプション。整数を入力します。デフォルトは `Buffer` の長さです。
- `Buffer.prototype.equals(otherBuffer)`

`Buffer` と `otherBuffer` を比較します。true または false を返します。

- `otherBuffer`: 文字列を入力します。
 - `Buffer.prototype.fill(value[, offset[, end][, encoding])`
- `value` に `Buffer` を入力します。
- `value`: 文字列、`Buffer`、または整数を入力します。
 - `offset`: オプション。整数を入力します。
 - `end`: オプション。整数を入力します。
 - `encoding`: オプション。utf8、hex、base64、base64url のいずれかを入力します。デフォルトは utf8 です。
- `Buffer.prototype.includes(value[, byteOffset][, encoding])`

`value` で `Buffer` を検索します。true または false を返します。

- `value`: 文字列、`Buffer`、`Uint8Array`、または整数を入力します。
 - `byteOffset`: オプション。整数を入力します。
 - `encoding`: オプション。utf8、hex、base64、base64url のいずれかを入力します。デフォルトは utf8 です。
- `Buffer.prototype.indexOf(value[, byteOffset][, encoding])`

`Buffer` で最初の `value` を検索します。見つかった場合は `index` を返し、見つからなかった場合は -1 を返します。

- `value`: 文字列、`Buffer`、`Unit8Array`、または 0 から 255 までの整数を入力します。
 - `byteOffset`: オプション。整数を入力します。
 - `encoding`: オプション。value が文字列の場合、utf8、hex、base64、base64url のいずれかを入力します。デフォルトは utf8 です。
- `Buffer.prototype.lastIndexOf(value[, byteOffset][, encoding])`

`Buffer` で最後の `value` を検索します。見つかった場合は `index` を返し、見つからなかった場合は -1 を返します。

- `value`: 文字列、`Buffer`、`Unit8Array`、または 0 から 255 までの整数を入力します。

- `encoding`: オプション。value が文字列の場合、`utf8`、`hex`、`base64`、`base64url` のいずれかを入力します。デフォルトは `utf8` です。
- `Buffer.prototype.readInt8(offset)`

Buffer から `offset` で `Int8` を読み込みます。

- `offset`: 整数を入力します。
- `Buffer.prototype.readIntBE(offset, byteLength)`

Buffer から `offset` で `Int` をビッグエンディアンとして読み取ります。

- `offset`: 整数を入力します。
- `byteLength`: オプション。1 から 6 までの整数を入力します。
- `Buffer.prototype.readInt16BE(offset)`

Buffer から `offset` で `Int16` をビッグエンディアンとして読み取ります。

- `offset`: 整数を入力します。
- `Buffer.prototype.readInt32BE(offset)`

Buffer から `offset` で `Int32` をビッグエンディアンとして読み取ります。

- `offset`: 整数を入力します。
- `Buffer.prototype.readIntLE(offset, byteLength)`

Buffer から `offset` で `Int` をリトルエンディアンとして読み取ります。

- `offset`: 整数を入力します。
- `byteLength`: 1 から 6 までの整数を入力します。
- `Buffer.prototype.readInt16LE(offset)`

Buffer から `offset` で `Int16` をリトルエンディアンとして読み取ります。

- `offset`: 整数を入力します。
- `Buffer.prototype.readInt32LE(offset)`

Buffer から `offset` で `Int32` をリトルエンディアンとして読み取ります。

- `offset`: 整数を入力します。
- `Buffer.prototype.readUInt8(offset)`

Buffer から `offset` で `UInt8` を読み込みます。

- `offset`: 整数を入力します。

- `Buffer.prototype.readUIntBE(offset, byteLength)`

`Buffer` から `offset` で `UInt` をビッグエンディアンとして読み取ります。

- `offset`: 整数を入力します。
- `byteLength`: 1 から 6 までの整数を入力します。

- `Buffer.prototype.readUInt16BE(offset)`

`Buffer` から `offset` で `UInt16` をビッグエンディアンとして読み取ります。

- `offset`: 整数を入力します。
- `Buffer.prototype.readUInt32BE(offset)`

`Buffer` から `offset` で `UInt32` をビッグエンディアンとして読み取ります。

- `offset`: 整数を入力します。

- `Buffer.prototype.readUIntLE(offset, byteLength)`

`Buffer` から `offset` で `UInt` をリトルエンディアンとして読み取ります。

- `offset`: 整数を入力します。
- `byteLength`: 1 から 6 までの整数を入力します。

- `Buffer.prototype.readUInt16LE(offset)`

`Buffer` から `offset` で `UInt16` をリトルエンディアンとして読み取ります。

- `offset`: 整数を入力します。

- `Buffer.prototype.readUInt32LE(offset)`

`Buffer` から `offset` で `UInt32` をリトルエンディアンとして読み取ります。

- `offset`: 整数を入力します。

- `Buffer.prototype.readDoubleBE([offset])`

`Buffer` から `offset` で 64 ビットダブルをビッグエンディアンとして読み込みます。

- `offset`: オプション。整数を入力します。

- `Buffer.prototype.readDoubleLE([offset])`

`Buffer` から `offset` で 64 リトルダブルをビッグエンディアンとして読み込みます。

- `offset`: オプション。整数を入力します。

- `Buffer.prototype.readFloatBE([offset])`

Buffer から `offset` で 32 ビットフロートをビッグエンディアンとして読み込みます。

- `offset`: オプション。整数を入力します。

- `Buffer.prototype.readFloatLE([offset])`

Buffer から `offset` で 32 ビットフロートをリトルエンディアンとして読み込みます。

- `offset`: オプション。整数を入力します。

- `Buffer.prototype.subarray([start[, end]])`

オフセットし、新しい `start` および `end` で切り取った Buffer のコピーを返します。

- `start`: オプション。整数を入力します。デフォルトは0です。
- `end`: オプション。整数を入力します。デフォルトはバッファ長です。

- `Buffer.prototype.swap16()`

Buffer 配列のバイト順を入れ替え、16 ビットの数値の配列として扱います。Buffer の長さは 2 で割り切れる必要があります。そうしないと、エラーになります。

- `Buffer.prototype.swap32()`

Buffer 配列のバイト順を入れ替え、32 ビットの数値の配列として扱います。Buffer の長さは 4 で割り切れる必要があります。そうしないと、エラーになります。

- `Buffer.prototype.swap64()`

Buffer 配列のバイト順を入れ替え、64 ビットの数値の配列として扱います。Buffer の長さは 8 で割り切れる必要があります。そうしないと、エラーになります。

- `Buffer.prototype.toJSON()`

JSON として Buffer を返します。

- `Buffer.prototype.toString([encoding[, start[, end]])`

`start` から `end` まで Buffer をエンコードされた文字列に変換します。

- `encoding`: オプション。utf8、hex、base64、base64url のいずれかを入力します。デフォルトは utf8 です。
- `start`: オプション。整数を入力します。デフォルトは0です。
- `end`: オプション。整数を入力します。デフォルトはバッファの長さです。

- `Buffer.prototype.write(string[, offset[, length]][, encoding])`

スペースがある場合はエンコードされた string を Buffer に書き込み、十分なスペースがない場合は切り捨てられた string になります。

- string: 文字列を入力します。
- offset: オプション。整数を入力します。デフォルトは0です。
- length: オプション。整数を入力します。デフォルトは文字列の長さです。
- encoding: オプション。オプションで、utf8、hex、base64、または base64url のいずれかを入力します。デフォルトは utf8 です。

- `Buffer.prototype.writeInt8(value, offset, byteLength)`

offset で byteLength の Int8 value を Buffer に書き込みます。

- value: 整数を入力します。
- offset: 整数を入力します
- byteLength: 1 から 6 までの整数を入力します。

- `Buffer.prototype.writeIntBE(value, offset, byteLength)`

ビッグエンディアンを使用して offset の value を Buffer に書き込みます。

- value: 整数を入力します。
- offset: 整数を入力します
- byteLength: 1 から 6 までの整数を入力します。

- `Buffer.prototype.writeInt16BE(value, offset, byteLength)`

ビッグエンディアンを使用して offset の value を Buffer に書き込みます。

- value: 整数を入力します。
- offset: 整数を入力します
- byteLength: 1 から 6 までの整数を入力します。

- `Buffer.prototype.writeInt32BE(value, offset, byteLength)`

ビッグエンディアンを使用して offset の value を Buffer に書き込みます。

- value: 整数を入力します。
- offset: 整数を入力します
- byteLength: 1 から 6 までの整数を入力します。

- `Buffer.prototype.writeIntLE(offset, byteLength)`

リトルエンディアンを使用して `offset` の `value` を `Buffer` に書き込みます。

- `offset`: 整数を入力します。
- `byteLength`: 1 から 6 までの整数を入力します。
- `Buffer.prototype.writeInt16LE(offset, byteLength)`

リトルエンディアンを使用して `offset` の `value` を `Buffer` に書き込みます。

- `offset`: 整数を入力します。
- `byteLength`: 1 から 6 までの整数を入力します。
- `Buffer.prototype.writeInt32LE(offset, byteLength)`

リトルエンディアンを使用して `offset` の `value` を `Buffer` に書き込みます。

- `offset`: 整数を入力します。
- `byteLength`: 1 から 6 までの整数を入力します。
- `Buffer.prototype.writeUInt8(value, offset, byteLength)`

`offset` で `byteLength` の `UInt8 value` を `Buffer` に書き込みます。

- `value`: 整数を入力します。
- `offset`: 整数を入力します
- `byteLength`: 1 から 6 までの整数を入力します。
- `Buffer.prototype.writeUIntBE(value, offset, byteLength)`

ビッグエンディアンを使用して `offset` の `value` を `Buffer` に書き込みます。

- `value`: 整数を入力します。
- `offset`: 整数を入力します
- `byteLength`: 1 から 6 までの整数を入力します。
- `Buffer.prototype.writeUInt16BE(value, offset, byteLength)`

ビッグエンディアンを使用して `offset` の `value` を `Buffer` に書き込みます。

- `value`: 整数を入力します。
- `offset`: 整数を入力します
- `byteLength`: 1 から 6 までの整数を入力します。
- `Buffer.prototype.writeUInt32BE(value, offset, byteLength)`

ビッグエンディアンを使用して `offset` の `value` を `Buffer` に書き込みます。

- `value`: 整数を入力します。
- `offset`: 整数を入力します
- `byteLength`: 1 から 6 までの整数を入力します。
- `Buffer.prototype.writeUIntLE(value, offset, byteLength)`

リトルエンディアンを使用して `offset` の `value` を `Buffer` に書き込みます。

- `value`: 整数を入力します。
- `offset`: 整数を入力します
- `byteLength`: 1 から 6 までの整数を入力します。
- `Buffer.prototype.writeUInt16LE(value, offset, byteLength)`

リトルエンディアンを使用して `offset` の `value` を `Buffer` に書き込みます。

- `value`: 整数を入力します。
- `offset`: 整数を入力します
- `byteLength`: 1 から 6 までの整数を入力します。
- `Buffer.prototype.writeUInt32LE(value, offset, byteLength)`

リトルエンディアンを使用して `offset` の `value` を `Buffer` に書き込みます。

- `value`: 整数を入力します。
- `offset`: 整数を入力します
- `byteLength`: 1 から 6 までの整数を入力します。
- `Buffer.prototype.writeDoubleBE(value, [offset])`

ビッグエンディアンを使用して `offset` の `value` を `Buffer` に書き込みます。

- `value`: 整数を入力します。
- `offset`: オプション。整数を入力します。デフォルトは0です。
- `Buffer.prototype.writeDoubleLE(value, [offset])`

リトルエンディアンを使用して `offset` の `value` を `Buffer` に書き込みます。

- `value`: 整数を入力します。
- `offset`: オプション。整数を入力します。デフォルトは0です。
- `Buffer.prototype.writeFloatBE(value, [offset])`

ビッグエンディアンを使用して offset の value を Buffer に書き込みます。

- value: 整数を入力します。
- offset: オプション。整数を入力します。デフォルトは0です。
- `Buffer.prototype.writeFloatLE(value, [offset])`

リトルエンディアンを使用して offset の value を Buffer に書き込みます。

- value: 整数を入力します。
- offset: オプション。整数を入力します。デフォルトは0です。

以下のインスタンスメソッドがサポートされています。

- `buffer[index]`

Buffer index でオクテット (バイト) を取得および設定します。

- 0 から 255 までの数値を取得します。または、0 から 255 までの数値を設定します。

以下のインスタンスプロパティがサポートされています。

- `buffer`

バッファの `ArrayBuffer` オブジェクトを取得します。

- `byteOffset`

バッファの `Arraybuffer` オブジェクトの `byteOffset` を取得します。

- `length`

バッファのバイト数を取得します。

Note

すべての Buffer モジュールメソッドは JavaScript、ランタイム 2.0 で新しく追加されました。

クエリ文字列

Note

[CloudFront Functions イベントオブジェクト](#)は、URL クエリ文字列を自動的に解析します。つまり、ほとんどの場合、このモジュールを使用する必要はありません。

クエリ文字列モジュール (`querystring`) は、URL クエリ文字列を解析および書式設定するためのメソッドを提供します。`require('querystring')` を使用してモジュールをロードできます。このモジュールは、以下のメソッドを提供します。

`querystring.escape(string)`

URL は `string` をエンコードし、エスケープしたクエリ文字列を返します。このメソッドは `querystring.stringify()` で使用するため、直接使用しないでください。

`querystring.parse(string[, separator[, equal[, options]])`

クエリ文字列 (`string`) を解析し、オブジェクトを返します。

`separator` パラメータは、クエリ文字列のキーと値のペアを区切る substring です。デフォルトでは、`&` です。

`equal` パラメータは、クエリ文字列のキーと値を区切る substring です。デフォルトでは、`=` です。

`options` パラメータは、以下のキーを持つオブジェクトです。

`decodeURIComponent function`

クエリ文字列のパーセントエンコーディングされた文字を decode する関数です。デフォルトでは、`querystring.unescape()` です。

`maxKeys number`

解析するキーの最大数。デフォルトでは、`1000` です。キーカウントの制限を解除するには、`0` の値を使用します。

デフォルトでは、クエリ文字列のパーセントエンコーディングされた文字は、UTF-8 エンコーディングを使用していると見なされます。無効な UTF-8 シーケンスは、`U+FFFD` 置換文字に置き換えられます。

たとえば、次のクエリ文字列の場合:

```
'name=value&abc=xyz&abc=123'
```

`querystring.parse()` の戻り値は次のとおりです。

```
{
  name: 'value',
  abc: ['xyz', '123']
}
```

`querystring.decode()` は `querystring.parse()` のエイリアスです。

`querystring.stringify(object[, separator[, equal[, options]])`

`object` をシリアル化し、クエリ文字列を返します。

`separator` パラメータは、クエリ文字列のキーと値のペアを区切る substring です。デフォルトでは、& です。

`equal` パラメータは、クエリ文字列のキーと値を区切る substring です。デフォルトでは、= です。

`options` パラメータは、以下のキーを持つオブジェクトです。

`encodeURIComponent function`

URL-unsafe 文字をクエリ文字列のパーセントエンコーディングに変換するために使用される関数です。デフォルトでは、`querystring.escape()` です。

デフォルトでは、クエリ文字列でパーセントエンコーディングが必要な文字は UTF-8 としてエンコードされます。別のエンコーディングを使用するには、`encodeURIComponent` オプションを指定します。

以下のコードでの例:

```
querystring.stringify({ name: 'value', abc: ['xyz', '123'], anotherName: '' });
```

戻り値:

```
'name=value&abc=xyz&abc=123&anotherName='
```

`querystring.encode()` は `querystring.stringify()` のエイリアスです。

`querystring.unescape(string)`

指定された `string` 内の URL パーセントエンコーディングされた文字をデコードし、エスケープしていないクエリ文字列を返します。このメソッドは `querystring.parse()` で使用するため、直接使用しないでください。

Crypto

暗号モジュール (`crypto`) は、標準のハッシュおよびハッシュベースのメッセージ認証コード (HMAC) ヘルパーを提供します。 `require('crypto')` を使用してモジュールをロードできます。

ハッシュメソッド

`crypto.createHash(algorithm)`

ハッシュオブジェクトを作成して返します。このハッシュオブジェクトは、指定されたアルゴリズム (`md5`、`sha1`、`sha256` のいずれか) を使用してハッシュダイジェストの生成に使用できません。

`hash.update(data)`

指定された `data` を使用してハッシュコンテンツを更新します。

`hash.digest([encoding])`

`hash.update()` を使用して渡されたすべてのデータのダイジェストを計算します。エンコードは `hex`、`base64`、`base64url` のいずれかを使用します。

HMACメソッド

`crypto.createHmac(algorithm, secret key)`

指定された `algorithm` と `secret key` を使用する HMAC オブジェクトを作成して返します。アルゴリズムは `md5`、`sha1`、`sha256` のいずれかを使用します。

`hmac.update(data)`

指定された `data` を使用して HMAC コンテンツを更新します。

`hmac.digest([encoding])`

`hmac.update()` を使用して渡されたすべてのデータのダイジェストを計算します。エンコードは `hex`、`base64`、`base64url` のいずれかを使用します。

制限された機能

以下の JavaScript 言語機能は、セキュリティ上の問題によりサポートされていないか、制限されています。

動的コード評価

動的コード評価はサポートされていません。eval()、Function 両方のコンストラクタが試行された場合、エラーをスローします。たとえば、`const sum = new Function('a', 'b', 'return a + b')` はエラーをスローします。

タイマー

setTimeout()、setImmediate()、clearTimeout() 関数はサポートされていません。関数実行中に defer または yield する規定はありません。関数は同期的に実行しないと完了できません。

日付とタイムスタンプ

セキュリティ上の理由から、高解像度タイマーにはアクセスできません。現在の時刻を照会するすべての Date メソッドは、単一の関数実行の存続期間中は常に同じ値を返します。返されるタイムスタンプは、関数の実行を開始した時刻です。したがって、関数内で経過時間を測定することはできません。

ファイルシステムへのアクセス

ファイルシステムにはアクセスできません。

ネットワークアクセス

ネットワークコールはサポートされていません。たとえば、XHR、HTTP (S)、ソケットはサポートされていません。

キー値ストアのヘルパーメソッド

このセクションは、[CloudFront キー値ストア](#)を使用して、作成する関数にキー値を含める場合に適用されます。CloudFront 関数には、キー値ストアから値を読み取る 3 つのヘルパーメソッドを提供するモジュールがあります。

このモジュールを関数コードで使用するには、関数に[キー値ストアを関数に関連付けて](#)から、関数コードの最初の行に次のステートメントを含めます。

```
import cf from 'cloudfront';
const kvsId = "key value store ID";
```

```
const kvsHandle = cf.kvs(kvsId);
```

ID は「a1b2c3d4-5678-90ab-cdef-EXAMPLE1」のようになります。

get () メソッド

このメソッドは、指定したキー名のキー値を取得します。

リクエスト

```
get("key", options);
```

- key: 値をフェッチする必要があるキーの名前
- options: 1つのオプション、format があります。これにより、関数はデータを正しく解析します。使用できる値:
 - string: (デフォルト) UTF8 エンコード
 - json
 - bytes: 未加工のバイナリデータバッファ

リクエストの例

```
const value = await kvsHandle.get("myFunctionKey", { format: "string"});
```

レスポンス

レスポンスは 文字列です。

exists() メソッド

このメソッドは、キーがキー値ストアに存在するかどうかを示すブール値 (true または false) を返します。

リクエスト

```
exists("key");
```

リクエストの例

```
const exist = await kvsHandle.exists("myFunctionkey");
```

meta() メソッド

このメソッドは、キー値ストアに関するメタデータを返します。

リクエスト

```
meta();
```

リクエストの例

```
const meta = await kvsHandle.meta();
```

レスポンス

レスポンスは、以下のプロパティを持つオブジェクトに解決する promise です。

- `creationDateTime`: キー値ストアが作成された ISO 8601 形式の日付と時刻。
- `lastUpdatedDateTime`: キー値ストアがソースから最後に同期された ISO 8601 形式の日付と時刻。値にはエッジへの伝達時間は含まれていません。
- `keyCount`: ソースからの最後の同期後の KVS 内のキーの合計数。

レスポンスの例

```
{keyCount:3,creationDateTime:2023-11-30T23:07:55.765Z,lastUpdatedDateTime:2023-12-15T03:57:52.4
```

CloudFront Functions のコード例

関数の関数コードの記述を開始するには、次のサンプル CloudFront 関数を使用します。これらの例はすべて、の [amazon-cloudfront-functions リポジトリ GitHub](#)にあります。

例

- [レスポンスに Cache-Control ヘッダーを追加する](#)
- [Cross-Origin Resource Sharing \(CORS\) ヘッダーをレスポンスに追加](#)
- [Cross-Origin Resource Sharing \(CORS\) ヘッダーをリクエストに追加](#)
- [レスポンスにセキュリティヘッダーを追加する](#)
- [リクエストに True-Client-IP ヘッダーを追加する](#)
- [ビューワーを新しい URL にリダイレクトさせる](#)

- [index.html を追加してファイル名を含まない URL をリクエストする](#)
- [リクエストの単純なトークンを検証する](#)
- [async および await を使用する](#)
- [クエリ文字列パラメータの正規化](#)
- [関数でキーと値のペアを使用する](#)

レスポンスに **Cache-Control** ヘッダーを追加する

この例では、Cache-Control HTTP ヘッダーをレスポンスに追加します。ヘッダーは max-age ディレクティブを使用して、最大 2 年 (63,072,000 秒) の応答をキャッシュするようにウェブブラウザに指示します。詳細については、MDN Web Docs の Web サイトの「[Cache-Control](#)」を参照してください。

これはビューワー応答関数です。

[でこの例を参照してください GitHub。](#)

JavaScript runtime 2.0

```
async function handler(event) {
  const response = event.response;
  const headers = response.headers;

  // Set the cache-control header
  headers['cache-control'] = {value: 'public, max-age=63072000'};

  // Return response to viewers
  return response;
}
```

JavaScript runtime 1.0

```
function handler(event) {
  var response = event.response;
  var headers = response.headers;

  // Set the cache-control header
  headers['cache-control'] = {value: 'public, max-age=63072000'};

  // Return response to viewers
  return response;
}
```

```
}  
}
```

Cross-Origin Resource Sharing (CORS) ヘッダーをレスポンスに追加

この例では、レスポンスにこのヘッダーが含まれていない場合に `Access-Control-Allow-Origin` HTTP ヘッダーをレスポンスに追加します。このヘッダーは、[Cross-Origin Resource Sharing \(CORS\)](#) の一部です。ヘッダーの値 (*) は、任意のオリジンからのコードがこのリソースにアクセスできるように Web ブラウザに指示します。詳細については、MDN Web Docs Web サイトの「[Access-Control-Allow-Origin](#)」を参照してください。

これはビューワー応答関数です。

[でこの例を参照してください GitHub。](#)

JavaScript runtime 2.0

```
async function handler(event) {  
  const request = event.request;  
  const response = event.response;  
  
  // If Access-Control-Allow-Origin CORS header is missing, add it.  
  // Since JavaScript doesn't allow for hyphens in variable names, we use the  
  dict["key"] notation.  
  if (!response.headers['access-control-allow-origin'] &&  
    request.headers['origin']) {  
    response.headers['access-control-allow-origin'] = {value:  
    request.headers['origin'].value};  
    console.log("Access-Control-Allow-Origin was missing, adding it now.");  
  }  
  
  return response;  
}
```

JavaScript runtime 1.0

```
function handler(event) {  
  var response = event.response;  
  var headers = response.headers;  
  
  // If Access-Control-Allow-Origin CORS header is missing, add it.  
  // Since JavaScript doesn't allow for hyphens in variable names, we use the  
  dict["key"] notation.  
  if (!headers['access-control-allow-origin'] &&  
    request.headers['origin']) {  
    headers['access-control-allow-origin'] = {value:  
    request.headers['origin'].value};  
    console.log("Access-Control-Allow-Origin was missing, adding it now.");  
  }  
  
  return response;  
}
```

```
if (!headers['access-control-allow-origin']) {
  headers['access-control-allow-origin'] = {value: "*"};
  console.log("Access-Control-Allow-Origin was missing, adding it now.");
}

return response;
}
```

Cross-Origin Resource Sharing (CORS) ヘッダーをリクエストに追加

この例では、リクエストにこのヘッダーが含まれていない場合に Origin HTTP ヘッダーをリクエストに追加します。このヘッダーは、[Cross-Origin Resource Sharing \(CORS\)](#) の一部です。この例では、ヘッダーの値をリクエストの Host ヘッダーの値に設定します。詳細については、MDN Web Docs の Web サイトの「[Origin](#)」を参照してください。

これはビューワーリクエスト機能です。

[でこの例を参照してください GitHub。](#)

JavaScript runtime 2.0

```
async function handler(event) {
  const request = event.request;
  const headers = request.headers;
  const host = request.headers.host.value;

  // If origin header is missing, set it equal to the host header.
  if (!headers.origin)
    headers.origin = {value: `https://${host}`};

  return request;
}
```

JavaScript runtime 1.0

```
function handler(event) {
  var request = event.request;
  var headers = request.headers;
  var host = request.headers.host.value;

  // If origin header is missing, set it equal to the host header.
  if (!headers.origin)
```

```
headers.origin = {value: `https://${host}`};

return request;
}
```

レスポンスにセキュリティヘッダーを追加する

この例では、いくつかの一般的なセキュリティ関連の HTTP ヘッダーをレスポンスに追加します。詳細については、MDN Web Docs Web サイトの以下のページを参照してください。

- [Strict-Transport-Security](#)
- [Content-Security-Policy](#)
- [X-Content-Security-Policy](#)
- [X-Frame-Options](#)
- [X-XSS-Protection](#)

これはビューワー応答関数です。

[でこの例を参照してください GitHub。](#)

JavaScript runtime 2.0

```
async function handler(event) {
  const response = event.response;
  const headers = response.headers;

  // Set HTTP security headers
  // Since JavaScript doesn't allow for hyphens in variable names, we use the
  dict["key"] notation
  headers['strict-transport-security'] = { value: 'max-age=63072000;
includeSubdomains; preload'};
  headers['content-security-policy'] = { value: "default-src 'none'; img-src
'self'; script-src 'self'; style-src 'self'; object-src 'none'; frame-ancestors
'none'"};
  headers['x-content-type-options'] = { value: 'nosniff'};
  headers['x-frame-options'] = {value: 'DENY'};
  headers['x-xss-protection'] = {value: '1; mode=block'};
  headers['referrer-policy'] = {value: 'same-origin'};

  // Return the response to viewers
```

```
    return response;
}
```

JavaScript runtime 1.0

```
function handler(event) {
    var response = event.response;
    var headers = response.headers;

    // Set HTTP security headers
    // Since JavaScript doesn't allow for hyphens in variable names, we use the
    dict["key"] notation
    headers['strict-transport-security'] = { value: 'max-age=63072000;
includeSubdomains; preload'};
    headers['content-security-policy'] = { value: "default-src 'none'; img-src
'self'; script-src 'self'; style-src 'self'; object-src 'none'"};
    headers['x-content-type-options'] = { value: 'nosniff'};
    headers['x-frame-options'] = {value: 'DENY'};
    headers['x-xss-protection'] = {value: '1; mode=block'};

    // Return the response to viewers
    return response;
}
```

リクエストに **True-Client-IP** ヘッダーを追加する

この例では、ビューワの IP アドレスをヘッダーの値として True-Client-IP HTTP ヘッダーをリクエストに追加します。ガリクエストをオリジン CloudFront に送信する場合、オリジンはリクエストを送信した CloudFront ホストの IP アドレスを決定できますが、元のリクエストを に送信したビューワ (クライアント) の IP アドレスは決定できません CloudFront。この関数は、True-Client-IP ヘッダーを追加してオリジンがビューワの IP アドレスを確認できるようにします。

Important

がこのヘッダーをオリジンリクエスト CloudFront に含めるようにするには、[オリジンリクエストポリシー](#) の許可されたヘッダーリストに追加する必要があります。

これはビューワリクエスト機能です。

[でこの例を参照してください GitHub。](#)

JavaScript runtime 2.0

```
async function handler(event) {
  var request = event.request;
  var clientIP = event.viewer.ip;

  //Add the true-client-ip header to the incoming request
  request.headers['true-client-ip'] = {value: clientIP};

  return request;
}
```

JavaScript runtime 1.0

```
function handler(event) {
  var request = event.request;
  var clientIP = event.viewer.ip;

  //Add the true-client-ip header to the incoming request
  request.headers['true-client-ip'] = {value: clientIP};

  return request;
}
```

ビューワーを新しい URL にリダイレクトさせる

この例では、リクエストが特定の国内から送信されたときに、ビューワーを国固有の URL にリダイレクトするレスポンスを生成します。この関数は、CloudFront-Viewer-Country ヘッダーの値に依存して、ビューワーの国を判断します。

Important

この関数を機能させるには、[キャッシュポリシー](#)またはオリジンリクエストポリシーの許可された CloudFront CloudFront-Viewer-Country ヘッダーに追加して、受信リクエストにヘッダーを追加するようにを設定する必要があります。 [???](#)

この例では、ビューワーリクエストがドイツから送信された場合、ビューワーをドイツ固有の URL にリダイレクトします。ビューワーリクエストがドイツから来ていない場合、この関数は元の変更されていないリクエストを返します。

これはビューワーリクエスト機能です。

[でこの例を参照してください GitHub。](#)

JavaScript runtime 2.0

```
async function handler(event) {
  const request = event.request;
  const headers = request.headers;
  const host = request.headers.host.value;
  const country = Symbol.for('DE'); // Choose a country code
  const newurl = `https://${host}/de/index.html`; // Change the redirect URL to
  your choice

  if (headers['cloudfront-viewer-country']) {
    const countryCode = Symbol.for(headers['cloudfront-viewer-country'].value);
    if (countryCode === country) {
      const response = {
        statusCode: 302,
        statusDescription: 'Found',
        headers:
          { "location": { "value": newurl } }
      }

      return response;
    }
  }
  return request;
}
```

JavaScript runtime 1.0

```
function handler(event) {
  var request = event.request;
  var headers = request.headers;
  var host = request.headers.host.value;
  var country = 'DE' // Choose a country code
  var newurl = `https://${host}/de/index.html` // Change the redirect URL to your
  choice

  if (headers['cloudfront-viewer-country']) {
    var countryCode = headers['cloudfront-viewer-country'].value;
    if (countryCode === country) {
```

```
        var response = {
            statusCode: 302,
            statusDescription: 'Found',
            headers:
                { "location": { "value": newurl } }
        }

        return response;
    }
}
return request;
}
```

書き換えとリダイレクトの詳細については、AWSワークショップスタジオの「[エッジ関数を使用した書き換えとリダイレクトの処理](#)」を参照してください。

index.html を追加してファイル名を含まない URL をリクエストする

この例では、URL にファイル名や拡張子を含まないリクエストに `index.html` を追加します。この機能は、単一ページアプリケーションや Amazon S3 バケットでホストされている静的に生成されたウェブサイト便利です。

これはビューワーリクエスト機能です。

[でこの例を参照してください GitHub。](#)

JavaScript runtime 2.0

```
async function handler(event) {
    const request = event.request;
    const uri = request.uri;

    // Check whether the URI is missing a file name.
    if (uri.endsWith('/')) {
        request.uri += 'index.html';
    }
    // Check whether the URI is missing a file extension.
    else if (!uri.includes('.')) {
        request.uri += '/index.html';
    }

    return request;
}
```

```
}
```

JavaScript runtime 1.0

```
function handler(event) {
  var request = event.request;
  var uri = request.uri;

  // Check whether the URI is missing a file name.
  if (uri.endsWith('/')) {
    request.uri += 'index.html';
  }
  // Check whether the URI is missing a file extension.
  else if (!uri.includes('.')) {
    request.uri += '/index.html';
  }

  return request;
}
```

リクエストの単純なトークンを検証する

この例では、リクエストのクエリ文字列の [JSON ウェブトークン \(JWT\)](#) を検証します。トークンが有効な場合、関数は元の変更されていないリクエストを に返します CloudFront。トークンが有効でない場合、関数はエラーレスポンスを生成します。この関数は crypto モジュールを使用します。詳細については、「[ビルトインモジュール](#)」を参照してください。

この関数は、リクエストの jwt という名前のクエリ文字列パラメータに JWT 値が含まれていることを前提としています。

Warning

この関数を使用するには、関数コードにシークレットキーを配置する必要があります。

これはビューワーリクエスト機能です。

[でこの例を参照してください GitHub。](#)

JavaScript runtime 2.0

```
const crypto = require('crypto');

//Response when JWT is not valid.
const response401 = {
  statusCode: 401,
  statusDescription: 'Unauthorized'
};

function jwt_decode(token, key, noVerify, algorithm) {
  // check token
  if (!token) {
    throw new Error('No token supplied');
  }
  // check segments
  const segments = token.split('.');
  if (segments.length !== 3) {
    throw new Error('Not enough or too many segments');
  }

  // All segment should be base64
  const headerSeg = segments[0];
  const payloadSeg = segments[1];
  const signatureSeg = segments[2];

  // base64 decode and parse JSON
  const header = JSON.parse(_base64urlDecode(headerSeg));
  const payload = JSON.parse(_base64urlDecode(payloadSeg));

  if (!noVerify) {
    const signingMethod = 'sha256';
    const signingType = 'hmac';

    // Verify signature. `sign` will return base64 string.
    const signingInput = [headerSeg, payloadSeg].join('.');

    if (!_verify(signingInput, key, signingMethod, signingType, signatureSeg)) {
      throw new Error('Signature verification failed');
    }
  }

  // Support for nbf and exp claims.
  // According to the RFC, they should be in seconds.
  if (payload.nbf && Date.now() < payload.nbf*1000) {
```

```
        throw new Error('Token not yet active');
    }

    if (payload.exp && Date.now() > payload.exp*1000) {
        throw new Error('Token expired');
    }
}

return payload;
}

//Function to ensure a constant time comparison to prevent
//timing side channels.
function _constantTimeEquals(a, b) {
    if (a.length != b.length) {
        return false;
    }

    var xor = 0;
    for (var i = 0; i < a.length; i++) {
        xor |= (a.charCodeAt(i) ^ b.charCodeAt(i));
    }

    return 0 === xor;
}

function _verify(input, key, method, type, signature) {
    if(type === "hmac") {
        return _constantTimeEquals(signature, _sign(input, key, method));
    }
    else {
        throw new Error('Algorithm type not recognized');
    }
}

function _sign(input, key, method) {
    return crypto.createHmac(method, key).update(input).digest('base64url');
}

function _base64urlDecode(str) {
    return Buffer.from(str, 'base64url')
}

function handler(event) {
```

```
const request = event.request;
//Secret key used to verify JWT token.
//Update with your own key.
var key = "LzdWGpAToQ1DqYuzHxE6Y0qi7G3X2yvNBot9mCXfx5k";

// If no JWT token, then generate HTTP redirect 401 response.
if(!request.querystring.jwt) {
    console.log("Error: No JWT in the querystring");
    return response401;
}

const jwtToken = request.querystring.jwt.value;

try{
    jwt_decode(jwtToken, key);
}
catch(e) {
    console.log(e);
    return response401;
}

//Remove the JWT from the query string if valid and return.
delete request.querystring.jwt;
console.log("Valid JWT token");
return request;
}
```

JavaScript runtime 1.0

```
var crypto = require('crypto');

//Response when JWT is not valid.
var response401 = {
    statusCode: 401,
    statusDescription: 'Unauthorized'
};

function jwt_decode(token, key, noVerify, algorithm) {
    // check token
    if (!token) {
        throw new Error('No token supplied');
    }
    // check segments
```

```
var segments = token.split('.');
if (segments.length !== 3) {
    throw new Error('Not enough or too many segments');
}

// All segment should be base64
var headerSeg = segments[0];
var payloadSeg = segments[1];
var signatureSeg = segments[2];

// base64 decode and parse JSON
var header = JSON.parse(_base64urlDecode(headerSeg));
var payload = JSON.parse(_base64urlDecode(payloadSeg));

if (!noVerify) {
    var signingMethod = 'sha256';
    var signingType = 'hmac';

    // Verify signature. `sign` will return base64 string.
    var signingInput = [headerSeg, payloadSeg].join('.');

    if (!_verify(signingInput, key, signingMethod, signingType, signatureSeg)) {
        throw new Error('Signature verification failed');
    }

    // Support for nbf and exp claims.
    // According to the RFC, they should be in seconds.
    if (payload.nbf && Date.now() < payload.nbf*1000) {
        throw new Error('Token not yet active');
    }

    if (payload.exp && Date.now() > payload.exp*1000) {
        throw new Error('Token expired');
    }
}

return payload;
}

function _verify(input, key, method, type, signature) {
    if(type === "hmac") {
        return (signature === _sign(input, key, method));
    }
    else {
```

```
        throw new Error('Algorithm type not recognized');
    }
}

function _sign(input, key, method) {
    return crypto.createHmac(method, key).update(input).digest('base64url');
}

function _base64urlDecode(str) {
    return String.bytesFrom(str, 'base64url')
}

function handler(event) {
    var request = event.request;

    //Secret key used to verify JWT token.
    //Update with your own key.
    var key = "LzdWgpAToQ1DqYuzHxE6Y0qi7G3X2yvNBot9mCXfx5k";

    // If no JWT token, then generate HTTP redirect 401 response.
    if(!request.querystring.jwt) {
        console.log("Error: No JWT in the querystring");
        return response401;
    }

    var jwtToken = request.querystring.jwt.value;

    try{
        jwt_decode(jwtToken, key);
    }
    catch(e) {
        console.log(e);
        return response401;
    }

    //Remove the JWT from the query string if valid and return.
    delete request.querystring.jwt;
    console.log("Valid JWT token");
    return request;
}
```

async および await を使用する

Amazon CloudFront JavaScript ランタイム関数 2.0 には、Promise オブジェクトを処理するための `async` および `await` 構文が用意されています。Promise は遅延した結果を表し、`async` とマークされた関数のキーワード `await` を使用してアクセスできます。さまざまな新しい WebCrypto 関数は Promises を使用します。

Promise オブジェクトの詳細については、「[Promise](#)」を参照してください。

```
async function answer() {
    return 42;
}

// Note: async, await can be used only inside an async function.

async function handler(event) {
    // var answer_value = answer(); // returns Promise, not a 42 value
    let answer_value = await answer(); // resolves Promise, 42
    console.log("Answer"+answer_value);
    event.request.headers['answer'] = { value : ""+answer_value };
    return event.request;
}
```

次の JavaScript コード例は、チェーン `then` メソッドで promise を表示する方法を示しています。 `catch` を使用してエラーを表示できます。

```
async function answer() {
    return 42;
}

async function squared_answer() {
    // before, in NJS 0.4.3 we have to write as following
    // return answer().then(function(value) { return value * value; })

    // in NJS 0.7.11 we can simplify
    return answer().then(value => value * value)
}

// note async, await can be used only inside async function
async function handler(event) {
    // var answer_value = answer(); // returns Promise, not a 42 value
    let answer_value = await squared_answer(); // resolves Promise, 42
    console.log("Answer"+answer_value);
    event.request.headers['answer'] = { value : ""+answer_value };
}
```

```
    return event.request;
}
```

Note

async および await は、JavaScript ランタイム 2.0 を使用する場合にのみ使用できます。

クエリ文字列パラメータの正規化

クエリ文字列パラメータを正規化して、キャッシュヒット率を高めることができます。

次の例は、ガリクエストをオリジン CloudFront に転送する前にクエリ文字列をアルファベット順に配置することで、キャッシュヒット率を向上させる方法を示しています。

```
function handler(event) {
    var qs=[];
    for (var key in event.request.querystring) {
        if (event.request.querystring[key].multiValue) {
            event.request.querystring[key].multiValue.forEach((mv) => {qs.push(key +
            "=" + mv.value)});
        } else {
            qs.push(key + "=" + event.request.querystring[key].value);
        }
    }
};

event.request.querystring = qs.sort().join('&');

return event.request;
}
```

関数でキーと値のペアを使用する

[キー値ストア](#)のキーと値のペアを関数で使用できます。

次の例は、HTTP リクエスト内の URL のコンテンツを使用してキー値ストア内のカスタムパスを検索する関数を示しています。CloudFront このカスタムパスを使用してリクエストを行います。この関数は、ウェブサイトに含まれる複数のパスを管理するのに役立ちます。

```
import cf from 'cloudfront';
```

```
// Declare the ID of the key value store that you have associated with this function
// The import fails at runtime if the specified key value store is not associated with
  the function

const kvsId = "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111";

const kvsHandle = cf.kvs(kvsId);

async function handler(event) {
  const request = event.request;
  // Use the first segment of the pathname as key
  // For example http(s)://domain/<key>/something/else
  const pathSegments = request.uri.split('/')
  const key = pathSegments[1]
  try {
    // Replace the first path of the pathname with the value of the key
    // For example http(s)://domain/<value>/something/else
    pathSegments[1] = await kvsHandle.get(key);
    const newUri = pathSegments.join('/');
    console.log(`${request.uri} -> ${newUri}`)
    request.uri = newUri;
  } catch (err) {
    // No change to the pathname if the key is not found
    console.log(`${request.uri} | ${err}`);
  }
  return request;
}
```

Functions での CloudFront 関数の管理

CloudFront Functions を使用すると、で軽量な関数を記述 JavaScript して、レイテンシーの影響を受けやすい CDN のカスタマイズを大規模に実行できます。[関数コードを記述](#)した後、以下のトピックは、CloudFront 関数の作成、テスト、更新、公開、デイス CloudFront トリビューションへの関連付けに役立ちます。

関数をデプロイするための完全な手順は次のとおりです。

- 関数を作成します。この関数は、最初にサンプル関数コードを使用して作成されます。このコードはライブステージでも有効です。この関数は開発段階にあります。
- テスト、更新、関連付け。関数が開発段階にある場合は、関数をテストし、更新することができます (キー値ストアを関数に関連付けることも含む)。

- 発行する。CloudFront デイストリビューションで関数を使用する準備ができたなら、関数を公開して、DEVELOPMENTステージから にコピーしますLIVE。
- デイストリビューションに関連付けます。関数が LIVE ステージのとき、デイストリビューションのキャッシュ動作と関数を関連付けることができます。

トピック

- [関数の作成](#)
- [関数のテスト](#)
- [関数の更新](#)
- [関数の公開](#)
- [関数とデイストリビューションの関連付け](#)

関数の作成

関数は 2 段階で作成します。まず、 の外部で Java スクリプトとして関数コードを作成します CloudFront。次に、 CloudFront を使用して関数を作成し、コードを含めます。コードは関数内にあります (リファレンスとしてではありません)。

新しい関数が DEVELOPMENT ステージに追加されます。関数を LIVE ステージにコピーするには、 [その関数をパブリッシュする](#) 必要があります。

Console

関数を作成するには (コンソール)

1. にサインインAWS Management Consoleし、 CloudFront コンソールの 関数 ページを開きます <https://console.aws.amazon.com/cloudfront/v4/home#/functions>。
2. [関数の作成] を選択します。
3. AWS アカウント内で一意の関数名を入力し、Java Script バージョンを選択して、[続行] を選択します。これで、この関数は存在するようになりました。新しい関数の詳細ページが表示されます。

Note

関数で [キーと値のペア](#) を使用する場合は、Java Script 2.0 を選択する必要があります。

4. [関数コード] セクションで、[ビルド] タブを選択し、関数コードを入力します。[ビルド] タブに含まれるサンプルコードは、関数コードの基本的な構文を示しています。コードは次のように入力できます。
 - 使用を開始するには、デフォルトの関数を使用してください。
 - のコード [例からコピーしたコードに置き換えます GitHub](#)。
 - これを独自のコードに置き換えます。

関数コードの記述の詳細については、以下を参照してください。

- [関数コードの記述](#)
 - [the section called “イベントの構造”](#)
5. [変更を保存] を必要な回数だけ選択して、関数コードを保存します。
 6. 関数コードがキーと値のペアを使用する場合は、キーと値のストアを関連付ける必要があります。

キーと値のストアは、関数の初回作成時に関連付けることができます。また、後で [関数を更新して](#) 関連付けることもできます。

キーと値のストアを今すぐ関連付けるには、次の手順に従います。

- 「関連付け KeyValueCollection」セクションに移動し、「既存の を関連付け KeyValueCollectionる」を選択します。
- 関数のキーと値のペアを含むキーと値のストアを選択し、 の関連付け KeyValueCollection を選択します。

CloudFront は、ストアを 関数に直ちに関連付けます。関数を保存する必要はありません。

CLI

CLI を使用する場合、通常は最初に関数コードをファイルに作成し、次に AWS CLI を使用して関数を作成します。

1. 関数コードをファイルに作成し、コンピュータが接続できるディレクトリに保存します。関数コードの記述の詳細については、以下を参照してください。
 - [関数コードの記述](#)

- [the section called “イベントの構造”](#)

2. 次の例に示すようにコマンドを実行します。この例では、fileb:// 表記を使用してファイルに渡します。コマンドを読みやすくするために改行も含まれています。

```
aws cloudfront create-function \  
  --name MaxAge \  
  --function-config Comment="Max Age 2 years",\  
    Runtime="cloudfront-js-2.0", \  
    KeyValueStoreAssociations= \  
      {Quantity=1, \  
        Items=[{KeyValueStoreARN='arn:aws:cloudfront::1:key-value-store/  
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111'}]}"} \  
  --function-code fileb://function-max-age-v1.js
```

注記：

- Runtime: Java Script のバージョン。関数で [キーと値のペア](#) を使用する場合は、バージョン 2.0 を指定する必要があります。
- KeyValueStoreAssociations: 関数がキーと値のペアを使用する場合、関数の初回作成時にキー値ストアを関連付けることができます。または、update-function を使用して後で関連付けることもできます。各関数に関連付けることができるキー値ストアは 1 つだけのため、Quantity は常に 1 です。

コマンドが成功した場合は、以下のような出力が表示されます。

```
ETag: ETVABCEXAMPLE  
FunctionSummary:  
  FunctionConfig:  
    Comment: Max Age 2 years  
    Runtime: cloudfront-js-2.0  
    KeyValueStoreAssociations= \  
      {Quantity=1, \  
        Items=[{KeyValueStoreARN='arn:aws:cloudfront::1:key-value-store/  
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111'}]}"} \  
  FunctionMetadata:  
    CreatedTime: '2021-04-18T20:38:56.915000+00:00'  
    FunctionARN: arn:aws:cloudfront::111122223333:function/MaxAge  
    LastModifiedTime: '2023-11-19T20:38:56.915000+00:00'  
    Stage: DEVELOPMENT
```

```
Name: MaxAge
Status: UNPUBLISHED
Location: https://cloudfront.amazonaws.com/2020-05-31/function/
arn:aws:cloudfront::111122223333:function/MaxAge
```

ほとんどの情報はリクエストから繰り返されます。その他の情報は によって追加されます CloudFront。次の点に注意してください。

- ETag: この値は、キー値ストアを変更するたびに変わります。この値と関数名を使用して、今後この関数を参照します。必ず現在の ETag を使用してください。
- FunctionARN
- Stage
- Status

関数のテスト

[CloudFront 関数](#)をライブステージ (本番環境) にデプロイする前に、関数をテストして、意図したとおりに動作することを確認できます。関数をテストするには、CloudFront デイストリビューションが本番環境で受信できる HTTP リクエストまたはレスポンスを表すイベントオブジェクトを指定します。CloudFront Functions は以下を実行します。

1. 指定されたイベントオブジェクトを入力として使用して、関数を実行します。
2. 関数の結果 (変更されたイベントオブジェクト) を、関数ログまたはエラーメッセージ、および関数のコンピューティング使用率とともに返します。コンピューティング使用率の詳細については、「[the section called “コンピューティング使用率について”](#)」を参照してください。

トピック

- [イベントオブジェクトをセットアップする](#)
- [関数をテストする](#)
- [コンピューティング使用率について](#)

イベントオブジェクトをセットアップする

関数をテストする前に、テストに使用するイベントオブジェクトをセットアップする必要があります。これには複数のオプションがあります。

オプション 1: イベントオブジェクトを保存せずにセットアップする

コンソールのビジュアルエディタ CloudFront でイベントオブジェクトを設定し、保存することはできません。

このイベントオブジェクトを使用して、保存されていない場合でも、コンソールから関数を CloudFront テストできます。

オプション 2: ビジュアルエディターでイベントオブジェクトを作成する

コンソールのビジュアルエディタ CloudFront でイベントオブジェクトを設定し、保存することはできません。関数ごとに 10 個のイベントオブジェクトを作成して、例えば、考えられるさまざまな入力をテストできます。

この方法でイベントオブジェクトを作成すると、イベントオブジェクトを使用して CloudFront コンソールで関数をテストできます。AWS API や SDK を使用して関数をテストする場合には使用できません。

オプション 3: テキストエディターを使用してイベントオブジェクトを作成する

テキストエディターを使用して、イベントオブジェクトを JSON 形式で作成できます。イベントオブジェクトの構造については、「[イベントの構造](#)」を参照してください。

このイベントオブジェクトを使用して、CLI で関数をテストできます。ただし、コンソールで CloudFront 関数をテストするために使用することはできません。

オプション 1 または 2 を使用して作成するには

1. CloudFront コンソールで関数ページを表示し、テストする関数を選択します。
2. 関数の詳細ページで、[テスト] タブを選択します。[関数をテスト] セクションには、[JSON を編集] と [関数のテスト] を含むボタンが表示されます。
3. [イベントタイプ] を入力する:
 - 関数が HTTP リクエストを変更したり、リクエストに基づいてレスポンスを生成する場合は、[ビューワーリクエスト] を選択します。既に表示されている [リクエスト] セクションがこのタイプに適用されます。
 - または [ビューアーレスポンス] を選択します。既に表示されている [リクエスト] セクションがこのタイプに適用されます。さらに、[レスポンス] セクションが表示されます。

4. イベントに含めるすべてのフィールドを入力します。作業中に [JSON を編集] を選択すると、未加工の JSON を表示できます。
5. 必要に応じてイベントを保存します。

JSON の編集 を選択して raw JSON をコピーし、 の外部にある独自のファイルに保存することもできます CloudFront。

オプション 3 を使用して作成するには

テキストエディタを使用してイベントオブジェクトを作成します。このファイルは、コンピューターが接続できるディレクトリに保存します。

必ず以下のガイドラインに従ってください。

- distributionDomainName、distributionId、requestId の各フィールドは省略します。
- ヘッダー、Cookie、クエリ文字列の名前は必ず小文字にしてください。

この方法でイベントオブジェクトを作成する方法の 1 つは、ビジュアルエディターを使用してサンプルを作成することです。サンプルが正しいフォーマットになっていることを確認できます。そして、未加工の JSON をコピーし、テキストエディターに貼り付け、ファイルを保存することができます。

イベントの構造の詳細については、「[イベントの構造](#)」を参照してください。

関数をテストする

CloudFront コンソールまたは を使用して関数をテストできますAWS CLI。

Console

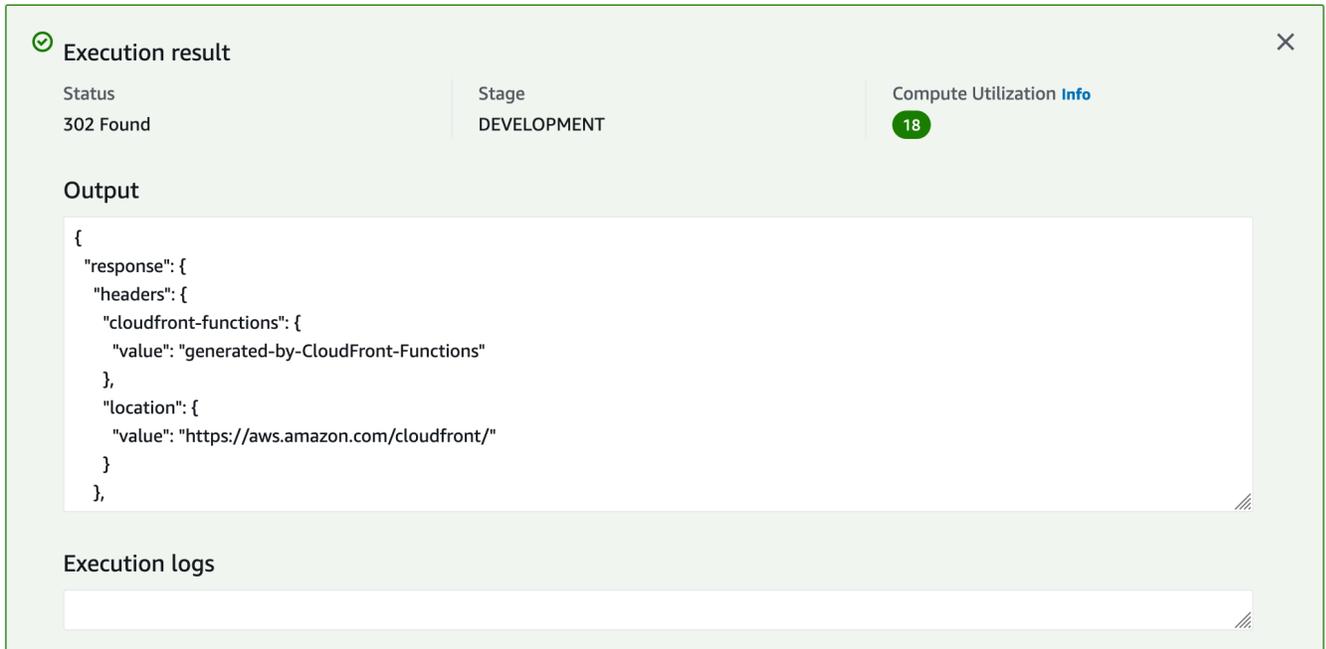
CloudFront コンソールでは、コンソールを使用して作成した関数をテストできます。

関数をテストするには

1. コンソールで CloudFront関数ページを表示し、テストする関数を選択します。
2. 関数ページで、[Test] タブを選択します。[関数をテスト] セクションには、[JSON を編集] と [関数のテスト] を含むボタンが表示されます。
3. 正しいイベントが表示されていることを確認します。

現在表示されているイベントから切り替える場合は、[テストイベントを選択] フィールドで別のイベントを選択します。

4. [関数をテスト] ボタンを選択します。コンソールには、関数ログを含む関数の出力が表示されます。また、コンピューティング使用率も表示されます。詳細については、「[the section called “コンピューティング使用率について”](#)」を参照してください。



CLI

aws cloudfront test-function コマンドを使用して関数をテストできます。

1. 次の例に示すようにコマンドを実行します。このファイルが含まれているのと同じディレクトリからコマンドを実行します。

この例では、fileb:// 表記を使用してイベントオブジェクトファイルに渡します。コマンドを読みやすくするために改行も含まれています。

```
aws cloudfront test-function \  
  --name MaxAge \  
  --if-match ETVABCEXAMPLE \  
  --event-object fileb://event-maxage-test01.json \  
  --stage DEVELOPMENT
```

注記：

- 関数は名前と ETag (if-match パラメータ内) で参照します。イベントオブジェクトはファイルシステム内のロケーションによって参照します。
- ステージは、DEVELOPMENT または LIVE の場合があります。

コマンドが成功した場合は、以下のような出力が表示されます。

```
TestResult:
  ComputeUtilization: '21'
  FunctionErrorMessage: ''
  FunctionExecutionLogs: []
  FunctionOutput: '{"response":{"headers":{"cloudfront-functions":
{"value":"generated-by-CloudFront-Functions"},"location":{"value":"https://
aws.amazon.com/cloudfront/"}},"statusDescription":"Found","cookies":
{},"statusCode":302}}'
  FunctionSummary:
    FunctionConfig:
      Comment: MaxAge function
      Runtime: cloudfront-js-2.0
      KeyValueStoreAssociations= \
      {Quantity=1, \
      Items=[{KeyValueStoreARN='arn:aws:cloudfront::1:key-value-store/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111'}]}\" \
    FunctionMetadata:
      CreatedTime: '2021-04-18T20:38:56.915000+00:00'
      FunctionARN: arn:aws:cloudfront::111122223333:function/MaxAge
      LastModifiedTime: '2023-17-20T10:38:57.057000+00:00'
      Stage: DEVELOPMENT
    Name: MaxAge
    Status: UNPUBLISHED
```

注記：

- FunctionExecutionLogs には、関数が console.log() ステートメントに書き込んだ (該当する場合) ログ行のリストが含まれます。
- ComputeUtilization。 [「the section called “コンピューティング使用率について”」](#) を参照してください。
- FunctionOutput には、関数が返したイベントオブジェクトが含まれます。

コンピューティング使用率について

コンピューティング使用率は、関数の実行にかかった時間 (最大許容時間に対するパーセンテージ) です。たとえば、値 35 は、関数が最大許容時間の 35% で完了したことを意味します。

関数が最大許容時間を継続的に超えると、は関数を CloudFront スロットリングします。次のリストは、コンピューティング使用率の値に基づいて関数がスロットリングされる可能性を説明しています。

コンピューティング使用率値:

- 1~50 — 関数は最大許容時間を十分に下回っており、スロットリングなしで実行する必要があります。
- 51~70 — 関数が最大許容時間に近づいています。関数コードの最適化を検討してください。
- 71~100 – 関数が最大許容時間に非常に近いが、それを超えています。ディストリビューションに関連付けると、この関数がスロットリング CloudFront される可能性があります。

関数の更新

関数はいつでも更新できます。変更は、DEVELOPMENT ステージ内の関数のバージョンに対してのみ行われます。関数を DEVELOPMENT ステージから LIVE にコピーするには、[その関数をパブリッシュする](#)必要があります。

関数のコードは、CloudFront コンソールまたは [で更新できます](#) AWS CLI。

Console

関数コードを更新するには (コンソール)

1. CloudFront コンソールの 関数 ページを で開き <https://console.aws.amazon.com/cloudfront/v4/home#/functions>、更新する関数を選択します。
2. 変更を加える:
 - [編集] ボタンを選択し、[詳細] セクションのフィールドを変更できます。
 - 関連付けられたキー値ストアは変更または削除できます。適切なボタンを選択します。キー値ストアの詳細については、「[the section called “の使用 CloudFront KeyValueCollection”](#)」を参照してください。

- 関数コードは変更することはできません。[ビルド] タブを選択し、変更を加え、[変更を保存] を選択してコードへの変更のみを保存します。

CLI

関数コード (CLI) を更新するには

次の例に示すようにコマンドを実行します。

この例では、fileb:// 表記を使用してファイルに渡します。コマンドを読みやすくするために改行も含まれています。

```
aws cloudfront update-function \  
  --name MaxAge \  
  --function-config Comment="Max Age 2 years",\  
    Runtime="cloudfront-js-2.0", \  
    KeyValueStoreAssociations= \  
      {Quantity=1, \  
        Items=[{KeyValueStoreARN='arn:aws:cloudfront::1:key-value-store/  
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111'}]} \  
  --function-code fileb://function-max-age-v1.js \  
  --if-match ETVABCEXAMPLE
```

注記：

- 関数は名前と ETag (if-match パラメータ内) の両方で識別します。必ず現在の ETag を使用してください。describe オペレーションを使用して取得できます。
- 変更したくない場合でも、function-code を含める必要があります。そうしないと、コードは削除されます。 **true?
- function-config には注意してください。設定に保持するすべてのものを渡す必要があります。具体的には、キー値ストアを次のように処理してください。
 - 既存のキー値ストアの関連付け (存在する場合) を維持したい場合は、既存のストアの名前を指定します。
 - 関連付けを変更する場合は、新しいキー値ストアの名前を指定します。
 - 関連付けを削除する場合は、KeyValueStoreAssociations パラメータを省略します。

コマンドが成功した場合は、以下のような出力が表示されます。

```
ETag: ETVXYZEXAMPLE
FunctionSummary:
  FunctionConfig:
    Comment: Max Age 2 years \
    Runtime: cloudfront-js-2.0 \
    KeyValueStoreAssociations= \
      {Quantity=1, \
      Items=[{KeyValueStoreARN='arn:aws:cloudfront::1:key-value-store/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111'}]} \
    FunctionMetadata: \
      CreatedTime: '2021-04-18T20:38:56.915000+00:00' \
      FunctionARN: arn:aws:cloudfront::111122223333:function/MaxAge \
      LastModifiedTime: '2023-12-19T23:41:15.389000+00:00' \
      Stage: DEVELOPMENT \
    Name: MaxAge \
    Status: UNPUBLISHED
```

ほとんどの情報はリクエストから繰り返されます。その他の情報は によって追加されます CloudFront。次の点に注意してください。

- ETag: この値は、キー値ストアを変更するたびに変更されます。
- FunctionARN
- Stage
- Status

関数の公開

関数を公開すると、ステージ DEVELOPMENT からステージ LIVE にコピーされます。

Important

関数を公開すると、ディストリビューションのデプロイが完了しだい、関数に関連付けられたすべてのキャッシュ動作は自動的に新しく公開されたコピーを使用するようになります。

関数にキャッシュ動作が関連付けられていない場合、関数を公開すると、関数をキャッシュ動作に関連付けることができます。キャッシュ動作は、LIVE ステージの関数にのみ関連付けることができます。

関数は、CloudFront コンソールまたは を使用して公開できますAWS CLI。

パブリッシュする前に、[関数をテストする](#)必要があります。

Console

関数を公開するには、CloudFront コンソールを使用できます。コンソールには、関数に関連付けられている CloudFront ディストリビューションも表示されます。

関数を公開するには (コンソール)

1. 関数を公開するには、CloudFront のコンソールで関数ページを開き<https://console.aws.amazon.com/cloudfront/v4/home#/functions>、公開する関数を選択します。
2. 関数ページで、[Publish] タブを選択します。次に、[Publish] ボタン (または、関数が 1 つまたは複数のキャッシュ動作に既にアタッチされている場合は、[Publish and update] ボタン) を選択します。
3. (オプション) 関数に関連付けられているディストリビューションを表示するには、関連 CloudFrontディストリビューションを選択してそのセクションを展開します。

正常に完了すると、ページの上部に「###が正常に公開されました」というバナーが表示されます。[Build] タブから [Live] を選択して、関数コードのライブバージョンを表示することもできます。

CLI

関数をパブリッシュするには、例にある `aws cloudfront publish-function` コマンドを実行します。この例では、例を読みやすくするために改行されています。

```
aws cloudfront publish-function \  
  --name MaxAge \  
  --if-match ETVXYZEXAMPLE
```

コマンドが成功した場合は、以下のような出力が表示されます。

```
FunctionSummary:  
  FunctionConfig:  
    Comment: Max Age 2 years  
    Runtime: cloudfront-js-2.0  
  FunctionMetadata:  
    CreatedTime: '2021-04-18T21:24:21.314000+00:00'
```

```
FunctionARN: arn:aws:cloudfront::111122223333:function/ExampleFunction
LastModifiedTime: '2023-12-19T23:41:15.389000+00:00'
Stage: LIVE
Name: MaxAge
Status: UNASSOCIATED
```

関数とディストリビューションの関連付け

CloudFront Functions の関数を CloudFront ディストリビューションで使用するには、関数をディストリビューション内の 1 つ以上のキャッシュ動作に関連付けます。関数を [複数のディストリビューション](#) の複数のキャッシュ動作に関連付けることができます。関数を関連付ける前に、[その関数をステージ LIVE に公開](#)する必要があります。

関数をキャッシュ動作に関連付ける場合は、イベントタイプを選択する必要があります。イベントタイプによって、CloudFront 関数が実行されるタイミングが決まります。次の 2 つのイベントタイプから選択できます。

イベントタイプの詳細については、「[CloudFront Lambda@Edge 関数をトリガーできる イベント](#)」を参照してください。CloudFront Functions では、オリジン向けイベントタイプ (オリジンリクエストおよびオリジンレスポンス) を使用できません。

- ビューワーリクエスト – 関数はビューワーからリクエスト CloudFront を受信したときに実行されます。
- ビューワーレスポンス – 関数は、 がビューワーにレスポンスを CloudFront 返す前に実行されます。

CloudFront コンソールまたは `aws` で、関数をディストリビューションに関連付けることができますAWS CLI。

Console

CloudFront コンソールを使用して、関数を既存の CloudFront ディストリビューションの既存のキャッシュ動作に関連付けることができます。ディストリビューションの作成については、「[the section called “ディストリビューションの作成”](#)」を参照してください。

関数を既存のキャッシュ動作に関連付けるには (コンソール)

1. コンソールの 関数 ページを CloudFront で開き <https://console.aws.amazon.com/cloudfront/v4/home#/functions>、関連付ける関数を選択します。

2. 関数ページで、[Publish] タブを選択します。[関連付けられているディストリビューション] セクションが表示されます。
3. [関連付けを追加] ボタンを選択します。表示されるダイアログで、ディストリビューション、イベントタイプ、キャッシュ動作を選択します。

イベントタイプでは、この関数を実行するタイミングを選択します。

- がリクエスト CloudFront を受信するたびに関数を実行するには、ビューワーリクエストを選択します。
- がレスポンスを CloudFront 返すたびに関数を実行するには、ビューワーレスポンスを選択します。

設定を保存するには、[関連付けを追加] を選択します。

CloudFront は、ディストリビューションを関数に関連付けます。関連付けられたディストリビューションのデプロイが完了するまで数分待ちます。関数の詳細ページで [ディストリビューションを表示] を選択すると、進行状況を確認できます。

CLI

関数は以下のいずれかに関連付けることができます。

- 既存のキャッシュ動作。
- 既存のディストリビューションの新しいキャッシュ動作。
- 新しいディストリビューションの新しいキャッシュ動作。

次の手順は、関数を既存のキャッシュ動作に関連付ける方法を示しています。

関数を既存のキャッシュ動作に関連付けるには (AWS CLI)

1. 関数に関連付けるキャッシュ動作を有するディストリビューションのディストリビューション設定を保存するには、次のコマンドを使用します。このコマンドは、ディストリビューション設定を `dist-config.yaml` という名前のファイルに保存します。このコマンドを使用するには、次の操作を行います。
 - ***DistributionID*** をディストリビューションの ID に置き換えます。
 - コマンドを 1 行で実行します。この例では、例を読みやすくするために改行されています。

```
aws cloudfront get-distribution-config \  
  --id DistributionID \  
  --output yaml > dist-config.yaml
```

コマンドが正常に完了した場合、AWS CLI は出力を返しません。

2. 先ほど作成した `dist-config.yaml` という名前のファイルを開きます。ファイルを編集して以下の変更を加えます。
 - a. ETag フィールドの名前を `IfMatch` に変更しますが、フィールドの値は変更しないでください。
 - b. キャッシュ動作で、`FunctionAssociations` という名前のオブジェクトを見つけます。このオブジェクトを更新して、関数の関連付けを追加します。関数を関連付ける YAML 構文は、次の例のようになります。
 - 次の例は、ビューワーリクエストイベントタイプ (トリガー) を示しています。ビューアレスポンスイベントタイプを使用するには、`viewer-request` を `viewer-response` に置き換えます。
 - `arn:aws:cloudfront::111122223333:function/ExampleFunction` は、このキャッシュ動作に関連付ける関数の Amazon リソースネーム (ARN) に置き換えます。関数 ARN を取得するには、`aws cloudfront list-functions` コマンドを使用します。

```
FunctionAssociations:  
  Items:  
    - EventType: viewer-request  
      FunctionARN: arn:aws:cloudfront::111122223333:function/ExampleFunction  
  Quantity: 1
```

変更が完了したら、ファイルを保存します。

3. 関数の関連付けを追加してディストリビューションを更新するには、次のコマンドを使用します。このコマンドを使用するには、次の操作を行います。
 - `DistributionID` をディストリビューションの ID に置き換えます。
 - コマンドを 1 行で実行します。この例では、例を読みやすくするために改行されています。

```
aws cloudfront update-distribution \  
  --id DistributionID \  
  --cli-input-yaml file://dist-config.yaml
```

コマンドが正常に完了すると、関数の関連付けで更新されたばかりのディストリビューションを説明する次のような出力が表示されます。次の出力例は、読みやすくするために切り詰めています。

```
Distribution:  
  ARN: arn:aws:cloudfront::111122223333:distribution/EBEDLT3BGRBBW  
  ... truncated ...  
DistributionConfig:  
  ... truncated ...  
DefaultCacheBehavior:  
  ... truncated ...  
FunctionAssociations:  
  Items:  
  - EventType: viewer-request  
    FunctionARN: arn:aws:cloudfront::111122223333:function/ExampleFunction  
    Quantity: 1  
  ... truncated ...  
DomainName: d111111abcdef8.cloudfront.net  
Id: EDFDVBD6EXAMPLE  
LastModifiedTime: '2021-04-19T22:39:09.158000+00:00'  
Status: InProgress  
ETag: E2VJGGQEG1JT8S
```

ディストリビューションを関連付けることの効果

ディストリビューションが再デプロイされる間に、ディストリビューションの Status は InProgress に変更されます。新しいディストリビューション設定がエッジロケーションに到達すると CloudFront すぐに、そのエッジロケーションは関連付けられた関数の使用を開始します。ディストリビューションが完全にデプロイされると、は Status に戻ります。これは Deployed、関連付けられた CloudFront 関数が世界中のすべての CloudFront エッジロケーションで稼働していることを示します。これには通常数分かかります。

Amazon CloudFront KeyValueCollection

CloudFront KeyValueCollection は、CloudFront 関数内からの読み取りアクセスを可能にする、安全でグローバルな低レイテンシーのキーバリューストアです。これにより、エッジロケーションで CloudFront 高度なカスタマイズ可能なロジックが可能になります。

では CloudFront KeyValueCollection、関数コードを更新し、関数に関連付けられたデータを互いに独立して更新します。このように分離することで関数コードが簡略化され、コードの変更をデプロイしなくてもデータを簡単に更新できます。

キーと値のペアを使用する一般的な手順は次のとおりです。

- キーバリューストアを作成し、そのストアにキーと値のペアのセットを設定します。
- キーバリューストアを CloudFront 関数に関連付けます。
- 関数コード内で、キーの名前を使用してキーに関連付けられた値を取得するか、キーの存在を評価します。関数コードでキーと値のペアを使用する方法の詳細とヘルパーメソッドについては、[「the section called “キーバリューストアのヘルパーメソッド”」](#)を参照してください。

ユースケース

以下は、キーと値のペアの一般的なユースケースです。

- URL の書き換えまたはリダイレクト。キーと値のペアには、書き換えられた URL またはリダイレクト URL を含めることができます。
- A/B テストと機能フラグ。ウェブサイトの特定のバージョンにトラフィックの割合を割り当てることで、テストを実行する関数を作成できます。
- アクセスの許可。アクセスコントロールを実装して、ユーザーが定義した基準とキーバリューストアに保存されているデータに基づいてリクエストを許可または拒否できます。

サポートされている値の形式

キーと値のペアの値は、以下のいずれかの形式で保存できます。

- 文字列
- バイトでエンコードされた文字列
- JSON

セキュリティ

CloudFront 関数とそのすべてのキー値ストアデータは、次のように安全に処理されます。

- CloudFront は、保管時および転送時 (キー値ストアの読み取りまたは書き込み時) に各キー値ストアを暗号化します。
- 関数が実行されると、は CloudFront エッジロケーションのメモリ内の各キーと値のペアを復 CloudFront 号します。

トピック

- [キー値ストアの使用](#)
- [キーと値のデータでの操作](#)

キー値ストアの使用

CloudFront Functions で使用するキーと値のペアを保持するには、キーと値のストアを作成する必要があります。

キー値ストアのキーと値のペアは、次の方法で操作できます。

- CloudFront コンソールの使用。
- 任意の CloudFront API または SDK を使用します。

キー値ストアを作成し、キーと値のペアを追加したら、CloudFront 関数コードでキー値を使用できます。JavaScript ランタイム 2.0 には、関数コードでキー値を操作するためのヘルパーメソッドがいくつか含まれています。詳細については、「[the section called “キー値ストアのヘルパーメソッド”](#)」を参照してください。

トピック

- [キー値ストアの作成](#)
- [キー値ストアの関数への関連付け](#)
- [キー値ストアの変更](#)
- [キー値ストアの削除](#)
- [キー値ストアへのリファレンスの取得](#)
- [キーと値のペアのファイルの作成](#)

キー値ストアの作成

空のキー値ストアを作成し、後でキーと値のペアを追加することができます。または、キー値ストアとそのキーと値のペアを同時に作成することもできます。

Note

Amazon S3 バケットからデータソースを指定する場合は、そのバケットに対する `s3:GetObject` および `アクセスs3:GetBucketLocation` 許可が必要です。これらのアクセス許可がない場合は、キー値ストアを正常に作成 CloudFront できません。

Console

キー値ストアを作成するには (コンソール)

1. キー値ストアの作成と同時にキーと値のペアを追加するかどうかを決定します。([キーと値のペアは後で追加することもできます](#))。このインポート機能は、CloudFront コンソールと CloudFront APIs および SDKs。ただし、キー値ストアを最初に作成した場合にのみサポートされます。

ファイルを使用する場合は、今すぐ [それを作成](#) します。

2. にサインイン AWS Management Console し、CloudFront コンソールの 関数 ページを開きます <https://console.aws.amazon.com/cloudfront/v4/home#/functions>。
3. [KeyValueStores] タブを選択します。[サーバーの作成] KeyValueStore ボタンを選択します。
4. キー値ストアの名前とオプションで説明を入力します。
5. [S3 URI] を入力します。
 - キーと値のペアのファイルを準備したら、ファイルを保存する Amazon S3 バケットへのパスを入力します。
 - キーと値のペアを手動で入力する場合は、このフィールドを空白のままにします。
6. [作成] ボタンを選択します。これで、キー値ストアが作成されました。

新しいキー値ストアの詳細ページが表示されます。ページの情報には、キー値ストアの ID と ARN が含まれます。

- ID は、AWS アカウント内で固有のランダムな文字列です。

- ARN の構文は次のとおりです。

AWS #####:key-value-store/##### ID

7. [キーと値のペア] セクションを見てください。ファイルをインポートした場合、このセクションにはペアがいくつ也表示されます。それ以外の場合は、空です。以下の操作を行うことができます。
 - Amazon S3 バケットからファイルをインポートしておらず、キーと値のペアを今すぐ追加したい場合は、このセクションを完了できます。
 - ファイルをインポートした場合は、値を手動で追加することもできます。
 - このセクションは空のままにしておき、後でキー値ストアを編集してペアを追加できます。

ここでペアを追加するには:

- [キーと値のペアを追加] ボタンを選択します。
- [ペアの追加] ボタンを選択して名前と値を入力します。
- [ペアの追加] ボタンを選択してさらにペアを追加します。

完了したら、[変更内容を保存] を選択してキー値ストア内のすべてのペアを保存します。表示される確認ダイアログで [完了] を選択します。

8. キー値ストアを関数に今すぐ関連付ける場合は、[関連付けられている関数] セクションに入力します。この関連付けは、このキー値ストアの詳細ページまたは関数の詳細ページから後で作成することもできます。

関連付けを今すぐ作成するには、[関数へ移動] ボタンを選択します。詳細については、[???](#)または[???](#)を参照してください。

Programmatically

1. キー値ストアの作成と同時にキーと値のペアを追加するかどうかを決定します。(キーと値のペアは[後で](#)追加することもできます)。このインポート機能は、CloudFront コンソールと CloudFront APIs および SDKs。ただし、キー値ストアを最初に作成した場合にのみサポートされます。

ファイルを使用する場合は、今すぐ[それを作成](#)します。

2. 任意の CloudFront API または SDK の作成オペレーションを使用します。例えば、REST API の場合は、[CloudFront. CreateKeyValueStore](#)このオペレーションには複数のパラメータが必要です。
 - 名前。
 - コメントを含む configuration パラメータ。
 - Amazon S3 バケットに保存されているファイルからキーと値のペアをインポートするための `import-source` パラメータ。ファイルからインポートできるのは、キー値ストアの初回作成時のみであることに注意してください。ファイルの形式の詳細については、「[the section called “キーと値のペアのファイルの作成”](#)」を参照してください。

オペレーションレスポンスには、以下の情報が含まれます。

- リクエストで渡された値 (割り当てた名前を含む)。
- 作成時間などのデータ。
- ETag (ETVABCEXAMPLE2 など)、キー値ストアの名前 (arn:aws:cloudfront::111122223333:key-value-store/MaxAge など) を含む ARN。

ETag、ARN、名前の組み合わせを使用して、キー値ストアをプログラムで操作します。

キー値ストアのステータス

キー値ストアを作成すると、データストアのステータス値は次のようになります。

値	説明
プロビジョニング	キー値ストアが作成され、指定したデータソース CloudFront を処理していません。
準備完了	キー値ストアが作成され、指定したデータソースが正常に CloudFront 処理されました。
インポートに失敗しました	CloudFront は、指定したデータソースを処理できませんでした。このステータスは、ファイル形式が有効でない場合、またはサイズ制限を超えている場合に表示されます。詳細については、「 キーと値のペアのファイルの作成 」を参照してください。

キー値ストアの関数への関連付け

キー値ストアを関数に関連付けるには、[その関数进行操作](#)します。そのストアにあるキーと値のペアをその関数で使用するには、この関連付けを行う必要があります。以下のルールが適用されます。

- 1つの関数には1つのキー値ストアを含めることができます。
- 1つのキー値ストアを複数の関数に関連付けることができます。

次の方法で関連付けを操作することができます。

- 関数とキー値ストアの間に関連付けを作成することができます。
 - CloudFront コンソールで、キー値ストアの詳細ページを表示し、関数に移動ボタンを選択します。該当するページ、つまり [関数] リスト (現在関連する関数がない場合) または関数の詳細ページ (現在関連付けがある場合) が表示されます。詳細については、「[the section called “キー値ストアの関数への関連付け”](#)」を参照してください。
 - プログラムで、任意の CloudFront API または SDK の関数更新オペレーションを使用します。

関連付けを作成したら (または関連付けを変更した場合)、関数を[テスト](#)し、関数を[再パブリッシュ](#)する必要があります。

- キーと値のペアを変更せずにキー値ストアを変更した場合、関連付けを更新する必要はありません (つまり、再パブリッシュする必要はありません)。ただし、この関数は[テスト](#)する必要があります。
- キー値ストアでキーと値のペアを変更する場合、関連付けを更新する必要はありません (つまり、再パブリッシュする必要はありません)。ただし、関数を[テスト](#)して、キーと値のペアの変更が反映されることを確認する必要があります。
- 特定のキー値ストアを使用するすべての関数を表示できます。CloudFront コンソールで、キー値ストアの詳細ページを確認します。

キー値ストアの変更

キーと値のペアを操作したり、キー値ストアと関数の関連付けを変更できます。

CloudFront コンソールの使用

1. にサインインAWS Management Consoleし、CloudFront コンソールの 関数 ページを開きま
す<https://console.aws.amazon.com/cloudfront/v4/home#/functions>。

2. [KeyValueStores] タブを選択します。変更するキー値ストアを選択します。[詳細] ページが表示されます。
 - キーと値のペアを操作するには、[キーと値のペア] セクションで、[編集] ボタンを選択します。キーと値のペアをさらに追加したり、任意のキーと値のペアを削除したり、既存のキーと値のペアの値を変更できます。完了したら、[変更を保存] を選択します。
 - このキー値ストアの関連付けを操作するには、[関数へ移動] ボタンを選択します。該当するページ、つまり [関数] リスト (現在関連する関数がない場合) または関数の詳細ページ (現在関連付けがある場合) が表示されます。詳細については、「[the section called “キー値ストアの関数への関連付け”](#)」を参照してください。

ストアのプログラムによる変更

キー値ストアは次の方法で操作できます。

キーと値のペアを変更する

キーと値のペアをさらに追加したり、1 つ以上のキーと値のペアを削除したり、既存のキーと値のペアの値を変更できます。詳細については、「[the section called “プログラムによる操作”](#)」を参照してください。

キー値ストアの関数関連付けを変更する

このキー値ストアの関連付けを操作するには、「[the section called “関数の更新”](#)」を参照してください。キー値ストアの ARN が必要になります。詳細については、「[the section called “キー値ストアへのリファレンスの取得”](#)」を参照してください。

キー値ストアの削除

CloudFront コンソールの使用

1. にサインインAWS Management Consoleし、 CloudFront コンソールの 関数 ページを開きます <https://console.aws.amazon.com/cloudfront/v4/home#/functions>。
2. キー値ストアが関数に関連付けられているかどうかを確認します。関連付けられている場合は、それを削除します。これらの両方のステップの詳細については、「[???](#)」を参照してください。
3. [KeyValueStores] タブを選択します。変更するキー値ストアを選択します。削除 ボタンを選択します。

プログラムによる削除

1. ETag とキー値ストアの名前を取得します。詳細については、「[the section called “キー値ストアへのリファレンスの取得”](#)」を参照してください。
2. キー値ストアが関数に関連付けられているかどうかを確認します。関連付けられている場合は、それを削除します。これらの両方のステップの詳細については、「[???](#)」を参照してください。
3. キー値ストアを削除するには、任意の CloudFront API または SDK の削除オペレーションを使用します。例えば、REST API の場合は、[CloudFront. DeleteKeyValueStore](#) を使用します。

キー値ストアへのリファレンスの取得

キー値ストアをプログラムで操作するには、ETag とキー値ストアの名前が必要です。このデータを取得するには、以下のステップを実行してください。

1. 任意の CloudFront API または SDK のリストオペレーションを使用します。例えば、REST API の場合は、[CloudFront. ListKeyValueStores](#) を使用します。このレスポンスには、キー値ストアのリストが含まれています。変更するキー値ストアの名前を探します。
2. 任意の CloudFront API または SDK の describe オペレーションを使用します。例えば、REST API の場合は、[CloudFront. DescribeKeyValueStore](#) を使用します。取得した名前を渡します。

describe オペレーションに関する重要な情報については、「[the section called “について CloudFront KeyValueStore”](#)」を参照してください。

レスポンスには、UUID、キー値ストアの ARN、およびキー値ストアの ETag が含まれます。

- UUID は 128 ビットです。例えば、次のようになります: a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
- ARN には、AWS アカウント番号、定数 key-value-store、UUID が含まれます。例:

```
arn:aws:cloudfront::111122223333:key-value-store/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

- ETag は ETVABCEXAMPLE2 のようになります。

キーと値のペアのファイルの作成

UTF-8 でエンコードされたファイルを作成するときは、次の形式を使用します。

```
{  
  "data": [  
    {  
      "key": "key",  
      "value": "value"  
    }  
  ]  
}
```

```
{
  "key": "key1",
  "value": "value"
},
{
  "key": "key2",
  "value": "value"
}
]
```

Note

データソースのファイルとそのキーと値のペアには、以下の制限があります。

- ファイルサイズ — 5 MB
- キーサイズ — 512 文字
- 値のサイズ — 1024 文字

キーと値のデータでの操作

既存のキー値ストアのキーと値のペアは、以下の方法で操作できます。

- CloudFront コンソールの使用。
- 任意の CloudFront KeyValueCollection API または SDK を使用します。

Note

これは、通常の API または SDK とは異なる CloudFront API または SDK です。

このセクションでは、既存のキー値ストアにキーと値のペアを追加する方法について説明します。キー値ストアを最初に作成するときにキーと値のペアを含める方法については、「[the section called “キー値ストアの作成”](#)」を参照してください。

トピック

- [CloudFront コンソールを使用したキーと値のペアの操作](#)
- [キーと値のペアによる操作](#)

CloudFront コンソールを使用したキーと値のペアの操作

1. にサインインAWS Management Consoleし、 のコンソールで 関数 ページを開きます<https://console.aws.amazon.com/cloudfront/v4/home#/functions>。CloudFront
2. [KeyValueStores] タブを選択します。変更するキー値ストアを選択します。[詳細] ページが表示されます。
3. [キーと値のペア] セクションで、[編集] ボタンを選択します。
4. キーと値のペアを追加したり、キーと値のペアを削除したり、既存のキーと値のペアの値を変更できます。
5. 完了したら、[変更を保存] を選択します。

キーと値のペアによる操作

トピック

- [キー値ストアへのリファレンスの取得](#)
- [キー値ストア内のキーと値のペアの変更](#)
- [について CloudFront KeyValueStore](#)
- [のコード例 CloudFront KeyValueStore](#)

キー値ストアへのリファレンスの取得

を使用して書き込みオペレーションを入力するときは CloudFront KeyValueStore、キー値ストアの ARN と ETag を渡す必要があります。このデータを取得するには、以下を実行します。

1. 任意の CloudFront API または SDKs。例えば、REST API の場合は [CloudFront.ListKeyValueStores](#) を使用します。このレスポンスには、キー値ストアのリストが含まれています。変更するキー値ストアの名前を探します。
2. 任意の CloudFront KeyValueStore API または SDK の describe オペレーションを使用します。例えば、REST API の場合は [CloudFrontKeyValueStore.DescribeKeyValueStore](#) を使用します。前のステップで取得した名前に渡します。

Note

CloudFront KeyValueCollection API からではなく、CloudFront API から オペレーションを使用します。詳細については、「[the section called “について CloudFront KeyValueCollection”](#)」を参照してください。

レスポンスには、キー値ストアの ARN および ETag が含まれます。

- ARN には、AWS アカウント番号、定数 key-value-store、UUID が含まれます。例:

```
arn:aws:cloudfront::111122223333:key-value-store/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
```

- ETag は ETVABCEXAMPLE2 のようになります。

キー値ストア内のキーと値のペアの変更

任意の CloudFront KeyValueCollection API または SDK の次のオペレーションを使用して、キーと値のペアを操作できます。これらのオペレーションはすべて、指定した 1 つのキー値ストアで機能します。

- `CloudFrontKeyValueCollection.DeleteKey`: キーを 1 つ削除します。「[DeleteKey](#)」を参照してください。
- `CloudFrontKeyValueCollection.GetKey`: キーを 1 つ取得します。「[GetKey](#)」を参照してください。
- `CloudFrontKeyValueCollection.ListKeys`: キーを一覧表示します。「[ListKeys](#)」を参照してください。
- `CloudFrontKeyValueCollection.PutKey`: 次の 2 つのアクションを実行できます。
 - 1 つのキー値ストアに新しいキーと値のペアを作成する: この場合は、新しいキー名と値を渡します。
 - 1 つの既存のキー値ペアに別の値を設定する: この場合は、既存のキー名と新しいキー値を渡します。

「[PutKey](#)」を参照してください。

- `CloudFrontKeyValueStore.UpdateKeys`: 1 回の all-or-nothing オペレーションで次の 1 つ以上のアクションを実行できます。
 - 1 つまたは複数のキーと値のペアを削除します。
 - 1 つまたは複数の新しいキーと値のペアを作成します。
 - 1 つまたは複数の既存のキーと値のペアに別の値を設定します。

「[UpdateKeys](#)」を参照してください。

について CloudFront KeyValueStore

既存のキー値ストアでキーと値のペアをプログラムで操作するには、CloudFront KeyValueStore サービスを使用します。

最初にキー値ストアを作成するとき、キー値ストアにいくつかのキーと値のペアを含めるには、CloudFront サービスを使用します。

describe オペレーション

CloudFront API と CloudFront KeyValueStore API の両方に、キー値ストアに関するデータを返す describe オペレーションがあります。

- CloudFront API は、ステータスやストア自体が最後に変更された日付などのデータを提供します。
- CloudFront KeyValueStore API は、ストレージリソースの内容、つまりストア内のキーと値のペア、コンテンツのサイズに関するデータを提供します。

この 2 つの API の describe オペレーションは、キー値ストアを識別する若干異なるデータを返します。

- CloudFront API の describe オペレーションは、ETag、UUID、およびキー値ストアの ARN を返します。
- CloudFront KeyValueStore API の describe オペレーションは、ETag とキー値ストアの ARN を返します。

Note

describe オペレーションはそれぞれ異なる ETag を返します。ETag は置き換え可能ではありません。

いずれかの API でオペレーションを実行する場合、適切な API から ETag を渡す必要があります。例えば、の削除オペレーションで CloudFront KeyValueStore、の describe オペレーションから取得した ETag を渡します CloudFront KeyValueStore。

のコード例 CloudFront KeyValueStore

Example : DescribeKeyValueStore API オペレーションの呼び出し

次のサンプルコードは、キー値ストアに対して DescribeKeyValueStore API オペレーションを呼び出す方法を示しています。

```
const {
  CloudFrontKeyValueStoreClient,
  DescribeKeyValueStoreCommand,
} = require("@aws-sdk/client-cloudfront-keyvaluestore");

require("@aws-sdk/signature-v4-crt");

(async () => {
  try {
    const client = new CloudFrontKeyValueStoreClient({
      region: "us-east-1"
    });
    const input = {
      KvsARN: "arn:aws:cloudfront::123456789012:key-value-store/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    };
    const command = new DescribeKeyValueStoreCommand(input);

    const response = await client.send(command);
  } catch (e) {
    console.log(e);
  }
})();
```

Lambda@Edge を使用したエッジでのカスタマイズ

Lambda@Edge は AWS Lambda の拡張です。Lambda@Edge は、 が CloudFront 配信するコンテンツをカスタマイズする関数を実行できるコンピューティングサービスです。1つのリージョン、米国東部 (バージニア北部) で Node.js または Python 関数を作成し、それらをビューワーに近い AWS ロケーションで実行することができ、サーバーをプロビジョニングしたり管理したりする必要はありません。Lambda@Edge は、1日あたり数個のリクエストから1秒あたり数千のリクエストまで自動的にスケーリングします。オリジンサーバーの代わりに、ビューワーの最寄りの AWS ロケーションでリクエストを処理すると、レイテンシーが大幅に軽減され、ユーザーエクスペリエンスが向上します。

CloudFront デイストリビューションを Lambda@Edge 関数に関連付けると、 は CloudFront エッジロケーションでリクエストとレスポンスを CloudFront インターセプトします。Lambda 関数は、次の CloudFront イベントが発生したときに実行できます。

- がビューワーからリクエスト CloudFront を受け取る場合 (ビューワーリクエスト)
- がリクエストをオリジン CloudFront に転送する前に (オリジンリクエスト)
- がオリジンからレスポンス CloudFront を受信する場合 (オリジンレスポンス)
- がビューワーにレスポンスを CloudFront 返す前 (ビューワーレスポンス)

Lambda@Edge の処理にはさまざまな用途があります。次に例を示します。

- Lambda 関数は Cookie をチェックして URL を書き換え、A/B テスト用に異なるバージョンのサイトをユーザーに表示できます。
- CloudFront は、デバイスに関する情報を含む User-Agentヘッダーをチェックすることで、使用しているデバイスに基づいて異なるオブジェクトをビューワーに返すことができます。例えば、CloudFront はデバイスの画面サイズに基づいて異なるイメージを返すことができます。同様に、この関数は Refererヘッダーの値を考慮し、CloudFront が使用可能な解像度が最も低いポットにイメージを返すように設定できます。
- または、他の条件の Cookie を確認できます。例えば、靴を販売する小売ウェブサイトで、Cookie を使用してユーザーがどの色のレストランを選択したかを示す場合、Lambda 関数はリクエストを変更して、 が CloudFront選択した色のレストランのイメージを返すようにできます。
- Lambda 関数は、CloudFront ビューワーリクエストまたはオリジンリクエストイベントが発生したときに HTTP レスポンスを生成できます。
- 関数は、リクエストをオリジン CloudFront に転送する前に、ヘッダーまたは認証トークンを検査し、コンテンツへのアクセスを制御するヘッダーを挿入できます。

- Lambda 関数は、外部リソースのネットワーク呼び出しを実行して、ユーザー認証情報を確認したり、追加のコンテンツを取得してレスポンスをカスタマイズしたりすることもできます。

サンプルコードと追加の例については、「[Lambda@Edge 関数の例](#)」を参照してください。

トピック

- [Lambda@Edge 関数の作成と使用の開始](#)
- [Lambda@Edge 用の IAM アクセス権限とロールの設定](#)
- [Lambda@Edge 関数の記述と作成](#)
- [Lambda@Edge 関数のトリガーの追加](#)
- [Lambda@Edge 関数のテストとデバッグ](#)
- [Lambda@Edge 関数とレプリカの削除](#)
- [Lambda@Edge イベント構造](#)
- [リクエストとレスポンスを使用する](#)
- [Lambda@Edge 関数の例](#)

Lambda@Edge 関数の作成と使用の開始

Lambda@Edge 関数を使用すると多くの有益な操作を実行できますが、初めて使用を開始するときには少し複雑に感じられる場合があります。このセクションでは、Lambda@Edge が と連携する方法を大まかに説明 CloudFront し、簡単な例を順を追って説明する [チュートリアルを提供します](#)。

Tip

Lambda@Edge の動作に精通し、Lambda@Edge 関数を作成したら、お客様独自のカスタムソリューションに Lambda@Edge を使用する方法について詳しく学習します。[関数の作成と更新](#)、[イベント構造](#)、トリガー [CloudFront の追加](#)について詳しく説明します。また、さらに多くのアイデアやコードサンプルを「[Lambda@Edge 関数の例](#)」で参照できます。

ここでは、で Lambda 関数を作成して使用する方法の概要を示します CloudFront。

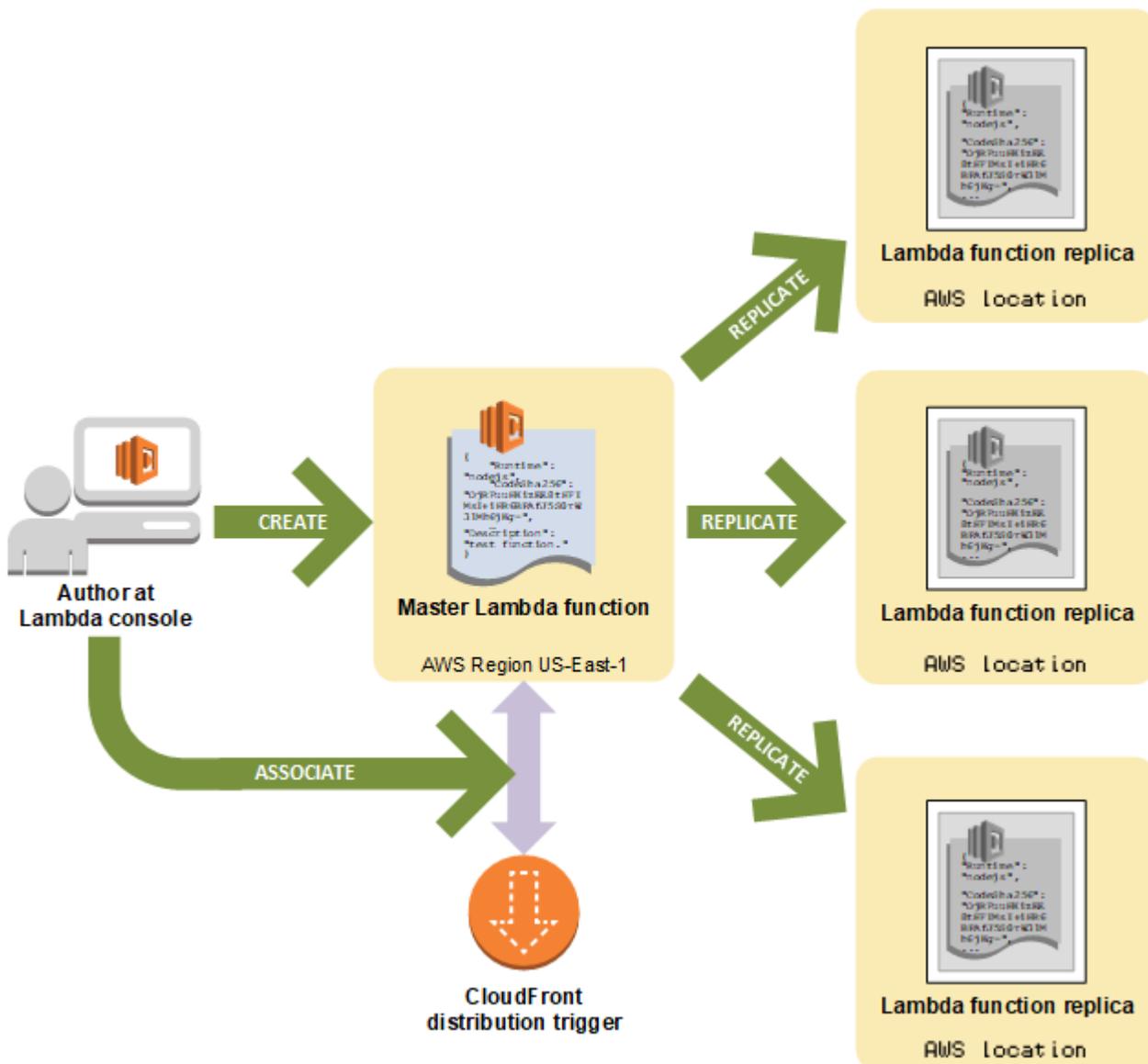
1. AWS Lambda コンソールで、米国東部 (バージニア北部) リージョンに Lambda 関数を作成します。(関数をプログラムで作成することもできます。たとえば、次のいずれかの AWS SDK を使用します)。

2. 番号付きバージョンの関数を保存して発行します。

関数を変更する場合、米国東部 (バージニア北部) リージョンで関数の \$LATEST バージョンを編集する必要があります。次に、と連携するように設定する前に CloudFront、新しい番号付きバージョンを公開します。

3. 関数が適用される CloudFront デイストリビューションとキャッシュ動作を選択します。次に、関数を実行する CloudFront イベント (トリガー) を 1 つ以上指定します。例えば、ガビューワーからリクエスト CloudFront を受信したときに実行する 関数のトリガーを作成できます。

4. トリガーを作成すると、Lambda が世界中の AWS ロケーションに関数をレプリケートします。



トピック

- [「チュートリアル:シンプルな Lambda@Edge 関数の作成」](#)

「チュートリアル:シンプルな Lambda@Edge 関数の作成」

このチュートリアルでは、CloudFront で実行されるサンプル Node.js 関数を作成および追加する支援を行うことで、Lambda@Edge の使用を開始する方法を示します。説明するこの例では、HTTP セキュリティヘッダーをレスポンスに追加します。これにより、ウェブサイトのセキュリティとプライバシーが向上します。このチュートリアルにはウェブサイトは必要ありません。その中で、ガフファイル CloudFront を取得するときに、レスポンスにセキュリティヘッダーを追加するだけです。

この例では、Lambda@Edge 関数を作成して設定するステップについて説明します。独自の Lambda@Edge ソリューションを作成するとき、同様のステップに従って同じオプションから選択できます。

トピック

- [ステップ 1: AWS アカウント にサインアップする](#)
- [ステップ 2: CloudFront ディストリビューションを作成する](#)
- [ステップ 3: 関数を作成する](#)
- [ステップ 4: 関数を実行する CloudFront トリガーを追加する](#)
- [ステップ 5: 関数の実行を確認する](#)
- [ステップ 6: 問題のトラブルシューティングを行う](#)
- [ステップ 7: リソース例をクリーンアップする](#)
- [その他のリソース](#)

ステップ 1: AWS アカウント にサインアップする

まだ Amazon Web Services にサインアップしていない場合は、<https://aws.amazon.com/> からサインアップします。[今すぐ申し込む] を選択し、必須事項を入力します。

ステップ 2: CloudFront ディストリビューションを作成する

Lambda@Edge 関数の例を作成する前に、コンテンツの提供元となるオリジンを含む を使用する CloudFront 環境が必要です。

を初めてお使いになる方 CloudFront向けに、CloudFront はエッジロケーションのグローバルネットワークを通じてコンテンツを配信します。を使用して Lambda 関数を設定すると CloudFront、この

関数はビューワーに近いコンテンツをカスタマイズできるため、パフォーマンスが向上します。に慣れていない場合は CloudFront、チュートリアルを完了する前に数分待って、[簡単な概要を読み、コンテンツの CloudFront キャッシュと提供方法について学習してください](#)。

この例では、Amazon S3 バケットを CloudFront ディストリビューションのオリジンとして使用するディストリビューションを作成します。使用する環境が既にある場合、このステップは省略できます。

Amazon S3 オリジンを使用して CloudFront ディストリビューションを作成するには

1. サンプルコンテンツ用に、イメージファイルなど 1~2 つのファイルで Amazon S3 バケットを作成します。そのためには、[コンテンツを Amazon S3 にアップロード](#)するステップに従います。必ず、バケットのオブジェクトへのパブリック読み取りアクセス権を付与するアクセス許可を設定します。
2. 「ウェブ CloudFront ディストリビューションの作成」の手順に従って、ディストリビューションを作成し、S3 バケットをオリジンとして追加します。[CloudFront](#) ディストリビューションが既にある場合は、代わりにそのディストリビューションのオリジンとしてバケットを追加できます。

Tip

ディストリビューション ID をメモします。このチュートリアルの後半で、関数の CloudFront トリガーを追加するときに、EE653W22221KDDLなどのドロップダウンリストからディストリビューションの ID を選択する必要があります。

ステップ 3: 関数を作成する

このステップでは、Lambda 関数を作成し、Lambda コンソールで提供される設計図テンプレートで開始します。関数は、CloudFront ディストリビューションのセキュリティヘッダーを更新するコードを追加します。

Lambda または Lambda@Edge を始めて利用する場合 Lambda@Edge では、CloudFront トリガーを使用して Lambda 関数を呼び出すことができます。CloudFront ディストリビューションを Lambda 関数に関連付けると、は CloudFront エッジロケーションで CloudFront [リクエストとレスポンスをインターセプト](#)し、関数を実行します。Lambda 関数はセキュリティを向上させたり、ビューワーに近い情報をカスタマイズして、パフォーマンスを向上させたりできます。このチュートリアルでは、作成する関数で、CloudFront レスポンスのセキュリティヘッダーを更新します。

Lambda 関数を作成するときに、実行する複数のステップがあります。このチュートリアルでは、設計図テンプレートを関数の基礎として使用し、セキュリティヘッダーを設定するコードで関数を更新します。最後に、CloudFront トリガーを追加してデプロイし、関数を実行します。

Lambda 関数を作成するには

1. AWS Management Console にサインインして、AWS Lambda で <https://console.aws.amazon.com/lambda/> コンソールを開きます。

 Important

US-East-1 (バージニア北部) リージョン (us-east-1) にいることを確認します。Lambda@Edge 関数を作成するには、このリージョンに設定されている必要があります。

2. [関数の作成] を選択します。
3. 「関数の作成」ページで「設計図の使用」を選択し、検索フィールドに「」と入力して CloudFront 設計図をフィルタリング **cloudfront** します。キーワード : cloudfront が表示され、タグ付けされているすべてのブループリント CloudFront が一覧表示されます。

 Note

CloudFront ブループリントは、US-East-1 (バージニア北部) リージョン (米国東部 1) のみ使用できます。

4. 関数のテンプレートとして cloudfront-modify-response-header ブループリントを選択します。
5. 関数に関する次の情報を入力します。

名前

関数の名前を入力します。

実行ロール

関数のアクセス許可を設定する方法を選択します。推奨される基本 Lambda@Edge アクセス許可ポリシーテンプレートを使用するには、[AWS ポリシーテンプレートから新しいロールを作成する] を選択します。

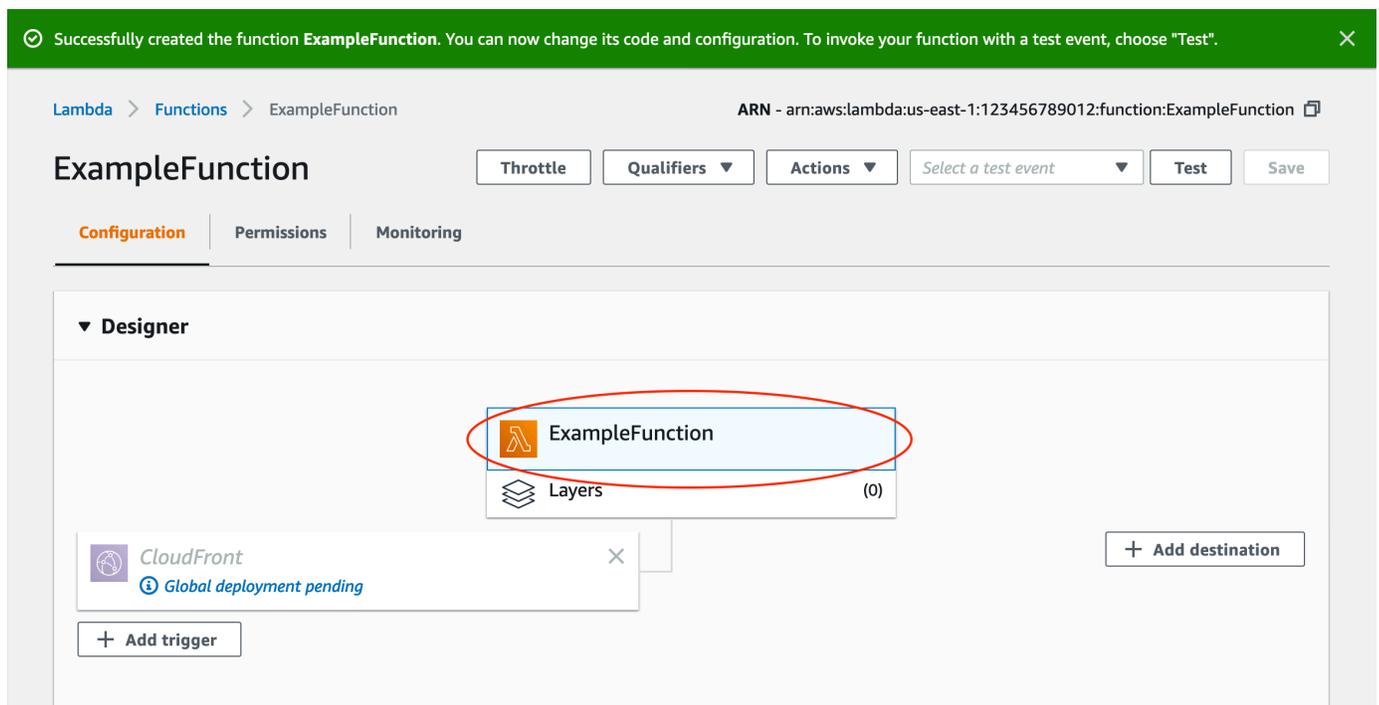
ロール名

ポリシーテンプレートが作成するロールの名前を入力します。

ポリシーテンプレート

Lambda は、ポリシーテンプレートの Basic Edge Lambda アクセス許可を自動的に追加します。これは、CloudFront関数のベースとしてブループリントを選択したためです。このポリシーテンプレートは CloudFront、が世界中の CloudFront 場所で Lambda 関数を実行できるようにする実行ロールのアクセス許可を追加します。詳細については、「[Lambda@Edge 用の IAM アクセス権限とロールの設定](#)」を参照してください。

6. [関数の作成] を選択します。Lambda により関数が作成され、次のページに関数の設定が表示されます。
7. ページの [デザイナー] セクションで、次の図に示すように関数名を選択します。この例では、関数名は `ExampleFunction` です。



8. 次の図に示すように、ページの [関数コード] セクションまで下にスクロールします。

The screenshot shows the AWS Lambda console for a function named 'ExampleFunction'. The 'Function code' section is selected, showing a code editor with the following JavaScript code:

```

1 exports.handler = async (event, context) => {
2   const response = event.Records[0].cf.response;
3   const headers = response.headers;
4
5   const headerNameSrc = 'X-Amz-Meta-Last-Modified';
6   const headerNameDst = 'Last-Modified';
7
8   if (headers[headerNameSrc.toLowerCase()]) {
9     headers[headerNameDst.toLowerCase()] = [{
10      key: headerNameDst,
11      value: headers[headerNameSrc.toLowerCase()][0].value,
12    }];
13     console.log(`Response header "${headerNameDst}" was set to ` +
14               `${headers[headerNameDst.toLowerCase()][0].value}`);
15   }
16   return response;
17 }
18 };
19

```

テンプレートコードを、オリジンが返すセキュリティヘッダーを変更する関数に置き換えます。たとえば、以下のようなコードを実行できます。

```

'use strict';
exports.handler = (event, context, callback) => {

  //Get contents of response
  const response = event.Records[0].cf.response;
  const headers = response.headers;

  //Set new headers
  headers['strict-transport-security'] = [{key: 'Strict-Transport-Security',
value: 'max-age= 63072000; includeSubdomains; preload'}];
  headers['content-security-policy'] = [{key: 'Content-Security-Policy', value:
"default-src 'none'; img-src 'self'; script-src 'self'; style-src 'self'; object-
src 'none'"}];
  headers['x-content-type-options'] = [{key: 'X-Content-Type-Options', value:
'nosniff'}];
  headers['x-frame-options'] = [{key: 'X-Frame-Options', value: 'DENY'}];
  headers['x-xss-protection'] = [{key: 'X-XSS-Protection', value: '1;
mode=block'}];
  headers['referrer-policy'] = [{key: 'Referrer-Policy', value: 'same-origin'}];

  //Return modified response

```

```
callback(null, response);
};
```

9. 更新されたコードを保存するには、[保存] を選択します。

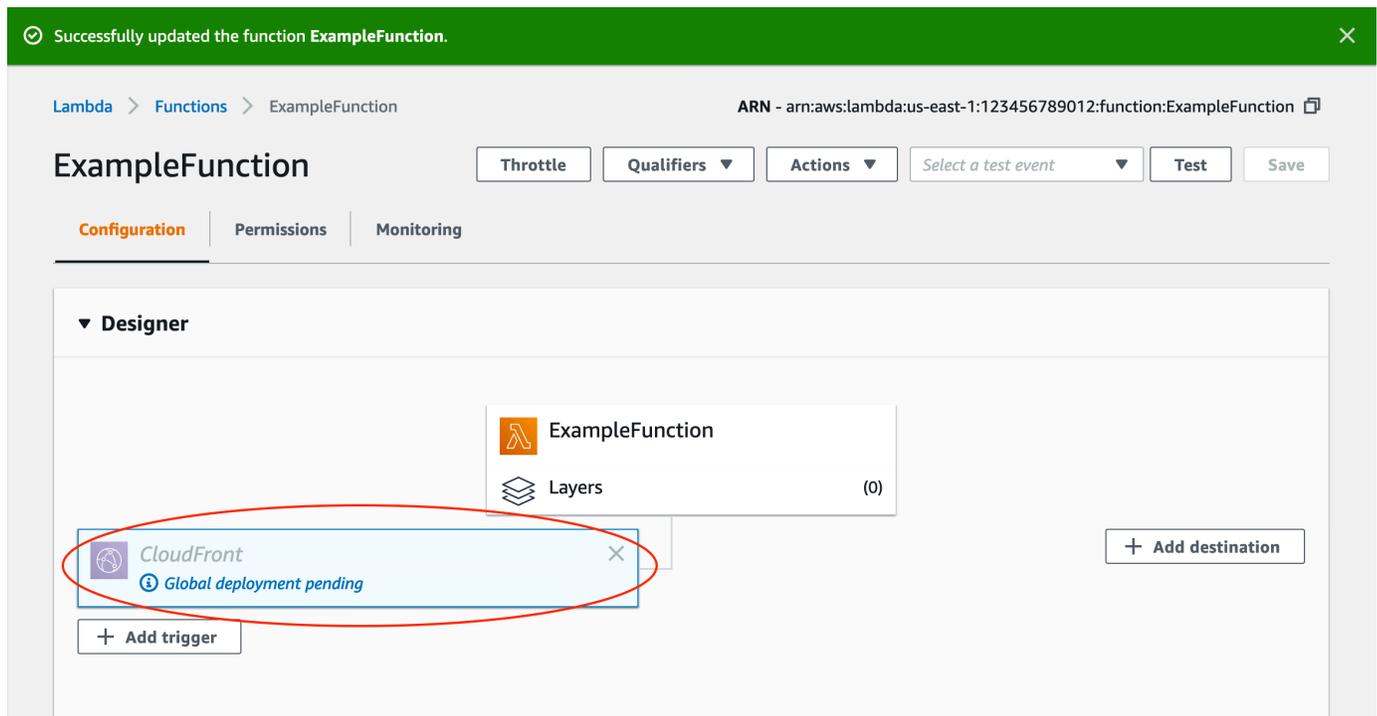
次のセクションに進み、関数を実行する CloudFront トリガーを追加します。

ステップ 4: 関数を実行する CloudFront トリガーを追加する

セキュリティヘッダーを更新する Lambda 関数を作成したので、ガディストリビューションのオリジンから CloudFront 受信するレスポンスにヘッダーを追加するように関数を実行するように CloudFront トリガーを設定します。

関数の CloudFront トリガーを設定するには

1. ページの [デザイナー] セクションで、次の図に示すように [CloudFront] を選択します。



2. ページの [トリガーの設定] セクションまで下にスクロールし、[Lambda@Edge にデプロイ] を選択します。
3. [Lambda@Edge へのデプロイ] ページの [CloudFront トリガーの設定] に、次の情報を入力します。

ディストリビューション

関数に関連付ける CloudFront ディストリビューション ID。ドロップダウンリストで、ディストリビューション ID を選択します。

キャッシュ動作

トリガーで使用するキャッシュ動作。この例では、値を * に設定したままにします。これは、ディストリビューションのデフォルトのキャッシュ動作を意味します。詳細については、「[ディストリビューションを作成または更新する場合に指定する値](#)」トピックの「[キャッシュ動作の設定](#)」を参照してください。

CloudFront イベント

関数をいつ実行するか指定するトリガー。がオリジンからレスポンスを CloudFront 返すたびに、セキュリティヘッダー関数を実行する必要があります。したがって、ドロップダウンリストで [Origin response (オリジンのレスポンス)] を選択します。詳細については、「[Lambda@Edge 関数のトリガーの追加](#)」を参照してください。

4. [Lambda@Edge へのデプロイを確認] で、このチェックボックスをオンにして、トリガーがデプロイされていることを確認し、AWS のすべての場所で関数を実行します。
5. [デプロイ] を選択して、トリガーを追加し、世界中の AWS の場所に関数をレプリケートします。次に、必要に応じて [Lambda@Edge にデプロイ] ページを閉じます。
6. 関数がレプリケートするまで待ちます。これには通常数分かかります。

[CloudFront コンソールに移動](#)してディストリビューションを表示することで、レプリケーションが完了したかどうかを確認できます。ディストリビューションのステータスが [進行中] から [デプロイ済み] に戻るまで待ちます。この場合、関数はレプリケートされたことを意味します。関数が機能することを確認するには、次のセクションのステップに従います。

ステップ 5: 関数の実行を確認する

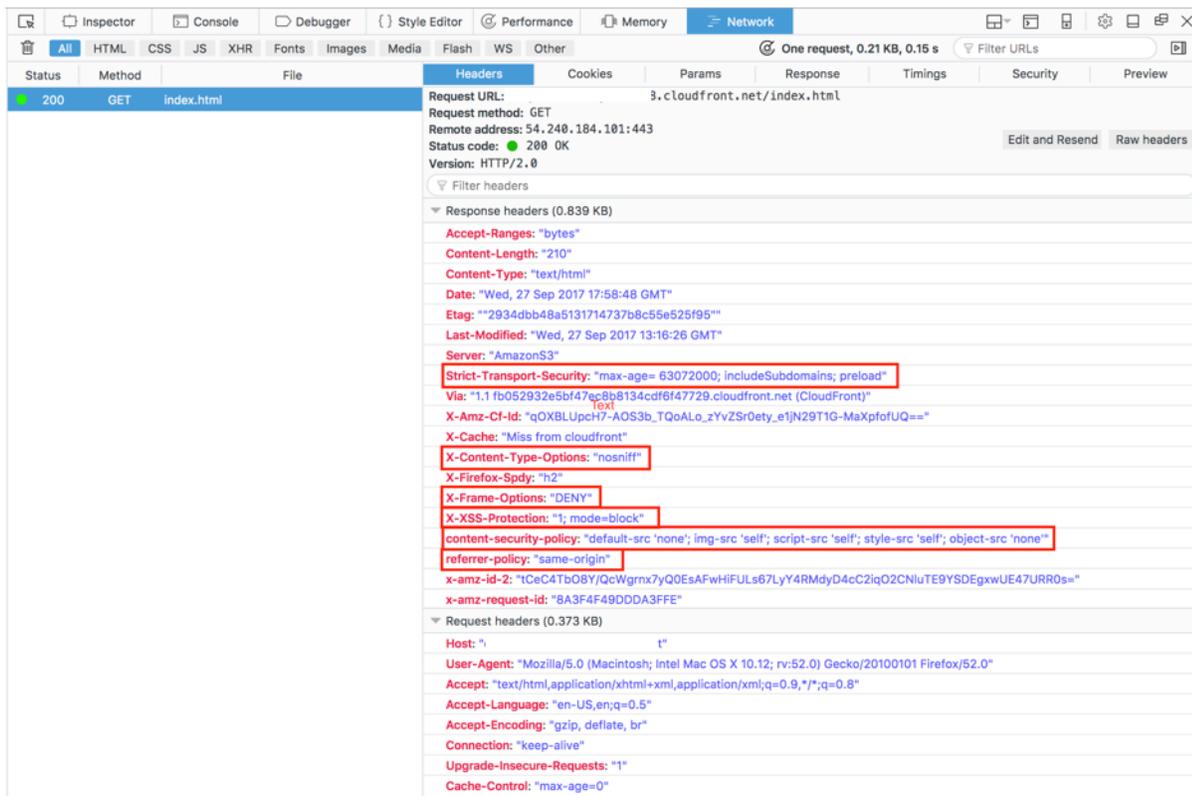
Lambda 関数を作成し、ディストリビューションに対して CloudFront 実行するようにトリガーを設定したので、関数が期待どおりに到達していることを確認します。この例では、が CloudFront 返す HTTP ヘッダーをチェックして、セキュリティヘッダーが追加されていることを確認します。

Lambda@Edge 関数でセキュリティヘッダーが追加されることを確認するには

1. ブラウザで、S3 バケット内のファイルの URL を入力します。たとえば、`https://d1111111abcdef8.cloudfront.net/image.jpg` のような URL を使用できます。

ファイル URL で使用する CloudFront ドメイン名の詳細については、「」を参照してくださいの[ファイルの URL 形式のカスタマイズ CloudFront](#)。

2. ブラウザのウェブデベロッパーツールバーを開きます。たとえば、Chrome のブラウザウィンドウで、コンテキスト (右クリック) メニューを開き、[Inspect (調査)] を選択します。
3. [Network (ネットワーク)] タブを選択します。
4. ページを再ロードしてイメージを表示し、左側のペインの HTTP リクエストを選択します。HTTP ヘッダーが別のペインに表示されます。
5. HTTP ヘッダーのリストを確認し、予期されるセキュリティヘッダーがリストに含まれていることを確認します。たとえば、次のスクリーンショットに示すようなヘッダーが表示されます。



セキュリティヘッダーがヘッダーのリストに含まれていれば、成功です。最初の Lambda@Edge 関数を正常に作成しました。がエラーを CloudFront 返すか、他の問題がある場合は、次のステップに進み、問題のトラブルシューティングを行います。

ステップ 6: 問題のトラブルシューティングを行う

がエラーを CloudFront 返すか、セキュリティヘッダーを期待どおりに追加しない場合は、CloudWatch Logs を確認することで関数の実行を調査できます。必ず、関数が実行された場所に最も近い AWS ロケーションで保存されたログを使用します。

例えば、ロンドンから ファイルを表示する場合は、CloudWatch コンソールでリージョンを欧州 (ロンドン) に変更してみてください。

Lambda@Edge 関数の CloudWatch ログを調べるには

1. にサインインAWS Management Consoleし、<https://console.aws.amazon.com/cloudwatch/> で CloudWatch コンソールを開きます。
2. [リージョン] を、ブラウザでファイルを表示したときに表示されるロケーションに変更します。これは、関数が実行されている場所です。
3. 左側のペインで、[ログ] を選択して、ディストリビューションのログを表示します。

詳細については、「[Amazon による CloudFront メトリクスのモニタリング CloudWatch](#)」を参照してください。

ステップ 7: リソース例をクリーンアップする

このチュートリアル専用の Amazon S3 バケットと CloudFront ディストリビューションを作成した場合は、割り当てたAWSリソースを削除して、料金が発生しないようにしてください。AWS リソースを削除すると、追加したコンテンツは使用できなくなります。

タスク

- [S3 バケットの削除](#)
- [ディストリビューションを削除する CloudFront](#)

S3 バケットの削除

Amazon S3 バケットを削除する前に、バケットのログ記録が無効であることを確認します。それ以外の場合、削除するバケットへのログの書き込みが AWS によって継続されます。

バケットのログ記録を無効にするには

1. <https://console.aws.amazon.com/s3/> で Amazon S3 コンソールを開きます。

2. バケットを選択し、[プロパティ] を選択します。
3. [プロパティ] から [ログ記録] を選択します。
4. [有効] チェックボックスをオフにします。
5. [Save] を選択します。

これで、バケットを削除できます。詳細については、Amazon Simple Storage Service コンソールユーザーガイドの「[バケットの削除](#)」を参照してください。

ディストリビューションを削除する CloudFront

CloudFront ディストリビューションを削除する前に、ディストリビューションを無効にする必要があります。無効になったディストリビューションは機能しなくなり、料金も発生しません。無効にしたディストリビューションはいつでも有効にすることができます。無効にしたディストリビューションを削除すると、使用できなくなります。

CloudFront ディストリビューションを無効にして削除するには

1. で CloudFront コンソールを開きます <https://console.aws.amazon.com/cloudfront/v4/home>。
2. 無効にするディストリビューションを選択してから [Disable (無効化)] を選択します。
3. 確認を求められたら、[Yes, Disable (はい、無効化する)] を選択します。
4. 無効にしたディストリビューションを選択してから [削除] を選択します。
5. 確認を求めるメッセージが表示されたら、[Yes, Delete (はい、削除します)] を選択します。

その他のリソース

Lambda@Edge 関数の動作について基本的な理解を得たので、以下を参照してさらに詳しく学習します。

- [Lambda@Edge 関数の例](#)
- [Lambda@Edge 設計のベストプラクティス](#)
- [Lambda@Edge を使用したレイテンシーの軽減とエッジへのコンピューティングの移行](#)

Lambda@Edge 用の IAM アクセス権限とロールの設定

Lambda@Edge を設定するには、特定の IAM アクセス許可および IAM 実行ロールを設定する必要があります。Lambda@Edge は、Lambda 関数を CloudFront リージョンにレプリケートし、

CloudFront ログファイル CloudWatch を使用できるようにするサービスにリンクされたロールも作成します。

トピック

- [Lambda@Edge 関数を CloudFront デистриビューションに関連付けるために必要な IAM アクセス許可](#)
- [サービスプリンシパルの関数実行ロール](#)
- [Lambda@Edge 用のサービスにリンクされたロール](#)

Lambda@Edge 関数を CloudFront デистриビューションに関連付けるために必要な IAM アクセス許可

を使用するために必要な IAM アクセス許可に加えてAWS Lambda、ユーザーは Lambda 関数 CloudFrontをデистриビューションに関連付けるために次の IAM アクセス許可が必要です。

- `lambda:GetFunction`

ユーザーが Lambda 関数の設定情報を取得し、関数を含む.zip ファイルをダウンロードするための署名付き URL を取得できるようにします。

リソースには、次の例に示すように、CloudFront イベントが発生したときに実行する関数バージョンの ARN を指定します。

```
arn:aws:lambda:us-east-1:123456789012:function:TestFunction:2
```

- `lambda:EnableReplication*`

関数コードと設定を取得するための Lambda レプリケーションサービスのアクセス許可を与えるリソースポリシーにアクセス許可を追加します。

Important

アクセス許可の末尾にあるアスタリスク (*) が必要です。`lambda:EnableReplication*`

リソースには、次の例に示すように、CloudFront イベントが発生したときに実行する関数バージョンの ARN を指定します。

```
arn:aws:lambda:us-east-1:123456789012:function:TestFunction:2
```

- `lambda:DisableReplication*`

関数を削除するための Lambda レプリケーションサービスのアクセス許可を与えるリソースポリシーにアクセス許可を追加します。

Important

アクセス許可の末尾にあるアスタリスク (*) が必要です。`lambda:DisableReplication*`

リソースには、次の例に示すように、CloudFront イベントが発生したときに実行する関数バージョンの ARN を指定します。

```
arn:aws:lambda:us-east-1:123456789012:function:TestFunction:2
```

- `iam:CreateServiceLinkedRole`

ユーザーが Lambda@Edge がで Lambda 関数をレプリケートするために使用するサービスリンクロールを作成できるようにします CloudFront。このロールを Lambda@Edge で使用する最初のディストリビューションで作成した後では、Lambda@Edge で使用する他のディストリビューションにアクセス許可を追加する必要はありません。

- `cloudfront:UpdateDistribution` または `cloudfront:CreateDistribution`

`cloudfront:UpdateDistribution` を使用してディストリビューションを更新するか、`cloudfront:CreateDistribution` を選択してディストリビューションを作成します。

詳細については、次のドキュメントを参照してください。

- このガイドの「[Amazon の Identity and Access Management CloudFront](#)」を参照してください。
- AWS Lambda デベロッパーガイドの「[AWS Lambda の認証とアクセスコントロール](#)」

サービスプリンシパルの関数実行ロール

サービスプリンシパル `lambda.amazonaws.com` と `edgelambda.amazonaws.com` が引き受けることができる IAM ロールを作成する必要があります。このロールは、関数を実行するときに、サー

ビспリンシパルが引き受けることができます。詳細については、IAM ユーザーガイドの[ロールの作成とポリシーのアタッチ \(コンソール\)](#)を参照してください。

このロールを IAM の [信頼関係] タブで追加します ([アクセス許可] タブでは追加しないでください)。

ロールの信頼ポリシーの例を示します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "lambda.amazonaws.com",
          "edgelambda.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

実行ロールに付与する必要がある許可の詳細については、AWS Lambda デベロッパーガイドの「[許可の管理: IAM ロール \(実行ロール\) の使用](#)」を参照してください。次の点に注意してください。

- デフォルトでは、CloudFront イベントが Lambda 関数をトリガーするたびに、データは CloudWatch Logs に書き込まれます。これらのログを使用する場合は、実行ロールに CloudWatch ログにデータを書き込むためのアクセス許可が必要です。定義済み を使用して AWSLambdaBasicExecutionRole 、実行ロールにアクセス許可を付与できます。

CloudWatch ログの詳細については、「」を参照してください [the section called “エッジ関数のログ”](#)。

- S3 バケットからのオブジェクトの読み取りなど、Lambda 関数コードが他の AWS リソースにアクセスする場合、そのオペレーションを実行するためのアクセス権限が実行ロールに必要です。

Lambda@Edge 用のサービスにリンクされたロール

Lambda@Edge は AWS Identity and Access Management (IAM) [サービスリンクロール](#)を使用します。サービスにリンクされたロールは、サービスに直接リンクされた一意のタイプの IAM ロールです。サービスにリンクされたロールは、サービスによって事前定義されており、お客様の代わりにサービスから他の AWS サービスを呼び出す必要のあるアクセス許可がすべて含まれています。

Lambda@Edge は、以下の IAM サービスにリンクされたロールを使用します。

- `AWSServiceRoleForLambdaReplicator` - Lambda@Edge はこのロールを使用し、Lambda@Edge が関数を AWS リージョン にレプリケートできるようにします。
- `AWSServiceRoleForCloudFrontLogger` - このロール CloudFront を使用してログファイルを CloudWatch アカウントにプッシュし、Lambda@Edge 検証エラーをデバッグできるようにします。

で Lambda@Edge トリガーを初めて追加すると CloudFront、Lambda@Edge `AWSServiceRoleForLambdaReplicator`が関数を にレプリケートできるように、 という名前のロールが自動的に作成されますAWS リージョン。このロールは、Lambda@Edge 関数を使用するために必要です。`AWSServiceRoleForLambdaReplicator` の ARN は次のようになります。

```
arn:aws:iam::123456789012:role/aws-service-role/  
replicator.lambda.amazonaws.com/AWSServiceRoleForLambdaReplicator
```

Lambda@Edge 関数の関連付けを追加すると、 という名前の 2 番目のロールが自動的に作成され `AWSServiceRoleForCloudFrontLogger`、 が Lambda@Edge エラーログファイル CloudFront を にプッシュできるようになります CloudWatch。`AWSServiceRoleForCloudFrontLogger` の ARN は次のようになります。

```
arn:aws:iam::account_number:role/aws-service-role/  
logger.cloudfront.amazonaws.com/AWSServiceRoleForCloudFrontLogger
```

サービスにリンクされたロールを使用することで、必要なアクセス許可を手動で追加する必要がなくなるため、Lambda@Edge のセットアップと使用が簡単になります。Lambda@Edge はそのサービスにリンクされたロールのアクセス許可を定義し、Lambda@Edge のみがそのロールを引き受けることができます。定義されたアクセス権限には、信頼ポリシーとアクセス権限ポリシーが含まれます。アクセス許可ポリシーを他の IAM エンティティにアタッチすることはできません。

サービスにリンクされたロールを削除する前に、関連付けられた CloudFront または Lambda@Edge リソースをすべて削除する必要があります。このようにして、アクティブなリソースにアク

セスするためにまだ必要な、サービスにリンクされたロールが削除されないようにすることで、Lambda@Edge リソースが保護されます。

サービスにリンクされたロールをサポートする他のサービスについては、「[IAM と連携する AWS サービス](#)」を開き、「サービスにリンクされたロール」列が「あり」になっているサービスを探してください。

Lambda@Edge 用のサービスにリンクされたロールのアクセス許可

Lambda@Edge は、AWSServiceRoleForLambdaReplicator および AWSServiceRoleForCloudFrontLogger という名前の 2 つのサービスにリンクされたロールを使用します。以下のセクションでは、それらの各ロールのアクセス許可を管理する方法について説明します。

Lambda Replicator 用のサービスにリンクされたロールのアクセス許可

このサービスにリンクされたロールにより、Lambda が Lambda@Edge 関数を AWS リージョンにレプリケートできるようになります。

サービスにリンクされたロールである AWSServiceRoleForLambdaReplicator は、その引き受け時に次のサービスを信頼します: `replicator.lambda.amazonaws.com`

このロールのアクセス権限ポリシーは、Lambda@Edge が以下のアクションを指定されたリソースに対して実行することを許可します。

- アクション: `arn:aws:lambda:*:*:function:*` 上で `lambda:CreateFunction`
- アクション: `arn:aws:lambda:*:*:function:*` 上で `lambda>DeleteFunction`
- アクション: `arn:aws:lambda:*:*:function:*` 上で `lambda:DisableReplication`
- アクション: all AWS resources 上で `iam:PassRole`
- アクション: all AWS resources 上で `cloudfront:ListDistributionsByLambdaFunction`

ロガーのサービス CloudFront にリンクされたロールのアクセス許可

このサービスにリンクされたロールにより、CloudFront は Lambda@Edge 検証エラーのデバッグに役立つログファイルを CloudWatch アカウントにプッシュできます。

サービスにリンクされたロールである AWSServiceRoleForCloudFrontLogger は、その引き受け時に次のサービスを信頼します: `logger.cloudfront.amazonaws.com`

このロールのアクセス権限ポリシーは、Lambda@Edge が以下のアクションを指定されたリソースに対して実行することを許可します。

- アクション: `arn:aws:logs:*:*:log-group:/aws/cloudfront/*` 上で `logs:CreateLogGroup`
- アクション: `arn:aws:logs:*:*:log-group:/aws/cloudfront/*` 上で `logs:CreateLogStream`
- アクション: `arn:aws:logs:*:*:log-group:/aws/cloudfront/*` 上で `logs:PutLogEvents`

IAM エンティティ (ユーザー、グループ、ロールなど) で Lambda@Edge のサービスにリンクされたロールを削除できるように、アクセス許可を設定する必要があります。詳細については、『IAM ユーザーガイド』の「[サービスにリンクされたロールの権限](#)」を参照してください。

Lambda@Edge 用のサービスにリンクされたロールの作成

通常、Lambda@Edge のサービスにリンクされたロールを手動で作成することはありません。以下のシナリオで、サービスによってロールが自動的に作成されます。

- トリガーを初めて作成すると、サービスによってロール `AWSServiceRoleForLambdaReplicator` が作成されます (まだ存在しない場合)。このロールにより、Lambda が Lambda@Edge 関数を AWS リージョン にレプリケートできるようになります。

このサービスにリンクされたロールを削除した場合、Lambda@Edge の新しいトリガーをディストリビューションに追加すると、そのロールは再び作成されます。

- Lambda@Edge の関連付けを持つ CloudFront ディストリビューションを更新または作成すると、サービスによってロール が作成されます。このロールがまだ存在しない `AWSServiceRoleForCloudFrontLogger` 場合は、 ログファイルを CloudFront にプッシュできるようになります CloudWatch。

サービスにリンクされたロールを削除すると、Lambda@Edge の関連付けを持つ CloudFront ディストリビューションを更新または作成すると、そのロールが再び作成されます。

これらのサービスにリンクされたロールを手動で作成する必要がある場合は、AWS CLI を使用して次のコマンドを実行します。

AWSServiceRoleForLambdaReplicator ロールを作成するには

```
aws iam create-service-linked-role --aws-service-name
replicator.lambda.amazonaws.com
```

AWSServiceRoleForCloudFrontLogger ロールを作成するには

```
aws iam create-service-linked-role --aws-service-name
logger.cloudfront.amazonaws.com
```

Lambda@Edge のサービスにリンクされたロールの編集

Lambda@Edge のサービスにリンクされたロール AWSServiceRoleForLambdaReplicator または AWSServiceRoleForCloudFrontLogger を編集することはできません。サービスによってサービスにリンクされたロールが作成された後は、多くのエンティティでそのロールが参照されるため、そのロール名は変更できません。ただし、IAM を使用してロールの説明を編集することはできます。詳細については、『IAM ユーザーガイド』の「[サービスにリンクされたロールの編集](#)」を参照してください。

CloudFront サービスにリンクされたロールAWS リージョンでサポートされています

CloudFront は、以下の で Lambda@Edge のサービスにリンクされたロールの使用をサポートしますAWS リージョン。

- 米国東部 (バージニア北部) – us-east-1
- 米国東部 (オハイオ) – us-east-2
- 米国西部 (北カリフォルニア) – us-west-1
- 米国西部 (オレゴン) – us-west-2
- アジアパシフィック (ムンバイ) – ap-south-1
- アジアパシフィック (ソウル) – ap-northeast-2
- アジアパシフィック (シンガポール) – ap-southeast-1
- アジアパシフィック (シドニー) – ap-southeast-2
- アジアパシフィック (東京) – ap-northeast-1
- 欧州 (フランクフルト) – eu-central-1
- 欧州 (アイルランド) – eu-west-1
- 欧州 (ロンドン) – eu-west-2
- 南米 (サンパウロ) – sa-east-1

Lambda@Edge 関数の記述と作成

Lambda@Edge を使用するには、Lambda 関数のコードを記述し、特定の CloudFront イベント (トリガー) に基づいて関数を実行するAWS Lambdaのようにを設定します。関数を実行するように Lambda 関数をセットアップするには、Lambda で関数の create 関数オプションを使用します。

AWS コンソールを使用して Lambda 関数と CloudFront トリガーを操作することも、APIs を使用してプログラムで Lambda@Edge を操作することもできます。

- AWS Lambda コンソールを使用する場合、Lambda 関数を作成するには Lambda コンソールのみを使用できることに注意してください。Amazon CloudFront コンソールを使用して関数を作成することはできません。
- Lambda@Edge をプログラムで使用する場合、いくつかのリソースが役立ちます。詳細については、「[Lambda@Edge 関数と CloudFront トリガーをプログラムで作成する](#)」を参照してください。

Note

AWS Lambda コンソールまたは CloudFront コンソールを使用して、Lambda@Edge 関数のトリガーを追加できます。

トピック

- [Lambda@Edge 関数のコンテンツの記述](#)
- [Lambda コンソールで Lambda@Edge 関数を作成する](#)
- [Lambda@Edge 関数と CloudFront トリガーをプログラムで作成する](#)
- [Lambda@Edge 関数の編集](#)

Lambda@Edge 関数のコンテンツの記述

Lambda@Edge 関数の記述を支援するため、いくつかのリソースがあります。

- Lambda@Edge 関数に使用できるイベント構造の詳細については、「[Lambda@Edge イベント構造](#)」を参照してください。
- A/B テストの関数や、HTTP リダイレクトを生成する関数など、Lambda@Edge 関数の例を確認するには、「[Lambda@Edge 関数の例](#)」を参照してください。

Lambda@Edge で Node.js または Python を使用するためのプログラミングモデルは、AWS リージョンで Lambda を使用するプログラミングモデルと同じです。詳細については、「[Node.js を使用した Lambda 関数の作成](#)」または「[Python を使用した Lambda 関数の作成](#)」を参照してください。

Lambda@Edge コードで、callback パラメータを含めて、リクエストまたはレスポンスイベントの該当するオブジェクトを返します。

- リクエストイベント - レスポンスに `cf.request` オブジェクトを含めます。

レスポンスを生成している場合は、レスポンスに `cf.response` オブジェクトを含めます。詳細については、「[リクエストトリガーでの HTTP レスポンスの生成](#)」を参照してください。

- レスポンスイベント - レスポンスに `cf.response` オブジェクトを含めます。

Lambda コンソールで Lambda@Edge 関数を作成する

CloudFront イベントに基づく Lambda 関数を実行する AWS Lambda ように を設定するには、次の手順に従います。

ターゲットの Lambda@Edge 関数を作成するには

1. AWS Management Console にサインインして AWS Lambda コンソール (<https://console.aws.amazon.com/lambda/>) を開きます。
2. 1 つ以上の Lambda 関数が既にある場合は、[Create function] を選択します。

関数がない場合は、[Get Started Now] を選択します。

3. ページの上部にあるリージョンのリストで、[米国東部 (バージニア北部)] を選択します。
4. 独自のコードを使用して関数を作成するか、CloudFront 設計図から始まる関数を作成します。
 - 独自のコードを使用して関数を作成するには、[Author from scratch] を選択します。
 - のブループリントのリストを表示するには CloudFront、フィルターフィールドに `cloudfront` と入力し、Enter キーを押します。

使用したい設計図を見つけたら、設計図の名前を選択します。

5. [Basic information] セクションで、以下の値を指定します。

名前

関数の名前を入力します。

ロール

[Create new role from template(s)] を選択します。

Note

この値を選択すると、すぐに開始されます。または、[Choose an existing role] か [Create a custom role] を選択できます。これらのいずれかを選択する場合は、プロンプトに従ってこのセクションの情報を入力します。

ロール名

ロールの名前を入力します。

ポリシーテンプレート

[Basic Edge Lambda permissions] を選択します。

- ステップ 4 で [Author from scratch] を選択した場合は、ステップ 7 に進んでください。

ステップ 4 で設計図を選択した場合、cloudfront セクションでは、この関数を CloudFront ディストリビューションおよび CloudFront イベント内のキャッシュに関連付けるトリガーを 1 つ作成できます。この時点で [Remove] を選択することをお勧めします。そのため、関数の作成時にトリガーはありません。後でトリガーを追加できます。

Important

後でトリガーを追加するのはなぜでしょうか。通常、トリガーを追加する前に、関数をテストおよびデバッグすることが最適です。代わりにトリガーを追加するように選択する場合、関数を作成するとすぐに関数が実行され、世界各地の AWS ロケーションへのレプリケーションが完了し、対応するディストリビューションがデプロイされます。

- [Create function] (関数の作成) を選択します。

Lambda は関数 \$LATEST とバージョン 1 の 2 つのバージョンを作成します。\$LATEST バージョンのみを編集できますが、コンソールに最初はバージョン 1 が表示されます。

- 関数を編集するには、関数の ARN の下にあるページの上にある [Version 1] を選択します。次に [Versions] タブで [\$LATEST] を選択します。(その関数をそのままにしてから戻った場合、ポタンラベルは [Qualifiers] となります)。

9. [Configuration] タブで、該当する [Code entry type] を選択します。次に、プロンプトに従ってコードを編集またはアップロードします。
10. [ランタイム] で、関数のコードに基づいて値を選択します。
11. [Tags] セクションで、該当するタグを追加します。
12. [Actions] を選択し、[Publish new version] を選択します。
13. 新しいバージョンの関数の説明を入力します。
14. [Publish] を選択します。
15. 関数をテストおよびデバッグします。Lambda コンソールのテストの詳細については、AWS Lambda デベロッパーガイドの「[コンソールを使用して Lambda 関数を作成する](#)」で Lambda 関数を手動で呼び出して、結果、ログ、メトリクスを確認するセクションを参照してください。
16. CloudFront イベントに対して関数を実行する準備ができたなら、別のバージョンを発行し、関数を編集してトリガーを追加します。詳細については、「[Lambda@Edge 関数のトリガーの追加](#)」を参照してください。

Lambda@Edge 関数と CloudFront トリガーをプログラムで作成する

Lambda@Edge 関数と CloudFront トリガーは、AWSコンソールではなく API アクションを使用してプログラムで設定できます。詳細については、次を参照してください。

- AWS Lambda デベロッパーガイドの「[API リファレンス](#)」
- [Amazon CloudFront API リファレンス](#)
- AWS CLI
 - [Lambda の create-function コマンド](#)
 - [CloudFront create-distribution コマンド](#)
 - [CloudFront create-distribution-with-tags コマンド](#)
 - [CloudFront update-distribution コマンド](#)
- [AWS SDK](#) (「SDK とツールキット」セクションを参照)。
- [AWS Tools for PowerShell コマンドレットリファレンス](#)

Lambda@Edge 関数の編集

Lambda 関数を編集する場合は、次の点に注意してください。

- 元のバージョンのラベルは [\$LATEST] です。

- \$LATEST バージョンのみを編集できます。
- \$LATEST バージョンを編集するたびに、新しい番号付きバージョンを公開する必要があります。
- \$LATEST のトリガーを作成することはできません。
- 新しいバージョンの関数を発行する場合、Lambda は以前のバージョンから新しいバージョンにトリガーを自動的にコピーしません。新しいバージョン用のトリガーを再現する必要があります。
- CloudFront イベント用のトリガーを関数に追加するときに、同じディストリビューション、キャッシュ動作、および同じ関数の以前のバージョンのイベント用のトリガーがすでに存在する場合、Lambda は以前のバージョンからトリガーを削除します。
- トリガーの追加など、CloudFront ディストリビューションを更新した後、トリガーで指定した関数が機能する前に、変更がエッジロケーションに反映されるのを待つ必要があります。

Lambda 関数を編集するには (AWS Lambda コンソール)

1. AWS Management Console にサインインして AWS Lambda コンソール (<https://console.aws.amazon.com/lambda/>) を開きます。
2. ページの上部にあるリージョンのリストで、[米国東部 (バージニア北部)] を選択します。
3. 関数のリストで、編集する関数の名前を選択します。

デフォルトでは、\$LATEST バージョンがコンソールに表示されます。以前のバージョンを表示することはできますが ([Qualifiers] を選択します)、編集できるのは \$LATEST のみです。

4. [Code (コード)] タブの [Code entry type (コードの入力タイプ)] で、ブラウザでのコードの編集、.zip ファイルのアップロード、または Amazon S3 からのファイルのアップロードを選択します。
5. [Save] または [Save and test] を選択します。
6. [Actions] を選択し、[Publish new version] を選択します。
7. [Publish new version from \$LATEST] ダイアログボックスで、新しいバージョンの説明を入力します。この説明は、自動的に生成されたバージョン番号とともにバージョンのリストに表示されます。
8. [Publish] を選択します。

新しいバージョンが自動的に最新バージョンになります。バージョン番号はページの左上隅にある [Version] ボタンに表示されます。

9. [Triggers] タブを選択します。
10. [Add trigger] を選択します。

11. [Add trigger] ダイアログボックスでチェックボックスをオンにし、[CloudFront] を選択します。

Note

関数のトリガーをすでに 1 つ以上作成している場合は、CloudFront がデフォルトのサービスです。

12. Lambda 関数をいつ実行するかを示す、次の値を指定します。

ディストリビューション ID

トリガーの追加先となるディストリビューションの ID を選択します。

キャッシュ動作

関数を実行するオブジェクトを指定するキャッシュ動作を選択します。

CloudFront イベント

関数を実行する CloudFront イベントを選択します。

トリガーとレプリケートの有効化

このチェックボックスをオンにし、Lambda が関数をリージョンに対してグローバルにレプリケートするようにします。

13. 送信 を選択します。

14. この関数のトリガーをさらに追加するには、ステップ 10~13 を繰り返します。

Lambda@Edge 関数のトリガーの追加

Lambda@Edge トリガーは、関数の実行を引き起こす CloudFront ディストリビューション、キャッシュ動作、およびイベントの 1 つの組み合わせです。関数を実行する CloudFront トリガーを 1 つ以上指定できます。例えば、ディストリビューションに設定した特定のキャッシュ動作のリクエストをビューワーから が CloudFront 受信したときに、関数を実行するトリガーを作成できます。

Tip

CloudFront キャッシュ動作に慣れていない場合は、以下に概要を示します。CloudFront ディストリビューションを作成するときは、異なるリクエストを受信したときの応答 CloudFront 方法を に指示する設定を指定します。デフォルトの設定は、ディストリビューションのデフォルトのキャッシュ動作と呼ばれます。特定のファイルタイプのリクエストを

受け取る場合など、特定の状況での CloudFront の応答方法を定義する追加のキャッシュ動作を設定できます。詳細については、「[キャッシュ動作設定](#)」を参照してください。

Lambda 関数を作成するときに、1 つのトリガーのみを指定できます。ただし、Lambda コンソールを使用するか、コンソールで CloudFront デイストリビューションを編集するという 2 つの方法のいずれかで、同じ関数にトリガーをさらに追加できます。

- 同じ CloudFront デイストリビューションの関数にトリガーを追加する場合、Lambda コンソールを使用するとうまく機能します。
- 更新するデイストリビューションを見つけやすいため、複数のデイストリビューションのトリガーを追加する場合は、CloudFront コンソールを使用する方が適しています。他の CloudFront 設定も同時に更新できます。

Note

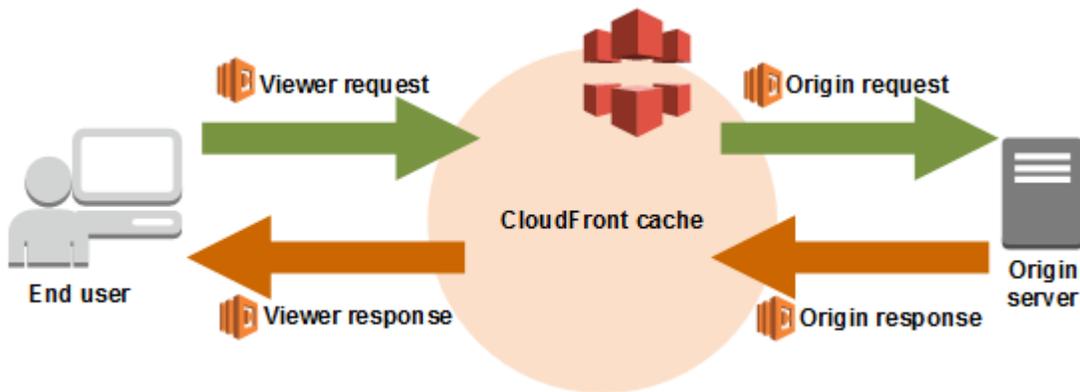
Lambda@Edge をプログラムで使用する場合、いくつかのリソースが役立ちます。詳細については、「[Lambda@Edge 関数と CloudFront トリガーをプログラムで作成する](#)」を参照してください。

トピック

- [CloudFront Lambda@Edge 関数をトリガーできる イベント](#)
- [Lambda@Edge 関数のトリガーに使用する CloudFront イベントを決定する方法](#)
- [Lambda コンソールを使ってトリガーを追加する](#)
- [CloudFront コンソールを使用したトリガーの追加](#)

CloudFront Lambda@Edge 関数をトリガーできる イベント

CloudFront デイストリビューション内のキャッシュ動作ごとに、特定の CloudFront イベントが発生したときに Lambda 関数が実行されるトリガー (関連付け) を最大 4 つ追加できます。CloudFront トリガーは、次の図に示すように、4 つの CloudFront イベントのいずれかに基づくことができます。



Lambda@Edge 関数をトリガーするために使用できる CloudFront イベントは次のとおりです。

ビューワーリクエスト

関数は、ビューワーからリクエスト CloudFront を受信したときに実行され、リクエストされたオブジェクトが CloudFront キャッシュ内にあるかどうかを確認します。

オリジンリクエスト

関数は、 がリクエストをオリジン CloudFront に転送する場合にのみ実行されます。リクエストされたオブジェクトが CloudFront キャッシュにある場合、関数は実行されません。

オリジンレスポンス

関数は、オリジンからレスポンス CloudFront を受信した後、レスポンスでオブジェクトをキャッシュする前に実行されます。関数は、オリジンからエラーが返された場合でも実行されることに注意してください。

次の場合には関数は実行されません。

- リクエストされたファイルが CloudFront キャッシュにあり、有効期限が切れていない場合。
- オリジンリクエストイベントによってトリガーされた関数からレスポンスが生成された場合。

ビューワーレスポンス

リクエストされたファイルがビューワーに返される前に関数が実行されます。ファイルが CloudFront キャッシュ内にすでに存在するかどうかに関係なく、関数が実行されることに注意してください。

次の場合には関数は実行されません。

- オリジンが HTTP ステータスコードとして 400 以上を返した場合。
- カスタムエラーページが返された場合。

- ビューワーリクエストイベントによってトリガーされた関数からレスポンスが生成された場合。
- が HTTP リクエストを HTTPS CloudFront に自動的にリダイレクトする場合 (の値が HTTP を [ビューワープロトコルポリシー](#) HTTPS にリダイレクトする場合) 。

同じキャッシュ動作に複数のトリガーを追加する場合、各トリガーに対して同じ関数を実行することも、異なる関数を実行することもできます。また、複数のディストリビューションに同じ関数を関連付けることもできます。

Note

CloudFront イベントによって Lambda 関数の実行がトリガーされると、 が CloudFront 続行する前に関数が終了する必要があります。例えば、Lambda 関数が CloudFront ビューワーリクエストイベントによってトリガーされた場合、Lambda CloudFront 関数の実行が終了するまで、ビューワーにレスポンスを返したり、リクエストをオリジンに転送したりすることはありません。つまり、Lambda 関数をトリガーするリクエストごとにリクエストのレイテンシーが長くなるため、関数をできるだけ速く実行する必要があります。

Lambda@Edge 関数のトリガーに使用する CloudFront イベントを決定する方法

Lambda 関数のトリガーに使用する CloudFront イベントを決定するときは、次の点を考慮してください。

Lambda 関数によって変更されたオブジェクトをキャッシュ CloudFront しますか？

Lambda 関数によって変更されたオブジェクトをキャッシュ CloudFront して、次回リクエストされたときに がエッジロケーションからオブジェクトを提供 CloudFront できるようにする場合は、オリジンリクエストまたはオリジンレスポンスイベントを使用します。これにより、オリジンの負荷と以降のリクエストのレイテンシーが軽減され、以降のリクエストで Lambda@Edge を呼び出すコストが削減されます。

例えば、オリジンから返されたオブジェクトのヘッダーを追加、削除、または変更し、結果を CloudFront キャッシュする場合は、オリジンレスポンスイベントを使用します。

すべてのリクエストに対して関数を実行するかどうか

がディストリビューションに対して CloudFront 受信するすべてのリクエストに対して関数を実行する場合は、ビューワーリクエストまたはビューワーレスポンスイベントを使用します。オリジン

ンリクエストイベントとオリジンレスポンスイベントは、リクエストされたオブジェクトがエッジロケーションにキャッシュされておらず、リクエストをオリジン CloudFront に転送する場合にのみ発生します。

関数でキャッシュキーを変更するかどうか

キャッシュ条件として使用している値を関数で変更する場合はビューワーリクエストイベントを使用します。たとえば、関数で URL を変更してパスに言語の省略形を含める場合 (ユーザーがドロップダウンリストから言語を選択した場合など) は、ビューワーリクエストイベントを使用します。

- ビューワーリクエストの URL – `https://example.com/en/index.html`
- リクエストがドイツの IP アドレスから送信された場合の URL – `https://example.com/de/index.html`

Cookie またはリクエストヘッダーをキャッシュ条件として使用している場合もビューワーリクエストイベントを使用します。

Note

関数が Cookie またはヘッダーを変更する場合は、リクエストの該当する部分をオリジン CloudFront に転送するようにを設定します。詳細については、次のトピックを参照してください。

- [Cookie に基づくコンテンツのキャッシュ](#)
- [リクエストヘッダーに基づくコンテンツのキャッシュ](#)

関数がオリジンからのレスポンスに影響するかどうか

関数でリクエストに加える変更がオリジンからのレスポンスに影響する場合は、オリジンリクエストイベントを使用します。通常、ほとんどのビューワーリクエストイベントは、既にエッジキャッシュに存在するオブジェクトを使用して CloudFront がリクエストに回答するため、オリジンに転送されません。関数がオリジンリクエストイベントに基づいてリクエストを変更すると、は変更されたオリジンリクエストへのレスポンスを CloudFront キャッシュします。

Lambda コンソールを使ってトリガーを追加する

Lambda@Edge 関数にトリガーを追加するには (AWS Lambda コンソール)

1. AWS Management Console にサインインして AWS Lambda コンソール (<https://console.aws.amazon.com/lambda/>) を開きます。
2. ページの上部にあるリージョンのリストで、[米国東部 (バージニア北部)] を選択します。
3. [Functions] ページで、トリガーを追加する関数の名前を選択します。
4. [Qualifiers] を選択し、次に [Versions] タブを選択します。
5. トリガーを追加するバージョンを選択します。

Important

\$LATEST バージョンのトリガーを作成することはできず、番号付きバージョンのトリガーを作成する必要があります。

バージョンを選択すると、ボタンの名前が [Version: \$LATEST] または [Version: バージョン番号] に変わります。

6. [Triggers] タブを選択します。
7. [Add triggers] を選択します。
8. [Add trigger] ダイアログボックスでチェックボックスをオンにし、[CloudFront] を選択します。

Note

すでに 1 つ以上のトリガーを作成している場合は、CloudFront がデフォルトのサービスです。

9. Lambda 関数をいつ実行するかを示す、次の値を指定します。

ディストリビューション ID

トリガーの追加先となるディストリビューションの ID を選択します。

キャッシュ動作

関数を実行するオブジェクトを指定するキャッシュ動作を選択します。

Note

キャッシュ動作に * を指定すると、Lambda 関数はデフォルトのキャッシュ動作にデプロイされます。

CloudFront イベント

関数を実行する CloudFront イベントを選択します。

ボディを含める

関数のリクエストボディにアクセスするには、このチェックボックスをオンにします。

トリガーとレプリケートの有効化

このチェックボックスをオンにし、AWS Lambda が関数をリージョンに対してグローバルにレプリケートするようにします。

10. 送信 を選択します。

この関数は、更新された CloudFront デイストリビューションがデプロイされたときに、指定された CloudFront イベントのリクエストの処理を開始します。デイストリビューションがデプロイされているかどうかを確認するには、ナビゲーションペインで [Distributions] を選択します。デイストリビューションがデプロイされると、デイストリビューションの [Status] 列の値が、[In Progress] から [Deployed] に変わります。

CloudFront コンソールを使用したトリガーの追加

イベントのトリガー CloudFront を Lambda 関数に追加するには (CloudFront コンソール)

1. トリガーを追加する Lambda 関数の ARN を取得します。
 - a. AWS Management Console にサインインして AWS Lambda コンソール (<https://console.aws.amazon.com/lambda/>) を開きます。
 - b. ページの上部にあるリージョンのリストで、[米国東部 (バージニア北部)] を選択します。
 - c. 関数のリストで、トリガーを追加する関数の名前を選択します。
 - d. [Qualifiers] を選択し、[Versions] タブを選択して、トリガーを追加する番号付きバージョンを選択します。

⚠ Important

トリガーを追加できるのは番号付きバージョンのみです。\$LATEST には追加できません。

- e. ページの上部に表示される ARN をコピーします。次に例を示します。

```
arn:aws:lambda:us-east-1:123456789012:function:TestFunction:2
```

末尾の番号 (この例では 2) は関数のバージョン番号です。

2. で CloudFront コンソールを開きます <https://console.aws.amazon.com/cloudfront/v4/home>。
3. ディストリビューションのリストで、トリガーを追加するディストリビューションの ID を選択します。
4. [Behaviors] タブを選択します。
5. トリガーを追加するキャッシュ動作のチェックボックスをオンにし、[Edit] を選択します。
6. [Lambda Function Associations] の [Event Type] リストで、関数をいつ実行するかを選択します (ビューワーリクエスト、ビューワーレスポンス、オリジンリクエスト、またはオリジンレスポンス)。

詳細については、「[Lambda@Edge 関数のトリガーに使用する CloudFront イベントを決定する方法](#)」を参照してください。

7. 選択したイベントの発生時に実行する Lambda 関数の ARN を貼り付けます。これは、ステップ 1 でコピーした値です。
8. 関数のリクエストボディにアクセスするには、[Include Body] を選択します。

リクエストボディを置き換えるだけの場合は、このオプションを選択する必要はありません。

9. 他のイベントタイプで同じ関数を実行するには、[+] を選択してステップ 6 ~ 7 を繰り返します。
10. [Yes, Edit (はい、編集します)] を選択します。
11. このディストリビューションの他のキャッシュ動作にトリガーを追加するには、ステップ 5 ~ 9 を繰り返します。

この関数は、更新された CloudFront ディストリビューションがデプロイされたときに、指定された CloudFront イベントのリクエストの処理を開始します。ディストリビューションがデプロイされているかどうかを確認するには、ナビゲーションペインで [Distributions] を選択します。

ディストリビューションがデプロイされると、ディストリビューションの [Status] 列の値が、[In Progress] から [Deployed] に変わります。

Lambda@Edge 関数のテストとデバッグ

このトピックでは、Lambda@Edge 関数をテストおよびデバッグするための戦略を説明するセクションを示します。Lambda@Edge 関数コードをスタンドアロンでテストし、目的のタスクが完了し、統合テストを実行して、関数が で正しく動作することを確認することが重要です CloudFront。

統合テスト中または関数のデプロイ後に、HTTP 5xx CloudFront エラーなどのエラーのデバッグが必要になる場合があります。エラーは、Lambda 関数から返される無効なレスポンス、関数がトリガーされるときの実行時のエラー、または Lambda サービスによる実行スロットリングが原因のエラーの可能性があり、このトピックのセクションでは、どのタイプの障害が問題であるかを判別するための戦略、そしてその問題を解決するためのステップを共有します。

Note

エラーのトラブルシューティング時に CloudWatch ログファイルまたはメトリクスを確認するときは、関数が実行された場所に最も近いリージョンに表示または保存されることに注意してください。したがって、例えば英国のユーザーを含むウェブサイトまたはウェブアプリケーションがあり、ディストリビューションに関連付けられた Lambda 関数がある場合は、リージョンを変更してロンドンリージョンの CloudWatch メトリクスまたはログファイルを表示する必要があります AWS。詳細については、このトピックの後半の「Lambda@Edge リージョンの判別」を参照してください。

トピック

- [Lambda@Edge 関数のテスト](#)
- [での Lambda@Edge 関数エラーの特定 CloudFront](#)
- [無効な Lambda@Edge 関数レスポンス \(検証エラー\) のトラブルシューティング](#)
- [Lambda@Edge 関数実行エラーのトラブルシューティング](#)
- [Lambda@Edge リージョンの判別](#)
- [アカウントがログを にプッシュするかどうかの確認 CloudWatch](#)

Lambda@Edge 関数のテスト

Lambda 関数をテストするには、スタンドアロンテストと統合テストの 2 つのステップがあります。

スタンドアロン機能のテスト

Lambda 関数を に追加する前に CloudFront、まず Lambda コンソールのテスト機能を使用するか、他の方法を使用して機能をテストしてください。Lambda コンソールのテストの詳細については、AWS Lambda デベロッパーガイドの「[コンソールを使用して Lambda 関数を作成する](#)」で Lambda 関数を手動で呼び出して、結果、ログ、メトリクスを確認するセクションを参照してください。

で関数の オペレーションをテストする CloudFront

関数がディストリビューションに関連付けられ、CloudFront イベントに基づいて実行される統合テストを完了することが重要です。関数が正しいイベントに対してトリガーされることを確認し、CloudFront に対して有効で正しいレスポンスを返します。例えば、イベント構造が正しいこと、有効なヘッダーのみが含まれていることなどを確認してください。

Lambda コンソールで関数との統合テストを繰り返すときは、コードを変更するか、関数を呼び出す CloudFront トリガーを変更するときに、Lambda@Edge チュートリアルの手順を参照してください。例えば、チュートリアルの「[ステップ 4: 関数を実行する CloudFront トリガーを追加する](#)」のステップで説明しているように、関数の番号付きバージョンを操作していることを確認します。

変更を加えてデプロイすると、更新された関数と CloudFront トリガーがすべてのリージョンにレプリケートされるまでに数分かかることに注意してください。通常、これには数分かかりますが、最大で 15 分かかる場合があります。

CloudFront コンソールに移動してディストリビューションを表示することで、レプリケーションが完了したかどうかを確認できます。

- で CloudFront コンソールを開きます <https://console.aws.amazon.com/cloudfront/v4/home>。

ディストリビューションのステータスが [進行中] から [デプロイ済み] に戻ったことを確認します。この場合、関数はレプリケートされたことを意味します。続いて、次のセクションのステップに従って関数が機能することを確認します。

コンソールでのテストでは関数のロジックのみを検証します。また、Lambda@Edge に固有のサービスクォータ (以前は制限と呼ばれていました) は適用されないことに注意してください。

での Lambda@Edge 関数エラーの特定 CloudFront

関数ロジックが正しく動作することを確認した後も、で関数を実行すると HTTP 5xx エラーが表示されることがあります CloudFront。HTTP 5xx エラーは、Lambda 関数エラーやのその他の問題など、さまざまな理由で返されることがあります CloudFront。

- Lambda@Edge 関数を使用する場合、CloudFront コンソールのグラフを使用してエラーの原因を追跡し、修正することができます。例えば、HTTP 5xx エラーの原因が Lambda 関数によるものか、Lambda 関数 CloudFront によるものかを確認し、特定の関数については、関連するログファイルを表示して問題を調査できます。
- で HTTP エラーの一般的なトラブルシューティングを行うには CloudFront、次のトピックのトラブルシューティング手順を参照してください: [オリジンからのエラーレスポンスのトラブルシューティング](#)。

で Lambda@Edge 関数エラーが発生する原因 CloudFront

Lambda 関数が HTTP 5xx エラーの原因となる可能性がある理由はいくつかあります。実行するトラブルシューティングステップはエラーのタイプによって異なります。エラーは次のように分類されます。

Lambda 関数実行エラー

実行エラーは、関数に未処理の例外があるか、コードにエラーがあるためにが Lambda からレスポンスを受け取 CloudFront らない場合に発生します。たとえば、コードにコールバック (エラー) が含まれている場合です。詳細については、AWS Lambda デベロッパーガイドの「[Lambda 関数のエラー](#)」を参照してください。

無効な Lambda 関数レスポンスがに返されます CloudFront

関数が実行されると、は Lambda からレスポンス CloudFront を受け取ります。レスポンスのオブジェクト構造が [Lambda@Edge イベント構造](#) に従わない場合、またはレスポンスに無効なヘッダーや他の無効なフィールドが含まれている場合、エラーが返されます。

Lambda サービスクォータ (以前 CloudFront は制限と呼ばれていました) により、での実行がスロットリングされます。

Lambda サービスは各リージョンでの実行を制限し、クォータに達するとエラーが返されます。

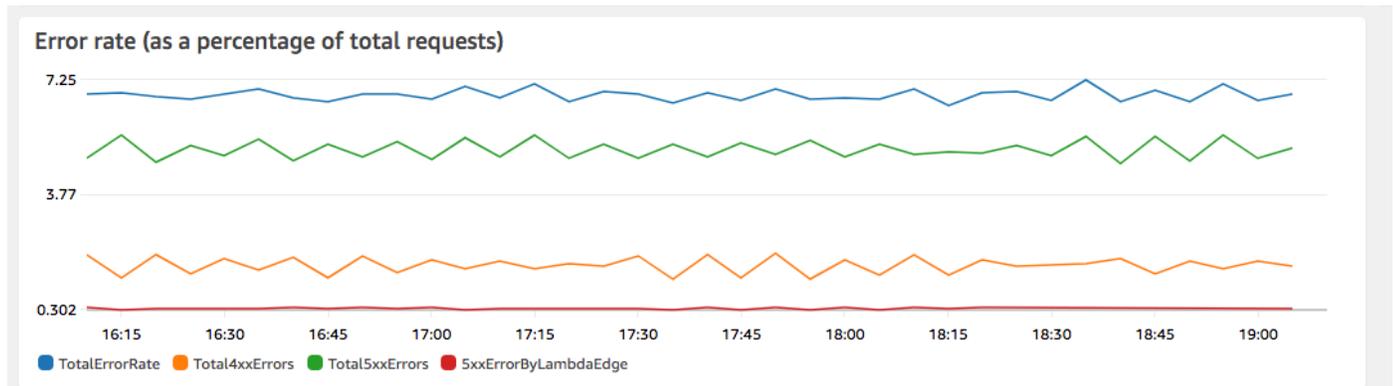
障害のタイプを判断する方法

デバッグして から返されるエラーを解決する作業を行う際にどこに焦点を絞るべきかを判断するには CloudFront、CloudFront が HTTP エラーを返す理由を特定すると便利です。開始するには、の CloudFront コンソールのモニタリングセクションに記載されているグラフを使用できますAWS Management Console。CloudFront コンソールのモニタリングセクションでグラフを表示する方法の詳細については、「」を参照してください[Amazon による CloudFront メトリクスのモニタリング CloudWatch](#)。

次のグラフは、エラーが発生源によって返されたのか Lambda 関数によって返されたのかを追跡し、Lambda 関数からのエラーである場合に問題の種類を絞り込む場合に特に役立ちます。

エラー率グラフ

各ディストリビューションの [Overview] タブに表示できるグラフの1つが、[Error rates] グラフです。このグラフは、ディストリビューションに対するすべてのリクエストに対するエラーの割合をパーセンテージで表示します。グラフは、Lambda 関数の合計エラー率、合計 4xx エラー、合計 5xx エラー、合計 5xx エラーを示しています。エラーの種類と量に基づいて、原因を調査してトラブルシューティングするための手順を実行できます。

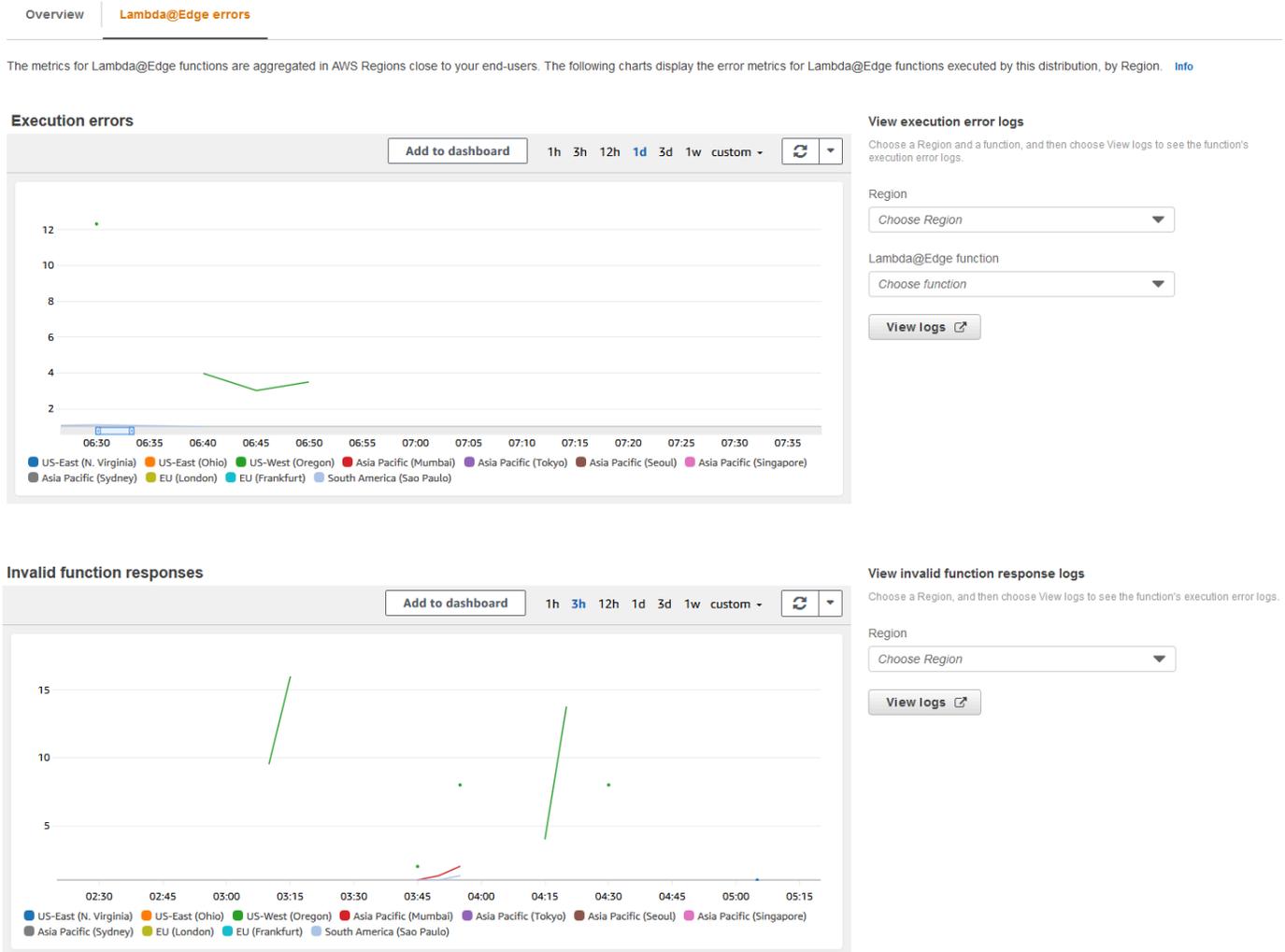


- Lambda エラーが表示された場合は、関数が返す特定の種類のエラーを調べることで、さらに詳しく調べることができます。[Lambda@Edge errors] タブには、特定の関数に関する問題を特定するのに役立つように、関数エラーをタイプ別に分類したグラフが含まれています。
- CloudFront エラーが表示された場合は、トラブルシューティングを行い、オリジンエラーを修正したり、CloudFront 設定を変更したりできます。詳細については、「[オリジンからのエラーレスポンスのトラブルシューティング](#)」を参照してください。

Execution エラーと無効な関数レスポンスグラフ

[Lambda@Edge errors] タブには、特定のディストリビューションに対する Lambda@Edge エラーをタイプ別に分類したグラフが含まれています。たとえば、1 つのグラフに AWS リージョ

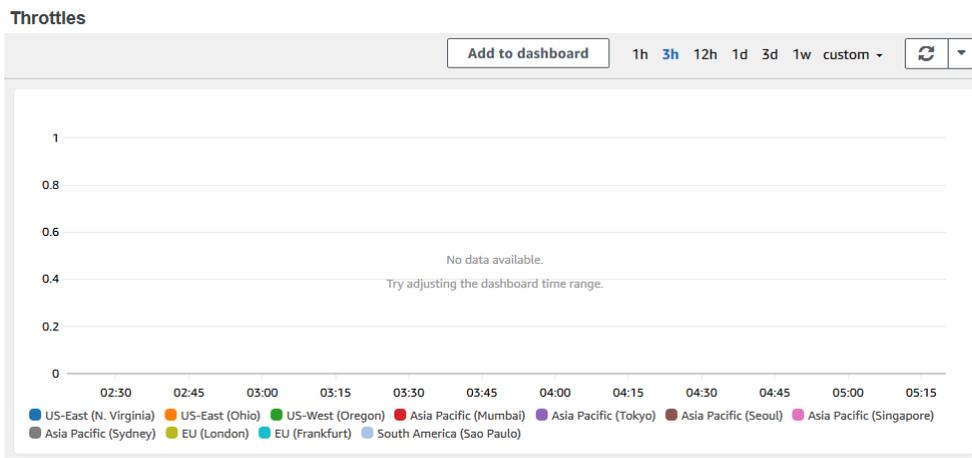
別の実行エラーがすべて表示されます。問題のトラブルシューティングを容易にするために、同じページで、地域別に特定の関数のログファイルを開いて調べることで、特定の問題を探ることができます。[View execution error logs] または [View invalid function response] で、リージョン (実行エラーの場合は関数) を選択し、[View logs] を選択します。



さらに、トラブルシューティングとエラーの修正に関する推奨事項については、この章の次のセクションを参照してください。

スロットルグラフ

[Lambda@Edge errors] タブには、[Throttles] グラフも含まれます。場合によっては、リージョンの同時実行性のクォータに達すると、Lambda サービスがリージョンごとに関数呼び出しを調整します。制限の超過エラーが表示される場合は、Lambda サービスがリージョンの実行に課すクォータに関数が達しています。クォータの増加をリクエストする方法など、詳細については、「[Lambda@Edge のクォータ](#)」を参照してください。



HTTP エラーのトラブルシューティングでこの情報を使用する方法の例については、「[AWS でコンテンツ配信をデバッグするための 4 つのステップ](#)」を参照してください。

無効な Lambda@Edge 関数レスポンス (検証エラー) のトラブルシューティング

問題が Lambda 検証エラーであることがわかった場合、Lambda 関数が無効なレスポンスを返していることを意味します CloudFront。このセクションのガイダンスに従って、関数を確認し、レスポンスが CloudFront 要件に準拠していることを確認します。

CloudFront は、次の 2 つの方法で Lambda 関数からのレスポンスを検証します。

- Lambda レスポンスは、必要なオブジェクト構造に従う必要があります。不正なオブジェクト構造の例には次のようなものがあります。解析できない JSON、必須フィールドの欠落、レスポンスの無効なオブジェクト。詳細については、「[Lambda@Edge イベント構造](#)」を参照してください。
- レスポンスには有効なオブジェクト値のみを含める必要があります。レスポンスに有効なオブジェクトが含まれるがサポートされていない値がある場合、エラーが発生します。例には、許可されていない、または読み取り専用のヘッダーの追加または更新（「[エッジ関数に対する制限](#)」を参照）、ボディサイズの上限の超過 (Lambda@Edge [エラー](#) トピックの「生成されるレスポンスのサイズに対する制限」を参照)、および無効な文字または値（「[Lambda@Edge イベント構造](#)」を参照）などがあります。

Lambda が無効なレスポンスを返すと CloudFront、Lambda 関数が実行されたリージョン CloudWatch で CloudFront プッシュするログファイルにエラーメッセージが書き込まれます。これは、無効なレスポンスがあるときにログファイルを CloudWatch に送信するデフォルトの動作です。ただし、Lambda 関数がリリースされる CloudFront 前に Lambda 関数に関連付けた場合、関

数に対して有効になっていない可能性があります。詳細については、このトピックの後半の「アカウントがログを CloudWatch にプッシュするかどうかを判断する」を参照してください。

CloudFront は、ディストリビューションに関連付けられているロググループ内の関数が実行された場所に対応するリージョンにログファイルをプッシュします。ロググループの形式は `/aws/cloudfront/LambdaEdge/DistributionId`、*DistributionId* はディストリビューションの ID です。CloudWatch ログファイルを見つけることができるリージョンを確認するには、このトピックで後述する「Lambda@Edge リージョンの確認」を参照してください。

エラーが解消された場合は、エラーになる新しいリクエストを作成し、失敗した CloudFront レスポンス (`X-Amz-Cf-Id` ヘッダー) でリクエスト ID を検索して、ログファイル内の 1 つの失敗を見つけることができます。ログファイルのエントリには、エラーが返される理由を特定するのに役立つ情報が含まれます。また、対応する Lambda リクエスト ID もリスト表示されるため、1 つのリクエストのコンテキストでの根本原因を分析することもできます。

エラーが断続的な場合は、CloudFront アクセスログを使用して失敗したリクエストのリクエスト ID を検索し、対応するエラーメッセージの CloudWatch ログを検索できます。詳細については、前のセクションの「Determining the Type of Failure」を参照してください。

Lambda@Edge 関数実行エラーのトラブルシューティング

問題が Lambda 実行エラーである場合は、Lambda 関数のログ記録ステートメントを作成し、関数の実行をモニタリングする CloudWatch ログファイルにメッセージを書き込み CloudFront、期待どおりに動作しているかどうかを判断できます。その後、CloudWatch ログファイルでこれらのステートメントを検索して、関数が機能していることを確認できます。

Note

Lambda@Edge 関数を変更していない場合でも、Lambda 関数の実行環境を更新すると、この関数に影響を与え、実行エラーが発生する可能性があります。テストおよび新しいバージョンへの移行の詳細については、「[Upcoming updates to the AWS Lambda and AWS Lambda@Edge execution environment](#)」を参照してください。

Lambda@Edge リージョンの判別

Lambda@Edge 関数がトラフィックを受信しているリージョンを確認するには、AWS Management Console の CloudFront コンソールでその関数のメトリックを表示します。メトリクスは AWS リージョンごとに表示されます。同じページで、リージョンを選択してそのリージョンのログファイルを表示し、問題を調査することができます。Lambda 関数 CloudFront の実行時に作成された

CloudWatch ログファイルを確認するには、正しいAWSリージョンのログファイルを確認する必要があります。

CloudFront コンソールのモニタリングセクションでグラフを表示する方法の詳細については、「」を参照してください[Amazon による CloudFront メトリクスのモニタリング CloudWatch](#)。

アカウントがログを にプッシュするかどうかの確認 CloudWatch

デフォルトでは、は無効な Lambda 関数レスポンスのログ記録 CloudFront を有効にし、のいずれか CloudWatch を使用してログファイルを にプッシュします[Lambda@Edge 用のサービスにリンクされたロール](#)。無効な Lambda 関数レスポンスログ機能がリリースされる CloudFront 前に に追加した Lambda@Edge 関数がある場合、トリガーを追加する CloudFrontなどして、次回 Lambda@Edge 設定を更新するとログ記録が有効になります。

アカウントでログファイルのへのプッシュ CloudWatch が有効になっていることを確認するには、次の手順を実行します。

- ログが に表示されるかどうかを確認します CloudWatch。必ず、Lambda@Edge 関数が実行されたリージョンを確認します。詳細については、「[Lambda@Edge リージョンの判別](#)」を参照してください。
- 関連するサービスにリンクされたロールが IAM のアカウントに存在するかどうかを確認する。これを行うには、<https://console.aws.amazon.com/iam/> で IAM コンソールを開いてから [ロール] を選択して、アカウントのサービスにリンクされたロールのリストを表示します。次のロールを探してください。AWSServiceRoleForCloudFrontLogger

Lambda@Edge 関数とレプリカの削除

Lambda@Edge 関数を削除できるのは、関数のレプリカが CloudFront によって削除された場合のみです。Lambda 関数のレプリカは、次のような状況では自動的に削除されます。

- すべての CloudFront ディストリビューションから関数の最後の関連付けを削除した後。複数のディストリビューションで関数が使用されている場合、最後のディストリビューションから関数の関連付けを削除した後にのみ、レプリカが削除されます。
- 関数が関連付けられた最後のディストリビューションを削除した後。

レプリカは通常、数時間以内に削除されます。Lambda@Edge 関数のレプリカを手動で削除することはできません。これにより、まだ使用中のレプリカが削除され、エラーが発生する状況を防ぐことができます。

の外部で Lambda@Edge 関数レプリカを使用するアプリケーションを構築しないでください CloudFront。これらのレプリカは、ディストリビューションとの関連付けが削除されるか、ディストリビューション自体が削除されると削除されます。外部のアプリケーションが依存するレプリカが警告なしに削除されて、エラーが発生することがあります。

CloudFront ディストリビューションから Lambda@Edge 関数の関連付けを削除するには (コンソール)

1. にサインインAWS Management Consoleし、 で CloudFront コンソールを開きます <https://console.aws.amazon.com/cloudfront/v4/home>。
2. 削除する Lambda@Edge 関数の関連付けがあるディストリビューションの ID を選択します。
3. [Behaviors] タブを選択します。
4. 削除する Lambda@Edge 関数の関連付けがあるキャッシュ動作の横にあるチェックボックスをオンにし、[Edit] を選択します。
5. [Lambda Function Associations] まで下にスクロールし、削除する各 Lambda@Edge 関数の関連付けの横にある [X] アイコンを選択します。
6. [Yes, Edit] を選択して変更を保存します。

CloudFront ディストリビューションから Lambda@Edge 関数の関連付けを削除した後、オプションで から Lambda 関数または関数バージョンを削除できますAWS Lambda。バージョンにディストリビューションが CloudFront関連付けられていない場合は、Lambda 関数の特定のバージョンを削除することもできます。Lambda 関数バージョンの関連付けをすべて削除する場合、通常は数時間後に関数を削除できます。

Lambda@Edge イベント構造

以下のトピックでは、トリガーされたときに Lambda@Edge 関数 CloudFront に渡されるリクエストおよびレスポンスイベントオブジェクトについて説明します。

トピック

- [動的オリジン選択](#)
- [リクエストイベント](#)
- [レスポンスイベント](#)

動的オリジン選択

[キャッシュ動作でパスパターン](#)を使用すると、リクエストされたオブジェクトのパスと名前 (images/*.jpg など) に基づいて、リクエストをオリジンにルーティングできます。Lambda@Edge を使用すると、リクエストヘッダーの値など他のプロパティに基づいても、リクエストをオリジンにルーティングできます。

この動的オリジン選択が便利な状況がいくつかあります。たとえば、グローバルな負荷分散に役立つように、地理的に異なるリージョンのオリジンにリクエストを分散させる場合です。あるいは、機能 (ボット処理、SEO 最適化、認証など) が異なるさまざまなオリジンに、リクエストを選択的にルーティングする場合です。この機能の使用方法を示すコードサンプルについては、「[コンテンツベースの動的オリジンの選択 - 例](#)」を参照してください。

CloudFront オリジンリクエストイベントでは、イベント構造の origin オブジェクトに、パスパターンに基づいてリクエストがルーティングされるオリジンに関する情報が含まれます。リクエストを別のオリジンにルーティングするように、origin オブジェクトの値を更新できます。origin オブジェクトを更新するときに、ディストリビューションのオリジンを定義する必要はありません。Amazon S3 オリジンオブジェクトをカスタムオリジンオブジェクトに置き換えたり、その逆にすることもできます。ただし、カスタムオリジンと Amazon S3 オリジンのどちらか (両方は不可) を通じて、リクエストごとに 1 つのオリジンしか指定できません。

リクエストイベント

以下のトピックでは、[ビューワーおよびオリジンリクエストイベント](#)用に が Lambda 関数に CloudFront 渡す オブジェクトの構造を示します。これらの例は、本文のない GET リクエストを示しています。例に続いて、ビューワーとオリジンリクエストイベントで使用可能なすべてのフィールドのリストを示します。

トピック

- [ビューワーリクエストの例](#)
- [オリジンリクエストの例](#)
- [リクエストイベントフィールド](#)

ビューワーリクエストの例

次の例は、ビューワーリクエストイベントオブジェクトを示しています。

```
{
  "Records": [
```

```
{
  "cf": {
    "config": {
      "distributionDomainName": "d111111abcdef8.cloudfront.net",
      "distributionId": "EDFDVBD6EXAMPLE",
      "eventType": "viewer-request",
      "requestId": "4TyzHTaYWb1GX1qTfsHhEqV6HUDd_BzoBZnwfnvQc_1oF26C1koUSEQ=="
    },
    "request": {
      "clientIp": "203.0.113.178",
      "headers": [
        {
          "key": "Host",
          "value": "d111111abcdef8.cloudfront.net"
        }
      ],
      "user-agent": [
        {
          "key": "User-Agent",
          "value": "curl/7.66.0"
        }
      ],
      "accept": [
        {
          "key": "accept",
          "value": "*/*"
        }
      ]
    },
    "method": "GET",
    "queryString": "",
    "uri": "/"
  }
}
```

オリジンリクエストの例

次の例は、オリジンリクエストイベントオブジェクトを示しています。

```
{
```

```
"Records": [
  {
    "cf": {
      "config": {
        "distributionDomainName": "d111111abcdef8.cloudfront.net",
        "distributionId": "EDFDVBD6EXAMPLE",
        "eventType": "origin-request",
        "requestId": "4TyzHTaYWb1GX1qTfsHhEqV6HUDd_BzoBZnwfnvQc_1oF26ClkoUSEQ=="
      },
      "request": {
        "clientIp": "203.0.113.178",
        "headers": [
          {
            "key": "X-Forwarded-For",
            "value": "203.0.113.178"
          }
        ],
        "user-agent": [
          {
            "key": "User-Agent",
            "value": "Amazon CloudFront"
          }
        ],
        "via": [
          {
            "key": "Via",
            "value": "2.0 2afae0d44e2540f472c0635ab62c232b.cloudfront.net
(CloudFront)"
          }
        ],
        "host": [
          {
            "key": "Host",
            "value": "example.org"
          }
        ],
        "cache-control": [
          {
            "key": "Cache-Control",
            "value": "no-cache"
          }
        ]
      }
    }
  ],
},
```

```
    "method": "GET",
    "origin": {
      "custom": {
        "customHeaders": {},
        "domainName": "example.org",
        "keepaliveTimeout": 5,
        "path": "",
        "port": 443,
        "protocol": "https",
        "readTimeout": 30,
        "sslProtocols": [
          "TLSv1",
          "TLSv1.1",
          "TLSv1.2"
        ]
      }
    },
    "queryString": "",
    "uri": "/"
  }
}
]
```

リクエストイベントフィールド

リクエストイベントオブジェクトデータは、`config (Records.cf.config)` と `request (Records.cf.request)` の 2 つのサブオブジェクトに含まれています。次のリストは、各サブオブジェクトのフィールドを示しています。

設定オブジェクトのフィールド

次のリストでは、`config` オブジェクト (`Records.cf.config`) のフィールドについて説明します。

distributionDomainName (読み取り専用)

リクエストに関連付けられているディストリビューションのドメイン名。

distributionID (読み取り専用)

リクエストに関連付けられているディストリビューションの ID。

eventType (読み取り専用)

リクエストに関連付けられているトリガーのタイプ (viewer-request または origin-request)。

requestId (読み取り専用)

ビューワーから CloudFront へのリクエストを一意に識別する暗号化された文字列。requestId の値は CloudFront アクセスログにも x-edge-request-id として表示されます。詳細については、「[標準ログ \(アクセスログ\) の設定および使用](#)」および「[標準ログファイルフィールド](#)」を参照してください。

リクエストオブジェクトのフィールド

次のリストでは、request オブジェクト (Records.cf.request) のフィールドについて説明します。

clientIp (読み取り専用)

リクエストを行ったビューワーの IP アドレス。ビューワーが HTTP プロキシまたはロードバランサーを使用してリクエストを送った場合、この値はプロキシまたはロードバランサーの IP アドレスです。

headers (読み書き)

リクエストのヘッダー。次の点に注意してください。

- headers オブジェクトのキーは標準の HTTP ヘッダー名を小文字にしたものです。小文字のキーを使用して、大文字と小文字を区別せずにヘッダー値にアクセスできます。
- 各ヘッダーオブジェクト (headers["accept"], headers["host"] など) はキーと値のペアの配列です。返されたヘッダーの配列には、リクエストの値ごとに 1 つのキーと値のペアが含まれます。
- key には、HTTP リクエストに表示されるヘッダーの大文字と小文字を区別する名前が含まれます (Host、User-Agent、X-Forwarded-For など)。
- value には、HTTP リクエストに表示されるヘッダー値が含まれます。
- Lambda 関数がリクエストヘッダーを追加または変更し、ヘッダー key フィールドを含めない場合、Lambda@Edge は指定したヘッダー名を使用してヘッダー key を自動的に挿入します。ヘッダー名をどのようにフォーマットしたかにかかわらず、自動的に挿入されるヘッダーキーの各部分は、先頭が大文字になり、ハイフン (-) で区切られます。

たとえば、ヘッダー key なしで次のようなヘッダーを追加できます。

```
"user-agent": [  
  {  
    "value": "ExampleCustomUserAgent/1.X.0"  
  }  
]
```

この例では、Lambda@Edge は "key": "User-Agent" を自動的に挿入します。

ヘッダー使用の制限の詳細については、「[エッジ関数に対する制限](#)」を参照してください。

method (読み取り専用)

リクエストの HTTP メソッド。

queryString (読み書き)

リクエスト内のクエリ文字列 (存在する場合)。リクエストにクエリ文字列が含まれていない場合でも、イベントオブジェクトには `queryString` が含まれ、値が空になります。クエリ文字列の詳細については、「[クエリ文字列パラメータに基づくコンテンツのキャッシュ](#)」を参照してください。

uri (読み書き)

リクエストされたオブジェクトの相対パス。Lambda 関数が `uri` 値を変更する場合、次の点に注意してください。

- 新しい `uri` 値は、スラッシュ (/) で始める必要があります。
- 関数で `uri` 値を変更すると、ビューワーがリクエストしているオブジェクトが変更されます。
- 関数で `uri` 値を変更しても、リクエストのキャッシュ動作や送信先オリジンは変わりません。

body (読み書き)

HTTP リクエストの本文。body 構造には、次のフィールドを含めることができます。

inputTruncated (読み取り専用)

本文が Lambda@Edge で切り捨てられたかどうかを示すブーリアン型フラグ。詳細については、「[Include Body オプションがあるリクエストボディに対する制限](#)」を参照してください。

action (読み書き)

本文で実行する予定のアクション。action のオプションは次のとおりです。

- **read-only**: これがデフォルト値です。Lambda 関数からレスポンスを返す際に、**action** が読み取り専用の場合、Lambda@Edge は **encoding** または **data** への変更をすべて無視します。
- **replace**: オリジンに送信される本文を置き換えるときに指定します。

encoding (読み書き)

本文のエンコード。Lambda@Edge が Lambda 関数に本文を公開すると、まず本文を base64-encoding に変換します。本文を置き換える **action** として **replace** を選択した場合、base64 (デフォルト) または text エンコードを使用することもできます。encoding を base64 と指定したが本文が有効な base64 でない場合、CloudFront はエラーを返します。

data (読み書き)

リクエストボディのコンテンツ。

origin (読み書き) (オリジンイベントのみ)

リクエストの送信先のオリジン。origin 構造には、オリジンが 1 つだけ含まれていなければなりません。オリジンはカスタムオリジンでも Amazon S3 オリジンでも構いません。オリジン構造には、次のフィールドを含めることができます。

customHeaders (読み取り/書き込み) (カスタムおよび Amazon S3 オリジン)

各カスタムヘッダーの名前と値のペアを指定することで、カスタムヘッダーをリクエストに含めることができます。許可されていないヘッダーを追加することはできず、同じ名前のヘッダーを `Records.cf.request.headers` に含めることもできません。[リクエストヘッダーに関する注意事項](#)は、カスタムヘッダーにも適用されます。詳細については、「[がオリジンリクエストに追加 CloudFront できないカスタムヘッダー](#)」および「[エッジ関数に対する制限](#)」を参照してください。

domainName (読み取り/書き込み) (カスタムおよび Amazon S3 オリジン)

オリジンのドメイン名。ドメイン名を空にすることはできません。

- カスタムオリジンの場合 - DNS ドメイン名を指定します (`www.example.com` など)。ドメイン名にコロン (:) を含めることはできません。また、IP アドレスにすることはできません。ドメイン名の最大長は 253 文字です。
- Amazon S3 オリジンの場合 - Amazon S3 バケットの DNS ドメイン名を指定します (`awsexamplebucket.s3.eu-west-1.amazonaws.com` など)。この名前は最大 128 文字で、すべて小文字であることが必要です。

path (読み取り/書き込み) (カスタムおよび Amazon S3 オリジン)

リクエストがコンテンツを検索するサーバーのディレクトリパス。パスは、先頭をスラッシュ (/) にする必要があります。末尾をスラッシュ (/) にすることはできません (例えば、末尾が `example-path/` は不可です)。カスタムオリジンの場合のみ、パスは URL エンコードされ、最大長は 255 文字にする必要があります。

keepaliveTimeout (読み書き) (カスタムオリジンのみ)

レスポンスの最後のパケットを受信した後、がオリジンへの接続を維持 CloudFront しようとする秒単位の時間。この値には、1 ~ 60 の範囲の数値を指定する必要があります。

port (読み書き) (カスタムオリジンのみ)

カスタムオリジンで が接続 CloudFront する必要があるポート。ポートは 80 または 443 であるか、1024 ~ 65535 の範囲の数値であることが必要です。

protocol (読み書き) (カスタムオリジンのみ)

がオリジンに接続するときに CloudFront 使用する接続プロトコル。ここには、`http` または `https` が表示されます。

readTimeout (読み書き) (カスタムオリジンのみ)

リクエストをオリジンに送信した後、レスポンスを待機 CloudFront する秒単位の時間。また、レスポンスのパケットを受信してから次のパケットを受信するまでの CloudFront 待機時間も指定します。この値には、4 ~ 60 の範囲の数値を指定する必要があります。

sslProtocols (読み書き) (カスタムオリジンのみ)

オリジンとの HTTPS 接続を確立するときに CloudFront が使用できる最低限の SSL/TLS プロトコル。値は、`TLSv1.2`、`TLSv1.1`、`TLSv1`、または `SSLv3` のいずれかです。

authMethod (読み取り/書き込み) (Amazon S3 オリジンのみ)

[オリジンアクセスアイデンティティ \(OAI\)](#) を使用している場合は、このフィールドを `origin-access-identity` に設定します。OAI を使用していない場合は、`none` に設定します。`authMethod` を `origin-access-identity` に設定した場合、いくつかの要件があります。

- `region` を指定する必要があります (次のフィールドを参照)。
- リクエストをある Amazon S3 オリジンから別のオリジンに変更する場合は、同じ OAI を使用する必要があります。
- リクエストをカスタムオリジンから Amazon S3 オリジンに変更する場合、OAI を使用することはできません。

Note

このフィールドは[オリジンアクセスコントロール \(OAC\)](#)をサポートしていません。

region (読み取り/書き込み) (Amazon S3 オリジンのみ)

Amazon S3 バケットの AWS リージョン。これは、authMethod を origin-access-identity に設定した場合にのみ必要です。

レスポンスイベント

以下のトピックでは、[ビューワーおよびオリジンレスポンスイベント](#)用に Lambda 関数 CloudFront に渡される オブジェクトの構造を示します。例に続いて、ビューワーとオリジンレスポンスイベントで使用可能なすべてのフィールドのリストを示します。

トピック

- [オリジンレスポンスの例](#)
- [ビューワーレスポンスの例](#)
- [レスポンスイベントのフィールド](#)

オリジンレスポンスの例

次の例は、オリジンレスポンスイベントオブジェクトを示しています。

```
{
  "Records": [
    {
      "cf": {
        "config": {
          "distributionDomainName": "d1111111abcdef8.cloudfront.net",
          "distributionId": "EDFDVBD6EXAMPLE",
          "eventType": "origin-response",
          "requestId": "4TyzHTaYwb1GX1qTfsHhEqV6HUDD_BzoBZnwfNvQc_1oF26ClkoUSEQ=="
        },
        "request": {
          "clientIp": "203.0.113.178",
          "headers": {
            "x-forwarded-for": [
```

```
    {
      "key": "X-Forwarded-For",
      "value": "203.0.113.178"
    }
  ],
  "user-agent": [
    {
      "key": "User-Agent",
      "value": "Amazon CloudFront"
    }
  ],
  "via": [
    {
      "key": "Via",
      "value": "2.0 8f22423015641505b8c857a37450d6c0.cloudfront.net
(CloudFront)"
    }
  ],
  "host": [
    {
      "key": "Host",
      "value": "example.org"
    }
  ],
  "cache-control": [
    {
      "key": "Cache-Control",
      "value": "no-cache"
    }
  ]
},
"method": "GET",
"origin": {
  "custom": {
    "customHeaders": {},
    "domainName": "example.org",
    "keepaliveTimeout": 5,
    "path": "",
    "port": 443,
    "protocol": "https",
    "readTimeout": 30,
    "sslProtocols": [
      "TLSv1",
      "TLSv1.1",
```

```
        "TLSv1.2"
      ]
    }
  },
  "querystring": "",
  "uri": "/"
},
"response": {
  "headers": [
    "access-control-allow-credentials": [
      {
        "key": "Access-Control-Allow-Credentials",
        "value": "true"
      }
    ],
    "access-control-allow-origin": [
      {
        "key": "Access-Control-Allow-Origin",
        "value": "*"
      }
    ],
    "date": [
      {
        "key": "Date",
        "value": "Mon, 13 Jan 2020 20:12:38 GMT"
      }
    ],
    "referrer-policy": [
      {
        "key": "Referrer-Policy",
        "value": "no-referrer-when-downgrade"
      }
    ],
    "server": [
      {
        "key": "Server",
        "value": "ExampleCustomOriginServer"
      }
    ],
    "x-content-type-options": [
      {
        "key": "X-Content-Type-Options",
        "value": "nosniff"
      }
    ]
  ]
}
```

```
    ],
    "x-frame-options": [
      {
        "key": "X-Frame-Options",
        "value": "DENY"
      }
    ],
    "x-xss-protection": [
      {
        "key": "X-XSS-Protection",
        "value": "1; mode=block"
      }
    ],
    "content-type": [
      {
        "key": "Content-Type",
        "value": "text/html; charset=utf-8"
      }
    ],
    "content-length": [
      {
        "key": "Content-Length",
        "value": "9593"
      }
    ]
  },
  "status": "200",
  "statusDescription": "OK"
}
}
}
]
```

ビューワーレスポンスの例

次の例は、ビューワーレスポンスイベントオブジェクトを示しています。

```
{
  "Records": [
    {
      "cf": {
        "config": {
          "distributionDomainName": "d1111111abcdef8.cloudfront.net",
```

```
"distributionId": "EDFDVBD6EXAMPLE",
"eventType": "viewer-response",
"requestId": "4TyzHTaYWb1GX1qTfsHhEqV6HUDD_BzoBZnwfnc_1oF26C1koUSEQ=="
},
"request": {
  "clientIp": "203.0.113.178",
  "headers": [
    {
      "key": "Host",
      "value": "d1111111abcdef8.cloudfront.net"
    }
  ],
  "user-agent": [
    {
      "key": "User-Agent",
      "value": "curl/7.66.0"
    }
  ],
  "accept": [
    {
      "key": "accept",
      "value": "*/*"
    }
  ]
},
"method": "GET",
"queryString": "",
"uri": "/"
},
"response": {
  "headers": [
    {
      "access-control-allow-credentials": [
        {
          "key": "Access-Control-Allow-Credentials",
          "value": "true"
        }
      ],
      "access-control-allow-origin": [
        {
          "key": "Access-Control-Allow-Origin",
          "value": "*"
        }
      ]
    }
  ]
},
```

```
"date": [
  {
    "key": "Date",
    "value": "Mon, 13 Jan 2020 20:14:56 GMT"
  }
],
"referrer-policy": [
  {
    "key": "Referrer-Policy",
    "value": "no-referrer-when-downgrade"
  }
],
"server": [
  {
    "key": "Server",
    "value": "ExampleCustomOriginServer"
  }
],
"x-content-type-options": [
  {
    "key": "X-Content-Type-Options",
    "value": "nosniff"
  }
],
"x-frame-options": [
  {
    "key": "X-Frame-Options",
    "value": "DENY"
  }
],
"x-xss-protection": [
  {
    "key": "X-XSS-Protection",
    "value": "1; mode=block"
  }
],
"age": [
  {
    "key": "Age",
    "value": "2402"
  }
],
"content-type": [
  {
```

```
        "key": "Content-Type",
        "value": "text/html; charset=utf-8"
      }
    ],
    "content-length": [
      {
        "key": "Content-Length",
        "value": "9593"
      }
    ]
  },
  "status": "200",
  "statusDescription": "OK"
}
}
}
]
```

レスポンスイベントのフィールド

レスポンスイベントオブジェクトデータは、`config` (`Records.cf.config`)、`request` (`Records.cf.request`)、`response` (`Records.cf.response`) の3つのサブオブジェクトに含まれています。リクエストオブジェクトのフィールドの詳細については、「[リクエストオブジェクトのフィールド](#)」を参照してください。次のリストでは、`config` および `response` サブオブジェクトのフィールドについて説明します。

設定オブジェクトのフィールド

次のリストでは、`config` オブジェクト (`Records.cf.config`) のフィールドについて説明します。

distributionDomainName (読み取り専用)

レスポンスに関連付けられているディストリビューションのドメイン名。

distributionID (読み取り専用)

レスポンスに関連付けられているディストリビューションの ID。

eventType (読み取り専用)

レスポンスに関連付けられているトリガーのタイプ (`origin-response` または `viewer-response`)。

requestId (読み取り専用)

このレスポンスが関連付けられているCloudFront リクエストへのビューワーを一意に識別する暗号化された文字列。このrequestId値は CloudFront アクセスログにも として表示されます x-edge-request-id。詳細については、「[標準ログ \(アクセスログ\) の設定および使用](#)」および「[標準ログファイルフィールド](#)」を参照してください。

レスポンスオブジェクトのフィールド

次のリストでは、response オブジェクト (Records.cf.response) のフィールドについて説明します。Lambda@Edge 関数を使用して HTTP レスポンスを生成する方法については、「[リクエストトリガーでの HTTP レスポンスの生成](#)」を参照してください。

headers (読み書き)

レスポンスのヘッダー。次の点に注意してください。

- headers オブジェクトのキーは標準の HTTP ヘッダー名を小文字にしたものです。小文字のキーを使用して、大文字と小文字を区別せずにヘッダー値にアクセスできます。
- 各ヘッダーオブジェクト (headers["content-type"]、headers["content-length"] など) はキーと値のペアの配列です。返されたヘッダーの配列には、レスポンスの値ごとに 1 つのキーと値のペアが含まれます。
- key には、HTTP レスポンスに表示されるヘッダーの大文字と小文字を区別する名前が含まれます (Content-Type、Content-Length など)。
- value には、HTTP レスポンスに表示されるヘッダー値が含まれます。
- Lambda 関数がレスポンスヘッダーを追加または変更し、ヘッダー key フィールドを含めない場合、Lambda@Edge は指定したヘッダー名を使用してヘッダー key を自動的に挿入します。ヘッダー名をどのようにフォーマットしたかにかかわらず、自動的に挿入されるヘッダーキーの各部分は、先頭が大文字になり、ハイフン (-) で区切られます。

たとえば、ヘッダー key なしで次のようなヘッダーを追加できます。

```
"content-type": [  
  {  
    "value": "text/html;charset=UTF-8"  
  }  
]
```

この例では、Lambda@Edge は "key": "Content-Type" を自動的に挿入します。

ヘッダー使用の制限の詳細については、「[エッジ関数に対する制限](#)」を参照してください。

status

レスポンスの HTTP ステータスコード。

statusDescription

レスポンスの HTTP ステータスの説明。

リクエストとレスポンスを使用する

このセクションのトピックでは、Lambda@Edge リクエストとレスポンスを使用するいくつかの方法について説明します。

トピック

- [オリジンフェイルオーバーで Lambda@Edge 関数を使用する](#)
- [リクエストトリガーでの HTTP レスポンスの生成](#)
- [オリジンレスポンストリガーでの HTTP レスポンスの更新](#)
- [Include Body オプションを選択するリクエストボディへのアクセス](#)

オリジンフェイルオーバーで Lambda@Edge 関数を使用する

Lambda@Edge 関数は、オリジングループで設定した CloudFront ディストリビューションで使用できます。例えば、高可用性を確保するために設定したオリジンフェイルオーバーに使用できます。オリジングループで Lambda 関数を使用するには、キャッシュ動作を作成するときにオリジングループのオリジンリクエストまたはオリジンレスポンストリガーで関数を指定します。

詳細については、以下を参照してください。

- オリジングループの作成: [オリジングループの作成](#)
- Lambda@Edge でのオリジンフェイルオーバーの機能 [Lambda@Edge 関数でのオリジンフェイルオーバーの使用](#)

リクエストトリガーでの HTTP レスポンスの生成

がリクエスト CloudFront を受信すると、Lambda 関数を使用して、レスポンスをオリジンに転送せずにビューワーに直接 CloudFront 返す HTTP レスポンスを生成できます。HTTP レスポンスを生成することで、オリジンの負荷が軽減され、通常はビューワーのレイテンシーも短縮されます。

以下に示しているのは、HTTP レスポンスを生成する一般的なシナリオです。

- 小さいウェブページをビューワーに返す。
- HTTP 301 または 302 ステータスコードを返して、ユーザーを別のウェブページにリダイレクトする。
- ユーザーが認証されない場合に HTTP 401 ステータスコードをビューワーに返す。

Lambda@Edge 関数は、次の CloudFront イベントが発生したときに HTTP レスポンスを生成できません。

ビューワーリクエストイベント

関数がビューワーリクエストイベントによってトリガーされると、はレスポンスをビューワーに CloudFront 返し、キャッシュしません。

オリジンリクエストイベント

関数がオリジンリクエストイベントによってトリガーされると、は、以前に関数によって生成されたレスポンスについてエッジキャッシュ CloudFront をチェックします。

- レスポンスがキャッシュにある場合、関数は実行されず、キャッシュされたレスポンスがビューワーに CloudFront 返されます。
- レスポンスがキャッシュにない場合、関数が実行され、レスポンスがビューワーに CloudFront 返され、キャッシュされます。

HTTP レスポンスを生成するためのサンプルコードを見るには、「[Lambda@Edge 関数の例](#)」を参照してください。レスポンストリガーの HTTP レスポンスを置き換えることもできます。詳細については、「[オリジンレスポンストリガーでの HTTP レスポンスの更新](#)」を参照してください。

プログラミングモデル

このセクションでは、Lambda@Edge を使用して HTTP レスポンスを生成するためのプログラミングモデルについて説明します。

トピック

- [レスポンスオブジェクト](#)
- [エラー](#)
- [必須フィールド](#)

レスポンスオブジェクト

result メソッドの callback パラメータとして返すレスポンスには、以下の構造が必要です (status フィールドのみが必須)。

```
const response = {
  body: 'content',
  bodyEncoding: 'text' | 'base64',
  headers: {
    'header name in lowercase': [{
      key: 'header name in standard case',
      value: 'header value'
    }],
    ...
  },
  status: 'HTTP status code (string)',
  statusDescription: 'status description'
};
```

レスポンスオブジェクトには、以下の値が含まれる場合があります。

body

生成されたレスポンスで返 CloudFront す本文がある場合。

bodyEncoding

body で指定した値のエンコード。有効なエンコードは text と base64 のみです。を response オブジェクト body に含めても を省略すると bodyEncoding、 は本文をテキストとして CloudFront 扱います。

bodyEncoding base64 として を指定しても本文が有効な base64 ではない場合、 はエラー CloudFront を返します。

headers

生成されたレスポンスで CloudFront 返すヘッダー。次の点に注意してください。

- headers オブジェクトのキーは標準の HTTP ヘッダー名を小文字にしたものです。小文字のキーを使用して、大文字と小文字を区別せずにヘッダー値にアクセスできます。
- 各ヘッダー (headers["accept"]、headers["host"] など) はキーと値のペアの配列です。返されたヘッダーの配列には、生成されたレスポンスの値ごとに 1 つのキーと値のペアが含まれます。

- key (省略可能) は、HTTP リクエストに表示されるヘッダーの大文字と小文字を区別する名前です (accept、host など)。
- ヘッダー値として value を指定します。
- キーと値のペアのヘッダーキー部分を含めない場合、Lambda@Edge は指定したヘッダー名を使用してヘッダーキーを自動的に挿入します。ヘッダー名をどのようにフォーマットしたかにかかわらず、挿入されるヘッダーキーは、各パートの先頭の大文字がハイフン (-) で区切られて自動的にフォーマットされます。

たとえば、ヘッダーキー 'content-type': [{ value: 'text/html; charset=UTF-8' }] なしで次のようなヘッダーを追加できます。

この例で、Lambda@Edge はヘッダーキー Content-Type を作成します。

ヘッダー使用の制限の詳細については、「[エッジ関数に対する制限](#)」を参照してください。

status

HTTP ステータスコード。ステータスコードを文字列として指定します。は、提供された次のステータスコード CloudFront を使用します。

- レスポンスでの返却
- オリジンリクエストイベントによってトリガーされた関数によってレスポンスが生成されたときの CloudFront エッジキャッシュ
- ログイン CloudFront [標準ログ \(アクセスログ\) の設定および使用](#)

status 値が 200 ~ 599 でない場合、はビューワーにエラー CloudFront を返します。

statusDescription

HTTP ステータスコードに付随して、レスポンスで CloudFront 返す説明。標準の説明 (HTTP ステータスコード 200 の場合の OK など) を使用する必要はありません。

エラー

生成された HTTP レスポンスで発生する可能性があるエラーを以下に示します。

レスポンスに本文が含まれ、ステータスに 204 (No Content) が指定されている

ビューワーリクエストによって関数がトリガーされると、は、HTTP 502 ステータスコード (Bad Gateway) の両方が true である場合にビューワーに CloudFront HTTP 502 ステータスコードを返します。

- status の値が 204 (No Content) である。
- このレスポンスに body の値が含まれている。

これは、HTTP 204 レスポンスにはメッセージ本文を含める必要がないことを述べている RFC 2616 のオプションの制限を Lambda@Edge が適用しているためです。

生成されるレスポンスのサイズ制限を超えている

Lambda 関数によって生成されるレスポンスの最大サイズは、関数をトリガーするイベントによって異なります。

- ビューワーリクエストイベント - 40 KB
- オリジンリクエストイベント - 1 MB

レスポンスが許容サイズより大きい場合、は HTTP 502 ステータスコード (Bad Gateway) をビューワーに CloudFront 返します。

必須フィールド

status フィールドは必須です。

その他のすべてのフィールドはオプションです。

オリジンレスポンストリガーでの HTTP レスポンスの更新

がオリジンサーバーから HTTP レスポンス CloudFront を受信するときに、キャッシュ動作に関連付けられたオリジンレスポンストリガーがある場合は、オリジンから返された内容を上書きするように HTTP レスポンスを変更できます。

以下に示しているのは、HTTP レスポンスを更新する一般的なシナリオです。

- オリジンがエラーステータスコード (4xx または 5xx) を返すと、ステータスを変更して HTTP 200 ステータスコードを設定し、ビューワーに返す静的な本文コンテンツを作成する。サンプルコードについては、「[例: オリジンレスポンストリガーを使用してエラーステータスコードを 200 に更新する](#)」を参照してください。
- オリジンがエラーステータスコード (4xx または 5xx) を返すと、ステータスを変更して HTTP 301 または HTTP 302 ステータスコードを設定し、ユーザーを別のウェブサイトのリダイレクトする。サンプルコードについては、「[例: オリジンレスポンストリガーを使用してエラーステータスコードを 302 に更新する](#)」を参照してください。

Note

関数は、200と 599 (両端を含む) の間のステータス値を返す必要があります。それ以外の場合は、ビューワーにエラー CloudFront を返します。

ビューワーとオリジンのリクエストイベントの HTTP レスポンスを置き換えることもできます。詳細については、「[リクエストトリガーでの HTTP レスポンスの生成](#)」を参照してください。

HTTP レスポンスを使用する場合、Lambda@Edge は、オリジンサーバーから返された本文を origin-response トリガーに公開しません。必要な値に設定することで静的なコンテンツ本文を生成したり、値を空に設定することで関数内の本文を削除したりできます。関数内の本文フィールドを更新しない場合は、オリジンサーバーによって返された元の本文がビューワーに返されます。

Include Body オプションを選択するリクエストボディへのアクセス

書き込み可能な HTTP メソッド (POST、PUT、DELETE など) のリクエストのボディを Lambda@Edge で公開することを選択できるため、Lambda 関数でそのボディにアクセスできます。読み取り専用アクセスを選択することも、ボディを置き換えることを指定することもできます。

このオプションを有効にするには、ビューワーリクエストまたはオリジンリクエストイベントの関数の CloudFront トリガーを作成するときに、Include Body を選択します。詳細については、「[Lambda@Edge 関数のトリガーの追加](#)」を参照してください。または、関数で [ボディを含める] を使用する方法については、「[Lambda@Edge イベント構造](#)」を参照してください。

この機能を使用する場合のシナリオには次のようなものがあります。

- お客様の入力データをオリジンサーバーに返送することなく、「お問い合わせ」フォームのようなウェブフォームを処理します。
- ビューワーブラウザによって送信されるウェブビーコンデータを収集し、エッジで処理します。

サンプルコードについては、「[Lambda@Edge 関数の例](#)」を参照してください。

Note

リクエストボディが大きい場合は、Lambda@Edge によって切り捨てられます。最大サイズ制限と切り捨ての詳細については、「[Include Body オプションがあるリクエストボディに対する制限](#)」を参照してください。

Lambda@Edge 関数の例

で Lambda 関数を使用する例については、以下のセクションを参照してください CloudFront。

トピック

- [一般的な例](#)
- [レスポンスの生成 - 例](#)
- [クエリ文字列の操作 - 例](#)
- [国またはデバイスタイプヘッダー別のコンテンツのパーソナライズ - 例](#)
- [コンテンツベースの動的オリジンの選択 - 例](#)
- [エラーステータスの更新 - 例](#)
- [リクエストボディへのアクセス - 例](#)

一般的な例

このセクションの例では、CloudFront で Lambda@Edge を使用する一般的な方法をいくつか示しています。

トピック

- [例: A/B テスト](#)
- [例: レスポンスヘッダーのオーバーライド](#)

例: A/B テスト

次の例を使用すると、リダイレクトを作成したり URL を変更したりすることなく、2 つの異なるバージョンのイメージをテストできます。この例では、ビューワーリクエスト内の Cookie を読み取り、それに応じてリクエスト URL を変更します。ビューワーがいずれかの期待値を使用して Cookie を送信しない場合、例ではビューワーを URL のいずれかにランダムに割り当てます。

Node.js

```
'use strict';

exports.handler = (event, context, callback) => {
  const request = event.Records[0].cf.request;
  const headers = request.headers;
```

```
if (request.uri !== '/experiment-pixel.jpg') {
  // do not process if this is not an A-B test request
  callback(null, request);
  return;
}

const cookieExperimentA = 'X-Experiment-Name=A';
const cookieExperimentB = 'X-Experiment-Name=B';
const pathExperimentA = '/experiment-group/control-pixel.jpg';
const pathExperimentB = '/experiment-group/treatment-pixel.jpg';

/*
 * Lambda at the Edge headers are array objects.
 *
 * Client may send multiple Cookie headers, i.e.:
 * > GET /viewerRes/test HTTP/1.1
 * > User-Agent: curl/7.18.1 (x86_64-unknown-linux-gnu) libcurl/7.18.1
OpenSSL/1.0.1u zlib/1.2.3
 * > Cookie: First=1; Second=2
 * > Cookie: ClientCode=abc
 * > Host: example.com
 *
 * You can access the first Cookie header at headers["cookie"][0].value
 * and the second at headers["cookie"][1].value.
 *
 * Header values are not parsed. In the example above,
 * headers["cookie"][0].value is equal to "First=1; Second=2"
 */
let experimentUri;
if (headers.cookie) {
  for (let i = 0; i < headers.cookie.length; i++) {
    if (headers.cookie[i].value.indexOf(cookieExperimentA) >= 0) {
      console.log('Experiment A cookie found');
      experimentUri = pathExperimentA;
      break;
    } else if (headers.cookie[i].value.indexOf(cookieExperimentB) >= 0) {
      console.log('Experiment B cookie found');
      experimentUri = pathExperimentB;
      break;
    }
  }
}

if (!experimentUri) {
```

```
    console.log('Experiment cookie has not been found. Throwing dice...');
    if (Math.random() < 0.75) {
        experimentUri = pathExperimentA;
    } else {
        experimentUri = pathExperimentB;
    }
}

request.uri = experimentUri;
console.log(`Request uri set to "${request.uri}"`);
callback(null, request);
};
```

Python

```
import json
import random

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
    headers = request['headers']

    if request['uri'] != '/experiment-pixel.jpg':
        # Not an A/B Test
        return request

    cookieExperimentA, cookieExperimentB = 'X-Experiment-Name=A', 'X-Experiment-Name=B'
    pathExperimentA, pathExperimentB = '/experiment-group/control-pixel.jpg', '/experiment-group/treatment-pixel.jpg'

    ...

    Lambda at the Edge headers are array objects.

    Client may send multiple cookie headers. For example:
    > GET /viewerRes/test HTTP/1.1
    > User-Agent: curl/7.18.1 (x86_64-unknown-linux-gnu) libcurl/7.18.1
    OpenSSL/1.0.1u zlib/1.2.3
    > Cookie: First=1; Second=2
    > Cookie: ClientCode=abc
    > Host: example.com

    You can access the first Cookie header at headers["cookie"][0].value
```

```
and the second at headers["cookie"][1].value.

Header values are not parsed. In the example above,
headers["cookie"][0].value is equal to "First=1; Second=2"
'''

experimentUri = ""

for cookie in headers.get('cookie', []):
    if cookieExperimentA in cookie['value']:
        print("Experiment A cookie found")
        experimentUri = pathExperimentA
        break
    elif cookieExperimentB in cookie['value']:
        print("Experiment B cookie found")
        experimentUri = pathExperimentB
        break

if not experimentUri:
    print("Experiment cookie has not been found. Throwing dice...")
    if random.random() < 0.75:
        experimentUri = pathExperimentA
    else:
        experimentUri = pathExperimentB

request['uri'] = experimentUri
print(f"Request uri set to {experimentUri}")
return request
```

例: レスポンスヘッダーのオーバーライド

以下の例は、レスポンスヘッダーの値を別のヘッダーの値に基づいて変更する方法を示しています。

Node.js

```
'use strict';

exports.handler = (event, context, callback) => {
    const response = event.Records[0].cf.response;
    const headers = response.headers;

    const headerNameSrc = 'X-Amz-Meta-Last-Modified';
    const headerNameDst = 'Last-Modified';
```

```
if (headers[headerNameSrc.toLowerCase()]) {
  headers[headerNameDst.toLowerCase()] = [
    headers[headerNameSrc.toLowerCase()][0],
  ];
  console.log(`Response header "${headerNameDst}" was set to ` +
    `${headers[headerNameDst.toLowerCase()][0].value}`);
}

callback(null, response);
};
```

Python

```
import json

def lambda_handler(event, context):
    response = event["Records"][0]["cf"]["response"]
    headers = response["headers"]

    headerNameSrc = "X-Amz-Meta-Last-Modified"
    headerNameDst = "Last-Modified"

    if headers.get(headerNameSrc.lower(), None):
        headers[headerNameDst.lower()] = [headers[headerNameSrc.lower()][0]]
        print(f"Response header {headerNameDst.lower()} was set to
{headers[headerNameSrc.lower()][0]}")

    return response
```

レスポンスの生成 - 例

このセクションの例では、Lambda@Edge を使用してレスポンスを生成する方法を示しています。

トピック

- [例: 静的コンテンツの提供 \(生成されたレスポンス\)](#)
- [例: HTTP リダイレクトの生成 \(生成されたレスポンス\)](#)

例: 静的コンテンツの提供 (生成されたレスポンス)

次の例は、Lambda 関数を使用して静的ウェブサイトコンテンツを提供する方法を示しています。これにより、オリジンサーバーの負荷と全体的なレイテンシーが軽減されます。

Note

HTTP レスポンスは、ビューワーリクエストおよびオリジンリクエストのイベントに対して生成できます。詳細については、「[the section called “リクエストトリガーでの HTTP レスポンスの生成”](#)」を参照してください。

オリジンレスポンスイベントで HTTP レスポンスのボディを置き換えたり、削除することもできます。詳細については、「[the section called “オリジンレスポンストリガーでの HTTP レスポンスの更新”](#)」を参照してください。

Node.js

```
'use strict';

const content = `
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Simple Lambda@Edge Static Content Response</title>
  </head>
  <body>
    <p>Hello from Lambda@Edge!</p>
  </body>
</html>
`;

exports.handler = (event, context, callback) => {
  /*
   * Generate HTTP OK response using 200 status code with HTML body.
   */
  const response = {
    status: '200',
    statusDescription: 'OK',
    headers: {
      'cache-control': [{
        key: 'Cache-Control',
```

```
        value: 'max-age=100'
    }],
    'content-type': [{
        key: 'Content-Type',
        value: 'text/html'
    }]
},
body: content,
};
callback(null, response);
};
```

Python

```
import json

CONTENT = """
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Simple Lambda@Edge Static Content Response</title>
</head>
<body>
    <p>Hello from Lambda@Edge!</p>
</body>
</html>
"""

def lambda_handler(event, context):
    # Generate HTTP OK response using 200 status code with HTML body.
    response = {
        'status': '200',
        'statusDescription': 'OK',
        'headers': {
            'cache-control': [
                {
                    'key': 'Cache-Control',
                    'value': 'max-age=100'
                }
            ],
            "content-type": [
                {
```

```
        'key': 'Content-Type',
        'value': 'text/html'
      }
    ]
  },
  'body': CONTENT
}
return response
```

例: HTTP リダイレクトの生成 (生成されたレスポンス)

次の例は、HTTP リダイレクトを生成する方法を示しています。

Note

HTTP レスポンスは、ビューワーリクエストおよびオリジンリクエストのイベントに対して生成できます。詳細については、「[リクエストトリガーでの HTTP レスポンスの生成](#)」を参照してください。

Node.js

```
'use strict';

exports.handler = (event, context, callback) => {
  /*
   * Generate HTTP redirect response with 302 status code and Location header.
   */
  const response = {
    status: '302',
    statusDescription: 'Found',
    headers: {
      location: [{
        key: 'Location',
        value: 'https://docs.aws.amazon.com/lambda/latest/dg/lambda-
edge.html',
      }],
    },
  };
  callback(null, response);
};
```

Python

```
def lambda_handler(event, context):

    # Generate HTTP redirect response with 302 status code and Location header.

    response = {
        'status': '302',
        'statusDescription': 'Found',
        'headers': {
            'location': [{
                'key': 'Location',
                'value': 'https://docs.aws.amazon.com/lambda/latest/dg/lambda-
edge.html'
            }]
        }
    }

    return response
```

クエリ文字列の操作 - 例

このセクションの例には、クエリ文字列で Lambda@Edge を使用する方法が含まれています。

トピック

- [例: クエリ文字列パラメータに基づくヘッダーを追加する](#)
- [例: キャッシュヒット率を向上させるためのクエリ文字列パラメータの標準化](#)
- [例: 認証されていないユーザーをサインインページにリダイレクトする](#)

例: クエリ文字列パラメータに基づくヘッダーを追加する

以下の例では、クエリ文字列パラメータのキーと値のペアを取得してから、それらの値に基づいてヘッダーを追加する方法を示します。

Node.js

```
'use strict';

const querystring = require('querystring');
exports.handler = (event, context, callback) => {
```

```
const request = event.Records[0].cf.request;

/* When a request contains a query string key-value pair but the origin server
 * expects the value in a header, you can use this Lambda function to
 * convert the key-value pair to a header. Here's what the function does:
 * 1. Parses the query string and gets the key-value pair.
 * 2. Adds a header to the request using the key-value pair that the function
got in step 1.
 */

/* Parse request querystring to get javascript object */
const params = querystring.parse(request.querystring);

/* Move auth param from querystring to headers */
const headerName = 'Auth-Header';
request.headers[headerName.toLowerCase()] = [{ key: headerName, value:
params.auth }];
delete params.auth;

/* Update request querystring */
request.querystring = querystring.stringify(params);

callback(null, request);
};
```

Python

```
from urllib.parse import parse_qs, urlencode

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']

    ...

    When a request contains a query string key-value pair but the origin server
    expects the value in a header, you can use this Lambda function to
    convert the key-value pair to a header. Here's what the function does:
        1. Parses the query string and gets the key-value pair.
        2. Adds a header to the request using the key-value pair that the function
got in step 1.
    ...

    # Parse request querystring to get dictionary/json
    params = {k : v[0] for k, v in parse_qs(request['querystring']).items()}
```

```
# Move auth param from querystring to headers
headerName = 'Auth-Header'
request['headers'][headerName.lower()] = [{'key': headerName, 'value':
params['auth']}]
del params['auth']

# Update request querystring
request['querystring'] = urlencode(params)

return request
```

例: キャッシュヒット率を向上させるためのクエリ文字列パラメータの標準化

次の例は、ガリクエストをオリジン CloudFront に転送する前にクエリ文字列に次の変更を加えることで、キャッシュヒット率を向上させる方法を示しています。

- パラメータの名前によりキーと値のペアをアルファベット順に並べ替える
- キーと値のペアを小文字に変更する

詳細については、「[クエリ文字列パラメータに基づくコンテンツのキャッシュ](#)」を参照してください。

Node.js

```
'use strict';

const querystring = require('querystring');

exports.handler = (event, context, callback) => {
  const request = event.Records[0].cf.request;
  /* When you configure a distribution to forward query strings to the origin and
  * to cache based on an allowlist of query string parameters, we recommend
  * the following to improve the cache-hit ratio:
  * - Always list parameters in the same order.
  * - Use the same case for parameter names and values.
  *
  * This function normalizes query strings so that parameter names and values
  * are lowercase and parameter names are in alphabetical order.
  *
  * For more information, see:
```

```
    * https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/QueryStringParameters.html
    */

    console.log('Query String: ', request.querystring);

    /* Parse request query string to get javascript object */
    const params = querystring.parse(request.querystring.toLowerCase());
    const sortedParams = {};

    /* Sort param keys */
    Object.keys(params).sort().forEach(key => {
        sortedParams[key] = params[key];
    });

    /* Update request querystring with normalized */
    request.querystring = querystring.stringify(sortedParams);

    callback(null, request);
};
```

Python

```
from urllib.parse import parse_qs, urlencode

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
    ...

    When you configure a distribution to forward query strings to the origin and
    to cache based on an allowlist of query string parameters, we recommend
    the following to improve the cache-hit ratio:
    Always list parameters in the same order.
    - Use the same case for parameter names and values.

    This function normalizes query strings so that parameter names and values
    are lowercase and parameter names are in alphabetical order.

    For more information, see:
    https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/QueryStringParameters.html
    ...

    print("Query string: ", request["querystring"])
```

```
# Parse request query string to get js object
params = {k : v[0] for k, v in parse_qs(request['querystring'].lower()).items()}

# Sort param keys
sortedParams = sorted(params.items(), key=lambda x: x[0])

# Update request querystring with normalized
request['querystring'] = urlencode(sortedParams)

return request
```

例: 認証されていないユーザーをサインインページにリダイレクトする

次の例では、ユーザーが認証情報を入力していない場合にサインインページにリダイレクトする方法を示します。

Node.js

```
'use strict';

function parseCookies(headers) {
  const parsedCookie = {};
  if (headers.cookie) {
    headers.cookie[0].value.split(';').forEach((cookie) => {
      if (cookie) {
        const parts = cookie.split('=');
        parsedCookie[parts[0].trim()] = parts[1].trim();
      }
    });
  }
  return parsedCookie;
}

exports.handler = (event, context, callback) => {
  const request = event.Records[0].cf.request;
  const headers = request.headers;

  /* Check for session-id in request cookie in viewer-request event,
   * if session-id is absent, redirect the user to sign in page with original
   * request sent as redirect_url in query params.
   */
```

```
/* Check for session-id in cookie, if present then proceed with request */
const parsedCookies = parseCookies(headers);
if (parsedCookies && parsedCookies['session-id']) {
    callback(null, request);
    return;
}

/* URI encode the original request to be sent as redirect_url in query params */
const encodedRedirectUrl = encodeURIComponent(`https://${headers.host[0].value}${request.uri}?${request.querystring}`);
const response = {
    status: '302',
    statusDescription: 'Found',
    headers: {
        location: [{
            key: 'Location',
            value: `https://www.example.com/signin?redirect_url=${encodedRedirectUrl}`,
        }],
    },
};
callback(null, response);
};
```

Python

```
import urllib

def parseCookies(headers):
    parsedCookie = {}
    if headers.get('cookie'):
        for cookie in headers['cookie'][0]['value'].split(';'):
            if cookie:
                parts = cookie.split('=')
                parsedCookie[parts[0].strip()] = parts[1].strip()
    return parsedCookie

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
    headers = request['headers']

    ...

    Check for session-id in request cookie in viewer-request event,
```

```
if session-id is absent, redirect the user to sign in page with original
request sent as redirect_url in query params.
...

# Check for session-id in cookie, if present, then proceed with request
parsedCookies = parseCookies(headers)

if parsedCookies and parsedCookies['session-id']:
    return request

# URI encode the original request to be sent as redirect_url in query params
redirectUrl = "https://%s%s?%s" % (headers['host'][0]['value'], request['uri'],
request['querystring'])
encodedRedirectUrl = urllib.parse.quote_plus(redirectUrl.encode('utf-8'))

response = {
    'status': '302',
    'statusDescription': 'Found',
    'headers': {
        'location': [{
            'key': 'Location',
            'value': 'https://www.example.com/signin?redirect_url=%s' %
encodedRedirectUrl
        }]
    }
}
return response
```

国またはデバイスタイプヘッダー別のコンテンツのパーソナライズ - 例

このセクションの例では、Lambda@Edge を使用し、ビューワーが使用しているデバイスの場所またはタイプに基づいて動作をカスタマイズする方法を示しています。

トピック

- [例: ビューワーリクエストを国に固有の URL にリダイレクトする](#)
- [例: デバイスに基づいて異なるバージョンのオブジェクトを供給する](#)

例: ビューワーリクエストを国に固有の URL にリダイレクトする

次の例では、HTTP リダイレクト応答を国に固有の URL で生成し、ビューワーにレスポンスを返す方法を示します。これは、国ごとに異なる応答を提供する場合に便利です。次に例を示します。

- 国別のサブドメイン (us.example.com および tw.example.com など) がある場合は、ビューワーが example.com をリクエストしたときにリダイレクト応答を生成できます。
- 動画をストリーミングしていて、そのコンテンツを特定の国でストリーミングする権限がない場合は、その国のユーザーを別のページにリダイレクトして動画を閲覧できない理由について説明できます。

次の点に注意してください。

- CloudFront-Viewer-Country ヘッダーに基づいてキャッシュするようにディストリビューションを設定する必要があります。詳細については、「[選択されたリクエストヘッダーに基づいたキャッシュ](#)」を参照してください。
- CloudFront は、ビューワーリクエストイベントの後に CloudFront-Viewer-Country ヘッダーを追加します。この例を使用するには、オリジンリクエストイベントのトリガーを作成する必要があります。

Node.js

```
'use strict';

/* This is an origin request function */
exports.handler = (event, context, callback) => {
  const request = event.Records[0].cf.request;
  const headers = request.headers;

  /*
   * Based on the value of the CloudFront-Viewer-Country header, generate an
   * HTTP status code 302 (Redirect) response, and return a country-specific
   * URL in the Location header.
   * NOTE: 1. You must configure your distribution to cache based on the
   *         CloudFront-Viewer-Country header. For more information, see
   *         https://docs.aws.amazon.com/console/cloudfront/cache-on-selected-
headers
   *         2. CloudFront adds the CloudFront-Viewer-Country header after the
viewer
   *         request event. To use this example, you must create a trigger for
the
   *         origin request event.
   */

  let url = 'https://example.com/';
```

```
if (headers['cloudfront-viewer-country']) {
  const countryCode = headers['cloudfront-viewer-country'][0].value;
  if (countryCode === 'TW') {
    url = 'https://tw.example.com/';
  } else if (countryCode === 'US') {
    url = 'https://us.example.com/';
  }
}

const response = {
  status: '302',
  statusDescription: 'Found',
  headers: {
    location: [{
      key: 'Location',
      value: url,
    }],
  },
};
callback(null, response);
};
```

Python

```
# This is an origin request function

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
    headers = request['headers']

    ...

    Based on the value of the CloudFront-Viewer-Country header, generate an
    HTTP status code 302 (Redirect) response, and return a country-specific
    URL in the Location header.
    NOTE: 1. You must configure your distribution to cache based on the
           CloudFront-Viewer-Country header. For more information, see
           https://docs.aws.amazon.com/console/cloudfront/cache-on-selected-headers
           2. CloudFront adds the CloudFront-Viewer-Country header after the viewer
           request event. To use this example, you must create a trigger for the
           origin request event.

    ...

    url = 'https://example.com/'
```

```
viewerCountry = headers.get('cloudfront-viewer-country')
if viewerCountry:
    countryCode = viewerCountry[0]['value']
    if countryCode == 'TW':
        url = 'https://tw.example.com/'
    elif countryCode == 'US':
        url = 'https://us.example.com/'

response = {
    'status': '302',
    'statusDescription': 'Found',
    'headers': {
        'location': [{
            'key': 'Location',
            'value': url
        }]
    }
}

return response
```

例: デバイスに基づいて異なるバージョンのオブジェクトを供給する

次の例では、ユーザーが使用しているモバイルデバイスまたはタブレットのようなデバイスのタイプに基づいてオブジェクトの異なるバージョンを提供する方法を示します。次の点に注意してください。

- CloudFront-Is-*-Viewer ヘッダーに基づいてキャッシュするようにディストリビューションを設定する必要があります。詳細については、「[選択されたリクエストヘッダーに基づいたキャッシュ](#)」を参照してください。
- CloudFront は、ビューワーリクエストイベントの後に CloudFront-Is-*-Viewer ヘッダーを追加します。この例を使用するには、オリジンリクエストイベントのトリガーを作成する必要があります。

Node.js

```
'use strict';

/* This is an origin request function */
exports.handler = (event, context, callback) => {
```

```
const request = event.Records[0].cf.request;
const headers = request.headers;

/*
 * Serve different versions of an object based on the device type.
 * NOTE: 1. You must configure your distribution to cache based on the
 *        CloudFront-Is-*-Viewer headers. For more information, see
 *        the following documentation:
 *        https://docs.aws.amazon.com/console/cloudfront/cache-on-selected-
headers
 *        https://docs.aws.amazon.com/console/cloudfront/cache-on-device-type
 *        2. CloudFront adds the CloudFront-Is-*-Viewer headers after the viewer
 *        request event. To use this example, you must create a trigger for
the
 *        origin request event.
 */

const desktopPath = '/desktop';
const mobilePath = '/mobile';
const tabletPath = '/tablet';
const smarttvPath = '/smarttv';

if (headers['cloudfront-is-desktop-viewer']
    && headers['cloudfront-is-desktop-viewer'][0].value === 'true') {
    request.uri = desktopPath + request.uri;
} else if (headers['cloudfront-is-mobile-viewer']
    && headers['cloudfront-is-mobile-viewer'][0].value === 'true') {
    request.uri = mobilePath + request.uri;
} else if (headers['cloudfront-is-tablet-viewer']
    && headers['cloudfront-is-tablet-viewer'][0].value === 'true') {
    request.uri = tabletPath + request.uri;
} else if (headers['cloudfront-is-smarttv-viewer']
    && headers['cloudfront-is-smarttv-viewer'][0].value === 'true') {
    request.uri = smarttvPath + request.uri;
}
console.log(`Request uri set to "${request.uri}"`);

callback(null, request);
};
```

Python

```
# This is an origin request function
```

```
def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
    headers = request['headers']

    ...

    Serve different versions of an object based on the device type.
    NOTE: 1. You must configure your distribution to cache based on the
           CloudFront-Is-*-Viewer headers. For more information, see
           the following documentation:
           https://docs.aws.amazon.com/console/cloudfront/cache-on-selected-headers
           https://docs.aws.amazon.com/console/cloudfront/cache-on-device-type
           2. CloudFront adds the CloudFront-Is-*-Viewer headers after the viewer
           request event. To use this example, you must create a trigger for the
           origin request event.

    ...

    desktopPath = '/desktop';
    mobilePath = '/mobile';
    tabletPath = '/tablet';
    smarttvPath = '/smarttv';

    if 'cloudfront-is-desktop-viewer' in headers and headers['cloudfront-is-desktop-viewer'][0]['value'] == 'true':
        request['uri'] = desktopPath + request['uri']
    elif 'cloudfront-is-mobile-viewer' in headers and headers['cloudfront-is-mobile-viewer'][0]['value'] == 'true':
        request['uri'] = mobilePath + request['uri']
    elif 'cloudfront-is-tablet-viewer' in headers and headers['cloudfront-is-tablet-viewer'][0]['value'] == 'true':
        request['uri'] = tabletPath + request['uri']
    elif 'cloudfront-is-smarttv-viewer' in headers and headers['cloudfront-is-smarttv-viewer'][0]['value'] == 'true':
        request['uri'] = smarttvPath + request['uri']

    print("Request uri set to %s" % request['uri'])

    return request
```

コンテンツベースの動的オリジンの選択 - 例

このセクションの例では、Lambda@Edge を使用し、リクエスト内の情報に基づいて異なるオリジンにルーティングする方法を示しています。

トピック

- [例: オリジンリクエストトリガーを使用してカスタムオリジンを Amazon S3 オリジンに変更する](#)
- [例: オリジンリクエストトリガーを使用して Amazon S3 オリジンのリージョンを変更する](#)
- [例: オリジンリクエストトリガーを使用して Amazon S3 オリジンからカスタムオリジンに変更する](#)
- [例: オリジンリクエストトリガーを使用して Amazon S3 バケットから別のバケットにトラフィックを徐々に転送する](#)
- [例: オリジンリクエストトリガーを使用して Country ヘッダーに基づいてオリジンのドメイン名を変更する](#)

例: オリジンリクエストトリガーを使用してカスタムオリジンを Amazon S3 オリジンに変更する

この関数では、origin-request トリガーを使用して、リクエストのプロパティに基づいて、カスタムオリジンから、コンテンツがフェッチされる Amazon S3 オリジンに変更する方法を示しています。

Node.js

```
'use strict';

const querystring = require('querystring');

exports.handler = (event, context, callback) => {
  const request = event.Records[0].cf.request;

  /**
   * Reads query string to check if S3 origin should be used, and
   * if true, sets S3 origin properties.
   */

  const params = querystring.parse(request.querystring);

  if (params['useS3Origin']) {
    if (params['useS3Origin'] === 'true') {
      const s3DomainName = 'my-bucket.s3.amazonaws.com';

      /* Set S3 origin fields */
      request.origin = {
        s3: {
          domainName: s3DomainName,
          region: '',
```

```
        authMethod: 'none',
        path: '',
        customHeaders: {}
    }
};
request.headers['host'] = [{ key: 'host', value: s3DomainName}];
}
}

callback(null, request);
};
```

Python

```
from urllib.parse import parse_qs

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
    ...

    Reads query string to check if S3 origin should be used, and
    if true, sets S3 origin properties
    ...

    params = {k: v[0] for k, v in parse_qs(request['queryString']).items()}
    if params.get('useS3Origin') == 'true':
        s3DomainName = 'my-bucket.s3.amazonaws.com'

    # Set S3 origin fields
    request['origin'] = {
        's3': {
            'domainName': s3DomainName,
            'region': '',
            'authMethod': 'none',
            'path': '',
            'customHeaders': {}
        }
    }
    request['headers']['host'] = [{'key': 'host', 'value': s3DomainName}]
    return request
```

例: オリジンリクエストトリガーを使用して Amazon S3 オリジンのリージョンを変更する

この関数では、origin-request トリガーを使用して、リクエストのプロパティに基づいて、コンテンツがフェッチされる Amazon S3 オリジンを変更する方法を示しています。

この例では、CloudFront-Viewer-Country ヘッダーの値を使用して、S3 バケットのドメイン名を、ビューワーに近いリージョンのバケットに更新します。これは、以下のように役立ちます。

- 指定したリージョンがビューワーの国に近いほど、レイテンシーが短縮されます。
- リクエスト元と同じ国にあるオリジンからデータが提供されることになり、データ主権が確保されます。

この例を使用するには、以下を実行する必要があります。

- CloudFront-Viewer-Country ヘッダーに基づいてキャッシュするようにディストリビューションを設定します。詳細については、「[選択されたリクエストヘッダーに基づいたキャッシュ](#)」を参照してください。
- オリジンリクエストイベントでこの関数のトリガーを作成します。CloudFront はビューワーリクエストイベントの後に CloudFront-Viewer-Country ヘッダーを追加するため、この例を使用するには、関数がオリジンリクエストに対して実行されることを確認する必要があります。

Node.js

```
'use strict';

exports.handler = (event, context, callback) => {
  const request = event.Records[0].cf.request;

  /**
   * This blueprint demonstrates how an origin-request trigger can be used to
   * change the origin from which the content is fetched, based on request
   * properties.
   * In this example, we use the value of the CloudFront-Viewer-Country header
   * to update the S3 bucket domain name to a bucket in a Region that is closer to
   * the viewer.
   *
   * This can be useful in several ways:
   *   1) Reduces latencies when the Region specified is nearer to the viewer's
   *       country.
   *   2) Provides data sovereignty by making sure that data is served from an
```

```

*      origin that's in the same country that the request came from.
*
* NOTE: 1. You must configure your distribution to cache based on the
*         CloudFront-Viewer-Country header. For more information, see
*         https://docs.aws.amazon.com/console/cloudfront/cache-on-selected-
headers
*         2. CloudFront adds the CloudFront-Viewer-Country header after the
viewer
*         request event. To use this example, you must create a trigger for
the
*         origin request event.
*/

const countryToRegion = {
  'DE': 'eu-central-1',
  'IE': 'eu-west-1',
  'GB': 'eu-west-2',
  'FR': 'eu-west-3',
  'JP': 'ap-northeast-1',
  'IN': 'ap-south-1'
};

if (request.headers['cloudfront-viewer-country']) {
  const countryCode = request.headers['cloudfront-viewer-country'][0].value;
  const region = countryToRegion[countryCode];

  /**
   * If the viewer's country is not in the list you specify, the request
   * goes to the default S3 bucket you've configured.
   */
  if (region) {
    /**
     * If you've set up OAI, the bucket policy in the destination bucket
     * should allow the OAI GetObject operation, as configured by default
     * for an S3 origin with OAI. Another requirement with OAI is to provide
     * the Region so it can be used for the SIGV4 signature. Otherwise, the
     * Region is not required.
     */
    request.origin.s3.region = region;
    const domainName = `my-bucket-in-${region}.s3.amazonaws.com`;
    request.origin.s3.domainName = domainName;
    request.headers['host'] = [{ key: 'host', value: domainName }];
  }
}

```

```
    callback(null, request);
};
```

Python

```
def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']

    ...

    This blueprint demonstrates how an origin-request trigger can be used to
    change the origin from which the content is fetched, based on request
    properties.
    In this example, we use the value of the CloudFront-Viewer-Country header
    to update the S3 bucket domain name to a bucket in a Region that is closer to
    the viewer.

    This can be useful in several ways:
        1) Reduces latencies when the Region specified is nearer to the viewer's
           country.
        2) Provides data sovereignty by making sure that data is served from an
           origin that's in the same country that the request came from.

    NOTE: 1. You must configure your distribution to cache based on the
           CloudFront-Viewer-Country header. For more information, see
           https://docs.aws.amazon.com/console/cloudfront/cache-on-selected-headers
        2. CloudFront adds the CloudFront-Viewer-Country header after the viewer
           request event. To use this example, you must create a trigger for the
           origin request event.

    ...

    countryToRegion = {
        'DE': 'eu-central-1',
        'IE': 'eu-west-1',
        'GB': 'eu-west-2',
        'FR': 'eu-west-3',
        'JP': 'ap-northeast-1',
        'IN': 'ap-south-1'
    }

    viewerCountry = request['headers'].get('cloudfront-viewer-country')
    if viewerCountry:
        countryCode = viewerCountry[0]['value']
```

```
region = countryToRegion.get(countryCode)

# If the viewer's country is not in the list you specify, the request
# goes to the default S3 bucket you've configured
if region:
    '''
    If you've set up OAI, the bucket policy in the destination bucket
    should allow the OAI GetObject operation, as configured by default
    for an S3 origin with OAI. Another requirement with OAI is to provide
    the Region so it can be used for the SIGV4 signature. Otherwise, the
    Region is not required.
    '''
    request['origin']['s3']['region'] = region
    domainName = 'my-bucket-in-%s.s3.amazonaws.com' % region
    request['origin']['s3']['domainName'] = domainName
    request['headers']['host'] = [{'key': 'host', 'value': domainName}]

return request
```

例: オリジンリクエストトリガーを使用して Amazon S3 オリジンからカスタムオリジンに変更する

この関数では、origin-request トリガーを使用して、リクエストのプロパティに基づいて、コンテンツがフェッチされるカスタムオリジンを変更する方法を示しています。

Node.js

```
'use strict';

const querystring = require('querystring');

exports.handler = (event, context, callback) => {
    const request = event.Records[0].cf.request;

    /**
     * Reads query string to check if custom origin should be used, and
     * if true, sets custom origin properties.
     */

    const params = querystring.parse(request.querystring);

    if (params['useCustomOrigin']) {
        if (params['useCustomOrigin'] === 'true') {
```

```
/* Set custom origin fields*/
request.origin = {
  custom: {
    domainName: 'www.example.com',
    port: 443,
    protocol: 'https',
    path: '',
    sslProtocols: ['TLSv1', 'TLSv1.1'],
    readTimeout: 5,
    keepaliveTimeout: 5,
    customHeaders: {}
  }
};
request.headers['host'] = [{ key: 'host', value: 'www.example.com'}];
}
}
callback(null, request);
};
```

Python

```
from urllib.parse import parse_qs

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']

    # Reads query string to check if custom origin should be used, and
    # if true, sets custom origin properties

    params = {k: v[0] for k, v in parse_qs(request['queryString']).items()}

    if params.get('useCustomOrigin') == 'true':
        # Set custom origin fields
        request['origin'] = {
            'custom': {
                'domainName': 'www.example.com',
                'port': 443,
                'protocol': 'https',
                'path': '',
                'sslProtocols': ['TLSv1', 'TLSv1.1'],
                'readTimeout': 5,
                'keepaliveTimeout': 5,
```

```
        'customHeaders': {}
      }
    }
    request['headers']['host'] = [{ 'key': 'host', 'value':
'www.example.com' }]

    return request
  }
}
```

例: オリジンリクエストトリガーを使用して Amazon S3 バケットから別のバケットにトラフィックを徐々に転送する

この関数では、Amazon S3 バケットから別のバケットにトラフィックを制御しながら徐々に転送する方法を示しています。

Node.js

```
'use strict';

function getRandomInt(min, max) {
  /* Random number is inclusive of min and max*/
  return Math.floor(Math.random() * (max - min + 1)) + min;
}

exports.handler = (event, context, callback) => {
  const request = event.Records[0].cf.request;
  const BLUE_TRAFFIC_PERCENTAGE = 80;

  /**
   * This Lambda function demonstrates how to gradually transfer traffic from
   * one S3 bucket to another in a controlled way.
   * We define a variable BLUE_TRAFFIC_PERCENTAGE which can take values from
   * 1 to 100. If the generated randomNumber less than or equal to
  BLUE_TRAFFIC_PERCENTAGE, traffic
   * is re-directed to blue-bucket. If not, the default bucket that we've
  configured
   * is used.
   */

  const randomNumber = getRandomInt(1, 100);

  if (randomNumber <= BLUE_TRAFFIC_PERCENTAGE) {
    const domainName = 'blue-bucket.s3.amazonaws.com';
```

```
        request.origin.s3.domainName = domainName;
        request.headers['host'] = [{ key: 'host', value: domainName}];
    }
    callback(null, request);
};
```

Python

```
import math
import random

def getRandomInt(min, max):
    # Random number is inclusive of min and max
    return math.floor(random.random() * (max - min + 1)) + min

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
    BLUE_TRAFFIC_PERCENTAGE = 80

    ...

    This Lambda function demonstrates how to gradually transfer traffic from
    one S3 bucket to another in a controlled way.
    We define a variable BLUE_TRAFFIC_PERCENTAGE which can take values from
    1 to 100. If the generated randomNumber less than or equal to
    BLUE_TRAFFIC_PERCENTAGE, traffic
    is re-directed to blue-bucket. If not, the default bucket that we've configured
    is used.
    ...

    randomNumber = getRandomInt(1, 100)

    if randomNumber <= BLUE_TRAFFIC_PERCENTAGE:
        domainName = 'blue-bucket.s3.amazonaws.com'
        request['origin']['s3']['domainName'] = domainName
        request['headers']['host'] = [{'key': 'host', 'value': domainName}]

    return request
```

例: オリジンリクエストトリガーを使用して Country ヘッダーに基づいてオリジンのドメイン名を変更する

この関数では、CloudFront-Viewer-Country ヘッダーに基づいてオリジンのドメイン名を変更する方法を示しています。これにより、コンテンツはビューワーの国に近いオリジンから配信されます。

ディストリビューションに対してこの機能を実装すると、次のような利点があります。

- 指定したリージョンがビューワーの国に近いほど、レイテンシーが短縮されます。
- リクエスト元と同じ国にあるオリジンからデータが提供されることになり、データ主権が確保されます。

この機能を有効にするには、CloudFront-Viewer-Country ヘッダーに基づいてキャッシュするようにディストリビューションを設定する必要があります。詳細については、「[the section called “選択されたリクエストヘッダーに基づいたキャッシュ”](#)」を参照してください。

Node.js

```
'use strict';

exports.handler = (event, context, callback) => {
  const request = event.Records[0].cf.request;

  if (request.headers['cloudfront-viewer-country']) {
    const countryCode = request.headers['cloudfront-viewer-country'][0].value;
    if (countryCode === 'GB' || countryCode === 'DE' || countryCode === 'IE' )
    {
      const domainName = 'eu.example.com';
      request.origin.custom.domainName = domainName;
      request.headers['host'] = [{key: 'host', value: domainName}];
    }
  }

  callback(null, request);
};
```

Python

```
def lambda_handler(event, context):
```

```
request = event['Records'][0]['cf']['request']

viewerCountry = request['headers'].get('cloudfront-viewer-country')
if viewerCountry:
    countryCode = viewerCountry[0]['value']
    if countryCode == 'GB' or countryCode == 'DE' or countryCode == 'IE':
        domainName = 'eu.example.com'
        request['origin']['custom']['domainName'] = domainName
        request['headers']['host'] = [{'key': 'host', 'value': domainName}]
return request
```

エラーステータスの更新 - 例

このセクションの例では、Lambda@Edge を使用して、ユーザーに返されるエラーステータスを変更する方法を示しています。

トピック

- [例: オリジンレスポンストリガーを使用してエラーステータスコードを 200 に更新する](#)
- [例: オリジンレスポンストリガーを使用してエラーステータスコードを 302 に更新する](#)

例: オリジンレスポンストリガーを使用してエラーステータスコードを 200 に更新する

この関数では、レスポンスステータスを 200 に更新し、以下のシナリオでビューワーに返す静的な本文コンテンツを生成する方法を示しています。

- 関数がオリジンレスポンスでトリガーされる。
- オリジンサーバーからのレスポンスステータスがエラーステータスコード (4xx または 5xx) である。

Node.js

```
'use strict';

exports.handler = (event, context, callback) => {
    const response = event.Records[0].cf.response;

    /**
     * This function updates the response status to 200 and generates static
     * body content to return to the viewer in the following scenario:
```

```
* 1. The function is triggered in an origin response
* 2. The response status from the origin server is an error status code (4xx or
5xx)
*/

if (response.status >= 400 && response.status <= 599) {
    response.status = 200;
    response.statusDescription = 'OK';
    response.body = 'Body generation example';
}

callback(null, response);
};
```

Python

```
def lambda_handler(event, context):
    response = event['Records'][0]['cf']['response']

    ...

    This function updates the response status to 200 and generates static
    body content to return to the viewer in the following scenario:
    1. The function is triggered in an origin response
    2. The response status from the origin server is an error status code (4xx or
5xx)
    ...

    if int(response['status']) >= 400 and int(response['status']) <= 599:
        response['status'] = 200
        response['statusDescription'] = 'OK'
        response['body'] = 'Body generation example'
    return response
```

例: オリジンレスポンストリガーを使用してエラーステータスコードを 302 に更新する

この関数では、HTTP ステータスコードを 302 に更新して、異なるオリジンを設定した別のパス (キャッシュ動作) にリダイレクトする方法を示しています。次の点に注意してください。

- 関数がオリジンレスポンスでトリガーされる。
- オリジンサーバーからのレスポンスステータスがエラーステータスコード (4xx または 5xx) である。

Node.js

```
'use strict';

exports.handler = (event, context, callback) => {
  const response = event.Records[0].cf.response;
  const request = event.Records[0].cf.request;

  /**
   * This function updates the HTTP status code in the response to 302, to
  redirect to another
   * path (cache behavior) that has a different origin configured. Note the
  following:
   * 1. The function is triggered in an origin response
   * 2. The response status from the origin server is an error status code (4xx or
  5xx)
   */

  if (response.status >= 400 && response.status <= 599) {
    const redirect_path = `/plan-b/path?${request.querystring}`;

    response.status = 302;
    response.statusDescription = 'Found';

    /* Drop the body, as it is not required for redirects */
    response.body = '';
    response.headers['location'] = [{ key: 'Location', value: redirect_path }];
  }

  callback(null, response);
};
```

Python

```
def lambda_handler(event, context):
    response = event['Records'][0]['cf']['response']
    request = event['Records'][0]['cf']['request']

    ...

    This function updates the HTTP status code in the response to 302, to redirect
  to another
    path (cache behavior) that has a different origin configured. Note the
  following:
```

```
1. The function is triggered in an origin response
2. The response status from the origin server is an error status code (4xx or
5xx)
'''

if int(response['status']) >= 400 and int(response['status']) <= 599:
    redirect_path = '/plan-b/path?%s' % request['querystring']

    response['status'] = 302
    response['statusDescription'] = 'Found'

    # Drop the body as it is not required for redirects
    response['body'] = ''
    response['headers']['location'] = [{'key': 'Location', 'value':
redirect_path}]

return response
```

リクエストボディへのアクセス - 例

このセクションの例では、Lambda@Edge を使用して POST リクエストを操作する方法を示しています。

Note

これらの例を使用するには、デистриビューションの Lambda 関数の関連付けで [Include body] (ボディを含める) オプションを有効にする必要があります。デフォルトでは、有効になっていません。

- CloudFront コンソールでこの設定を有効にするには、Lambda 関数の関連付け のインクルード本文 のチェックボックスをオンにします。
- CloudFront API または でこの設定を有効にするにはAWS CloudFormation、trueの IncludeBodyフィールドを に設定しますLambdaFunctionAssociation。

トピック

- [例: リクエストトリガーを使用して HTML フォームを読み込む](#)
- [例: リクエストトリガーを使用して HTML フォームを変更する](#)

例: リクエストトリガーを使用して HTML フォームを読み込む

この関数では、「お問い合わせ」フォームなど HTML フォーム (ウェブフォーム) によって生成された POST リクエストボディを処理する方法を示しています。たとえば、次のような HTML フォームがあります。

```
<html>
  <form action="https://example.com" method="post">
    Param 1: <input type="text" name="name1"><br>
    Param 2: <input type="text" name="name2"><br>
    input type="submit" value="Submit">
  </form>
</html>
```

次の関数の例では、関数は CloudFront ビューワーリクエストまたはオリジンリクエストでトリガーされる必要があります。

Node.js

```
'use strict';

const querystring = require('querystring');

/**
 * This function demonstrates how you can read the body of a POST request
 * generated by an HTML form (web form). The function is triggered in a
 * CloudFront viewer request or origin request event type.
 */

exports.handler = (event, context, callback) => {
  const request = event.Records[0].cf.request;

  if (request.method === 'POST') {
    /* HTTP body is always passed as base64-encoded string. Decode it. */
    const body = Buffer.from(request.body.data, 'base64').toString();

    /* HTML forms send the data in query string format. Parse it. */
    const params = querystring.parse(body);

    /* For demonstration purposes, we only log the form fields here.
     * You can put your custom logic here. For example, you can store the
     * fields in a database, such as Amazon DynamoDB, and generate a response
     * right from your Lambda@Edge function.
     */
  }
}
```

```
    */
    for (let param in params) {
        console.log(`For "${param}" user submitted "${params[param]}".\n`);
    }
}
return callback(null, request);
};
```

Python

```
import base64
from urllib.parse import parse_qs

...
Say there is a POST request body generated by an HTML such as:

<html>
<form action="https://example.com" method="post">
    Param 1: <input type="text" name="name1"><br>
    Param 2: <input type="text" name="name2"><br>
    input type="submit" value="Submit">
</form>
</html>

...

...

This function demonstrates how you can read the body of a POST request
generated by an HTML form (web form). The function is triggered in a
CloudFront viewer request or origin request event type.
...

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']

    if request['method'] == 'POST':
        # HTTP body is always passed as base64-encoded string. Decode it
        body = base64.b64decode(request['body']['data'])

        # HTML forms send the data in query string format. Parse it
        params = {k: v[0] for k, v in parse_qs(body).items()}

    ...
```

```
For demonstration purposes, we only log the form fields here.  
You can put your custom logic here. For example, you can store the  
fields in a database, such as Amazon DynamoDB, and generate a response  
right from your Lambda@Edge function.
```

```
...
```

```
for key, value in params.items():  
    print("For %s use submitted %s" % (key, value))
```

```
return request
```

例: リクエストトリガーを使用して HTML フォームを変更する

この関数では、HTML フォーム (ウェブフォーム) によって生成された POST リクエストボディを変更する方法を示しています。関数は、CloudFront ビューワーリクエストまたはオリジンリクエストでトリガーされます。

Node.js

```
'use strict';  
  
const querystring = require('querystring');  
  
exports.handler = (event, context, callback) => {  
    var request = event.Records[0].cf.request;  
    if (request.method === 'POST') {  
        /* Request body is being replaced. To do this, update the following  
        /* three fields:  
        *   1) body.action to 'replace'  
        *   2) body.encoding to the encoding of the new data.  
        *  
        *       Set to one of the following values:  
        *  
        *       text - denotes that the generated body is in text format.  
        *           Lambda@Edge will propagate this as is.  
        *       base64 - denotes that the generated body is base64 encoded.  
        *           Lambda@Edge will base64 decode the data before sending  
        *           it to the origin.  
        *   3) body.data to the new body.  
        */  
        request.body.action = 'replace';  
        request.body.encoding = 'text';  
        request.body.data = getUpdatedBody(request);
```

```
    }
    callback(null, request);
};

function getUpdatedBody(request) {
  /* HTTP body is always passed as base64-encoded string. Decode it. */
  const body = Buffer.from(request.body.data, 'base64').toString();

  /* HTML forms send data in query string format. Parse it. */
  const params = querystring.parse(body);

  /* For demonstration purposes, we're adding one more param.
   *
   * You can put your custom logic here. For example, you can truncate long
   * bodies from malicious requests.
   */
  params['new-param-name'] = 'new-param-value';
  return querystring.stringify(params);
}
```

Python

```
import base64
from urllib.parse import parse_qs, urlencode

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
    if request['method'] == 'POST':
        ...

        Request body is being replaced. To do this, update the following
        three fields:
            1) body.action to 'replace'
            2) body.encoding to the encoding of the new data.

            Set to one of the following values:

                text - denotes that the generated body is in text format.
                    Lambda@Edge will propagate this as is.
                base64 - denotes that the generated body is base64 encoded.
                    Lambda@Edge will base64 decode the data before sending
                    it to the origin.
            3) body.data to the new body.
        ...
```

```
request['body']['action'] = 'replace'
request['body']['encoding'] = 'text'
request['body']['data'] = getUpdatedBody(request)
return request

def getUpdatedBody(request):
    # HTTP body is always passed as base64-encoded string. Decode it
    body = base64.b64decode(request['body']['data'])

    # HTML forms send data in query string format. Parse it
    params = {k: v[0] for k, v in parse_qs(body).items()}

    # For demonstration purposes, we're adding one more param

    # You can put your custom logic here. For example, you can truncate long
    # bodies from malicious requests
    params['new-param-name'] = 'new-param-value'
    return urlencode(params)
```

エッジ関数に対する制限

以下のトピックでは、CloudFront 関数と Lambda@Edge に適用される制限について説明します。一部の制限はすべてのエッジ関数に適用され、その他の制限は CloudFront 関数または Lambda@Edge にのみ適用されます。

クォータ (以前は制限と呼ばれていました) の詳細については、「[CloudFront 関数のクォータ](#)」と「[Lambda@Edge のクォータ](#)」を参照してください。

トピック

- [すべてのエッジ機能に対する制限](#)
- [CloudFront 関数の制限](#)
- [Lambda@Edge に対する制限](#)

すべてのエッジ機能に対する制限

関数と Lambda@Edge の両方のすべてのエッジ CloudFront 関数には、次の制限が適用されます。

トピック

- [AWS アカウント の所有権](#)

- [CloudFront 関数と Lambda@Edge の組み合わせ](#)
- [HTTP ステータスコード](#)
- [HTTP ヘッダー](#)
- [クエリ文字列](#)
- [\[URI\]](#)
- [URI とクエリ文字列のエンコーディング](#)
- [Microsoft Smooth Streaming](#)
- [タグ付け](#)

AWS アカウント の所有権

エッジ関数を CloudFront ディストリビューションに関連付けるには、関数とディストリビューションが同じ によって所有されている必要がありますAWS アカウント。

CloudFront 関数と Lambda@Edge の組み合わせ

所定のキャッシュ動作については、以下の制限が適用されます。

- イベントタイプ (ビューワーリクエスト、オリジンリクエスト、オリジンレスポンス、ビューワーレスポンス) はそれぞれ、エッジ関数の関連付けを 1 つしか持てません。
- ビューワーイベント (ビューワーリクエストとビューワーレスポンス) で CloudFront Functions と Lambda@Edge を組み合わせることはできません。

上記以外のすべてのエッジ関数の組み合わせが許可されます。以下の表は、許可される組み合わせの説明です。

		CloudFront 関数	
		ビューワーリクエスト	ビューワーレスポンス
Lambda@Edge	ビューワーリクエスト	許可されていません	許可されていません
	オリジンリクエスト	許可	許可

	オリジンレスポンス	許可	許可
	ビューワーレスポンス	許可されていません	許可されていません

HTTP ステータスコード

CloudFront オリジンが HTTP ステータスコード 400 以上を返す場合、 はビューワーレスポンスイベントのエッジ関数を呼び出しません。

オリジンレスポンスイベントの Lambda@Edge 関数は、オリジンが 400 以上の HTTP ステータスコードを返す場合を含め、すべてのオリジンレスポンスに対して呼び出されます。詳細については、「[オリジンレスポンストリガーでの HTTP レスポンスの更新](#)」を参照してください。

HTTP ヘッダー

特定の HTTP ヘッダーは許可されていません。これは、これらのヘッダーがエッジ関数に公開されておらず、関数がそれらを追加できないことを意味します。他のヘッダーは読み取り専用です。つまり、関数はこれらを読み取れますが、追加したり変更したりすることはできません。

トピック

- [許可されていないヘッダー](#)
- [読み取り専用ヘッダー](#)

許可されていないヘッダー

以下の HTTP ヘッダーはエッジ関数に公開されておらず、関数はこれらを追加できません。関数がこれらのヘッダーのいずれかを追加すると、検証に失敗 CloudFront し、ビューワーに HTTP ステータスコード 502 (Bad Gateway) CloudFront が返されます。

- Connection
- Expect
- Keep-Alive
- Proxy-Authenticate
- Proxy-Authorization
- Proxy-Connection

- Trailer
- Upgrade
- X-Accel-Buffering
- X-Accel-Charset
- X-Accel-Limit-Rate
- X-Accel-Redirect
- X-Amz-Cf-*
- X-Amzn-Auth
- X-Amzn-Cf-Billing
- X-Amzn-Cf-Id
- X-Amzn-Cf-Xff
- X-Amzn-Errortype
- X-Amzn-Fle-Profile
- X-Amzn-Header-Count
- X-Amzn-Header-Order
- X-Amzn-Lambda-Integration-Tag
- X-Amzn-RequestId
- X-Cache
- X-Edge-*
- X-Forwarded-Proto
- X-Real-IP

読み取り専用ヘッダー

以下のヘッダーは読み取り専用です。関数はこれらを読み取って関数ロジックへの入力として使用できますが、値を変更することはできません。関数が読み取り専用ヘッダーを追加または編集すると、リクエストは CloudFront 検証に失敗し CloudFront、ビューワーに HTTP ステータスコード 502 (Bad Gateway) を返します。

ビューワーリクエストイベントの読み取り専用ヘッダー

以下のヘッダーは、ビューワーリクエストイベントでは読み取り専用になります。

- Content-Length
- Host
- Transfer-Encoding
- Via

オリジンリクエストイベントの読み取り専用ヘッダー (Lambda@Edge 限定)

以下のヘッダーは、Lambda@Edge にしかないオリジンリクエストイベントでは読み取り専用になります。

- Accept-Encoding
- Content-Length
- If-Modified-Since
- If-None-Match
- If-Range
- If-Unmodified-Since
- Transfer-Encoding
- Via

オリジンレスポンスイベントの読み取り専用ヘッダー (Lambda@Edge 限定)

以下のヘッダーは、Lambda@Edge にしかないオリジンレスポンスイベントでは読み取り専用になります。

- Transfer-Encoding
- Via

ビューワーレスポンスイベントの読み取り専用ヘッダー

以下のヘッダーは、CloudFront Functions と Lambda@Edge の両方のビューワーレスポンスイベントでは読み取り専用です。

- Warning
- Via

以下のヘッダーは、Lambda@Edge のビューワレスポンスイベントでは読み取り専用になります。

- Content-Length
- Content-Encoding
- Transfer-Encoding

クエリ文字列

以下の制限は、リクエスト URI 内のクエリ文字列を読み取る、更新する、または作成する関数に適用されます。

- (Lambda@Edge 限定) オリジンリクエストまたはオリジンレスポンス関数のクエリ文字列にアクセスするには、キャッシュポリシーまたはオリジンリクエストポリシーが [Query strings] (クエリ文字列) に対して [All] (すべて) に設定されている必要があります。
- 関数は、ビューワリクエストイベントとオリジンリクエストイベントのクエリ文字列を作成または更新できます (オリジンリクエストイベントがあるのは Lambda@Edge だけです)。
- 関数は、オリジンレスポンスイベントとビューワレスポンスイベントのクエリ文字列を読み取ることができますが、それらを作成または更新することはできません (オリジンレスポンスイベントがあるのは Lambda@Edge だけです)。
- 関数がクエリ文字列を作成または更新する場合は、以下の制限が適用されます。
 - クエリ文字列に、スペース、制御文字、またはフラグメント識別子 (#) を含めることはできません。
 - クエリ文字列を含めた URI の合計サイズは、8,192 文字未満にする必要があります。
 - URI およびクエリ文字列には、パーセントエンコーディングを使用することをお勧めします。詳細については、「[URI とクエリ文字列のエンコーディング](#)」を参照してください。

[URI]

関数でリクエストの URI を変更しても、リクエストのキャッシュ動作や転送先オリジンは変わりません。

クエリ文字列を含めた URI の合計サイズは、8,192 文字未満にする必要があります。

URI とクエリ文字列のエンコーディング

エッジ関数に渡される URI とクエリ文字列値は、UTF-8 でエンコードされています。関数は、それが返す URI とクエリ文字列値に UTF-8 エンコーディングを使用する必要があります。パーセントエンコーディングには、UTF-8 エンコーディングと互換性があります。

次のリストでは、が URI とクエリ文字列値のエンコード CloudFront を処理する方法について説明します。

- リクエスト内の値が UTF-8 でエンコードされている場合、CloudFront は値を変更せずに関数に転送します。
- リクエスト内の値が [ISO-8859-1 でエンコードされた場合](#)、は値を UTF-8 エンコード CloudFront に変換してから関数に転送します。
- リクエスト内の値が他の文字エンコードを使用してエンコードされている場合、は ISO-8859-1 エンコードされている CloudFront と仮定し、ISO-8859-1 から UTF-8 への変換を試みます。

Important

変換された文字は、元のリクエストの値の正しい解釈ではない可能性があります。これは、関数またはオリジンが意図しない結果を生成する原因になる場合があります。

がオリジン CloudFront に転送する URI とクエリ文字列の値は、関数が値を変更するかどうかによって異なります。

- 関数が URI またはクエリ文字列を変更しない場合、CloudFront はリクエストで受け取った値をオリジンに転送します。
- 関数が URI またはクエリ文字列を変更すると、は UTF-8 でエンコードされた値を CloudFront 転送します。

Microsoft Smooth Streaming

Microsoft Smooth Streaming 形式にトランスコードしたメディアファイルのストリーミングに使用している CloudFront デイストリビューションでは、エッジ関数を使用できません。

タグ付け

エッジ関数にタグを追加することはできません。でのタグ付けの詳細については CloudFront、「」を参照してください [Amazon CloudFront デистриビューションのタグ付け](#)。

CloudFront 関数の制限

以下の制限は CloudFront 関数にのみ適用されます。

クォータ (以前は制限と呼ばれていました) の詳細については、「」を参照してください [CloudFront 関数のクォータ](#)。

ログ

Functions の CloudFront 関数ログは 10 KB で切り捨てられます。

リクエスト本文

CloudFront 関数は HTTP リクエストの本文にアクセスできません。

CloudFront KeyValueCollectionStore API を使用する場合のリージョンAWS Security Token Serviceエンドポイント

一時的なセキュリティ認証情報で Signature Version 4A (SigV4A) を使用して [CloudFront KeyValueCollectionStore API](#) を呼び出す場合。例えば、AWS Identity and Access Management (IAM) ロールを使用する場合は、のリージョンエンドポイントから一時的な認証情報をリクエストしてください AWS STS。AWS STS (sts.amazonaws.com) にグローバルエンドポイントを使用する場合、AWS STSはグローバルエンドポイントから一時的な認証情報を生成します。これは SigV4A ではサポートされていません。その結果、認証エラーが発生します。この問題を解決するには、「IAM ユーザーガイド」の「」に記載されている [のリージョンエンドポイントAWS STS](#)のいずれかを使用してください。AWS STS リージョンのエンドポイントを使用するように SAML を設定している場合は、「[フェイルオーバーにリージョンの SAML エンドポイントを使用する方法](#)」ブログ記事を参照してください。

ランタイム

CloudFront Functions ランタイム環境は動的コード評価をサポートしておらず、ネットワーク、ファイルシステム、タイマーへのアクセスを制限します。詳細については、「[制限された機能](#)」を参照してください。

コンピューティング使用率

CloudFront 関数の実行にかかる時間は制限されており、コンピューティング使用率として測定されます。コンピューティング使用率は、関数の実行にかかった時間を最大許容時間に対する割合として示す 0 から 100 の数値です。例えば、コンピューティング使用率が 35 の場合、関数は最大許容時間の 35% で完了したことを意味します。

[関数をテストする](#)ときは、テストイベントの出力でコンピューティング使用率の値を確認できます。本稼働関数の場合、[コンピューティング使用率メトリクス](#)は、[CloudFront コンソールのモニタリングページ](#)、または で確認できます CloudWatch。

Lambda@Edge に対する制限

以下の制限は、Lambda@Edge のみに適用されます。

クォータ (以前は制限と呼ばれていました) の詳細については、「」を参照してください [Lambda@Edge のクォータ](#)。

HTTP ステータスコード

ビューワーレスポンスイベントの Lambda@Edge 関数は、レスポンスがオリジンからのものか CloudFront キャッシュからのものかにかかわらず、レスポンスの HTTP ステータスコードを変更することはできません。

Lambda 関数のバージョン

\$LATEST やエイリアスではなく、Lambda 関数の番号付きバージョンを使用する必要があります。

Lambda のリージョン

Lambda 関数は、米国東部 (バージニア北部) リージョンにある必要があります。

Lambda のロール許可

Lambda 関数に関連付けられている IAM 実行ロールは、サービスプリンシパル `lambda.amazonaws.com` と `edgelambda.amazonaws.com` によるそのロールの引き受けを許可する必要があります。詳細については、「[Lambda@Edge 用の IAM アクセス権限とロールの設定](#)」を参照してください。

Lambda の機能

以下の Lambda 機能は、Lambda@Edge でサポートされていません。

- [Auto] (自動) (デフォルト) 以外の [Lambda ランタイム管理設定](#)。
- VPC 内のリソースにアクセスするための Lambda 関数の設定。
- [Lambda 関数のデッドレターキュー](#)。
- [Lambda 環境変数](#)。
- [AWS Lambda レイヤー](#)を使用する Lambda 関数。
- [AWS X-Ray の使用](#)。
- [Lambda の予約された同時実行とプロビジョニングされた同時実行](#)。
- [コンテナイメージとして定義された Lambda 関数](#)。
- [arm64 アーキテクチャを使用する Lambda 関数](#)。
- 512 MB を超えるエフェメラルストレージを持つ Lambda 関数。

ランタイムのサポート

Lambda@Edge は、以下のランタイムで Lambda 関数をサポートします。

Node.js	Python
<ul style="list-style-type: none">• Node.js 20• Node.js 18• Node.js 161• Node.js 142• Node.js 12²• Node.js 10²• Node.js 82• Node.js 62	<ul style="list-style-type: none">• 「Python 3.11」• 「Python 3.10」• Python 3.9• Python 3.8• Python 3.7

1このバージョンの Node.js はサポートが終了し、間もなく によって廃止されますAWS Lambda。

2このバージョンの Node.js はサポートが終了し、 によって完全に廃止されましたAWS Lambda。

Node.js の非推奨バージョンでは、関数を作成または更新できません。既存の関数をこれらのバージョンに関連付けることができるのは、CloudFront デイストリビューションのみです。デイストリビューションに関連付けられているこれらのバージョンを持つ関数は、引き続き実行されます。ただ

し、関数を新しいバージョンの Node.js に移動することをお勧めします。詳細については、「AWS Lambdaデベロッパーガイド」の「[ランタイム廃止ポリシー](#)」および「」の「[Node.js リリーススケジュール](#)」を参照してください GitHub。

Tip

ベストプラクティスとして、提供されているランタイムの最新バージョンを使用して、パフォーマンスの向上と新機能を取得します。

CloudFront ヘッダー

Lambda@Edge 関数は、次のいずれか CloudFrontのヘッダーを読み取る、編集、削除、または追加できます。

- CloudFront-Forwarded-Proto
- CloudFront-Is-Desktop-Viewer
- CloudFront-Is-Mobile-Viewer
- CloudFront-Is-SmartTV-Viewer
- CloudFront-Is-Tablet-Viewer
- CloudFront-Viewer-Country1

次の点に注意してください。

- これらのヘッダー CloudFront を追加する場合は、[キャッシュポリシー](#)または[オリジンリクエストポリシー](#)を使用してヘッダーを追加する CloudFront ように を設定する必要があります。
- CloudFront は、ビューワーリクエストイベントの後にヘッダーを追加します。つまり、ビューワーリクエスト関数で Lambda@Edge がヘッダーを使用できないことを意味します。
- ビューワーリクエストにこれらの名前のヘッダーが含まれ、[キャッシュポリシー](#)または[オリジンリクエストポリシー](#)を使用してこれらのヘッダーを追加する CloudFront ように を設定した場合、はビューワーリクエストにあったヘッダー値を CloudFront 上書きします。ビューワー向けの関数はビューワーリクエストのヘッダー値を確認し、オリジン向けの関数は が CloudFront 追加したヘッダー値を表示します。
- 1CloudFront-Viewer-Country ヘッダー – ビューワーリクエスト関数がこのヘッダーを追加すると、検証に失敗し、ビューワーに HTTP ステータスコード 502 (Bad Gateway) CloudFront が返されます。

Include Body オプションがあるリクエストボディに対する制限

Lambda@Edge 関数にリクエストボディを公開するために [Include Body] オプションを選択する場合は、公開または置き換えられたボディの一部に以下の情報クォータとサイズクォータが適用されます。

- CloudFront は、常にリクエストボディを base64 でエンコードしてから Lambda@Edge に公開します。
- リクエストボディが大きい場合、は次のように、Lambda@Edge に公開する前に CloudFront 切り捨てます。
 - ビューワーリクエストでは、ボディが 40 KB で切り捨てられます。
 - オリジンリクエストでは、ボディが 1 MB で切り捨てられます。
- リクエストボディに読み取り専用としてアクセスする場合、は元のリクエストボディ全体をオリジン CloudFront に送信します。
- Lambda@Edge 関数がリクエストボディを置き換える場合、関数が返すボディに以下のサイズクォータが適用されます。
 - Lambda@Edge 関数がボディをプレーンテキストとして返す場合:
 - ビューワーリクエストでは、ボディが 40 KB で切り捨てられます。
 - オリジンリクエストでは、ボディが 1 MB で切り捨てられます。
 - Lambda@Edge 関数がボディを base64 でエンコードされたテキストとして返す場合:
 - ビューワーリクエストイベントでは、ボディが 53.2 KB で切り捨てられます。
 - オリジンリクエストイベントでは、ボディが 1.33 MB で切り捨てられます。

レポート、メトリクス、ログ

CloudFront には、リソースのレポート、モニタリング、ログ記録のためのオプションがいくつか用意されています CloudFront。

- レポートを表示およびダウンロードして、請求レポート、キャッシュ統計、人気のあるコンテンツ、上位リファラなど、ディストリビューションの CloudFront 使用状況とアクティビティを確認できます。
- [エッジコンピューティング関数を含む](#) を CloudFront コンソールで直接 CloudFront モニタリングおよび追跡できます。または、Amazon を使用してモニタリングおよび追跡できます CloudWatch。は、Lambda@Edge と CloudFront Functions の両方のディストリビューションおよびエッジ関数 CloudWatch のさまざまなメトリクスを CloudFront に送信します。
- ディストリビューション CloudFront ションが受信するビューワーリクエストのログは、標準ログまたはリアルタイムログで表示できます。ビューワーリクエストログに加えて、CloudWatch ログを使用して、Lambda@Edge と CloudFront 関数の両方のエッジ関数のログを取得できます。AWS CloudTrail を使用して、の CloudFront API アクティビティのログを取得することもできます AWS アカウント。
- を使用して、CloudFront リソースの設定変更を追跡できます AWS Config。

これらのオプションのそれぞれの詳細については、以下のトピックを参照してください。

トピック

- [AWS の 請求および使用状況レポート CloudFront](#)
- [CloudFront コンソールの レポート](#)
- [Amazon による CloudFront メトリクスのモニタリング CloudWatch](#)
- [CloudFront およびエッジ関数のログ記録](#)
- [AWS Config による設定変更の追跡](#)

AWS の 請求および使用状況レポート CloudFront

AWS には、の 2 つの使用状況レポートが用意されています CloudFront。

- 請求レポートは、を含む、AWS 使用しているサービスのすべてのアクティビティの概要です CloudFront。詳細については、「[the section called “AWS の 請求レポート CloudFront”](#)」を参照してください。

- 使用状況レポートは、特定のサービスのアクティビティの概要を時間、日、または月単位で集約して示します。また、使用状況をグラフィカルに表現するCloudFront 使用状況グラフも含まれています。詳細については、「[the section called “AWS の 使用状況レポート CloudFront”](#)」を参照してください。

これらのレポートを理解していただくために、「[the section called “のAWS請求書とAWS使用状況レポートの解釈 CloudFront”](#)」の詳細情報を参照してください。

Note

他の AWS サービスと同様に、は実際に使用した分だけ CloudFront 課金します。詳細については、「[the section called “CloudFront 料金”](#)」を参照してください。

トピック

- [AWS の 請求レポート CloudFront](#)
- [AWS の 使用状況レポート CloudFront](#)
- [のAWS請求書とAWS使用状況レポートの解釈 CloudFront](#)

AWS の 請求レポート CloudFront

AWS Management Consoleの請求ページで、AWS の使用状況と料金の概要をサービス別に表示できます。

また、レポートの詳細版を CSV 形式でダウンロードすることもできます。請求明細レポートには、に適用される以下の値が含まれます CloudFront。

- ProductCode — AmazonCloudFront
- UsageType — 次のいずれかの値。
 - データ転送のタイプを識別するコード
 - Invalidations
 - SSL-Cert-Custom

詳細については、「[the section called “のAWS請求書とAWS使用状況レポートの解釈 CloudFront”](#)」を参照してください。

- ItemDescription — の請求レポートの説明UsageType。

- Usage Start Date/Usage End Date — 使用状況に該当する協定世界時 (UTC) による日付。
- Usage Quantity — 以下の値のいずれかです。
 - 指定した期間のリクエストの数
 - データ転送量 (GB)
 - 無効にされたオブジェクトの数
 - 有効な CloudFront デイストリビューションに関連付けられた SSL 証明書がある日割り計算月の合計。たとえば、ある証明書を有効なデイストリビューションに 1 か月まるまる関連付け、別の証明書を有効なデイストリビューションに半月だけ関連付けた場合、この値は 1.5 になります。

請求情報の概要を表示し、詳細請求レポートをダウンロードするには

1. AWS Management Console (<https://console.aws.amazon.com/console/home>) にサインインします。
2. タイトルバーで自分のユーザー名を選択し、[Billing Dashboard] (請求ダッシュボード) を選択します。
3. ナビゲーションペインで [Bills (請求書)] を選択します。
4. の概要情報を表示するには CloudFront、詳細 で を選択します CloudFront。
5. 詳細請求レポートを CSV 形式でダウンロードするには、[Download CSV] (CSV のダウンロード) を選択し、画面の指示に従ってレポートを保存します。

AWS の 使用状況レポート CloudFront

AWS は、請求レポートよりも詳細で、CloudFront アクセスログよりも詳細ではない CloudFront 使用状況レポートを提供します。使用状況レポートには、使用状況データが時間、日、または月単位で集計され、リージョンと使用タイプ別に操作が一覧で示されます (たとえば、オーストラリアリージョンからデータが転送されたなど)。

CloudFront 使用状況レポートには、次の値が含まれます。

- Service — AmazonCloudFront
- Operation — HTTP メソッド。値には、DELETE、GET、HEAD、OPTIONS、PATCH、POST、PUT があります。
- UsageType — 次のいずれかの値。
 - データ転送のタイプを識別するコード

- Invalidations
- SSL-Cert-Custom

詳細については、「[the section called “のAWS請求書とAWS使用状況レポートの解釈 CloudFront”](#)」を参照してください。

- リソース — 使用状況に関連付けられた CloudFront デイストリビューションの ID、またはデイストリビューションに関連付けた SSL 証明書の証明書 ID のいずれかです。
- StartTime/EndTime — 使用状況が適用される協定世界時 (UTC) の日。
- UsageValue — (1) 指定された期間中のリクエスト数、または (2) バイト単位で転送されるデータ量。

のオリジンとして Amazon S3 を使用している場合は CloudFront、Amazon S3 の使用状況レポートの実行も検討してください。ただし、デイストリビューションの CloudFrontオリジンとして以外の目的で Amazon S3 を使用する場合、使用量に適用される部分が明確でない可能性があります CloudFront。

Tip

がオブジェクトに対して CloudFront 受信するすべてのリクエストの詳細については、デイストリビューションのアクセスログを有効にします CloudFront。詳細については、「[the section called “標準ログ \(アクセスログ\) の使用”](#)」を参照してください。

のAWS請求書とAWS使用状況レポートの解釈 CloudFront

のAWS請求書 CloudFront には、明確にわからないコードと略語が含まれています。次の表の 1 列目には、請求書に記載される項目が一覧になっており、各項目の内容が説明されています。

さらに、のAWS請求書よりも詳細な CloudFront 内容を含む のAWS使用状況レポートを取得できます CloudFront。表の 2 列目には、使用状況レポートに記載される項目が一覧になっており、請求書の項目と使用状況レポートの項目との相互関係が示されています。

2 つの列のコードには、たいてい、アクティビティの場所を示す 2 文字の略語が入っています。次の表のコードの *region* は、AWS 請求書と使用状況レポートでは以下の 2 文字の略語に置き換えられます。

- AP: 香港、フィリピン、韓国、シンガポール、台湾、シンガポール (アジアパシフィック)

- AU: オーストラリア
- CA: カナダ
- EU: ヨーロッパおよびイスラエル
- IN: インド
- JP: 日本
- ME: 中東
- SA: 南米
- US: 米国
- ZA: 南アフリカ

リージョン別の料金の詳細については、[「Amazon CloudFront 料金表」](#)を参照してください。

Note

この表には、Amazon S3 バケットからエッジロケーションへの CloudFront オブジェクトの転送料金は含まれていません。この料金が発生している場合、AWS 請求書の [AWS Data Transfer] (データ転送) の部分に料金が表記されます。

CloudFront 請求書の項目	使用状況レポートの CloudFront 使用状況タイプ列の値
##### DataTransfer--出力バイト	<i>region</i> -Out-Bytes-HTTP-Static:
ユーザーGETとHEADリクエストに応じて##### #の CloudFront エッジロケーションから供給された合計バイト数。	TTL ≥ 3,600 秒のオブジェクトのために HTTP 経由で供給されたバイト数。
	<i>region</i> -Out-Bytes-HTTPS-Static:
	TTL ≥ 3,600 秒のオブジェクトのために HTTPS 経由で供給されたバイト数。
	<i>region</i> -Out-Bytes-HTTP-Dynamic:
	TTL < 3,600 秒のオブジェクトのために HTTP 経由で提供されたバイト数。

CloudFront 請求書の項目	使用状況レポートの CloudFront 使用状況タイプ列の値
	<p><i>region</i>-Out-Bytes-HTTPS-Dynamic:</p> <p>TTL < 3,600 秒のオブジェクトのために HTTPS 経由で配信されたバイト数。</p> <p><i>region</i>-Out-OBytes-HTTP-Proxy</p> <p>、DELETE、PATCHPOSTおよびPUTリクエストへのレスポンスとして OPTIONSHTTP 経由で からビューワー CloudFront に返されたバイト数。</p> <p><i>region</i>-Out-Bytes-HTTPS-Proxy:</p> <p>、DELETE、OPTIONS、PATCHPOSTおよびPUTリクエストに回答して HTTPS 経由で からビューワー CloudFront に返されるバイト数。</p>
<p>#### DataTransfer--Out-OBytes</p> <p>、DELETE、OPTIONSPATCHPOST、およびPUTリクエストに回答して CloudFront エッジロケーションからオリジンまたはエッジ関数に転送された合計バイト数。料金には、クライアントからサーバーへの WebSocket データ転送が含まれます。</p>	<p><i>region</i>-Out-OBytes-HTTP-Proxy</p> <p>、DELETE、PATCH、POSTおよびPUTリクエストに回答して CloudFront 、エッジロケーションからオリジンまたはエッジ関数に HTTP OPTIONS経由で転送された合計バイト数。</p> <p><i>region</i>-Out-OBytes-HTTPS-Proxy</p> <p>、DELETE、PATCH、POSTおよびPUTリクエストに回答して CloudFront 、エッジロケーションからオリジンまたはエッジ関数に HTTPS OPTIONS経由で転送された合計バイト数。</p>

CloudFront 請求書の項目	使用状況レポートの CloudFront 使用状況タイプ列の値
<p><i>region</i>-Requests-Tier1</p> <p>HTTP GET および HEAD リクエストの数。</p>	<p><i>region</i>-Requests-HTTP-Static</p> <p>TTL ≥ 3,600 秒のオブジェクトのために供給された HTTP GET および HEAD リクエストの数。</p> <p><i>region</i>-Requests-HTTP-Dynamic</p> <p>TTL < 3,600 秒のオブジェクトのために供給された HTTP GET および HEAD リクエストの数。</p>
<p><i>region</i>-Requests-Tier2-HTTPS</p> <p>HTTPS GET および HEAD リクエストの数。</p>	<p><i>region</i>-Requests-HTTPS-Static</p> <p>TTL ≥ 3,600 秒のオブジェクトのために供給された HTTPS GET および HEAD リクエストの数。</p> <p><i>region</i>-Requests-HTTPS-Dynamic</p> <p>TTL < 3,600 秒のオブジェクトのために供給された HTTPS GET および HEAD リクエストの数。</p>
<p><i>region</i>-Requests-HTTP-Proxy</p> <p>がオリジンまたは エッジ関数 に転送する OPTIONS CloudFront HTTP DELETE、PATCH、POST、および PUT リクエストの数。</p> <p>がオリジンまたはエッジ関数 CloudFront に転送する HTTP WebSocket リクエスト (GET Upgrade: websocket ヘッダーを持つ リクエスト) の数も含まれます。</p>	<p><i>region</i>-Requests-HTTP-Proxy</p> <p>CloudFront 請求書の対応する項目と同じです。</p>

CloudFront 請求書の項目	使用状況レポートの CloudFront 使用状況タイプ列の値
<p><i>region-Requests-HTTPS-Proxy</i></p> <p>がオリジンまたはエッジ関数に転送する DELETE OPTIONS CloudFront HTTPS PATCH、POST、、、および PUTリクエストの数。</p> <p>がオリジンまたはエッジ関数 CloudFront に転送する HTTPS WebSocketリクエスト (GET Upgrade: websocket ヘッダー付きのリクエスト) の数も含まれます。</p>	<p><i>region-Requests-HTTPS-Proxy</i></p> <p>CloudFront 請求書の対応する項目と同じです。</p>
<p><i>region-Requests-HTTPS-Proxy-FLE</i></p> <p>オリジンまたはエッジ関数 CloudFront に転送するフィールドレベル暗号化で処理された HTTPS DELETEOPTIONS、PATCH、、、および POSTリクエストの数。</p>	<p><i>region-Requests-HTTPS-Proxy-FLE</i></p> <p>CloudFront 請求書の対応する項目と同じです。</p>
<p><i>##### - バイト-OriginShield</i></p> <p>オリジンから任意のリージョン別エッジキャッシュに転送された合計バイト数。Origin Shieldとして有効になっているリージョン別エッジキャッシュを含みます。</p>	<p><i>##### - バイト-OriginShield</i></p> <p>オリジンから任意のリージョン別エッジキャッシュに転送された合計バイト数。Origin Shieldとして有効になっているリージョン別エッジキャッシュを含みます。</p>
<p><i>##### -OBytes -OriginShield</i></p> <p>任意のリージョン別エッジキャッシュからオリジンに転送された合計バイト数。Origin Shieldとして有効になっているリージョン別エッジキャッシュを含みます。</p>	<p><i>##### -OBytes -OriginShield</i></p> <p>任意のリージョン別エッジキャッシュからオリジンに転送された合計バイト数。Origin Shieldとして有効になっているリージョン別エッジキャッシュを含みます。</p>

CloudFront 請求書の項目	使用状況レポートの CloudFront 使用状況タイプ列の値
<p>region -Requests-OriginShield</p> <p>増分レイヤーとして Origin Shield に送信されたリクエストの数。オリジンにプロキシ化される動的 (キャッシュ不可能な) リクエストの場合、Origin Shield は常に増分レイヤーとなります。キャッシュ可能なリクエストの場合、Origin Shield は増分レイヤーになることがあります。</p> <p>詳細については、「the section called “Origin Shield のコストの見積もり”」を参照してください。</p> <p>無効化</p> <p>オブジェクトの無効化 (CloudFront エッジロケーションからのオブジェクトの削除) の料金の詳細については、「」を参照してください。ファイルの無効化に対する支払い。</p>	<p>##### -Requests-OriginShield</p> <p>増分レイヤーとして Origin Shield に送信されたリクエストの数。オリジンにプロキシ化される動的 (キャッシュ不可能な) リクエストの場合、Origin Shield は常に増分レイヤーとなります。キャッシュ可能なリクエストの場合、Origin Shield は増分レイヤーになることがあります。</p> <p>詳細については、「the section called “Origin Shield のコストの見積もり”」を参照してください。</p> <p>無効化</p> <p>CloudFront 請求書の対応する項目と同じです。</p>
<p>SSL-Cert-Custom</p> <p>ディストリビューションに が CloudFront 割り当てたデフォルトの SSL 証明書とドメイン名を使用する代わりに、example.com などの CloudFront 代替ドメイン名で CloudFront SSL 証明書を使用する場合の料金。</p>	<p>SSL-Cert-Custom</p> <p>CloudFront 請求書の対応する項目と同じです。</p>

CloudFront コンソールの レポート

CloudFront コンソールには CloudFront 、アクティビティに関する次のようなさまざまなレポートが含まれています。

- [CloudFront cache statistics reports](#)

- [CloudFront popular objects report](#)
- [CloudFront top referrers report](#)
- [CloudFront usage reports](#)
- [CloudFront viewers reports](#)

これらのレポートのほとんどは、が CloudFront 受信するすべてのユーザーリクエストに関する詳細情報を含む CloudFront アクセスログのデータに基づいています。このレポートを表示するために、アクセスログを有効にする必要はありません。詳細については、「[標準ログ \(アクセスログ\) の設定および使用](#)」を参照してください。CloudFront 使用状況レポートは、特別な設定 CloudFrontを必要としないのAWS使用状況レポートに基づいています。詳細については、「[AWS の 使用状況レポート CloudFront](#)」を参照してください。

CloudFront キャッシュ統計レポート

CloudFront キャッシュ統計レポートには、次の情報が含まれます。

- Total requests – すべての HTTP ステータスコード (200、404 など) およびすべてのメソッド (GET、HEAD、POST など) に対するリクエストの総数を示します。
- 結果タイプ別のビューワーリクエストの割合 — 選択したCloudFront デистриビューションの合計ビューワーリクエストの割合として、ヒット、ミス、エラーを表示します。
- Bytes transferred to viewers – バイト総数とミスのバイト数を示します。
- HTTP status codes – HTTP ステータスコード別のビューワーリクエスト数を示します。
- Percentage of GET requests that didn't finish downloading – リクエストされたオブジェクトのダウンロードを終了しなかったビューワーの GET リクエストが、リクエスト総数に占める割合 (%) を示します。

詳細については、「[CloudFront キャッシュ統計レポート](#)」を参照してください。

CloudFront 人気オブジェクトレポート

CloudFront 人気オブジェクトレポートには、オブジェクトのリクエスト数、ヒット数とミス数、ヒット率、ミスのために供給されたバイト数、供給された合計バイト数、不完全なダウンロード数、HTTP ステータスコード (2xx、3xx、4xx、5xx) によるリクエスト数など、これらのオブジェクトに関して最も人気のある 50 件のオブジェクトと統計が一覧表示されます。

詳細については、「[CloudFront 人気オブジェクトレポート](#)」を参照してください。

CloudFront トップリファラレポート

CloudFront トップリファラレポートには、上位 25 件のリファラ、リファラからのリクエスト数、およびリファラからのリクエスト数を、指定した期間のリクエストの総数に対するパーセンテージで示します。

詳細については、「[CloudFront トップリファラレポート](#)」を参照してください。

CloudFront 使用状況レポート

CloudFront 使用状況レポートには、次の情報が含まれます。

- リクエスト数 – 指定した CloudFront ディストリビューションの各時間間隔で、選択したリージョンのエッジロケーションから CloudFront 応答するリクエストの合計数を示します。
- プロトコルごとに転送されるデータと送信先によって転送されるデータ – どちらも、指定された CloudFront ディストリビューションの各時間間隔で、選択したリージョンの CloudFront エッジロケーションから転送されるデータの合計量を示します。データは、以下のように異なる方法で分けられます。
 - プロトコル別 – プロトコル (HTTP または HTTPS) 別にデータを分けます。
 - 送信先別 – 送信先 (ユーザーまたはオリジン) 別にデータを分けます。

詳細については、「[CloudFront 使用状況レポート](#)」を参照してください。

CloudFront ビューワーレポート

CloudFront ビューワーレポートには、次の情報が含まれます。

- Devices – ユーザーがコンテンツにアクセスするために使用するデバイスのタイプ (デスクトップやモバイルなど) を示します
- Browsers – Chrome や Firefox など、コンテンツにアクセスするときに、ユーザーが最も頻繁に使用するブラウザの名前 (または名前とバージョン) を示します
- Operating systems – Linux、macOS、Windows など、コンテンツにアクセスするときにビューワーが最も頻繁に実行するオペレーティングシステムの名前 (または名前とバージョン) を示します
- Locations – コンテンツに最も頻繁にアクセスするビューワーの場所 (国、または米国の州/準州) を示します

詳細については、「[CloudFront ビューワーレポート](#)」を参照してください。

CloudFront キャッシュ統計レポート

Amazon CloudFront コンソールを使用して、CloudFront エッジロケーションに関連する統計をグラフィカルに表示できます。これらの統計のデータは、CloudFront アクセスログと同じソースから取得されます。毎時間または毎日のデータポイントを使用して、過去 60 日間の指定した日付範囲のグラフを表示できます。通常、最近 1 時間前だけが CloudFront 受信したリクエストに関するデータを表示できますが、データが 24 時間遅れることがあります。

Note

キャッシュ統計を表示するために、アクセスログを有効にする必要はありません。

CloudFront キャッシュ統計を表示するには

1. にサインインAWS Management Consoleし、 で CloudFront コンソールを開きます<https://console.aws.amazon.com/cloudfront/v4/home>。
 2. ナビゲーションペインで、[Cache Statistics] をクリックします。
 3. [CloudFront Cache Statistics Reports] ペインの [Start Date] と [End Date] で、キャッシュ統計のグラフを表示する日付範囲を選択します。使用できる範囲は、[Granularity] で選択した値によって決まります。
 - Daily – 1 日につき 1 つのデータポイントを使用してグラフを表示するには、過去 60 日の中で任意の日付範囲を選択します。
 - Hourly – 1 時間につき 1 つのデータポイントを使用してグラフを表示するには、過去 60 日以内で最大 14 日間の任意の日付範囲を選択します。
- 日付と時刻は協定世界時 (UTC) です。
4. [Granularity] では、グラフに 1 日につき 1 つのデータポイントを表示するか、1 時間につき 1 つのデータポイントを表示するかを指定します。14 日を超える日付範囲を指定した場合、1 時間につき 1 つのデータポイントを指定することはできなくなります。
 5. [Viewer Location] で、ビューワのリクエストが発信された大陸を選択するか、[All Locations] を選択します。キャッシュ統計グラフには、指定された場所から が CloudFront 受信したリクエストのデータが含まれます。
 6. [Distribution] リストでは、使用状況グラフにデータを表示するディストリビューションを選択します。

- 個々のディストリビューション – グラフには、選択した CloudFront ディストリビューションのデータが表示されます。[Distribution] リストには、ディストリビューションのディストリビューション ID と代替ドメイン名 (CNAME) が表示されます (ある場合)。ディストリビューションに代替ドメイン名がない場合、リストにはディストリビューションのオリジンドメイン名が含まれます。
 - All distributions - 現在の AWS アカウントに関連付けられているすべてのディストリビューションのデータが集計されてグラフに表示されます。ただし、削除したディストリビューションは除外されます。
7. [Update] をクリックします。
 8. グラフ内の毎日または毎時間のデータポイントのデータを表示するには、データポイントの上にマウスポインタを移動します。
 9. 転送データを示すグラフの場合、各グラフの Y 軸の単位をギガバイト、メガバイト、キロバイトのいずれかに変更できることに注意してください。

トピック

- [CSV 形式でのデータのダウンロード](#)
- [キャッシュ統計グラフと標準ログ \(アクセスログ\) CloudFrontのデータとの関連](#)

CSV 形式でのデータのダウンロード

キャッシュ統計レポートは CSV 形式でダウンロードできます。このセクションでは、レポートをダウンロードする方法と、レポートの値について説明します。

キャッシュ統計レポートを CSV 形式でダウンロードするには

1. キャッシュ統計レポートを表示しているときに、[CSV] をクリックします。
2. [Opening file name] ダイアログボックスで、ファイルを開くか保存するかを選択します。

レポートに関する情報

レポートの先頭数行には次の情報が含まれます。

バージョン

この CSV ファイルの形式のバージョン。

レポート

レポートの名前。

DistributionID

レポートを実行した対象のディストリビューションの ID。または、すべてのディストリビューションを対象にレポートを実行した場合は ALL。

StartDateUTC

協定世界時 (UTC) によるレポートを実行した日付範囲の開始日。

EndDateUTC

協定世界時 (UTC) によるレポートを実行した日付範囲の終了日。

GeneratedTimeUTC

協定世界時 (UTC) によるレポートを実行した日時。

詳細度

レポートの各行が 1 時間と 1 日のどちらを表すか。

ViewerLocation

ビューワーリクエストが発信された大陸。または、すべての場所についてレポートをダウンロードする場合は ALL。

キャッシュ統計レポートのデータ

レポートには次の値が含まれています。

DistributionID

レポートを実行した対象のディストリビューションの ID。または、すべてのディストリビューションを対象にレポートを実行した場合は ALL。

FriendlyName

ディストリビューションの代替ドメイン名 (CNAME)、(存在する場合)。ディストリビューションに代替ドメイン名がない場合、リストにはディストリビューションのオリジンドメイン名が含まれます。

ViewerLocation

ビューワーリクエストが発信された大陸。または、すべての場所についてレポートをダウンロードする場合は ALL。

TimeBucket

協定世界時 (UTC) によるデータに該当する時間または日付。

RequestCount

すべての HTTP ステータスコード (200、404 など) およびすべてのメソッド (GET、HEAD、POST など) のリクエストの総数。

HitCount

オブジェクトが CloudFront エッジキャッシュから提供されるビューワーリクエストの数。

MissCount

オブジェクトが現在エッジキャッシュに存在せず、オリジンからオブジェクトを取得 CloudFront する必要があるビューワーリクエストの数。

ErrorCount

エラーの原因となったビューワーリクエストの数。はオブジェクトを提供し CloudFront ませんでした。

IncompleteDownloadCount

ビューワーがオブジェクトのダウンロードを開始したが、ダウンロードを終了できなかったビューワーリクエストの数。

HTTP2xx

HTTP ステータスコードが 2xx 値 (成功) であるビューワーリクエストの数。

HTTP3xx

HTTP ステータスコードが 3xx 値 (追加のアクションが必要) であるビューワーリクエストの数。

HTTP4xx

HTTP ステータスコードが 4xx 値 (クライアントエラー) であるビューワーリクエストの数。

HTTP5xx

HTTP ステータスコードが 5xx 値 (サーバーエラー) であるビューワーリクエストの数。

TotalBytes

すべての HTTP メソッドに対するすべてのリクエストに応じて CloudFront によってビューワーに提供される合計バイト数。

BytesFromMisses

リクエストの発生時にエッジキャッシュに存在しなかったオブジェクトのビューワーに提供されたバイト数。この値は、オリジンから CloudFront エッジキャッシュに転送されたバイトの正確な概算です。ただし、エッジキャッシュに既に存在していても、有効期限が切れているオブジェクトのリクエストは除きます。

キャッシュ統計グラフと標準ログ (アクセスログ) CloudFrontのデータとの関連

次の表は、CloudFront コンソールのキャッシュ統計グラフが CloudFront アクセスログの値とどのように対応しているかを示しています。CloudFront アクセスログの詳細については、「」を参照してください [標準ログ \(アクセスログ\) の設定および使用](#)。

Total requests

このグラフには、すべての HTTP ステータスコード (200 または 404 など) およびすべてのメソッド (GET、HEAD、または POST など) のリクエストの総数が表示されます。このグラフに表示されるリクエストの総数は、同じ期間のアクセスログファイルのリクエストの総数と同じです。

Percentage of viewer requests by result type

このグラフには、選択した CloudFront デイストリビューションのビューワーリクエストの合計に対するヒット、ミス、エラーの割合が表示されます。

- hit – オブジェクトが CloudFront エッジキャッシュから提供されるビューワーリクエスト。アクセスログでは、これらのリクエストの `x-edge-response-result-type` の値は Hit です。
- ミス – オブジェクトが現在エッジキャッシュにないため、オリジンからオブジェクトを取得 CloudFront する必要があるビューワーリクエスト。アクセスログでは、これらのリクエストの `x-edge-response-result-type` の値は Miss です。
- エラー — エラーが発生したビューワーリクエスト。オブジェクト CloudFront を提供できませんでした。アクセスログでは、これらのリクエストの `x-edge-response-result-type` の値は Error、LimitExceeded または CapacityExceeded です。

グラフには、エッジキャッシュに存在しても、有効期限が切れているオブジェクトのリフレッシュヒットリクエストは含まれません。アクセスログでは、リフレッシュヒットのリクエストの `x-edge-response-result-type` の値は RefreshHit です。

Bytes transferred to viewers

このグラフには 2 つの値が表示されます。

- 合計バイト数 – すべての HTTP メソッドに対するすべてのリクエストに応じて、CloudFront によってビューワーに提供される合計バイト数。CloudFront アクセスログの合計バイト数は、同じ期間中のすべてのリクエストの `sc-bytes` 列の値の合計です。
- Bytes from misses – リクエストの発生時にエッジキャッシュに存在しなかったオブジェクトのビューワーに提供されたバイト数。CloudFront アクセスログでは、ミスからのバイトは、の値がであるリクエストの `sc-bytes` 列の値の合計 `x-edge-result-type` です Miss。この値は、オリジンから CloudFront エッジキャッシュに転送されたバイトの正確な概算です。ただし、エッジキャッシュに既に存在していても、有効期限が切れているオブジェクトのリクエストは除きます。

HTTP ステータスコード

このグラフには HTTP ステータスコードごとのビューワーリクエストが表示されます。

CloudFront アクセスログでは、ステータスコードが `sc-status` 列に表示されます。

- 2xx – 成功したリクエスト。
- 3xx – 追加のアクションが必要です。たとえば、301 (Moved Permanently) は、リクエストされたオブジェクトが異なる場所に移動されていることを意味します。
- 4xx – クライアント側のエラー。たとえば、404 (Not Found) は、クライアントが、検出できないオブジェクトをリクエストしたことを意味します。
- 5xx – オリジンサーバーがリクエストを実行しませんでした。たとえば、503 (Service Unavailable) は、オリジンサーバーが現在利用できないことを意味します。

Percentage of GET requests that didn't finish downloading

このグラフでは、合計リクエストに対して、リクエストされたオブジェクトのダウンロードが終了していない、ビューワーの GET リクエストの割合が表示されます。通常、オブジェクトのダウンロードが完了しないのは、たとえば別のリンクをクリックしたり、ブラウザを閉じたりして、ビューワーによってキャンセルされたときです。CloudFront アクセスログでは、これらのリクエストの `sc-status` 列 200 の値は、`x-edge-result-type` 列の値は Error です。

CloudFront 人気オブジェクトレポート

Amazon CloudFront コンソールでは、過去 60 日間の指定した日付範囲内のディストリビューションで最も人気のある 50 個のオブジェクトのリストを表示できます。

人気オブジェクトレポートのデータは、CloudFront アクセスログと同じソースから取得されます。上位 50 個のオブジェクトの正確な数を取得するために、は、午前 0 時から始まる 10 分間隔ですべてのオブジェクトのリクエストを CloudFront カウントし、次の 24 時間にわたって上位 150 個のオブジェクトの累計を保持します。(CloudFront 上位 150 個のオブジェクトの日次合計も 60 日間保持されます)。リストの最下位に近いオブジェクトは、リストに加わったり、リストからなくなったりするため、これらのオブジェクトの合計は概算です。150 件のオブジェクトのリストの中の上位 50 件のオブジェクトもリスト内で上がったたり下がったりする可能性はありますが、リストから完全になくなることはほとんどないため、通常これらのオブジェクトの合計は信頼できます。

オブジェクトが上位 150 件のオブジェクトのリストから削除され、その 1 日のうちに再びリストに加わった場合、は、そのオブジェクトがリストになかった期間の推定リクエスト数 CloudFront を追加します。この予測は、その期間中にリストの最後にあったオブジェクトから受け取ったリクエストの数に基づいています。オブジェクトが 1 日のうちに上位 50 件に増えた場合、オブジェクトが上位 150 件から外れている間に が CloudFront 受信したリクエスト数の見積もりにより、通常、人気オブジェクトレポートのリクエスト数が、そのオブジェクトのアクセスログに表示されるリクエストの数を超えます。

Note

人気オブジェクトのリストを参照するために、アクセスログを有効にする必要はありません。

ディストリビューションの人気オブジェクトを表示するには

1. にサインインAWS Management Consoleし、 で CloudFront コンソールを開きます<https://console.aws.amazon.com/cloudfront/v4/home>。
2. ナビゲーションペインで、[Popular Objects] をクリックします。
3. [CloudFront Popular Objects Report] ペインの [Start Date] と [End Date] で、人気オブジェクトのリストを表示する日付範囲を選択します。過去 60 日間の任意の日付範囲を選択できます。

日付と時刻は協定世界時 (UTC) です。

4. [Distribution] リストで、人気オブジェクトのリストを表示するディストリビューションを選択します。
5. [Update] をクリックします。

トピック

- [CSV 形式でのデータのダウンロード](#)
- [人気オブジェクトレポートのデータと CloudFront 標準ログ \(アクセスログ\) のデータとの関連](#)

CSV 形式でのデータのダウンロード

人気オブジェクトレポートは CSV 形式でダウンロードできます。このセクションでは、レポートをダウンロードする方法と、レポートの値について説明します。

人気オブジェクトレポートを CSV 形式でダウンロードするには

1. 人気オブジェクトレポートを表示しているときに、[CSV] をクリックします。
2. [Opening file name] ダイアログボックスで、ファイルを開くか保存するかを選択します。

レポートに関する情報

レポートの先頭数行には次の情報が含まれます。

バージョン

この CSV ファイルの形式のバージョン。

レポート

レポートの名前。

DistributionID

レポートを実行した対象のディストリビューションの ID。

StartDateUTC

協定世界時 (UTC) によるレポートを実行した日付範囲の開始日。

EndDateUTC

協定世界時 (UTC) によるレポートを実行した日付範囲の終了日。

GeneratedTimeUTC

協定世界時 (UTC) によるレポートを実行した日時。

人気オブジェクトレポートのデータ

レポートには次の値が含まれています。

DistributionID

レポートを実行した対象のディストリビューションの ID。

FriendlyName

ディストリビューションの代替ドメイン名 (CNAME)、(存在する場合)。ディストリビューションに代替ドメイン名がない場合、リストにはディストリビューションのオリジンドメイン名が含まれます。

オブジェクト

オブジェクトの URL の最後の 500 文字。

RequestCount

このオブジェクトに対するリクエストの総数。

HitCount

オブジェクトが CloudFront エッジキャッシュから提供されるビューワーリクエストの数。

MissCount

オブジェクトが現在エッジキャッシュに存在せず、オリジンからオブジェクトを取得 CloudFront する必要があるビューワーリクエストの数。

HitCountPct

HitCount の値に対する RequestCount の値の割合。

BytesFromMisses

リクエストの発生時にエッジキャッシュにオブジェクトが存在しなかった場合に、このオブジェクトについてビューワーに提供されたバイト数。

TotalBytes

すべての HTTP メソッドに対するすべてのリクエストに応じて、このオブジェクト CloudFront に対してによってビューワーに提供される合計バイト数。

IncompleteDownloadCount

このオブジェクトについて、ビューワーがオブジェクトのダウンロードを開始したが、ダウンロードを終了できなかったビューワーリクエストの数。

HTTP2xx

HTTP ステータスコードが 2xx 値 (成功) であるビューワーリクエストの数。

HTTP3xx

HTTP ステータスコードが 3xx 値 (追加のアクションが必要) であるビューワーリクエストの数。

HTTP4xx

HTTP ステータスコードが 4xx 値 (クライアントエラー) であるビューワーリクエストの数。

HTTP5xx

HTTP ステータスコードが 5xx 値 (サーバーエラー) であるビューワーリクエストの数。

人気オブジェクトレポートのデータと CloudFront 標準ログ (アクセスログ) のデータとの関連

次のリストは、CloudFront コンソールの人気オブジェクトレポートの値が、CloudFront アクセスログの値とどのように対応しているかを示しています。CloudFront アクセスログの詳細については、「」を参照してください [標準ログ \(アクセスログ\) の設定および使用](#)。

URL

オブジェクトへのアクセスにビューワーが使用する URL の末尾 500 文字です。

リクエスト

オブジェクトに対するリクエストの総数。この値は通常、CloudFront アクセスログ内のオブジェクトに対する GET リクエストの数とほぼ一致します。

Hits

オブジェクトが CloudFront エッジキャッシュから提供されたビューワーリクエストの数。アクセスログでは、これらのリクエストの `x-edge-response-result-type` の値は Hit です。

Misses

オブジェクトがエッジキャッシュに存在せず、オリジンからオブジェクトを CloudFront 取得したビューワーリクエストの数。アクセスログでは、これらのリクエストの `x-edge-response-result-type` の値は Miss です。

Hit ratio

[Requests] 列の値に対する、[Hits] 列の値の割合。

Bytes from misses

リクエストの発生時にエッジキャッシュに存在しなかったオブジェクトのビューワーに提供されたバイト数。CloudFront アクセスログでは、ミスからのバイトは、`sc-bytes`列の値の合計`x-edge-result-type`ですMiss。

Total bytes

すべての HTTP メソッドのオブジェクトに対するすべてのリクエストに応じてビューワーに CloudFront 提供された合計バイト数。CloudFront アクセスログの合計バイト数は、同じ期間中のすべてのリクエストの `sc-bytes`列の値の合計です。

Incomplete downloads

リクエストされたオブジェクトのダウンロードが終了しなかったビューワーリクエストの数。通常、ダウンロードが完了しないのは、たとえば別のリンクをクリックしたり、ブラウザを閉じたりして、ビューワーによってキャンセルされたときです。CloudFront アクセスログでは、これらのリクエストの `sc-status`列200の値は、`x-edge-result-type`列の値は `Error` です。

2xx

HTTP ステータスコードが 2xx、Successful であるリクエストの数。CloudFront アクセスログでは、ステータスコードが `sc-status`列に表示されます。

3xx

HTTP ステータスコードが 3xx (Redirection) であるリクエストの数です。3xx のステータスコードは追加のアクションが必要であることを表します。たとえば、301 (Moved Permanently) は、リクエストされたオブジェクトが異なる場所に移動されていることを意味します。

4xx

HTTP ステータスコードが 4xx (Client Error) であるリクエストの数です。4xx のステータスコードはクライアント側でエラーが発生したことを表します。たとえば、404 (Not Found) は、クライアントが、検出できないオブジェクトをリクエストしたことを意味します。

5xx

HTTP ステータスコードが 5xx (Server Error) であるリクエストの数です。5xx のステータスコードはオリジンサーバーでリクエストが実行されなかったことを表します。たとえば、503 (Service Unavailable) は、オリジンサーバーが現在利用できないことを意味します。

CloudFront トップリファラレポート

CloudFront コンソールには、指定したディストリビューションに配布 CloudFront しているオブジェクトに対するほとんどの HTTP および HTTPS リクエストを発信したウェブサイトの 25 個のドメインのリストを表示できます。これらのトップリファラーは、検索エンジン、オブジェクトに直接リンクされた他のウェブサイト、またはユーザー自身のウェブサイトである場合もあります。例えば、`https://example.com/index.html` が 10 個のグラフィックにリンクする場合、`example.com` は 10 個のグラフィックすべてのリファラです。トップリファラレポートは、過去 60 日のあらゆる日付範囲で表示できます。

Note

ユーザーがブラウザのアドレス行に直接 URL を入力した場合、リクエストしたオブジェクトのリファラはありません。

トップリファラレポートのデータは、CloudFront アクセスログと同じソースから取得されます。上位 25 件のリファラの正確な数を取得するために、はすべてのオブジェクトのリクエストを 10 分間隔で CloudFront カウントし、上位 75 件のリファラの累計を保持します。リストの最下位に近いリファラは、リストに加わったり、リストからなくなったりするため、これらのリファラの合計は概算です。75 件のリファラのリストの中の上位 25 件のリファラもリスト内で上がったたり下がったりする可能性はありますが、リストから完全になくなることはほとんどないため、通常これらのリファラの合計は信頼できます。

Note

トップリファラのリストを参照するために、アクセスログを有効にする必要はありません。

ディストリビューションのリファラを表示するには

1. にサインインAWS Management Consoleし、 で CloudFront コンソールを開きます<https://console.aws.amazon.com/cloudfront/v4/home>。
2. ナビゲーションペインで、[Top Referrers] をクリックします。
3. [CloudFront Top Referrers Report] ペインの [Start Date] と [End Date] で、トップリファラのリストを表示する日付範囲を選択します。

日付と時刻は協定世界時 (UTC) です。

4. [Distribution] リストで、トップリファラのリストを表示するディストリビューションを選択します。
5. [Update] をクリックします。

トピック

- [CSV 形式でのデータのダウンロード](#)
- [トップリファラレポートのデータと CloudFront 標準ログ \(アクセスログ\) のデータとの関連](#)

CSV 形式でのデータのダウンロード

トップリファラレポートは CSV 形式でダウンロードできます。このセクションでは、レポートをダウンロードする方法と、レポートの値について説明します。

トップリファラレポートを CSV 形式でダウンロードするには

1. トップリファラレポートを表示しているときに、[CSV] をクリックします。
2. [Opening file name] ダイアログボックスで、ファイルを開くか保存するかを選択します。

レポートに関する情報

レポートの先頭数行には次の情報が含まれます。

バージョン

この CSV ファイルの形式のバージョン。

レポート

レポートの名前。

DistributionID

レポートを実行した対象のディストリビューションの ID。または、すべてのディストリビューションを対象にレポートを実行した場合は ALL。

StartDateUTC

協定世界時 (UTC) によるレポートを実行した日付範囲の開始日。

EndDateUTC

協定世界時 (UTC) によるレポートを実行した日付範囲の終了日。

GeneratedTimeUTC

協定世界時 (UTC) によるレポートを実行した日時。

トップリファラレポートのデータ

レポートには次の値が含まれています。

DistributionID

レポートを実行した対象のディストリビューションの ID。または、すべてのディストリビューションを対象にレポートを実行した場合は ALL。

FriendlyName

ディストリビューションの代替ドメイン名 (CNAME)、(存在する場合)。ディストリビューションに代替ドメイン名がない場合、リストにはディストリビューションのオリジンドメイン名が含まれます。

Referrer

リファラのドメイン名。

RequestCount

[Referrer] 列のドメイン名からのリクエストの総数。

RequestsPct

指定した期間のリクエストの総数に対してリファラによって送信されたリクエストの数の割合。

トップリファラレポートのデータと CloudFront 標準ログ (アクセスログ) のデータとの関連

次のリストは、CloudFront コンソールのトップリファラレポートの値が CloudFront アクセスログの値とどのように対応しているかを示しています。CloudFront アクセスログの詳細については、「」を参照してください [標準ログ \(アクセスログ\) の設定および使用](#)。

Referrer

リファラのドメイン名。アクセスログでは、リファラは cs(Referer) 列に表示されます。

Request count

[Referrer] 列のドメイン名からのリクエストの総数。この値は通常、CloudFront アクセスログのリファラからのGETリクエストの数とほぼ一致します。

リクエスト %

指定した期間のリクエストの総数に対してリファラによって送信されたリクエストの数の割合。リファラが 25 個以上あると、[request count] (リクエスト数) 列に指定した期間のすべてのリクエストを含めることができないため、このテーブルのデータに基づいて [Request %] (リクエスト%) を計算することはできません。

CloudFront 使用状況レポート

Amazon CloudFront コンソールでは、CloudFront 使用状況レポートデータのサブセットに基づく使用状況をグラフィカルに表示できます。毎時間または毎日のデータポイントを使用して、過去 60 日間の指定した日付範囲のグラフを表示できます。通常、最近 4 時間前が CloudFront 受信したリクエストに関するデータを表示できますが、データが 24 時間遅れることがあります。

詳細については、「[使用状況グラフと CloudFront 使用状況レポートのデータとの関連](#)」を参照してください。

CloudFront 使用状況グラフを表示するには

1. にサインインAWS Management Consoleし、 で CloudFront コンソールを開きます<https://console.aws.amazon.com/cloudfront/v4/home>。
2. [navigation] ペインで、[Usage Reports] をクリックします。
3. [CloudFront Usage Reports] ペインの [Start Date] と [End Date] で、使用状況グラフを表示する日付範囲を選択します。使用できる範囲は、[Granularity] で選択した値によって決まります。
 - Daily — 1 日につき 1 つのデータポイントを使用してグラフを表示するには、過去 60 日の中で任意の日付範囲を選択します。
 - Hourly — 1 時間につき 1 つのデータポイントを使用してグラフを表示するには、過去 60 日以内で最大 14 日間の任意の日付範囲を選択します。

日付と時刻は協定世界時 (UTC) です。

4. [Granularity] では、グラフに 1 日につき 1 つのデータポイントを表示するか、1 時間につき 1 つのデータポイントを表示するかを指定します。14 日を超える日付範囲を指定した場合、1 時間につき 1 つのデータポイントを指定することはできなくなります。
5. 請求リージョンで、表示するデータがある CloudFront 請求リージョンを選択するか、すべてのリージョンを選択します。使用状況グラフには、指定されたリージョンのエッジロケーションで CloudFront 処理するリクエストのデータが含まれます。がリクエスト CloudFront を処理するリージョンは、ユーザーの場所に対応している場合もありますが、そうでない場合もあります。

ディストリビューションの価格クラスに含まれるリージョンのみを選択してください。それ以外の場合、おそらく使用状況グラフにはデータが含まれません。例えば、ディストリビューションに料金クラス 200 を選択した場合、南米およびオーストラリアの請求リージョンは含まれません。したがって、CloudFront は通常、これらのリージョンからのリクエストを処理しません。価格クラスの詳細については、「[CloudFront ディストリビューションの価格クラスの選択](#)」を参照してください。

6. [Distribution] リストでは、使用状況グラフにデータを表示するディストリビューションを選択します。
 - 個々のディストリビューション — グラフには、選択した CloudFront ディストリビューションのデータが表示されます。[Distribution] リストには、ディストリビューションのディストリビューション ID と代替ドメイン名 (CNAME) が表示されます (ある場合)。ディストリビューションに代替ドメイン名がない場合、リストにはディストリビューションのオリジンドメイン名が含まれます。
 - All distributions (excludes deleted) - 現在の AWS アカウントに関連付けられているすべてのディストリビューションのデータが集計されてグラフに表示されます。ただし、削除したディストリビューションは除外されます。
 - All Deleted Distributions - 現在の AWS アカウントに関連付けられていて過去 60 日間に削除されたすべてのディストリビューションのデータが集計されてグラフに表示されます。
7. [Update Graphs] をクリックします。
8. グラフ内の毎日または毎時間のデータポイントのデータを表示するには、データポイントの上にマウスポインタを移動します。
9. 転送データを示すグラフの場合、各グラフの Y 軸の単位をギガバイト、メガバイト、キロバイトのいずれかに変更できることに注意してください。

トピック

- [CSV 形式でのデータのダウンロード](#)
- [使用状況グラフと CloudFront 使用状況レポートのデータとの関連](#)

CSV 形式でのデータのダウンロード

使用状況レポートは CSV 形式でダウンロードできます。このセクションでは、レポートをダウンロードする方法と、レポートの値について説明します。

使用状況レポートを CSV 形式でダウンロードするには

1. 使用状況レポートを表示しているときに、[CSV] をクリックします。
2. [Opening file name] ダイアログボックスで、ファイルを開くか保存するかを選択します。

レポートに関する情報

レポートの先頭数行には次の情報が含まれます。

バージョン

この CSV ファイルの形式のバージョン。

レポート

レポートの名前。

DistributionID

レポートを実行した対象のディストリビューションの ID。すべてのディストリビューションを対象にレポートを実行した場合は ALL。削除したディストリビューションを対象にレポートを実行した場合は ALL_DELETED。

StartDateUTC

協定世界時 (UTC) によるレポートを実行した日付範囲の開始日。

EndDateUTC

協定世界時 (UTC) によるレポートを実行した日付範囲の終了日。

GeneratedTimeUTC

協定世界時 (UTC) によるレポートを実行した日時。

詳細度

レポートの各行が 1 時間と 1 日のどちらを表すか。

BillingRegion

ビューワーリクエストが発信された大陸。または、すべての請求リージョンについてレポートをダウンロードする場合は ALL。

使用状況レポートのデータ

レポートには次の値が含まれています。

DistributionID

レポートを実行した対象のディストリビューションの ID。すべてのディストリビューションを対象にレポートを実行した場合は ALL。削除したディストリビューションを対象にレポートを実行した場合は ALL_DELETED。

FriendlyName

ディストリビューションの代替ドメイン名 (CNAME)、(存在する場合)。ディストリビューションに代替ドメイン名がない場合、リストにはディストリビューションのオリジンドメイン名が含まれます。

BillingRegion

レポートを実行した CloudFront 請求リージョン、または ALL。

TimeBucket

協定世界時 (UTC) によるデータに該当する時間または日付。

HTTP

指定した CloudFront ディストリビューションの各時間間隔で、選択したリージョンのエッジロケーションからに CloudFront 応答した HTTP リクエストの数。値には以下のものが含まれます。

- ガユーザーにデータ CloudFront を転送する GETおよび HEADリクエストの数
- ガデータをオリジン CloudFront に転送する、DELETEOPTIONSPATCH、、、POSTおよび PUTリクエストの数

HTTPS

指定した CloudFront デистриビューションの各時間間隔で、選択したリージョンのエッジロケーションから CloudFront 応答した HTTPS リクエストの数。値には以下のものが含まれます。

- がユーザーにデータ CloudFront を転送する GET および HEAD リクエストの数
- がデータをオリジン CloudFront に転送する、DELETE、OPTIONS、PATCH、POST および PUT リクエストの数

HTTPBytes

指定した CloudFront デистриビューションの期間中に、選択した請求リージョンの CloudFront エッジロケーションから HTTP 経由で転送されたデータの合計量。値には以下のものが含まれます。

- GET および HEAD リクエストに回答して CloudFront からユーザーに転送されるデータ
- DELETE、OPTIONS、PATCH、POST および PUT リクエストのために から CloudFront オリジンに転送されるデータ
- DELETE、OPTIONS、PATCH、POST、PUT リクエストへのレスポンスで から CloudFront ユーザーに転送されるデータ

HTTPSBytes

指定した CloudFront デистриビューションの期間中に、選択した請求リージョンの CloudFront エッジロケーションから HTTPS 経由で転送されたデータの合計量。値には以下のものが含まれます。

- GET および HEAD リクエストに回答して CloudFront からユーザーに転送されるデータ
- DELETE、OPTIONS、PATCH、POST および PUT リクエストのために から CloudFront オリジンに転送されるデータ
- DELETE、OPTIONS、PATCH、POST、PUT リクエストへのレスポンスで から CloudFront ユーザーに転送されるデータ

BytesIn

指定された CloudFront デистриビューションの各時間間隔で、選択したリージョンの DELETE、OPTIONS、PATCH、POST、および PUT リクエストについて から CloudFront オリジンに転送されたデータの合計量。

BytesOut

指定した CloudFront デистриビューションの各時間間隔で CloudFront、HTTP および HTTPS 経由で から選択したリージョンのユーザーに転送されたデータの合計量。値には以下のものが含まれます。

- GET および HEAD リクエストに回答して CloudFront からユーザーに転送されるデータ
- 、DELETE、 、OPTIONS、PATCHPOST および PUT リクエストに回答して から CloudFront ユーザーに転送されるデータ

使用状況グラフと CloudFront 使用状況レポートのデータとの関連

次のリストは、CloudFront コンソールの使用グラフが、CloudFront 使用状況レポートの Usage Type 列の値にどのように対応しているかを示しています。

トピック

- [リクエストの数](#)
- [プロトコルごとのデータ転送](#)
- [デистриビューションごとのデータ転送](#)

リクエストの数

このグラフには、指定した CloudFront デистриビューションの各時間間隔で、 が選択したリージョンのエッジロケーションから CloudFront 応答するリクエストの合計数が、プロトコル (HTTP または HTTPS) とタイプ (静的、動的、またはプロキシ) で区切られて表示されます。

Number of HTTP requests

- *region*-Requests-HTTP-Static: TTL \geq 3600 秒のオブジェクトのために供給された HTTP GET および HEAD リクエストの数
- *region*-Requests-HTTP-Dynamic: TTL $<$ 3600 秒のオブジェクトのために供給された HTTP GET および HEAD リクエストの数
- *region* -Requests-HTTP-Proxy: がオリジン CloudFront に転送する HTTP DELETE、OPTIONS、PATCH、POST、および PUT リクエストの数

Number of HTTPS requests

- *region*-Requests-HTTPS-Static: TTL \geq 3600 秒のオブジェクトのために供給された HTTPS GET および HEAD リクエストの数

- **region**-Requests-HTTPS-Dynamic: TTL < 3600 秒のオブジェクトのために供給された HTTPS GET および HEAD リクエストの数
- **region** -Requests-HTTPS-Proxy: がオリジン CloudFront に転送する HTTPS DELETE、OPTIONS、PATCH、POST、および PUT リクエストの数

プロトコルごとのデータ転送

このグラフには、指定した CloudFront デイストリビューションの各時間間隔で、選択したリージョンの CloudFront エッジロケーションから転送されたデータの合計が、プロトコル (HTTP または HTTPS)、タイプ (静的、動的、プロキシ)、および送信先 (ユーザーまたはオリジン) で区切られて表示されます。

Data transferred over HTTP

- **region**-Out-Bytes-HTTP-Static: TTL \geq 3600 秒のオブジェクトのために HTTP 経由で供給されたバイト数
- **region**-Out-Bytes-HTTP-Dynamic: TTL < 3600 秒のオブジェクトのために HTTP 経由で供給されたバイト数
- **region** -Out-Bytes-HTTP-Proxy: 、DELETE、 、PATCH、POST および PUT リクエストへのレスポンスとして HTTP 経由で OPTIONS からビューワー CloudFront に返されたバイト数
- **region** -Out-OBytes -HTTP-Proxy: 、DELETE、 、PATCH、POST および PUT リクエストへのレスポンスのために CloudFront エッジロケーションからオリジンに HTTP OPTIONS 経由で転送された合計バイト数

Data transferred over HTTPS

- **region**-Out-Bytes-HTTPS-Static: TTL \geq 3600 秒のオブジェクトのために HTTPS 経由で供給されたバイト数
- **region**-Out-Bytes-HTTPS-Dynamic: TTL < 3600 秒のオブジェクトのために HTTPS 経由で供給されたバイト数
- **region** -Out-Bytes-HTTPS-Proxy: 、DELETE、 、PATCH、POST および PUT リクエストへの応答として HTTPS 経由で OPTIONS からビューワー CloudFront に返されたバイト数
- **region** -Out-OBytes -HTTPS-Proxy: 、DELETE、 、PATCH、POST および PUT リクエストへのレスポンスのために CloudFront エッジロケーションからオリジンに HTTPS OPTIONS 経由で転送された合計バイト数

ディストリビューションごとのデータ転送

このグラフには、指定した CloudFront ディストリビューションの各時間間隔で、選択したリージョンの CloudFront エッジロケーションから転送されたデータの合計が、送信先 (ユーザーまたはオリジン)、プロトコル (HTTP または HTTPS)、タイプ (静的、動的、プロキシ) で区切られて表示されます。

からユーザー CloudFront に転送されるデータ

- *region*-Out-Bytes-HTTP-Static: TTL \geq 3600 秒のオブジェクトのために HTTP 経由で供給されたバイト数
- *region*-Out-Bytes-HTTPS-Static: TTL \geq 3600 秒のオブジェクトのために HTTPS 経由で供給されたバイト数
- *region*-Out-Bytes-HTTP-Dynamic: TTL $<$ 3600 秒のオブジェクトのために HTTP 経由で供給されたバイト数
- *region*-Out-Bytes-HTTPS-Dynamic: TTL $<$ 3600 秒のオブジェクトのために HTTPS 経由で供給されたバイト数
- *region* -Out-Bytes-HTTP-Proxy: 、DELETE、 、PATCH、POSTおよび PUTリクエストへのレスポンスとして HTTP 経由で OPTIONSからビューワー CloudFront に返されたバイト数
- *region* -Out-Bytes-HTTPS-Proxy: 、DELETE、 、PATCH、POSTおよび PUTリクエストへの応答として HTTPS OPTIONS経由で からビューワー CloudFront に返されたバイト数

からオリジン CloudFront に転送されるデータ

- *region* -Out-OBytes -HTTP-Proxy: 、DELETE、 、PATCH、POSTおよび PUTリクエストへのレスポンスのために CloudFront エッジロケーションからオリジンに HTTP OPTIONS経由で転送された合計バイト数
- *region* -Out-OBytes -HTTPS-Proxy: 、DELETE、 、PATCH、POSTおよび PUTリクエストへのレスポンスのために CloudFront エッジロケーションからオリジンに HTTPS OPTIONS経由で転送された合計バイト数

CloudFront ビューワーレポート

CloudFront コンソールには、物理デバイス (デスクトップコンピュータ、モバイルデバイス) と、コンテンツにアクセスしているビューワー (通常はウェブブラウザ) に関する 4 つのレポートを表示できます。

- デバイス – コンテンツにアクセスするユーザーが最も頻繁に使用するデバイスのタイプ (デスクトップやモバイルなど)。

- ブラウザ – Chrome や Firefox など、コンテンツにアクセスするときに、ユーザーが最も頻繁に使用するブラウザの名前 (または名前とバージョン)。このレポートには、上位 10 件のブラウザが表示されます。
- オペレーティングシステム – Linux、macOS、Windows など、コンテンツにアクセスするときにビューワーが最も頻繁に実行するオペレーティングシステムの名前 (または名前とバージョン)。このレポートには、上位 10 件のオペレーティングシステムが表示されます。
- ロケーション – コンテンツに最も頻繁にアクセスするビューワーの場所 (国、または米国の州/準州)。このレポートには、上位 50 件の国、または米国の州/準州が表示されます。

4 つのビューワーレポートはすべて、過去 60 日の中のどの日付範囲でも表示できます。ロケーションレポートでは、過去 60 日の中で最大 14 日間の日付範囲の間、1 時間ごとにデータポイントのあるレポートを表示することもできます。

Note

ビューワーのグラフおよびレポートを参照するために、アクセスログを有効にする必要はありません。

トピック

- [ビューワーのグラフおよびレポートの表示](#)
- [CSV 形式でのデータのダウンロード](#)
- [ロケーションレポートのデータと CloudFront 標準ログ \(アクセスログ\) のデータとの関連](#)

ビューワーのグラフおよびレポートの表示

CloudFront ビューワーのグラフとレポートを表示するには、次の手順を実行します。

CloudFront ビューワーのグラフとレポートを表示するには

1. にサインインAWS Management Consoleし、 で CloudFront コンソールを開きます<https://console.aws.amazon.com/cloudfront/v4/home>。
2. ナビゲーションペインで [Viewers] をクリックします。
3. [CloudFront Viewers] ペインの [Start Date] と [End Date] で、ビューワーのグラフおよびレポートを表示する日付範囲を選択します。

ロケーショングラフで使用できる範囲は、[Granularity] で選択した値によって異なります。

- Daily – 1 日につき 1 つのデータポイントを使用してグラフを表示するには、過去 60 日の中で任意の日付範囲を選択します。
- Hourly – 1 時間につき 1 つのデータポイントを使用してグラフを表示するには、過去 60 日以内で最大 14 日間の任意の日付範囲を選択します。

日付と時刻は協定世界時 (UTC) です。

4. (ブラウザとオペレーティングシステムのグラフのみ) [Grouping] で、ブラウザおよびオペレーティングシステムを名前 (Chrome、Firefox) ごと、または名前とバージョン (Chrome 40.0、Firefox 35.0) ごとにグループ化するかどうかを指定します。
5. (ロケーショングラフのみ) [Granularity] で、グラフに 1 日につき 1 つのデータポイントを表示するか、1 時間につき 1 つのデータポイントを表示するかを指定します。14 日を超える日付範囲を指定した場合、1 時間につき 1 つのデータポイントを指定することはできなくなります。
6. (ロケーショングラフのみ) [Details] で、上位のロケーションを国ごとに表示するか、米国の州ごとに表示するかを指定します。
7. [Distribution] リストでは、使用状況グラフにデータを表示するディストリビューションを選択します。
 - 個々のディストリビューション – グラフには、選択した CloudFront ディストリビューションのデータが表示されます。[Distribution] リストには、ディストリビューションのディストリビューション ID と代替ドメイン名 (CNAME) が表示されます (ある場合)。ディストリビューションに代替ドメイン名がない場合、リストにはディストリビューションのオリジンドメイン名が含まれます。
 - All distributions (excludes deleted) - 現在の AWS アカウントに関連付けられているすべてのディストリビューションのデータが集計されてグラフに表示されます。ただし、削除したディストリビューションは除外されます。
8. [Update] をクリックします。
9. グラフ内の毎日または毎時間のデータポイントのデータを表示するには、データポイントの上にマウスポインタを移動します。

CSV 形式でのデータのダウンロード

各ビューワーレポートは CSV 形式でダウンロードできます。このセクションでは、レポートをダウンロードする方法と、レポートの値について説明します。

ビューワーレポートを CSV 形式でダウンロードするには

1. ビューワーのレポートを表示しているときに、[CSV] をクリックします。
2. ダウンロードするデータ ([Devices] や [Devices Trends] など) を選択します。
3. [Opening file name] ダイアログボックスで、ファイルを開くか保存するかを選択します。

トピック

- [レポートに関する情報](#)
- [デバイスレポート](#)
- [デバイストレンドレポート](#)
- [ブラウザレポート](#)
- [ブラウザトレンドレポート](#)
- [オペレーティングシステムレポート](#)
- [オペレーティングシステムトレンドレポート](#)
- [ロケーションレポート](#)
- [ロケーショントレンドレポート](#)

レポートに関する情報

各レポートの先頭数行には次の情報が含まれます。

バージョン

この CSV ファイルの形式のバージョン。

レポート

レポートの名前。

DistributionID

レポートを実行した対象のディストリビューションの ID。または、すべてのディストリビューションを対象にレポートを実行した場合は ALL。

StartDateUTC

協定世界時 (UTC) によるレポートを実行した日付範囲の開始日。

EndDateUTC

協定世界時 (UTC) によるレポートを実行した日付範囲の終了日。

GeneratedTimeUTC

協定世界時 (UTC) によるレポートを実行した日時。

Grouping (ブラウザとオペレーティングシステムのレポートのみ)

データをブラウザやオペレーティングシステムの名前によってグループ化するか、または名前とバージョンによってグループ化するか。

詳細度

レポートの各行が 1 時間と 1 日のどちらを表すか。

Details (ロケーションレポートのみ)

リクエストを国別にリストするか、米国の州別にリストするか。

デバイスレポート

レポートには次の値が含まれています。

DistributionID

レポートを実行した対象のディストリビューションの ID。または、すべてのディストリビューションを対象にレポートを実行した場合は ALL。

FriendlyName

ディストリビューションの代替ドメイン名 (CNAME)、(存在する場合)。ディストリビューションに代替ドメイン名がない場合、リストにはディストリビューションのオリジンドメイン名が含まれます。

リクエスト

が各タイプのデバイスから CloudFront 受信したリクエストの数。

RequestsPct

が各タイプのデバイスから CloudFront 受信したリクエストの数を、すべてのデバイスから CloudFront 受信したリクエストの合計数に対する割合で示します。

デバイストレンドレポート

レポートには次の値が含まれています。

DistributionID

レポートを実行した対象のディストリビューションの ID。または、すべてのディストリビューションを対象にレポートを実行した場合は ALL。

FriendlyName

ディストリビューションの代替ドメイン名 (CNAME)、(存在する場合)。ディストリビューションに代替ドメイン名がない場合、リストにはディストリビューションのオリジンドメイン名が含まれます。

TimeBucket

協定世界時 (UTC) によるデータに該当する時間または日付。

Desktop

期間中にデスクトップコンピュータから CloudFront 受信したリクエストの数。

モバイル

期間中にモバイルデバイスから CloudFront 受信したリクエストの数。モバイルデバイスには、タブレットと携帯電話の両方が含まれる場合があります。ガリクエストがモバイルデバイスまたはタブレットのいずれから送信されたかを特定 CloudFront できない場合、Mobile列にカウントされます。

Smart-TV

期間中にスマートTVsから CloudFront 受信したリクエストの数。

Tablet

期間中に がタブレットから CloudFront 受け取ったリクエストの数。ガリクエストがモバイルデバイスまたはタブレットのいずれから送信されたかを特定 CloudFront できない場合、Mobile列にカウントされます。

不明

User-Agent HTTP ヘッダーの値が標準デバイスタイプのいずれか (Desktop や Mobile など) に関連付けられていなかったリクエスト。

Empty

期間中に HTTP User-Agent ヘッダーに値が含まれなかった、 が CloudFront 受信したリクエストの数。

ブラウザレポート

レポートには次の値が含まれています。

DistributionID

レポートを実行した対象のディストリビューションの ID。または、すべてのディストリビューションを対象にレポートを実行した場合は ALL。

FriendlyName

ディストリビューションの代替ドメイン名 (CNAME)、(存在する場合)。ディストリビューションに代替ドメイン名がない場合、リストにはディストリビューションのオリジンドメイン名が含まれます。

グループ

の値に応じて、 がリクエスト CloudFront を受け取ったブラウザまたはブラウザとバージョン Grouping。ブラウザ名に加えて、次の値が含まれる場合があります。

- Bot/Crawler – 主にコンテンツのインデックスを作成する検索エンジンからのリクエスト。
- Empty – User-Agent HTTP ヘッダーの値が空であったリクエスト。
- Other – CloudFront は識別したが、最も人気のないブラウザです。Bot/Crawler、Empty、Unknown が最初の 9 個の値に表示されない場合、Other にも含まれています。
- Unknown – User-Agent HTTP ヘッダーの値が標準的なブラウザに関連付けられていなかったリクエスト。このカテゴリのほとんどのリクエストは、カスタムアプリケーションまたはスクリプトからのリクエストです。

リクエスト

が各タイプのブラウザから CloudFront 受信したリクエストの数。

RequestsPct

各タイプのブラウザから が CloudFront 受信したリクエストの数を、その期間中に が CloudFront 受信したリクエストの総数に対する割合で示します。

ブラウザトレンドレポート

レポートには次の値が含まれています。

DistributionID

レポートを実行した対象のディストリビューションの ID。または、すべてのディストリビューションを対象にレポートを実行した場合は ALL。

FriendlyName

ディストリビューションの代替ドメイン名 (CNAME)、(存在する場合)。ディストリビューションに代替ドメイン名がない場合、リストにはディストリビューションのオリジンドメイン名が含まれます。

TimeBucket

協定世界時 (UTC) によるデータに該当する時間または日付。

(ブラウザ)

レポートの残りの列には、Grouping の値に応じて、ブラウザまたはブラウザとバージョンがリストされます。ブラウザ名に加えて、次の値が含まれる場合があります。

- Bot/Crawler – 主にコンテンツのインデックスを作成する検索エンジンからのリクエスト。
- Empty – User-Agent HTTP ヘッダーの値が空であったリクエスト。
- Other – CloudFront は識別したが、最も人気のないブラウザです。Bot/Crawler、Empty、Unknown が最初の 9 個の値に表示されない場合、Other にも含まれています。
- Unknown – User-Agent HTTP ヘッダーの値が標準的なブラウザに関連付けられていなかったリクエスト。このカテゴリのほとんどのリクエストは、カスタムアプリケーションまたはスク립トからのリクエストです。

オペレーティングシステムレポート

レポートには次の値が含まれています。

DistributionID

レポートを実行した対象のディストリビューションの ID。または、すべてのディストリビューションを対象にレポートを実行した場合は ALL。

FriendlyName

ディストリビューションの代替ドメイン名 (CNAME)、(存在する場合)。ディストリビューションに代替ドメイン名がない場合、リストにはディストリビューションのオリジンドメイン名が含まれます。

グループ

の値に応じて、 がリクエスト CloudFront を受け取ったオペレーティングシステムまたはオペレーティングシステムとバージョンGrouping。オペレーティングシステム名に加えて、次の値が含まれる場合があります。

- Bot/Crawler – 主にコンテンツのインデックスを作成する検索エンジンからのリクエスト。
- Empty – User-Agent HTTP ヘッダーの値が空であったリクエスト。
- Other – が最も人気のない CloudFront を識別したオペレーティングシステム。Bot/Crawler、Empty、Unknown が最初の 9 個の値に表示されない場合、Other にも含まれています。
- Unknown – User-Agent HTTP ヘッダーの値が標準的なブラウザに関連付けられていなかったリクエスト。このカテゴリのほとんどのリクエストは、カスタムアプリケーションまたはスク립トからのリクエストです。

リクエスト

が各タイプのオペレーティングシステムから CloudFront 受信したリクエストの数。

RequestsPct

各タイプのオペレーティングシステムから が CloudFront 受信したリクエストの数を、期間中に が CloudFront 受信したリクエストの合計数に対する割合で示します。

オペレーティングシステムトレンドレポート

レポートには次の値が含まれています。

DistributionID

レポートを実行した対象のディストリビューションの ID。または、すべてのディストリビューションを対象にレポートを実行した場合は ALL。

FriendlyName

ディストリビューションの代替ドメイン名 (CNAME)、(存在する場合)。ディストリビューションに代替ドメイン名がない場合、リストにはディストリビューションのオリジンドメイン名が含まれます。

TimeBucket

協定世界時 (UTC) によるデータに該当する時間または日付。

(オペレーティングシステム)

レポートの残りの列には、Grouping の値に応じて、オペレーティングシステムまたはオペレーティングシステムとバージョンがリストされます。オペレーティングシステム名に加えて、次の値が含まれる場合があります。

- Bot/Crawler – 主にコンテンツのインデックスを作成する検索エンジンからのリクエスト。
- Empty – User-Agent HTTP ヘッダーの値が空であったリクエスト。
- Other – が最も人気のない CloudFront を特定したオペレーティングシステム。Bot/Crawler、Empty、Unknown が最初の 9 個の値に表示されない場合、Other にも含まれています。
- Unknown – User-Agent HTTP ヘッダーでオペレーティングシステムが指定されていないリクエスト。

ロケーションレポート

レポートには次の値が含まれています。

DistributionID

レポートを実行した対象のディストリビューションの ID。または、すべてのディストリビューションを対象にレポートを実行した場合は ALL。

FriendlyName

ディストリビューションの代替ドメイン名 (CNAME)、(存在する場合)。ディストリビューションに代替ドメイン名がない場合、リストにはディストリビューションのオリジンドメイン名が含まれます。

LocationCode

がリクエスト CloudFront を受け取った場所の略語。表示される可能性がある値の詳細については、「[ロケーションレポートのデータとCloudFront 標準ログ \(アクセスログ\) のデータとの関連](#)」のロケーションの説明を参照してください。

LocationName

リクエストを CloudFront 受信した場所の名前。

リクエスト

が各場所から CloudFront 受け取ったリクエストの数。

RequestsPct

各ロケーションから が CloudFront 受信したリクエストの数を、その期間中にすべてのロケーションから CloudFront 受信したリクエストの合計数に対する割合で示します。

TotalBytes

指定されたディストリビューションおよび期間に、この国または州のビューワーに CloudFront 提供されたバイト数。

ロケーショントレンドレポート

レポートには次の値が含まれています。

DistributionID

レポートを実行した対象のディストリビューションの ID。または、すべてのディストリビューションを対象にレポートを実行した場合は ALL。

FriendlyName

ディストリビューションの代替ドメイン名 (CNAME)、(存在する場合)。ディストリビューションに代替ドメイン名がない場合、リストにはディストリビューションのオリジンドメイン名が含まれます。

TimeBucket

協定世界時 (UTC) によるデータに該当する時間または日付。

(ロケーション)

レポートの残りの列には、 がリクエスト CloudFront を受け取った場所が一覧表示されます。表示される可能性がある値の詳細については、「[ロケーションレポートのデータとCloudFront 標準ログ \(アクセスログ\) のデータとの関連](#)」のロケーションの説明を参照してください。

ロケーションレポートのデータとCloudFront 標準ログ (アクセスログ) のデータとの関連

次のリストは、 CloudFront コンソールのロケーションレポートのデータが CloudFront アクセスログの値とどのように対応するかを示しています。 CloudFront アクセスログの詳細については、「」を参照してください[標準ログ \(アクセスログ\) の設定および使用](#)。

ロケーション

ビューワーがいる国または米国の州。アクセスログの c-ip 列には、ビューワーが実行中のデバイスの IP アドレスが含まれています。位置情報データを使用して、IP アドレスに基づくデバイスの地理的場所を識別します。

[Locations] レポートを国ごとに表示している場合、国のリストは「[ISO 3166-2、国および地方行政区画に対するコード – Part 2: 行政区画コード](#)」を基にしている点に注意してください。国のリストには、以下の追加の値が含まれています。

- Anonymous Proxy (匿名プロキシ) – 匿名のプロキシからのリクエスト。
- Satellite Provider (衛星プロバイダー) – 複数の国にインターネットサービスを提供している衛星プロバイダーからのリクエスト。ユーザーは、不正行為のリスクが高い国にいる可能性があります。
- Europe (Unknown) (欧州 (不明)) – 複数の欧州諸国で使用されているブロックの IP からのリクエスト。リクエスト送信元の国は決定できません。デフォルトとして欧州 (不明) CloudFront を使用します。
- Asia/Pacific (Unknown) (アジアパシフィック (不明)) – アジアパシフィックリージョンの複数の国で使用されているブロックの IP からのリクエスト。リクエスト元の国は決定できません。デフォルトとしてアジア/太平洋 (不明) CloudFront を使用します。

[Locations (所在地)] レポートを米国の州ごとに表示している場合、レポートには以下の米国準州と米軍基地所在地域が含まれることがあります。

Note

CloudFront がユーザーの場所を特定できない場合、ビューワーレポートには場所が不明と表示されます。

Request Count

ビューワーがいる国または米国の州からの、指定したディストリビューションおよび期間のリクエストの総数。一般的にこの値は、CloudFront アクセスログのその国または州の IP アドレスからの GET リクエストの数とほぼ一致します。

リクエスト %

[Details] で選択した値に基づき、次のうちのいずれか。

- Countries – リクエストの総数に対するこの国からのリクエストの割合。
- U.S. States - 米国からのリクエストの総数に対するこの州からのリクエストの割合。

50 以上の国からリクエストがあると、[Request Count] 列に指定した期間のすべてのリクエストを含めることができないため、このテーブルのデータに基づいて [Request %] を計算することはできません。

バイト

指定されたディストリビューションおよび期間に、この国または州のビューワーに CloudFront 提供されたバイト数。この列のデータの表示を KB、MB、または GB に変更するには、列見出しのリンクをクリックします。

Amazon による CloudFront メトリクスのモニタリング CloudWatch

Amazon CloudFront は Amazon と統合 CloudWatch されており、ディストリビューションと [エッジ関数 \(Lambda@Edge と CloudFront Functions の両方\)](#) の運用メトリクスを自動的に発行します。これらのメトリクスの多くは、[CloudFront コンソールの一連のグラフに表示され](#)、CloudFront API または CLI を使用してアクセスすることもできます。これらのメトリクスはすべて、[CloudWatch コンソール](#)、API、CLI CloudWatch を通じて使用できます。CloudFront メトリクスは [CloudWatch クォータ \(以前は制限と呼ばれていました\)](#) にはカウントされず、追加コストも発生しません。

CloudFront デイストリビューションのデフォルトのメトリクスに加えて、追加料金で追加のメトリクスを有効にできます。追加のメトリクスは CloudFront デイストリビューションに適用され、デイストリビューションごとに個別に有効にする必要があります。料金の詳細については、「[the section called “追加 CloudFront メトリクスのコストの見積もり”](#)」を参照してください。

これらのメトリクスを表示すると、問題のトラブルシューティング、追跡、およびデバッグに役立ちます。CloudFront コンソールでこれらのメトリクスを表示するには、「[モニタリング](#)」ページを参照してください。特定の CloudFront デイストリビューションまたはエッジ関数のアクティビティに関するグラフを表示するには、1つを選択し、デイストリビューションメトリクスの表示またはメトリクスの表示を選択します。

これらのメトリクスに基づいてアラームは、CloudFront コンソール、または CloudWatch コンソール、API、または CLI で設定することもできます ([標準 CloudWatch 料金](#)が適用されます)。例えば、5xxErrorRate メトリクスに基づくアラームを設定できます。このメトリクスは、レスポンスの HTTP ステータスコードが 500 から 599 の範囲内にあるすべてのビューワーリクエストの割合 (%) を示します。エラー率が一定時間内に特定の値 (連続した 5 分以内のリクエスト数の 5% など) に達すると、アラームがトリガーされます。アラームの作成時に、アラームの値と時間単位を指定します。詳細については、「[アラームの作成](#)」を参照してください。

Note

CloudFront コンソールで CloudWatch アラームを作成すると、米国東部 (バージニア北部) リージョン () にアラームが作成されます us-east-1。CloudWatch コンソールからアラームを作成する場合は、同じリージョンを使用する必要があります。CloudFront はグローバルサービスであるため、サービスのメトリクスは米国東部 (バージニア北部) に送信されます。

トピック

- [CloudFront およびエッジ関数メトリクスの表示](#)
- [メトリクスのアラームを作成する](#)
- [CSV 形式でのメトリクスデータのダウンロード](#)
- [CloudWatch API を使用したメトリクスの取得](#)

CloudFront およびエッジ関数メトリクスの表示

デイストリビューションと [エッジ関数](#)に関する運用メトリクスは、CloudFront コンソールで表示できます。これらのメトリクスを表示するには、[CloudFront コンソールのモニタリング](#)

[ページ](#)を参照してください。特定の CloudFront デイストリビューションまたはエッジ関数のアクティビティに関するグラフを表示するには、1つを選択し、デイストリビューションメトリクスの表示またはメトリクスの表示を選択します。

トピック

- [デフォルトの CloudFront デイストリビューションメトリクスの表示](#)
- [追加の CloudFront デイストリビューションメトリクスを有効にする](#)
- [Lambda@Edge 関数のデフォルトメトリクスの表示](#)
- [デフォルトの CloudFront Functions メトリクスの表示](#)

デフォルトの CloudFront デイストリビューションメトリクスの表示

以下のデフォルトメトリクスは、すべての CloudFront デイストリビューションに追加コストなしで含まれます。

リクエスト

すべての HTTP メソッド、および HTTP リクエストと HTTPS リクエストの両方について CloudFront、 が受信したビューワーリクエストの合計数。

ダウンロードされたバイト数

GET リクエスト、HEAD リクエスト、および OPTIONS リクエストに対してビューワーがダウンロードしたバイト総数。

アップロードされたバイト数

POST リクエストと PUT リクエストを使用して CloudFront でビューワーがオリジンにアップロードしたバイト総数。

4xx エラー率

レスポンスの HTTP ステータスコードが 4xx であるすべてのビューワーリクエストの割合 (%)。

5xx エラー率

レスポンスの HTTP ステータスコードが 5xx であるすべてのビューワーリクエストの割合 (%)。

合計エラー率

レスポンスの HTTP ステータスコードが 4xx または 5xx であるすべてのビューワーリクエストの割合 (%)。

これらのメトリクスは、コンソールのモニタリングページの各 CloudFront デイストリビューションのグラフに表示されます。 [CloudFront](#) 各グラフでは、総数が 1 分単位で表示されます。グラフを表示するだけでなく、[メトリクスレポートを CSV ファイルとしてダウンロード](#)することもできます。

次の手順を実行してグラフをカスタマイズできます。

- グラフに表示される情報の時間範囲を変更するには、1h (1 時間)、3h (3 時間)、または別の範囲、またはカスタムの範囲を指定します。
- がグラフ内の情報 CloudFront を更新する頻度を変更するには、更新アイコンの横にある下向き矢印を選択し、次に更新レートを選択します。デフォルトの更新間隔は 1 分ですが、10 秒、2 分、または他のオプションを指定できます。

CloudWatch コンソールで CloudFront グラフを表示するには、ダッシュボードに追加を選択します。

追加の CloudFront デイストリビューションメトリクスを有効にする

デフォルトメトリクスに加えて、追加のメトリクスを追加料金で有効にすることができます。料金の詳細については、「[the section called “追加 CloudFront メトリクスのコストの見積もり”](#)」を参照してください。

以下の追加のメトリクスは、デイストリビューションごとに個別に有効にする必要があります。

キャッシュヒットレート

がキャッシュからコンテンツを CloudFront 供給したすべてのキャッシュ可能なリクエストの割合。HTTP POST/PUT リクエストおよびエラーは、キャッシュ可能なリクエストとは見なされません。

オリジンのレイテンシー

がリクエスト CloudFront を受信してから、CloudFront キャッシュではなくオリジンから提供されるリクエストに対して、ネットワーク (ビューワーではなく) へのレスポンスの提供を開始するまでに費やされた合計時間。これは、最初のバイトレイテンシー、またはとも呼ばれます time-to-first-byte。

ステータスコード別のエラー率

レスポンスの HTTP ステータスコードが 4xx 範囲または 5xx 範囲内の特定のコードであるすべてのビューワーリクエストの割合 (%)。このメトリクスは、401、403、404、502、503、および 504 のすべてのエラーコードで使用できます。

追加メトリクスの有効化

CloudFront コンソール、AWS Command Line Interface (AWS CLI)、または CloudFront API を使用して AWS CloudFormation、追加のメトリクスを有効にできます。

Console

追加のメトリクスを有効にするには (コンソール)

1. [サインイン](#) AWS Management Console し、[CloudFront コンソールのモニタリングページ](#)を開きます。
2. 追加のメトリクスを有効にするディストリビューションを選択し、[View distribution metrics] (ディストリビューションメトリクスの表示) を選択します。
3. [Manage additional metrics] (追加のメトリクスの管理) を選択します。
4. [Manage additional metrics] (追加のメトリクスの管理) ウィンドウで、[Enabled] (有効) をオンにします。追加のメトリクスを有効にしたら、[Manage additional metrics] (追加のメトリクスの管理) ウィンドウを閉じることができます。

有効にした追加のメトリクスがグラフに表示されます。各グラフでは、総数が 1 分単位で表示されます。グラフを表示するだけでなく、[メトリクスレポートを CSV ファイルとしてダウンロード](#)することもできます。

次の手順を実行してグラフをカスタマイズできます。

- グラフに表示される情報の時間範囲を変更するには、1h (1 時間)、3h (3 時間)、または別の範囲、またはカスタムの範囲を指定します。
- グラフ内の情報 CloudFront を更新する頻度を変更するには、更新アイコンの横にある下向き矢印を選択し、次に更新レートを選択します。デフォルトの更新間隔は 1 分ですが、10 秒、2 分、または他のオプションを指定できます。

CloudWatch コンソールで CloudFront グラフを表示するには、ダッシュボードに追加を選択します。

AWS CloudFormation

AWS CloudFormation で追加のメトリクスを有効にするには、AWS::`CloudFront::MonitoringSubscription` リソースタイプを使用します。次の例は、追加のメトリクスを有効にするための AWS CloudFormation テンプレート構文を YAML 形式で示しています。

```
Type: AWS::CloudFront::MonitoringSubscription
Properties:
  DistributionId: EDFDVBD6EXAMPLE
  MonitoringSubscription:
    RealtimeMetricsSubscriptionConfig:
      RealtimeMetricsSubscriptionStatus: Enabled
```

CLI

AWS Command Line Interface (AWS CLI) を使用して追加のメトリクスを管理するには、次のいずれかのコマンドを使用します。

ディストリビューション (CLI) の追加メトリクスを有効にするには

- 以下の例のように、`create-monitoring-subscription` コマンドを使用します。*EDFDVBD6EXAMPLE* を追加メトリクスを有効にするディストリビューションの ID と置き換えます。

```
aws cloudfront create-monitoring-subscription --
distribution-id EDFDVBD6EXAMPLE --monitoring-subscription
RealtimeMetricsSubscriptionConfig={RealtimeMetricsSubscriptionStatus=Enabled}
```

ディストリビューション (CLI) の追加メトリクスが有効になっているかどうかを確認するには

- 以下の例のように、`get-monitoring-subscription` コマンドを使用します。*EDFDVBD6EXAMPLE* をチェックしているディストリビューションの ID と置き換えます。

```
aws cloudfront get-monitoring-subscription --distribution-id EDFDVBD6EXAMPLE
```

ディストリビューション (CLI) の追加メトリクスを無効にするには

- 以下の例のように、`delete-monitoring-subscription` コマンドを使用します。*EDFDVBD6EXAMPLE* を追加メトリクスを無効にするディストリビューションの ID と置き換えます。

```
aws cloudfront delete-monitoring-subscription --distribution-id EDFDVBD6EXAMPLE
```

API

CloudFront API を使用して追加のメトリクスを管理するには、次のいずれかの API オペレーションを使用します。

- ディストリビューションの追加メトリクスを有効にするには、[CreateMonitoringSubscription](#) を使用します。
- ディストリビューションで追加のメトリクスが有効になっているかどうかを確認するには、[GetMonitoringSubscription](#) を使用します。
- ディストリビューションの追加メトリクスを無効にするには、[DeleteMonitoringSubscription](#) を使用します。

これらの API コールの詳細については、AWS SDK またはその他 API クライアントの API リファレンスドキュメントを参照してください。

追加 CloudFront メトリクスのコストの見積もり

ディストリビューションの追加メトリクスを有効にすると、は、米国東部 (バージニア北部) リージョン CloudWatch の に最大 8 つのメトリクス CloudFront を送信します。は、メトリクスごとに低い固定レート CloudWatch で課金します。この料金は、メトリクスごとに毎月 1 回のみ請求されます (ディストリビューションごとに最大 8 つのメトリクス)。これは固定レートであるため、CloudFront ディストリビューションが受信または送信するリクエストまたはレスポンスの数に関係なく、コストは変わりません。メトリクスごとの料金については、[Amazon CloudWatch 料金表ページ](#)と[CloudWatch 料金計算ツール](#)を参照してください。API を使用してメトリクスを取得する場合は、追加の CloudWatch API 料金が適用されます。

Lambda@Edge 関数のデフォルトメトリクスの表示

CloudWatch メトリクスを使用して、Lambda@Edge 関数の問題をリアルタイムでモニタリングできます。これらのメトリクスに対する追加料金はありません。

Lambda@Edge 関数を CloudFront ディストリビューションのキャッシュ動作にアタッチすると、Lambda は へのメトリクスの送信 CloudWatch を自動的に開始します。メトリクスはすべての Lambda リージョンで使用できますが、CloudWatch コンソールでメトリクスを表示したり、CloudWatch API からメトリクスデータを取得したりするには、米国東部 (バージニア北部)

リージョン () を使用する必要があります us-east-1。メトリクスグループ名は の形式です。ここで AWS/CloudFront/*distribution-ID* distribution-ID は、Lambda@Edge 関数が関連付けられている CloudFront ディストリビューションの ID です。メトリクスの詳細については、CloudWatch [「Amazon CloudWatch ユーザーガイド」](#) を参照してください。

次のデフォルトのメトリクスは、[CloudFront コンソールのモニタリングページの各 Lambda@Edge 関数のグラフ](#)に表示されます。

- 5xxLambda@Edge のエラー率
- Lambda 実行エラー
- Lambda 無効レスポンス
- Lambda スロットリング

グラフには、呼び出し数、エラー数、スロットル数などが表示されます。各グラフでは、合計が 1 分単位で AWS リージョンごとにグループ化されて表示されます。

調査したいエラーが急増した場合は、問題が発生している関数と AWS リージョンを特定するまで、各関数を選択して AWS リージョン別にログファイルを表示できます。Lambda@Edge エラーのトラブルシューティングの詳細については、以下を参照してください。

- [the section called “障害のタイプを判断する方法”](#)
- [AWS](#) でコンテンツ配信をデバッグする 4 つのステップ

次の手順を実行してグラフをカスタマイズできます。

- グラフに表示される情報の時間範囲を変更するには、1h (1 時間)、3h (3 時間)、または別の範囲、またはカスタムの範囲を指定します。
- がグラフ内の情報 CloudFront を更新する頻度を変更するには、更新アイコンの横にある下向き矢印を選択し、次に更新レートを選択します。デフォルトの更新間隔は 1 分ですが、10 秒、2 分、または他のオプションを指定できます。

CloudWatch コンソールでグラフを表示するには、ダッシュボードに追加を選択します。コンソールで CloudWatch グラフを表示するには、米国東部 (バージニア北部) リージョン (米国東部 1) を使用する必要があります。

デフォルトの CloudFront Functions メトリクスの表示

CloudFront 関数は、関数をモニタリング CloudWatch できるように、運用メトリクスを Amazon に送信します。これらのメトリクスを表示すると、問題のトラブルシューティング、追跡、およびデバッグに役立ちます。CloudFront 関数は、次のメトリクスを に発行します CloudWatch。

- 呼び出し (FunctionInvocations) - 指定された期間に関数が起動 (呼び出し) された回数。
- 検証エラー (FunctionValidationErrors) - 指定した期間内に関数によって生成された検証エラーの数。検証エラーは、関数は正常に実行されたが、無効なデータ (無効な [イベントオブジェクト](#)) を返した場合に発生します。
- 実行エラー (FunctionExecutionErrors) - 特定の期間に発生した実行エラーの数。実行エラーは、関数が正常に完了しなかった場合に発生します。
- コンピューティング使用率 (FunctionComputeUtilization) - 関数の実行にかかった時間 (最大許容時間に対するパーセンテージ)。たとえば、値 35 は、関数が最大許容時間の 35% で完了したことを意味します。このメトリクスは、0から100までの数値です。
- スロットリング (FunctionThrottles) - 指定された期間に関数がスロットリングされた回数。関数は、次の理由でスロットリングできます。
 - この関数は、実行に許容される最大時間を継続的に超えている
 - この関数によってコンパイルエラーが発生する
 - 1秒あたりのリクエスト数が異常に多い

CloudFront コンソールでこれらのメトリクスを表示するには、[モニタリングページ](#) に移動します。特定の関数のグラフを表示するには、[Functions] で関数を選択した後、[View function metrics] を選択します。

これらのメトリクスはすべて、CloudFront 名前空間 CloudWatch の米国東部 (バージニア北部) リージョン (us-east-1) の に発行されます。これらのメトリクスは、CloudWatch コンソールで表示することもできます。CloudWatch コンソールでは、関数ごと、またはディストリビューションごとの関数ごとにメトリクスを表示できます。

を使用して CloudWatch、これらのメトリクスに基づいてアラームを設定することもできます。例えば、実行時間 (FunctionComputeUtilization) メトリクスに基づいてアラームを設定できます。このメトリクスは、関数の実行にかかった許容時間の割合を表します。実行時間が一定時間内に特定の値 (許容時間の 70% 以上を 15 分継続して超過など) に達すると、アラームがトリガーされます。アラームの作成時に、アラームの値と時間単位を指定します。

Note

CloudFront 関数は、本番リクエストとレスポンスに回答して実行されるLIVEステージの関数に対して CloudWatch のみ、メトリクスを送信します。[関数をテスト](#)する場合、CloudFront はにメトリクスを送信しませんCloudWatch。テスト出力には、エラー、コンピューティング使用率、関数ログ (console.log() ステートメント) に関する情報が含まれていますが、この情報はに送信されません CloudWatch。

CloudWatch API を使用してこれらのメトリクスを取得する方法については、「」を参照してください[the section called “API を使用したメトリクスの取得”](#)。

メトリクスのアラームを作成する

CloudFront コンソールでは、特定の CloudFront メトリクスに基づいて Amazon Simple Notification Service (Amazon SNS) から通知を受け取るようにアラームを設定できます。[CloudFront コンソールのアラームページ](#)でアラームを設定できます。

コンソールでアラームを作成するには、以下の値を指定します。

メトリクス

アラームを作成する対象のメトリクス。

ディストリビューション

アラームを作成する対象の CloudFront ディストリビューション。

アラーム名

アラームの名前。

通知の送信先

このメトリクスがアラームをトリガーした場合に通知を送信する先の Amazon SNS トピック。

Whenever **<metric>** **<operator>** **<value>**

CloudWatch がアラームをトリガーし、Amazon SNS トピックに通知を送信するタイミングを指定します。たとえば、5xx エラー率が 1% を超えた場合に通知を受け取るには、次のように指定します。

平均が 5 になるたびにxxErrorRate > 1

値を指定する際には、以下の点に注意します。

- 区切り文字を使用せず、整数のみを入力します。たとえば、1,000 を指定する場合は、「**1000**」と入力します。
- 4xx、5xx、および合計エラー率の場合、指定する値は割合 (%) です。
- リクエスト、ダウンロードされたバイト数、アップロードされたバイト数の場合、指定する値は単位です。たとえば、1073742000 バイトと指定します。

For at least **<number>** consecutive periods of **<time period>**

がアラームを CloudWatch トリガーする前に、メトリクスが基準を満たす必要がある指定された期間の連続した期間の数を指定します。値を選択するときは、一時的または瞬間的な問題に対してはアラームをトリガーしない値と、持続的な問題や実際の問題に対してはアラームをトリガーする値の間で適切なバランスを取ります。

CSV 形式でのメトリクスデータのダウンロード

ディストリビューションの CloudWatch メトリクスデータは CSV 形式でダウンロードできます。[CloudFront コンソール](#)で特定のディストリビューションのディストリビューションメトリクスを表示するときに、データをダウンロードできます。

レポートに関する情報

レポートの先頭数行には次の情報が含まれます。

バージョン

CloudFront レポートバージョン。

レポートを行う

レポートの名前。

DistributionID

レポートを実行した対象のディストリビューションの ID。

StartDateUTC

協定世界時 (UTC) によるレポートを実行した日付範囲の開始日。

EndDateUTC

協定世界時 (UTC) によるレポートを実行した日付範囲の終了日。

GeneratedTimeUTC

協定世界時 (UTC) によるレポートを実行した日時。

詳細度

レポートの行ごとの期間 (ONE_MINUTE など)。

メトリクスレポートのデータ

レポートには次の値が含まれています。

DistributionID

レポートを実行した対象のディストリビューションの ID。

FriendlyName

ディストリビューションの代替ドメイン名 (CNAME)、(存在する場合)。ディストリビューションに代替ドメイン名がない場合、リストにはディストリビューションのオリジンドメイン名が含まれます。

TimeBucket

協定世界時 (UTC) によるデータに該当する時間または日付。

リクエスト

該当期間中のすべての HTTP ステータスコード (200、404 など) およびすべてのメソッド (GET、HEAD、POST など) に対するリクエストの総数。

BytesDownloaded

期間中に指定したディストリビューションについてビューワーがダウンロードしたバイト数。

BytesUploaded

指定したディストリビューションについて、該当期間中にビューワーがダウンロードしたバイト数。

TotalErrorRatePct

指定したディストリビューションについて、該当期間中に HTTP ステータスコードが 4xx または 5xx エラーであったリクエストの割合 (%)。

4xxErrorRatePct

指定したディストリビューションについて、該当期間中に HTTP ステータスコードが 4xx エラーであったリクエストの割合 (%)。

5xxErrorRateピクセル

指定したディストリビューションについて、該当期間中に HTTP ステータスコードが 5xx エラーであったリクエストの割合 (%)。

ディストリビューションの[追加のメトリクスを有効化](#)している場合、レポートには以下の追加の値も表示されます。

401ErrorRatePct

指定したディストリビューションについて、該当期間中に HTTP ステータスコードが 401 エラーであったリクエストの割合 (%)。

403ErrorRatePct

指定したディストリビューションについて、該当期間中に HTTP ステータスコードが 403 エラーであったリクエストの割合 (%)。

404ErrorRatePct

指定したディストリビューションについて、該当期間中に HTTP ステータスコードが 404 エラーであったリクエストの割合 (%)。

502ErrorRatePct

指定したディストリビューションについて、該当期間中に HTTP ステータスコードが 502 エラーであったリクエストの割合 (%)。

503ErrorRatePct

指定したディストリビューションについて、該当期間中に HTTP ステータスコードが 503 エラーであったリクエストの割合 (%)。

504ErrorRatePct

指定したディストリビューションについて、該当期間中に HTTP ステータスコードが 504 エラーであったリクエストの割合 (%)。

OriginLatency

リクエスト CloudFront を受信してから、CloudFront キャッシュではなくオリジンから提供されたリクエストに対して、ネットワーク (ビューワーではなく) へのレスポンスの提供を開始してから経過した合計時間をミリ秒単位で表したものです。これは、最初のバイトレイテンシー、またはとも呼ばれますtime-to-first-byte。

CacheHitRate

がキャッシュからコンテンツを CloudFront 供給したすべてのキャッシュ可能なリクエストの割合。HTTP POST/PUT リクエストおよびエラーは、キャッシュ可能なリクエストとは見なされません。

CloudWatch API を使用したメトリクスの取得

Amazon CloudWatch API または CLI を使用して、構築するプログラムまたはアプリケーションの CloudFront メトリクスを取得できます。raw データを使用して、独自のカスタムダッシュボードや独自のアラームツールなどを構築できます。

CloudWatch API から CloudFront メトリクスを取得するには、米国東部 (バージニア北部) リージョン () を使用する必要がありますus-east-1。また、各メトリクスの特定の値とタイプも知っておく必要があります。

トピック

- [すべての CloudFront メトリクスの値](#)
- [ディストリビューションメトリクスの値 CloudFront](#)
- [CloudFront 関数メトリクスの値](#)

すべての CloudFront メトリクスの値

次の値は、すべての CloudFront メトリクスに適用されます。

Namespace

Namespace の値は常に AWS/CloudFront です。

ディメンション

各 CloudFront メトリクスには、次の 2 つのディメンションがあります。

DistributionId

メトリクスを取得する CloudFront デистриビューションの ID。

FunctionName

メトリクスを取得する関数の名前 (CloudFront 関数内)。

このディメンションは、関数にのみ適用されます。

Region

Region はグローバルサービスであるため Global、 の値は常に CloudFront です。

Note

CloudWatch API から CloudFront メトリクスを取得するには、米国東部 (バージニア北部) リージョン (米国東部 1) を使用する必要があります。

デистриビューションメトリクスの値 CloudFront

次のリストから情報を使用して、CloudWatch API から特定の CloudFront デистриビューションメトリクスの詳細を取得します。これらのメトリクスの一部は、デистриビューションで追加のメトリクスを有効にしている場合にのみ使用できます。

Note

各メトリクスには 1 つの統計 (Average または Sum) のみを適用できます。次のリストは、各メトリクスに適用できる統計を示しています。

4xx エラー率

レスポンスの HTTP ステータスコードが 4xx であるすべてのビューワーリクエストの割合 (%)。

- メトリクス名: 4xxErrorRate
- 有効な統計: Average
- 単位: Percent

401 エラー率

レスポンスの HTTP ステータスコードが 401 であるすべてのビューワーリクエストの割合 (%)。このメトリクスを取得するには、まず[追加のメトリクスを有効にする](#)必要があります。

- メトリクス名: 401ErrorRate
- 有効な統計: Average
- 単位: Percent

403 エラー率

レスポンスの HTTP ステータスコードが 403 であるすべてのビューワーリクエストの割合 (%)。このメトリクスを取得するには、まず[追加のメトリクスを有効にする](#)必要があります。

- メトリクス名: 403ErrorRate
- 有効な統計: Average
- 単位: Percent

404 エラー率

レスポンスの HTTP ステータスコードが 404 であるすべてのビューワーリクエストの割合 (%)。このメトリクスを取得するには、まず[追加のメトリクスを有効にする](#)必要があります。

- メトリクス名: 404ErrorRate
- 有効な統計: Average
- 単位: Percent

5xx エラー率

レスポンスの HTTP ステータスコードが 5xx であるすべてのビューワーリクエストの割合 (%)。

- メトリクス名: 5xxErrorRate
- 有効な統計: Average
- 単位: Percent

502 エラー率

レスポンスの HTTP ステータスコードが 502 であるすべてのビューワーリクエストの割合 (%)。このメトリクスを取得するには、まず[追加のメトリクスを有効にする](#)必要があります。

- メトリクス名: 502ErrorRate
- 有効な統計: Average

- 単位: Percent

503 エラー率

レスポンスの HTTP ステータスコードが 503 であるすべてのビューワーリクエストの割合 (%)。このメトリクスを取得するには、まず[追加のメトリクスを有効にする](#)必要があります。

- メトリクス名: 503ErrorRate
- 有効な統計: Average
- 単位: Percent

504 エラー率

レスポンスの HTTP ステータスコードが 504 であるすべてのビューワーリクエストの割合 (%)。このメトリクスを取得するには、まず[追加のメトリクスを有効にする](#)必要があります。

- メトリクス名: 504ErrorRate
- 有効な統計: Average
- 単位: Percent

ダウンロードされたバイト数

GET リクエスト、HEAD リクエスト、および OPTIONS リクエストに対してビューワーがダウンロードしたバイト総数。

- メトリクス名: BytesDownloaded
- 有効な統計: Sum
- 単位: None

アップロードされたバイト数

POST リクエストと PUT リクエストを使用して CloudFront でビューワーがオリジンにアップロードしたバイト総数。

- メトリクス名: BytesUploaded
- 有効な統計: Sum
- 単位: None

キャッシュヒットレート

がキャッシュからコンテンツを CloudFront 供給したすべてのキャッシュ可能なリクエストの割合。HTTP POST/PUT リクエストおよびエラーは、キャッシュ可能なリクエストとは見なされません。このメトリクスを取得するには、まず[追加のメトリクスを有効にする](#)必要があります。

- メトリクス名: CacheHitRate
- 有効な統計: Average
- 単位: Percent

オリジンのレイテンシー

がリクエスト CloudFront を受信してから、CloudFront キャッシュではなくオリジンから提供されるリクエストに対して、ネットワーク (ビューワーではなく) へのレスポンスの提供を開始するまでに費やされた合計時間をミリ秒単位で表します。これは、最初のバイトレイテンシー、またはとも呼ばれますtime-to-first-byte。このメトリクスを取得するには、まず[追加のメトリクスを有効にする](#)必要があります。

- メトリクス名: OriginLatency
- 有効な統計: Percentile
- 単位: Milliseconds

Note

CloudWatch API からPercentile統計を取得するには、ではなくExtendedStatisticsパラメータを使用しますStatistics。詳細については、[GetMetricStatistics](#) 「Amazon CloudWatch API リファレンス」の[AWS SDKs](#)のリファレンスドキュメントを参照してください。

リクエスト

すべての HTTP メソッド CloudFront、および HTTP リクエストと HTTPS リクエストの両方について、が受信したビューワーリクエストの合計数。

- メトリクス名: Requests
- 有効な統計: Sum
- 単位: None

合計エラー率

レスポンスの HTTP ステータスコードが 4xx または 5xx であるすべてのビューワーリクエストの割合 (%)。

- メトリクス名: TotalErrorRate

- 有効な統計: Average
- 単位: Percent

CloudFront 関数メトリクスの値

次のリストの情報を使用して、CloudWatch API から特定の CloudFront関数メトリクスの詳細を取得します。

Note

各メトリクスには 1 つの統計 (Average または Sum) のみを適用できます。次のリストは、各メトリクスに適用できる統計を示しています。

呼び出し

指定された期間に関数が起動 (呼び出し) された回数。

- メトリクス名: FunctionInvocations
- 有効な統計: Sum
- 単位: None

検証エラー

指定された期間に関数によって生成された検証エラーの数。検証エラーは、関数は正常に実行されたが、無効なデータ (無効なイベントオブジェクト) を返した場合に発生します。

- メトリクス名: FunctionValidationErrors
- 有効な統計: Sum
- 単位: None

実行エラー

特定の期間に発生した実行エラーの数。実行エラーは、関数が正常に完了しなかった場合に発生します。

- メトリクス名: FunctionExecutionErrors
- 有効な統計: Sum
- 単位: None

コンピューティング使用率

関数の実行にかかった時間の長さ (0-100) を、最大許容時間に対するパーセンテージで示します。たとえば、値 35 は、関数が最大許容時間の 35% で完了したことを意味します。

- メトリクス名: FunctionComputeUtilization
- 有効な統計: Average
- 単位: Percent

Throttles

指定された期間に関数がスロットリングされた回数。

- メトリクス名: FunctionThrottles
- 有効な統計: Sum
- 単位: None

CloudFront およびエッジ関数のログ記録

Amazon CloudFront にはさまざまな種類のログ記録が用意されています。CloudFront デイストリビューションに送信されるビューワーリクエストをログに記録することも、AWSアカウントのCloudFront サービスアクティビティ (API アクティビティ) をログに記録することもできます。[エッジコンピューティング](#)関数からログを取得することもできます。

リクエストのログ記録

CloudFront には、デイストリビューションに送信されるリクエストをログに記録する以下の方法が用意されています。

標準ログ (アクセスログ)

CloudFront 標準ログは、デイストリビューションに対して行われたすべてのリクエストに関する詳細なレコードを提供します。これらのログは、セキュリティやアクセスの監査などの多くのシナリオに役立ちます。

CloudFront 標準ログは、選択した Amazon S3 バケットに配信されます。CloudFront では、ログファイルの保存とアクセスには Amazon S3 の料金が発生しますが、標準ログには課金されません。

詳細については、「[標準ログ \(アクセスログ\) の使用](#)」を参照してください。

リアルタイムログ

CloudFront リアルタイムログは、ディストリビューションに対して行われたリクエストに関する情報をリアルタイムで提供します (ログレコードはリクエストを受信してから数秒以内に配信されます)。リアルタイムログのサンプリングレート、つまり、リアルタイムのログ記録を受信するリクエストの割合を選択できます。ログ記録で受信が行われる特定のフィールドを選択することもできます。

CloudFront リアルタイムログは、Amazon Kinesis Data Streams で選択したデータストリームに配信されます。は、Kinesis Data Streams の使用に対して発生する料金に加えて、リアルタイムログに対して CloudFront 課金します。

詳細については、「[リアルタイムログ](#)」を参照してください。

エッジ関数をログ記録する

Amazon CloudWatch Logs を使用して、Lambda@Edge と CloudFront Functions の両方の[エッジ関数](#)のログを取得できます。コンソール CloudWatch または CloudWatch Logs API を使用してログにアクセスできます。詳細については、「[the section called “エッジ関数のログ”](#)」を参照してください。

サービスアクティビティのログ記録

を使用してAWS CloudTrail、AWSアカウントの CloudFront サービスアクティビティ (API アクティビティ) をログに記録できます。は、 のユーザー、ロール、またはAWSサービスによって実行された API アクシオンの記録 CloudTrail を提供します CloudFront。で収集された情報を使用して CloudTrail、 に対して行われた API リクエスト CloudFront、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

詳細については、「[AWS CloudTrail を使用して API に送信されたリクエストを CloudFront キャプチャする](#)」を参照してください。

トピック

- [標準ログ \(アクセスログ\) の設定および使用](#)
- [リアルタイムログ](#)
- [エッジ関数のログ](#)
- [AWS CloudTrail を使用して API に送信されたリクエストを CloudFront キャプチャする](#)

標準ログ (アクセスログ) の設定および使用

が CloudFront 受信するすべてのユーザーリクエストに関する詳細情報を含むログファイルを作成する CloudFront ように を設定できます。これらは、標準ログと呼ばれます。また、アクセスログとも呼ばれています。標準ログを有効にする場合は、ファイル CloudFront を保存する Amazon S3 バケットを指定することもできます。

ディストリビューションを作成または更新するとき、標準ログを有効にできます。詳細については、「[ディストリビューションを作成または更新する場合に指定する値](#)」を参照してください。

CloudFront は、ディストリビューションに対して行われたリクエストに関する情報をリアルタイムで提供するリアルタイムログも提供します (ログはリクエストを受信してから数秒以内に配信されます)。リアルタイムログを使用して、コンテンツ配信のパフォーマンスに基づいて監視、分析、アクションを実行できます。詳細については、「[リアルタイムログ](#)」を参照してください。

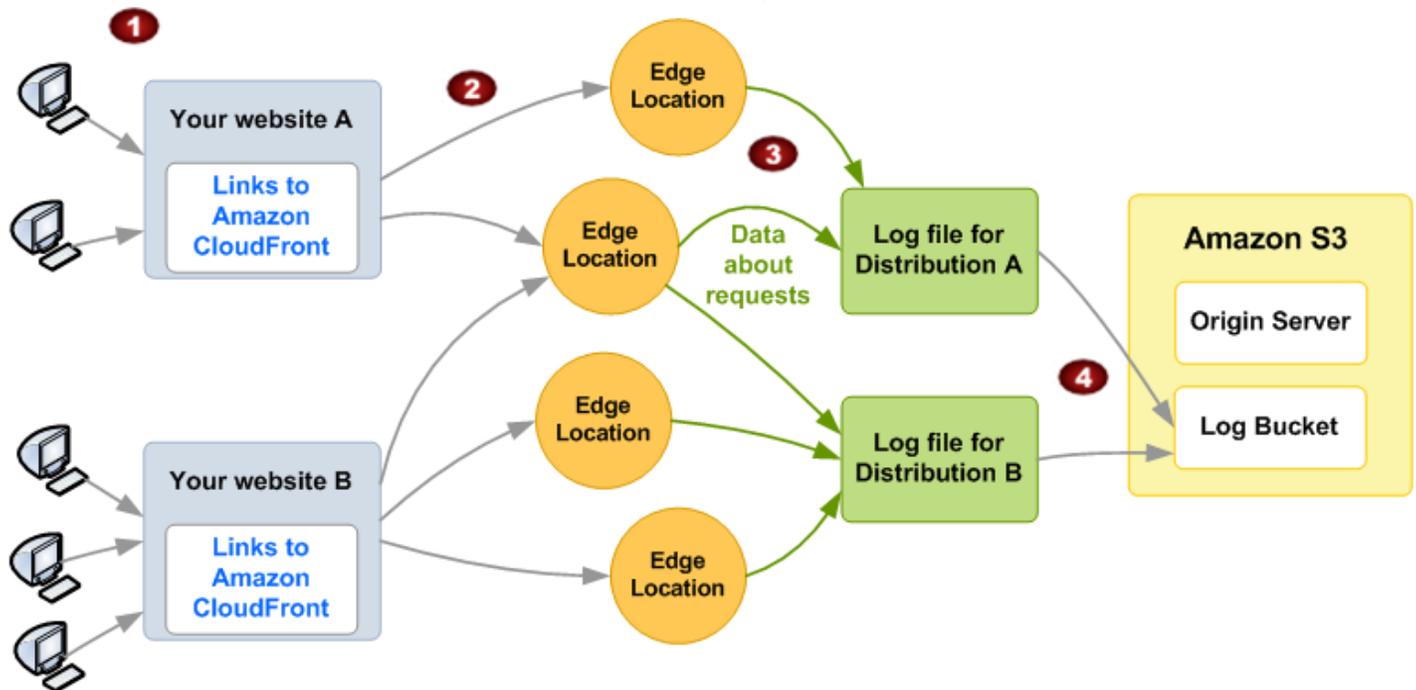
トピック

- [標準ログ記録のしくみ](#)
- [標準ログ用の Amazon S3 バケットの選択](#)
- [標準ログ記録の設定およびログファイルへのアクセスに必要なアクセス許可](#)
- [SSE-KMS バケット必須のキーポリシー](#)
- [ファイル名の形式](#)
- [標準ログファイル配信のタイミング](#)
- [リクエスト URL またはヘッダーが最大のサイズを超えた場合にリクエストがどのようにログに記録されるか](#)
- [標準ログの分析](#)
- [標準ログ記録設定の編集](#)
- [Amazon S3 バケットからの標準ログファイルの削除](#)
- [標準ログファイル形式](#)
- [標準ログの料金](#)

標準ログ記録のしくみ

次の図は、 がオブジェクトのリクエストに関する情報を CloudFront ログに記録する方法を示しています。

Users in different locations



次の図に示すように、ガオブジェクトのリクエストに関する情報を CloudFront ログに記録する方法について説明します。

1. この図には、A と B の 2 つのウェブサイトと、対応する 2 つの CloudFront デイストリビューションがあります。ユーザーは、デイストリビューションに関連付けられている URL を使用してオブジェクトをリクエストします。
2. CloudFront は、各リクエストを適切なエッジロケーションにルーティングします。
3. CloudFront は、各リクエストに関するデータを、そのデイストリビューションに固有のログファイルに書き込みます。この例では、デイストリビューション A に関連するリクエストについての情報がデイストリビューション A 専用のログファイルに、デイストリビューション B に関連するリクエストについての情報がデイストリビューション B 専用のログファイルに書き込まれます。
4. CloudFront は、ログ記録を有効にしたときに指定した Amazon S3 バケットに、デイストリビューションのログファイルを定期的に保存します。CloudFront その後、はデイストリビューションの新しいログファイルに後続のリクエストに関する情報の保存を開始します。

一定の時間、お客様のコンテンツに対してユーザーアクセスがない場合、その時間のログファイルを受け取ることはありません。

ログファイルには、1つのリクエストの詳細が1エントリとして記録されます。ログファイル形式の詳細については、「[標準ログファイル形式](#)」を参照してください。

Note

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、ログを使用してコンテンツに対するリクエストの性質を理解することをお勧めします。は、ベストエフォートベースでアクセスログ CloudFront を提供します。特定のリクエストのログエントリが、リクエストが実際に処理されてからかなり後に配信されることも、(まれに)一切配信されないこともあります。ログエントリをアクセスログから省略すると、アクセスログ内のエントリ数は AWS の請求と使用状況レポートに表示される使用量と一致しなくなります。

標準ログ用の Amazon S3 バケットの選択

ディストリビューションのログ記録を有効にするときは、ログファイルを保存する Amazon S3 バケット CloudFrontを指定します。オリジンとして Amazon S3 を使用する場合は、同じバケットをログファイルに使用しないことをお勧めします。別々のバケットを使用すると、メンテナンスが容易になります。

Important

強制を行ったバケット所有者に設定される [S3 オブジェクトの所有権](#) を使用して Amazon S3 バケットを選択してはいけません。この設定により、バケットとその中のオブジェクトACLs が無効になり、はバケットにログファイルを配信できなくなります CloudFront。

Important

は、これらのリージョンのバケットに標準ログを配信しないため、以下のリージョンでは Amazon S3 バケットを選択しないでください。 CloudFront

- アフリカ (ケープタウン)
- アジアパシフィック (香港)
- アジアパシフィック (ハイデラバード)
- アジアパシフィック (ジャカルタ)
- アジアパシフィック (メルボルン)

- カナダ (中部)
- 欧州 (ミラノ)
- 欧州 (スペイン)
- 欧州 (チューリッヒ)
- イスラエル (テルアビブ)
- 中東 (バーレーン)
- 中東 (アラブ首長国連邦)

複数のディストリビューションのログファイルを同じバケットに保存することもできます。ログ記録を有効にする際には、ファイル名のプレフィックスをオプションで指定できます。これにより、どのログファイルがどのディストリビューションに関連しているか追跡できます。

標準ログ記録の設定およびログファイルへのアクセスに必要なアクセス許可

Important

2023年4月から、CloudFront 標準ログに使用される新しい S3 バケットの S3 アクセスコントロールリスト (ACLs) を有効にする必要があります。ACL は、[バケット作成のステップ中](#)、または[バケットが作成された後](#)に有効にできます。

変更の詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[新しい S3 バケットのデフォルト設定に関するよくある質問](#)」と、「AWS ニュースブログ」の「[注意喚起: 2023年4月に予定されている Amazon S3 のセキュリティ変更](#)」を参照してください。

AWS アカウントには、ログファイル用に指定するバケットに対する以下の許可が必要です。

- バケットの S3 アクセスコントロールリスト (ACL) は FULL_CONTROL を付与する必要があります。バケット所有者のアカウントには、デフォルトでこのアクセス許可があります。権限がない場合、バケット所有者はバケットの ACL を更新する必要があります。
- s3:GetBucketAcl
- s3:PutBucketAcl

次の点に注意してください。

バケットの ACL

ディストリビューションを作成または更新してログ記録を有効にすると、これらのアクセス許可 CloudFront を使用してバケットの ACL を更新し、awslogsdelivery アカウントに アクセス FULL_CONTROL 許可を付与します。awslogsdelivery アカウントはログファイルをバケットに書き込みます。アカウントに ACL を更新するために必要なアクセス許可がない場合、ディストリビューションの作成または更新は失敗します。

状況によっては、バケットを作成するリクエストをプログラムで送信したが、指定した名前のバケットが既に存在する場合、S3 ではバケットのアクセス許可をデフォルト値にリセットします。アクセスログを S3 バケットに保存 CloudFront するようにを設定していて、そのバケットへのログの取得を停止した場合は、バケットのアクセス許可をチェックして、に必要なアクセス許可 CloudFront があることを確認してください。

バケットの ACL を復元する

awslogsdelivery アカウントのアクセス許可を削除した場合、CloudFront はログを S3 バケットに保存できません。CloudFront がディストリビューションのログを再度保存できるようにするには、次のいずれかを実行して ACL アクセス許可を復元します。

- [ディストリビューションのログ記録を無効にし CloudFront、再度有効にします。](#) 詳細については、「[ディストリビューションを作成または更新する場合に指定する値](#)」を参照してください。
- Amazon S3 コンソールで S3 バケットに移動してアクセス許可を追加することで、awslogsdelivery の ACL アクセス許可を手動で追加します。awslogsdelivery の ACL を追加するには、アカウントの正規 ID を入力する必要があります。これは次のとおりです。

```
c4c1ede66af53448b93c283ce9448c4ba468c9432aa01d700d3878632f77d2d0
```

ACL を S3 バケットに追加する方法の詳細については、Amazon Simple Storage Service ユーザーガイドの「[ACL バケットの許可を設定する方法](#)」を参照してください。

各ログファイルの ACL

バケットの ACL に加えて、各ログファイルの ACL があります。バケット所有者にはログファイルに対する FULL_CONTROL アクセス許可があり、ディストリビューション所有者 (バケット所有者と異なる場合) にはアクセス許可がありません。awslogsdelivery アカウントには読み取りアクセス許可と書き込みアクセス許可があります。

ログ記録の無効化

ログ記録を無効にすると、CloudFront はバケットまたはログファイルの ACLs を削除しません。これはお客様自身で行うことができます。

SSE-KMS バケット必須のキーポリシー

標準ログ用の S3 バケットで、カスタマーマネージド型キーを使用する AWS KMS keys (SSE-KMS) を用いたサーバー側の暗号化が使用されている場合は、カスタマーマネージド型キーのキーポリシーに次のステートメントを追加する必要があります。これにより、CloudFront はバケットにログファイルを書き込むことができます。(ではログファイルをバケットに書き込むことができないAWS マネージドキーため、で CloudFront SSE-KMS を使用することはできません。)

```
{
  "Sid": "Allow CloudFront to use the key to deliver logs",
  "Effect": "Allow",
  "Principal": {
    "Service": "delivery.logs.amazonaws.com"
  },
  "Action": "kms:GenerateDataKey*",
  "Resource": "*"
}
```

標準ログの S3 バケットで [S3 バケットキー](#) を有する SSE-KMS を使用する場合は、ポリシーステートメントに kms:Decrypt 許可を追加する必要もあります。この場合、完全なポリシーステートメントは次のようになります。

```
{
  "Sid": "Allow CloudFront to use the key to deliver logs",
  "Effect": "Allow",
  "Principal": {
    "Service": "delivery.logs.amazonaws.com"
  },
  "Action": [
    "kms:GenerateDataKey*",
    "kms:Decrypt"
  ],
  "Resource": "*"
}
```

ファイル名の形式

Amazon S3 バケット CloudFront に保存される各ログファイルの名前には、次のファイル名形式が使用されます。

```
<optional prefix>/<distribution ID>.YYYY-MM-DD-HH.unique-ID.gz
```

日付と時刻は協定世界時 (UTC) です。

たとえば、example-prefix をプレフィックスとして使用している場合に、ディストリビューション ID が EMLARXS9EXAMPLE であれば、ファイル名は次のようになります。

```
example-prefix/EMLARXS9EXAMPLE.2019-11-14-20.RT4KCN4SGK9.gz
```

ディストリビューションのログ記録を有効にする際には、ファイル名のプレフィックスをオプションで指定できます。これにより、どのログファイルがどのディストリビューションに関連しているか追跡できます。ログファイルプレフィックスの値を含め、プレフィックスがスラッシュ (/) で終わらない場合は、`example-prefix/` によって自動的に CloudFront 追加されます。プレフィックスがスラッシュで終わる場合は、別のスラッシュを追加 CloudFront しないでください。

ファイル名の .gz 末尾にある `example-prefix/` は、CloudFront が gzip を使用してログファイルを圧縮したことを示します。

標準ログファイル配信のタイミング

CloudFront は、ディストリビューションの標準ログを 1 時間に数回配信します。一般に、ログファイルには、特定の期間中に が CloudFront 受信したリクエストに関する情報が含まれています。CloudFront は通常、ログに表示されるイベントから 1 時間以内に、その期間のログファイルを Amazon S3 バケットに配信します。ただし、ある期間のログファイルエントリの一部またはすべてが、最大で 24 時間遅れることもあります。ログエントリが遅れた場合、CloudFront はこれらをログファイルに保存します。そのファイル名には、ファイルが配信された日時ではなく、リクエストが発生した期間の日時が含まれます。

ログファイルを作成する場合、`example-prefix/` は、ログファイルがカバーする期間中にオブジェクトのリクエストを受信したすべてのエッジロケーションからのディストリビューションの情報を CloudFront 統合します。

CloudFront は、ディストリビューションに関連付けられたオブジェクトに対して CloudFront 受信したリクエストの数に応じて、1 つの期間に複数のファイルを保存できます。

CloudFront は、ログ記録を有効にしてから約 4 時間後にアクセスログを確実に配信し始めます。この時間以前にも少しのアクセスログを取得できる場合もあります。

Note

期間中にオブジェクトに対してユーザーによるリクエストがなければ、その期間のログファイルは配信されません。

CloudFront は、ディストリビューションに対して行われたリクエストに関する情報をリアルタイムで提供するリアルタイムログも提供します (ログはリクエストを受信してから数秒以内に配信されます)。リアルタイムログを使用して、コンテンツ配信のパフォーマンスに基づいて監視、分析、アクションを実行できます。詳細については、「[リアルタイムログ](#)」を参照してください。

リクエスト URL またはヘッダーが最大のサイズを超えた場合にリクエストがどのようにログに記録されるか

クッキーを含むすべてのリクエストヘッダーの合計サイズが 20 KB を超える場合、または URL が 8192 バイトの URL サイズ制限を超える場合、CloudFront ではリクエストを完全に解析できないため、リクエストをログに記録できません。リクエストがログ記録されないため、返された HTTP エラーステータスコードをログファイルで表示できません。

リクエストボディが最大サイズを超えると、HTTP エラー状態コードを含むリクエストがログに記録されます。

標準ログの分析

1 時間ごとに複数のアクセスログが配信される可能性があるため、特定の期間に対して受信したすべてのログファイルをまとめて 1 つのファイルにしておくことをお勧めします。これにより、その期間のデータをより正確かつ完全に分析することができます。

アクセスログを分析する方法の 1 つとして [Amazon Athena](#) を使用する方法があります。Athena は、AWS を含む サービスのデータの分析に役立つインタラクティブなクエリサービスです CloudFront。詳細については、「[Amazon Athena ユーザーガイド](#)」の「[Amazon CloudFront Logs のクエリ](#)」を参照してください。 Amazon Athena

さらに、次の AWS ブログ投稿では、アクセスログを分析するいくつかの方法について説明しています。

- [Amazon CloudFront リクエストのログ記録](#) (HTTP 経由で配信されるコンテンツの場合)
- [CloudFront のログ機能の拡張 – クエリ文字列](#)

⚠ Important

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、ログを使用してコンテンツに対するリクエストの性質を理解することをお勧めします。は、ベストエフォートベースでアクセスログ CloudFront を提供します。特定のリクエストのログエントリが、リクエストが実際に処理されてからかなり後に配信されることも、(まれに) 一切配信されないこともあります。ログエントリがアクセスログから省略された場合、アクセスログ内のエントリ数が AWS の利用状況レポートと請求レポートに表示される利用量と一致しなくなります。

標準ログ記録設定の編集

コンソールまたは [CloudFront](#) CloudFront API を使用して、ログ記録を有効または無効にしたり、ログが保存されている Amazon S3 バケットを変更したり、ログファイルのプレフィックスを変更したりできます。ログ作成設定の変更は 12 時間以内に有効になります。

詳細については、次のトピックを参照してください。

- CloudFront コンソールを使用してディストリビューションを更新するには、「」を参照してください [ディストリビューションの更新](#)。
- CloudFront API を使用してディストリビューションを更新するには、「Amazon CloudFront API リファレンス」の [UpdateDistribution](#) 「」を参照してください。

Amazon S3 バケットからの標準ログファイルの削除

CloudFront は、Amazon S3 バケットからログファイルを自動的に削除しません。Amazon S3 バケットからログファイルを削除する方法については、次のトピックを参照してください。

- Amazon S3 コンソールの使用: Amazon Simple Storage Service Console ユーザーガイドの「[オブジェクトの削除](#)」。
- Amazon Simple Storage Service API リファレンスの [DeleteObject](#) 「REST API の使用 : 」。

標準ログファイル形式

ログファイルには、1 つのビューワーリクエストの詳細が 1 エントリとして記録されます。ログファイルの特性は次のとおりです。

- [W3C 拡張ログファイル形式](#) を使用します。

- タブ区切りの値が含まれます。
- レコードが必ずしも時系列順に含まれているとは限りません。
- 2つのヘッダー行が含まれます。1つのヘッダー行にファイル形式のバージョンが示され、もう1つのヘッダー行に、各レコードに含まれる W3C フィールドが示されます。
- フィールド値に URL エンコードされたスペースおよび特定の他の文字を含めます。

URL エンコードされた同等の文字は、次の文字に使用されます。

- ASCII 文字コード 0~32 以内
- ASCII 文字コード 127 以上
- 次の表のすべての文字

URL エンコーディング標準は [RFC 1738](#) で定義されています。

URL エンコードされた値	文字
%3C	<
%3E	>
%22	"
%23	#
%25	%
%7B	{
%7D	}
%7C	
%5C	\
%5E	^
%7E	~
%5B	[

URL エンコードされた値	文字
%5D]
%60	,
%27	'
%20	スペース

標準ログファイルフィールド

ディストリビューションのログファイルには、33 のフィールドが含まれています。次のリストは、各フィールド名と、そのフィールドに保持される情報の説明を順番に示しています。

1. **date**

イベントが発生した日付。YYYY-MM-DD 形式です。たとえば、2019-06-30 と指定します。日付と時刻は協定世界時 (UTC) です。接続の場合 WebSocket、これは接続が閉じられた日付です。

2. **time**

CloudFront サーバーがリクエストへの応答を完了した時刻 (UTC)01:42:39。たとえば、。WebSocket 接続の場合、これは接続が閉じられた時刻です。

3. **x-edge-location**

リクエストを処理したエッジロケーション。各エッジロケーションは、3 文字コードと、割り当てられた任意の数字で識別されます (例: DFW3)。通常、この 3 文字コードは、エッジロケーションの地理的場所の近くにある空港の、国際航空運送協会 (IATA) の空港コードに対応します。(これらの略語は今後変更される可能性があります)。

4. **sc-bytes**

サーバーがリクエストに応じてビューワーに送信したデータ (ヘッダーを含む) のバイトの合計数。WebSocket 接続の場合、これは接続を介してサーバーからクライアントに送信された合計バイト数です。

5. **c-ip**

リクエスト元のビューワーの IP アドレス (192.0.2.183 または 2001:0db8:85a3::8a2e:0370:7334 など)。ビューワーが HTTP プロキシまたはロードバラ

ンサーを使用してリクエストを送った場合、このフィールドの値はプロキシまたはロードバランサーの IP アドレスです。x-forwarded-for フィールドも参照してください。

6. cs-method

ビューワーから受信した HTTP リクエストメソッド。

7. cs(Host)

CloudFront デイストリビューションのドメイン名 (d1111111abcdef8.cloudfront.net など)。

8. cs-uri-stem

パスとオブジェクトを識別するリクエスト URL の部分 (/images/cat.jpg など)。URL 内の疑問符 (?) およびクエリ文字列はログに含まれません。

9. sc-status

次のいずれかの値が含まれます。

- サーバーのレスポンスの HTTP ステータスコード (例: 200)。
- 000。この値は、サーバーがリクエストに回答する前に、ビューワーが接続を閉じたことを示します。サーバーがレスポンスの送信を開始した後にビューワーが接続を閉じた場合、このフィールドには、サーバーが送信を開始したレスポンスの HTTP ステータスコードが含まれません。

10.cs(Referer)

リクエスト内の Referer ヘッダーの値。これはリクエスト元のドメインの名前です。一般的なリファラーとして、検索エンジン、オブジェクトに直接リンクされた他のウェブサイト、ユーザー自身のウェブサイトなどがあります。

11.cs(User-Agent)

リクエスト内の User-Agent ヘッダーの値。User-Agent ヘッダーでリクエスト元 (リクエスト元のデバイスとブラウザのタイプなど) が識別されます。リクエスト元が検索エンジンの場合は、どの検索エンジンかも識別されます。

12.cs-uri-query

リクエスト URL のクエリ文字列の部分 (ある場合)。

URL にクエリ文字列が含まれない場合、このフィールドの値はハイフン (-) です。詳細については、「[クエリ文字列パラメータに基づくコンテンツのキャッシュ](#)」を参照してください。

13.cs(Cookie)

名前と値のペアおよび関連属性を含む、リクエスト内の Cookie ヘッダー。

Cookie ログ記録を有効にする CloudFront と、は、どの Cookie をオリジンに転送するかにかかわらず、すべてのリクエストの Cookie を記録します。リクエストに Cookie ヘッダーが含まれていない場合、このフィールドの値はハイフン (-) です。Cookie の詳細については、「[Cookie に基づくコンテンツのキャッシュ](#)」を参照してください。

14x-edge-result-type

サーバーが、最後のバイトを渡した後で、レスポンスを分類した方法。場合によっては、サーバーがレスポンスを送る準備ができたときから、サーバーがレスポンスを送り終わるまでの間に、結果タイプが変わることがあります。x-edge-response-result-type フィールドも参照してください。

例えば、HTTP ストリーミングで、サーバーがキャッシュ内でストリームのセグメントを検出するとします。そのシナリオでは、このフィールドの値は、通常 Hit になります。この場合、サーバーがセグメント全体を配信する前にビューワーが接続を閉じると、最終結果タイプ (およびこのフィールドの値) は Error になります。

WebSocket コンテンツはキャッシュできず、オリジンに直接プロキシされるため、接続では Miss このフィールドの値が になります。

以下に示しているのは、可能な値です。

- Hit – サーバーがキャッシュからビューワーにオブジェクトを渡しました。
- RefreshHit – サーバーはキャッシュ内でオブジェクトを検出しましたが、オブジェクトの有効期限が切れていたため、サーバーはオリジンに問い合わせ、キャッシュ内に最新バージョンのオブジェクトがあるかどうかを確認しました。
- Miss – キャッシュ内のオブジェクトでリクエストに対応できなかったため、サーバーはリクエストをオリジンに転送して結果をビューワーに返しました。
- LimitExceeded – CloudFront クォータ (以前は制限と呼ばれていました) を超えたため、リクエストは拒否されました。
- CapacityExceeded – リクエストの受信時にサーバーの容量不足でオブジェクトを渡すことができなかったために、サーバーから HTTP 503 ステータスコードが返されました。
- Error – 通常、これはリクエストがクライアントエラーとなった (sc-status フィールドが 4xx 範囲内の値となる)、またはサーバーエラーになった (sc-status フィールドが 5xx 範囲内の値となる) ことを意味します。sc-status フィールドの値が 200 であるか、このフィールドの値が Error で、x-edge-response-result-type フィールドの値が Error でない場合

は、HTTP リクエストは成功したが、クライアントがすべてのバイトを受信する前に切断されたことを意味します。

- **Redirect** – サーバーは、ディストリビューション設定に従って HTTP から HTTPS にビューワーをリダイレクトしました。

15x-edge-request-id

リクエストを一意に識別する不透明な文字列。CloudFront は `x-amz-cf-id` レスポンスヘッダーでもこの文字列を送信します。

16x-host-header

ビューワーが、このリクエストの `Host` ヘッダーに追加した値。オブジェクト URL で CloudFront ドメイン名 (`d111111abcdef8.cloudfront.net` など) を使用している場合、このフィールドにはそのドメイン名が含まれます。URLs 代替ドメイン名 (CNAME) をオブジェクト URL (`www.example.com`) に使用している場合、このフィールドにはその代替ドメイン名が含まれません。

代替ドメイン名を使っている場合には、フィールド 7 の `cs(Host)` で、ユーザーのディストリビューションに関連するドメイン名を確認します。

17cs-protocol

ビューワーリクエストのプロトコル (`http`、`https`、`ws`、`wss` のいずれか)。

18cs-bytes

ビューワーがリクエストに含めたデータ (ヘッダーを含む) のバイトの合計数。WebSocket 接続の場合、これは接続でクライアントからサーバーに送信された合計バイト数です。

19time-taken

サーバーが、ビューワーのリクエストを受信してからレスポンスの最後のバイトを出力キューに書き込むまでの秒数。サーバーで 1,000 分の 1 秒単位まで測定されます (例: 0.082)。ビューワーから見た場合、レスポンス全体を取得する合計所要時間は、ネットワークのレイテンシーと TCP バッファリングにより、この値よりも長くなります。

20x-forwarded-for

ビューワーが HTTP プロキシまたはロードバランサーを使用してリクエストを送信した場合、`c-ip` フィールドの値はプロキシまたはロードバランサーの IP アドレスです。この場合、このフィールドはリクエスト元のビューワーの IP アドレスです。このフィールドには、IPv4 アドレス

(192.0.2.183 など) または IPv6 アドレス (2001:0db8:85a3::8a2e:0370:7334 など) が含まれます。

ビューワーが HTTP プロキシまたはロードバランサーを使用しなかった場合、このフィールドの値はハイフン (-) です。

21ssl-protocol

リクエストが HTTPS を使用した場合、このフィールドには、リクエストとレスポンスを送信するためにビューワーとサーバーがネゴシエートした SSL/TLS プロトコルが含まれます。指定可能な値のリストについては、[ビューワーとの間でサポートされているプロトコルと暗号 CloudFront](#) でサポートされている SSL/TLS プロトコルを参照してください。

フィールド 17 の `cs-protocol` が `http` である場合、このフィールドの値はハイフン (-) です。

22ssl-cipher

リクエストが HTTPS を使用した場合、このフィールドには、リクエストとレスポンスを暗号化するためにビューワーとサーバーがネゴシエートした SSL/TLS 暗号が含まれます。使用できる値のリストについては、「[ビューワーとの間でサポートされているプロトコルと暗号 CloudFront](#)」で、サポートされている SSL/TLS 暗号化を参照してください。

フィールド 17 の `cs-protocol` が `http` である場合、このフィールドの値はハイフン (-) です。

23x-edge-response-result-type

ビューワーにレスポンスを返す直前にサーバーがレスポンスを分類した方法。x-edge-result-type フィールドも参照してください。以下に示しているのは、可能な値です。

- Hit – サーバーがキャッシュからビューワーにオブジェクトを渡しました。
- RefreshHit – サーバーはキャッシュ内でオブジェクトを検出しましたが、オブジェクトの有効期限が切れていたため、サーバーはオリジンに問い合わせ、キャッシュ内に最新バージョンのオブジェクトがあるかどうかを確認しました。
- Miss – キャッシュ内のオブジェクトでリクエストに対応できなかったため、サーバーはリクエストをオリジンサーバーに転送して結果をビューワーに返しました。
- LimitExceeded – CloudFront クォータ (以前は制限と呼ばれていました) を超えたため、リクエストは拒否されました。
- CapacityExceeded – リクエストの受信時にサーバーの容量不足でオブジェクトを渡すことができなかったために、サーバーから 503 エラーが返されました。

- **Error** – 通常、これはリクエストがクライアントエラーとなった (`sc-status` フィールドが `4xx` 範囲内の値となる)、またはサーバーエラーになった (`sc-status` フィールドが `5xx` 範囲内の値となる) ことを意味します。

`x-edge-result-type` フィールドの値が `Error` であり、このフィールドの値が `Error` でない場合、ダウンロードが完了する前にクライアントが切断されました。

- **Redirect** – サーバーは、ディストリビューション設定に従って HTTP から HTTPS にビューワーをリダイレクトしました。

24.cs-protocol-version

ビューワーがリクエストで指定した HTTP バージョン。指定できる値には、HTTP/0.9、HTTP/1.0、HTTP/1.1、HTTP/2.0 および HTTP/3.0 があります。

25.file-status

[フィールドレベル暗号化](#)がディストリビューション用に設定されている場合、このフィールドにはリクエストボディが正常に処理されたかどうかを示すコードが含まれます。サーバーがリクエストボディを正常に処理し、指定したフィールドの値を暗号化してリクエストをオリジンに転送すると、このフィールドの値は `Processed` になります。`x-edge-result-type` の値は、この場合でもクライアント側またはサーバー側のエラーを示すことができます。

このフィールドで使用できる値は次のとおりです。

- **ForwardedByContentType** – コンテンツタイプが設定されていないため、サーバーは解析や暗号化を行わずにリクエストをオリジンに転送しました。
- **ForwardedByQueryArgs** – フィールドレベル暗号化の設定にないクエリ引数がリクエストに含まれているため、サーバーは解析や暗号化を行わずにリクエストをオリジンに転送しました。
- **ForwardedDueToNoProfile** – フィールドレベル暗号化の設定でプロファイルを指定しなかったため、サーバーは解析や暗号化を行わずにリクエストをオリジンに転送しました。
- **MalformedContentTypeClientError Content-Type** – ヘッダーの値が無効な形式であるため、サーバーはリクエストを拒否し、HTTP 400 ステータスコードをビューワーに返しました。
- **MalformedInputClientError** – リクエストボディが無効な形式であるため、サーバーはリクエストを拒否し、HTTP 400 ステータスコードをビューワーに返しました。
- **MalformedQueryArgsClientError** – クエリ引数が空であるか無効な形式であるため、サーバーはリクエストを拒否し、HTTP 400 ステータスコードをビューワーに返しました。

- `RejectedByContentType` – フィールドレベル暗号化の設定でコンテンツタイプを指定しなかったため、サーバーはリクエストを拒否し、HTTP 400 ステータスコードをビューワーに返しました。
- `RejectedByQueryArgs` – フィールドレベル暗号化の設定でクエリ引数を指定しなかったため、サーバーはリクエストを拒否し、HTTP 400 ステータスコードをビューワーに返しました。
- `ServerError` – オリジンサーバーがエラーを返しました。

リクエストがフィールドレベル暗号化のクォータ (以前は制限と呼ばれていました) を超えた場合、このフィールドには次のいずれかのエラーコードが含まれ、サーバーは HTTP ステータスコード 400 をビューワーに返します。フィールドレベル暗号化に関する最新のクォータのリストについては、「[フィールドレベル暗号化のクォータ](#)」を参照してください。

- `FieldLengthLimitClientError` – 暗号化されるように設定されているフィールドが最大の長さを超えています。
- `FieldNumberLimitClientError` – ディストリビューションによって暗号化されるように設定されているリクエストがフィールド数の制限を超えています。
- `RequestLengthLimitClientError` – フィールドレベル暗号化が設定されている場合にリクエストボディが最大の長さを超えています。

フィールドレベル暗号化がディストリビューション用に設定されていない場合、このフィールドの値はハイフン (-) です。

26. `file-encrypted-fields`

サーバーが暗号化してオリジンに転送した [フィールドレベル暗号化](#) フィールドの数。CloudFront サーバー `file-status` は、データを暗号化するとき処理されたリクエストをオリジンにストリーミングするため、の値がエラーであってもこのフィールドに値を設定できます。

フィールドレベル暗号化がディストリビューション用に設定されていない場合、このフィールドの値はハイフン (-) です。

27. `c-port`

閲覧者からのリクエストのポート番号。

28. `time-to-first-byte`

サーバー上で測定される、要求を受信してから応答の最初のバイトを書き込むまでの秒数。

29. `x-edge-detailed-result-type`

このフィールドには、以下の場合を除き、x-edge-result-type フィールドと同じ値が含まれません。

- オブジェクトが [Origin Shield](#) レイヤーからビューワーに渡された場合、このフィールドには OriginShieldHit が含まれています。
- オブジェクトが CloudFront キャッシュに存在せず、レスポンスが [オリジンリクエストの Lambda@Edge 関数](#) によって生成された場合、このフィールドには `MissGeneratedResponse` が含まれます。
- x-edge-result-type フィールドの値が Error の場合、このフィールドにはエラーに関する詳細情報を含む次のいずれかの値が含まれます。
 - AbortedOrigin – サーバーでオリジンに関する問題が発生しました。
 - ClientCommError – サーバーとビューワーとの通信の問題により、ビューワーへのレスポンスが中断されました。
 - ClientGeoBlocked – ディストリビューションは、ビューワーの地理的位置からのリクエストを拒否するように設定されています。
 - ClientHungUpRequest – リクエストの送信中にビューワーが途中で停止しました。
 - Error – エラータイプが他のどのカテゴリにも適合しないエラーが発生しました。このエラータイプは、キャッシュからのエラーレスポンスをサーバーが渡すときに発生する可能性があります。
 - InvalidRequest – サーバーがビューワーから無効なリクエストを受信しました。
 - InvalidRequestBlocked – 要求されたリソースへのアクセスがブロックされます。
 - InvalidRequestCertificate – ディストリビューションが、HTTPS 接続の確立に使用した SSL/TLS 証明書と一致しません。
 - InvalidRequestHeader – リクエストに無効なヘッダーが含まれていました。
 - InvalidRequestMethod – ディストリビューションは、使用された HTTP リクエストメソッドを処理するように設定されていません。これは、ディストリビューションがキャッシュ可能なリクエストのみをサポートしている場合に発生します。
 - OriginCommError – オリジンに接続中、またはオリジンからデータを読み取るときに、リクエストがタイムアウトしました。
 - OriginConnectError – サーバーがオリジンに接続できませんでした。
 - OriginContentRangeLengthError – オリジンのレスポンスの Content-Length ヘッダーが、Content-Range ヘッダーの長さとは一致しません。
 - OriginDnsError – サーバーがオリジンのドメイン名を解決できませんでした。

- `OriginError` – オリジンが誤ったレスポンスを返しました。
- `OriginHeaderTooBigError` – オリジンから返されたヘッダーが大きすぎてエッジサーバーで処理できません。
- `OriginInvalidResponseError` – オリジンが無効なレスポンスを返しました。
- `OriginReadError` – サーバーがオリジンから読み取れませんでした。
- `OriginWriteError` – サーバーがオリジンに書き込めませんでした。
- `OriginZeroSizeObjectError` – オリジンから送信されたサイズゼロのオブジェクトがエラーになりました。
- `SlowReaderOriginError` – オリジンエラーの原因となったメッセージの読み取りに時間がかかりました。

30 `sc-content-type`

レスポンスの HTTP Content-Type ヘッダーの値。

31 `sc-content-len`

レスポンスの HTTP Content-Length ヘッダーの値。

32 `sc-range-start`

レスポンスに HTTP Content-Range ヘッダーが含まれている場合、このフィールドには範囲の開始値が含まれます。

33 `sc-range-end`

レスポンスに HTTP Content-Range ヘッダーが含まれている場合、このフィールドには範囲の終了値が含まれます。

ディストリビューションのログファイルの例を以下に示します。

```
#Version: 1.0
#Fields: date time x-edge-location sc-bytes c-ip cs-method cs(Host) cs-uri-stem sc-
status cs(Referer) cs(User-Agent) cs-uri-query cs(Cookie) x-edge-result-type x-edge-
request-id x-host-header cs-protocol cs-bytes time-taken x-forwarded-for ssl-protocol
ssl-cipher x-edge-response-result-type cs-protocol-version fle-status fle-encrypted-
fields c-port time-to-first-byte x-edge-detailed-result-type sc-content-type sc-
content-len sc-range-start sc-range-end
2019-12-04 21:02:31 LAX1 392 192.0.2.100 GET d111111abcdef8.cloudfront.net /
index.html 200 - Mozilla/5.0%20(Windows%20NT%2010.0;%20Win64;
%20x64)%20AppleWebKit/537.36%20(KHTML,%20like
```

```
%20Gecko)%20Chrome/78.0.3904.108%20Safari/537.36 - - Hit
S0X4xwn4XV6Q4rgb7XiVG0Hms_BG1TAC4KyHmureZmBNrjGdRLiNIQ== d111111abcdef8.cloudfront.net
https 23 0.001 - TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256 Hit HTTP/2.0 - - 11040 0.001 Hit
text/html 78 - -
2019-12-04 21:02:31 LAX1 392 192.0.2.100 GET d111111abcdef8.cloudfront.net /
index.html 200 - Mozilla/5.0%20(Windows%20NT%2010.0;%20Win64;
%20x64)%20AppleWebKit/537.36%20(KHTML,%20like
%20Gecko)%20Chrome/78.0.3904.108%20Safari/537.36 - - Hit
k6WGMNkEzR5BEM_SaF47gjtX9zBD02m3490Y2an0QPEaUum1Z0Lrow== d111111abcdef8.cloudfront.net
https 23 0.000 - TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256 Hit HTTP/2.0 - - 11040 0.000 Hit
text/html 78 - -
2019-12-04 21:02:31 LAX1 392 192.0.2.100 GET d111111abcdef8.cloudfront.net /
index.html 200 - Mozilla/5.0%20(Windows%20NT%2010.0;%20Win64;
%20x64)%20AppleWebKit/537.36%20(KHTML,%20like
%20Gecko)%20Chrome/78.0.3904.108%20Safari/537.36 - - Hit
f37nTMVvnKvV2ZSvEsivup_c2kZ7VXzYdjC-GUQZ5qNs-89BlWazbw== d111111abcdef8.cloudfront.net
https 23 0.001 - TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256 Hit HTTP/2.0 - - 11040 0.001 Hit
text/html 78 - -
2019-12-13 22:36:27 SEA19-C1 900 192.0.2.200 GET d111111abcdef8.cloudfront.net /
favicon.ico 502 http://www.example.com/ Mozilla/5.0%20(Windows
%20NT%2010.0;%20Win64;%20x64)%20AppleWebKit/537.36%20(KHTML,
%20like%20Gecko)%20Chrome/78.0.3904.108%20Safari/537.36 - - Error
1pkpNfBQ39sYMnjjUQjmH2w1wdJnbHYTbag21o_30fcQgPzdL2RSSQ== www.example.com http 675
0.102 - - - Error HTTP/1.1 - - 25260 0.102 OriginDnsError text/html 507 - -
2019-12-13 22:36:26 SEA19-C1 900 192.0.2.200 GET d111111abcdef8.cloudfront.net / 502
- Mozilla/5.0%20(Windows%20NT%2010.0;%20Win64;%20x64)%20AppleWebKit/537.36%20(KHTML,
%20like%20Gecko)%20Chrome/78.0.3904.108%20Safari/537.36 - - Error
3AqrZGcNf_g0-5K0vfA7c9XLcf4YGvMFSeFdIetR1N_2y8jSis8Zxg== www.example.com http 735
0.107 - - - Error HTTP/1.1 - - 3802 0.107 OriginDnsError text/html 507 - -
2019-12-13 22:37:02 SEA19-C2 900 192.0.2.200 GET d111111abcdef8.cloudfront.net / 502
- curl/7.55.1 - - Error kBkDzGnceVtWHqSCqBUqtA_cEs2T3tFUBbnBNkB9E1_uVRhHgcZfcw==
www.example.com http 387 0.103 - - - Error HTTP/1.1 - - 12644 0.103 OriginDnsError
text/html 507 - -
```

標準ログの料金

標準ログ記録は、のオプション機能です CloudFront。標準ログ記録を有効にしても追加料金はかかりません。ただし、Amazon S3 でのファイルの保存とアクセス用に通常の Amazon S3 料金が発生します (ファイルはいつでも削除できます)。

Amazon S3 の料金に関する詳細については、「[Amazon S3 の料金](#)」を参照してください。

CloudFront 料金の詳細については、「[CloudFront の料金](#)」を参照してください。

リアルタイムログ

CloudFront リアルタイムログを使用すると、ディストリビューションに対して行われたリクエストに関する情報をリアルタイムで取得できます (ログはリクエストを受信してから数秒以内に配信されます)。リアルタイムログを使用して、コンテンツ配信のパフォーマンスに基づいて監視、分析、アクションを実行できます。

CloudFront リアルタイムログは設定可能です。以下を選択することができます。

- リアルタイムログのサンプリング率 (リアルタイムのログ記録を受信するリクエストの割合) を選択できます。
- ログレコードで受信する特定のフィールド。
- リアルタイムログを受信する特定のキャッシュ動作 (パスパターン)。

CloudFront リアルタイムログは、Amazon Kinesis Data Streams で選択したデータストリームに配信されます。独自の [Kinesis データストリームコンシューマー](#) を構築するか、Amazon Data Firehose を使用して Amazon Simple Storage Service (Amazon S3)、Amazon Redshift、Amazon OpenSearch Service (OpenSearch サービス)、またはサードパーティーのログ処理サービスにログデータを送信できます。

CloudFront は、Kinesis Data Streams の使用に対して発生する料金に加えて、リアルタイムログの料金を請求します。料金の詳細については、[「Amazon CloudFront 料金表」](#) および [「Amazon Kinesis Data Streams 料金表」](#) を参照してください。

Important

ログを使用して、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、コンテンツに対するリクエストの性質を理解することをお勧めします。はベストエフォートベースでリアルタイムログ CloudFront を提供します。特定のリクエストのログエントリが、リクエストが実際に処理されてからかなり後に配信されることも、(まれに) 一切配信されないこともあります。ログエントリをリアルタイムログから省略すると、リアルタイムログ内のエントリ数は AWS の請求と使用状況レポートに表示される使用量と一致しなくなります。

リアルタイムログ設定について

CloudFront リアルタイムログを使用するには、まずリアルタイムログ設定を作成します。リアルタイムログ設定には、受信するログフィールド、ログレコードの サンプリングレート、およびログを配信する Kinesis データストリームに関する情報が含まれます。

具体的には、リアルタイムログ設定には、次の設定が含まれます。

- [名前](#)
- [サンプリングレート](#)
- [フィールド](#)
- [エンドポイント \(Kinesis データストリーム\)](#)
- [IAM ロール](#)

名前

リアルタイムログ設定を識別する名前。

サンプリングレート

サンプリングレートは、リアルタイムログレコードとして Kinesis Data Streams に送信されるビューワーリクエストの割合を決定する 1 ~ 100 の整数です。すべてのビューワーリクエストをリアルタイムログに含めるには、サンプリングレートに 100 を指定します。リクエストデータの代表的なサンプルをリアルタイムログに受信しながら、コストを削減するために、より低いサンプリングレートを選択することもできます。

フィールド

各リアルタイムログレコードに含まれるフィールドのリスト。各ログレコードには、最大 40 個のフィールドを含めることができます。使用可能なすべてのフィールドを受信するか、パフォーマンスのモニタリングと分析に必要なフィールドのみを受信するかを選択できます。

次のリストは、各フィールド名と、そのフィールドに保持される情報の説明を示しています。フィールドは、Kinesis Data Streams に配信されるログレコードに表示される順序で示されています。

1. **timestamp**

エッジサーバーがリクエストへの応答を終了した日時。

2. **c-ip**

リクエスト元のビューワーの IP アドレス (192.0.2.183 または 2001:0db8:85a3::8a2e:0370:7334 など)。ビューワーが HTTP プロキシまたはロードバランサーを使用してリクエストを送った場合、このフィールドの値はプロキシまたはロードバランサーの IP アドレスです。x-forwarded-for フィールドも参照してください。

3. **time-to-first-byte**

サーバー上で測定される、要求を受信してから応答の最初のバイトを書き込むまでの秒数。

4. **sc-status**

サーバーのレスポンスの HTTP ステータスコード (例: 200)。

5. **sc-bytes**

サーバーがリクエストに応じてビューワーに送信したデータ (ヘッダーを含む) のバイトの合計数。WebSocket 接続の場合、これは接続を介してサーバーからクライアントに送信された合計バイト数です。

6. **cs-method**

ビューワーから受信した HTTP リクエストメソッド。

7. **cs-protocol**

ビューワーリクエストのプロトコル (http、https、ws、wss のいずれか)。

8. **cs-host**

ビューワーが、このリクエストの Host ヘッダーに追加した値。オブジェクト URL で CloudFront ドメイン名 (d1111111abcdef8.cloudfront.net など) を使用している場合、このフィールドにはそのドメイン名が含まれます。URLs 代替ドメイン名 (CNAME) をオブジェクト URL (www.example.com) に使用している場合、このフィールドにはその代替ドメイン名が含まれません。

9. **cs-uri-stem**

クエリ文字列 (存在する場合) を含むが、ドメイン名を含まないリクエスト URL 全体。たとえば、/images/cat.jpg?mobile=true と指定します。

Note

[標準ログ](#)では、cs-uri-stem 値にクエリ文字列は含まれません。

10.cs-bytes

ビューワーがリクエストに含めたデータ (ヘッダーを含む) のバイトの合計数。WebSocket 接続の場合、これはクライアントから接続上のサーバーに送信された合計バイト数です。

11x-edge-location

リクエストを処理したエッジロケーション。各エッジロケーションは、3 文字コードと、割り当てられた任意の数字で識別されます (例: DFW3)。通常、この 3 文字コードは、エッジロケーションの地理的場所の近くにある空港の、国際航空運送協会 (IATA) の空港コードに対応します。(これらの略語は今後変更される可能性があります)。

12x-edge-request-id

リクエストを一意に識別する不透明な文字列。CloudFront はx-amz-cf-idレスポンスヘッダーでもこの文字列を送信します。

13x-host-header

CloudFront デイストリビューションのドメイン名 (d111111abcdef8.cloudfront.net など)。

14.time-taken

サーバーが、ビューワーのリクエストを受信してからレスポンスの最後のバイトを出力キューに書き込むまでの秒数。サーバーで 1,000 分の 1 秒単位まで測定されます (例: 0.082)。ビューワーから見た場合、レスポンス全体を取得する合計所要時間は、ネットワークのレイテンシーと TCP バッファリングにより、この値よりも長くなります。

15.cs-protocol-version

ビューワーがリクエストで指定した HTTP バージョン。指定できる値には、HTTP/0.9、HTTP/1.0、HTTP/1.1、HTTP/2.0および HTTP/3.0 があります。

16c-ip-version

リクエストの IP バージョン (IPv4 または IPv6)。

17.cs-user-agent

リクエスト内の User-Agent ヘッダーの値。User-Agent ヘッダーでリクエスト元 (リクエスト元のデバイスとブラウザのタイプなど) が識別されます。リクエスト元が検索エンジンの場合は、どの検索エンジンかも識別されます。

18.cs-referer

リクエスト内の `Referer` ヘッダーの値。これはリクエスト元のドメインの名前です。一般的なリファラーとして、検索エンジン、オブジェクトに直接リンクされた他のウェブサイト、ユーザー自身のウェブサイトなどがあります。

19.cs-cookie

名前と値のペアおよび関連属性を含む、リクエスト内の `Cookie` ヘッダー。

Note

このフィールドは 800 バイトに切り捨てられます。

20.cs-uri-query

リクエスト URL のクエリ文字列の部分 (ある場合)。

21.x-edge-response-result-type

ビューワーにレスポンスを返す直前にサーバーがレスポンスを分類した方法。x-edge-result-type フィールドも参照してください。以下に示しているのは、可能な値です。

- `Hit` – サーバーがキャッシュからビューワーにオブジェクトを渡しました。
- `RefreshHit` – サーバーはキャッシュ内でオブジェクトを検出しましたが、オブジェクトの有効期限が切れていたため、サーバーはオリジンに問い合わせ、キャッシュ内に最新バージョンのオブジェクトがあるかどうかを確認しました。
- `Miss` – キャッシュ内のオブジェクトでリクエストに対応できなかったため、サーバーはリクエストをオリジンサーバーに転送して結果をビューワーに返しました。
- `LimitExceeded` – CloudFront クォータ (以前は制限と呼ばれていました) を超えたため、リクエストは拒否されました。
- `CapacityExceeded` – リクエストの受信時にサーバーの容量不足でオブジェクトを渡すことができなかったために、サーバーから 503 エラーが返されました。
- `Error` – 通常、これはリクエストがクライアントエラーとなった (`sc-status` フィールドが 4xx 範囲内の値となる)、またはサーバーエラーになった (`sc-status` フィールドが 5xx 範囲内の値となる) ことを意味します。

x-edge-result-type フィールドの値が `Error` であり、このフィールドの値が `Error` でない場合、ダウンロードが完了する前にクライアントが切断されました。

- `Redirect` – サーバーは、ディストリビューション設定に従って HTTP から HTTPS にビューワーをリダイレクトしました。

22x-forwarded-for

ビューワーが HTTP プロキシまたはロードバランサーを使用してリクエストを送信した場合、`c-ip` フィールドの値はプロキシまたはロードバランサーの IP アドレスです。この場合、このフィールドはリクエスト元のビューワーの IP アドレスです。このフィールドには、IPv4 アドレス (192.0.2.183 など) または IPv6 アドレス (2001:0db8:85a3::8a2e:0370:7334 など) が含まれます。

23ssl-protocol

リクエストが HTTPS を使用した場合、このフィールドには、リクエストとレスポンスを送信するためにビューワーとサーバーがネゴシエートした SSL/TLS プロトコルが含まれます。指定可能な値のリストについては、[ビューワーとの間でサポートされているプロトコルと暗号 CloudFront](#) でサポートされている SSL/TLS プロトコルを参照してください。

24ssl-cipher

リクエストが HTTPS を使用した場合、このフィールドには、リクエストとレスポンスを暗号化するためにビューワーとサーバーがネゴシエートした SSL/TLS 暗号が含まれます。使用できる値のリストについては、「[ビューワーとの間でサポートされているプロトコルと暗号 CloudFront](#)」で、サポートされている SSL/TLS 暗号化を参照してください。

25x-edge-result-type

サーバーが、最後のバイトを渡した後で、レスポンスを分類した方法。場合によっては、サーバーがレスポンスを送る準備ができたときから、サーバーがレスポンスを送り終わるまでの間に、結果タイプが変わることがあります。x-edge-response-result-type フィールドも参照してください。

例えば、HTTP ストリーミングで、サーバーがキャッシュ内でストリームのセグメントを検出するとします。そのシナリオでは、このフィールドの値は、通常 Hit になります。この場合、サーバーがセグメント全体を配信する前にビューワーが接続を閉じると、最終結果タイプ (およびこのフィールドの値) は Error になります。

WebSocket コンテンツはキャッシュできず、オリジンに直接プロキシされるため、接続のMissこのフィールドの値は になります。

以下に示しているのは、可能な値です。

- Hit – サーバーがキャッシュからビューワーにオブジェクトを渡しました。

- RefreshHit – サーバーはキャッシュ内でオブジェクトを検出しましたが、オブジェクトの有効期限が切れていたため、サーバーはオリジンに問い合わせ、キャッシュ内に最新バージョンのオブジェクトがあるかどうかを確認しました。
- Miss – キャッシュ内のオブジェクトでリクエストに対応できなかったため、サーバーはリクエストをオリジンに転送して結果をビューワーに返しました。
- LimitExceeded – CloudFront クォータ (以前は制限と呼ばれていました) を超えたため、リクエストは拒否されました。
- CapacityExceeded – リクエストの受信時にサーバーの容量不足でオブジェクトを渡すことができなかったために、サーバーから HTTP 503 ステータスコードが返されました。
- Error – 通常、これはリクエストがクライアントエラーとなった (sc-status フィールドが 4xx 範囲内の値となる)、またはサーバーエラーになった (sc-status フィールドが 5xx 範囲内の値となる) ことを意味します。sc-status フィールドの値が 200 であるか、このフィールドの値が Error で、x-edge-response-result-type フィールドの値が Error でない場合は、HTTP リクエストは成功したが、クライアントがすべてのバイトを受信する前に切断されたことを意味します。
- Redirect – サーバーは、ディストリビューション設定に従って HTTP から HTTPS にビューワーをリダイレクトしました。

26.fle-encrypted-fields

サーバーが暗号化してオリジンに転送した [フィールドレベル暗号化](#) フィールドの数。CloudFront サーバー fle-status は、データを暗号化するとき処理されたリクエストをオリジンにストリーミングするため、の値がエラーであってもこのフィールドに値を設定できます。

27.fle-status

[フィールドレベル暗号化](#) がディストリビューション用に設定されている場合、このフィールドにはリクエストボディが正常に処理されたかどうかを示すコードが含まれます。サーバーがリクエストボディを正常に処理し、指定したフィールドの値を暗号化してリクエストをオリジンに転送すると、このフィールドの値は Processed になります。x-edge-result-type の値は、この場合でもクライアント側またはサーバー側のエラーを示すことができます。

このフィールドで使用できる値は次のとおりです。

- ForwardedByContentType – コンテンツタイプが設定されていないため、サーバーは解析や暗号化を行わずにリクエストをオリジンに転送しました。

- `ForwardedByQueryArgs` – フィールドレベル暗号化の設定にないクエリ引数がリクエストに含まれているため、サーバーは解析や暗号化を行わずにリクエストをオリジンに転送しました。
- `ForwardedDueToNoProfile` – フィールドレベル暗号化の設定でプロファイルを指定しなかったため、サーバーは解析や暗号化を行わずにリクエストをオリジンに転送しました。
- `MalformedContentTypeClientError Content-Type` – ヘッダーの値が無効な形式であるため、サーバーはリクエストを拒否し、HTTP 400 ステータスコードをビューワーに返しました。
- `MalformedInputClientError` – リクエストボディが無効な形式であるため、サーバーはリクエストを拒否し、HTTP 400 ステータスコードをビューワーに返しました。
- `MalformedQueryArgsClientError` – クエリ引数が空であるか無効な形式であるため、サーバーはリクエストを拒否し、HTTP 400 ステータスコードをビューワーに返しました。
- `RejectedByContentType` – フィールドレベル暗号化の設定でコンテンツタイプを指定しなかったため、サーバーはリクエストを拒否し、HTTP 400 ステータスコードをビューワーに返しました。
- `RejectedByQueryArgs` – フィールドレベル暗号化の設定でクエリ引数を指定しなかったため、サーバーはリクエストを拒否し、HTTP 400 ステータスコードをビューワーに返しました。
- `ServerError` – オリジンサーバーがエラーを返しました。

リクエストがフィールドレベル暗号化のクォータ (以前は制限と呼ばれていました) を超えた場合、このフィールドには次のいずれかのエラーコードが含まれ、サーバーは HTTP ステータスコード 400 をビューワーに返します。フィールドレベル暗号化に関する最新のクォータのリストについては、「[フィールドレベル暗号化のクォータ](#)」を参照してください。

- `FieldLengthLimitClientError` – 暗号化されるように設定されているフィールドが最大の長さを超えています。
- `FieldNumberLimitClientError` – ディストリビューションによって暗号化されるように設定されているリクエストがフィールド数の制限を超えています。
- `RequestLengthLimitClientError` – フィールドレベル暗号化が設定されている場合にリクエストボディが最大の長さを超えています。

28sc-content-type

レスポンスの HTTP Content-Type ヘッダーの値。

29sc-content-len

レスポンスの HTTP Content-Length ヘッダーの値。

30sc-range-start

レスポンスに HTTP Content-Range ヘッダーが含まれている場合、このフィールドには範囲の開始値が含まれます。

31sc-range-end

レスポンスに HTTP Content-Range ヘッダーが含まれている場合、このフィールドには範囲の終了値が含まれます。

32c-port

閲覧者からのリクエストのポート番号。

33x-edge-detailed-result-type

このフィールドには、以下の場合を除き、x-edge-result-type フィールドと同じ値が含まれます。

- オブジェクトが [Origin Shield](#) レイヤーからビューワーに渡された場合、このフィールドには OriginShieldHit が含まれています。
- オブジェクトが CloudFront キャッシュに存在せず、レスポンスが [オリジンリクエストの Lambda@Edge 関数](#) によって生成された場合、このフィールドには `MissGeneratedResponse` が含まれます。
- x-edge-result-type フィールドの値が `Error` の場合、このフィールドにはエラーに関する詳細情報を含む次のいずれかの値が含まれます。
 - `AbortedOrigin` – サーバーでオリジンに関する問題が発生しました。
 - `ClientCommError` – サーバーとビューワーとの通信の問題により、ビューワーへのレスポンスが中断されました。
 - `ClientGeoBlocked` – デイストリビューションは、ビューワーの地理的位置からのリクエストを拒否するように設定されています。
 - `ClientHungUpRequest` – リクエストの送信中にビューワーが途中で停止しました。
 - `Error` – エラータイプが他のどのカテゴリにも適合しないエラーが発生しました。このエラータイプは、キャッシュからのエラーレスポンスをサーバーが渡すときに発生する可能性があります。
 - `InvalidRequest` – サーバーがビューワーから無効なリクエストを受信しました。
 - `InvalidRequestBlocked` – 要求されたリソースへのアクセスがブロックされます。

- `InvalidRequestCertificate` – デイストリビューションが、HTTPS 接続の確立に使用した SSL/TLS 証明書と一致しません。
- `InvalidRequestHeader` – リクエストに無効なヘッダーが含まれていました。
- `InvalidRequestMethod` – デイストリビューションは、使用された HTTP リクエストメソッドを処理するように設定されていません。これは、デイストリビューションがキャッシュ可能なリクエストのみをサポートしている場合に発生します。
- `OriginCommError` – オリジンに接続中、またはオリジンからデータを読み取るときに、リクエストがタイムアウトしました。
- `OriginConnectError` – サーバーがオリジンに接続できませんでした。
- `OriginContentRangeLengthError` – オリジンのレスポンスの `Content-Length` ヘッダーが、`Content-Range` ヘッダーの長さとは一致しません。
- `OriginDnsError` – サーバーがオリジンのドメイン名を解決できませんでした。
- `OriginError` – オリジンが誤ったレスポンスを返しました。
- `OriginHeaderTooBigError` – オリジンから返されたヘッダーが大きすぎてエッジサーバーで処理できません。
- `OriginInvalidResponseError` – オリジンが無効なレスポンスを返しました。
- `OriginReadError` – サーバーがオリジンから読み取れませんでした。
- `OriginWriteError` – サーバーがオリジンに書き込めませんでした。
- `OriginZeroSizeObjectError` – オリジンから送信されたサイズゼロのオブジェクトがエラーになりました。
- `SlowReaderOriginError` – オリジンエラーの原因となったメッセージの読み取りに時間がかかりました。

34.c-country

ビューワの IP アドレスによって決定される、ビューワの地理的位置を表す国コード。国コードの一覧については、「[ISO 3166-1 alpha-2](#)」を参照してください。

35.cs-accept-encoding

ビューワリクエスト内の `Accept-Encoding` ヘッダーの値。

36.cs-accept

ビューワリクエスト内の `Accept` ヘッダーの値。

37.cache-behavior-path-pattern

ビューワーリクエストに一致したキャッシュ動作を識別するパスパターン。

38.cs-headers

ビューワーリクエスト内の HTTP ヘッダー (名前と値)。

Note

このフィールドは 800 バイトに切り捨てられます。

39.cs-header-names

ビューワーリクエスト内の HTTP ヘッダーの名前 (値ではない)。

Note

このフィールドは 800 バイトに切り捨てられます。

40.cs-headers-count

ビューワーリクエスト内の HTTP ヘッダーの数。

41.primary-distribution-id

継続的デプロイが有効になっている場合、この ID は現在のディストリビューションのプライマリディストリビューションを識別します。

42.primary-distribution-dns-name

継続的デプロイが有効になっている場合、この値は現在の CloudFront ディストリビューションに関連するプライマリドメイン名 (d1111111abcdef8.cloudfront.net など) を表示します。

43.origin-fbl

CloudFront とオリジン間の最初のバイトレイテンシーの秒数。

44.origin-lbl

CloudFront とオリジン間の最後のバイトレイテンシーの秒数。

45.asn

ビューワーの AS 番号 (ASN)。

エンドポイント (Kinesis データストリーム)

エンドポイントには、リアルタイムログを送信する Kinesis データストリームに関する情報が含まれています。データストリームの Amazon リソースネーム (ARN) を指定します。

Kinesis データストリームの作成の詳細については、Amazon Kinesis Data Streams 開発者ガイドの以下のトピックを参照してください。

- [コンソールを使用したストリームの管理](#)
- [AWS CLI を使用した基本的な Kinesis Data Stream オペレーションの実行](#)
- [ストリームの作成](#) (AWS SDK for Java を使用)

データストリームを作成するときは、シャードの数を指定する必要があります。次の情報を使用して、必要なシャードの数を見積もることができます。

Kinesis データストリームのシャード数を推定するには

1. CloudFront デイストリビューションが受信する 1 秒あたりのリクエスト数を計算 (または見積もり) します。

[CloudFront 使用状況レポート](#) (CloudFront コンソール内) と [CloudFront メトリクス](#)

(CloudFront および Amazon CloudWatch コンソール内) を使用して、1 秒あたりのリクエスト数を計算することができます。

2. 1 つのリアルタイムログレコードの一般的なサイズを決定します。

一般に、1 つのログレコードは約 500 バイトです。使用可能なすべてのフィールドを含む大きなレコードは、通常、約 1 KB です。

ログレコードのサイズが不明な場合は、サンプルレートを低く (1% などに) 設定して、リアルタイムログを有効化し、Kinesis Data Streams でのデータのモニタリングを使用して平均的なレコードサイズを割り出します (受信バイト数の合計をレコード数の合計で割ります)。

3. Amazon Kinesis Data Streams の料金のページの [\[Pricing calculator\]](#) (料金見積りツール) で、1 秒あたりのリクエスト (レコード) の数と 1 つのログレコードの平均レコードサイズを入力します。その後、[\[Show calculations\]](#) (計算を表示) を選択します。

料金見積りツールには、必要なシャードの数が表示されます。(見積もりコストも表示されません。)

次の例は、平均レコードサイズが 0.5 KB、1 秒あたり 50,000 リクエストの場合、50 個のシャードが必要であることを示しています。

▼ Show calculations

0.50 KB / 1024 KB to MB conversion factor = 0.00048828 MB (Record size)

0.00048828 MB x 50,000 records per sec = 24.41 MB/sec (Data ingress rate)

24.41 MB/sec (Data ingress rate) / 1 MB per second per shard ingress capacity = 24.41 shards needed for ingress

50,000 records per sec / 1000 factor for records per shard = 50.00 shards needed for records

Max (24.41 shards needed for ingress, 0 shards needed for egress, 50.000 shards needed for records) = 50.00 Number of shards

RoundUp (50.000) = 50 shards

50 shards x 730 hours in a month = 36,500.00 Shard hours per month

36,500.00 Shard hours per month x 0.015 USD = 547.50 USD

Shard hours per month cost: 547.50 USD

0.50 KB / 25 Payload Unit factor = 0.02 PUT Payload Units fraction

RoundUp (0.02) = 1 PUT Payload Units

1 PUT Payload Units x 50,000 records per sec x 2628000 seconds in a month = 131,400,000,000.00 PUT Payload Units per month

131,400,000,000.00 PUT Payload Units x 0.000000014 USD = 1,839.60 USD

PUT Payload Units per month cost: 1,839.60 USD

Extended data retention cost: 0 USD

IAM ロール

Kinesis データストリームにリアルタイムログを配信する CloudFront アクセス許可を に付与する AWS Identity and Access Management (IAM) ロール。

CloudFront コンソールでリアルタイムログ設定を作成する場合、新しいサービスロールの作成を選択して、コンソールで IAM ロールを作成させることができます。

AWS CloudFormation または CloudFront API (AWS CLI または SDK) を使用してリアルタイムログ設定を作成する場合は、IAM ロールを自分で作成し、ロール ARN を指定する必要があります。IAM ロールを自分で作成するには、次のポリシーを使用します。

IAM ロール信頼ポリシー

次の IAM ロール信頼ポリシーを使用するには、**111122223333** を AWS アカウント 数字に置き換えます。このポリシーの Condition 要素は、[が](#) のディストリビューションに代わってこのロールを引き受けることのみ CloudFront できるため、[混乱した代理問題](#)を防ぐのに役立ちます AWS アカウント。

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

```

    "Effect": "Allow",
    "Principal": {
      "Service": "cloudfront.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": "111122223333"
      }
    }
  }
]
}

```

暗号化されていないデータストリーム用の IAM ロールアクセス許可ポリシー

次のポリシーを使用するには、***arn:aws:kinesis:us-east-2:123456789012:stream/StreamName*** を Kinesis データストリームの ARN に置き換えます。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kinesis:DescribeStreamSummary",
        "kinesis:DescribeStream",
        "kinesis:PutRecord",
        "kinesis:PutRecords"
      ],
      "Resource": [
        "arn:aws:kinesis:us-east-2:123456789012:stream/StreamName"
      ]
    }
  ]
}

```

暗号化されたデータストリーム用の IAM ロールアクセス許可ポリシー

次のポリシーを使用するには、***arn:aws:kinesis:us-east-2:123456789012:stream/StreamName*** を Kinesis データストリームの ARN に置き換え、***arn:aws:kms:us-***

`east-2:123456789012:key/e58a3d0b-fe4f-4047-a495-ae03cc73d486` を の ARN に置き換えますAWS KMS key。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kinesis:DescribeStreamSummary",
        "kinesis:DescribeStream",
        "kinesis:PutRecord",
        "kinesis:PutRecords"
      ],
      "Resource": [
        "arn:aws:kinesis:us-east-2:123456789012:stream/StreamName"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKey"
      ],
      "Resource": [
        "arn:aws:kms:us-east-2:123456789012:key/e58a3d0b-fe4f-4047-a495-ae03cc73d486"
      ]
    }
  ]
}
```

リアルタイムログ設定の作成と使用

リアルタイムログ設定を使用してディストリビューションに対して行われたリクエストに関する情報をリアルタイムで取得できます (ログはリクエストを受信してから数秒以内に配信されます)。リアルタイムログ設定は、CloudFront コンソール、AWS Command Line Interface (AWS CLI)、または CloudFront API を使用して作成できます。

リアルタイムログ設定を使用するには、ディストリビューションの 1 つ以上のキャッシュ動作にアタッチします。

リアルタイムログ設定の作成 (コンソール)

リアルタイムログ設定を作成するには

1. にサインインAWS Management Consoleし、 の CloudFront コンソールでログページを開きます <https://console.aws.amazon.com/cloudfront/v4/home?#/logs>。
2. リアルタイム設定 タブを選択します。
3. [Create configuration] (設定を作成) をクリックします。
4. 名前に、設定の名前を入力します。
5. サンプリングレート には、ログレコードを受信するリクエストの割合を入力します。
6. フィールド で、リアルタイムログで受信するフィールドを選択します。
7. エンドポイント で、リアルタイムログを受信する Kinesis データストリームを 1 つ以上選択します。

Note

CloudFront リアルタイムログは、Kinesis Data Streams で指定したデータストリームに配信されます。リアルタイムログを読み取って分析するには、独自の Kinesis データストリームコンシューマーを構築できます。Firehose を使用して、Amazon S3、Amazon Redshift、Amazon OpenSearch Service、またはサードパーティーのログ処理サービスにログデータを送信することもできます。

8. IAM ロール では、新しいサービスロールの作成 を選択するか、既存のロールを選択します。ただし、IAM ロールを作成するアクセス許可が必要です。
9. (オプション) ディストリビューション で CloudFront、リアルタイムログ設定にアタッチするディストリビューションとキャッシュ動作を選択します。
10. [Create configuration] (設定を作成) をクリックします。

正常に終了すると、作成したリアルタイムログ設定の詳細がコンソールに表示されます。

詳細については、「[リアルタイムログ設定について](#)」を参照してください。

リアルタイムログ設定を作成する (AWS CLI)

AWS Command Line Interface (AWS CLI) を使用してリアルタイムログ設定を作成するには、`aws cloudfront create-realtime-log-config` コマンドを使用します。コマンドの入力パラメータは、コマンドライン入力として個別に指定せずに、入力ファイルを使用して指定できます。

リアルタイムログ設定を作成するには (入力ファイルを含む CLI)

1. 次のコマンドを使用して、`rtl-config.yaml` コマンドのすべての入力パラメータを含む `create-realtime-log-config` という名前のファイルを作成します。

```
aws cloudfront create-realtime-log-config --generate-cli-skeleton yaml-input > rtl-config.yaml
```

2. 先ほど作成した `rtl-config.yaml` という名前のファイルを開きます。ファイルを編集して、必要なリアルタイムログ設定を指定し、ファイルを保存します。次の点に注意してください。

- `StreamType` では、唯一の有効な値は、`Kinesis` です。

リアルタイムログ設定の詳細については、「[リアルタイムログ設定について](#)」を参照してください。

3. 次のコマンドを使用して、`rtl-config.yaml` ファイルの入力パラメータを使用してリアルタイムログ設定を作成します。

```
aws cloudfront create-realtime-log-config --cli-input-yaml file://rtl-config.yaml
```

成功した場合、このコマンドの出力には、先ほど作成したリアルタイムログ設定の詳細が表示されます。

リアルタイムログ設定を既存のディストリビューション (入力ファイル付き CLI) にアタッチするには

1. 次のコマンドを使用して、更新するディストリビューションの CloudFront ディストリビューション設定を保存します。`distribution_ID` をディストリビューションの ID に置き換えます。

```
aws cloudfront get-distribution-config --id distribution_ID --output yaml > dist-config.yaml
```

2. 先ほど作成した `dist-config.yaml` という名前のファイルを開きます。ファイルを編集し、リアルタイムログ設定を使用するように更新する各キャッシュ動作に次の変更を加えます。

- キャッシュ動作で、`RealtimeLogConfigArn` という名前のフィールドを追加します。フィールドの値には、このキャッシュ動作にアタッチするリアルタイムログ設定の ARN を使用します。
- ETag フィールドの名前を `IfMatch` に変更します。ただし、フィールドの値は変更しないでください。

完了したら、ファイルを保存します。

3. リアルタイムログ設定を使用するようにディストリビューションを更新するには、次のコマンドを使用します。`distribution_ID` をディストリビューションの ID に置き換えます。

```
aws cloudfront update-distribution --id distribution_ID --cli-input-yaml file://  
dist-config.yaml
```

成功した場合、コマンドの出力には、先ほど更新したディストリビューションの詳細が表示されません。

リアルタイムログ設定 (API) の作成

CloudFront API を使用してリアルタイムログ設定を作成するには、[awscli](#) を使用します [CreateRealtimeLogConfig](#)。これらの API コールで指定するパラメータの詳細については、「[リアルタイムログ設定について](#)」と、AWS SDK またはその他 API クライアントの API リファレンスドキュメントを参照してください。

リアルタイムログ設定を作成したら、次の API コールのいずれかを使用して、それをキャッシュ動作にアタッチできます。

- 既存のディストリビューションのキャッシュ動作にアタッチするには、[awscli](#) を使用します [UpdateDistribution](#)。
- 新しいディストリビューションのキャッシュ動作にアタッチするには、[awscli](#) を使用します [CreateDistribution](#)。

これらの API コールの両方について、キャッシュ動作内で、`RealtimeLogConfigArn` フィールドにリアルタイムログ設定の ARN を指定します。これらの API コールで指定する他のフィールドの詳細については、「[ディストリビューションを作成または更新する場合に指定する値](#)」と、AWS SDK またはその他 API クライアントの API リファレンスドキュメントを参照してください。

Kinesis Data Streams コンシューマーの作成

リアルタイムログを読み取って分析するには、Kinesis Data Streams コンシューマーを構築または使用します。CloudFront リアルタイムログのコンシューマーを構築する場合、すべてのリアルタイムログレコードのフィールドは、[フィールド](#) セクションに記載されている順序と同じ順序で常に配信されることを理解することが重要です。この固定注文に対応するためにコンシューマーを構築することを確認してください。

たとえば、time-to-first-byte、sc-status、および c-country の 3 つのフィールドのみを含むリアルタイムログ設定を考えてみます。このシナリオでは、最後のフィールド、c-country は、すべてのログレコードで常にフィールド番号 3 です。ただし、後でリアルタイムログ設定にフィールドを追加すると、レコード内の各フィールドの配置が変更される可能性があります。

たとえば、フィールド sc-bytes と time-taken をリアルタイムログ設定に追加した場合、これらのフィールドは、[フィールド](#) セクションに示されている順序に従って各ログレコードに挿入されます。5 つのフィールドすべての順序は time-to-first-byte、sc-status、sc-bytes、time-taken、および c-country です。c-country フィールドはもともとフィールド番号 3 でしたが、現在はフィールド番号 5 です。リアルタイムログ設定にフィールドを追加する場合は、コンシューマーアプリケーションがログレコード内の位置を変更するフィールドを処理できることを確認してください。

リアルタイムログのトラブルシューティング

リアルタイムログ設定を作成した後、レコードが Kinesis Data Streams にまったく配信されない (または一部のレコードが配信されない) 場合があります。この場合、まずディストリビューションがビューワーリクエストを受信し CloudFront ていることを確認する必要があります。その場合は、次の設定を確認してトラブルシューティングを続行できます。

IAM ロールのアクセス許可

リアルタイムログレコードを Kinesis データストリームに配信するために、はリアルタイムログ設定で IAM ロール CloudFront を使用します。ロールの信頼ポリシーとロールのアクセス許可ポリシーが、[IAM ロール](#) に示されているポリシーと一致していることを確認してください。

Kinesis Data Streams のスロットリング

がリアルタイムログレコードを Kinesis データストリームに CloudFront 書き込む速度がストリームが処理できる速度よりも速い場合、Kinesis Data Streams はからのリクエストをスロットリングすることがあります CloudFront。この場合、Kinesis データストリームのシャードの数を増や

することができます。各シャードは、1 秒あたり 1,000 レコードまでの書き込みをサポートし、1 秒あたり 1 MB の最大データ書き込みをサポートします。

エッジ関数のログ

Amazon CloudWatch Logs を使用して、Lambda@Edge と CloudFront Functions の両方の[エッジ関数](#)のログを取得できます。コンソールまたは CloudWatch Logs API を使用して CloudWatch ログにアクセスします。

Important

ログを使用して、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、コンテンツに対するリクエストの性質を理解することをお勧めします。CloudFront はベストエフォートベースでエッジ関数ログを提供します。特定のリクエストのログエントリが、リクエストが実際に処理されてからかなり後に配信されることも、(まれに) 一切配信されないこともあります。ログエントリをエッジ関数のログから省略すると、エッジ関数のログ内のエントリ数は AWS の請求と使用状況レポートに表示される使用量と一致しなくなります。

Lambda@Edge のログ

Lambda@Edge は関数ログを CloudWatch Logs に自動的に送信し、関数が実行されるリージョンにロググループを作成します。ロググループ名は `/aws/lambda/us-east-1.function-name` の形式です。ここで、*function-name* は作成時に関数に付けた名前であり、*us-east-1* は関数が実行された AWS リージョンのリージョンコードです。

Note

Lambda@Edge は、リクエストのボリュームとログのサイズに基づいてログを調整します。

Lambda@Edge 関数の CloudWatch ログファイルを表示するには AWS リージョン、正しいのログファイルを確認する必要があります。Lambda@Edge 関数が実行されているリージョンを表示するには、CloudFront コンソールで関数のメトリクスのグラフを表示します。メトリクスは AWS リージョンごとに表示されます。同じページで、リージョンを選択してそのリージョンのログファイルを表示し、問題を調査することができます。

Lambda@Edge 関数で CloudWatch Logs を使用方法の詳細については、以下を参照してください。

- CloudFront コンソールのモニタリングセクションでグラフを表示する方法の詳細については、「」を参照してください [the section called “Amazon による CloudFront メトリクスのモニタリング CloudWatch”](#)。
- CloudWatch ログにデータを送信するために必要なアクセス許可については、「」を参照してください [the section called “IAM のアクセス許可とロールの設定”](#)。
- Lambda@Edge 関数のログ作成の追加については、AWS Lambda デベロッパーガイドの「[Node.js の AWS Lambda 関数ログ作成](#)」または「[Python の AWS Lambda 関数ログ作成](#)」を参照してください。
- CloudWatch Logs クォータ (以前は制限と呼ばれていました) の詳細については、「Amazon Logs ユーザーガイド [CloudWatch](#)」の「[ログクォータ](#)」を参照してください。 CloudWatch

CloudFront 関数ログ

CloudFront 関数のコードに `console.log()` ステートメントが含まれている場合、CloudFront Functions は自動的にこれらのログ行を CloudWatch Logs に送信します。`console.log()` ステートメントがない場合、CloudWatch ログには何も送信されません。

CloudFront 関数は、関数を実行したエッジロケーションに関係なく、常に米国東部 (バージニア北部) リージョン (us-east-1) にログストリームを作成します。ロググループ名の形式は `/aws/cloudfront/function/FunctionName`、*FunctionName* は関数を作成したときに指定した名前です。ログストリーム名の形式は `YYYY/M/D/UUID` です。

ログに送信される CloudWatch ログメッセージの例を次に示します。各行は、CloudFront リクエストを一意に識別する ID で始まります。メッセージは、CloudFront デイストリビューション ID を含む START 行で始まり、END 行で終わります。START 行と END 行の間には、関数の `console.log()` ステートメントによって生成されるログ行があります。

```
U7b4hR_RaxMADupvKAvr8_m9gsGXvioUggLV50yq-vmAtH8HADpjhw== START DistributionID:
E3E5D42GADAXZZ
U7b4hR_RaxMADupvKAvr8_m9gsGXvioUggLV50yq-vmAtH8HADpjhw== Example function log output
U7b4hR_RaxMADupvKAvr8_m9gsGXvioUggLV50yq-vmAtH8HADpjhw== END
```

Note

CloudFront 関数は、本番リクエストとレスポンスに回答して実行されるLIVEステージの関数に対して CloudWatch のみ、ログを に送信します。[関数をテスト](#)する場合、CloudFront は にログを送信しません CloudWatch。テスト出力には、エラー、コンピューティング使用率、関数ログ (console.log() ステートメント) に関する情報が含まれていますが、この情報は に送信されません CloudWatch。

CloudFront Functions は、AWS Identity and Access Management (IAM) [サービスにリンクされたロール](#)を使用して、アカウントの CloudWatch Logs にログを送信します。サービスにリンクされたロールは、AWS のサービスに直接リンクされた IAM ロールです。サービスにリンクされたロールは、サービスによって事前定義されており、ユーザーに代わってサービスから他の AWS のサービスを呼び出すために必要なすべてのアクセス許可が含まれています。CloudFront Functions は、と呼ばれるサービスにリンクされたロールを使用しますAWSServiceRoleForCloudFrontLogger。このロールの詳細については、「[the section called “Lambda@Edge 用のサービスにリンクされたロール”](#)」を参照してください (Lambda@Edge は同じサービスリンクされたロールを使用します)。

関数が検証エラーまたは実行エラーで失敗すると、情報は CloudFrontの[標準ログ](#)と[リアルタイムログ](#)に記録されます。エラーに関する情報は x-edge-result-type、x-edge-response-result-type、x-edge-detailed-result-type の各フィールドに記録されます。

AWS CloudTrail を使用して API に送信されたリクエストを CloudFront キャプチャする

CloudFront は と統合されています。このAWSサービスは CloudTrail、IAM ユーザーを含め、AWS アカウントによって CloudFront API に送信されるすべてのリクエストに関する情報をキャプチャします。CloudTrail は、これらのリクエストのログファイルをAmazon S3、CloudFront コンソール、CloudFront API、AWS SDKs、CloudFront CLI、または などの別のサービスを使用して行われたかどうかにかかわらず、すべてのリクエストに関する情報を CloudTrail キャプチャしますAWS CloudFormation。

CloudTrail ログファイルの情報を使用して、 に対して行われたリクエスト CloudFront、各リクエストが行われた送信元 IP アドレス、リクエストの実行者、リクエストの実行日時などを判断できます。設定および有効化の方法など CloudTrail、 の詳細については、[AWS CloudTrail 「ユーザーガイド」](#)を参照してください。

Note

CloudFront はグローバルサービスです。CloudTrail ログで CloudFront リクエストを表示するには、既存の証跡を更新してグローバルサービスを含める必要があります。詳細については、AWS CloudTrail ユーザーガイドの「[証跡の更新](#)」と「[グローバルサービスイベントについて](#)」を参照してください。

トピック

- [CloudFront 内の情報 CloudTrail](#)
- [CloudFront ログファイルエントリについて](#)

CloudFront 内の情報 CloudTrail

CloudTrail AWSアカウントを作成すると、[ガ](#)アカウントで有効になります。でアクティビティが発生すると CloudFront、そのアクティビティは CloudTrail イベント履歴の他のAWSサービスイベントとともに イベントに記録されます。AWS アカウントで最近のイベントを表示、検索、ダウンロードできます。CloudFront はグローバルサービスであるため、サービスのイベントは米国東部 (バージニア北部) に記録されます。詳細については、[「イベント履歴での CloudTrail イベントの表示」](#)を参照してください。

のイベントなど、AWSアカウントのイベントの継続的な記録については CloudFront、証跡を作成します。証跡にはグローバルサービスイベントを含める必要があります。証跡により、はログファイル CloudTrail を Amazon S3 バケットに配信できます。デフォルトでは、コンソールで証跡を作成する場合、証跡がすべてのリージョンに適用され、グローバルサービスイベントが含まれます。証跡は、AWS パーティションのすべてのリージョンからのイベントをログに記録し、ユーザー指定の Amazon S3 バケットにログファイルを配信します。さらに、CloudTrail ログで収集されたデータをより詳細に分析し、それに基づく対応を行うように他の AWS サービスを設定できます。詳細については、以下をご覧ください。

- [証跡を作成するための概要](#)
- [CloudTrail サポートされているサービスと統合](#)
- [の Amazon SNS 通知の設定 CloudTrail](#)
- [複数のリージョンからの CloudTrail ログファイルの受信と複数のアカウントからの CloudTrail ログファイルの受信](#)

すべての CloudFront API アクションは、[「Amazon CloudFront API リファレンス」](#)によってログに記録 CloudTrail され、[「Amazon CloudFront API リファレンス」](#)に記載されています。例えば、`ListInvalidations` APIs を呼び出す `GetDistribution` と `CreateDistribution`、CloudTrail ログファイルにエントリが生成されません。

各イベントまたはログエントリには、誰がリクエストを生成したかという情報が含まれます。同一性情報は次の判断に役立ちます。

- リクエストが、ルートと IAM ユーザー認証情報のどちらを使用して送信されたか。
- リクエストがロールまたはフェデレーションユーザーの一時的なセキュリティ認証情報を使用して行われたかどうか。
- リクエストが、別の AWS サービスによって送信されたかどうか。

詳細については、[「CloudTrail userIdentity 要素」](#)を参照してください。

CloudFront ログファイルエントリについて

各 JSON 形式の CloudTrail ログファイルには、1 つ以上のログエントリを含めることができます。各ログエントリは任意の送信元からの単一のリクエストを表し、パラメータやアクションの日時など、リクエストされたアクションに関する情報を含みます。ログエントリは、特定の順序で生成されるわけではなく、API 呼び出しのスタックトレース順に並んではいません。

`eventName` 要素は、発生したアクションと、そのアクションを実行するときに使用された API バージョンを特定します。たとえば、以下の `eventName` 値は、ディストリビューションが更新され、そのアクションを実行するときに 2014-01-31 という API バージョンが使用されたことを示しています。

`UpdateDistribution2014_01_31`

次の例は、5 つのアクションを示す CloudTrail ログエントリを示しています。

- ディストリビューション設定の更新。 `eventName` の値は `UpdateDistribution` です。
- 現在のアカウントに関連付けられているディストリビューションの一覧取得。 `eventName` の値は `ListDistributions` です。
- 特定のディストリビューションの設定の取得。 `eventName` の値は `GetDistribution` です。
- 無効化バッチリクエストの作成。 `eventName` の値は `CreateInvalidation` です。
- 現在のアカウントに関連付けられているオリジンアクセス ID の一覧取得。 `eventName` の値は `ListCloudFrontOriginAccessIdentities` です。

```
{
  "Records": [{
    "eventVersion": "1.01",
    "userIdentity": {
      "type": "IAMUser",
      "principalId": "A1B2C3D4E5F6G7EXAMPLE",
      "arn": "arn:aws:iam::111122223333:user/smithj",
      "accountId": "111122223333",
      "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
      "userName": "smithj"
    },
    "eventTime": "2014-05-06T18:00:32Z",
    "eventName": "UpdateDistribution2014_01_31",
    "sourceIPAddress": "192.0.2.17",
    "userAgent": "aws-sdk-ruby/1.39.0 ruby/1.9.3 x86_64-linux",
    "requestParameters": {
      "id": "EDFDVBD6EXAMPLE",
      "ifMatch": "E9LHASXEXAMPLE",
      "distributionConfig": {
        "restrictions": {
          "geoRestriction": {
            "quantity": 0,
            "restrictionType": "none"
          }
        },
        "customErrorResponses": {
          "quantity": 0
        },
        "defaultRootObject": "index.html",
        "aliases": {
          "quantity": 1,
          "items": ["example.com"]
        },
        "logging": {
          "bucket": "",
          "enabled": false,
          "prefix": "",
          "includeCookies": false
        },
        "viewerCertificate": {
          "iAMCertificateId": "A1B2C3D4E5F6G7EXAMPLE",
          "sSSLSupportMethod": "sni-only"
        }
      },
    }
  ]
}
```

```
"callerReference": "2014-05-06 64832",
"defaultCacheBehavior": {
  "targetOriginId": "Images",
  "allowedMethods": {
    "items": ["GET",
      "HEAD"],
    "quantity": 2
  },
  "forwardedValues": {
    "cookies": {
      "forward": "none"
    },
    "queryString": false
  },
  "minTTL": 300,
  "trustedSigners": {
    "enabled": false,
    "quantity": 0
  },
  "viewerProtocolPolicy": "redirect-to-https",
  "smoothStreaming": false
},
"origins": {
  "items": [{
    "customOriginConfig": {
      "hTTPSPort": 443,
      "originProtocolPolicy": "http-only",
      "hTTPPort": 80
    },
    "domainName": "myawsbucket.s3-website-us-east-2.amazonaws.com",
    "id": "Web page origin"
  },
  {
    "customOriginConfig": {
      "hTTPSPort": 443,
      "originProtocolPolicy": "http-only",
      "hTTPPort": 80
    },
    "domainName": "myotherawsbucket.s3-website-us-west-2.amazonaws.com",
    "id": "Images"
  }
],
  "quantity": 2
},
"enabled": true,
```

```
    "cacheBehaviors": {
      "allowedMethods": {
        "items": ["GET",
          "HEAD"],
        "quantity": 2
      },
      "trustedSigners": {
        "enabled": false,
        "quantity": 0
      },
      "targetOriginId": "Web page origin",
      "smoothStreaming": false,
      "viewerProtocolPolicy": "redirect-to-https",
      "minTTL": 300,
      "forwardedValues": {
        "cookies": {
          "forward": "none"
        },
        "queryString": false
      },
      "pathPattern": "*.html"
    }],
    "quantity": 1
  },
  "priceClass": "PriceClass_All",
  "comment": "Added an origin and a cache behavior"
}
},
"responseElements": {
  "eTag": "E2QWRUHEXAMPLE",
  "distribution": {
    "domainName": "d111111abcdef8.cloudfront.net",
    "status": "InProgress",
    "distributionConfig": {
      distributionConfig response omitted
    },
    "id": "EDFDVBD6EXAMPLE",
    "lastModifiedTime": "May 6, 2014 6:00:32 PM",
    "activeTrustedSigners": {
      "quantity": 0,
      "enabled": false
    },
    "inProgressInvalidationBatches": 0
  }
}
```

```
    },
    "requestID": "4e6b66f9-d548-11e3-a8a9-73e33example",
    "eventID": "5ab02562-0fc5-43d0-b7b6-90293example"
  },
  {
    "eventVersion": "1.01",
    "userIdentity": {
      "type": "IAMUser",
      "principalId": "A1B2C3D4E5F6G7EXAMPLE",
      "arn": "arn:aws:iam::111122223333:user/smithj",
      "accountId": "111122223333",
      "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
      "userName": "smithj"
    },
    "eventTime": "2014-05-06T18:01:35Z",
    "eventName": "ListDistributions2014_01_31",
    "sourceIPAddress": "192.0.2.17",
    "userAgent": "aws-sdk-ruby/1.39.0 ruby/1.9.3 x86_64-linux",
    "requestParameters": null,
    "responseElements": null,
    "requestID": "52de9f97-d548-11e3-8fb9-4dad0example",
    "eventID": "eb91f423-6dd3-4bb0-a148-3cdfbexample"
  },
  {
    "eventVersion": "1.01",
    "userIdentity": {
      "type": "IAMUser",
      "principalId": "A1B2C3D4E5F6G7EXAMPLE",
      "arn": "arn:aws:iam::111122223333:user/smithj",
      "accountId": "111122223333",
      "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
      "userName": "smithj"
    },
    "eventTime": "2014-05-06T18:01:59Z",
    "eventName": "GetDistribution2014_01_31",
    "sourceIPAddress": "192.0.2.17",
    "userAgent": "aws-sdk-ruby/1.39.0 ruby/1.9.3 x86_64-linux",
    "requestParameters": {
      "id": "EDFDVBD6EXAMPLE"
    },
    "responseElements": null,
    "requestID": "497b3622-d548-11e3-8fb9-4dad0example",
    "eventID": "c32289c7-005a-46f7-9801-cba41example"
  },
}
```

```
{
  "eventVersion": "1.01",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "A1B2C3D4E5F6G7EXAMPLE",
    "arn": "arn:aws:iam::111122223333:user/smithj",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "smithj"
  },
  "eventTime": "2014-05-06T18:02:27Z",
  "eventName": "CreateInvalidation2014_01_31",
  "sourceIPAddress": "192.0.2.17",
  "userAgent": "aws-sdk-ruby/1.39.0 ruby/1.9.3 x86_64-linux",
  "requestParameters": {
    "invalidationBatch": {
      "callerReference": "2014-05-06 64947",
      "paths": {
        "quantity": 3,
        "items": ["/images/new.jpg",
                  "/images/logo.jpg",
                  "/images/banner.jpg"]
      }
    }
  },
  "distributionId": "EDFDVBD6EXAMPLE"
},
"responseElements": {
  "invalidation": {
    "createTime": "May 6, 2014 6:02:27 PM",
    "invalidationBatch": {
      "callerReference": "2014-05-06 64947",
      "paths": {
        "quantity": 3,
        "items": ["/images/banner.jpg",
                  "/images/logo.jpg",
                  "/images/new.jpg"]
      }
    }
  },
  "status": "InProgress",
  "id": "ISRZ85EXAMPLE"
},
  "location": "https://cloudfront.amazonaws.com/2014-01-31/distribution/EDFDVBD6EXAMPLE/invalidation/ISRZ85EXAMPLE"
},
```

```
    "requestID": "4e200613-d548-11e3-a8a9-73e33example",
    "eventID": "191ebb93-66b7-4517-a741-92b0eexample"
  },
  {
    "eventVersion": "1.01",
    "userIdentity": {
      "type": "IAMUser",
      "principalId": "A1B2C3D4E5F6G7EXAMPLE",
      "arn": "arn:aws:iam::111122223333:user/smithj",
      "accountId": "111122223333",
      "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
      "userName": "smithj"
    },
    "eventTime": "2014-05-06T18:03:08Z",
    "eventName": "ListCloudFrontOriginAccessIdentities2014_01_31",
    "sourceIPAddress": "192.0.2.17",
    "userAgent": "aws-sdk-ruby/1.39.0 ruby/1.9.3 x86_64-linux",
    "requestParameters": null,
    "responseElements": null,
    "requestID": "42ca4299-d548-11e3-8fb9-4dad0example",
    "eventID": "7aeb434f-eb55-4e2a-82d8-417d5example"
  ]
}
```

AWS Config による設定変更の追跡

を使用してAWS Config、CloudFront デイストリビューション設定の変更に関する設定変更を記録できます。例えば、デイストリビューションの状態や、料金クラス、オリジン、地域制限の設定、および Lambda@Edge 設定における変更をキャプチャできます。

Note

AWS Config は、CloudFront ストリーミングデイストリビューションのキーと値のタグを記録しません。

AWS Config で をセットアップする CloudFront

を設定するときにAWS Config、サポートされているすべてのAWSリソースを記録するか、の変更を記録するだけでなく、設定の変更を記録する特定のリソースのみを指定することもできます

CloudFront。でサポートされている特定のリソースを確認するには CloudFront、「AWS Configデベロッパーガイド」の「[サポートされているAWSリソースタイプ](#)」のリストを参照してください。

CloudFront ディストリビューションの設定変更を追跡するには、米国東部 (バージニア北部) パブリックリージョンで AWS コンソールにログインする必要があります。

 Note

AWS Config でのリソースの記録には遅延が生じる可能性があります。AWS Config は、リソースを検出した後でしかリソースを記録しません。

CloudFront を使用して AWS Configをセットアップする AWS Management Console

1. AWS Management Console にサインインして、AWS Config コンソール (<https://console.aws.amazon.com/config/>) を開きます。
2. [Get Started Now] を選択します。
3. [設定] ページの [記録するリソースタイプ] で、AWS で記録する AWS Config リソースタイプを指定します。CloudFront 変更のみを記録する場合は、特定のタイプを選択し、で変更を追跡するディストリビューションまたはストリーミングディストリビューションCloudFrontを選択します。

追跡するディストリビューションを追加あるいは変更するには、最初のステップを完了した後に左側で [設定] を選択します。

4. AWS Config で追加の必須オプションを指定する: 通知の設定、設定情報の場所の指定、リソースタイプ評価のルールの追加。

詳細については、AWS Config デベロッパーガイドの「[コンソールによる AWS Config の設定](#)」を参照してください。

AWS CLI または API CloudFront を使用して AWS Configをセットアップするには、次のいずれかを参照してください。

- AWS CLI を使用する場合: AWS Config デベロッパーガイドの「[AWS CLI を使用した AWS Config の設定](#)」
- API を使用する : AWS Config API リファレンスの [StartConfigurationRecorder](#) アクションおよびその他の情報

CloudFront 設定履歴の表示

AWS Config がディストリビューションの設定変更の記録を開始したら、CloudFront で設定したすべてのディストリビューションの設定履歴を取得できます。

設定履歴は以下のいずれかの方法で閲覧できます。

- AWS Config コンソールを使用します。記録されたリソースごとに、設定の詳細の履歴を提供するタイムラインページを表示することができます。このページを表示するには、[Dedicated Hosts] ページの [設定タイムライン] 列にあるグレーのアイコンを選択します。詳細については、『AWS Config デベロッパーガイド』の「[AWS Config コンソールでの設定詳細の表示](#)」を参照してください。
- AWS CLI コマンドを実行します。すべてのディストリビューションのリストを取得するには、[list-discovered-resources](#) コマンドを使用します。特定の時間間隔のディストリビューションの設定の詳細を取得するには、[get-resource-config-history](#) コマンドを使用します。詳細については、『AWS Config デベロッパーガイド』の「[CLI による設定詳細の表示](#)」を参照してください。
- アプリケーションで AWS Config API を使用します。すべてのディストリビューションのリストを取得するには、[ListDiscoveredResources](#) アクションを使用します。特定の時間間隔のディストリビューションの設定の詳細を取得するには、[GetResourceConfigHistory](#) アクションを使用します。詳細については、「[AWS Config API リファレンス](#)」を参照してください。

たとえば、AWS Config からすべてのディストリビューションのリストを取得するには、次のような CLI コマンドを実行できます。

```
aws configservice list-discovered-resources --resource-type
AWS::CloudFront::Distribution
```

Amazon のセキュリティ CloudFront

AWS ではクラウドセキュリティが最優先事項です。セキュリティを最も重視する組織の要件を満たすために構築された AWS のデータセンターとネットワークアーキテクチャは、お客様に大きく貢献します。

セキュリティは、AWS と顧客の間の責任共有です。[責任共有モデル](#)では、この責任がクラウドのセキュリティおよびクラウド内のセキュリティとして説明されています。

- クラウドのセキュリティ - AWS は、AWS クラウドで AWS のサービスを実行するインフラストラクチャを保護する責任を負います。また、AWS は、使用するサービスを安全に提供します。[AWS コンプライアンスプログラム](#)の一環として、サードパーティーの監査が定期的にセキュリティの有効性をテストおよび検証しています。Amazon に適用されるコンプライアンスプログラムの詳細については CloudFront、「[コンプライアンスAWSプログラムによる 対象範囲内のサービス](#)」を参照してください。
- クラウド内のセキュリティ - お客様の責任は、使用する AWS のサービスに応じて異なります。お客様は、データの機密性、組織の要件、および適用法令と規制などのその他要因に対する責任も担います。

このドキュメントは、を使用する際に責任共有モデルを適用する方法を理解するのに役立ちます CloudFront。以下のトピックでは、セキュリティおよびコンプライアンスの目的 CloudFront を達成するためにを設定する方法を示します。また、CloudFront リソースのモニタリングや保護に役立つ他の AWS のサービスの使用方法についても説明します。

トピック

- [Amazon でのデータ保護 CloudFront](#)
- [Amazon の Identity and Access Management CloudFront](#)
- [Amazon でのログ記録とモニタリング CloudFront](#)
- [Amazon のコンプライアンス検証 CloudFront](#)
- [Amazon の耐障害性 CloudFront](#)
- [Amazon のインフラストラクチャセキュリティ CloudFront](#)

Amazon でのデータ保護 CloudFront

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

データを保護するため、AWS アカウント の認証情報を保護し、AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーをセットアップすることをお勧めします。こうすると、それぞれのジョブを遂行するために必要なアクセス許可のみを各ユーザーに付与できます。また、次の方法でデータを保護することをおすすめします。

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 が必須です。TLS 1.3 が推奨されます。
- AWS CloudTrail で API とユーザーアクティビティロギングをセットアップします。
- AWS のサービス内でデフォルトである、すべてのセキュリティ管理に加え、AWS の暗号化ソリューションを使用します。
- Amazon Macie などの高度なマネージドセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API により AWS にアクセスするときに FIPS 140-2 検証済み暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「[連邦情報処理規格 \(FIPS\) 140-2](#)」を参照してください。

お客様の E メールアドレスなどの機密情報やセンシティブ情報は、タグや名前フィールドなどの自由形式のフィールドに配置しないことを強くお勧めします。これは、コンソール、API、AWS CLI または AWS SDK AWS のサービスで CloudFront または他の を使用する場合も同様です。SDKs 名前に使用する自由記述のテキストフィールドやタグに入力したデータは、課金や診断ログに使用される場合があります。外部サーバーへの URL を提供する場合は、そのサーバーへのリクエストを検証するための認証情報を URL に含めないように強くお勧めします。

Amazon CloudFront には、配信するコンテンツを保護するために使用できるオプションがいくつか用意されています。

- HTTPS 接続を設定します。
- フィールドレベルの暗号化を構成して、転送中の特定のデータのセキュリティを強化します。
- コンテンツへのアクセスを制限して、特定の人々、または特定のエリアのユーザーだけが閲覧できるようにします。

以下のトピックでオプションについて詳しく説明します。

トピック

- [転送中の暗号化](#)
- [保管中の暗号化](#)
- [コンテンツへのアクセス制限](#)

転送中の暗号化

転送中にデータを暗号化 CloudFront するには、ビューワーが HTTPS を使用してファイルをリクエストするように Amazon を設定します。これにより、がビューワーと通信するときに CloudFront 接続が暗号化されます。HTTPS CloudFront を使用してオリジンからファイルを取得するようにを設定して、がオリジンと CloudFront 通信するときに接続が暗号化されるようにすることもできます。

詳細については、「[で HTTPS を使用する CloudFront](#)」を参照してください。

フィールドレベル暗号化では、HTTPS と共にセキュリティのレイヤーが追加されます。これにより、システムの処理中に特定のデータに特定のアプリケーションのみがアクセスできるように、そのデータを保護できます。でフィールドレベル暗号化を設定することで CloudFront、ユーザーが送信した機密情報をウェブサーバーに安全にアップロードできます。クライアントが提供した機密情報は、ユーザーに近いエッジで暗号化され、アプリケーションスタック全体で暗号化されたままになります。これにより、データを必要とするアプリケーションのみが (復号化するための認証情報があれば) そのデータを復号化できます。

詳細については、「[フィールドレベル暗号化を使用した機密データの保護](#)」を参照してください。

CloudFront API エンドポイント `cloudfront.amazonaws.com` および `cloudfront-fips.amazonaws.com`、HTTPS トラフィックのみを受け入れます。つまり、CloudFront API を使用して情報を送受信する場合、ディストリビューション設定、キャッシュポリシーとオリジンリクエ

ストポリシー、キーグループとパブリックキー、CloudFront 関数の関数コードなどのデータは、転送中に常に暗号化されます。さらに、CloudFront API エンドポイントに送信されるすべてのリクエストは、AWS 認証情報で署名され、AWS CloudTrail にログインされます。

Functions の CloudFront 関数コードと設定は、エッジロケーションの Point of Presence (POPs) にコピーするとき、およびで使用される他のストレージロケーション間で、転送時に常に暗号化されます CloudFront。

保管中の暗号化

Functions の CloudFront 関数コードと設定は、エッジロケーション POPs およびで使用される他のストレージロケーションに、常に暗号化された形式で保存されます CloudFront。

コンテンツへのアクセス制限

インターネット経由でコンテンツを配信する多くの企業が、ユーザーのサブセットのドキュメント、ビジネスデータ、メディアストリーム、またはコンテンツに対して、アクセスを制限する必要があると考えています。Amazon を使用してこのコンテンツを安全に配信するには CloudFront、次のいずれかを実行します。

署名付き URL または署名付き Cookie を使用する

署名付き URLs または署名付き Cookie CloudFront を使用してこのプライベートコンテンツを提供するなど、選択したユーザーを対象としたコンテンツへのアクセスを制限できます。詳細については、「[署名付き URL と署名付き Cookie を使用したプライベートコンテンツの提供](#)」を参照してください。

Amazon S3 バケットのコンテンツへのアクセスを制限する

CloudFront 署名URLs や署名付き Cookie などを使用してコンテンツへのアクセスを制限する場合、ファイルの直接 URL を使用してファイルを表示することも望ましくありません。代わりに、CloudFront URL を使用してファイルへのアクセスのみ許可するため、正常に保護されます。

Amazon S3 バケットを CloudFront ディストリビューションのオリジンとして使用する場合は、オリジンアクセスコントロール (OAC) を設定して、S3 バケットへのアクセスを制限できます。詳細については、「[the section called “Amazon S3 オリジンへのアクセスの制限”](#)」を参照してください。

Application Load Balancer が提供するコンテンツへのアクセスを制限する

Elastic Load Balancing CloudFront の Application Load Balancer をオリジンとして使用する場合、ユーザーが Application Load Balancer に直接アクセスできない CloudFront ようにを設定

できます。これにより、ユーザーは を介してのみ Application Load Balancer にアクセスでき CloudFront、 を使用する利点が得られます CloudFront。詳細については、「[Application Load Balancers へのアクセスを制限する](#)」を参照してください。

AWS WAF ウェブ ACL の使用

ウェブアプリケーションファイアウォールサービスである AWS WAF を使用して、コンテンツへのアクセスを制限するためのウェブアクセス制御リスト (ウェブ ACL) を作成できます。リクエストの送信元の IP アドレスやクエリ文字列の値など、指定した条件に基づいて、 はリクエストされたコンテンツまたは HTTP 403 ステータスコード (禁止) でリクエスト CloudFront に対応します。詳細については、「[AWS WAF 保護の使用](#)」を参照してください。

地域制限を使用する

地域制限 (地理的ブロック) を使用すると、 CloudFront デイストリビューションを通じて配信しているコンテンツについて、特定地域のユーザーによるアクセスを回避できます。地域制限を設定するときには選択できるオプションがいくつかあります。詳細については、「[コンテンツの地理的デистриビューションの制限](#)」を参照してください。

Amazon の Identity and Access Management CloudFront

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御するために役立つ AWS のサービスです。IAM 管理者は、誰を認証 (サインイン) し、誰に CloudFront リソースの使用を承認する (アクセス許可を付与する) かを制御します。IAM は、追加費用なしで使用できる AWS のサービスです。

トピック

- [対象者](#)
- [アイデンティティによる認証](#)
- [ポリシーを使用したアクセス権の管理](#)
- [Amazon と IAM の CloudFront 連携方法](#)
- [Amazon のアイデンティティベースのポリシーの例 CloudFront](#)
- [AWS Amazon の マネージドポリシー CloudFront](#)
- [Amazon CloudFront アイデンティティとアクセスのトラブルシューティング](#)

対象者

AWS Identity and Access Management (IAM) の用途は、CloudFront で行う作業によって異なります。

サービスユーザー – CloudFront サービスを使用してジョブを実行する場合は、管理者から必要な認証情報とアクセス許可が与えられます。さらに多くの CloudFront 機能を使用して作業を行う場合は、追加のアクセス許可が必要になることがあります。アクセスの管理方法を理解しておく、管理者に適切な許可をリクエストするうえで役立ちます。CloudFront 機能にアクセスできない場合は、「[Amazon CloudFront アイデンティティとアクセスのトラブルシューティング](#)」を参照してください。

サービス管理者 – 社内の CloudFront リソースを担当している場合は、通常、へのフルアクセスがあります CloudFront。サービスのユーザーがどの CloudFront 機能やリソースにアクセスするかを決めるのは管理者の仕事です。その後、IAM 管理者にリクエストを送信して、サービスユーザーの権限を変更する必要があります。このページの情報を点検して、IAM の基本概念を理解してください。お客様の会社で IAM を利用する方法の詳細については CloudFront、「」を参照してください [Amazon と IAM の CloudFront 連携方法](#)。

IAM 管理者 - 管理者は、CloudFront へのアクセスを管理するポリシーの書き込み方法の詳細について確認する場合があります。IAM で使用できる CloudFront アイデンティティベースのポリシーの例を表示するには、「」を参照してください [Amazon のアイデンティティベースのポリシーの例 CloudFront](#)。

アイデンティティによる認証

認証とは、アイデンティティ認証情報を使用して AWS にサインインする方法です。ユーザーは、AWS アカウントのルートユーザーとして、または IAM ロールを引き受けることによって、認証済み (AWS にサインイン済み) である必要があります。

ID ソースから提供された認証情報を使用して、フェデレーテッドアイデンティティとして AWS にサインインできます。AWS IAM Identity Center フェデレーテッドアイデンティティの例としては、IAM アイデンティティセンターユーザー、会社のシングルサインオン認証、Google または Facebook の認証情報などがあります。フェデレーテッドアイデンティティとしてサインインする場合、IAM ロールを使用して、前もって管理者により ID フェデレーションが設定されています。フェデレーションを使用して AWS にアクセスする場合、間接的にロールを引き受けることとなります。

ユーザーのタイプに応じて、AWS Management Console または AWS アクセスポータルにサインインできます。AWS へのサインインの詳細については、『AWS サインイン ユーザーガイド』の「[AWS アカウントにサインインする方法](#)」を参照してください。

プログラムで AWS にアクセスする場合、AWS は Software Development Kit (SDK) とコマンドラインインターフェイス (CLI) を提供し、認証情報でリクエストに暗号で署名します。AWS ツールを使用しない場合は、リクエストに自分で署名する必要があります。リクエストに署名する推奨方法の使用については、『IAM ユーザーガイド』の「[AWS API リクエストの署名](#)」を参照してください。

使用する認証方法を問わず、追加のセキュリティ情報の提供が求められる場合もあります。例えば、AWS では多要素認証 (MFA) を使用してアカウントのセキュリティを高めることを推奨しています。詳細については、「AWS IAM Identity Center ユーザーガイド」の「[多要素認証](#)」および「IAM ユーザーガイド」の「[AWS での多要素認証 \(MFA\) の使用](#)」を参照してください。

AWS アカウントのルートユーザー

AWS アカウントを作成する場合、このアカウントのすべての AWS のサービスとリソースに対して完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。このアイデンティティは AWS アカウントのルートユーザーと呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサインインすることによってアクセスできます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザーの認証情報を保護し、それらを使用してルートユーザーのみが実行できるタスクを実行してください。ルートユーザーとしてサインインする必要があるタスクの完全なリストについては、「IAM ユーザーガイド」の「[ルートユーザー認証情報が必要なタスク](#)」を参照してください。

フェデレーテッド ID

ベストプラクティスとして、管理者アクセスを必要とするユーザーを含む人間のユーザーに対し、ID プロバイダーとのフェデレーションを使用して、一時的な認証情報の使用により、AWS のサービスへのアクセスを要求します。

フェデレーテッドアイデンティティは、エンタープライズユーザーディレクトリ、ウェブ ID プロバイダー、AWS Directory Service、アイデンティティセンターディレクトリのユーザーが、または ID ソースから提供された認証情報を使用して AWS のサービスにアクセスするユーザーです。フェデレーテッド ID が AWS アカウントにアクセスすると、ロールが継承され、そのロールが一時的な認証情報を提供します。

アクセスを一元管理する場合は、AWS IAM Identity Center を使用することをお勧めします。IAM アイデンティティセンターでユーザーとグループを作成するか、すべての AWS アカウントとアプリケーションで使用するために、独自の ID ソースで一連のユーザーとグループに接続して同期するこ

ともできます。IAM アイデンティティセンターの詳細については、「AWS IAM Identity Center ユーザーガイド」の「[What is IAM アイデンティティセンター?](#)」(IAM アイデンティティセンターとは)を参照してください。

IAM ユーザーとグループ

[IAM ユーザー](#) は、1人のユーザーまたは1つのアプリケーションに対して特定の許可を持つ AWS アカウント内のアイデンティティです。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を保有する IAM ユーザーを作成する代わりに、一時的な認証情報を使用することをお勧めします。ただし、IAM ユーザーでの長期的な認証情報が必要な特定のユースケースがある場合は、アクセスキーをローテーションすることをお勧めします。詳細については、「IAM ユーザーガイド」の「[長期的な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションする](#)」を参照してください。

[IAM グループ](#) は、IAM ユーザーの集団を指定するアイデンティティです。グループとしてサインインすることはできません。グループを使用して、複数のユーザーに対して一度に権限を指定できます。多数のユーザーグループがある場合、グループを使用することで権限の管理が容易になります。例えば、IAMAdmins という名前のグループを設定して、そのグループに IAM リソースを管理する権限を与えることができます。

ユーザーは、ロールとは異なります。ユーザーは1人の人または1つのアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユーザーには永続的な長期の認証情報がありますが、ロールでは一時的な認証情報が提供されます。詳細については、「IAM ユーザーガイド」の「[IAM ユーザー \(ロールではなく\) の作成が適している場合](#)」を参照してください。

IAM ロール

[IAM ロール](#) は、特定の許可を持つ、AWS アカウント内のアイデンティティです。これは IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。[ロールを切り替える](#) ことにより、AWS Management Console で一時的に IAM ロールを引き受けることができます。ロールを引き受けるには、AWS CLI または AWS API オペレーションを呼び出すか、カスタム URL を使用します。ロールを使用する方法の詳細については、「IAM ユーザーガイド」の「[IAM ロールの使用](#)」を参照してください。

IAM ロールと一時的な認証情報は、次のような状況で役立ちます。

- フェデレーションユーザーアクセス – フェデレーテッドアイデンティティに許可を割り当てるには、ロールを作成してそのロールの許可を定義します。フェデレーテッドアイデンティティが認証されると、そのアイデンティティはロールに関連付けられ、ロールで定義されている権限が

付与されます。フェデレーションの詳細については、「IAM ユーザーガイド」の「[Creating a role for a third-party Identity Provider](#)」(サードパーティーアイデンティティプロバイダー向けロールの作成)を参照してください。IAM Identity Center を使用する場合、許可セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM アイデンティティセンターは、アクセス許可セットを IAM のロールに関連付けます。アクセス許可セットの詳細については、「AWS IAM Identity Center ユーザーガイド」の「[アクセス許可セット](#)」を参照してください。

- 一時的な IAM ユーザー権限 - IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる権限を一時的に IAM ロールで引き受けることができます。
- クロスアカウントアクセス - IAM ロールを使用して、自分のアカウントのリソースへのアクセスを別のアカウントの人物(信頼できるプリンシパル)に許可できます。クロスアカウントアクセス権を付与する主な方法は、ロールを使用することです。ただし、一部の AWS のサービスでは、(ロールをプロキシとして使用する代わりに)リソースにポリシーを直接アタッチできます。クロスアカウントアクセスにおけるロールとリソースベースのポリシーの違いについては、「IAM ユーザーガイド」の「[IAM ロールとリソースベースのポリシーとの相違点](#)」を参照してください。
- クロスサービスアクセス - 一部の AWS のサービスでは、他の AWS のサービスの機能を使用します。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。サービスでは、呼び出し元プリンシパルの権限、サービスロール、またはサービスリンクロールを使用してこれを行う場合があります。
- 転送アクセスセッション (FAS) - IAM ユーザーまたはロールを使用して AWS でアクションを実行するユーザーは、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、AWS のサービスを呼び出すプリンシパルの権限を、AWS のサービスのリクエストと合わせて使用し、ダウンストリームのサービスに対してリクエストを行います。FAS リクエストは、サービスが、完了するために他の AWS のサービスまたはリソースとのやりとりを必要とするリクエストを受け取ったときにのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「[転送アクセスセッション](#)」を参照してください。
- サービスロール - サービスがユーザーに代わってアクションを実行するために引き受ける [IAM ロール](#)です。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、「IAM ユーザーガイド」の「[AWS のサービスにアクセス許可を委任するロールの作成](#)」を参照してください。
- サービスにリンクされたロール - サービスにリンクされたロールは、AWS のサービスにリンクされたサービスロールの一種です。サービスがロールを引き受け、ユーザーに代わってアクショ

ンを実行できるようになります。サービスにリンクされたロールは、AWS アカウント に表示され、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールの許可を表示できますが、編集はできません。

- Amazon EC2 で実行されているアプリケーション - EC2 インスタンスで実行され、AWS CLI または AWS API 要求を行っているアプリケーションの一時的な認証情報を管理するには、IAM ロールを使用できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。AWS ロールを EC2 インスタンスに割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスに添付されたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、「IAM ユーザーガイド」の「[Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用して許可を付与する](#)」を参照してください。

IAM ロールと IAM ユーザーのどちらを使用するかについては、「IAM ユーザーガイド」の「[\(IAM ユーザーではなく\) IAM ロールをいつ作成したら良いのか?](#)」を参照してください。

ポリシーを使用したアクセス権の管理

AWS でアクセスを制御するには、ポリシーを作成して AWS アイデンティティまたはリソースにアタッチします。ポリシーは AWS のオブジェクトであり、アイデンティティやリソースに関連付けて、これらのアクセス許可を定義します。AWS は、プリンシパル (ユーザー、ルートユーザー、またはロールセッション) がリクエストを行うと、これらのポリシーを評価します。ポリシーでの権限により、リクエストが許可されるか拒否されるかが決まります。大半のポリシーは JSON ドキュメントとして AWS に保存されます。JSON ポリシードキュメントの構造と内容の詳細については、「IAM ユーザーガイド」の「[JSON ポリシー概要](#)」を参照してください。

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

デフォルトでは、ユーザーやロールに権限はありません。IAM 管理者は、リソースで必要なアクションを実行するためのアクセス許可をユーザーに付与するため、IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き継ぐことができます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションの権限を定義します。例えば、iam:GetRole アクションを許可するポリシーがあるとします。このポリシーがあるユーザーは、AWS Management Console、AWS CLI、または AWS API からロール情報を取得できます。

アイデンティティベースポリシー

アイデンティティベースポリシーは、IAM ユーザー、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 権限ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件を制御します。アイデンティティベースのポリシーを作成する方法については、「IAM ユーザーガイド」の「[IAM ポリシーの作成](#)」を参照してください。

アイデンティティベースのポリシーは、さらに [インラインポリシー](#) または [マネージドポリシー](#) に分類できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれます。管理ポリシーは、AWS アカウント内の複数のユーザー、グループ、およびロールにアタッチできるスタンドアロンポリシーです。マネージドポリシーには、AWS マネージドポリシーとカスタマー管理ポリシーがあります。マネージドポリシーまたはインラインポリシーのいずれかを選択する方法については、「IAM ユーザーガイド」の「[マネージドポリシーとインラインポリシーの比較](#)」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロール信頼ポリシーや Amazon S3 バケットポリシーがあげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、[プリンシパルを指定する](#) 必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーションユーザー、または AWS のサービスを含めることができます。

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

アクセスコントロールリスト (ACL)

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

Amazon S3、AWS WAF、および Amazon VPC は、ACL をサポートするサービスの例です。ACL の詳細については、「Amazon Simple Storage Service デベロッパーガイド」の「[アクセスコントロールリスト \(ACL\) の概要](#)」を参照してください。

その他のポリシータイプ

AWS では、他の一般的ではないポリシータイプをサポートしています。これらのポリシータイプでは、より一般的なポリシータイプで付与される最大の許可を設定できます。

- **権限の境界** - 権限の境界は、アイデンティティベースのポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる許可の上限を設定する高度な機能です。エンティティに権限の境界を設定できます。結果として得られる権限は、エンティティのアイデンティティベースポリシーとその権限の境界の共通部分になります。Principal フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、権限の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。アクセス許可の境界の詳細については、「IAM ユーザーガイド」の「[IAM エンティティのアクセス許可の境界](#)」を参照してください。
- **サービスコントロールポリシー (SCP)** - SCP は、AWS Organizations で組織や組織単位 (OU) の最大許可を指定する JSON ポリシーです。AWS Organizations は、ユーザーのビジネスが所有する複数の AWS アカウントをグループ化し、一元的に管理するサービスです。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP はメンバーアカウントのエンティティに対する権限を制限します (各 AWS アカウントのルートユーザーなど)。Organizations と SCP の詳細については、「AWS Organizations ユーザーガイド」の「[SCP の仕組み](#)」を参照してください。
- **セッションポリシー** - セッションポリシーは、ロールまたはフェデレーションユーザーの一時的なセッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果としてセッションの権限の範囲は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポリシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合があります。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。詳細については、「IAM ユーザーガイド」の「[セッションポリシー](#)」を参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される権限を理解するのがさらに難しくなります。複数のポリシータイプが関連するとき、リクエストを許可するかどうかを AWS が決定する方法の詳細については、「IAM ユーザーガイド」の「[ポリシーの評価ロジック](#)」を参照してください。

Amazon と IAM の CloudFront 連携方法

IAM を使用してへのアクセスを管理する前に CloudFront、で利用できる IAM の機能について学びます CloudFront。

Amazon で使用できる IAM の機能 CloudFront

IAM 機能	CloudFront サポート
アイデンティティベースのポリシー	あり
リソースベースのポリシー	いいえ
ポリシーアクション	あり
ポリシーリソース	はい
ポリシー条件キー (サービス固有)	はい
ACL	なし
ABAC (ポリシー内のタグ)	部分的
一時的な認証情報	はい
転送アクセスセッション (FAS)	いいえ
サービスロール	いいえ
サービスリンクロール	はい

CloudFront およびその他の AWS のサービスがほとんどの IAM 機能と連携する方法の概要を把握するには、「IAM ユーザーガイド」の [AWS 「IAM と連携する のサービス」](#) を参照してください。

のアイデンティティベースのポリシー CloudFront

アイデンティティベースポリシーをサポートする	あり
------------------------	----

アイデンティティベースのポリシーは、IAM ユーザー、ユーザーグループ、ロールなどのアイデンティティにアタッチできる JSON アクセス許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件を制御します。アイデンティティ

ベースのポリシーを作成する方法については、「IAM ユーザーガイド」の「[IAM ポリシーの作成](#)」を参照してください。

IAM アイデンティティベースのポリシーでは、許可または拒否するアクションとリソース、およびアクションを許可または拒否する条件を指定できます。プリンシパルは、それがアタッチされているユーザーまたはロールに適用されるため、アイデンティティベースのポリシーでは指定できません。JSON ポリシーで使用できるすべての要素について学ぶには、『IAM ユーザーガイド』の「[IAM JSON ポリシーの要素のリファレンス](#)」を参照してください。

のアイデンティティベースのポリシーの例 CloudFront

CloudFront アイデンティティベースのポリシーの例を表示するには、「」を参照してください [Amazon のアイデンティティベースのポリシーの例 CloudFront](#)。

内のリソースベースのポリシー CloudFront

リソースベースのポリシーのサポート	なし
-------------------	----

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロール信頼ポリシーや Amazon S3 バケットポリシーがあげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、[プリンシパルを指定する](#) 必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーションユーザー、または AWS のサービスを含めることができます。

クロスアカウントアクセスを有効にするには、全体のアカウント、または別のアカウントの IAM エンティティを、リソースベースのポリシーのプリンシパルとして指定します。リソースベースのポリシーにクロスアカウントのプリンシパルを追加しても、信頼関係は半分しか確立されない点に注意してください。プリンシパルとリソースが異なる AWS アカウントにある場合、信頼できるアカウントの IAM 管理者は、リソースへのアクセス許可をプリンシパルエンティティ (ユーザーまたはロール) に付与する必要があります。IAM 管理者は、アイデンティティベースのポリシーをエンティティにアタッチすることで権限を付与します。ただし、リソースベースのポリシーで、同じアカウントのプリンシパルへのアクセス権が付与されている場合は、アイデンティティベースのポリシーを追加する必要はありません。詳細については、『IAM ユーザーガイド』の「[IAM ロールとリソースベースのポリシーとの相違点](#)」を参照してください。

のポリシーアクション CloudFront

ポリシーアクションに対するサポート あり

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

JSON ポリシーの Action 要素には、ポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。ポリシーアクションの名前は通常、関連する AWS API オペレーションと同じです。一致する API オペレーションのない権限のみのアクションなど、いくつかの例外があります。また、ポリシーに複数アクションが必要なオペレーションもあります。これらの追加アクションは、依存アクションと呼ばれます。

このアクションは、関連付けられたオペレーションを実行するための権限を付与するポリシーで使用されます。

CloudFront アクションのリストを確認するには、「サービス認証リファレンス」の [「Amazon で定義されるアクション CloudFront」](#) を参照してください。

のポリシーアクションは、アクションの前に次のプレフィックス CloudFront を使用します。

```
cloudfront
```

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

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

CloudFront アイデンティティベースのポリシーの例を表示するには、「」を参照してください [Amazon のアイデンティティベースのポリシーの例 CloudFront](#)。

のポリシーリソース CloudFront

ポリシーリソースに対するサポート あり

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

Resource JSON ポリシーの要素は、オブジェクトあるいはアクションが適用されるオブジェクトを指定します。ステートメントには、Resource または NotResource 要素を含める必要があります。ベストプラクティスとしては、[Amazon リソースネーム \(ARN\)](#) を使用してリソースを指定します。これは、リソースレベルの権限と呼ばれる特定のリソースタイプをサポートするアクションに対して実行できます。

オペレーションのリスト化など、リソースレベルの権限をサポートしないアクションの場合は、ステートメントがすべてのリソースに適用されることを示すために、ワイルドカード (*) を使用します。

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

CloudFront リソースタイプとその ARNs」の「[Amazon で定義されるリソース CloudFront](#)」を参照してください。どのアクションで各リソースの ARN を指定できるかについては、「[Amazon で定義されるアクション CloudFront](#)」を参照してください。

CloudFront アイデンティティベースのポリシーの例を表示するには、「」を参照してください [Amazon のアイデンティティベースのポリシーの例 CloudFront](#)。

のポリシー条件キー CloudFront

サービス固有のポリシー条件キーのサポート	はい
----------------------	----

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

Condition 要素 (または Condition ブロック) を使用すると、ステートメントが有効な条件を指定できます。Condition 要素はオプションです。equal や less than などの[条件演算子](#)を使用して条件式を作成することによって、ポリシーの条件とリクエスト内の値を一致させることができます。

1 つのステートメントに複数の Condition 要素が指定されている場合、または 1 つの Condition 要素に複数のキーが指定されている場合、AWS では AND 論理演算子を使用してそれらを評価しま

す。単一の条件キーに複数の値が指定されている場合、AWS では OR 論理演算子を使用して条件を評価します。ステートメントの権限が付与される前にすべての条件が満たされる必要があります。

条件を指定する際にプレースホルダー変数も使用できます。例えば IAM ユーザーに、IAM ユーザー名がタグ付けされている場合のみリソースにアクセスできる許可を付与できます。詳細については、「IAM ユーザーガイド」の「[IAM ポリシーの要素: 変数およびタグ](#)」を参照してください。

AWS はグローバル条件キーとサービス固有の条件キーをサポートしています。すべての AWS グローバル条件キーを確認するには、『IAM ユーザーガイド』の「[AWS グローバル条件コンテキストキー](#)」を参照してください。

CloudFront 条件キーのリストを確認するには、「サービス認証リファレンス」の「[Amazon の条件キー CloudFront](#)」を参照してください。条件キーを使用できるアクションとリソースについては、「[Amazon で定義されるアクション CloudFront](#)」を参照してください。

CloudFront アイデンティティベースのポリシーの例を表示するには、「」を参照してください。[Amazon のアイデンティティベースのポリシーの例 CloudFront](#)。

ACLs CloudFront

ACL のサポート

なし

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

での ABAC CloudFront

ABAC (ポリシー内のタグ) のサポート

部分的

属性ベースのアクセスコントロール (ABAC) は、属性に基づいて権限を定義する認可戦略です。AWS では、これらの属性はタグと呼ばれます。タグは、IAM エンティティ (ユーザーまたはロール)、および多数の AWS リソースにアタッチできます。エンティティとリソースのタグ付けは、ABAC の最初の手順です。次に、プリンシパルのタグがアクセスを試行するリソースのタグと一致したときにオペレーションを許可するよう、ABAC ポリシーを設計します。

ABAC は、急成長する環境やポリシー管理が煩雑になる状況で役立ちます。

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

サービスがすべてのリソースタイプに対して 3 つの条件キーすべてをサポートする場合、そのサービスの値は Yes です。サービスが一部のリソースタイプに対してのみ 3 つの条件キーすべてをサポートする場合、値は Partial です。

ABAC の詳細については、『IAM ユーザーガイド』の「[ABAC とは?](#)」を参照してください。ABAC をセットアップするステップを説明するチュートリアルについては、「IAM ユーザーガイド」の「[属性に基づくアクセスコントロール \(ABAC\) を使用する](#)」を参照してください。

での一時的な認証情報の使用 CloudFront

一時的な認証情報のサポート	あり
---------------	----

AWS のサービスには、一時的な認証情報を使用してサインインしても機能しないものがあります。一時的な認証情報で機能する AWS のサービスなどの詳細については、「IAM ユーザーガイド」の「[IAM と連携する AWS のサービス](#)」を参照してください。

ユーザー名とパスワード以外の方法で AWS Management Console にサインインする場合は、一時的な認証情報を使用していることとなります。例えば、会社のシングルサインオン (SSO) リンクを使用して AWS にアクセスすると、そのプロセスは自動的に一時的な認証情報を作成します。また、ユーザーとしてコンソールにサインインしてからロールを切り替える場合も、一時的な認証情報が自動的に作成されます。ロールの切り替えに関する詳細については、「IAM ユーザーガイド」の「[ロールへの切り替え \(コンソール\)](#)」を参照してください。

一時的な認証情報は、AWS CLI または AWS API を使用して手動で作成できます。作成後、一時的な認証情報を使用して AWS にアクセスできるようになります。AWS は、長期的なアクセスキーを使用する代わりに、一時的な認証情報を動的に生成することをお勧めします。詳細については、「[IAM の一時的なセキュリティ認証情報](#)」を参照してください。

の転送アクセスセッション CloudFront

転送アクセスセッション (FAS) をサポート	いいえ
-------------------------	-----

IAM ユーザーまたはロールを使用して AWS でアクションを実行するユーザーは、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行してから、別のサービスの別のアクションを開始することがあります。FAS は、AWS のサービスを呼び出すプリンシパルの権限を、AWS のサービスのリクエストと合わせて使用し、ダウンストリームのサービスに対してリクエストを行います。FAS リクエストは、サービスが、完了するために他の AWS のサービスまたはリソースとのやりとりを必要とするリクエストを受け取ったときにのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「[転送アクセスセッション](#)」を参照してください。

CloudFront のサービスロール

サービスロールのサポート

いいえ

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

Warning

サービスロールのアクセス許可を変更すると、CloudFront 機能が破損する可能性があります。が指示する場合以外 CloudFront は、サービスロールを編集しないでください。

のサービスにリンクされたロール CloudFront

サービスリンクロールのサポート

はい

サービスリンクロールは、AWS のサービスにリンクされているサービスロールの一種です。サービスがロールを引き受け、ユーザーに代わってアクションを実行できるようになります。サービスにリンクされたロールは、AWS アカウントに表示され、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールの許可を表示できますが、編集はできません。

サービスにリンクされたロールの作成または管理の詳細については、「[IAM と提携する AWS のサービス](#)」を参照してください。表の中から、[サービスにリンクされたロール] (サービスにリンクされ

たロール) 列に Yes と記載されたサービスを見つけます。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[はい] リンクを選択します。

Amazon のアイデンティティベースのポリシーの例 CloudFront

デフォルトでは、ユーザーおよびロールには、CloudFront リソースを作成または変更する権限はありません。また、AWS Management Console、AWS Command Line InterfaceAWS CLI、または AWS API を使用してタスクを実行することもできません。IAM 管理者は、リソースに必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者がロールに IAM ポリシーを追加すると、ユーザーはロールを引き受けることができます。

これらサンプルの JSON ポリシードキュメントを使用して、IAM アイデンティティベースのポリシーを作成する方法については、『IAM ユーザーガイド』の「[IAM ポリシーの作成](#)」を参照してください。

各リソースタイプの ARN の形式など CloudFront、 で定義されるアクションとリソースタイプの詳細については、「サービス認証リファレンス」の「[Amazon のアクション、リソース、および条件キー CloudFront](#)」を参照してください。 ARNs

トピック

- [ポリシーのベストプラクティス](#)
- [CloudFront コンソールを使用する](#)
- [自分の権限の表示をユーザーに許可する](#)
- [CloudFront プログラムによる へのアクセス許可](#)
- [CloudFront コンソールを使用するために必要なアクセス許可](#)
- [AWS の マネージド \(事前定義\) ポリシー CloudFront](#)
- [カスタマーマネージドポリシーの例](#)

ポリシーのベストプラクティス

ID ベースのポリシーは、ユーザーのアカウントで誰かが CloudFront リソースを作成、アクセス、または削除できるかどうかを決定します。これらのアクションを実行すると、AWS アカウント に料金が発生する可能性があります。アイデンティティベースのポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください。

- AWS マネージドポリシーを使用して開始し、最小特権の権限に移行する – ユーザーとワークロードへの権限の付与を開始するには、多くの一般的なユースケースのために権限を付与する AWS マ

マネージドポリシーを使用します。これらは AWS アカウントで使用できます。ユースケースに応じた AWS カスタマー管理ポリシーを定義することで、許可をさらに減らすことをお勧めします。詳細については、「IAM ユーザーガイド」の「[AWS マネージドポリシー](#)」または「[AWS ジョブ機能の管理ポリシー](#)」を参照してください。

- 最小特権を適用する - IAM ポリシーで許可を設定するときは、タスクの実行に必要な許可のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権権限とも呼ばれています。IAM を使用して許可を適用する方法の詳細については、「IAM ユーザーガイド」の「[IAM でのポリシーとアクセス許可](#)」を参照してください。
- IAM ポリシーで条件を使用してアクセスをさらに制限する - ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。例えば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。また、AWS のサービス など特定の AWS CloudFormation を介して使用する場合、条件を使用してサービスアクションへのアクセスを許可することもできます。詳細については、「IAM ユーザーガイド」の「[IAM JSON policy elements: Condition](#)」(IAM JSON ポリシー要素 : 条件) を参照してください。
- IAM アクセスアナライザーを使用して IAM ポリシーを検証し、安全で機能的な許可を確保する - IAM アクセスアナライザーは、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、「IAM ユーザーガイド」の「[IAM Access Analyzer ポリシーの検証](#)」を参照してください。
- 多要素認証 (MFA) を要求する - AWS アカウント で IAM ユーザーまたはルートユーザーを要求するシナリオがある場合は、セキュリティを強化するために MFA をオンにします。API オペレーションが呼び出されるときに MFA を必須にするには、ポリシーに MFA 条件を追加します。詳細については、「IAM ユーザーガイド」の「[MFA 保護 API アクセスの設定](#)」を参照してください。

IAM でのベストプラクティスの詳細については、『IAM ユーザーガイド』の「[IAM でのセキュリティのベストプラクティス](#)」を参照してください。

CloudFront コンソールを使用する

Amazon CloudFront コンソールにアクセスするには、一連の最小限のアクセス許可が必要です。これらのアクセス許可により、の CloudFront リソースの詳細をリストおよび表示できますAWS アカウント。最小限必要なアクセス許可よりも制限が厳しいアイデンティティベースのポリシーを作成すると、そのポリシーを持つエンティティ (ユーザーまたはロール) ではコンソールが意図したとおりに機能しません。

AWS CLI または AWS API のみ呼び出すユーザーには、最小限のコンソール権限を付与する必要はありません。代わりに、実行しようとしている API オペレーションに一致するアクションのみへのアクセスを許可します。

ユーザーとロールが引き続き CloudFront コンソールを使用できるようにするには、エンティティに CloudFront *ConsoleAccess* または *ReadOnly* AWS 管理ポリシーもアタッチします。詳細については、『IAM ユーザーガイド』の「[ユーザーへの権限の追加](#)」を参照してください。

自分の権限の表示をユーザーに許可する

この例では、ユーザーアイデンティティにアタッチされたインラインおよびマネージドポリシーの表示を、IAM ユーザーに許可するポリシーの作成方法を示します。このポリシーには、コンソールで、または AWS CLI が AWS API を使用してプログラマ的に、このアクションを完了する権限が含まれています。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupForUser",
        "iam:ListAttachedUserPolicies",
        "iam:ListUserPolicies",
        "iam:GetUser"
      ],
      "Resource": ["arn:aws:iam::*:user/${aws:username}"]
    },
    {
      "Sid": "NavigateInConsole",
      "Effect": "Allow",
      "Action": [
        "iam:GetGroupPolicy",
        "iam:GetPolicyVersion",
        "iam:GetPolicy",
        "iam:ListAttachedGroupPolicies",
        "iam:ListGroupPolicies",
        "iam:ListPolicyVersions",
        "iam:ListPolicies",
        "iam:ListUsers"
      ]
    }
  ]
}
```

```
    ],
    "Resource": "*"
  }
]
}
```

CloudFront プログラムによる へのアクセス許可

以下に示しているのは、アクセス権限ポリシーです。Sid (ステートメント ID) はオプションです。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAllCloudFrontPermissions",
      "Effect": "Allow",
      "Action": ["cloudfront:*"],
      "Resource": "*"
    }
  ]
}
```

このポリシーは、すべての CloudFront オペレーションを実行するアクセス許可を付与します。これは、CloudFront プログラムでにアクセスするのに十分です。コンソールを使用してにアクセスする場合は CloudFront、「」を参照してください [CloudFront コンソールを使用するために必要なアクセス許可](#)。

各アクションを使用するアクセス許可を付与または拒否するために指定するアクションと ARN のリストについては、「サービス認証リファレンス」の [「Amazon のアクション、リソース、および条件キー CloudFront」](#) を参照してください。

CloudFront コンソールを使用するために必要なアクセス許可

CloudFront コンソールへのフルアクセスを許可するには、次のアクセス許可ポリシーでアクセス許可を付与します。

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

```
    "Action": [
      "acm:ListCertificates",
      "cloudfront:*",
      "cloudwatch:DescribeAlarms",
      "cloudwatch:PutMetricAlarm",
      "cloudwatch:GetMetricStatistics",
      "elasticloadbalancing:DescribeLoadBalancers",
      "iam:ListServerCertificates",
      "sns:ListSubscriptionsByTopic",
      "sns:ListTopics",
      "waf:GetWebACL",
      "waf:ListWebACLs"
    ],
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "s3:ListAllMyBuckets",
      "s3:PutBucketPolicy"
    ],
    "Resource": "arn:aws:s3:::*"
  }
]
```

アクセス許可が必要な理由は次のとおりです。

acm:ListCertificates

CloudFront コンソールを使用してディストリビューションを作成および更新していて、ビューワーと間、CloudFront または と CloudFront オリジンとの間で HTTPS を必須 CloudFront とするよう を設定する場合、では ACM 証明書のリストを表示できます。

CloudFront コンソールを使用していない場合、このアクセス許可は必要ありません。

cloudfront:*

すべてのアクションを実行できます CloudFront。

cloudwatch:DescribeAlarms および **cloudwatch:PutMetricAlarm**

CloudFront コンソールで CloudWatch アラームを作成および表示できます。

「sns:ListSubscriptionsByTopic」および「sns:ListTopics」も参照してください。

CloudFront コンソールを使用していない場合、これらのアクセス権限は必要ありません。

cloudwatch:GetMetricStatistics

CloudFront コンソールで CloudWatch メトリクスを CloudFront レンダリングできます。

CloudFront コンソールを使用していない場合、このアクセス許可は必要ありません。

elasticloadbalancing:DescribeLoadBalancers

ディストリビューションを作成および更新するときに、使用できるオリジンのリストで、Elastic Load Balancing ロードバランサーのリストを表示できるようにします。

CloudFront コンソールを使用していない場合、このアクセス許可は必要ありません。

iam:ListServerCertificates

CloudFront コンソールを使用してディストリビューションを作成および更新していて、ビューワーと 間、CloudFront または と CloudFront オリジン間で HTTPS を必須 CloudFront とするよう にを設定する場合、では IAM 証明書ストアで証明書のリストを表示できます。

CloudFront コンソールを使用していない場合、このアクセス許可は必要ありません。

s3:ListAllMyBuckets

ディストリビューションを作成および更新するときに、以下のオペレーションを実行できるようにします。

- 使用できるオリジンのリストで S3 バケットのリストを表示する
- アクセスログを保存できる S3 バケットのリストを表示する

CloudFront コンソールを使用していない場合、このアクセス許可は必要ありません。

S3:PutBucketPolicy

S3 バケットへのアクセスを制限するディストリビューションを作成または更新するときに、ユーザーがバケットポリシーを更新して、CloudFront オリジンアクセスアイデンティティへのアクセス権を付与できます。詳細については、「[the section called “オリジンアクセスアイデンティティの使用 \(レガシー、非推奨\)”](#)」を参照してください。

CloudFront コンソールを使用していない場合、このアクセス許可は必要ありません。

sns:ListSubscriptionsByTopic および **sns:ListTopics**

CloudFront コンソールで CloudWatch アラームを作成すると、では通知用の SNS トピックを選択できます。

CloudFront コンソールを使用していない場合、これらのアクセス権限は必要ありません。

waf:GetWebACL および **waf:ListWebACLs**

CloudFront コンソールでAWS WAFウェブ ACLs のリストを表示できます。

CloudFront コンソールを使用していない場合、これらのアクセス権限は必要ありません。

AWS の マネージド (事前定義) ポリシー CloudFront

AWS は、AWS によって作成され管理されるスタンドアロンの IAM ポリシーを提供することで、多くの一般的なユースケースに対応します。これらの AWS 管理ポリシーは、一般的なユースケースに必要なアクセス権限を付与することで、どの権限が必要なのかをユーザーが調査する必要をなくすことができます。詳細については、「IAM ユーザーガイド」の「[AWS マネージドポリシー](#)」を参照してください。の場合 CloudFront、IAM には 2 つの管理ポリシーがあります。

- CloudFrontFullAccess – CloudFront リソースへのフルアクセスを許可します。

Important

アクセスログ CloudFront を作成して保存する場合は、追加のアクセス許可を付与する必要があります。詳細については、「[標準ログ記録の設定およびログファイルへのアクセスに必要なアクセス許可](#)」を参照してください。

- CloudFrontReadOnlyAccess – CloudFront リソースへの読み取り専用アクセスを許可します。

カスタマーマネージドポリシーの例

独自のカスタム IAM ポリシーを作成して、CloudFront API アクションのアクセス許可を許可できます。これらのカスタムポリシーは、指定されたアクセス許可が必要な IAM ユーザーまたはグループにアタッチできます。これらのポリシーは、CloudFront API、AWS SDKs、またはを使用しているときに機能しますAWS CLI。次の例では、いくつかの一般的なユースケースのアクセス権限を示します。へのフルアクセスをユーザーに付与するポリシーについては CloudFront、「」を参照してください [CloudFront コンソールを使用するために必要なアクセス許可](#)。

例

- [例 1: すべてのディストリビューションへの読み取りアクセスを許可する](#)
- [例 2: ディストリビューションの作成、更新、削除を許可する](#)
- [例 3: 無効化の作成と一覧表示を許可する](#)

例 1: すべてのディストリビューションへの読み取りアクセスを許可する

次のアクセス許可ポリシーは、CloudFront コンソールですべてのディストリビューションを表示するアクセス許可をユーザーに付与します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "acm:ListCertificates",
        "cloudfront:GetDistribution",
        "cloudfront:GetDistributionConfig",
        "cloudfront:ListDistributions",
        "cloudfront:ListCloudFrontOriginAccessIdentities",
        "elasticloadbalancing:DescribeLoadBalancers",
        "iam:ListServerCertificates",
        "sns:ListSubscriptionsByTopic",
        "sns:ListTopics",
        "waf:GetWebACL",
        "waf:ListWebACLs"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListAllMyBuckets"
      ],
      "Resource": "arn:aws:s3:::*"
    }
  ]
}
```

例 2: ディストリビューションの作成、更新、削除を許可する

次のアクセス許可ポリシーでは、CloudFront コンソールを使用してディストリビューションを作成、更新、削除することをユーザーに許可します。

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

```
{
  "Effect": "Allow",
  "Action": [
    "acm:ListCertificates",
    "cloudfront:CreateDistribution",
    "cloudfront:DeleteDistribution",
    "cloudfront:GetDistribution",
    "cloudfront:GetDistributionConfig",
    "cloudfront:ListDistributions",
    "cloudfront:UpdateDistribution",
    "cloudfront:ListCloudFrontOriginAccessIdentities",
    "elasticloadbalancing:DescribeLoadBalancers",
    "iam:ListServerCertificates",
    "sns:ListSubscriptionsByTopic",
    "sns:ListTopics",
    "waf:GetWebACL",
    "waf:ListWebACLs"
  ],
  "Resource": "*"
},
{
  "Effect": "Allow",
  "Action": [
    "s3:ListAllMyBuckets",
    "s3:PutBucketPolicy"
  ],
  "Resource": "arn:aws:s3:::*"
}
]
```

`cloudfront:ListCloudFrontOriginAccessIdentities` アクセス許可では、ユーザーは、Amazon S3 バケットのオブジェクトを操作するアクセス許可が既存のオリジンアクセスアイデンティティに自動的に付与されるようにできます。また、ユーザーがオリジンアクセスアイデンティティを作成できるようにする場合は、`cloudfront:CreateCloudFrontOriginAccessIdentity` アクセス権限も許可する必要があります。

例 3: 無効化の作成と一覧表示を許可する

次のアクセス権限ポリシーでは、ユーザーが無効化を作成し、一覧表示できるようにします。デイス CloudFront トリビューションの設定を最初に表示して無効化を作成および表示するため、デイス トリビューションへの読み取りアクセスが含まれます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "acm:ListCertificates",
        "cloudfront:GetDistribution",
        "cloudfront:GetStreamingDistribution",
        "cloudfront:GetDistributionConfig",
        "cloudfront:ListDistributions",
        "cloudfront:ListCloudFrontOriginAccessIdentities",
        "cloudfront:CreateInvalidation",
        "cloudfront:GetInvalidation",
        "cloudfront:ListInvalidations",
        "elasticloadbalancing:DescribeLoadBalancers",
        "iam:ListServerCertificates",
        "sns:ListSubscriptionsByTopic",
        "sns:ListTopics",
        "waf:GetWebACL",
        "waf:ListWebACLs"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListAllMyBuckets"
      ],
      "Resource": "arn:aws:s3:::*"
    }
  ]
}
```

AWS Amazon の マネージドポリシー CloudFront

ユーザー、グループ、ロールにアクセス許可を追加するには、自分でポリシーを作成するよりも、AWS 管理ポリシーを使用する方が簡単です。ユーザーに必要なアクセス許可のみを提供する [IAM カスタマー管理ポリシーを作成する](#) には、時間と専門知識が必要です。すぐに使用を開始するために、AWS マネージドポリシーを使用できます。これらのポリシーは、一般的なユースケースをターゲット範囲に含めており、AWS アカウント で利用できます。AWS マネージドポリシーの詳細については、「IAM ユーザーガイド」の [「AWS マネージドポリシー」](#) を参照してください。

AWS のサービスは、AWS マネージドポリシーを維持および更新します。AWS 管理ポリシーのアクセス許可を変更することはできません。サービスでは、新しい機能を利用できるようにするために、AWS マネージドポリシーに権限が追加されることがあります。この種類の更新は、ポリシーがアタッチされている、すべてのアイデンティティ (ユーザー、グループおよびロール) に影響を与えます。新しい機能が立ち上げられたり、新しいアクセス許可が使用可能になったりすると、各サービスで AWS 管理ポリシーが更新される可能性が最も高くなります。各サービスでは、AWS 管理ポリシーからアクセス許可が削除されないため、ポリシーの更新によって既存のアクセス許可が破棄されることはありません。

さらに、AWS では、複数のサービスにまたがるジョブ機能のためのマネージドポリシーもサポートしています。例えば、ReadOnlyAccess AWS マネージドポリシーでは、すべての AWS のサービスおよびリソースへの読み取り専用アクセスを許可します。あるサービスで新しい機能を立ち上げる場合は、AWS は、追加された演算とリソースに対し、読み込み専用の権限を追加します。ジョブ機能のポリシー一覧と説明については、IAM ユーザーガイドの [AWS ジョブ機能の管理ポリシー](#) を参照してください。

AWS マネージドポリシー: CloudFrontReadOnlyAccess

CloudFrontReadOnlyAccess ポリシーは IAM ID にアタッチできます。このポリシーは、CloudFront リソースへの読み取り専用アクセス許可を付与します。また、に関連する他のAWSサービスリソース CloudFront や、CloudFront コンソールに表示されるサービスリソースへの読み取り専用アクセス許可も付与します。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- `cloudfront:Describe*` — プリンシパルが CloudFront リソースに関するメタデータに関する情報を取得できるようにします。

- `cloudfront:Get*` — プリンシパルが CloudFront リソースの詳細情報と設定を取得できるようにします。
- `cloudfront:List*` — プリンシパルがリソースのリスト CloudFrontを取得できるようにします。
- `cloudfront-keyvaluestore:Describe*` - プリンシパルがキー値ストアに関する情報を取得できるようにします。
- `cloudfront-keyvaluestore:Get*` - プリンシパルがキー値ストアの詳細情報と設定を取得できるようにします。
- `cloudfront-keyvaluestore:List*` - プリンシパルがキー値ストアのリストを取得できるようにします。
- `acm:ListCertificates` - プリンシパルが ACM 証明書のリストを取得できるようにします。
- `iam:ListServerCertificates` - プリンシパルが IAM に格納されるサーバー証明書のリストを許可できるようにします。
- `route53:List*` - プリンシパルが Route 53 リソースのリストを取得できるようにします。
- `waf:ListWebACLs` - プリンシパルが AWS WAF のウェブ ACL のリストを取得できるようにします。
- `waf:GetWebACL` - プリンシパルが AWS WAF のウェブ ACL に関する詳細情報を取得できるようにします。
- `wafv2:ListWebACLs` - プリンシパルが AWS WAF のウェブ ACL のリストを取得できるようにします。
- `wafv2:GetWebACL` - プリンシパルが AWS WAF のウェブ ACL に関する詳細情報を取得できるようにします。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "cfReadOnly",
      "Effect": "Allow",
      "Action": [
        "acm:ListCertificates",
        "cloudfront:Describe*",
        "cloudfront:Get*",
        "cloudfront:List*",
        "cloudfront-keyvaluestore:Describe*",
        "cloudfront-keyvaluestore:Get*",
```

```
        "cloudfront-keyvaluestore:List*",
        "iam:ListServerCertificates",
        "route53:List*",
        "waf:ListWebACLs",
        "waf:GetWebACL",
        "wafv2:ListWebACLs",
        "wafv2:GetWebACL"
    ],
    "Resource": "*"
}
]
```

AWS マネージドポリシー: CloudFrontFullAccess

CloudFrontFullAccess ポリシーは IAM ID にアタッチできます。このポリシーは、CloudFront リソースへの管理アクセス許可を付与します。また、に関連する他のAWSサービスリソース CloudFront や、CloudFront コンソールに表示されるサービスリソースへの読み取り専用アクセス許可も付与します。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- `s3:ListAllMyBuckets` - プリンシパルが、すべての Amazon S3 バケットのリストを取得できるようにします。
- `acm:ListCertificates` - プリンシパルが ACM 証明書のリストを取得できるようにします。
- `cloudfront:*` — プリンシパルがすべてのリソースに対してすべての CloudFront アクションを実行できるようにします。
- `cloudfront-keyvaluestore:*` - プリンシパルがキー値ストアですべてのアクションを実行できるようにします。
- `iam:ListServerCertificates` - プリンシパルが IAM に格納されるサーバー証明書のリストを許可できるようにします。
- `waf:ListWebACLs` - プリンシパルが AWS WAF のウェブ ACL のリストを取得できるようにします。
- `waf:GetWebACL` - プリンシパルが AWS WAF のウェブ ACL に関する詳細情報を取得できるようにします。
- `wafv2:ListWebACLs` - プリンシパルが AWS WAF のウェブ ACL のリストを取得できるようにします。

- `wafv2:GetWebACL` - プリンシパルが AWS WAF のウェブ ACL に関する詳細情報を取得できるようにします。
- `kinesis:ListStreams` - プリンシパルが Amazon Kinesis ストリームのリストを取得できるようにします。
- `kinesis:DescribeStream` - プリンシパルが Kinesis ストリームに関する詳細情報を取得できるようにします。
- `iam:ListRoles` - プリンシパルが IAM のロールのリストを取得できるようにします。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "cfflistbuckets",
      "Action": [
        "s3:ListAllMyBuckets"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::*"
    },
    {
      "Sid": "cfffullaccess",
      "Action": [
        "acm:ListCertificates",
        "cloudfront:*",
        "cloudfront-keyvaluestore:*",
        "iam:ListServerCertificates",
        "waf:ListWebACLs",
        "waf:GetWebACL",
        "wafv2:ListWebACLs",
        "wafv2:GetWebACL",
        "kinesis:ListStreams"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Sid": "cffdescribestream",
      "Action": [
        "kinesis:DescribeStream"
      ],
      "Effect": "Allow",
```

```
    "Resource": "arn:aws:kinesis:*:*:*"
  },
  {
    "Sid": "cfflistroles",
    "Action": [
      "iam:ListRoles"
    ],
    "Effect": "Allow",
    "Resource": "arn:aws:iam:*:*:*"
  }
]
```

AWS マネージドポリシー: AWSCloudFrontLogger

IAM ID にAWSCloudFrontLoggerポリシーをアタッチすることはできません。このポリシーは、ユーザーに代わって CloudFront アクションを実行することを許可する、サービスにリンクされたロールにアタッチされます。詳細については、「[the section called “Lambda@Edge 用のサービスにリンクされたロール”](#)」を参照してください。

このポリシーにより、ログファイル CloudFront を Amazon にプッシュできます CloudWatch。このポリシーに含まれるアクセス許可の詳細については、「[the section called “ロガーのサービス CloudFrontにリンクされたロールのアクセス許可”](#)」を参照してください。

AWS マネージドポリシー: AWSLambdaReplicator

IAM ID にAWSLambdaReplicatorポリシーをアタッチすることはできません。このポリシーは、ユーザーに代わって CloudFront アクションを実行することを許可する、サービスにリンクされたロールにアタッチされます。詳細については、「[the section called “Lambda@Edge 用のサービスにリンクされたロール”](#)」を参照してください。

このポリシーにより、関数 CloudFront を作成、削除、および無効にAWS Lambdaして、Lambda@Edge 関数をレプリケートできますAWS リージョン。このポリシーに含まれるアクセス許可の詳細については、「[the section called “Lambda Replicator 用のサービスにリンクされたロールのアクセス許可”](#)」を参照してください。

AWS マネージドポリシーに関する CloudFront の更新

このサービスがこれらの変更の追跡を開始した CloudFront 以降の、の AWSマネージドポリシーの更新に関する詳細を表示します。このページの変更に関する自動通知を入手するには、CloudFront [ドキュメント履歴](#)ページの RSS フィードを購読してください。

変更	説明	日付
CloudFrontReadOnlyAccess と CloudFrontFullAccess - 2 つの既存のポリシーに対する更新	<p>CloudFront は、キー値ストアの新しいアクセス許可を追加しました。</p> <p>新しいアクセス許可により、ユーザーはキー値ストアに関する情報を取得し、アクションを実行できます。</p>	2023 年 12 月 19 日
CloudFrontReadOnlyAccess - 既存ポリシーへの更新	<p>CloudFront は、CloudFront 関数を記述する新しいアクセス許可を追加しました。</p> <p>このアクセス許可により、ユーザー、グループ、またはロールは関数に関する情報とメタデータを読み取ることができますが、関数のコードは読み取れません。</p>	2021 年 9 月 8 日
CloudFront が変更の追跡を開始	CloudFront が AWS マネージドポリシーの変更の追跡を開始しました。	2021 年 9 月 8 日

Amazon CloudFront アイデンティティとアクセスのトラブルシューティング

次の情報は、と IAM の使用に伴って発生する可能性がある一般的な問題の診断 CloudFront や修復に役立ちます。

トピック

- [でアクションを実行する権限がない CloudFront](#)
- [iam を実行する権限がありません。PassRole](#)
- [自分の 以外のユーザーに CloudFront リソースAWS アカウントへのアクセスを許可したい](#)

でアクションを実行する権限がない CloudFront

「I am not authorized to perform an action in Amazon Bedrock」というエラーが表示された場合、そのアクションを実行できるようにポリシーを更新する必要があります。

次の例は、mateojackson という IAM ユーザーがコンソールを使用して架空の *my-example-widget* リソースに関する詳細を表示しようとしたとき、架空の `cloudfront:GetWidget` アクセス許可がない場合に発生するエラーを示しています。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
cloudfront:GetWidget on resource: my-example-widget
```

この場合、`cloudfront:GetWidget` アクションを使用して *my-example-widget* リソースへのアクセスを許可するように、mateojackson ユーザーのポリシーを更新する必要があります。

サポートが必要な場合は、AWS 管理者に問い合わせてください。管理者とは、サインイン認証情報を提供した担当者です。

iam を実行する権限がありません。PassRole

`iam:PassRole` アクションを実行する権限がないというエラーが表示された場合は、ポリシーを更新して CloudFront にロールを渡すことができるようにする必要があります。

一部の AWS のサービスでは、新しいサービスロールやサービスリンクロールを作成せずに、既存のロールをサービスに渡すことができます。そのためには、サービスにロールを渡す権限が必要です。

以下の例のエラーは、marymajor という IAM ユーザーがコンソールを使用して CloudFront でアクションを実行しようする場合に発生します。ただし、このアクションをサービスが実行するには、サービスロールから付与された権限が必要です。Mary には、ロールをサービスに渡す許可がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

この場合、Mary のポリシーを更新して Mary に `iam:PassRole` アクションの実行を許可する必要があります。

サポートが必要な場合は、AWS 管理者に問い合わせてください。管理者とは、サインイン認証情報を提供した担当者です。

自分の 以外のユーザーに CloudFront リソースAWS アカウントへのアクセスを許可したい

他のアカウントのユーザーや組織外の人が、リソースにアクセスするために使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまたはアクセス制御リスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください:

- がこれらの機能 CloudFront をサポートしているかどうかを確認するには、「」を参照してください [Amazon と IAM の CloudFront 連携方法](#)。
- 所有している AWS アカウント 全体のリソースへのアクセス権を提供する方法については、『IAM ユーザーガイド』の「[所有している別の AWS アカウント アカウントへのアクセス権を IAM ユーザーに提供](#)」を参照してください。
- サードパーティーの AWS アカウント にリソースへのアクセス権を提供する方法については、「IAM ユーザーガイド」の「[サードパーティーが所有する AWS アカウント にアクセス権を提供する](#)」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、「IAM ユーザーガイド」の「[外部で認証されたユーザー \(ID フェデレーション\) へのアクセスの許可](#)」を参照してください。
- クロスアカウントアクセスでのロールとリソースベースのポリシーの使用の違いの詳細については、『IAM ユーザーガイド』の「[IAM ロールとリソースベースのポリシーとの相違点](#)」を参照してください。

Amazon でのログ記録とモニタリング CloudFront

モニタリングは、CloudFront と AWS ソリューションの可用性とパフォーマンスを維持する上で重要な部分です。マルチポイント障害が発生した場合は、その障害をより簡単にデバッグできるように、AWSソリューションのすべての部分からモニタリングデータを収集する必要があります。AWSには、CloudFront リソースとアクティビティをモニタリングし、潜在的なインシデントに対応するための複数のツールが用意されています。

Amazon CloudWatch アラーム

CloudWatch アラームを使用すると、指定した期間にわたって 1 つのメトリクスを監視できます。メトリクスが特定の閾値を超えると、Amazon SNS トピックまたは AWS Auto Scaling ポリシーに通知が送信されます。CloudWatch メトリクスが特定の状態にある場合、アラームはアク

ションを呼び出しません。状態が変わり、それが指定した期間だけ維持される必要があります。詳細については、「[Amazon による CloudFront メトリクスのモニタリング CloudWatch](#)」を参照してください。

AWS CloudTrail ログ

CloudTrail は、 のユーザー、ロール、または AWSのサービスによって実行された API アクションの記録を提供します CloudFront。で収集された情報を使用して CloudTrail、 に対して行われた API リクエスト CloudFront、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。詳細については、「[AWS CloudTrail を使用して API に送信されたリクエストを CloudFront キャプチャする](#)」を参照してください。

CloudFront 標準ログとリアルタイムログ

CloudFront ログは、ディストリビューションに対して行われたリクエストに関する詳細なレコードを提供します。これらのログは、多くのアプリケーションで役立ちます。たとえば、ログ情報は、セキュリティやアクセスの監査に役立ちます。詳しくは、「[CloudFront およびエッジ関数のログ記録](#)」を参照してください。

エッジ関数のログ

CloudFront Functions と Lambda@Edge の両方のエッジ関数によって生成されたログは Amazon CloudWatch Logs に直接送信され、 によってどこにも保存されません CloudFront。CloudFront Functions は、AWS Identity and Access Management (IAM) [サービスにリンクされたロール](#)を使用して、カスタマー生成ログをアカウントの CloudWatch ログに直接送信します。

CloudFront コンソールレポート

CloudFront コンソールには、キャッシュ統計レポート、人気オブジェクトレポート、トップリファラレポートなど、さまざまなレポートが含まれています。ほとんどの CloudFront コンソールレポートは、 が CloudFront 受信するすべてのユーザーリクエストに関する詳細情報を含む CloudFront アクセスログのデータに基づいています。ただし、このレポートを表示するために、アクセスログを有効にする必要はありません。詳細については、「[CloudFront コンソールのレポート](#)」を参照してください。

Amazon のコンプライアンス検証 CloudFront

サードパーティーの監査者は、さまざまな コンプライアンスプログラム CloudFront の一環として Amazon のセキュリティとAWSコンプライアンスを評価します。このプログラムには、SOC、PCI、HIPAA などを含みます。

特定のコンプライアンスプログラムの対象範囲に含まれる AWS のサービスのリストについては、「[コンプライアンスプログラムによる対象範囲内の AWS のサービス](#)」を参照してください。一般的な情報については、「[AWS コンプライアンスプログラム](#)」を参照してください。

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

を使用する際のお客様のコンプライアンス責任 CloudFront は、お客様のデータの機密性、企業のコンプライアンス目的、適用法規によって決まります。AWSでは、コンプライアンスに役立つ以下のリソースを提供しています。

- [セキュリティおよびコンプライアンスのクイックスタートガイド](#) - これらのデプロイメントガイドでは、アーキテクチャー上の考慮事項について説明し、セキュリティとコンプライアンスに重点を置いたベースライン環境を AWS にデプロイするための手順を説明します。
- [HIPAA のセキュリティとコンプライアンスのためのアーキテクチャの設計 AWS](#) - このホワイトペーパーでは、企業が AWS を使用して HIPAA 準拠のアプリケーションを作成する方法について説明します。

HIPAA コンプライアンスプログラムには、HIPAA AWS 対応サービス CloudFront としてが含まれています。で事業提携契約 (BAA) を実行している場合は AWS、を使用して CloudFront 保護医療情報 (PHI) を含むコンテンツを配信できます。詳細については、「[HIPAA コンプライアンス](#)」を参照してください。

- [AWS コンプライアンスのリソース](#) - このワークブックとガイドのコレクションは、お客様の業界や所在地に適用される場合があります。
- [AWS Config](#) - この AWS のサービスでは、自社プラクティス、業界ガイドライン、および規制に対するリソースの設定の準拠状態を評価します。
- [AWS Security Hub](#) - この AWS のサービスは、セキュリティコントロールを使用してリソース設定とセキュリティ標準を評価し、お客様がさまざまなコンプライアンスフレームワークに準拠できるようにサポートします。Security Hub を使用して CloudFront リソースを評価する方法の詳細については、「ユーザーガイド」の「[Amazon CloudFront コントロール](#)」を参照してください。AWS Security Hub

CloudFront コンプライアンスのベストプラクティス

このセクションでは、Amazon を使用してコンテンツ CloudFront を提供する際のコンプライアンスに関するベストプラクティスと推奨事項について説明します。

[AWS 責任共有モデル](#) に基づく PCI 準拠または HIPAA 準拠のワークロードを実行する場合は、将来の監査のために過去 365 日間の CloudFront 使用状況データをログに記録することをお勧めします。使用状況データを記録するには、以下の操作を実行できます。

- CloudFront アクセスログを有効にします。詳細については、「[標準ログ \(アクセスログ\) の設定および使用](#)」を参照してください。
- CloudFront API に送信されるリクエストを取得します。詳細については、「[AWS CloudTrail を使用して API に送信されたリクエストを CloudFront キャプチャする](#)」を参照してください。

さらに、CloudFront が PCI DSS および SOC 標準にどのように準拠しているかの詳細については、以下を参照してください。

Payment Card Industry Data Security Standard (PCI DSS)

CloudFront は、マーチャントまたはサービスプロバイダーによるクレジットカードデータの処理、ストレージ、および伝送をサポートしており、Payment Card Industry (PCI) Data Security Standard (DSS) に準拠していることが確認されています。PCI DSS の詳細 (AWS PCI Compliance Package のコピーをリクエストする方法など) については、「[PCI DSS レベル 1](#)」を参照してください。

セキュリティのベストプラクティスとして、クレジットカード情報を CloudFront エッジキャッシュにキャッシュしないことをお勧めします。たとえば、クレジットカード番号の末尾 4 桁の数字やカード所有者の連絡先情報などのクレジットカード情報が含まれている `Cache-Control: no-cache="field-name"` ヘッダーをレスポンスに含めるようにオリジンを設定できます。

System and Organization Controls (SOC)

CloudFront は、SOC 1、SOC 2、SOC 3 などのシステムおよび組織規制 (SOC) 基準に準拠しています。SOC レポートは、重要なコンプライアンス管理および目標を AWS がどのように達成したかを実証する、独立したサードパーティーによる審査報告書です。これらの監査によって、お客様のデータや企業データのセキュリティ、機密保持、アベイラビリティに影響を及ぼす可能性のあるリスクから守るために、適切な安全策と手順を講じます。これらサードパーティー監査の結果は、[AWS SOC コンプライアンスのウェブサイト](#)で参照できます。このサイトでは、AWS の業務とコンプライアンスをサポートするために制定された統制についてさらに詳しく知ることのできる発行済みレポートを公開しています。

Amazon の耐障害性 CloudFront

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

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

CloudFront オリジンフェイルオーバー

Amazon CloudFront では、AWS グローバルインフラストラクチャのサポートに加えて、データの耐障害性のニーズをサポートするオリジンフェイルオーバー機能を提供しています。CloudFront は、エッジロケーションまたは Point of Presence (POPs)。コンテンツがまだエッジロケーションにキャッシュされていない場合、CloudFront はそのコンテンツの最終版のソースとして識別したオリジンからそれを取得します。

オリジンフェイルオーバーを使用して CloudFront を設定することで、回復力を向上させ、特定のシナリオの可用性を向上させることができます。開始するには、のプライマリオリジン CloudFront と 2 番目のオリジンを指定するオリジングループを作成します。プライマリオリジンが特定の HTTP ステータスコード失敗レスポンスを返すと、は 2 番目のオリジン CloudFront に自動的に切り替わります。詳細については、「[CloudFront オリジンフェイルオーバーによる高可用性の最適化](#)」を参照してください。

Amazon のインフラストラクチャセキュリティ CloudFront

マネージドサービスである Amazon CloudFront は AWS グローバルネットワークセキュリティで保護されています。AWS セキュリティサービスと AWS によるインフラストラクチャの保護方法については、「[AWS クラウドセキュリティ](#)」を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して AWS 環境を設計するには、「セキュリティの柱 - AWS Well-Architected フレームワーク」の「[インフラストラクチャ保護](#)」を参照してください。

が AWS 公開した API コールを使用して、ネットワーク CloudFront 経由で にアクセスします。クライアントは以下をサポートする必要があります:

- Transport Layer Security (TLS)。TLS 1.2、できれば TLS 1.3 が必要です。
- DHE (Ephemeral Diffie-Hellman) や ECDHE (Elliptic Curve Ephemeral Diffie-Hellman) などの Perfect Forward Secrecy (PFS) を使用した暗号スイート。これらのモードは Java 7 以降など、最近のほとんどのシステムでサポートされています。

また、リクエストには、アクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、[AWS Security Token Service \(AWS STS\)](#) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

CloudFront Functions は、AWSアカウント間で非常に安全な隔離手段を使用し、Spectrume や Meltdown などのサイドチャネル攻撃からお客様の環境を保護します。Functions は、他の顧客に属するデータにアクセスしたり、データを変更したりすることはできません。関数は、ハイパースレッディングなし、専用CPUの専用シングルスレッドプロセスで実行されます。どの CloudFront エッジロケーションの Point of Presence (POP) でも、CloudFront Functions は一度に 1 人の顧客にのみサービスを提供し、すべての顧客固有のデータは関数の実行間でクリアされます。

クォータ

CloudFront には以下のクォータが適用されます。

トピック

- [一般的なクォータ](#)
- [ディストリビューションの一般的なクォータ](#)
- [ポリシーの一般的なクォータ](#)
- [CloudFront 関数のクォータ](#)
- [キー値ストアのクォータ](#)
- [Lambda@Edge のクォータ](#)
- [SSL 証明書のクォータ](#)
- [無効化のクォータ](#)
- [キーグループのクォータ](#)
- [WebSocket 接続のクォータ](#)
- [フィールドレベル暗号化のクォータ](#)
- [Cookie のクォータ \(従来のキャッシュ設定\)](#)
- [クエリ文字列のクォータ \(従来のキャッシュ設定\)](#)
- [ヘッダーのクォータ](#)

一般的なクォータ

エンティティ	デフォルトのクォータ
ディストリビューションごとのデータ転送レート	150 Gbps クォータ引き上げのリクエスト

エンティティ	デフォルトのクォータ
1 秒あたり、ディストリビューションあたりのリクエスト	250,000 クォータ引き上げのリクエスト
ディストリビューションに追加できるタグ	50
ディストリビューションごとに配信可能なファイル数	クォータなし
ヘッダーとクエリ文字列を含むが、本文のコンテンツは含まないリクエストの最大長	20,480 バイト
URL の最大長	8,192 バイト

ディストリビューションの一般的なクォータ

エンティティ	デフォルトのクォータ
ディストリビューションあたりの代替ドメイン名 (CNAME)	100
詳細については、「 代替ドメイン名 (CNAME) を追加することによるカスタム URL の使用 」を参照してください。	クォータ引き上げのリクエスト
ディストリビューションあたりのキャッシュ動作	25 クォータ引き上げのリクエスト
オリジン別の接続試行回数	1 ~ 3
詳細については、「 接続の試行 」を参照してください。	
オリジン別の接続タイムアウト	1 ~ 10 秒
詳細については、「 接続タイムアウト 」を参照してください。	

エンティティ	デフォルトのクォータ
AWS アカウントあたりのディストリビューション 詳細については、「 ディストリビューションの作成 」を参照してください。	200 クォータ引き上げのリクエスト
オリジンアクセスコントロールあたりのディストリビューション	100 クォータ引き上げのリクエスト
ファイル圧縮: が CloudFront 圧縮するファイルサイズの範囲 詳細については、「 圧縮ファイルの供給 」を参照してください。	1,000 ~ 10,000,000 バイト
オリジンあたりのキープアライブタイムアウト 詳細については、「 キープアライブタイムアウト (カスタムオリジンのみ) 」を参照してください。	1 ~ 60 秒 クォータ引き上げのリクエスト
HTTP GET レスポンスあたりのキャッシュ可能な最大ファイルサイズ。 HTTP GET のレスポンスのみがキャッシュされます。POST または PUT のレスポンスはキャッシュされません。	50 GB
AWS アカウントあたりのオリジンアクセスコントロール	100
AWS アカウントあたりのオリジンアクセスアイデンティティ	100 クォータ引き上げのリクエスト
ディストリビューションあたりのオリジン	25 クォータ引き上げのリクエスト

エンティティ	デフォルトのクォータ
ディストリビューションあたりのオリジングループ	10 クォータ引き上げのリクエスト
オリジン別の応答タイムアウト 詳細については、「 応答タイムアウト (カスタムオリジンのみ) 」を参照してください。	1 ~ 60 秒 クォータ引き上げのリクエスト
AWS アカウントあたりのステージングディストリビューション 詳細については、「 the section called “継続的デプロイを使用して変更を安全にテストする” 」を参照してください。	20 クォータ引き上げのリクエスト

ポリシーの一般的なクォータ

エンティティ	デフォルトのクォータ
AWS アカウントあたりのキャッシュポリシー	20
同じキャッシュポリシーに関連付けられたディストリビューション	100
キャッシュポリシーあたりのクエリ文字列	10 クォータ引き上げのリクエスト
キャッシュポリシーあたりのヘッダー	10 クォータ引き上げのリクエスト
キャッシュポリシーあたりの Cookie	10

エンティティ	デフォルトのクォータ クォータ引き上げのリクエスト
キャッシュポリシー内のすべてのクエリ文字列、ヘッダー、および Cookie 名の合計長	1024
AWS アカウントあたりのオリジンリクエストポリシー	20
同じオリジンリクエストポリシーに関連付けられたディストリビューション	100
オリジンリクエストポリシーあたりのクエリ文字列	10 クォータ引き上げのリクエスト
オリジンリクエストポリシーあたりのヘッダー	10 クォータ引き上げのリクエスト
オリジンリクエストポリシーあたりの Cookie	10 クォータ引き上げのリクエスト
キャッシュリクエストポリシー内のすべてのクエリ文字列、ヘッダー、および Cookie 名の合計長	1024
AWS アカウントあたりのレスポンスヘッダーポリシー	20 クォータ引き上げのリクエスト
同じレスポンスヘッダーポリシーに関連付けられたディストリビューション	100 クォータ引き上げのリクエスト

エンティティ	デフォルトのクォータ
レスポンスヘッダーポリシーごとのカスタムヘッダー	10 クォータ引き上げのリクエスト
AWS アカウントあたりの継続的デプロイポリシー	20 クォータ引き上げのリクエスト

CloudFront 関数のクォータ

エンティティ	デフォルトのクォータ
AWS アカウント あたりの関数	100
最大関数サイズ	10 KB クォータ引き上げのリクエスト
最大関数メモリ	2 MB
同じ関数に関連付けられたディストリビューション	100

これらのクォータに加えて、CloudFront 関数を使用する際には、他にもいくつかの制限があります。詳細については、「[CloudFront 関数の制限](#)」を参照してください。

キー値ストアのクォータ

エンティティ	デフォルトのクォータ
キーと値のペアのキーの最大サイズ	512 バイト
キーと値のペアの値の最大サイズ	1 KB
1 回の API リクエストで更新できるキー値ペアの最大数	30 個のキーまたは 3 MB のペイロード (どちらか達するのが先の方)
個々のキーと値のストアの最大サイズ	5 MB
1 つのキー値ストアに関連付けることができる関数の最大数	10
関数ごとのキー値ストアの最大数	1
アカウントごとのキー値ストアの最大数	50

[クォータ引き上げのリクエスト](#)

Lambda@Edge のクォータ

このセクションのクォータは、Lambda@Edge に適用されます。これらのクォータは、同じく適用されるデフォルトの AWS Lambda クォータへの追加となります。Lambda クォータについては、AWS Lambda デベロッパーガイドの「[クォータ](#)」を参照してください。

Note

Lambda は、AWS アカウントのクォータ内で、トラフィックの増加に応じて容量を動的にスケールします。詳細については、AWS Lambda デベロッパーガイドの「[関数スケーリング](#)」を参照してください。

一般的なクォータ

エンティティ	デフォルトのクォータ
Lambda@Edge 関数を持つことができる AWS アカウントあたりのディストリビューション	500 クォータ引き上げのリクエスト
ディストリビューションごとの Lambda@Edge 関数	100 クォータ引き上げのリクエスト
1 秒あたりのリクエスト	10,000 (各 AWS リージョン 内) クォータ引き上げのリクエスト
同時実行数 詳細については、AWS Lambda デベロッパーガイドの「 関数スケーリング 」を参照してください。	1,000 (各 AWS リージョン 内) クォータ引き上げのリクエスト
同じ関数に関連付けられたディストリビューション	500

イベントタイプによって異なるクォータ

エンティティ	ビューワーリクエストイベントとビューワーレスポンスイベント	オリジンリクエストイベントとオリジンレスポンスイベント
関数のメモリサイズ	128 MB	Lambda のクォータ と同じ

エンティティ	ビューワーリクエストイベントとビューワーレスポンスイベント	オリジンリクエストイベントとオリジンレスポンスイベント
関数タイムアウト。関数は、AWS リージョン内の Amazon S3 バケット、DynamoDB テーブル、または Amazon EC2 インスタンスなどのリソースに対してネットワークコールを実行できます。	5 秒	30 秒
ヘッダーと本文を含む、Lambda 関数によって生成されたレスポンスのサイズ	40 KB	1 MB
Lambda 関数および組み込みライブラリの最大圧縮サイズ	1 MB	50 MB

これらのクォータに加えて、Lambda@Edge 関数を使用する場合は、他にもいくつかの制限があります。詳細については、「[Lambda@Edge に対する制限](#)」を参照してください。

SSL 証明書のクォータ

エンティティ	デフォルトのクォータ
専用 IP アドレスを使用して HTTPS リクエストに対応する際の AWS アカウントあたりの SSL 証明書数 (SNI を使用して HTTPS リクエストに対応する際はクォータなし) 詳細については、「 で HTTPS を使用する CloudFront 」を参照してください。	2 クォータ引き上げのリクエスト
ディストリビューションに関連付けることができる SSL CloudFront 証明書	1

無効化のクォータ

エンティティ	デフォルトのクォータ
ファイルの無効化: ワイルドカードの無効化を除く、アクティブな無効化リクエストで許可されるファイルの最大数 詳細については、「 ファイルの無効化 」を参照してください。	3,000
ファイルの無効化: 許可されるアクティブなワイルドカード無効化の最大数	15
ファイルの無効化: 1つのワイルドカードの無効化で処理できるファイルの最大数	クォータなし

キーグループのクォータ

エンティティ	デフォルトのクォータ
1つのキーグループのパブリックキー	5 クォータ引き上げのリクエスト
1つのキャッシュ動作に関連付けられたキーグループ	4 クォータ引き上げのリクエスト
AWS アカウントあたりのキーグループ数	10 クォータ引き上げのリクエスト
1つのキーグループに関連付けられたディストリビューション	100 クォータ引き上げのリクエスト

WebSocket 接続のクォータ

エンティティ	デフォルトのクォータ
オリジン応答タイムアウト (アイドルタイムアウト)	10 分
	CloudFront が過去 10 分以内にオリジンからクライアントに送信されたバイトを検出しない場合、接続はアイドル状態であると見なされ、閉じられます。

フィールドレベル暗号化のクォータ

エンティティ	デフォルトのクォータ
暗号化するフィールドの最大長	16 KB
詳細については、「 フィールドレベル暗号化を使用した機密データの保護 」を参照してください。	
フィールドレベル暗号化が設定されている場合のリクエストボディの最大フィールド数	10
フィールドレベル暗号化が設定されている場合のリクエストボディの最大長	1 MB
1 つの AWS アカウントに関連付けることができるフィールドレベル暗号化設定の最大数	10
1 つの AWS アカウントに関連付けることができるフィールドレベル暗号化プロファイルの最大数	10

エンティティ	デフォルトのクォータ
1つのAWSアカウントに追加できるパブリックキーの最大数	10
1つのプロファイルに指定できる暗号化するフィールドの最大数	10
フィールドレベル暗号化設定に関連付けることができる CloudFront デистриビューションの最大数	20
フィールドレベル暗号化設定に含めることができるクエリ引数プロファイルマッピングの最大数	5

Cookie のクォータ (従来のキャッシュ設定)

これらのクォータは、CloudFrontのレガシーキャッシュ設定に適用されます。従来の設定の代わりに、[キャッシュポリシーまたはオリジンリクエストポリシー](#)を使用することをお勧めします。

エンティティ	デフォルトのクォータ
キャッシュ動作あたりの Cookie	10
詳細については、「 Cookie に基づくコンテンツのキャッシュ 」を参照してください。	クォータ引き上げのリクエスト
Cookie 名の合計バイト数 (すべての Cookie をオリジンに転送 CloudFront するように設定した場合は適用されません)	512 から Cookie の数を引いたもの

クエリ文字列のクォータ (従来のキャッシュ設定)

これらのクォータは、CloudFrontのレガシーキャッシュ設定に適用されます。従来の設定の代わりに、[キャッシュポリシーまたはオリジンリクエストポリシー](#)を使用することをお勧めします。

エンティティ	デフォルトのクォータ
クエリ文字列の最大文字数	128 文字
同じパラメータ内のすべてのクエリ文字列の最大合計文字数	512 文字
キャッシュ動作あたりのクエリ文字列	10
詳細については、「 クエリ文字列パラメータに基づくコンテンツのキャッシュ 」を参照してください。	クォータ引き上げのリクエスト

ヘッダーのクォータ

エンティティ	デフォルトのクォータ
キャッシュ動作ごとのヘッダー (従来のキャッシュ設定)	10
詳細については、「 the section called “リクエストヘッダーに基づくコンテンツのキャッシュ” 」を参照してください。	クォータ引き上げのリクエスト
カスタムヘッダー: オリジンリクエストに追加する CloudFront ように設定できるカスタムヘッダーの最大数	10
詳細については、「 the section called “オリジンリクエストへのカスタムヘッダーの追加” 」を参照してください。	クォータ引き上げのリクエスト
カスタムヘッダー: レスポンスヘッダーポリシーに追加できるカスタムヘッダーの最大数	10
	クォータ引き上げのリクエスト
カスタムヘッダーあり: ヘッダー名の最大長	256 文字
カスタムヘッダー: ヘッダー値の最大長	1,783 文字
カスタムヘッダー: 結合されるすべてのヘッダー値および名前の最大長	10,240 文字

エンティティ	デフォルトのクォータ
Content-Security-Policy ヘッダーの値の最大長	1,783 文字 クォータ引き上げのリクエスト

AWS SDKs CloudFront を使用するためのコード例

次のコード例は、Software AWS Development Kit (SDK) CloudFront で を使用する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、「[AWS SDK CloudFront で使用する](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

コードサンプル

- [AWS SDKs CloudFront を使用するためのアクション](#)
 - [AWS SDK を使用して CloudFront ディストリビューションを作成する](#)
 - [AWS SDK を使用して CloudFront 関数を作成する](#)
 - [AWS SDK CloudFront を使用して のキーグループを作成する](#)
 - [AWS SDK を使用して CloudFront ディストリビューションを削除する](#)
 - [AWS SDK を使用して CloudFront 署名リソースを削除する](#)
 - [AWS SDK を使用して CloudFront ディストリビューション設定を取得する](#)
 - [AWS SDK を使用して CloudFront ディストリビューションを一覧表示する](#)
 - [AWS SDK を使用して CloudFront ディストリビューションを更新する](#)
 - [AWS SDK CloudFront を使用してパブリックキーを にアップロードする](#)
- [AWS SDKs CloudFront を使用するシナリオ](#)
 - [AWS SDK を使用して署名付き URL と Cookie を作成する](#)

AWS SDKs CloudFront を使用するためのアクション

以下は、AWS SDK を使用して個々の CloudFront アクションを実行する方法を説明するコード例です。これらは CloudFront API を呼び出すもので、コンテキスト内で実行する必要がある大規模なプ

プログラムからのコード抜粋です。各例には GitHub、コードの設定と実行の手順を示すへのリンクが含まれています。

以下の例には、最も一般的に使用されるアクションのみ含まれています。詳細なリストについては、「[Amazon CloudFront API リファレンス](#)」を参照してください。

例

- [AWS SDK を使用して CloudFront ディストリビューションを作成する](#)
- [AWS SDK を使用して CloudFront 関数を作成する](#)
- [AWS SDK CloudFront を使用してのキーグループを作成する](#)
- [AWS SDK を使用して CloudFront ディストリビューションを削除する](#)
- [AWS SDK を使用して CloudFront 署名リソースを削除する](#)
- [AWS SDK を使用して CloudFront ディストリビューション設定を取得する](#)
- [AWS SDK を使用して CloudFront ディストリビューションを一覧表示する](#)
- [AWS SDK を使用して CloudFront ディストリビューションを更新する](#)
- [AWS SDK CloudFront を使用してパブリックキーを にアップロードする](#)

AWS SDK を使用して CloudFront ディストリビューションを作成する

次のコード例は、CloudFront ディストリビューションを作成する方法を示しています。

CLI

AWS CLI

CloudFront ディストリビューションを作成するには

以下の例では、`awsexamplebucket` という名前の S3 バケットのディストリビューションを作成し、コマンドライン引数を使用してデフォルトのルートオブジェクトとして `index.html` を指定しています。

```
aws cloudfront create-distribution \  
  --origin-domain-name awsexamplebucket.s3.amazonaws.com \  
  --default-root-object index.html
```

次の例に示すように、コマンドライン引数を使用する代わりに、JSON ファイルでディストリビューション設定を指定できます。

```
aws cloudfront create-distribution \  
  --distribution-config file://dist-config.json
```

dist-config.json ファイルは、以下を含む現在のフォルダ内にある JSON ドキュメントです。

```
{  
  "CallerReference": "cli-example",  
  "Aliases": {  
    "Quantity": 0  
  },  
  "DefaultRootObject": "index.html",  
  "Origins": {  
    "Quantity": 1,  
    "Items": [  
      {  
        "Id": "awsexamplebucket.s3.amazonaws.com-cli-example",  
        "DomainName": "awsexamplebucket.s3.amazonaws.com",  
        "OriginPath": "",  
        "CustomHeaders": {  
          "Quantity": 0  
        },  
        "S3OriginConfig": {  
          "OriginAccessIdentity": ""  
        }  
      }  
    ]  
  },  
  "OriginGroups": {  
    "Quantity": 0  
  },  
  "DefaultCacheBehavior": {  
    "TargetOriginId": "awsexamplebucket.s3.amazonaws.com-cli-example",  
    "ForwardedValues": {  
      "QueryString": false,  
      "Cookies": {  
        "Forward": "none"  
      },  
    },  
    "Headers": {  
      "Quantity": 0  
    },  
    "QueryStringCacheKeys": {  
      "Quantity": 0  
    }  
  }  
}
```

```
    }
  },
  "TrustedSigners": {
    "Enabled": false,
    "Quantity": 0
  },
  "ViewerProtocolPolicy": "allow-all",
  "MinTTL": 0,
  "AllowedMethods": {
    "Quantity": 2,
    "Items": [
      "HEAD",
      "GET"
    ],
    "CachedMethods": {
      "Quantity": 2,
      "Items": [
        "HEAD",
        "GET"
      ]
    }
  },
  "SmoothStreaming": false,
  "DefaultTTL": 86400,
  "MaxTTL": 31536000,
  "Compress": false,
  "LambdaFunctionAssociations": {
    "Quantity": 0
  },
  "FieldLevelEncryptionId": ""
},
"CacheBehaviors": {
  "Quantity": 0
},
"CustomErrorResponses": {
  "Quantity": 0
},
"Comment": "",
"Logging": {
  "Enabled": false,
  "IncludeCookies": false,
  "Bucket": "",
  "Prefix": ""
},
},
```

```
"PriceClass": "PriceClass_All",
"Enabled": true,
"ViewerCertificate": {
  "CloudFrontDefaultCertificate": true,
  "MinimumProtocolVersion": "TLSv1",
  "CertificateSource": "cloudfront"
},
"Restrictions": {
  "GeoRestriction": {
    "RestrictionType": "none",
    "Quantity": 0
  }
},
"WebACLId": "",
"HttpVersion": "http2",
"IsIPV6Enabled": true
}
```

ディストリビューション情報をコマンドライン引数で指定する場合も、JSON ファイルで指定する場合も、出力は変わりません。

```
{
  "Location": "https://cloudfront.amazonaws.com/2019-03-26/distribution/
EMLARXS9EXAMPLE",
  "ETag": "E9LHASXEXAMPLE",
  "Distribution": {
    "Id": "EMLARXS9EXAMPLE",
    "ARN": "arn:aws:cloudfront::123456789012:distribution/EMLARXS9EXAMPLE",
    "Status": "InProgress",
    "LastModifiedTime": "2019-11-22T00:55:15.705Z",
    "InProgressInvalidationBatches": 0,
    "DomainName": "d111111abcdef8.cloudfront.net",
    "ActiveTrustedSigners": {
      "Enabled": false,
      "Quantity": 0
    },
    "DistributionConfig": {
      "CallerReference": "cli-example",
      "Aliases": {
        "Quantity": 0
      },
      "DefaultRootObject": "index.html",
      "Origins": {
```

```
    "Quantity": 1,
    "Items": [
      {
        "Id": "awsexamplebucket.s3.amazonaws.com-cli-example",
        "DomainName": "awsexamplebucket.s3.amazonaws.com",
        "OriginPath": "",
        "CustomHeaders": {
          "Quantity": 0
        },
        "S3OriginConfig": {
          "OriginAccessIdentity": ""
        }
      }
    ],
    "OriginGroups": {
      "Quantity": 0
    },
    "DefaultCacheBehavior": {
      "TargetOriginId": "awsexamplebucket.s3.amazonaws.com-cli-
example",
      "ForwardedValues": {
        "QueryString": false,
        "Cookies": {
          "Forward": "none"
        },
        "Headers": {
          "Quantity": 0
        },
        "QueryStringCacheKeys": {
          "Quantity": 0
        }
      },
      "TrustedSigners": {
        "Enabled": false,
        "Quantity": 0
      },
      "ViewerProtocolPolicy": "allow-all",
      "MinTTL": 0,
      "AllowedMethods": {
        "Quantity": 2,
        "Items": [
          "HEAD",
          "GET"
        ]
      }
    }
  }
}
```

```
    ],
    "CachedMethods": {
      "Quantity": 2,
      "Items": [
        "HEAD",
        "GET"
      ]
    }
  },
  "SmoothStreaming": false,
  "DefaultTTL": 86400,
  "MaxTTL": 31536000,
  "Compress": false,
  "LambdaFunctionAssociations": {
    "Quantity": 0
  },
  "FieldLevelEncryptionId": ""
},
"CacheBehaviors": {
  "Quantity": 0
},
"CustomErrorResponses": {
  "Quantity": 0
},
"Comment": "",
"Logging": {
  "Enabled": false,
  "IncludeCookies": false,
  "Bucket": "",
  "Prefix": ""
},
"PriceClass": "PriceClass_All",
"Enabled": true,
"ViewerCertificate": {
  "CloudFrontDefaultCertificate": true,
  "MinimumProtocolVersion": "TLSv1",
  "CertificateSource": "cloudfront"
},
"Restrictions": {
  "GeoRestriction": {
    "RestrictionType": "none",
    "Quantity": 0
  }
},
},
```

```
        "WebACLIId": "",
        "HttpVersion": "http2",
        "IsIPV6Enabled": true
    }
}
```

- APIの詳細については、「コマンドリファレンス[CreateDistribution](#)」の「」を参照してください。AWS CLI

Java

SDK for Java 2.x

Note

については、さらに詳しく説明します [GitHub](#)。用例一覧を検索し、[AWS コードサンプルリポジトリ](#)での設定と実行の方法を確認してください。

次の例では、Amazon Simple Storage Service (Amazon S3) バケットをコンテンツオリジンとして使用しています。

ディストリビューションを作成した後、コードはディストリビューションがデプロイされるまで待機[CloudFrontWaiter](#)する を作成し、ディストリビューションを返します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
    software.amazon.awssdk.services.cloudfront.model.CreateDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.ItemSelection;
import software.amazon.awssdk.services.cloudfront.model.Method;
import software.amazon.awssdk.services.cloudfront.model.ViewerProtocolPolicy;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;
import software.amazon.awssdk.services.s3.S3Client;
```

```
import java.time.Instant;

public class CreateDistribution {

    private static final Logger logger =
    LoggerFactory.getLogger(CreateDistribution.class);

    public static Distribution createDistribution(CloudFrontClient
    cloudFrontClient, S3Client s3Client,
        final String bucketName, final String keyGroupId, final
    String originAccessControlId) {

        final String region = s3Client.headBucket(b ->
    b.bucket(bucketName)).sdkHttpResponse().headers()
            .get("x-amz-bucket-region").get(0);
        final String originDomain = bucketName + ".s3." + region +
    ".amazonaws.com";
        String originId = originDomain; // Use the originDomain value for
    the originId.

        // The service API requires some deprecated methods, such as
        // DefaultCacheBehavior.Builder#minTTL and #forwardedValue.
        CreateDistributionResponse createDistResponse =
    cloudFrontClient.createDistribution(builder -> builder
            .distributionConfig(b1 -> b1
                .origins(b2 -> b2
                    .quantity(1)
                    .items(b3 -> b3

    .domainName(originDomain)

    .id(originId)

    .s3OriginConfig(builder4 -> builder4
        .originAccessIdentity(
            ""))

    .originAccessControlId(
        originAccessControlId)))

        .defaultCacheBehavior(b2 -> b2
```

```
.viewerProtocolPolicy(ViewerProtocolPolicy.ALLOW_ALL)

.targetOriginId(originId)

.minTTL(200L)

.forwardedValues(b5 -> b5

.cookies(cp -> cp

    .forward(ItemSelection.NONE))

.queryString(true))

.trustedKeyGroups(b3 -> b3

.quantity(1)

.items(keyGroupId)

.enabled(true))

.allowedMethods(b4 -> b4

.quantity(2)

.items(Method.HEAD, Method.GET)

.cachedMethods(b5 -> b5

    .quantity(2)

    .items(Method.HEAD,

        Method.GET))))

.cacheBehaviors(b -> b

    .quantity(1)

    .items(b2 -> b2

.pathPattern("/index.html")

.viewerProtocolPolicy(

    ViewerProtocolPolicy.ALLOW_ALL)
```

```
.targetOriginId(originId)

.trustedKeyGroups(b3 -> b3
    .quantity(1)
    .items(keyGroupId)
    .enabled(true))

.minTTL(200L)

.forwardedValues(b4 -> b4
    .cookies(cp -> cp
        .forward(ItemSelection.NONE))
    .queryString(true))

.allowedMethods(b5 -> b5.quantity(2)
    .items(Method.HEAD,
        Method.GET)
    .cachedMethods(b6 -> b6
        .quantity(2)
        .items(Method.HEAD,
            Method.GET))))
    .enabled(true)
    .comment("Distribution built with
java")

.callerReference(Instant.now().toString()));

    final Distribution distribution =
createDistResponse.distribution();
    logger.info("Distribution created. DomainName: [{}] Id: [{}]",
distribution.domainName(),
```

```
        distribution.id());
        logger.info("Waiting for distribution to be deployed ...");
        try (CloudFrontWaiter cfWaiter =
CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
            ResponseOrException<GetDistributionResponse>
responseOrException = cfWaiter
                .waitUntilDistributionDeployed(builder ->
builder.id(distribution.id()))
                .matched();
            responseOrException.response()
                .orElseThrow(() -> new
RuntimeException("Distribution not created"));
            logger.info("Distribution deployed. DomainName: [{}] Id:
[{}]", distribution.domainName(),
                distribution.id());
        }
        return distribution;
    }
}
```

- APIの詳細については、「APIリファレンス[CreateDistribution](#)」の「」を参照してください。AWS SDK for Java 2.x

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、「[AWS SDK CloudFront で使用する](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して CloudFront 関数を作成する

次のコード例は、Amazon CloudFront 関数を作成する方法を示しています。

Java

SDK for Java 2.x

Note

については、さらに詳しく説明します GitHub。用例一覧を検索し、[AWS コードサンプリング](#)での設定と実行の方法を確認してください。

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CloudFrontException;
import software.amazon.awssdk.services.cloudfront.model.CreateFunctionRequest;
import software.amazon.awssdk.services.cloudfront.model.CreateFunctionResponse;
import software.amazon.awssdk.services.cloudfront.model.FunctionConfig;
import software.amazon.awssdk.services.cloudfront.model.FunctionRuntime;
import java.io.InputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateFunction {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <functionName> <filePath>

            Where:
                functionName - The name of the function to create.\s
                filePath - The path to a file that contains the application
            logic for the function.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String functionName = args[0];
        String filePath = args[1];
        CloudFrontClient cloudFrontClient = CloudFrontClient.builder()
            .region(Region.AWS_GLOBAL)
            .build();
```

```
        String funArn = createNewFunction(cloudFrontClient, functionName,
filePath);
        System.out.println("The function ARN is " + funArn);
        cloudFrontClient.close();
    }

    public static String createNewFunction(CloudFrontClient cloudFrontClient,
String functionName, String filePath) {
        try {
            InputStream fileIs =
CreateFunction.class.getClassLoader().getResourceAsStream(filePath);
            SdkBytes functionCode = SdkBytes.fromInputStream(fileIs);

            FunctionConfig config = FunctionConfig.builder()
                .comment("Created by using the CloudFront Java API")
                .runtime(FunctionRuntime.CLOUDFRONT_JS_1_0)
                .build();

            CreateFunctionRequest functionRequest =
CreateFunctionRequest.builder()
                .name(functionName)
                .functionCode(functionCode)
                .functionConfig(config)
                .build();

            CreateFunctionResponse response =
cloudFrontClient.createFunction(functionRequest);
            return response.functionSummary().functionMetadata().functionARN();

        } catch (CloudFrontException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- APIの詳細については、「APIリファレンス[CreateFunction](#)」の「」を参照してください。
AWS SDK for Java 2.x

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、「[AWS SDK CloudFront を使用する](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK CloudFront を使用して のキーグループを作成する

次のコード例は、署名付き URL と署名付き Cookie で使用できるキーグループを作成する方法を示しています。

Java

SDK for Java 2.x

Note

については、さらに詳しく説明します [GitHub](#)。用例一覧を検索し、[AWS コードサンプルリポジトリ](#)での設定と実行の方法を確認してください。

キーグループには、署名付き URL または Cookie の検証に使用されるパブリックキーが少なくとも 1 つ必要です。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;

import java.util.UUID;

public class CreateKeyGroup {
    private static final Logger logger =
        LoggerFactory.getLogger(CreateKeyGroup.class);

    public static String createKeyGroup(CloudFrontClient cloudFrontClient, String
publicKeyId) {
        String keyGroupId = cloudFrontClient.createKeyGroup(b ->
b.keyGroupConfig(c -> c
            .items(publicKeyId)
            .name("JavaKeyGroup" + UUID.randomUUID()))
            .keyGroup().id());
        logger.info("KeyGroup created with ID: [{}]", keyGroupId);
        return keyGroupId;
    }
}
```

```
}  
}
```

- APIの詳細については、「API リファレンス [CreateKeyGroup](#)」の「」を参照してください。AWS SDK for Java 2.x

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、「[AWS SDK CloudFront で使用する](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して CloudFront デイストリビューションを削除する

次のコード例は、CloudFront デイストリビューションを削除する方法を示しています。

CLI

AWS CLI

CloudFront デイストリビューションを削除するには

次の例では、ID が の CloudFront デイストリビューションを削除します EDFDVBD6EXAMPLE。デイストリビューションを削除する前に、デイストリビューションを無効にする必要があります。デイストリビューションを無効にするには、update-distribution コマンドを使用します。詳細については、update-distribution の例を参照してください。

デイストリビューションを無効にすると、デイストリビューションを削除できます。デイストリビューションを削除するには、--if-match オプションを使用してデイストリビューションのETag を指定する必要があります。を取得するにはETag、get-distribution または get-distribution-config コマンドを使用します。

```
aws cloudfront delete-distribution \  
  --id EDFDVBD6EXAMPLE \  
  --if-match E2QWRUHEXAMPLE
```

成功した場合は、コマンドの出力はありません。

- APIの詳細については、「AWS CLI コマンドリファレンス」で以下のトピックを参照してください。
 - [DeleteDistribution](#)

- [UpdateDistribution](#)

Java

SDK for Java 2.x

Note

については、さらに詳しく説明します GitHub。用例一覧を検索し、[AWS コードサンプルリポジトリ](#)での設定と実行の方法を確認してください。

次のコード例では、配信を無効状態に更新し、ウェイターを使用して変更がデプロイされるのを待ってから、ディストリビューションを削除します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
    software.amazon.awssdk.services.cloudfront.model.DeleteDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionConfig;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;

public class DeleteDistribution {
    private static final Logger logger =
        LoggerFactory.getLogger(DeleteDistribution.class);

    public static void deleteDistribution(final CloudFrontClient
        cloudFrontClient, final String distributionId) {
        // First, disable the distribution by updating it.
        GetDistributionResponse response =
            cloudFrontClient.getDistribution(b -> b
                .id(distributionId));
        String etag = response.eTag();
        DistributionConfig distConfig =
            response.distribution().distributionConfig();

        cloudFrontClient.updateDistribution(builder -> builder
            .id(distributionId)
```

```

        .distributionConfig(builder1 -> builder1

.cacheBehaviors(distConfig.cacheBehaviors())

.defaultCacheBehavior(distConfig.defaultCacheBehavior())
                                .enabled(false)
                                .origins(distConfig.origins())
                                .comment(distConfig.comment())

.callerReference(distConfig.callerReference())

.defaultCacheBehavior(distConfig.defaultCacheBehavior())

.priceClass(distConfig.priceClass())
                                .aliases(distConfig.aliases())
                                .logging(distConfig.logging())

.defaultRootObject(distConfig.defaultRootObject())

.customErrorResponses(distConfig.customErrorResponses())

.httpVersion(distConfig.httpVersion())

.isIPV6Enabled(distConfig.isIPV6Enabled())

.restrictions(distConfig.restrictions())

.viewerCertificate(distConfig.viewerCertificate())
                                .webACLId(distConfig.webACLId())

.originGroups(distConfig.originGroups())
                .ifMatch(etag));

        logger.info("Distribution [{}] is DISABLED, waiting for
deployment before deleting ...",
                    distributionId);
        GetDistributionResponse distributionResponse;
        try (CloudFrontWaiter cfWaiter =
CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
            ResponseOrException<GetDistributionResponse>
responseOrException = cfWaiter
                                .waitUntilDistributionDeployed(builder ->
builder.id(distributionId)).matched();
            distributionResponse = responseOrException.response()

```

```
                .orElseThrow(() -> new
RuntimeException("Could not disable distribution"));
            }

            DeleteDistributionResponse deleteDistributionResponse =
cloudFrontClient
                .deleteDistribution(builder -> builder
                    .id(distributionId)

.ifMatch(distributionResponse.eTag()));
            if (deleteDistributionResponse.sdkHttpResponse().isSuccessful())
{
                logger.info("Distribution [{}] DELETED", distributionId);
            }
        }
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API Reference」の以下のトピックを参照してください。
 - [DeleteDistribution](#)
 - [UpdateDistribution](#)

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、「[AWS SDK CloudFront で使用する](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して CloudFront 署名リソースを削除する

次のコード例は、Amazon Simple Storage Service (Amazon S3) バケット内の制限付きコンテンツへのアクセス権を取得するために使用されるリソースを削除する方法を示しています。

Java

SDK for Java 2.x

Note

については、さらに詳しく説明します GitHub。用例一覧を検索し、[AWS コードサンプルリポジトリ](#)での設定と実行の方法を確認してください。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.DeleteKeyGroupResponse;
import
    software.amazon.awssdk.services.cloudfront.model.DeleteOriginAccessControlResponse;
import software.amazon.awssdk.services.cloudfront.model.DeletePublicKeyResponse;
import software.amazon.awssdk.services.cloudfront.model.GetKeyGroupResponse;
import
    software.amazon.awssdk.services.cloudfront.model.GetOriginAccessControlResponse;
import software.amazon.awssdk.services.cloudfront.model.GetPublicKeyResponse;

public class DeleteSigningResources {
    private static final Logger logger =
        LoggerFactory.getLogger(DeleteSigningResources.class);

    public static void deleteOriginAccessControl(final CloudFrontClient
        cloudFrontClient,
        final String originAccessControlId) {
        GetOriginAccessControlResponse getResponse = cloudFrontClient
            .getOriginAccessControl(b -> b.id(originAccessControlId));
        DeleteOriginAccessControlResponse deleteResponse =
            cloudFrontClient.deleteOriginAccessControl(builder -> builder
                .id(originAccessControlId)
                .ifMatch(getResponse.eTag()));
        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Origin Access Control [{}]",
                originAccessControlId);
        }
    }

    public static void deleteKeyGroup(final CloudFrontClient cloudFrontClient,
        final String keyGroupId) {

        GetKeyGroupResponse getResponse = cloudFrontClient.getKeyGroup(b ->
            b.id(keyGroupId));
        DeleteKeyGroupResponse deleteResponse =
            cloudFrontClient.deleteKeyGroup(builder -> builder
                .id(keyGroupId)
                .ifMatch(getResponse.eTag()));
        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Key Group [{}]", keyGroupId);
        }
    }
}
```

```
    }

    public static void deletePublicKey(final CloudFrontClient cloudFrontClient,
final String publicKeyId) {
        GetPublicKeyResponse getResponse = cloudFrontClient.getPublicKey(b ->
b.id(publicKeyId));

        DeletePublicKeyResponse deleteResponse =
cloudFrontClient.deletePublicKey(builder -> builder
            .id(publicKeyId)
            .ifMatch(getResponse.eTag()));

        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Public Key [{}]", publicKeyId);
        }
    }
}
```

- API の詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。
 - [DeleteKeyGroup](#)
 - [DeleteOriginAccessControl](#)
 - [DeletePublicKey](#)

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、「[AWS SDK CloudFront で使用する](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して CloudFront デイストリビューション設定を取得する

次のコード例は、Amazon CloudFront デイストリビューション設定を取得する方法を示しています。

CLI

AWS CLI

CloudFront デイストリビューション設定を取得するには

次の例では、EDFDVBD6EXAMPLEを含む ID を持つ CloudFront デイストリビューションに関するメタデータを取得しますETag。デイストリビューション ID は create-distribution コマンドと list-distributions コマンドで返されます。

```
aws cloudfront get-distribution-config --id EDFDVBD6EXAMPLE
```

出力:

```
{
  "ETag": "E2QWRUHEXAMPLE",
  "DistributionConfig": {
    "CallerReference": "cli-example",
    "Aliases": {
      "Quantity": 0
    },
    "DefaultRootObject": "index.html",
    "Origins": {
      "Quantity": 1,
      "Items": [
        {
          "Id": "awsexamplebucket.s3.amazonaws.com-cli-example",
          "DomainName": "awsexamplebucket.s3.amazonaws.com",
          "OriginPath": "",
          "CustomHeaders": {
            "Quantity": 0
          },
          "S3OriginConfig": {
            "OriginAccessIdentity": ""
          }
        }
      ]
    },
    "OriginGroups": {
      "Quantity": 0
    },
    "DefaultCacheBehavior": {
      "TargetOriginId": "awsexamplebucket.s3.amazonaws.com-cli-example",
      "ForwardedValues": {
        "QueryString": false,
        "Cookies": {
          "Forward": "none"
        }
      },
      "Headers": {
```

```
        "Quantity": 0
      },
      "QueryStringCacheKeys": {
        "Quantity": 0
      }
    },
    "TrustedSigners": {
      "Enabled": false,
      "Quantity": 0
    },
    "ViewerProtocolPolicy": "allow-all",
    "MinTTL": 0,
    "AllowedMethods": {
      "Quantity": 2,
      "Items": [
        "HEAD",
        "GET"
      ],
      "CachedMethods": {
        "Quantity": 2,
        "Items": [
          "HEAD",
          "GET"
        ]
      }
    },
    "SmoothStreaming": false,
    "DefaultTTL": 86400,
    "MaxTTL": 31536000,
    "Compress": false,
    "LambdaFunctionAssociations": {
      "Quantity": 0
    },
    "FieldLevelEncryptionId": ""
  },
  "CacheBehaviors": {
    "Quantity": 0
  },
  "CustomErrorResponses": {
    "Quantity": 0
  },
  "Comment": "",
  "Logging": {
    "Enabled": false,
```

```
        "IncludeCookies": false,
        "Bucket": "",
        "Prefix": ""
    },
    "PriceClass": "PriceClass_All",
    "Enabled": true,
    "ViewerCertificate": {
        "CloudFrontDefaultCertificate": true,
        "MinimumProtocolVersion": "TLSv1",
        "CertificateSource": "cloudfront"
    },
    "Restrictions": {
        "GeoRestriction": {
            "RestrictionType": "none",
            "Quantity": 0
        }
    },
    "WebACLId": "",
    "HttpVersion": "http2",
    "IsIPV6Enabled": true
}
}
```

- APIの詳細については、「コマンドリファレンス[GetDistributionConfig](#)」の「」を参照してください。AWS CLI

Python

SDK for Python (Boto3)

Note

については、さらに詳しく説明します GitHub。用例一覧を検索し、[AWS コードサンプルリポジトリ](#)での設定と実行の方法を確認してください。

```
class CloudFrontWrapper:
    """Encapsulates Amazon CloudFront operations."""

    def __init__(self, cloudfront_client):
        """
```

```
:param cloudfront_client: A Boto3 CloudFront client
"""
self.cloudfront_client = cloudfront_client

def update_distribution(self):
    distribution_id = input(
        "This script updates the comment for a CloudFront distribution.\n"
        "Enter a CloudFront distribution ID: "
    )

    distribution_config_response =
self.cloudfront_client.get_distribution_config(
        Id=distribution_id
    )
    distribution_config = distribution_config_response["DistributionConfig"]
    distribution_etag = distribution_config_response["ETag"]

    distribution_config["Comment"] = input(
        f"\nThe current comment for distribution {distribution_id} is "
        f"'{distribution_config['Comment']}'.\n"
        f"Enter a new comment: "
    )
    self.cloudfront_client.update_distribution(
        DistributionConfig=distribution_config,
        Id=distribution_id,
        IfMatch=distribution_etag,
    )
    print("Done!")
```

- API の詳細については、[GetDistributionConfig](#) AWS 「 SDK for Python (Boto3) API リファレンス」の「」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、「[AWS SDK CloudFront で使用する](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して CloudFront デイストリビューションを一覧表示する

次のコード例は、Amazon CloudFront デイストリビューションを一覧表示する方法を示しています。

CLI

AWS CLI

CloudFront デイストリビューションを一覧表示するには

次の例では、AWSアカウント内の CloudFront デイストリビューションのリストを取得します。

```
aws cloudfront list-distributions
```

出力:

```
{
  "DistributionList": {
    "Items": [
      {
        "Id": "EMLARXS9EXAMPLE",
        "ARN": "arn:aws:cloudfront::123456789012:distribution/EMLARXS9EXAMPLE",
        "Status": "InProgress",
        "LastModifiedTime": "2019-11-22T00:55:15.705Z",
        "InProgressInvalidationBatches": 0,
        "DomainName": "d1111111abcdef8.cloudfront.net",
        "ActiveTrustedSigners": {
          "Enabled": false,
          "Quantity": 0
        },
        "DistributionConfig": {
          "CallerReference": "cli-example",
          "Aliases": {
            "Quantity": 0
          },
          "DefaultRootObject": "index.html",
          "Origins": {
            "Quantity": 1,
            "Items": [
              {
```

```
        "Id": "awsexamplebucket.s3.amazonaws.com-cli-
example",
        "DomainName":
"awsexamplebucket.s3.amazonaws.com",
        "OriginPath": "",
        "CustomHeaders": {
            "Quantity": 0
        },
        "S3OriginConfig": {
            "OriginAccessIdentity": ""
        }
    }
],
},
"OriginGroups": {
    "Quantity": 0
},
"DefaultCacheBehavior": {
    "TargetOriginId": "awsexamplebucket.s3.amazonaws.com-cli-
example",
    "ForwardedValues": {
        "QueryString": false,
        "Cookies": {
            "Forward": "none"
        },
        "Headers": {
            "Quantity": 0
        },
        "QueryStringCacheKeys": {
            "Quantity": 0
        }
    },
    "TrustedSigners": {
        "Enabled": false,
        "Quantity": 0
    },
    "ViewerProtocolPolicy": "allow-all",
    "MinTTL": 0,
    "AllowedMethods": {
        "Quantity": 2,
        "Items": [
            "HEAD",
            "GET"
        ],
    },

```

```
        "CachedMethods": {
            "Quantity": 2,
            "Items": [
                "HEAD",
                "GET"
            ]
        }
    },
    "SmoothStreaming": false,
    "DefaultTTL": 86400,
    "MaxTTL": 31536000,
    "Compress": false,
    "LambdaFunctionAssociations": {
        "Quantity": 0
    },
    "FieldLevelEncryptionId": ""
},
"CacheBehaviors": {
    "Quantity": 0
},
"CustomErrorResponses": {
    "Quantity": 0
},
"Comment": "",
"Logging": {
    "Enabled": false,
    "IncludeCookies": false,
    "Bucket": "",
    "Prefix": ""
},
"PriceClass": "PriceClass_All",
"Enabled": true,
"ViewerCertificate": {
    "CloudFrontDefaultCertificate": true,
    "MinimumProtocolVersion": "TLSv1",
    "CertificateSource": "cloudfront"
},
"Restrictions": {
    "GeoRestriction": {
        "RestrictionType": "none",
        "Quantity": 0
    }
},
"WebACLId": "",
```

```
        "HttpVersion": "http2",
        "IsIPV6Enabled": true
    }
},
{
    "Id": "EDFDVBD6EXAMPLE",
    "ARN": "arn:aws:cloudfront::123456789012:distribution/EDFDVBD6EXAMPLE",
    "Status": "InProgress",
    "LastModifiedTime": "2019-12-04T23:35:41.433Z",
    "InProgressInvalidationBatches": 0,
    "DomainName": "d930174dauwrn8.cloudfront.net",
    "ActiveTrustedSigners": {
        "Enabled": false,
        "Quantity": 0
    },
    "DistributionConfig": {
        "CallerReference": "cli-example",
        "Aliases": {
            "Quantity": 0
        },
        "DefaultRootObject": "index.html",
        "Origins": {
            "Quantity": 1,
            "Items": [
                {
                    "Id": "awsexamplebucket1.s3.amazonaws.com-cli-example",
                    "DomainName": "awsexamplebucket1.s3.amazonaws.com",
                    "OriginPath": "",
                    "CustomHeaders": {
                        "Quantity": 0
                    },
                    "S3OriginConfig": {
                        "OriginAccessIdentity": ""
                    }
                }
            ]
        },
        "OriginGroups": {
            "Quantity": 0
        },
        "DefaultCacheBehavior": {
```

```
cli-example",
  "TargetOriginId": "awsexamplebucket1.s3.amazonaws.com-
  "ForwardedValues": {
    "QueryString": false,
    "Cookies": {
      "Forward": "none"
    },
    "Headers": {
      "Quantity": 0
    },
    "QueryStringCacheKeys": {
      "Quantity": 0
    }
  },
  "TrustedSigners": {
    "Enabled": false,
    "Quantity": 0
  },
  "ViewerProtocolPolicy": "allow-all",
  "MinTTL": 0,
  "AllowedMethods": {
    "Quantity": 2,
    "Items": [
      "HEAD",
      "GET"
    ],
    "CachedMethods": {
      "Quantity": 2,
      "Items": [
        "HEAD",
        "GET"
      ]
    }
  },
  "SmoothStreaming": false,
  "DefaultTTL": 86400,
  "MaxTTL": 31536000,
  "Compress": false,
  "LambdaFunctionAssociations": {
    "Quantity": 0
  },
  "FieldLevelEncryptionId": ""
},
"CacheBehaviors": {
```

```
        "Quantity": 0
      },
      "CustomErrorResponses": {
        "Quantity": 0
      },
      "Comment": "",
      "Logging": {
        "Enabled": false,
        "IncludeCookies": false,
        "Bucket": "",
        "Prefix": ""
      },
      "PriceClass": "PriceClass_All",
      "Enabled": true,
      "ViewerCertificate": {
        "CloudFrontDefaultCertificate": true,
        "MinimumProtocolVersion": "TLSv1",
        "CertificateSource": "cloudfront"
      },
      "Restrictions": {
        "GeoRestriction": {
          "RestrictionType": "none",
          "Quantity": 0
        }
      },
      "WebACLId": "",
      "HttpVersion": "http2",
      "IsIPV6Enabled": true
    }
  },
  {
    "Id": "E1X5IZQEXAMPLE",
    "ARN": "arn:aws:cloudfront::123456789012:distribution/
E1X5IZQEXAMPLE",
    "Status": "Deployed",
    "LastModifiedTime": "2019-11-06T21:31:48.864Z",
    "DomainName": "d2e04y12345678.cloudfront.net",
    "Aliases": {
      "Quantity": 0
    },
    "Origins": {
      "Quantity": 1,
      "Items": [
        {
```

```
        "Id": "awsexamplebucket2",
        "DomainName": "awsexamplebucket2.s3.us-
west-2.amazonaws.com",
        "OriginPath": "",
        "CustomHeaders": {
            "Quantity": 0
        },
        "S3OriginConfig": {
            "OriginAccessIdentity": ""
        }
    }
]
},
"OriginGroups": {
    "Quantity": 0
},
"DefaultCacheBehavior": {
    "TargetOriginId": "awsexamplebucket2",
    "ForwardedValues": {
        "QueryString": false,
        "Cookies": {
            "Forward": "none"
        },
        "Headers": {
            "Quantity": 0
        },
        "QueryStringCacheKeys": {
            "Quantity": 0
        }
    },
    "TrustedSigners": {
        "Enabled": false,
        "Quantity": 0
    },
    "ViewerProtocolPolicy": "allow-all",
    "MinTTL": 0,
    "AllowedMethods": {
        "Quantity": 2,
        "Items": [
            "HEAD",
            "GET"
        ],
        "CachedMethods": {
            "Quantity": 2,
```

```
        "Items": [
            "HEAD",
            "GET"
        ]
    },
    "SmoothStreaming": false,
    "DefaultTTL": 86400,
    "MaxTTL": 31536000,
    "Compress": false,
    "LambdaFunctionAssociations": {
        "Quantity": 0
    },
    "FieldLevelEncryptionId": ""
},
"CacheBehaviors": {
    "Quantity": 0
},
"CustomErrorResponses": {
    "Quantity": 0
},
"Comment": "",
"PriceClass": "PriceClass_All",
"Enabled": true,
"ViewerCertificate": {
    "CloudFrontDefaultCertificate": true,
    "MinimumProtocolVersion": "TLSv1",
    "CertificateSource": "cloudfront"
},
"Restrictions": {
    "GeoRestriction": {
        "RestrictionType": "none",
        "Quantity": 0
    }
},
"WebACLId": "",
"HttpVersion": "HTTP1_1",
"IsIPV6Enabled": true
}
]
}
```

- APIの詳細については、「コマンドリファレンス [ListDistributions](#)」の「」を参照してください。AWS CLI

Python

SDK for Python (Boto3)

Note

については、さらに詳しく説明します GitHub。用例一覧を検索し、[AWS コードサンプルリポジトリ](#)での設定と実行の方法を確認してください。

```
class CloudFrontWrapper:
    """Encapsulates Amazon CloudFront operations."""

    def __init__(self, cloudfront_client):
        """
        :param cloudfront_client: A Boto3 CloudFront client
        """
        self.cloudfront_client = cloudfront_client

    def list_distributions(self):
        print("CloudFront distributions:\n")
        distributions = self.cloudfront_client.list_distributions()
        if distributions["DistributionList"]["Quantity"] > 0:
            for distribution in distributions["DistributionList"]["Items"]:
                print(f"Domain: {distribution['DomainName']}")
                print(f"Distribution Id: {distribution['Id']}")
                print(
                    f"Certificate Source: "
                    f"{distribution['ViewerCertificate']['CertificateSource']}"
                )
                if distribution["ViewerCertificate"]["CertificateSource"] ==
                    "acm":
                    print(
                        f"Certificate: {distribution['ViewerCertificate']
                        ['Certificate']}"
                    )
                print("")
```

```
else:
    print("No CloudFront distributions detected.")
```

- APIの詳細については、[ListDistributions](#) AWS SDK for Python (Boto3) API リファレンスの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、「[AWS SDK CloudFront で使用する](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して CloudFront デイストリビューションを更新する

次のコード例は、Amazon CloudFront デイストリビューションを更新する方法を示しています。

CLI

AWS CLI

CloudFront デイストリビューションのデフォルトルートオブジェクトを更新するには

次の例では、ID のデイスCloudFront トリビューションindex.htmlのデフォルトのルートオブジェクトを に更新しますEDFDVBD6EXAMPLE。

```
aws cloudfront update-distribution --id EDFDVBD6EXAMPLE \
  --default-root-object index.html
```

出力:

```
{
  "ETag": "E2QWRUHEXAMPLE",
  "Distribution": {
    "Id": "EDFDVBD6EXAMPLE",
    "ARN": "arn:aws:cloudfront::123456789012:distribution/EDFDVBD6EXAMPLE",
    "Status": "InProgress",
    "LastModifiedTime": "2019-12-06T18:55:39.870Z",
    "InProgressInvalidationBatches": 0,
    "DomainName": "d1111111abcdef8.cloudfront.net",
    "ActiveTrustedSigners": {
      "Enabled": false,
      "Quantity": 0
    }
  }
}
```

```
},
"DistributionConfig": {
  "CallerReference": "6b10378d-49be-4c4b-a642-419ccaf8f3b5",
  "Aliases": {
    "Quantity": 0
  },
  "DefaultRootObject": "index.html",
  "Origins": {
    "Quantity": 1,
    "Items": [
      {
        "Id": "example-website",
        "DomainName": "www.example.com",
        "OriginPath": "",
        "CustomHeaders": {
          "Quantity": 0
        },
        "CustomOriginConfig": {
          "HTTPPort": 80,
          "HTTPSPort": 443,
          "OriginProtocolPolicy": "match-viewer",
          "OriginSslProtocols": {
            "Quantity": 2,
            "Items": [
              "SSLv3",
              "TLSv1"
            ]
          },
          "OriginReadTimeout": 30,
          "OriginKeepaliveTimeout": 5
        }
      }
    ]
  },
  "OriginGroups": {
    "Quantity": 0
  },
  "DefaultCacheBehavior": {
    "TargetOriginId": "example-website",
    "ForwardedValues": {
      "QueryString": false,
      "Cookies": {
        "Forward": "none"
      }
    }
  },
}
```

```
    "Headers": {
      "Quantity": 1,
      "Items": [
        "*"
      ]
    },
    "QueryStringCacheKeys": {
      "Quantity": 0
    }
  },
  "TrustedSigners": {
    "Enabled": false,
    "Quantity": 0
  },
  "ViewerProtocolPolicy": "allow-all",
  "MinTTL": 0,
  "AllowedMethods": {
    "Quantity": 2,
    "Items": [
      "HEAD",
      "GET"
    ]
  },
  "CachedMethods": {
    "Quantity": 2,
    "Items": [
      "HEAD",
      "GET"
    ]
  }
},
"SmoothStreaming": false,
"DefaultTTL": 86400,
"MaxTTL": 31536000,
"Compress": false,
"LambdaFunctionAssociations": {
  "Quantity": 0
},
"FieldLevelEncryptionId": ""
},
"CacheBehaviors": {
  "Quantity": 0
},
"CustomErrorResponses": {
  "Quantity": 0
```

```
    },
    "Comment": "",
    "Logging": {
      "Enabled": false,
      "IncludeCookies": false,
      "Bucket": "",
      "Prefix": ""
    },
    "PriceClass": "PriceClass_All",
    "Enabled": true,
    "ViewerCertificate": {
      "CloudFrontDefaultCertificate": true,
      "MinimumProtocolVersion": "TLSv1",
      "CertificateSource": "cloudfront"
    },
    "Restrictions": {
      "GeoRestriction": {
        "RestrictionType": "none",
        "Quantity": 0
      }
    },
    "WebACLId": "",
    "HttpVersion": "http1.1",
    "IsIPV6Enabled": true
  }
}
```

CloudFront デイストリビューションを更新するには

次の例では、 という名前の JSON ファイルに CloudFront デイストリビューション設定を指定 EMLARXS9EXAMPLE して、ID でデイストリビューションを無効にします dist-config-disable.json。デイストリビューションを更新するには、--if-match オプションを使用してデイストリビューションの ETag を指定する必要があります。を取得するには ETag、get-distribution または get-distribution-config コマンドを使用します。

次の例を使用してデイストリビューションを無効にした後は、delete-distribution コマンドを使用してデイストリビューションを削除できます。

```
aws cloudfront update-distribution \
  --id EMLARXS9EXAMPLE \
  --if-match E2QWRUHEXAMPLE \
```

```
--distribution-config file://dist-config-disable.json
```

dist-config-disable.json ファイルは、以下を含む現在のフォルダ内にある JSON ドキュメントです。Enabled フィールドが false に設定されていることに注意してください。

```
{
  "CallerReference": "cli-1574382155-496510",
  "Aliases": {
    "Quantity": 0
  },
  "DefaultRootObject": "index.html",
  "Origins": {
    "Quantity": 1,
    "Items": [
      {
        "Id": "awsexamplebucket.s3.amazonaws.com-1574382155-273939",
        "DomainName": "awsexamplebucket.s3.amazonaws.com",
        "OriginPath": "",
        "CustomHeaders": {
          "Quantity": 0
        },
        "S3OriginConfig": {
          "OriginAccessIdentity": ""
        }
      }
    ]
  },
  "OriginGroups": {
    "Quantity": 0
  },
  "DefaultCacheBehavior": {
    "TargetOriginId": "awsexamplebucket.s3.amazonaws.com-1574382155-273939",
    "ForwardedValues": {
      "QueryString": false,
      "Cookies": {
        "Forward": "none"
      }
    },
    "Headers": {
      "Quantity": 0
    },
    "QueryStringCacheKeys": {
      "Quantity": 0
    }
  }
}
```

```
    }
  },
  "TrustedSigners": {
    "Enabled": false,
    "Quantity": 0
  },
  "ViewerProtocolPolicy": "allow-all",
  "MinTTL": 0,
  "AllowedMethods": {
    "Quantity": 2,
    "Items": [
      "HEAD",
      "GET"
    ],
    "CachedMethods": {
      "Quantity": 2,
      "Items": [
        "HEAD",
        "GET"
      ]
    }
  },
  "SmoothStreaming": false,
  "DefaultTTL": 86400,
  "MaxTTL": 31536000,
  "Compress": false,
  "LambdaFunctionAssociations": {
    "Quantity": 0
  },
  "FieldLevelEncryptionId": ""
},
"CacheBehaviors": {
  "Quantity": 0
},
"CustomErrorResponses": {
  "Quantity": 0
},
"Comment": "",
"Logging": {
  "Enabled": false,
  "IncludeCookies": false,
  "Bucket": "",
  "Prefix": ""
},
},
```

```
"PriceClass": "PriceClass_All",
"Enabled": false,
"ViewerCertificate": {
  "CloudFrontDefaultCertificate": true,
  "MinimumProtocolVersion": "TLSv1",
  "CertificateSource": "cloudfront"
},
"Restrictions": {
  "GeoRestriction": {
    "RestrictionType": "none",
    "Quantity": 0
  }
},
"WebACLId": "",
"HttpVersion": "http2",
"IsIPV6Enabled": true
}
```

出力:

```
{
  "ETag": "E9LHASXEXAMPLE",
  "Distribution": {
    "Id": "EMLARXS9EXAMPLE",
    "ARN": "arn:aws:cloudfront::123456789012:distribution/EMLARXS9EXAMPLE",
    "Status": "InProgress",
    "LastModifiedTime": "2019-12-06T18:32:35.553Z",
    "InProgressInvalidationBatches": 0,
    "DomainName": "d111111abcdef8.cloudfront.net",
    "ActiveTrustedSigners": {
      "Enabled": false,
      "Quantity": 0
    },
    "DistributionConfig": {
      "CallerReference": "cli-1574382155-496510",
      "Aliases": {
        "Quantity": 0
      },
      "DefaultRootObject": "index.html",
      "Origins": {
        "Quantity": 1,
        "Items": [
          {

```

```
        "Id":
"awsexamplebucket.s3.amazonaws.com-1574382155-273939",
        "DomainName": "awsexamplebucket.s3.amazonaws.com",
        "OriginPath": "",
        "CustomHeaders": {
            "Quantity": 0
        },
        "S3OriginConfig": {
            "OriginAccessIdentity": ""
        }
    }
]
},
"OriginGroups": {
    "Quantity": 0
},
"DefaultCacheBehavior": {
    "TargetOriginId":
"awsexamplebucket.s3.amazonaws.com-1574382155-273939",
    "ForwardedValues": {
        "QueryString": false,
        "Cookies": {
            "Forward": "none"
        },
        "Headers": {
            "Quantity": 0
        },
        "QueryStringCacheKeys": {
            "Quantity": 0
        }
    },
    "TrustedSigners": {
        "Enabled": false,
        "Quantity": 0
    },
    "ViewerProtocolPolicy": "allow-all",
    "MinTTL": 0,
    "AllowedMethods": {
        "Quantity": 2,
        "Items": [
            "HEAD",
            "GET"
        ],
        "CachedMethods": {
```

```
        "Quantity": 2,
        "Items": [
            "HEAD",
            "GET"
        ]
    },
    "SmoothStreaming": false,
    "DefaultTTL": 86400,
    "MaxTTL": 31536000,
    "Compress": false,
    "LambdaFunctionAssociations": {
        "Quantity": 0
    },
    "FieldLevelEncryptionId": ""
},
"CacheBehaviors": {
    "Quantity": 0
},
"CustomErrorResponses": {
    "Quantity": 0
},
"Comment": "",
"Logging": {
    "Enabled": false,
    "IncludeCookies": false,
    "Bucket": "",
    "Prefix": ""
},
"PriceClass": "PriceClass_All",
"Enabled": false,
"ViewerCertificate": {
    "CloudFrontDefaultCertificate": true,
    "MinimumProtocolVersion": "TLSv1",
    "CertificateSource": "cloudfront"
},
"Restrictions": {
    "GeoRestriction": {
        "RestrictionType": "none",
        "Quantity": 0
    }
},
"WebACLId": "",
"HttpVersion": "http2",
```

```
        "IsIPV6Enabled": true
    }
}
}
```

- APIの詳細については、「コマンドリファレンス[UpdateDistribution](#)」の「」を参照してください。AWS CLI

Java

SDK for Java 2.x

Note

については、さらに詳しく説明します [GitHub](#)。用例一覧を検索し、[AWS コードサンプルリポジトリ](#)での設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionRequest;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.DistributionConfig;
import
    software.amazon.awssdk.services.cloudfront.model.UpdateDistributionRequest;
import software.amazon.awssdk.services.cloudfront.model.CloudFrontException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class ModifyDistribution {
    public static void main(String[] args) {
        final String usage = ""
```

```
Usage:
    <id>\s

Where:
    id - the id value of the distribution.\s
""";

if (args.length != 1) {
    System.out.println(usage);
    System.exit(1);
}

String id = args[0];
CloudFrontClient cloudFrontClient = CloudFrontClient.builder()
    .region(Region.AWS_GLOBAL)
    .build();

modDistribution(cloudFrontClient, id);
cloudFrontClient.close();
}

public static void modDistribution(CloudFrontClient cloudFrontClient, String
idVal) {
    try {
        // Get the Distribution to modify.
        GetDistributionRequest disRequest = GetDistributionRequest.builder()
            .id(idVal)
            .build();

        GetDistributionResponse response =
cloudFrontClient.getDistribution(disRequest);
        Distribution disObject = response.distribution();
        DistributionConfig config = disObject.distributionConfig();

        // Create a new DistributionConfig object and add new values to
comment and
        // aliases
        DistributionConfig config1 = DistributionConfig.builder()
            .aliases(config.aliases()) // You can pass in new values here
            .comment("New Comment")
            .cacheBehaviors(config.cacheBehaviors())
            .priceClass(config.priceClass())
            .defaultCacheBehavior(config.defaultCacheBehavior())
            .enabled(config.enabled())
```

```
.callerReference(config.callerReference())
.logging(config.logging())
.originGroups(config.originGroups())
.origins(config.origins())
.restrictions(config.restrictions())
.defaultRootObject(config.defaultRootObject())
.webACLId(config.webACLId())
.httpVersion(config.httpVersion())
.viewerCertificate(config.viewerCertificate())
.customErrorResponses(config.customErrorResponses())
.build();

UpdateDistributionRequest updateDistributionRequest =
UpdateDistributionRequest.builder()
    .distributionConfig(config1)
    .id(disObject.id())
    .ifMatch(response.eTag())
    .build();

cloudFrontClient.updateDistribution(updateDistributionRequest);

} catch (CloudFrontException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- API の詳細については、「API リファレンス [UpdateDistribution](#)」の「」を参照してください。AWS SDK for Java 2.x

Python

SDK for Python (Boto3)

Note

については、さらに詳しく説明します [GitHub](#)。用例一覧を検索し、[AWS コードサンプルリポジトリ](#)での設定と実行の方法を確認してください。

```
class CloudFrontWrapper:
    """Encapsulates Amazon CloudFront operations."""

    def __init__(self, cloudfront_client):
        """
        :param cloudfront_client: A Boto3 CloudFront client
        """
        self.cloudfront_client = cloudfront_client

    def update_distribution(self):
        distribution_id = input(
            "This script updates the comment for a CloudFront distribution.\n"
            "Enter a CloudFront distribution ID: "
        )

        distribution_config_response =
self.cloudfront_client.get_distribution_config(
            Id=distribution_id
        )
        distribution_config = distribution_config_response["DistributionConfig"]
        distribution_etag = distribution_config_response["ETag"]

        distribution_config["Comment"] = input(
            f"\nThe current comment for distribution {distribution_id} is "
            f"'{distribution_config['Comment']}'.\n"
            f"Enter a new comment: "
        )
        self.cloudfront_client.update_distribution(
            DistributionConfig=distribution_config,
            Id=distribution_id,
            IfMatch=distribution_etag,
        )
        print("Done!")
```

- API の詳細については、[UpdateDistribution](#) AWS SDK for Python (Boto3) API リファレンスの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、「[AWS SDK CloudFront で使用する](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK CloudFront を使用してパブリックキーを にアップロードする

以下は、パブリックキーをアップロードする方法を説明するコードの例です。

CLI

AWS CLI

CloudFront パブリックキーを作成するには

次の例では、 という名前の JSON ファイルにパラメータを指定して CloudFront パブリックキーを作成します pub-key-config.json。このコマンドを使用するには、事前に PEM でエンコードされたパブリックキーが必要です。詳細については、「Amazon CloudFront [デベロッパーガイド](#)」の「[RSA キーペアを作成する](#)」を参照してください。

```
aws cloudfront create-public-key \  
  --public-key-config file://pub-key-config.json
```

pub-key-config.json ファイルは、以下を含む現在のフォルダ内にある JSON ドキュメントです。パブリックキーは PEM 形式でエンコードされていることに注意してください。

```
{  
  "CallerReference": "cli-example",  
  "Name": "ExampleKey",  
  "EncodedKey": "-----BEGIN PUBLIC KEY-----  
\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAxPMbCA2Ks01nd7IR+3pw  
\nwd3H/7jPGwj8bLUmore7bX+oeGpZ6QmLAe/1U0WcmZX2u70dYcSIzB1ofZtcn4cJ  
\nenHBAz03ohBY/L1tQGJfS2A+omnN6H16VZE1JCK8XSJyfze7MDLcUyHZETdxuvRb  
\nA9X343/vMAuQPhinFJ8Wdy8YBXSPpy7r95y1UQd9LFYTBzVZYG2tSesplc0kjM3\n2Uu  
+oMwxQAw1NINnSLPinMVsutJy6Zq1V3McWNWe4T+STGtWhrPNqJEn45sIcCx4\nnq  
+kGZ2NQ0FyIyT2eiLK0X5Rrgb/a36E/aMk4VoDsaenBQgG7WLTnstb9sr7MIhS6A\nnrwIDAQAB\n-----  
END PUBLIC KEY-----\n",  
  "Comment": "example public key"  
}
```

出力:

```
{
```

```

    "Location": "https://cloudfront.amazonaws.com/2019-03-26/public-key/
KDFB19YGCR002",
    "ETag": "E2QWRUHEXAMPLE",
    "PublicKey": {
      "Id": "KDFB19YGCR002",
      "CreatedTime": "2019-12-05T18:51:43.781Z",
      "PublicKeyConfig": {
        "CallerReference": "cli-example",
        "Name": "ExampleKey",
        "EncodedKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAxPMbCA2Ks01nd7IR+3pw
\nwd3H/7jPGwj8bLUmore7bX+oeGpZ6QmLAe/1U0WcmZX2u70dYcSIzB1ofZtcn4cJ
\nenHBAz03ohBY/L1tQGJfS2A+omnN6H16VZE1JCK8XSJyfze7MDLcUyHZETdxuvRb
\nA9X343/vMAuQPnhinFJ8Wdy8YBXSPpy7r95y1UQd9LfYTBzVZYG2tSesplc0kjM3\n2Uu
+oMwxQAw1NINnSLPinMVsutJy6Zq1V3McWNwe4T+STGtWhrPNqJEn45sIcCx4\nq
+kGZ2NQ0FyIyT2eiLK0X5RgB/a36E/aMk4VoDsaenBQgG7WLTnstb9sr7MIhS6A\nnrwIDAQAB\n-----
END PUBLIC KEY-----\n",
        "Comment": "example public key"
      }
    }
  }
}

```

- APIの詳細については、「コマンドリファレンス[CreatePublicKey](#)」の「」を参照してください。AWS CLI

Java

SDK for Java 2.x

Note

については、さらに詳しく説明します [GitHub](#)。用例一覧を検索し、[AWS コードサンプルリポジトリ](#)での設定と実行の方法を確認してください。

次のコード例では、パブリックキーを読み取り、Amazon にアップロードします CloudFront。

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;

```

```
import software.amazon.awssdk.services.cloudfront.model.CreatePublicKeyResponse;
import software.amazon.awssdk.utils.IoUtils;

import java.io.IOException;
import java.io.InputStream;
import java.util.UUID;

public class CreatePublicKey {
    private static final Logger logger =
        LoggerFactory.getLogger(CreatePublicKey.class);

    public static String createPublicKey(CloudFrontClient cloudFrontClient,
        String publicKeyFileName) {
        try (InputStream is =
            CreatePublicKey.class.getClassLoader().getResourceAsStream(publicKeyFileName)) {
            String publicKeyString = IoUtils.toUtf8String(is);
            CreatePublicKeyResponse createPublicKeyResponse = cloudFrontClient
                .createPublicKey(b -> b.publicKeyConfig(c -> c
                    .name("JavaCreatedPublicKey" + UUID.randomUUID())
                    .encodedKey(publicKeyString)
                    .callerReference(UUID.randomUUID().toString())));
            String createdPublicKeyId = createPublicKeyResponse.publicKey().id();
            logger.info("Public key created with id: [{}]", createdPublicKeyId);
            return createdPublicKeyId;

        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```

- APIの詳細については、「API リファレンス [CreatePublicKey](#)」の「」を参照してください。AWS SDK for Java 2.x

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、「[AWS SDK CloudFront で使用する](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDKs CloudFront を使用するシナリオ

以下のコード例は、CloudFront で AWS SDK を使用した一般的なシナリオを実装する方法を示しています。これらのシナリオは、内で複数の関数を呼び出して特定のタスクを実行する方法を示しています CloudFront。各シナリオには [へのリンクがあり GitHub](#)、コードの設定と実行の手順が記載されています。

例

- [AWS SDK を使用して署名付き URL と Cookie を作成する](#)

AWS SDK を使用して署名付き URL と Cookie を作成する

次のコード例は、制限付きリソースへのアクセスを許可する署名付き URL と Cookie を作成する方法を示しています。

Java

SDK for Java 2.x

Note

については、さらに詳しく説明します GitHub。用例一覧を検索し、[AWS コードサンプルリポジトリ](#)での設定と実行の方法を確認してください。

[CannedSignerRequest](#) クラスを使用して、既定ポリシーで URLs または Cookie に署名します。

```
import software.amazon.awssdk.services.cloudfront.model.CannedSignerRequest;

import java.net.URL;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.temporal.ChronoUnit;

public class CreateCannedPolicyRequest {

    public static CannedSignerRequest createRequestForCannedPolicy(String
distributionDomainName,
```

```
        String fileNameToUpload,
        String privateKeyFullPath, String publicKeyId) throws Exception {
    String protocol = "https";
    String resourcePath = "/" + fileNameToUpload;

    String cloudFrontUrl = new URL(protocol, distributionDomainName,
resourcePath).toString();
    Instant expirationDate = Instant.now().plus(7, ChronoUnit.DAYS);
    Path path = Paths.get(privateKeyFullPath);

    return CannedSignerRequest.builder()
        .resourceUrl(cloudFrontUrl)
        .privateKey(path)
        .keyPairId(publicKeyId)
        .expirationDate(expirationDate)
        .build();
    }
}
```

[CustomSignerRequest](#) クラスを使用して、カスタムポリシーで URLs または Cookie に署名します。 `activeDate` および `ipRange` はオプションのメソッドです。

```
import software.amazon.awssdk.services.cloudfront.model.CustomSignerRequest;

import java.net.URL;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.temporal.ChronoUnit;

public class CreateCustomPolicyRequest {

    public static CustomSignerRequest createRequestForCustomPolicy(String
distributionDomainName,
        String fileNameToUpload,
        String privateKeyFullPath, String publicKeyId) throws Exception {
    String protocol = "https";
    String resourcePath = "/" + fileNameToUpload;

    String cloudFrontUrl = new URL(protocol, distributionDomainName,
resourcePath).toString();
    Instant expireDate = Instant.now().plus(7, ChronoUnit.DAYS);
```

```
// URL will be accessible tomorrow using the signed URL.
Instant activeDate = Instant.now().plus(1, ChronoUnit.DAYS);
Path path = Paths.get(privateKeyFullPath);

return CustomSignerRequest.builder()
    .resourceUrl(cloudFrontUrl)
    .privateKey(path)
    .keyPairId(publicKeyId)
    .expirationDate(expireDate)
    .activeDate(activeDate) // Optional.
    // .ipRange("192.168.0.1/24") // Optional.
    .build();
}
}
```

次の例は、クラスを使用して署名付き Cookie と URLs [CloudFrontUtilities](#) を生成する方法を示しています。でこのコード例 [を表示します](#) GitHub。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontUtilities;
import software.amazon.awssdk.services.cloudfront.cookie.CookiesForCannedPolicy;
import software.amazon.awssdk.services.cloudfront.cookie.CookiesForCustomPolicy;
import software.amazon.awssdk.services.cloudfront.model.CannedSignerRequest;
import software.amazon.awssdk.services.cloudfront.model.CustomSignerRequest;
import software.amazon.awssdk.services.cloudfront.url.SignedUrl;

public class SigningUtilities {
    private static final Logger logger =
        LoggerFactory.getLogger(SigningUtilities.class);
    private static final CloudFrontUtilities cloudFrontUtilities =
        CloudFrontUtilities.create();

    public static SignedUrl signUrlForCannedPolicy(CannedSignerRequest
        cannedSignerRequest) {
        SignedUrl signedUrl =
            cloudFrontUtilities.getSignedUrlWithCannedPolicy(cannedSignerRequest);
        logger.info("Signed URL: [{}]", signedUrl.url());
        return signedUrl;
    }
}
```

```
public static SignedUrl signUrlForCustomPolicy(CustomSignerRequest
customSignerRequest) {
    SignedUrl signedUrl =
cloudFrontUtilities.getSignedUrlWithCustomPolicy(customSignerRequest);
    logger.info("Signed URL: [{}]", signedUrl.url());
    return signedUrl;
}

public static CookiesForCannedPolicy
getCookiesForCannedPolicy(CannedSignerRequest cannedSignerRequest) {
    CookiesForCannedPolicy cookiesForCannedPolicy = cloudFrontUtilities
        .getCookiesForCannedPolicy(cannedSignerRequest);
    logger.info("Cookie EXPIRES header [{}]",
cookiesForCannedPolicy.expiresHeaderValue());
    logger.info("Cookie KEYPAIR header [{}]",
cookiesForCannedPolicy.keyPairIdHeaderValue());
    logger.info("Cookie SIGNATURE header [{}]",
cookiesForCannedPolicy.signatureHeaderValue());
    return cookiesForCannedPolicy;
}

public static CookiesForCustomPolicy
getCookiesForCustomPolicy(CustomSignerRequest customSignerRequest) {
    CookiesForCustomPolicy cookiesForCustomPolicy = cloudFrontUtilities
        .getCookiesForCustomPolicy(customSignerRequest);
    logger.info("Cookie POLICY header [{}]",
cookiesForCustomPolicy.policyHeaderValue());
    logger.info("Cookie KEYPAIR header [{}]",
cookiesForCustomPolicy.keyPairIdHeaderValue());
    logger.info("Cookie SIGNATURE header [{}]",
cookiesForCustomPolicy.signatureHeaderValue());
    return cookiesForCustomPolicy;
}
}
```

- APIの詳細については、「API リファレンス [CloudFrontUtilities](#)」の「」を参照してください。AWS SDK for Java 2.x

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、「[AWS SDK CloudFront で使用する](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

Amazon CloudFront 関連情報

ここに記載されている情報とリソースは、 の詳細を理解するのに役立ちます CloudFront。

トピック

- [Amazon CloudFront のその他のドキュメント](#)
- [サポート情報](#)
- [CloudFront デベロッパーツールと SDKs](#)
- [Amazon Web Services ブログからのヒント](#)

Amazon CloudFront のその他のドキュメント

このサービスを利用する際に役立つ関連リソースは次のとおりです。

- [Amazon CloudFront API リファレンス](#) – API アクション、パラメータ、データ型の完全な説明と、サービスが返すエラーのリストを示します。
- [CloudFront 最新情報](#) — CloudFront 新機能と最近追加されたエッジロケーションのお知らせ。
- [Amazon CloudFront 製品情報](#) – 機能や料金情報など CloudFront、に関する情報のメインウェブページです。

サポート情報

のサポート CloudFront は、さまざまな形式で利用できます。

- [AWS re:Post](#) – デベロッパーが に関連する技術的な質問について議論するためのコミュニティベースの質問および回答サイト CloudFront。
- [AWS サポートセンター](#) – このサイトでは、お客様の最近のサポートケース、AWS Trusted Advisor の助言とヘルスチェックの結果に関する情報がひとつにまとめられていて、フォーラム、技術上のよくある質問、サービスヘルスダッシュボード、および AWS サポートプランに関する情報へのリンクも掲載されています。
- [AWS プレミアムサポート](#) – プレミアムサポートに関する情報のメインウェブページです。AWS プレミアムサポートは one-on-one、でのアプリケーションの構築と実行に役立つ、の迅速な対応を行うサポートチャンネルですAWS。
- [AWS IQ](#) – AWS 認定プロフェッショナルとエキスパートからのヘルプを得ます。

- [お問い合わせ](#) – 請求やアカウントに関するお問い合わせ用のリンクです。技術的な質問の場合は、上記の AWS re:Post または AWS プレミアムサポートをご利用ください。

CloudFront デベロッパーツールと SDKs

ドキュメント、コード例、リリースノートなど、AWS を利用して革新的なアプリケーションを構築するのに役立つ情報を含むデベロッパーツールリソースへのリンクについては、[ツール](#)のページを参照してください。

さらに、Amazon Web Services は、プログラムでアクセス CloudFrontするためのソフトウェア開発キットを提供しています。SDK ライブラリは、サービスリクエストに対する署名の暗号化、リクエストの再試行、エラーレスポンスの処理など、多数の一般的なタスクを自動化します。

Amazon Web Services ブログからのヒント

AWS ブログには、[ネットワークとコンテンツ配信](#)カテゴリで CloudFrontを使用するのに役立つ投稿が多数あります。

ドキュメント履歴

次の表は、CloudFront ドキュメントに加えられた重要な変更点をまとめたものです。更新の通知を受け取る場合は、[RSS フィードにサブスクライブ](#)できます。

変更	説明	日付
AWS SDKs CloudFront を使用するためのコード例	AWS Software Development Kit (SDK) CloudFront でを使用する方法を示すコード例を追加しました。例は、個々のサービス関数を呼び出す方法を示すコード抜粋と、同じサービス内で複数の関数を呼び出して特定のタスクを達成する方法を示す例に分けられています。	2024 年 2 月 16 日
AWS マネージドポリシーの更新	CloudFrontReadOnlyAccess と CloudFrontFullAccess IAM ポリシーが KeyValueStore オペレーションをサポートするようになりました。	2023 年 12 月 19 日
JavaScript ランタイム 2.0	Functions の JavaScript ランタイム 2.0 CloudFront 機能を追加しました。	2023 年 11 月 21 日
CloudFront KeyValueStore	Amazon が をサポートする CloudFront ようになりました CloudFront KeyValueStore。この機能は、CloudFront 関数内からの読み取りアクセスを可能にする、安全でグローバルな低レイテンシーのキーバリューストアです。こ	2023 年 11 月 21 日

	れにより、CloudFront エッジロケーションで高度なカスタマイズ可能なロジックが可能になります。	
Lambda@Edge がより新しいランタイムバージョンをサポート	Lambda@Edge は、Lambda 関数を Node.js 20 ランタイムでサポートするようになりました。	2023 年 11 月 15 日
セキュリティダッシュボード	CloudFront デイストリビューションを作成すると、によってセキュリティダッシュボードが作成されます。AWS WAF を有効にして、地理的制限を管理し、リクエスト、ボット、ログの概要データを表示します。	2023 年 11 月 8 日
関数内のクエリ文字列のソート	CloudFront で Functions を使用した CloudFront クエリ文字列のソートがサポートされるようになりました	2023 年 10 月 3 日
AWS WAF のセキュリティに関する推奨事項	Amazon では、CloudFront コンソールに AWS WAF セキュリティに関する推奨事項が表示される CloudFront ようになりました。	2023 年 9 月 26 日
古い (期限切れの) キャッシュコンテンツ提供のサポート	CloudFront は、Stale-While-Revalidate および Stale-If-Error キャッシュ制御ディレクティブをサポートします。	2023 年 5 月 15 日

ワンクリックで AWS WAF 保護を有効にする	ディストリビューションに AWS WAF CloudFront セキュリティ保護を追加する効率的な方法。	2023 年 5 月 10 日
標準ログに使用される新しい S3 バケットの ACL を有効にする	新しい S3 バケットのデフォルトの ACL 設定に対処するためのメモとリンクを追加しました。	2023 年 4 月 11 日
Amazon S3 Object Lambda を使用してオリジンを作成する	Amazon S3 Object Lambda アクセスポイントエイリアスをディストリビューションのオリジンとして使用できます。	2023 年 3 月 31 日
CloudFront Functions を使用して HTTP ステータスと本文をカスタマイズする	CloudFront Functions を使用して、ビューワーレスポンスステータスコードを更新し、レスポンス本文を置換または削除できます。	2023 年 3 月 29 日
ポートの CORS ヘッダーのワイルドカードオプションを追加	CORS アクセスコントロールヘッダーで、ポートのワイルドカード設定を含めることができるようになりました。	2023 年 3 月 20 日
AWS Security Hub ユーザーガイドの新しいリンクを追加	言語を更新し、AWS Security Hub 「ユーザーガイド」に再編成された Amazon CloudFront コントロールへのリンクを追加しました。	2023 年 3 月 9 日

[CloudFront がオリジンリクエストポリシーでブロックリスト \(「すべて」を除く\) をサポートするようになりました](#)

オリジンリクエストポリシーでブロックリストを使用して、 がオリジン CloudFront に送信するリクエストに、指定されたものを除くすべてのクエリ文字列、HTTP ヘッダー、または Cookie を含めません。

2023 年 2 月 22 日

[CloudFront は、ホストヘッダーを除くすべてのビューワーヘッダーを転送する新しいマネージドオリジンリクエストポリシーを追加します。](#)

CloudFrontの新しいマネージドオリジンリクエストポリシーを使用して、 がオリジン CloudFront に送信するリクエストに、ヘッダーを除くビューワーリクエストのすべてのHostヘッダーを含めません。

2023 年 2 月 22 日

[Lambda@Edge に関する制限の更新](#)

Lambda@Edge は、[Auto] (自動) に設定された Lambda ランタイム管理設定をサポートします。

2023 年 2 月 16 日

[の IAM ガイダンスを更新しました CloudFront](#)

IAM ベストプラクティスに沿ってガイドを更新しました。詳細については、「[IAM のセキュリティのベストプラクティス](#)」を参照してください。

2023 年 2 月 15 日

[オリジンアクセスコントロールによるセキュリティ強化](#)

指定された CloudFront ディストリビューションのみへのアクセスを許可することで、MediaStore オリジンを保護できるようになりました。

2023 年 2 月 9 日

[ビューワのヘッダー構造を決定するための新しいヘッダー](#)

ビューワを特定しやすくするために、ビューワが送信するヘッダーに基づいてヘッダーの順序とヘッダー数を追加できるようになりました。

2023 年 1 月 13 日

[Lambda@Edge がより新しいランタイムバージョンをサポート](#)

Lambda@Edge は、Lambda 関数を Node.js 18 ランタイムでサポートするようになりました。

2023 年 1 月 12 日

[レスポンスヘッダーポリシーを使用してレスポンスヘッダーを削除する](#)

CloudFront レスポンスヘッダーポリシーを使用して、オリジンからレスポンスで受信したヘッダー CloudFront を削除できるようになりました。指定されたヘッダーは、がビューワ CloudFront に送信するレスポンスには含まれません。

2023 年 1 月 3 日

[新しいマネージドのオリジンリクエストポリシー](#)

AllViewerAndCloudFrontHeaders-2022-06 オリジンアクセスポリシーを追加しました。

2022 年 12 月 2 日

[設定の変更を安全にテストするための継続的デプロイ](#)

本番トラフィックのサブセットでテストすることで、CDN 設定に変更をデプロイできるようになりました。

2022 年 11 月 18 日

[CloudFront-Viewer-JA3-Fingerprint ヘッダーのリリース](#)

JA3 フィンガープリントを使用して、リクエストが既知のクライアントからのものかどうかを判断できるようになりました。

2022 年 11 月 16 日

CORS ヘッダーのワイルドカードオプションを追加	一部の CORS アクセスコントロールヘッダーで、さまざまなワイルドカード設定を使用できるようになりました。	2022 年 11 月 11 日
CloudFront デистриビューションの追加メトリクス	CloudFront API および MonitoringSubscription での のサポートAWS CloudFormation。	2022 年 10 月 3 日
オリジンアクセスコントローलによるセキュリティ強化	指定された CloudFront デистриビューションのみへのアクセスを許可することで、Amazon S3 オリジンを保護できるようになりました。	2022 年 8 月 24 日
CloudFront デистриビューションの HTTP/3 サポート	CloudFront デистриビューションに HTTP/3 を選択できるようになりました。	2022 年 8 月 15 日
ハンドシェイクの詳細を CloudFront-Viewer-TLS ヘッダーに追加する	使用している SSL/TLS ハンドシェイクに関する情報を表示できるようになりました。	2022 年 6 月 27 日
Server-Timing ヘッダーの新しいメトリクス	Server-Timing ヘッダーに新しい cdn-downstream-fb1 メトリクスを追加しました。	2022 年 6 月 13 日
TLS のバージョンと暗号に関する情報を取得するための新しいヘッダー	CloudFront-Viewer-TLS ヘッダーを使用して、TLS (または SSL) のバージョンと、ビューワーと間の接続に使用された暗号に関する情報を取得できるようになりました CloudFront。	2022 年 5 月 23 日

CloudFront Functions の新しい FunctionThrottles メトリクス	Amazon では CloudWatch、特定の期間に CloudFront 関数がスロットリングされた回数をモニタリングできるようになりました。	2022 年 5 月 4 日
CloudFront で Lambda 関数 URLs をサポート	関数 URLs を持つ Lambda 関数を使用してサーバーレスウェブアプリケーションを構築する場合、 を追加して CloudFront さまざまな利点を享受できるようになりました。	2022 年 4 月 6 日
HTTP レスポンスの Server-Timing ヘッダー	から送信された HTTP レスポンスの Server-Timing ヘッダーを有効に CloudFront して、 の動作とパフォーマンスに関するインサイトを得るのに役立つメトリクスを表示できるようになりました CloudFront。	2022 年 3 月 30 日
AWS マネージドプレフィックスのリストを使用してインバウンドトラフィックを制限	オリジンへのインバウンド HTTP および HTTPS トラフィックを、 CloudFront のオリジン向けサーバーに属する IP アドレスのみから制限できるようになりました。	2022 年 2 月 7 日

以前のエントリーについては、「[2022 より前の更新](#)」を参照してください。

2022 より前の更新

次の表は、2022 年より前に CloudFront ドキュメントに加えられた重要な変更点をまとめたものです。

変更	説明	日付
新機能	CloudFront は、レスポンスヘッダーポリシーのサポートを追加します。これにより、ビューワー (ウェブブラウザまたは他のクライアント) に送信する HTTP レスポンスに CloudFront 追加する HTTP ヘッダーを指定できます。オリジンを変更、またはコードを書き込むことなく、希望するヘッダー (およびその値) を指定できます。詳細については、「 the section called “レスポンスヘッダーの追加または削除” 」を参照してください。	2021 年 11 月 2 日
新しい CloudFront-Viewer-Address リクエストヘッダー	CloudFront は、HTTP リクエストを送信したビューワーの IP アドレス CloudFront-Viewer-Address を含む新しいヘッダーのサポートを追加します CloudFront。詳細については、「 the section called “CloudFront リクエストヘッダーの追加” 」を参照してください。	2021 年 10 月 25 日
Lambda@Edge が、新しいランタイムバージョンをサポートするようになりました	Lambda@Edge は、Lambda 関数を Python 3.9 ランタイムでサポートするようになりました。詳細については、「 the section called “ランタイムのサポート” 」を参照してください。	2021 年 9 月 22 日
AWS マネージドポリシーの更新	CloudFront が CloudFrontReadOnlyAccess ポリシーを更新しました。詳細については、「 the section called “ポリシーの更新” 」を参照してください。	2021 年 9 月 8 日
新機能	CloudFront は、ビューワー向け HTTPS 接続の ECDSA 証明書をサポートするようになりました。詳細については、「 the section called “ビューワーとの間でサポートされているプロトコルと暗号 CloudFront” 」および「 the section called “で SSL/TLS 証明書を使用するための要件 CloudFront” 」を参照してください。	2021 年 7 月 14 日
新機能	CloudFront は、に連絡せずに、1 つのディストリビューションから別のディストリビューションに代替ドメイン名を移動	2021 年 7 月 7 日

変更	説明	日付
	<p>するより多くの方法をサポートするようになりましたAWS Support。詳細については、「the section called “代替ドメイン名を別のディストリビューションに移動する”」を参照してください。</p>	
新しいセキュリティポリシー	<p>CloudFront は、サポートされる暗号のより小さいセットで、新しいセキュリティポリシー TLSv1.2_2021 をサポートするようになりました。詳細については、「ビューワーとの間でサポートされているプロトコルと暗号 CloudFront」を参照してください。</p>	2021 年 6 月 23 日
新機能	<p>Amazon CloudFront では、関数がサポートされる CloudFront ようになりました。CloudFront これは、レイテンシーの影響を受けやすい CDN カスタマイズを大規模に JavaScript 行うために、で軽量の関数を記述できるのネイティブ機能です。詳細については、「CloudFront Functions によるエッジでのカスタマイズ」を参照してください。</p>	2021 年 5 月 3 日
Lambda@Edge は新しいランタイムバージョンをサポートするようになりました	<p>Lambda@Edge は、Lambda関数を Node.js 14 ランタイムでサポートするようになりました。詳細については、「ランタイムのサポート」を参照してください。</p>	2021 年 4 月 29 日
RTMP ディストリビューションのドキュメントを削除	<p>Amazon は、2020 年 12 月 31 日にリアルタイムメッセージングプロトコル (RTMP) ディストリビューションを CloudFront 廃止しました。 RTMP ディストリビューションのドキュメントが Amazon CloudFront デベロッパーガイド から削除されました。</p>	2021 年 2 月 10 日
新しい料金オプション	<p>Amazon では、CloudFront Security Savings Bundle CloudFront が導入されています。これは、AWS請求書の CloudFront 料金を最大 30% 節約できる簡単な方法です。詳細については、「CloudFront Security Savings Bundle」を参照してください。</p>	2021 年 2 月 5 日

変更	説明	日付
チュートリアル の新規追加	Amazon CloudFront デベロッパーガイドに、Amazon を使用して Elastic Load Balancing の Application Load Balancer へのアクセス CloudFront を制限するためのチュートリアルが追加されました。詳細については、「 Application Load Balancers へのアクセスを制限する 」を参照してください。	2020 年 12 月 18 日
パブリック キーの管理 の新しいオプ ション	CloudFront は、AWS アカウントのルートユーザーにアクセスすることなく、CloudFront コンソールと API を通じて署名 URLs と署名付き Cookie のパブリックキー管理をサポートするようになりました。詳細については、「 署名付き URL と署名付き Cookie を作成できる署名者の指定 」を参照してください。	2020 年 10 月 22 日
新機能 – Origin Shield	CloudFront は、オリジンの負荷を最小限に抑え、可用性を向上させ、運用コストを削減するのに役立つ CloudFront キャッシュインフラストラクチャ内の追加のレイヤーである CloudFront Origin Shield をサポートするようになりました。詳細については、「 Amazon CloudFront Origin Shield の使用 」を参照してください。	2020 年 10 月 20 日
新しい圧縮形 式	CloudFront CloudFront エッジロケーションでオブジェクトを圧縮 CloudFront するようにを設定すると、が Brotli 圧縮形式をサポートするようになりました。正規化された Accept-Encoding ヘッダーを使用して Brotli オブジェクトをキャッシュ CloudFront するようにを設定することもできます。詳細については、「 圧縮ファイルの供給 」および「 圧縮のサポート 」を参照してください。	2020 年 9 月 14 日
新しい TLS プロトコル	CloudFront は、ビューワーと CloudFront デイストリビューション間の HTTPS 接続の TLS 1.3 プロトコルをサポートするようになりました。TLS 1.3 は、すべての CloudFront セキュリティポリシーでデフォルトで有効になっています。詳細については、「 ビューワーとの間でサポートされているプロトコルと暗号 CloudFront 」を参照してください。	2020 年 9 月 3 日

変更	説明	日付
新しいリアルタイムログ	CloudFront は、設定可能なリアルタイムログをサポートするようになりました。リアルタイムログを使用して、ディストリビューションに対して行われたリクエストに関する情報をリアルタイムで取得できます。リアルタイムログを使用して、コンテンツ配信のパフォーマンスに基づいて監視、分析、アクションを実行できます。詳細については、「 リアルタイムログ 」を参照してください。	2020 年 8 月 31 日
追加のメトリクスの API サポート	CloudFront で、CloudFront API を使用した 8 つの追加のリアルタイムメトリクスの有効化がサポートされるようになりました。詳細については、「 追加メトリクスの有効化 」を参照してください。	2020 年 8 月 28 日
新しい CloudFront HTTP ヘッダー	CloudFront は、デバイスタイプ、地理的位置など、ビューワーに関する情報を決定するための追加の HTTP ヘッダーを追加しました。詳細については、「 the section called “ CloudFront リクエストヘッダーの追加” 」を参照してください。	2020 年 7 月 23 日
新機能	CloudFront でキャッシュポリシーとオリジンリクエストポリシーがサポートされるようになりました。これにより、CloudFront ディストリビューションのキャッシュキーとオリジンリクエストをより細かく制御できます。詳細については、「 ポリシーの使用 」を参照してください。	2020 年 7 月 22 日
新しいセキュリティポリシー	CloudFront は、サポートされる暗号のより小さなセットで、新しいセキュリティポリシー TLSv1.2_2019 をサポートするようになりました。詳細については、「 ビューワーとの間でサポートされているプロトコルと暗号 CloudFront 」を参照してください。	2020 年 7 月 8 日
オリジンのタイムアウトと試行を制御する新しい設定	CloudFront は、オリジンのタイムアウトと試行を制御する新しい設定を追加しました。詳細については、「 オリジンのタイムアウトと試行の制御 」を参照してください。	2020 年 6 月 5 日

変更	説明	日付
安全な静的ウェブサイト CloudFront を作成しての使用を開始するための新しいドキュメント	の使用を開始するには、Amazon S3、CloudFront、Lambda@Edge などを使用して安全な静的ウェブサイト CloudFront を作成し、すべてでデプロイしますAWS CloudFormation。詳細については、「 安全な静的ウェブサイトの使用開始 」を参照してください。	2020 年 6 月 2 日
Lambda@Edge は新しいランタイムバージョンをサポートするようになりました	Lambda@Edge は、Lambda関数を Node.js 12 および Python 3.8 ランタイムでサポートするようになりました。詳細については、「 ランタイムのサポート 」を参照してください。	2020 年 2 月 27 日
の新しいリアルタイムメトリクス CloudWatch	Amazon では、Amazon で 8 つのリアルタイムメトリクスが追加され CloudFront ました CloudWatch。詳細については、「 追加の CloudFrontディストリビューションメトリクスを有効にする 」を参照してください。	2019 年 12 月 19 日
アクセスログの新しいフィールド	CloudFront は、ログにアクセスするための 7 つの新しいフィールドを追加します。詳細については、「 標準ログファイルフィールド 」を参照してください。	2019 年 12 月 12 日
AWS WordPress プラグイン	AWS WordPress プラグインを使用して、ウェブサイトへの訪問者に を使用した高速表示エクスペリエンスを提供できます WordPress CloudFront。(更新: 2022 年 9 月 30 日現在、for AWS WordPress プラグインは非推奨です。)	2019 年 10 月 30 日
タグベースとリソースレベルの IAM アクセス許可ポリシー	CloudFront では、IAM アクセス許可ポリシーを指定する 2 つの方法として、タグベースとリソースレベルのポリシーアクセス許可がサポートされるようになりました。詳細については、「 リソースへのアクセスの管理 」を参照してください。	2019 年 8 月 8 日

変更	説明	日付
Python プログラミング言語のサポート	Lambda@Edge で、Node.js に加えて Python プログラミング言語でも関数を開発できるようになりました。さまざまなシナリオに対応する関数の例については、「 Lambda@Edge 関数の例 」を参照してください。	2019 年 8 月 1 日
更新されたモニタリンググラフ	デイス CloudFront トリビューションに関連付けられた Lambda 関数を CloudFront コンソールから直接モニタリングし、エラーをより簡単に追跡およびデバッグするための新しい方法を説明するコンテンツの更新。詳細については、「 のモニタリング CloudFront 」を参照してください。	2013 年 6 月 20 日
一括セキュリティコンテンツ	新しい「セキュリティ CloudFront」の章では、データ保護、IAM、ログ記録、コンプライアンスなどに関するの機能および実装に関する情報を統合しています。詳細については、「 セキュリティ 」を参照してください。	2019 年 5 月 24 日
ドメイン検証が必要になりました	CloudFront では、デイストリビューションで代替ドメイン名を使用するアクセス許可があることを確認するために、SSL 証明書を使用する必要があるようになりました。詳細については、「 代替ドメイン名と HTTPS の使用 」を参照してください。	2019 年 4 月 9 日
PDF ファイル名の更新	Amazon CloudFront デベロッパーガイドの新しいファイル名は AmazonCloudFront_ ですDevGuide。以前の名称は cf-dg でした。	2019 年 1 月 7 日
新機能	CloudFront は をサポートするようになりました。TCP ベースのプロトコルは WebSocket、クライアントとサーバー間の存続期間の長い接続が必要な場合に便利です。高可用性が必要なシナリオでは、オリジンフェイルオーバーを使用して CloudFront を設定することもできるようになりました。詳細については、 CloudFront 「デイストリビューション WebSocket での使用」 および CloudFront 「オリジンフェイルオーバーによる高可用性の最適化」 を参照してください。	2018 年 11 月 20 日

変更	説明	日付
新機能	CloudFront は、Lambda 関数を実行する HTTP リクエストの詳細なエラーログ記録をサポートするようになりました。ログを保存 CloudWatch し、関数が無効なレスポンスを返すときに HTTP 5xx エラーのトラブルシューティングに役立てることができます。詳細については、 CloudWatch 「Lambda 関数のメトリクスと CloudWatch ログ」 を参照してください。	2018 年 10 月 8 日
新機能	書き込み可能な HTTP メソッド (POST、PUT、DELETE など) のリクエストのボディを Lambda@Edge で公開することを選択できるようになり、Lambda 関数でそのボディにアクセスできます。読み取り専用アクセスも選択でき、ボディを置き換えることも指定できます。詳細については、「 Include Body オプションの選択によるリクエストボディへのアクセス 」を参照してください。	2018 年 8 月 14 日
新機能	CloudFront は、gzip に加えて、または gzip の代わりに、brotli またはその他の圧縮アルゴリズムを使用して圧縮されたコンテンツの提供をサポートするようになりました。詳細については、「 圧縮ファイルの供給 」を参照してください。	2018 年 7 月 25 日
再編成	「Amazon CloudFront デベロッパーガイド」は、関連コンテンツの検索を簡素化し、スキャン可能性とナビゲーションを向上させるために再編成されました。	2018 年 6 月 28 日
新機能	オリジン側イベント内で、追加のヘッダー (カスタムヘッダーを含む) にアクセスできるようになったことで、Lambda@Edge を使用して、Amazon S3 バケットに保存されたコンテンツの配信をさらにカスタマイズできるようになりました。詳細については、 ビューワーの場所 および ビューワーのデバイスタイプ に基づいたコンテンツのパーソナライズを示す例を参照してください。	2018 年 3 月 20 日

変更	説明	日付
新機能	Amazon を使用して CloudFront、楕円曲線デジタル署名アルゴリズム (ECDSA) を使用してオリジンへの HTTPS 接続をネゴシートできるようになりました。ECDSA は旧 RSA アルゴリズムと比較してより小さいキーを使用し、より高速でありながら、安全性は同等です。詳細については、 「とオリジン間の通信でサポートされている SSL/TLS プロトコル CloudFront と暗号」 および 「RSA および ECDSA 暗号について」 を参照してください。	2018 年 3 月 15 日
新機能	Lambda@Edge では、Amazon がオリジンから CloudFront 受け取る HTTP エラーに回答して Lambda 関数を実行できるようにすることで、オリジンからのエラーレスポンスをカスタマイズできます。詳細については、 別のロケーションへのリダイレクトの例 および 200 ステータスコード (OK) を伴ったレスポンス生成の例 を参照してください。	2017 年 21 月 12 日
新機能	新しい CloudFront 機能であるフィールドレベル暗号化は、クレジットカード番号や社会保障番号などの個人を特定できる情報 (PII) などの機密データのセキュリティをさらに強化するのに役立ちます。詳細については、 「フィールドレベル暗号化を使用した機密データの保護」 を参照してください。	2017 年 14 月 12 日
ドキュメント履歴のアーカイブ	古いドキュメント履歴がアーカイブされました。	2017 年 12 月

AWS 用語集

AWS の最新の用語については、「AWS の用語集リファレンス」の「[AWS 用語集](#)」を参照してください。

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。