



Amazon CloudFront

(Cache Control 編)

AWS Black Belt Online Seminar

文珠 啓介

Solutions Architect
2023/04

AWS Black Belt Online Seminar とは

- ・ 「サービス別」「ソリューション別」「業種別」などのテーマに分け、アマゾン ウェブ サービス ジャパン合同会社が提供するオンラインセミナーシリーズです
- ・ AWSの技術担当者が、AWS の各サービスやソリューションについてテーマごとに動画を公開します
- ・ 動画を一時停止・スキップすることで、興味がある分野・項目だけの聴講も可能、スキマ時間の学習にもお役立ていただけます
- ・ 以下の URL より、過去のセミナー含めた資料などをダウンロードすることができます
 - ・ <https://aws.amazon.com/jp/aws-jp-introduction/aws-jp-webinar-service-cut/>
 - ・ <https://www.youtube.com/playlist?list=PLzWGOASvSx6FIwIC2X1nObr1KcMCBBlqY>

内容についての注意点

- ・ 本資料では 2023 年 3 月時点のサービス内容および価格についてご説明しています。最新の情報は AWS 公式ウェブサイト (<https://aws.amazon.com/>) にてご確認ください
- ・ 資料作成には十分注意しておりますが、資料内の価格と AWS 公式ウェブサイト記載の価格に相違があった場合、AWS公式ウェブサイトの価格を優先とさせていただきます
- ・ 価格は税抜表記となっています。日本居住者のお客様には別途消費税をご請求させていただきます

自己紹介

名前：文珠 啓介 (Keisuke Monju)

所属：ソリューションアーキテクト

経歴：

- ・オンプレミスのインフラ構築・運用
- ・AWSへのマイグレーション対応・運用

好きなAWSサービス：

Amazon CloudFront、AWS WAF、AWS Firewall Manager



本セミナーの対象者

例：

Amazon CloudFront の概要は知っているが、自社のサービス向けに導入する
メリットとして何があるのかが不明な方

Amazon CloudFront は既に活用しているが、コンテンツキャッシングの用途で
は利用していない方

Amazon CloudFront のコンテンツキャッシングを利用しているが、もっとコン
テンツキャッシングについて知って、より良い運用に繋げたい方

アジェンダ

1. Amazon CloudFrontについて
2. Amazon CloudFront 活用のメリットと今回のメイントピックについて
3. Web サイトを構成する要素
(静的コンテンツと動的コンテンツについて)
4. コンテンツキャッシュの運用方法について
5. Amazon CloudFront とオリジンで行うキャッシングの設定について
6. Amazon CloudFront を利用したキャッシングの運用における Tips
7. クライアント側のキャッシングについて
8. その他

Amazon CloudFront について

450+

Amazon
CloudFront
Points of
Presence

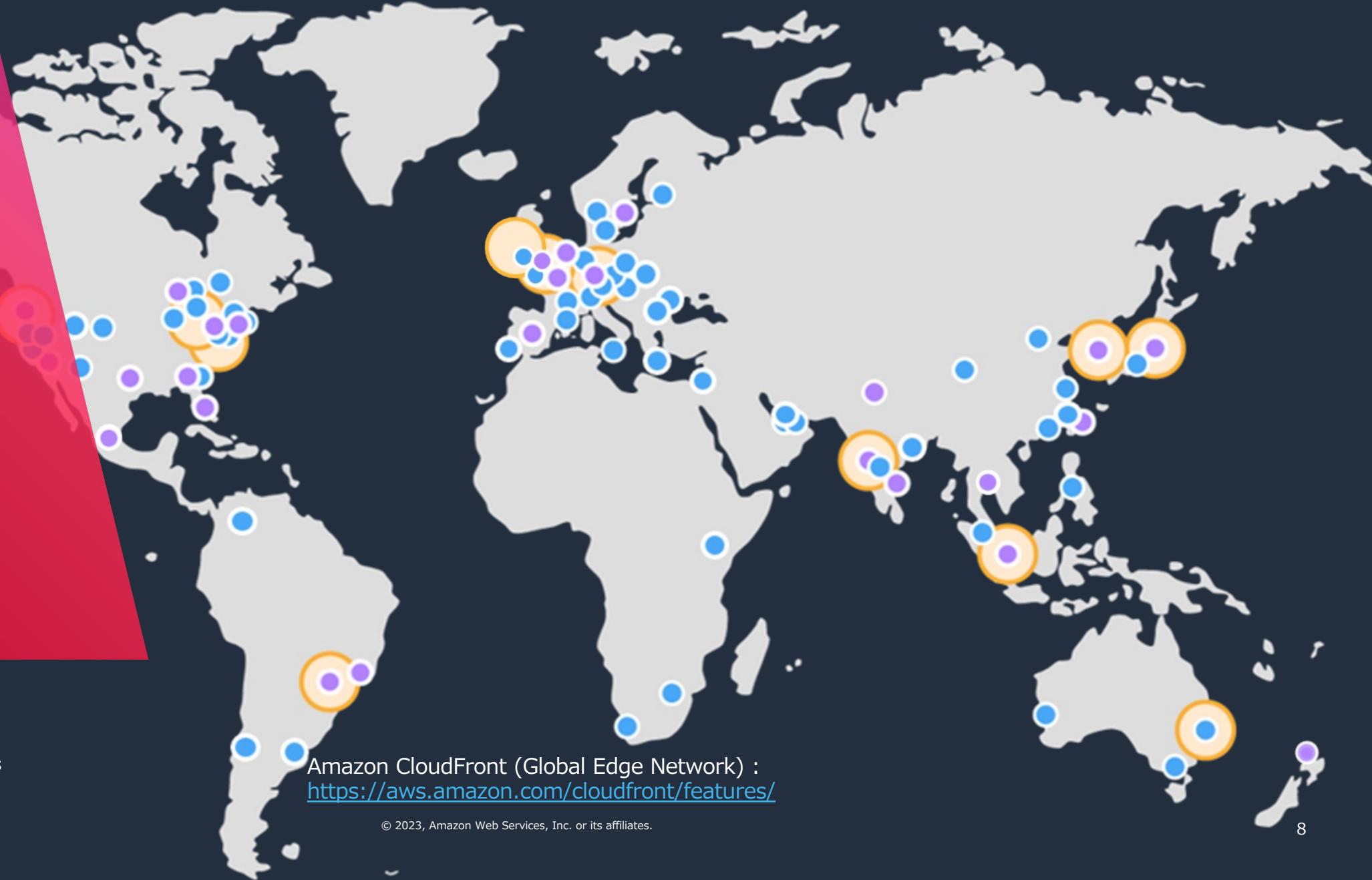
13 Regional
Edge Caches

90+ cities,
48 countries

China CDN

As of 03/09/2023

- Edge Locations
- Multiple Edge Locations
- aws  ● Regional Edge Caches



Amazon CloudFront (Global Edge Network) :
<https://aws.amazon.com/cloudfront/features/>

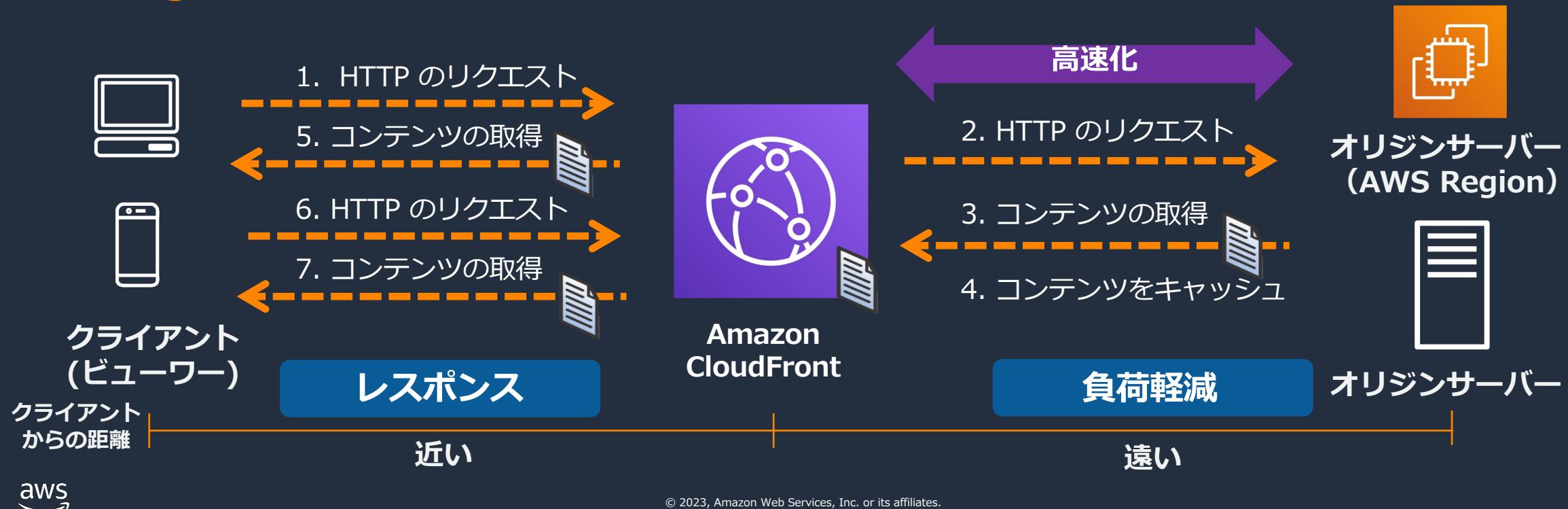
© 2023, Amazon Web Services, Inc. or its affiliates.

Amazon CloudFront の概要

Fast, highly secure and programmable content delivery network (CDN)

高い安全性と高性能を実現するプログラム可能なコンテンツデリバリー・ネットワーク

- ユーザーを一番近いエッジ・ロケーションに誘導することで **配信を高速化**
- エッジ・サーバーでコンテンツのキャッシングを行い **オリジンの負荷をオフロード**
- AWS global network を利用することによる非キャッシングコンテンツの高速化**



Amazon CloudFront 活用のメリットと 今回のメイントピックについて

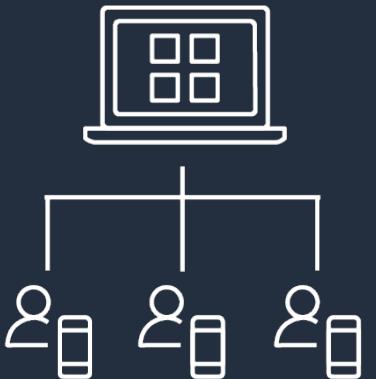
Amazon CloudFront を利用するメリット

AWS のグローバル インフラストラクチャ



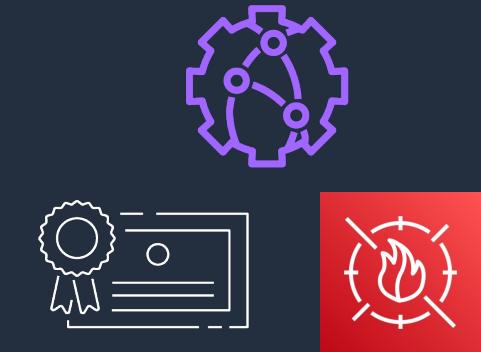
世界のどこからアクセスしても
最適な経路でオリジンまで
アクセスが可能なため
ネットワークのレイテンシーを
最適化できる

エッジロケーションを キャッシュサーバーとして活用



キャッシュされたコンテンツを
エッジロケーションから返すことで
エンドユーザーはレスポンスを早く
受けることができ、サーバーサイド
はコンピュートリソース節約できる

CloudFront の様々な機能や 他のAWSサービスとの連携



エッジ関数の実行、SSL/TLS
通信の終端、AWS WAFによる
セキュリティ対策など、アプリ
ケーション側の変更を必要とせず
動作のカスタマイズができる

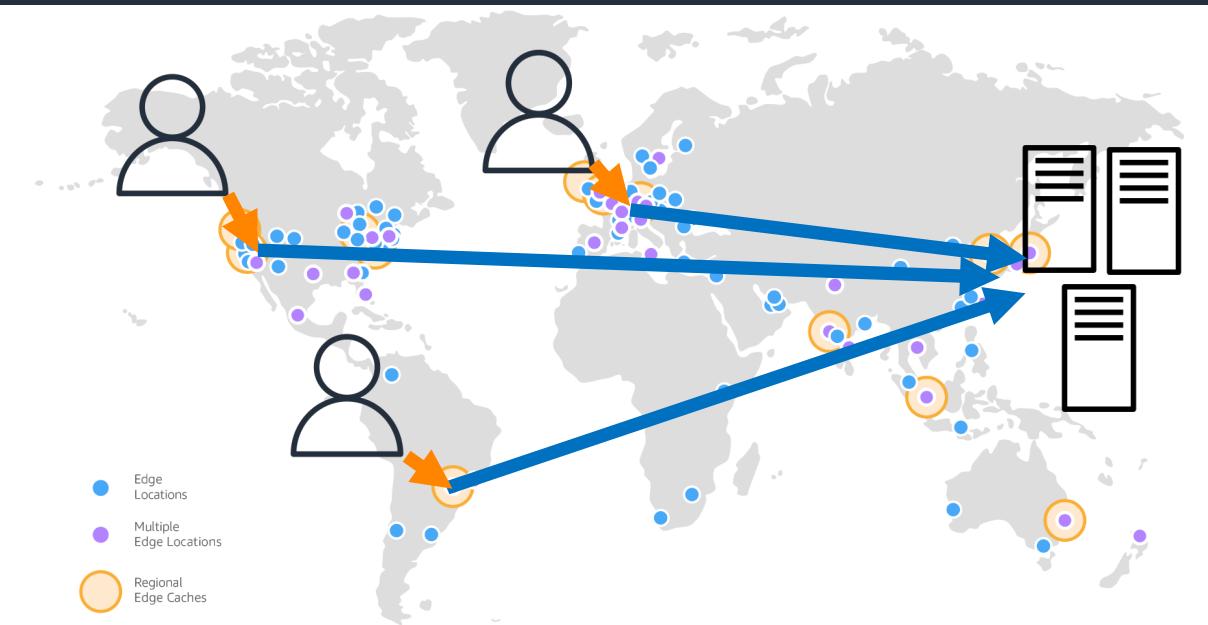
Amazon CloudFront を利用するメリット

エッジロケーションを
キャッシュサーバーとして活用

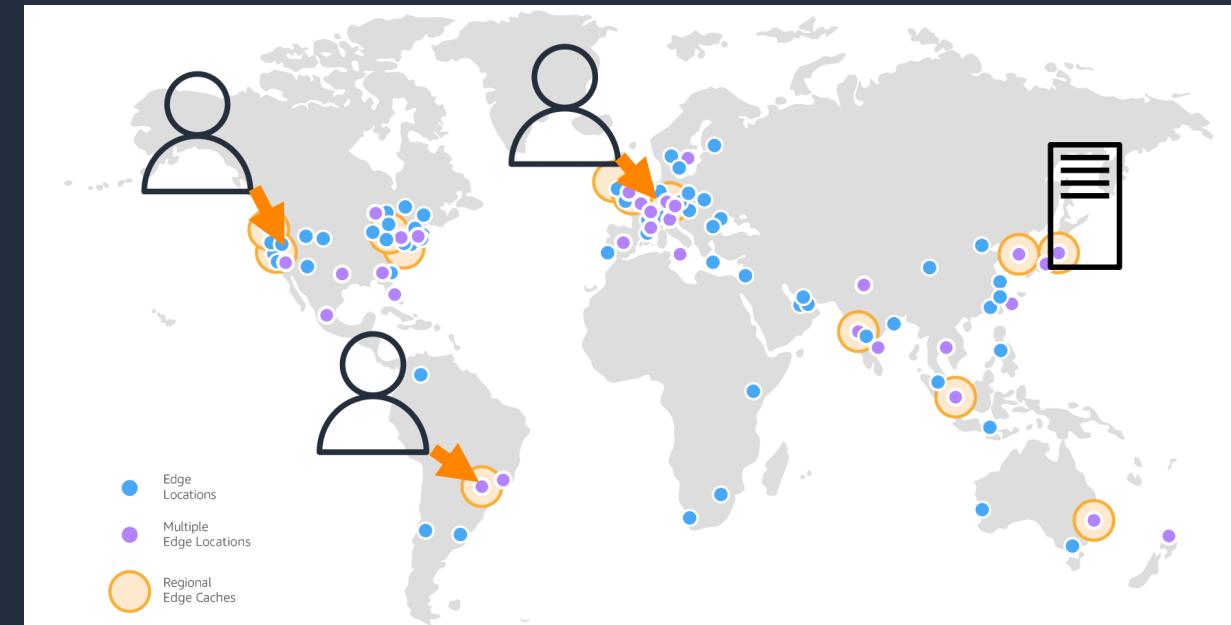


キャッシュされたコンテンツを
エッジロケーションから返すことで
エンドユーザーはレスポンスを早く
受けることができ、サーバーサイド
はコンピュートリソース節約できる

キャッシュがある場合と無い場合の違い



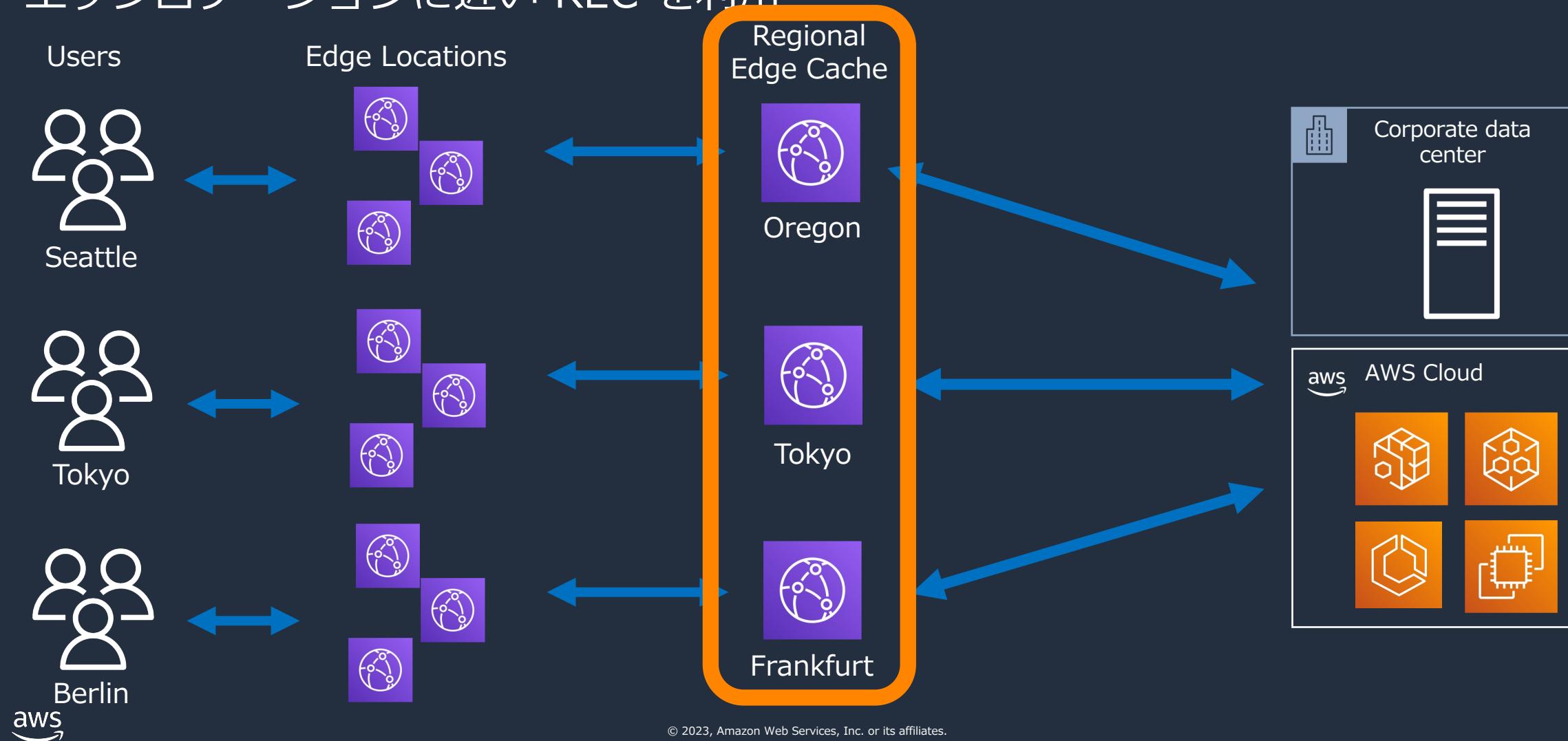
キャッシュが無いアクセス



キャッシュがあるアクセス

リージョナルエッジキャッシュ (REC)

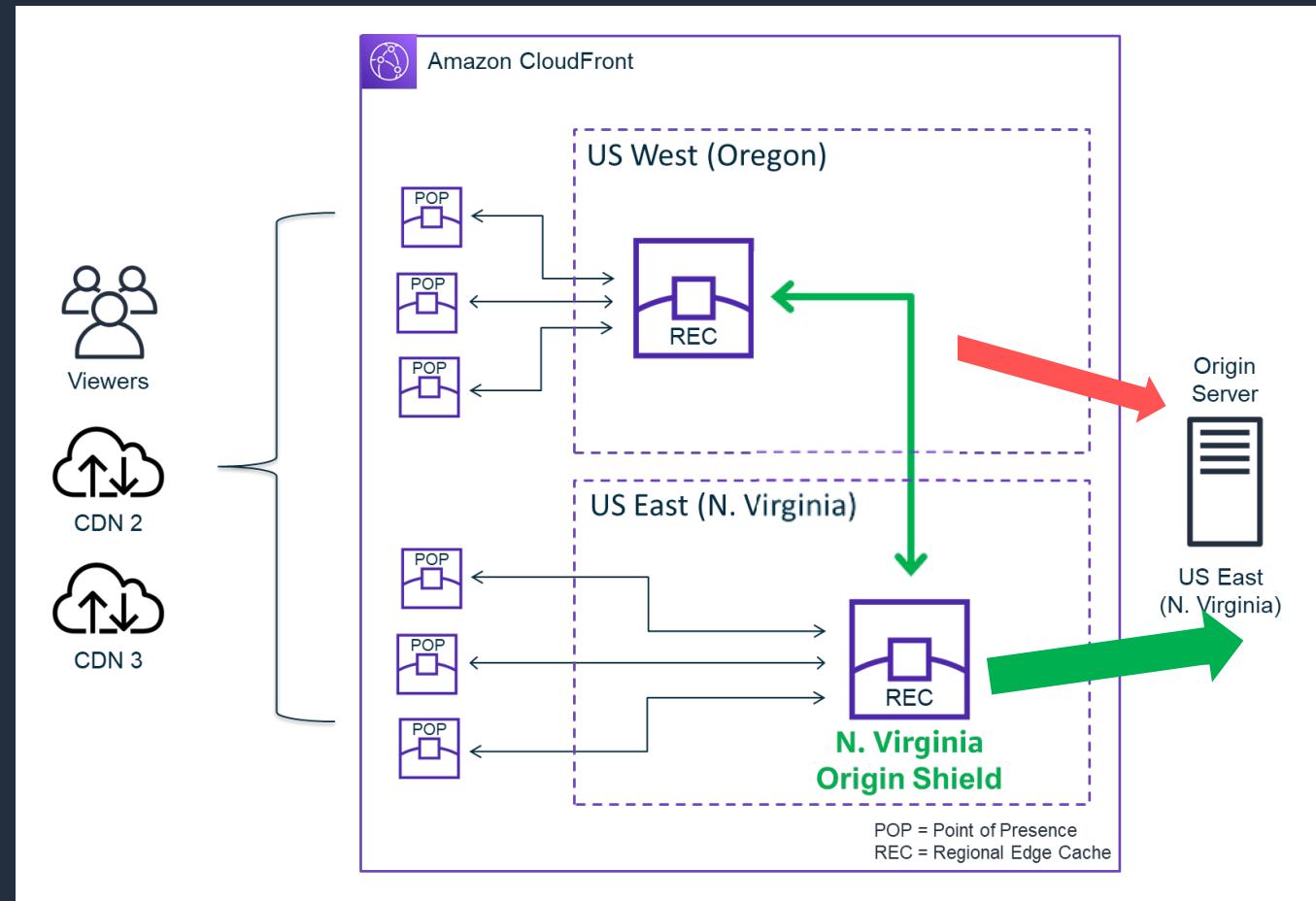
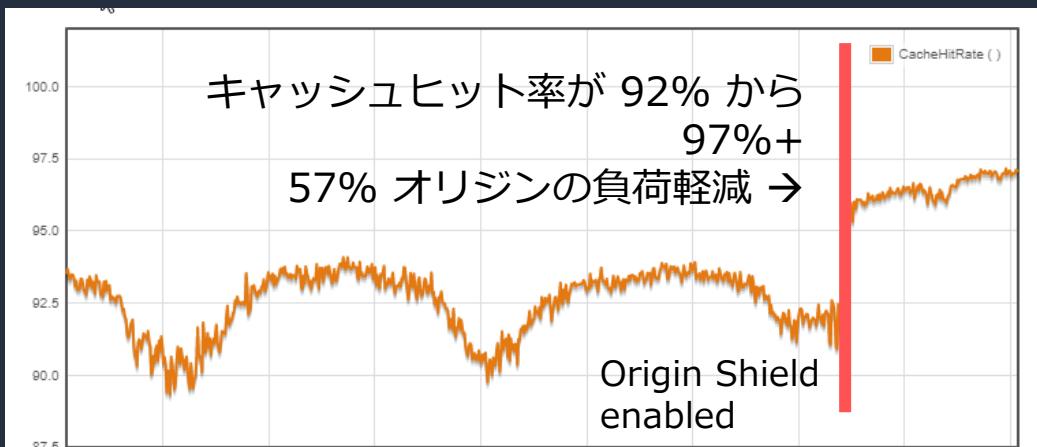
- ユーザーからのリクエストを REC で集約しオリジンにリクエスト
- エッジロケーションに近い REC を利用



Origin Shield

Origin Shield は、**オリジンの負荷と運用コストを削減する**ために、
オリジンの前に配置される**キャッシュレイヤー**

- キャッシュヒット率の向上
- リージョン間の重複したリクエストを集約
- オリジンの可用性向上
- 運用コストの削減 データ転送アウト、
ライブストリーミングのパッケージング処理
やオンザフライのイメージ変換



実際どれくらい早くなるのか？

The diagram illustrates a web application interface on the left and its corresponding Network tab in the developer tools on the right. The interface includes an input message box ('Input Message') containing 'こんにちは世界!!', a send button ('Send'), and an output message box ('Output Message') displaying 'hello world!!'. A red circle highlights the CloudFront logo in the header. Arrows point from the interface components to the developer tools: a blue arrow from the input message to the Network tab, a red arrow from the send button to the Network tab, and a blue arrow from the output message to the Network tab. The developer tools show a list of resources and a waterfall chart. A legend at the top right indicates that red bars represent 'キャッシュコンテンツ' (Cached Content) and blue bars represent '非キャッシュコンテンツ' (Non-Cached Content). In the waterfall chart, the first four items ('document', 'stylesheet', 'script', 'png') are highlighted in green, while the API request ('xhr') is highlighted in red.

HTML
画像
CSS
JavaScript

—— キャッシュコンテンツ
—— 非キャッシュコンテンツ

Input Message
こんにちは世界!!
Send
All Clear
Output Message
hello world!!

キャッシュコンテンツ
非キャッシュコンテンツ

API の結果(動的コンテンツ)

Name	Status	Type	Initiator	Size	Time
d3lnc4nmpnao2n.cloudfront.net	200	document	Other	1.3 kB	611 ms
styles.css	200	stylesheet	(index)	394 B	16 ms
script.js	200	script	(index)	865 B	16 ms
cloudfront.png	200	png	(index)	20.6 kB	9 ms
cloudfront.png	200	png	Other	20.6 kB	10 ms
api?input_text=%26%2312371%...	200	xhr	script.js:11	355 B	916 ms

AWS Hands-on for Beginners (Amazon CloudFrontおよびAWS WAFを用いて エッジサービスの活用方法を学ぼう)
https://pages.awscloud.com/JAPAN-event-OE-Hands-on-for-Beginners-CF_WAF-2022-reg-event.html



ただし、注意も必用



- ・間違えて古いコンテンツをキャッシュしたまま
新規の画面を公開してしまい、表示は古い画面のままだった
※まだ、取り返しがつく



- ・お客様の個人情報をキャッシュしており、設定のミスで
他のお客様に誤って、個人情報が公開されてしまった
※これは、取り返しがつかない



※どんなコンテンツをキャッシュさせるかの取捨選択が大事
(個人情報を含んだコンテンツのキャッシュは避ける！！)

Web サイトを構成する要素 (静的コンテンツと動的コンテンツについて)

Web サイトの構成要素について

www.example.com

HTML/CSS

画像

ブラウザに表示されるURL

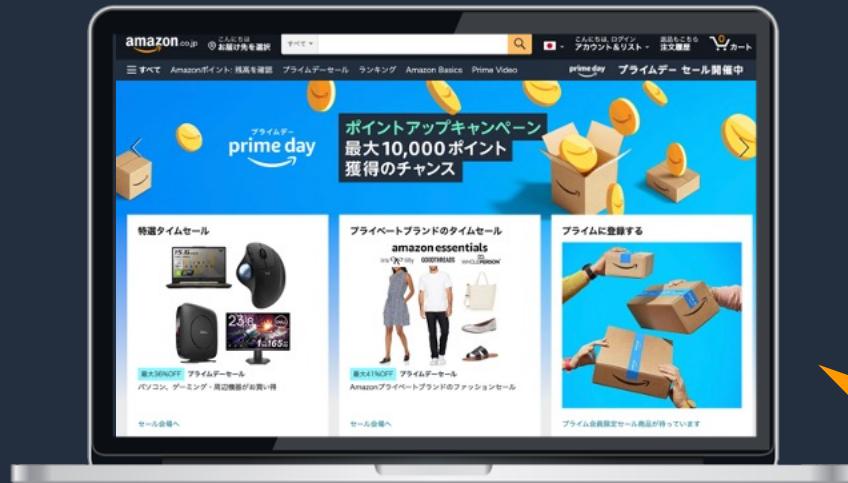
外部のリソース

ユーザー情報

動画など

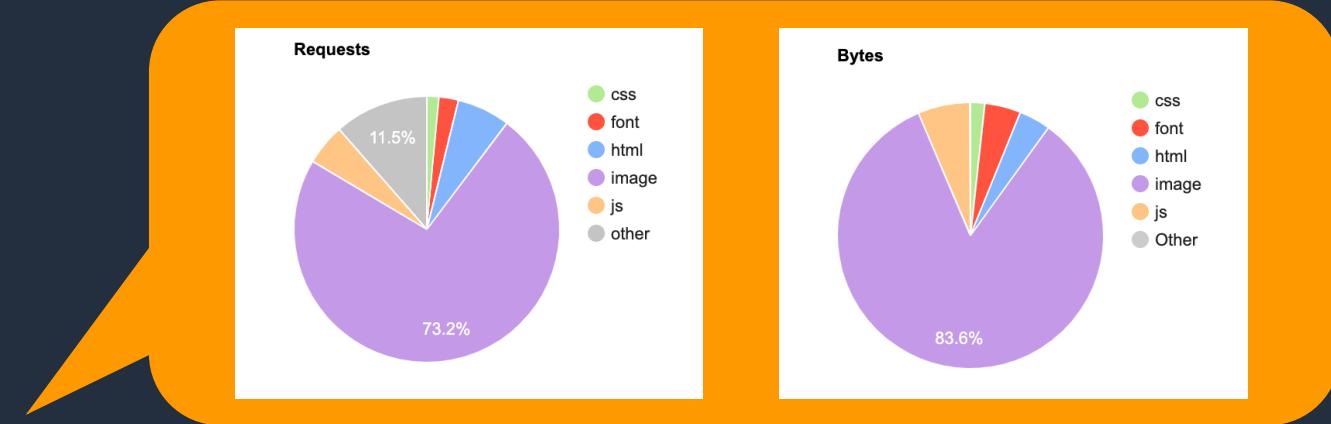
実際の Web サイトの例

画像などの静的コンテンツがリクエスト数、データ量の大半を占める

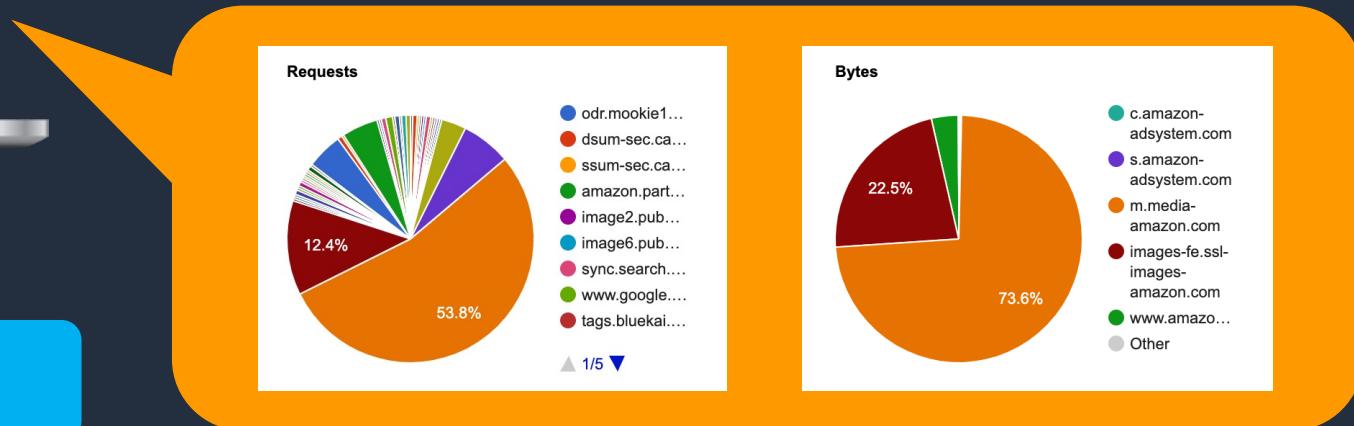


<https://www.amazon.co.jp/>

90-95% はキャッシュ可能



複数のドメインから様々なコンテンツ、アセットがダウンロードされる



Web サイトの様々な要素

- **動的コンテンツ**

ユーザーのリクエストによって変わるコンテンツのこと
(例) ユーザー名や、カートの情報など



- **静的コンテンツ**

ユーザーのリクエストによって変わらないコンテンツのこと
(例) HTML、CSS、画像、動画など



※ Web サイトの大半は**静的なコンテンツ**で構成される

コンテンツキャッシュの 運用方法について

コンテンツキャッシュの運用において最初に考えること



自分のサイトにどういう コンテンツがあるか整理

どのような静的コンテンツや
動的コンテンツによって
自分のサイトが構成されているのか。
また、それらのコンテンツを
運用している部署はどこか。



コンテンツ更新の頻度 (スケジュール)

各コンテンツの更新は
どのタイミングによって発生するか。
その頻度はどれくらいか。
キャッシングするコンテンツと
しないコンテンツの割合はどうか。



誰がどういう方法でコンテンツ のキャッシング更新を行うか

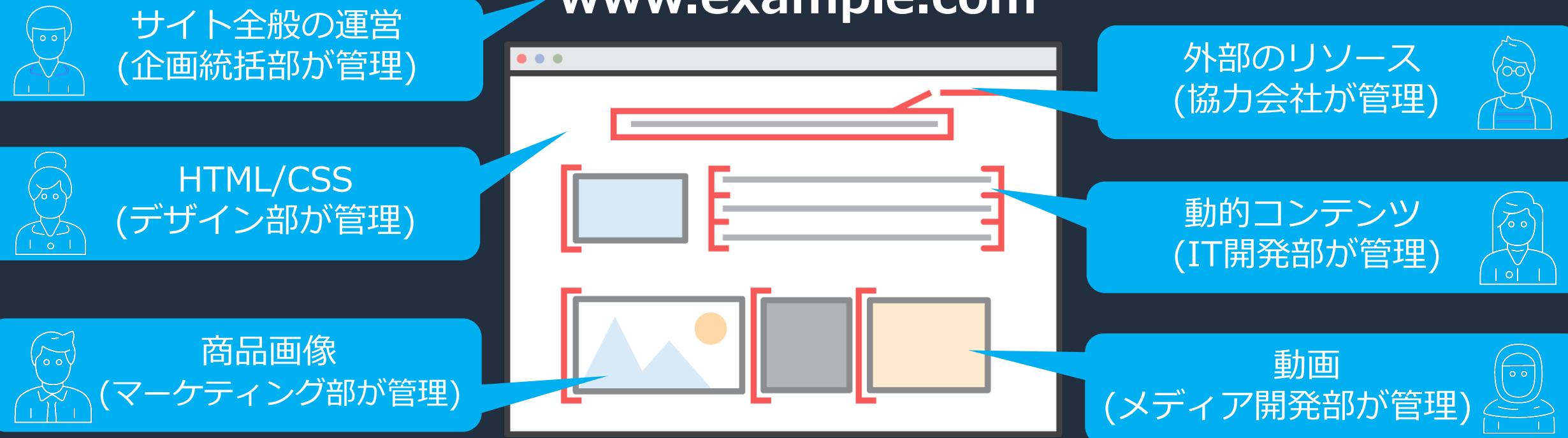
実際に誰がコンテンツの
更新を行うのか。
コンテンツを更新した時に
キャッシングをどのような方法
で更新するのか。

コンテンツキャッシュの運用において最初に考えること

※ Web サイトのコンテンツに、様々な部署が関わるのは一般的
部署ごとにコンテンツを保持しているサーバーが異なることも
(コンテンツを更新するスケジュールも方法もバラバラ)



www.example.com

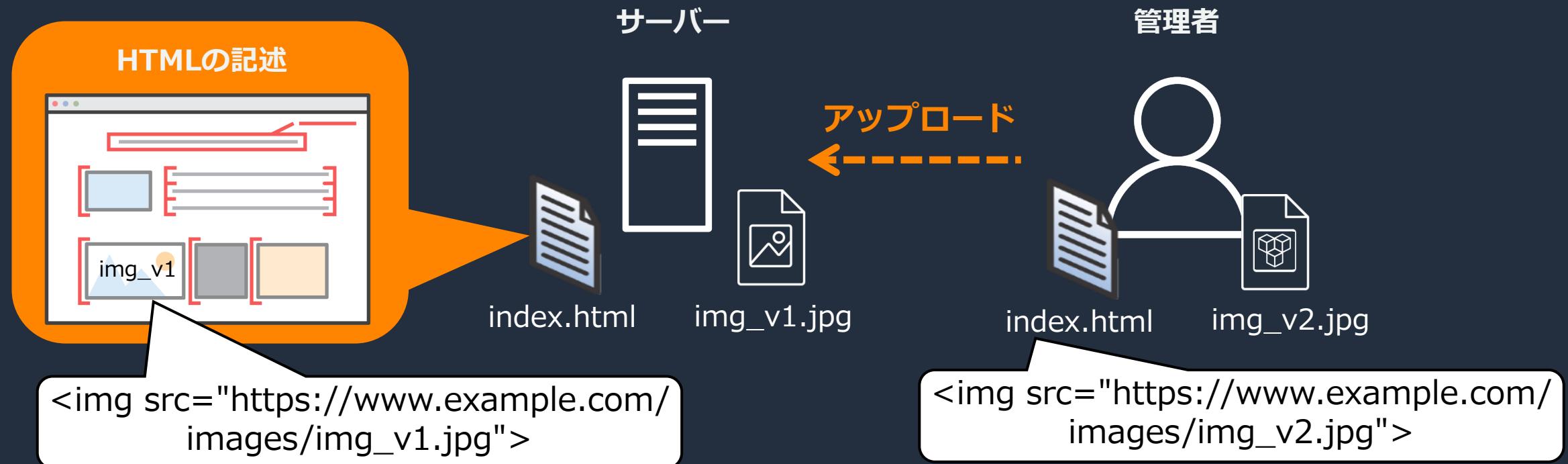


既存コンテンツのキャッシュを更新するための 2 つの方法

- ファイル名にバージョン識別子を使用して更新する
- 同じ名前を使用してファイルを更新する

既存コンテンツのキャッシュを更新するための 2 つの方法

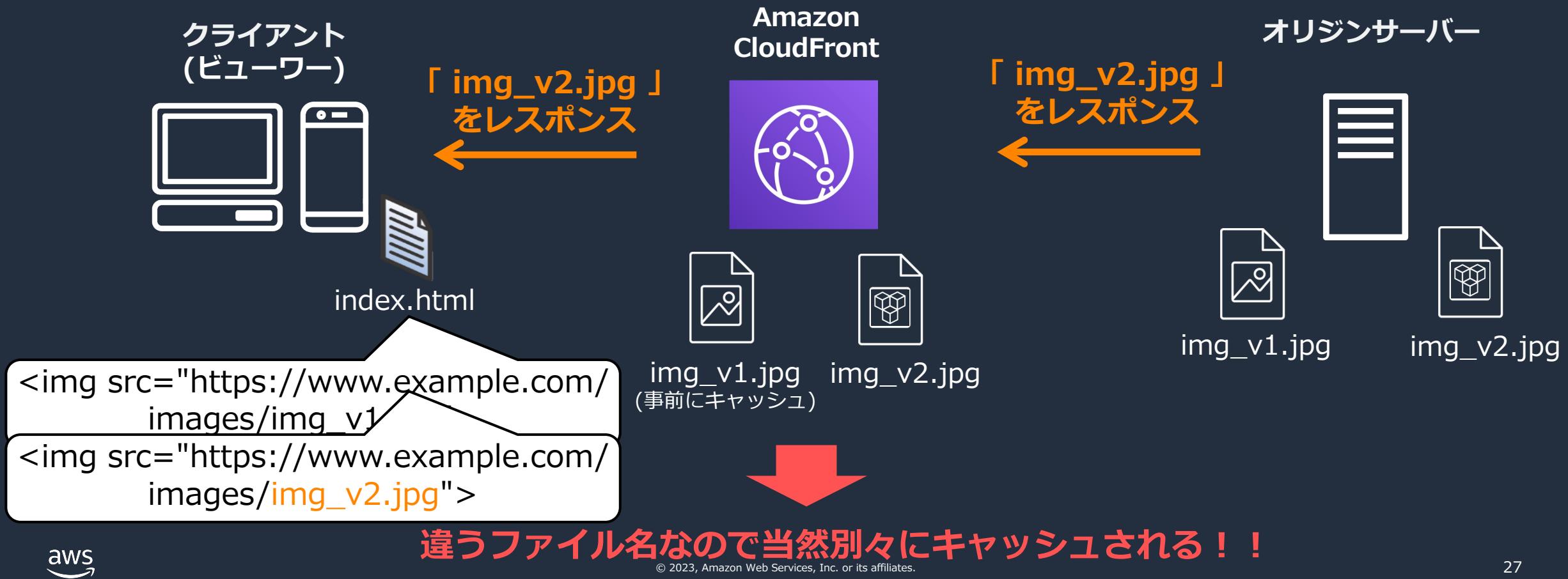
- ファイル名にバージョン識別子を使用して更新する
- 同じ名前を使用してファイルを更新する



既存のコンテンツと CloudFront ディストリビューションを更新する
aws https://docs.aws.amazon.com/ja_jp/AmazonCloudFront/latest/DeveloperGuide/UpdatingExistingObjects.html

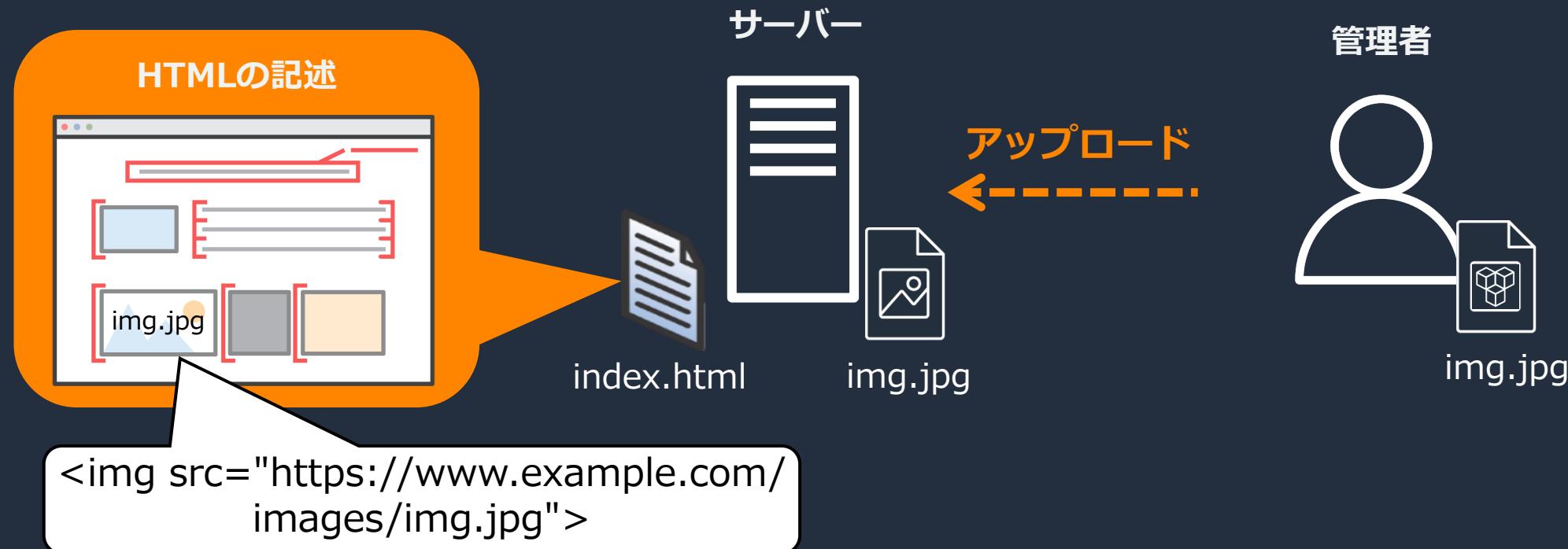
既存コンテンツのキャッシュを更新するための 2 つの方法

- ファイル名にバージョン識別子を使用して更新する
- 同じ名前を使用してファイルを更新する



既存コンテンツのキャッシュを更新するための 2 つの方法

- ファイル名にバージョン識別子を使用して更新する
- 同じ名前を使用してファイルを更新する



既存のコンテンツと CloudFront ディストリビューションを更新する
aws https://docs.aws.amazon.com/ja_jp/AmazonCloudFront/latest/DeveloperGuide/UpdatingExistingObjects.html

既存コンテンツのキャッシュを更新するための 2 つの方法

- ファイル名にバージョン識別子を使用して更新する
- 同じ名前を使用してファイルを更新する



1つのコンテンツに紐付いているメタ情報について

※実は1つのコンテンツに対して大量のメタ情報が付与されている！！
(ファイル名が同じでも、これらの情報を用いて別物だと認識させることが可能)



1コンテンツ

Name: storefront?storeType=browse&node=2275256051
Request URL: https://www.amazon.co.jp/kindle-dbs/storefront?storeType=browse&node=2275256051
Request Method: GET
Status Code: 200
Remote Address: 13.249.153.129:443
Referrer Policy: strict-origin-when-cross-origin

General
Headers
Payload
Preview
Response
Initiator
Timing
Cookies

Request URL: https://www.amazon.co.jp/kindle-dbs/storefront?storeType=browse&node=2275256051
Request Method: GET
Status Code: 200
Remote Address: 13.249.153.129:443
Referrer Policy: strict-origin-when-cross-origin

Response Headers
accept-ch: ect,rtt,downlink,device-memory,sec-ch-device-memory,viewport-width,sec-ch-viewport-width,dpr,sec-ch-dpr,sec-ch-ua-platform,sec-ch-ua-platform-version
accept-ch-lifetime: 86400
cache-control: no-cache
content-encoding: gzip
content-language: ja-JP
content-security-policy: upgrade-insecure-requests;report-uri https://metrics.media-amazon.com/
content-security-policy-report-only: default-src 'self' blob: https: data: mediastream: 'unsafe-eval' 'unsafe-inline';report-uri https://metrics.media-amazon.com/
content-type: text/html; charset=UTF-8

1つのコンテンツに紐づく情報
(ブラウザの開発者ツールより。)

Cookies

クエリ文字列

ヘッダー

etag: "1be29193e65f4fee5fa62d7a4d7d9305"
last-modified: Tue, 21 Mar 2023 13:28:52 GMT

(補足) キャッシュ期間が切れたあとの挙動について

キャッシュしたコンテンツに変更がない場合

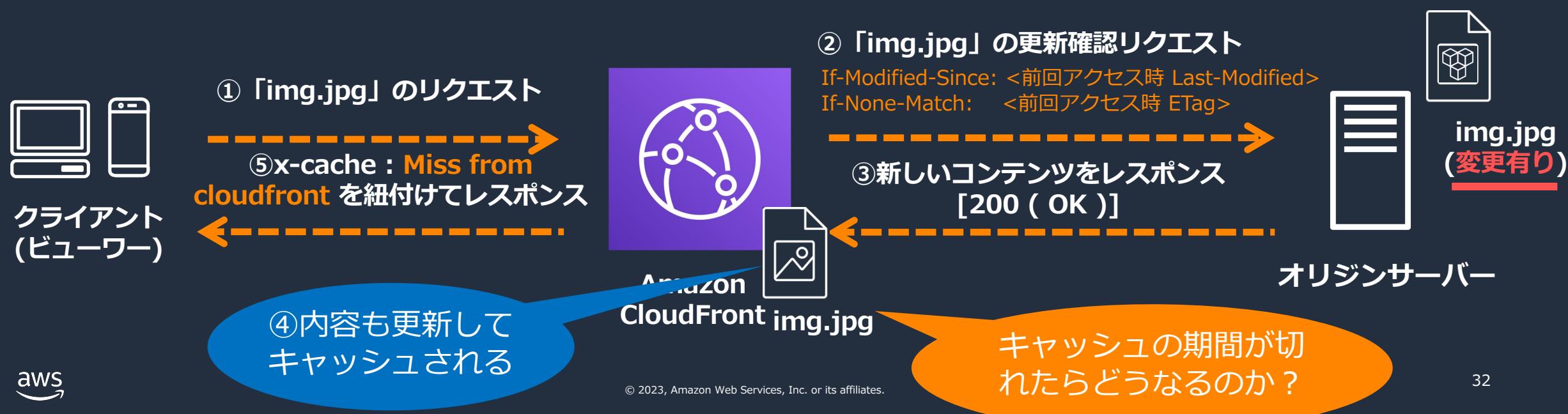
- CloudFront にてキャッシュしているコンテンツの期間がきた際、
Etag / LastModified といったコンテンツに紐付いている HTTP ヘッダー
の値を用いてオリジンサーバーに確認のリクエストを送る (If-Modified-Since / If-None-Match)
- レスポンスとして上記の値に変更がなければステータスコード 304 (Not Modified) が
返ってきてコンテンツのキャッシュ期間は更新される
(x-cache : RefreshHit from cloudfront をレスポンス)



(補足) キャッシュ期間が切れたあとの挙動について

キャッシュしたコンテンツに変更がある場合

- CloudFront にてキャッシュしているコンテンツの期間がきた際、
Etag / LastModified といったコンテンツに紐付いている HTTP ヘッダー
の値を用いてオリジンサーバーに確認のリクエストを送る (If-Modified-Since / If-None-Match)
- レスポンスとして上記の値に変更があれば、コンテンツを新しいものに差し替えて
ステータスコード 200 (OK) とともに新しいコンテンツをレスポンス
(x-cache : Miss from cloudfront を紐付けてレスポンス)



既存コンテンツのキャッシュを更新するための 2 つの方法

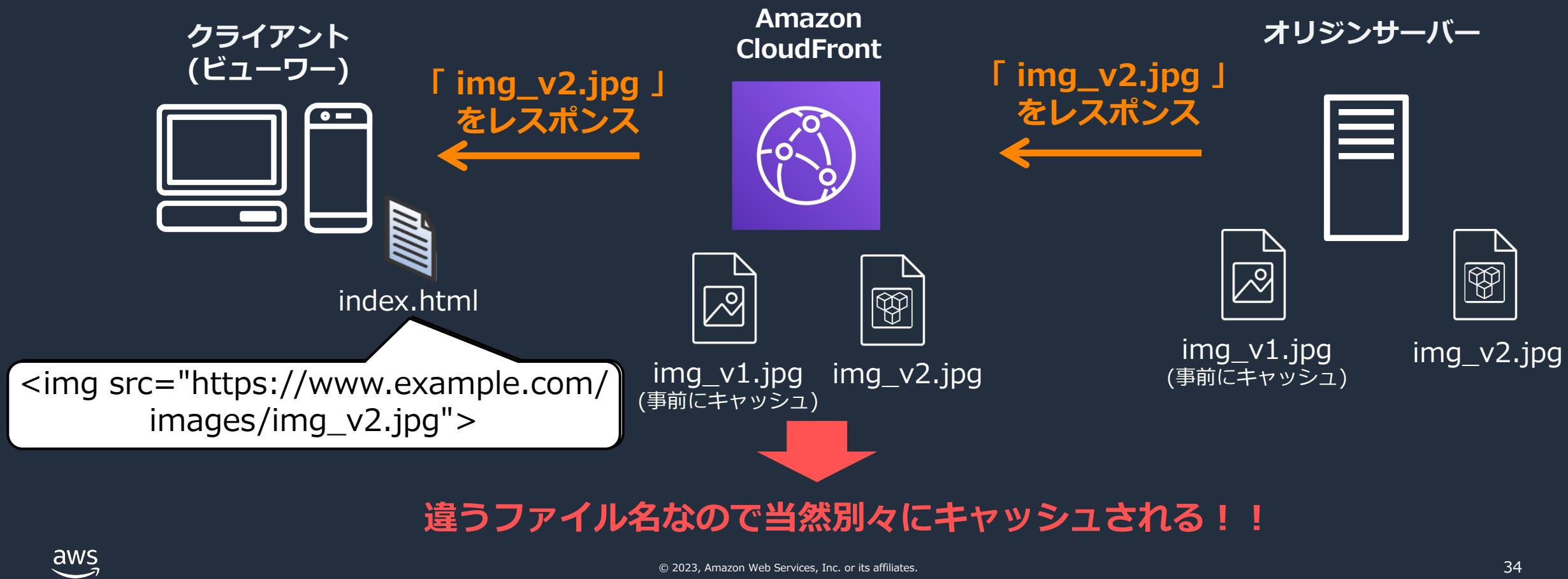
- ファイル名にバージョン識別子を使用して更新する
- 同じ名前を使用してファイルを更新する

注意点

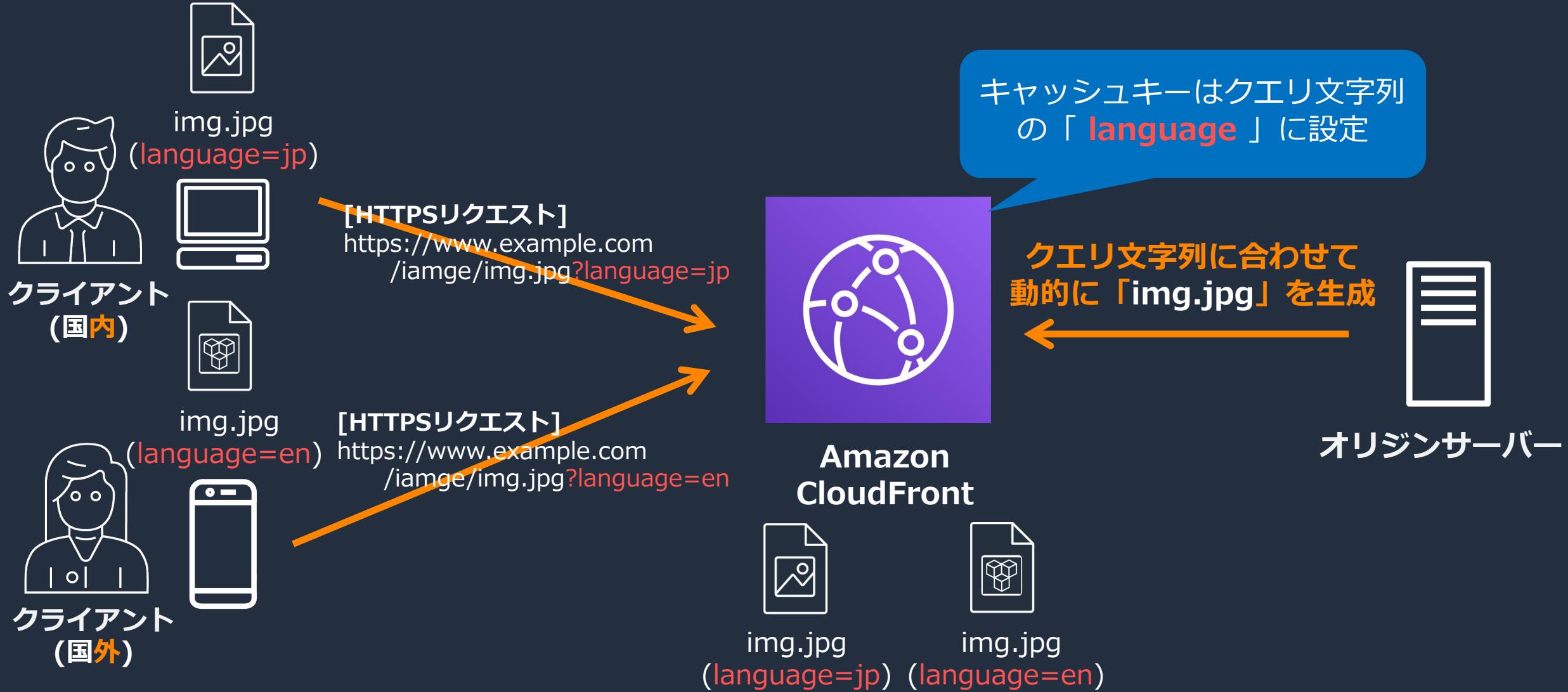
- キャッシュの有効期限を上手く制御して、コンテンツごとにどれくらいの期間をキャッシュさせるのか決めて運用する必要がある
- キャッシュの有効期限を長くすることで、オリジンサーバーへのリクエストを減らせるが、コンテンツの更新が頻繁に発生する場合には、有効期限を短くする必要がある
- 指定したタイミングで一斉にキャッシュの更新をしたい場合などには、キャッシュの無効化対応が必要になる

既存コンテンツのキャッシュを更新するための 2 つの方法

- ファイル名にバージョン識別子を使用して更新する
- 同じ名前を使用してファイルを更新する



キャッシュキーを用いて別のコンテンツとしてキャッシュする



CloudFront におけるキャッシュの設定について

大量に付与されているメタ情報の何をキーにして CloudFront は
“同じ名前のファイルを別のキャッシュとして保管する”のか？

重要！

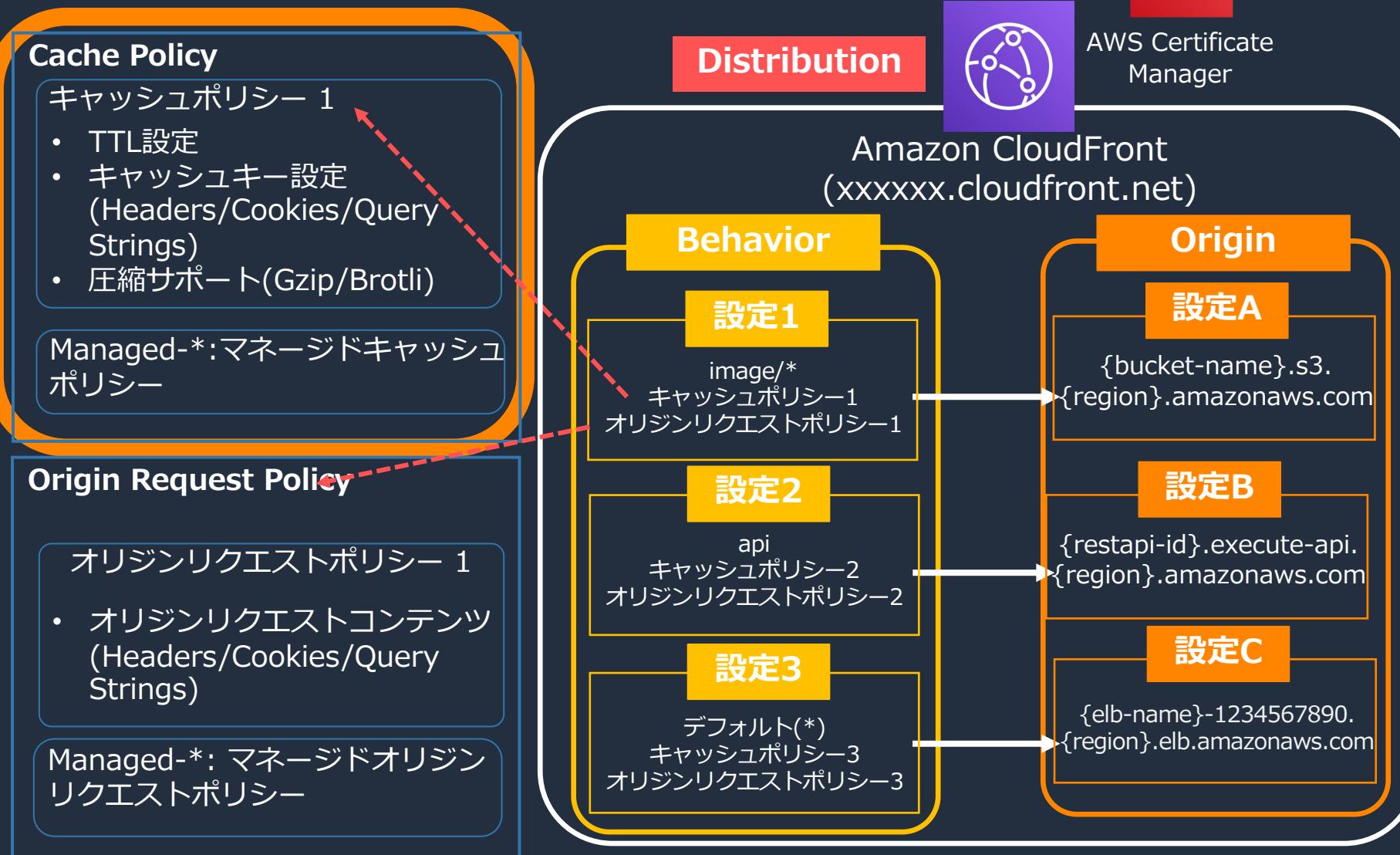


Amazon
CloudFront

Cache Policy
(キャッシュポリシー)

Amazon CloudFront とオリジンで 行うキャッシュの設定について

CloudFront の設定イメージ



AWS Certificate Manager

オリジンサーバー



Amazon S3



Amazon API Gateway



AWS Lambda



Application Load Balancer



Amazon EC2

Cache Policy の設定画面

● 3つのセクション

・ TTL (Time to live) 設定

CloudFront キャッシュ内のオブジェクトの有効期間を決定する（オリジン側の「Cache-Control」および「Expires」HTTP ヘッダーに連動する）

・ キャッシュキー設定

CloudFront がコンテンツをキャッシュする際に、一意のコンテンツであることを判断するための、キーを指定する（ヘッダー、クエリストリング、Cookie の要素を指定できる）

・ 圧縮サポート

ビューワーからのリクエストに基づいて、特定のタイプのコンテンツを必要に応じて自動的に圧縮し、キャッシュ及びレスポンスするための設定

圧縮ファイルの供給

https://docs.aws.amazon.com/ja_jp/AmazonCloudFront/latest/DeveloperGuide/ServingCompressedFiles.html



[補足] Cache Policy における「Amazon マネージドポリシー」

マネージドポリシーについて

- AWS 側が用途に合わせて予め用意しているテンプレート
- そのまま使えそうなら使っても良い(参考として、中身を覗いてみるのをオススメ)
- 自身で作成するポリシーはカスタムポリシーと呼ばれる

The screenshot shows the CloudFront Cache Policy configuration page. The top navigation bar includes 'CloudFront > ポリシー > キャッシュ' (CloudFront > Policies > Cache). Below the navigation, there are three tabs: 'キャッシュ' (Cache), 'オリジンリクエスト' (Origin Request), and 'レスポンスヘッダー' (Response Header). The 'キャッシュ' tab is selected. A large orange box highlights the 'Amazon マネージドポリシー' (Amazon Managed Policies) section. This section lists five managed policies with their descriptions:

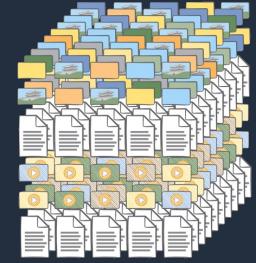
名前	説明
Amplify	Policy for Amplify Origin
CachingDisabled	Policy with caching disabled
CachingOptimized	Policy with caching enabled. Supports Gzip and Brotli
CachingOptimizedForUncompressedObjects	Default policy when compression is disabled
Elemental-MediaPackage	Policy for Elemental MediaPackage Origin

At the bottom of the highlighted section, there is a red box containing the text 'カスタムポリシー (3) 情報' (Custom Policies (3) Information) and three buttons: '編集' (Edit), '削除' (Delete), and 'キャッシュポリシーを作成' (Create Cache Policy). The entire 'Amazon マネージドポリシー' section is also enclosed in a red box.

名前	説明
Amplify	Policy for Amplify Origin
CachingDisabled	Policy with caching disabled
CachingOptimized	Policy with caching enabled. Supports Gzip and Brotli
CachingOptimizedForUncompressedObjects	Default policy when compression is disabled
Elemental-MediaPackage	Policy for Elemental MediaPackage Origin

カスタムポリシー (3) 情報 編集 削除 キャッシュポリシーを作成

[補足] キャッシュコントロール機能



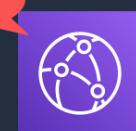
CloudFront のキャッシュコントロールにおける注意事項

- GET / HEAD / OPTION (選択可能) のリクエストがキャッシュ対象
(CloudFront はその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ない)
 - 単一リクエストで取得できるキャッシュの最大サイズは 30GB (範囲リクエストを使用しない場合)
※範囲リクエストを使用することで 30GB を超えるオブジェクトを分割してキャッシュが可能
 - URL および Cache Policy で有効化した HTTP ヘッダー、クエリ文字列、
Cookie パラメータ値の完全一致でキャッシュが再利用される
- ※ Cache Policy にて、全ての TTL の値が 0 の場合は
CloudFront ではキャッシュをしない
(マネージドポリシーの CachingDisabled の利用)

再掲

- 「同じ名前を使用してファイルを更新する」際の話
- 大量に付与されているメタ情報の何をキーにして CloudFront は
“同じ名前のファイルを別のキャッシュとして保管する”のか？

重要！



Cache Policy
(キャッシュポリシー)

CloudFront がオブジェクトの部分的リクエスト (レンジ GET) を処理する方法
https://docs.aws.amazon.com/ja_jp/AmazonCloudFront/latest/DeveloperGuide/RangeGETs.html



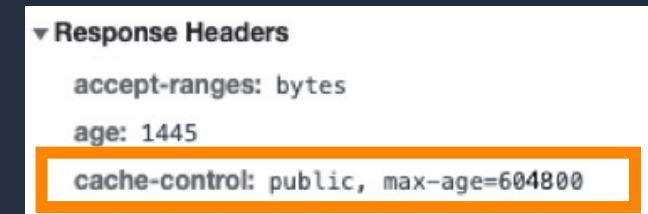
CloudFront の TTL と オリジン側の Cache-Control ヘッダーの関係について

Q. 例えば下記の各々の設定をした際にコンテンツはどれくらいの間
CloudFront にキャッシュされるだろうか？

「*.jpg」のパスに対するビヘイビアのキャッシュポリシーの設定



Apache HTTP Server の
Cache-Control ヘッダーの設定



① 「img.jpg」を初めてリクエスト



④ 「img.jpg」をレスポンス

クライアント
(ビューウィー)



Amazon
CloudFront
img.jpg

② 「img.jpg」を初めてリクエスト



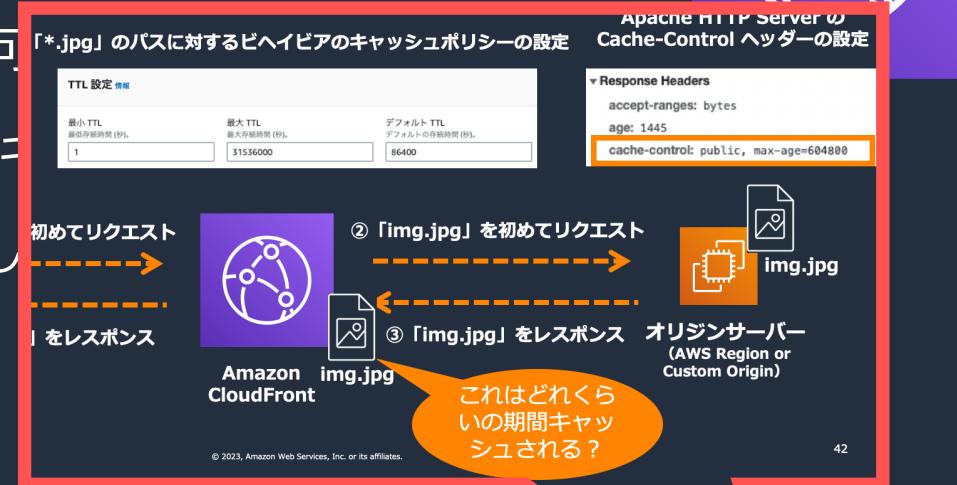
③ 「img.jpg」をレスポンス オリジンサーバー
(AWS Region or Custom Origin)

これはどれくらいの期間キャッシュされる？

CloudFront 側のキャッシングについて



- オリジンの Cache-Control ヘッダーでキャッシング時間の設定が可
 - Behavior 毎に異なる設定を行うことで、URL パスパターン毎に設定可能
- デフォルト TTL : オリジンが Cache-Control ヘッダーを指定しない場合
- 最小 TTL : CloudFront でキャッシングすべき最小期間
- 最大 TTL : CloudFront でキャッシングすべき最大期間



Cache Policy 最小 TTL 設定

オリジン HTTP ヘッダー	最小 TTL = 0 秒		最小 TTL > 0 秒を設定	
	Cache-Control max-age を指定	指定された max-age と最大 TTL で小さい値の期間キャッシング	最小 TTL < max-age < 最大 TTL	max-age 期間
Cache-Control 設定なし		デフォルト TTL 期間キャッシング	max-age < 最小 TTL	最小 TTL 期間
			最大 TTL < max-age	最大 TTL 期間

CloudFront 側のキャッシュについて



Q. 「Cache-Control : max-age/s-maxage/no-cache/no-store /private」、「Expires」HTTP ヘッダーとは何者なのか？

ヘッダーと HTTP リクエスト

Cache-Control max-age と s-maxage を指定

Expires を指定

Cache-Control no-cache, no-store、および(または) private ディレクティブを追加

Cache Policy 最小 TTL 設定

	最小TTL = 0秒	最小TTL > 0秒を設定	
Cache-Control max-age と s-maxage を指定	指定された s-max-age と最大 TTL で小さい値の期間キャッシュ	最小TTL < s-max-age < 最大 TTL	s-max-age 期間
Expires を指定	指定された Expires 日付と最大 TTL で早い日付の期間キャッシュ	s-max-age < 最小 TTL	最小 TTL 期間
Cache-Control no-cache, no-store、および(または) private ディレクティブを追加	ヘッダーを優先させる	最大 TTL < s-max-age	最大 TTL 期間
		最小 TTL < 最大 TTL	Expires 日付
		Expires < 最小 TTL	最小 TTL 期間
		最大 TTL < Expires	最大 TTL 期間
		最小 TTL の期間キャッシュ	

※ S3 がオリジンの場合は S3 オブジェクト Metadata に Cache-Control, Expires を指定可能

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

aws https://docs.aws.amazon.com/ja_jp/AmazonCloudFront/latest/DeveloperGuide/Expiration.html

© 2023, Amazon Web Services, Inc. or its affiliates.

オリジンサーバー側の「Cache-Control」および「Expires」HTTP ヘッダーの設定

Apache HTTP Server (ver 2.4) での設定例

- **Headers モジュール [mod_headers]**

```
Header set Cache-Control "public,max-age=604800"
```

- **Expires モジュール [mod_expires]**

```
ExpiresByType text/css "access plus 1 month 15 days 2 hours"
```



オリジンサーバー
(AWS Region or
Custom Origin)

- **Cache-Control ヘッダー**

- max-age . . . レスポンスが生成されてから N 秒後まで、レスポンスが新鮮なままであることを示す。
- no-cache . . . キャッシュに保存できることを示す。キャッシュがオリジンサーバーから切断された場合でも、再利用の前にオリジンサーバーで検証を行うことを指示する。
- no-store . . . あらゆる種類のキャッシュが、このレスポンスを保存しないようにすることを指示する。

- **Expires ヘッダー** . . . レスポンスが古くなると見なされる日時を指定する。

※Cache-Control max-age と Expires の両方の値を指定した場合、CloudFront は 前者の値のみを使用。

Apache HTTP Server バージョン 2.4 ドキュメント
https://httpd.apache.org/docs/2.4/ja/mod/mod_expires.html

MDN Web Docs (Cache-Control)
<https://developer.mozilla.org/ja/docs/Web/HTTP/Headers/Cache-Control>

Amazon CloudFront を利用した キャッシュの運用における Tips

CloudFront の設定イメージ

Cache Policy

- キャッシュポリシー 1
 - TTL設定
 - キャッシュキー設定 (Headers/Cookies/Query Strings)
 - 圧縮サポート(Gzip/Brotli)
- Managed-*: マネージドキャッシュポリシー

Origin Request Policy

- オリジンリクエストポリシー 1
 - オリジンリクエストコンテンツ (Headers/Cookies/Query Strings)
- Managed-*: マネージドオリジンリクエストポリシー

Distribution



AWS Certificate Manager



Amazon CloudFront
(xxxxxxxx.cloudfront.net)

Behavior

設定1

image/*
キャッシュポリシー1
オリジンリクエストポリシー1

設定2

api
キャッシュポリシー2
オリジンリクエストポリシー2

設定3

デフォルト(*)
キャッシュポリシー3
オリジンリクエストポリシー3

Origin

設定A

{bucket-name}.s3.
{region}.amazonaws.com

設定B

{restapi-id}.execute-api.
{region}.amazonaws.com

設定C

{elb-name}-1234567890.
{region}.elb.amazonaws.com

オリジンサーバー



Amazon S3



Amazon API
Gateway



AWS Lambda



Application
Load Balancer



Amazon EC2

[補足] Cache Policy と Origin Request Policy の連携

- オリジンに転送するリクエストとキャッシュキーを分離して取り扱うことにより、より柔軟なキャッシュ設定が可能
- 事前定義済みのマネージドポリシーの他に、カスタムポリシーの作成・適用が可能

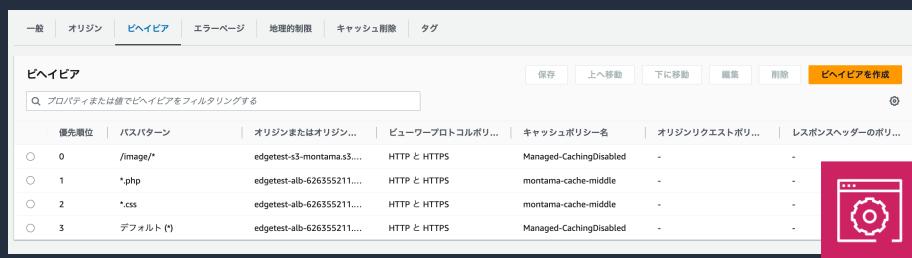


※ Origin Request Policy の設定画面



きめ細やかなキャッシングの実現

- Cache Policy / Origin Request Policy を組み合わせ、HTTP ヘッダー, Cookie, クエリ文字列をオリジンリクエストに含めるか否かをパスパターン毎に自由にカスタマイズできる
- クライアントのリクエストパターンをもとに、複数の URL パスパターンの Behavior とマルチオリジンを組み合わせ、きめ細かなキャッシングコントロールを実現



Behaviors Path Pattern の記述方法

- 「*」 0もしくはそれ以上の文字列
 - 「?」 1文字
- 例) /*.jpg, /image/*, /image/a*.jpg, /a???.jpg

キャッシュファイルの無効化 (Invalidation)

- ・ コンテンツ毎の無効化パス指定
同時に最大 3,000 個までのパス指定が可能
- ・ ワイルドカードを利用した無効化パス指定
 - ・ 同時に最大 15 個まで無効化パスリクエストが指定可能
 - ・ オブジェクト数の制限無し
- ・ AWS Management Console / CLI / SDK で実行可能
- ・ キャッシュファイルの無効リクエストは有償のため、オリジンレスポンスヘッダーの `Cache-Control: max-age` または `s-maxage` や `Expires` の指定、キャッシュポリシーのデフォルト TTL で適切なキャッシュ期間を設定することを推奨
 - ・ 最初の 1,000 パスまでは追加料金無し、それ以降は、無効をリクエストしたパスごとに \$0.005

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	3,000
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

キャッシュ削除を作成

オブジェクトパス

オブジェクトパスを追加
CloudFront キャッシュから削除する各オブジェクトのパスを追加します。ワイルドカード (*) を使用できます。

② オブジェクトパスを個別に追加するには、[標準エディタ](#) を使用します。

キャンセル

キャッシュ削除を作成

ファイルの無効化



https://docs.aws.amazon.com/ja_jp/AmazonCloudFront/latest/DeveloperGuide/Invalidation.html

© 2023, Amazon Web Services, Inc. or its affiliates.

CloudFront がオリジンからの HTTP 4xx および 5xx ステータスコードを処理してキャッシュする方法について

- エラーキャッシュの最小 TTL はデフォルトで 10秒 である
 - カスタムエラーレスポンスの定義で変更可能
- 4xx Error に関しては、ステータスコードによってキャッシュの挙動が異なる
 - 一部に関しては Cache-Control ヘッダーの max-age または s-maxage が返ってきたときのみキャッシュする
- CloudFront に既存のキャッシュが残っているか(有効期限切れを含む)、否かで CloudFront のレスポンスの挙動が変わる
- 5xx Error か 4xx Error かでキャッシュの有効期限が切れた際に、CloudFront が保持しているキャッシュをそのままレスポンスするか否かが異なる
- カスタムエラーレスポンス用のページが設定されているか否かで挙動が変わる

CloudFront が常にキャッシュする
HTTP ステータスコード

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 HTTP ステータスコード

400	Bad Request
403	Forbidden
405	Method Not Allowed
412	Precondition Failed
415	Unsupported Media Type

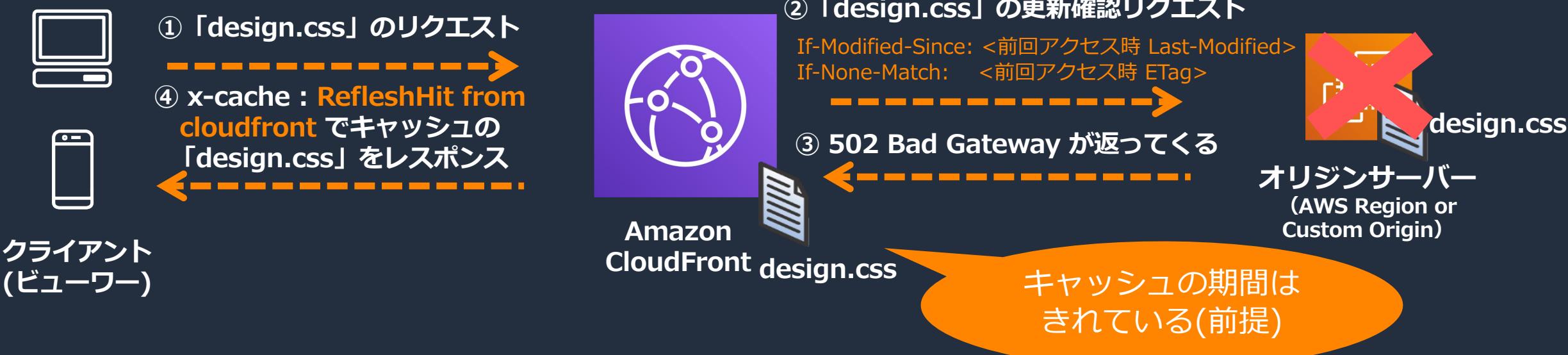
CloudFront がオリジンからの HTTP 4xx および 5xx ステータスコードを処理してキャッシュする方法について

具体的なフロー (キャッシュが CloudFront に既にある場合)

※下記は 5xx Error のフロー なので注意

→ 4xx Error の場合、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ビューワーに返す

“CloudFront にあるコンテンツのキャッシュは期間がきれているが、レスポンスとして返ってくる”
(これを回避するには「Cache-Control: stale-if-error=0」を含めるようにする必要がある)



クライアント
(ビューワー)



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

https://docs.aws.amazon.com/ja_jp/AmazonCloudFront/latest/DeveloperGuide/Expiration.html

© 2023, Amazon Web Services, Inc. or its affiliates.

CloudFront がオリジンからの HTTP 4xx および 5xx ステータスコードを処理してキャッシュする方法について

具体的なフロー (キャッシュが CloudFront に元々無い場合)

※エラーが CloudFront にキャッシュされる
(下記の工程の⑥と⑦は エラーのキャッシュがある間は、502 Bad Gateway を返し続ける)



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

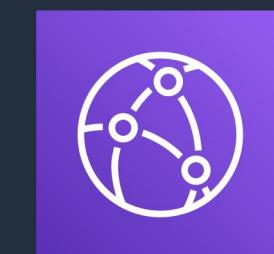
https://docs.aws.amazon.com/ja_jp/AmazonCloudFront/latest/DeveloperGuide/Expiration.html

© 2023, Amazon Web Services, Inc. or its affiliates.

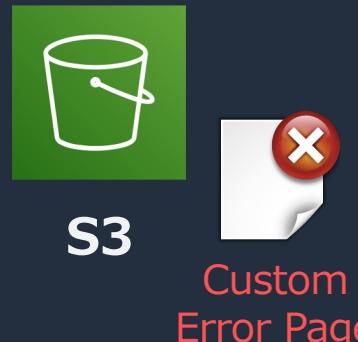
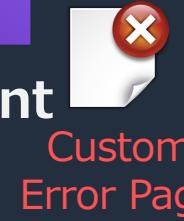
エラーレスポンス動作のカスタマイズ (Custom Error Page)



予め、4xx, 5xx のカスタム
レスポンスを設定(エラーぺ
ージの遷移先を S3 に設定)



CloudFront



- CloudFront は、エラーレスポンスをデフォルト 10 秒キャッシュ
- 4xx および 5xx ステータスコードそれぞれに対して、
 - エラーキャッシュ期間 (エラーキャッシュ最小 TTL) は 0 秒以上を指定可能、レスポンスヘッダーに Cache-Control: max-age または s-maxage や Expires を指定しオブジェクト毎にカスタマイズも可能
 - エラーレスポンスページおよびステータスコードのカスタマイズが可能

キャッシュの挙動を確認するための方法について(ブラウザ編)

x-cache レスポンスヘッダー

- CloudFront がレスポンス返す時に生成してくれるヘッダー
- CloudFront で保持しているキャッシュの状況に合わせて x-cache の内容が変わる

代表的な x-cache の内容

- | | |
|------------------------------|--|
| • Miss From cloudfront | • • • 現在 CloudFront にキャッシュが無い状態なので、オリジンサーバーにコンテンツを取りに行って返した |
| • Hit from cloudfront | • • • CloudFront にキャッシュがあったので、それを返した |
| • RefreshHit from cloudfront | • • • CloudFront にキャッシュはあったが、有効期限が切れたり、オリジンサーバーに最新か確認した後に返した |

とあるコンテンツのレスポンスヘッダ
(ブラウザの開発者ツールより。)

```
▼ Response Headers
accept-ranges: bytes
content-length: 5347
content-type: image/png
date: Tue, 07 Mar 2023 04:49:27 GMT
etag: "33dbdd0177549353eeeb785d02c294af"
last-modified: Sat, 11 Feb 2023 14:30:17 GMT
server: AmazonS3
via: 1.1 18fb8bbcd8ce7c8581681ccc40c56f10.cloudflare.net (CloudFront)
x-amz-cf-id: -1jzFdzrUwgHQfYwMo8CM0vQ0D978PA1eWpkf09E15SZE3I5gacQ2g==
x-amz-cf-pop: NRT57-P3
x-amz-server-side-encryption: AES256
x-cache: Miss from cloudfront
```

キャッシュの挙動を確認するための方法について(ログ編)

x-edge-result-type / x-edge-response-result-type

- HIT

CloudFront がキャッシュからビューワーにオブジェクトを渡したことを示す

- RefreshHit

CloudFront のキャッシュにてオブジェクトを検出したが、キャッシュの有効期限が切れていたため、オリジンに問い合わせて、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たことを示す

- Miss

CloudFront のキャッシュにあるオブジェクトでリクエストに対応できなかったため、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たことを示す

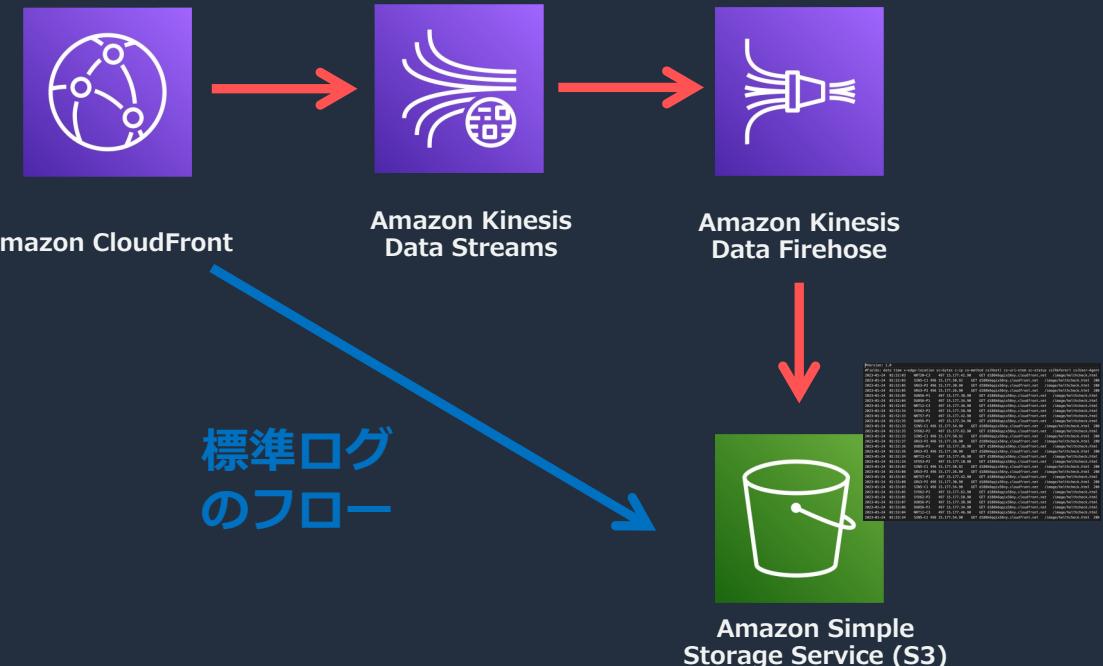
※上記以外にも LimitExceeded、CapacityExceeded、Error、Redirect などの値がある

標準ログ (アクセスログ) の設定および使用

https://docs.aws.amazon.com/ja_jp/AmazonCloudFront/latest/DeveloperGuide/AccessLogs.html



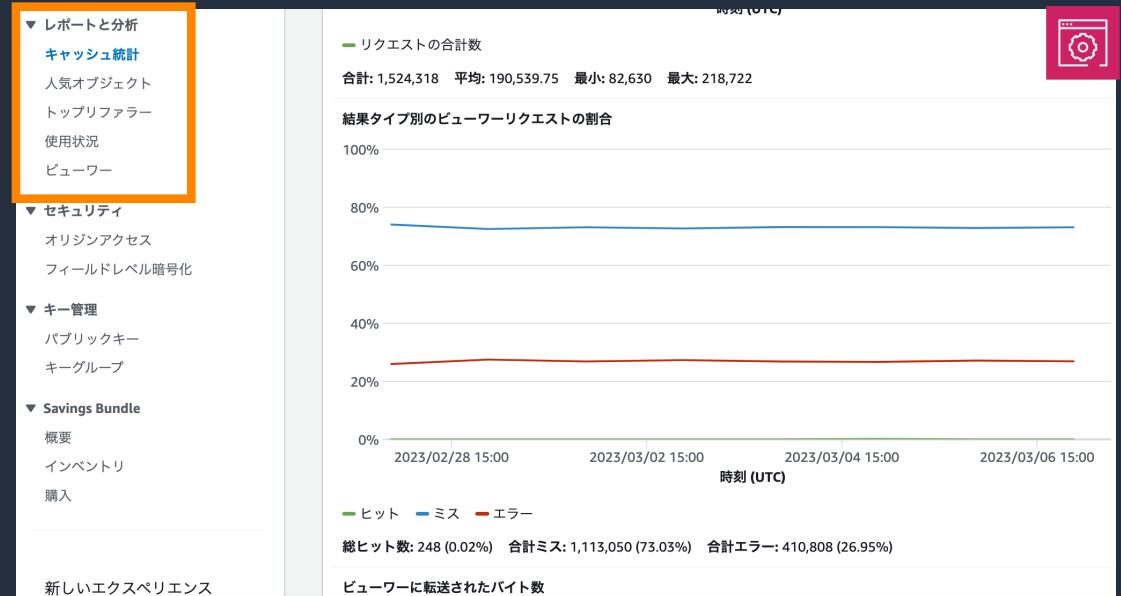
リアルタイムログのフロー



CloudFront レポートと分析(マネージメントコンソール)

マネージメントコンソールにてデフォルトで様々な情報が確認できるようになっている

- キャッシュ統計
 - キャッシュの統計情報
- 人気オブジェクト
 - 人気コンテンツの統計情報
- トップリファラー
 - リファラーの統計情報
- 使用状況
 - リクエスト数およびデータ転送量
- ビューウー
 - クライアントデバイスの統計情報



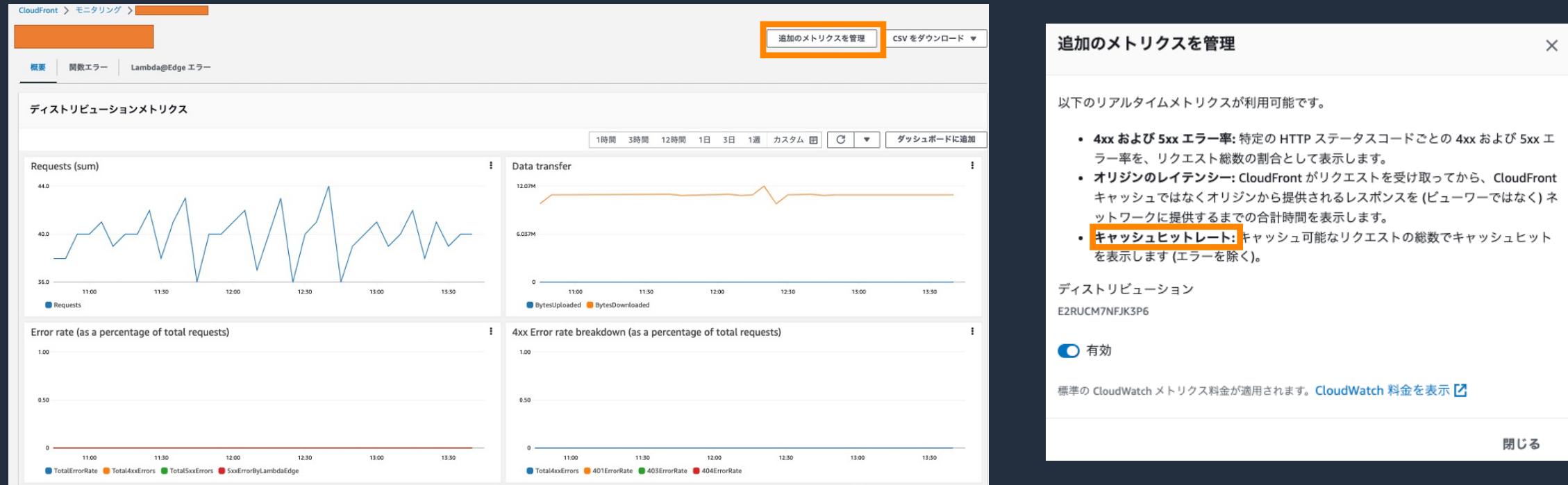
キャッシュ統計 / 人気オブジェクト /
トップリファラー / 使用状況 / ビューウーは
AWS Management Console のみで参照可能

コンソールの CloudFront レポート

https://docs.aws.amazon.com/ja_jp/AmazonCloudFront/latest/DeveloperGuide/reports.html

CloudWatch で管理する Cache hit rate メトリクスについて

デフォルトでは CloudFront のキャッシュヒットレートは CloudWatch のメトリクスとして取得されない（マネージメントコンソールのモニタリングの画面から追加が可能）



CloudFront 関数およびエッジ関数のメトリクスの表示

https://docs.aws.amazon.com/ja_jp/AmazonCloudFront/latest/DeveloperGuide/viewing-cloudfront-metrics.html

クライアント側のキャッシュについて

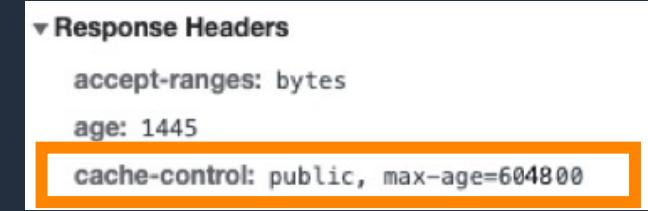
CloudFront の TTL と オリジン側の Cache-Control ヘッダーの関係について

Q. クライアント側にもキャッシュはされる、これはどの期間キャッシュされるのか？

「*.jpg」のパスに対するビヘイビアのキャッシュポリシーの設定



Apache HTTP Server の
Cache-Control ヘッダーの設定



① 「img.jpg」を初めてリクエスト



④ 「img.jpg」をレスポンス

img.jpg

クライアント
(ビューウィー)



これはどれくらいの期間キャッシュされる？



② 「img.jpg」を初めてリクエスト



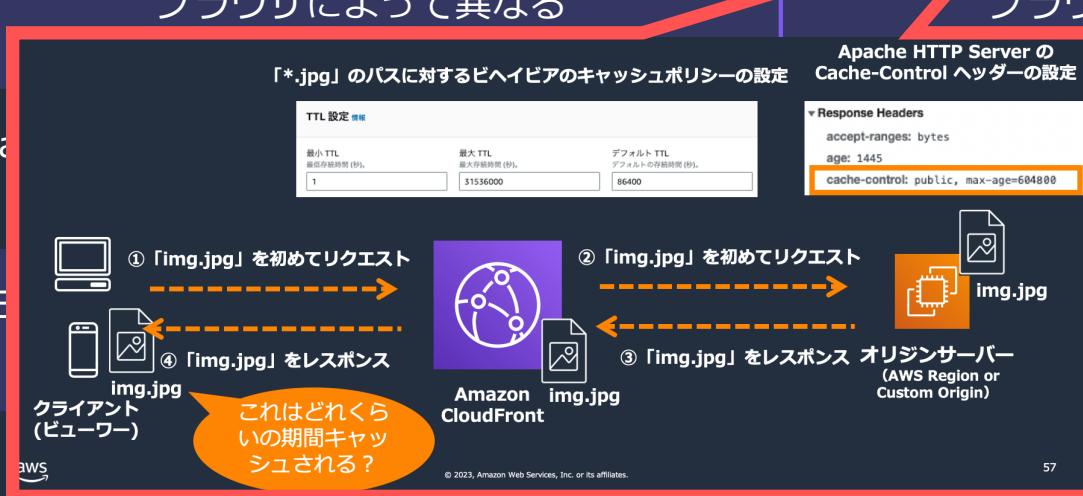
③ 「img.jpg」をレスポンス オリジンサーバー
(AWS Region or Custom Origin)

クライアント側のキャッシュについて



Cache Policy Minimum TTL 設定

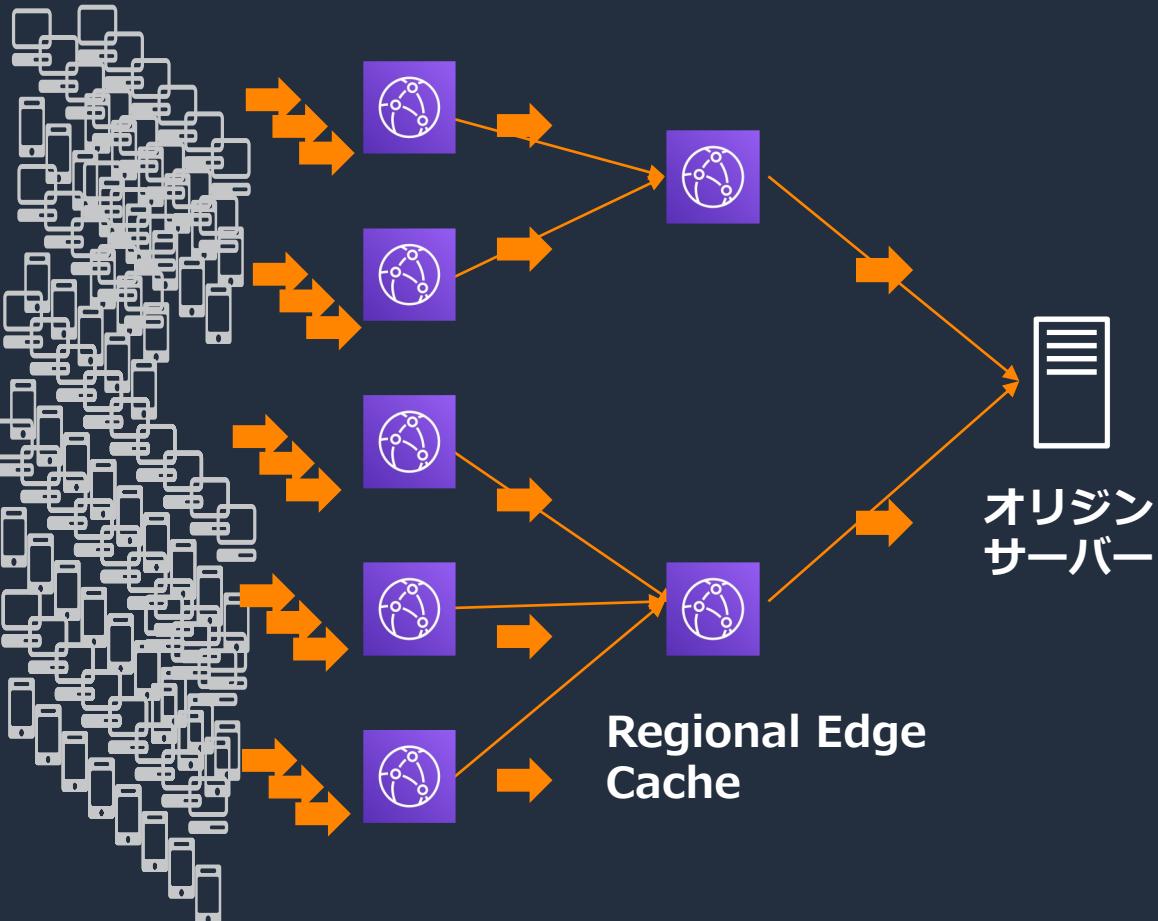
オリジン HTTP ヘッダー	Cache Policy Minimum TTL 設定	
	最小 TTL = 0 秒	最小 TTL > 0 秒を設定
	Cache-Control max-age を指定	Cache-Control: max-age ディレクティブの値に 対応する期間、オブジェクトをキャッシュ
	Cache-Control 設定なし	ブラウザによって異なる
	Cache-Control max-age と s-maxage を指定	Apache HTTP Server の Cache-Control ヘッダーの設定
	Expires を指定	max-age ディレクティブの値 オブジェクトをキャッシュ
Cache-Control no-cache, no-store 、および (または) private ディレクティブを追加	Cache	max-age ディレクティブの値 オブジェクトをキャッシュ



その他

オリジンインフラの保護

Automatic Flash Crowd Protection

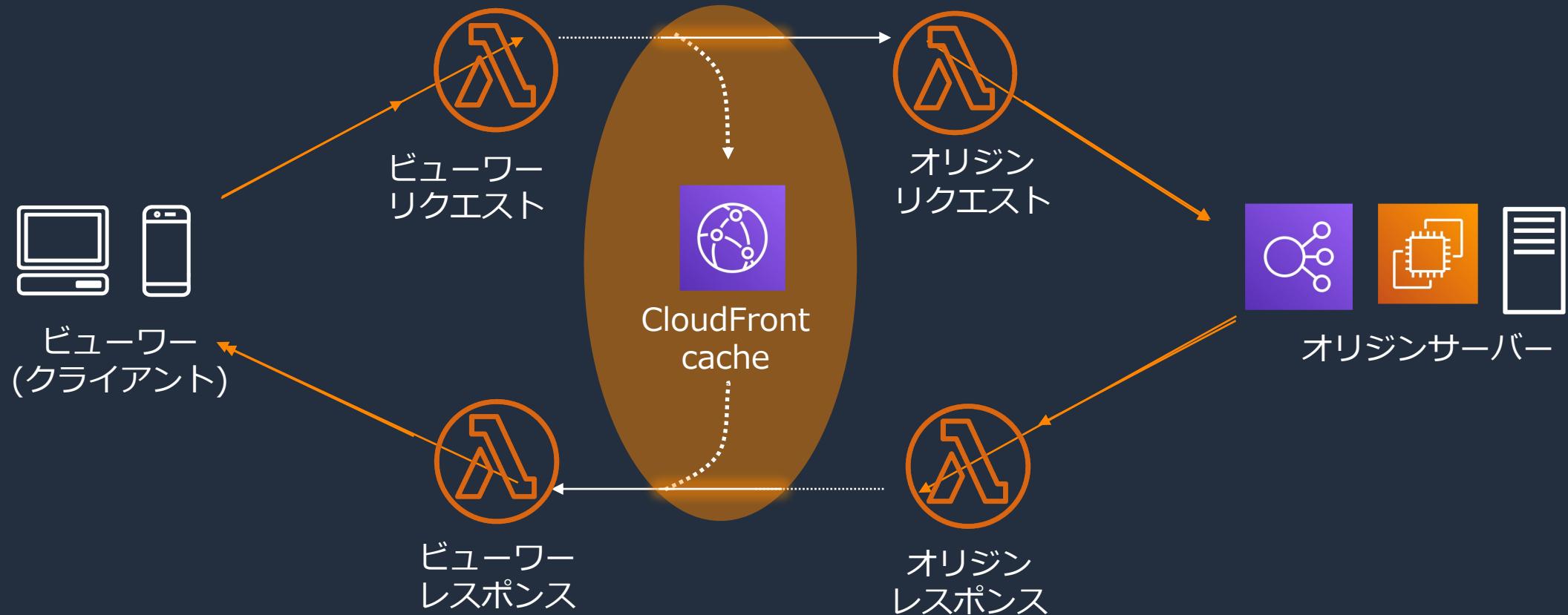


同一オブジェクトへの同時リクエスト (リクエストを折りたたむ)

https://docs.aws.amazon.com/ja_jp/AmazonCloudFront/latest/DeveloperGuide/RequestAndResponseBehaviorS3Origin.html

CloudFront のキャッシュヒットした時の Lambda@Edge の挙動

Q. CloudFront のキャッシュがヒットした際に、Lambda@Edge は実行されるのか？



Lambda 関数をトリガーできる CloudFront イベント

https://docs.aws.amazon.com/ja_jp/AmazonCloudFront/latest/DeveloperGuide/lambda-cloudfront-trigger-events.html

キャッシングに関する CloudFront のクオータ

キャッシングに関する CloudFront のクオータは、主にキャッシングポリシーに関するものである(下記抜粋)

- ・作成できるキャッシングポリシーの数
- ・キャッシングポリシーあたりのクエリ文字列
- ・キャッシングポリシーあたりのヘッダー
- ・キャッシングポリシーあたりの Cookie

など

ポリシーの一般的なクオータ	
	デフォルトのクオータ
エンティティ	20
AWS アカウントあたりのキャッシングポリシー	100
同じキャッシングポリシーに関連付けられたディストリビューション	10
キャッシングポリシーあたりのクエリ文字列	クォータ引き上げのリクエスト
キャッシングポリシーあたりのヘッダー	10
キャッシングポリシーあたりの Cookie	クォータ引き上げのリクエスト
キャッシング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
オリジンリクエストポリシーあたりのヘッダー	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	クォータ引き上げのリクエスト
キャッシング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
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
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
AWS アカウントあたりの継続的デプロイポリシー	クォータ引き上げのリクエスト
	クォータ引き上げのリクエスト



クオータ

https://docs.aws.amazon.com/ja_jp/AmazonCloudFront/latest/DeveloperGuide/cloudfront-limits.html

© 2023, Amazon Web Services, Inc. or its affiliates.

参考 URL

メディア配信におけるキャッシュの活用方法

- ・ ビデオオンデマンド (VOD) を配信
https://docs.aws.amazon.com/ja_jp/AmazonCloudFront/latest/DeveloperGuide/on-demand-video.html
- ・ ライブストリーミングビデオの配信
https://docs.aws.amazon.com/ja_jp/AmazonCloudFront/latest/DeveloperGuide/live-streaming.html

AWS Hands-on for Beginners

- ・ AWS 上で静的な Web サイトを公開しよう !
https://pages.awscloud.com/JAPAN-event-OE-Hands-on-for-Beginners-CF_WAF-2022-reg-event.html
- ・ Amazon CloudFrontおよびAWS WAFを用いて エッジサービスの活用方法を学ぼう
https://pages.awscloud.com/JAPAN-event-OE-Hands-on-for-Beginners-CF_WAF-2022-reg-event.html

まとめ

- ✓ CloudFront は AWS のグローバルレインフラストラクチャを利用した CDN (Content Delivery Network) サービス
- ✓ キャッシュの活用は CloudFront を利用するメリットの一つ
- ✓ Web サイトのコンテンツには沢山の種類が、あるがその殆どが静的コンテンツ
- ✓ コンテンツキャッシュの運用は、コンテンツ毎に誰がいつどうやって行うのかを決めた上で検討する
- ✓ CloudFront でキャッシュを活用する際には Cache Policy で設定し、オリジン側の Cache Control の設定もキャッシュ期間に影響する
- ✓ エラーレスポンスについても CloudFront にキャッシュをさせることができる

本資料に関するお問い合わせ・ご感想

技術的な内容に関しては、有料の AWS サポート窓口へ
お問い合わせください

<https://aws.amazon.com/jp/premiumsupport/>

料金面でのお問い合わせに関しては、カスタマーサポート窓口へ
お問い合わせください（マネジメントコンソールへのログインが必要です）

<https://console.aws.amazon.com/support/home#/case/create?issueType=customer-service>

具体的な案件に対する構成相談は、後述する個別相談会をご活用ください



ご感想は Twitter へ！ハッシュタグは以下をご利用ください
#awsblackbelt



その他コンテンツのご紹介

ウェビナーなど、AWS のイベントスケジュールをご参照いただけます

<https://aws.amazon.com/jp/events/>

ハンズオンコンテンツ

<https://aws.amazon.com/jp/aws-jp-introduction/aws-jp-webinar-hands-on/>

AWS 個別相談会

AWS のソリューションアーキテクトと直接会話いただけます

<https://pages.awscloud.com/JAPAN-event-SP-Weekly-Sales-Consulting-Seminar-2021-reg-event.html>



Thank you!