オートスケールするライブ配信システム powered by Google Cloud

株式会社 Castify
代表取締役社長
高橋 健太

Google Cloud

セッション概要

Castify は、ライブ配信に必要なコンポーネント一式を提供する SaaS プラットフォームです。

本セッションでは Kubernetes ネイティブに設計された私達のライブ配信システムが従来のものとどう違うのか、また Kubernetes Engine (GKE) をベースとすることで得られた優位性について説明します。

Google Cloud

Agenda

- Castify システム概要
- メディア配信システムの基本要素
- 配信系のスケーラビリティと可用性確保の工夫
  - 概要
  - Origin - Edge
  - Transcoder

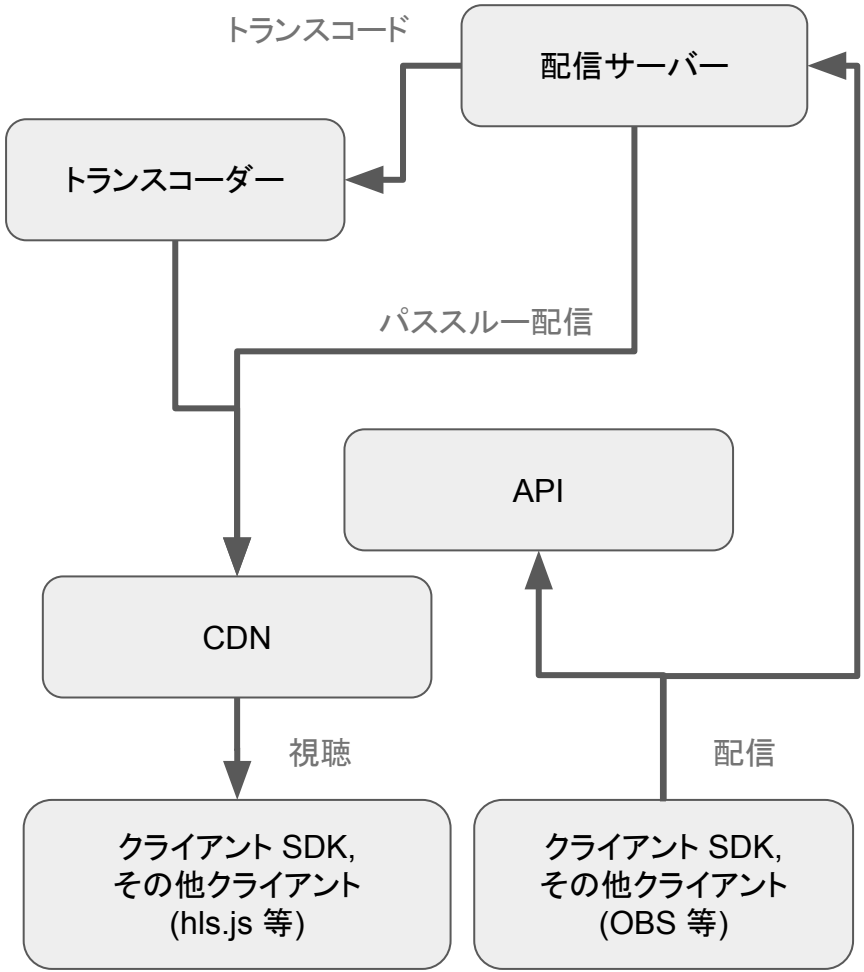
Castify システム概要

Castify 概要

ライブ配信を行うために必要な仕組み一式を提供する SaaS プラットフォーム

提供している機能

1. オートスケールする配信サーバー, トランスコーダー
2. 1. を制御するための API サーバー
3. 視聴／配信用クライアント SDK
    - 現在は iOS, Android 向けを提供



Google Cloud

Castify のゴール

ライブ配信に関する技術的な側面を Castify が完全に代行することで、利用者がライブ配信機能を持ったサービスやアプリを簡単に作成できるようにする。

Castify で利用している GCP の機能

メイン

- **Kubernetes Engine**

ストレージ

- Cloud SQL
- Cloud Storage

監視／分析など

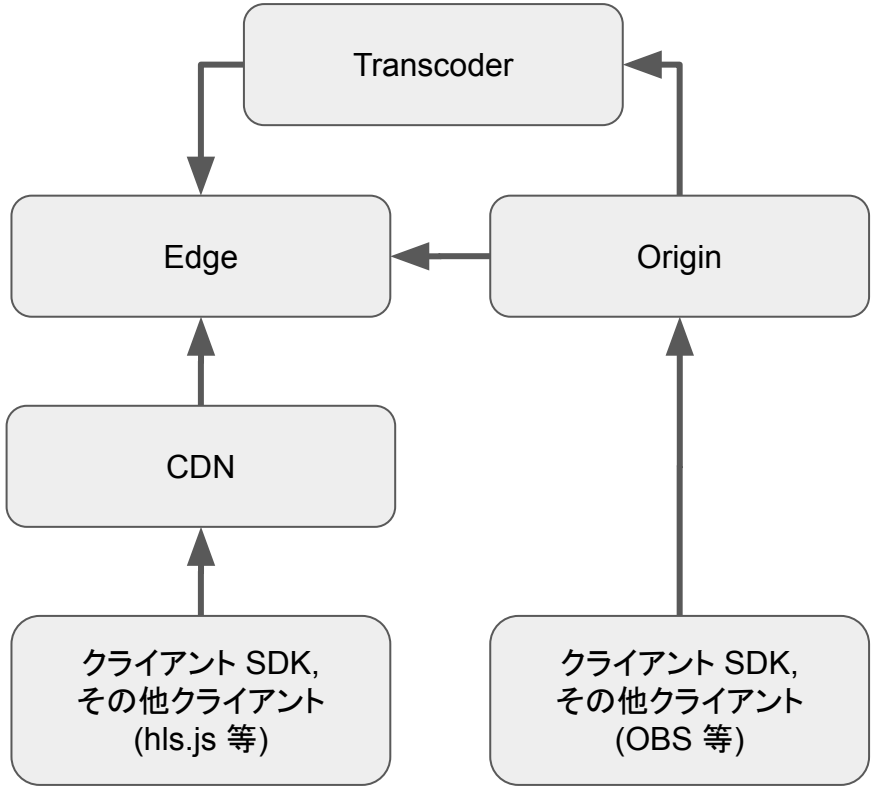
- BigQuery
- Cloud Logging
- Cloud Monitoring

Google Cloud

メディア配信システムの基本要素

メディア配信システムの構成要素

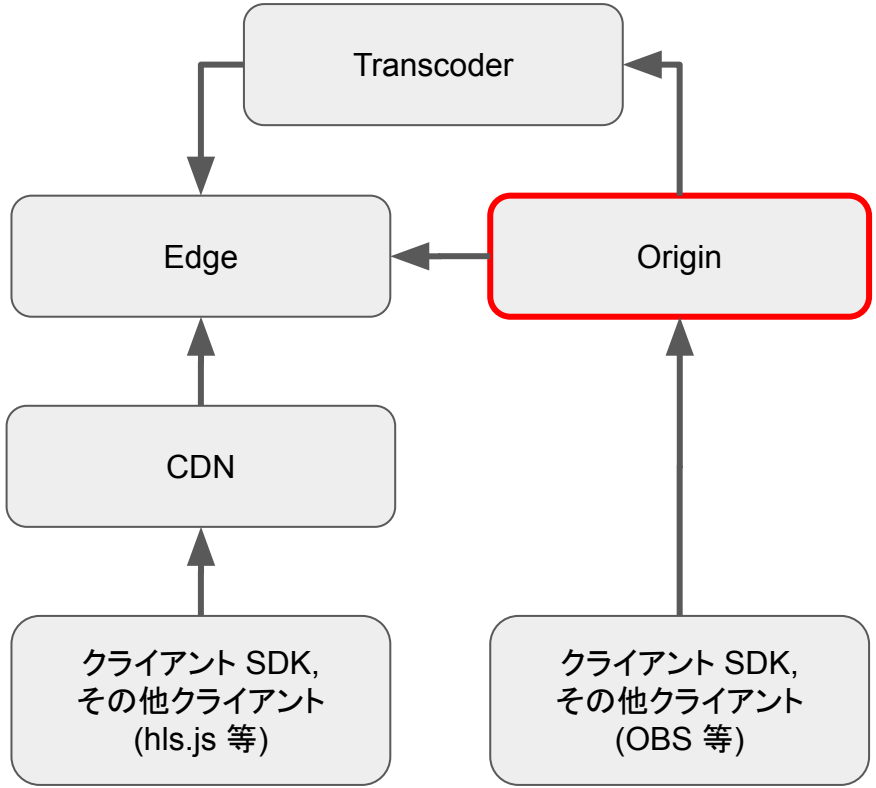
- Origin
  - メディアの受信
- Edge
  - メディアの配信
- Transcoder
  - メディアのトランスコード



Origin

メディアの受信を行うサービス

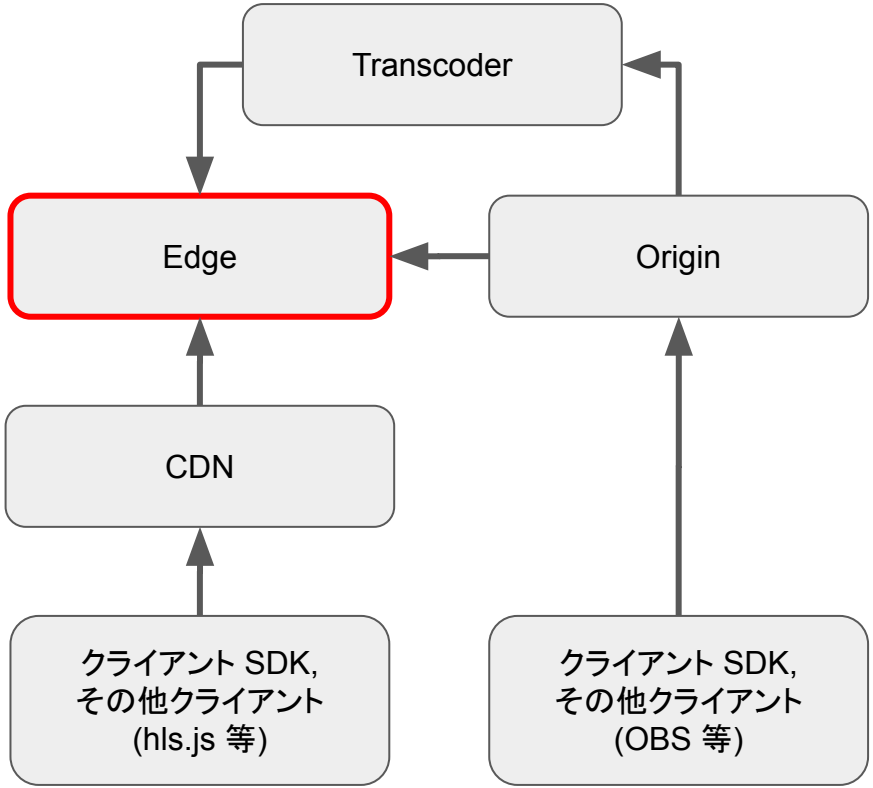
- 受信したメディアをローカルディスクに録画
- 対応プロトコル
  - RTMP
  - WebRTC
- GCP External TCP/UDP LB で負荷分散
- K8s 上は Service+Deployment



Edge

視聴者に対して HLS によるメディア配信の配信を行うサービス

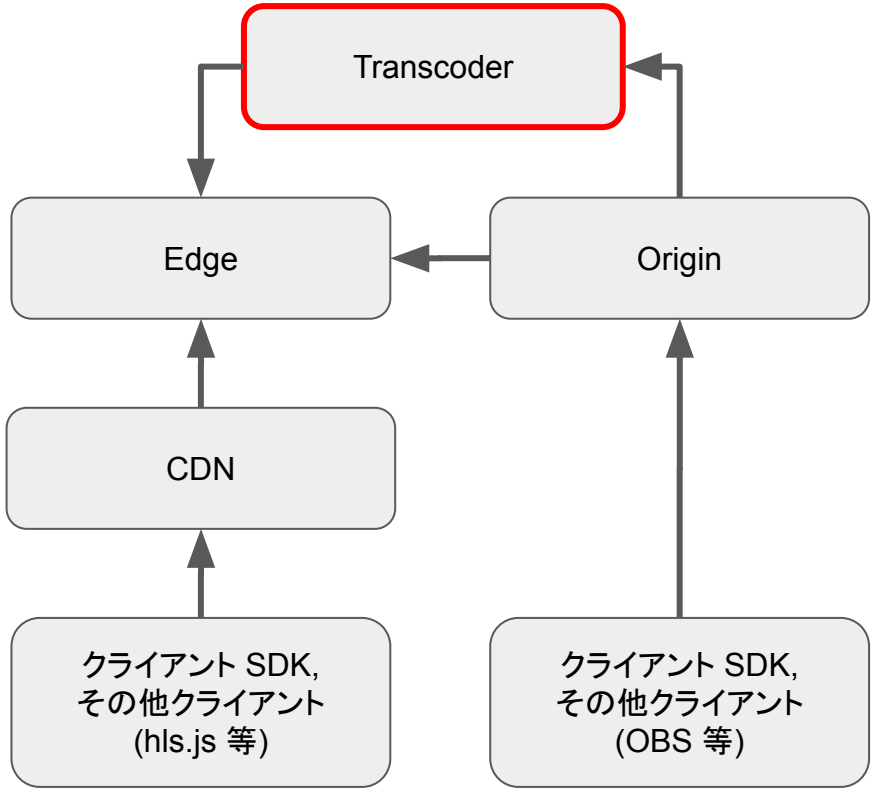
- 対応プロトコル
  - HTTP/HTTPS (HLS)
- GCP External HTTP(S) LB で負荷分散
  - メディア配信は Cloud CDN
- K8s 上は Service+Deployment**+HPA**
  - GKE のオートスケール設定も ON



Transcoder

メディアのサービス（再エンコード）を行うコンポーネント

- K8s 上の表現は DaemonSet
  - Deployment ではなく DaemonSet である理由は後述



Transcoder: 種別

Castify には以下の二種類のトランスコーダーが存在する。

1. Push 型トランスコーダー (現在メインで使用)
    - Origin が直接 Transcoder に接続し、受信メディアの再配信を行う
        - 変換されたメディアの録画はTranscoder 自体が行う
2. Pull 型トランスコーダー
    - メディアを要求されたときにEdge 同様に上流からメディアを受信→変換
        - 変換されたメディアはキャッシュのみで録画は行わない
    - 不要な変換は行わないため低コスト
        - ライブ配信にのみに利用可能

配信系のスケーラビリティと可用性の工夫：概要

Castify のスケーラビリティ・可用性

基本的にどのコンポーネントも好きなタイミングで落とせるべき

- できる限りステートレスにする
  - Edge は単なる HTTP サーバーのため落ちても問題ない
  - Origin, Transcoder は接続先が落ちても、再接続すれば別のOrigin 経由で配信は維持可 (次ページで解説)

ダウンタイムなしにデプロイやリスケールが可能

Castify のスケーラビリティ・可用性

Origin と Transcoder はディスク上にメディアの録画をしているため完全なステートレスではないが、以下のような工夫をすることで耐障害を行っている。

- ディスクに書かれるメディアは壊れにくい設計にしており修復も容易
  - パケットベース
  - 追記のみ
- 録画メディアは定期的に回収されCloud Storage にアップロードされる
  - 配信サーバーとは独立したDaemonSet がノード単位で回収処理

Castify のスケーラビリティ・可用性

Origin では GKE のオートスケール機能は OFF にしている。理由は..

1. ローカルディスクに受信メディアの録画を行うため、メディアの回収前にダウンスケールされると問題
2. その他のコンポーネントと違い急激に負荷が増えないため、ある程度余裕をもたせておけば問題ない

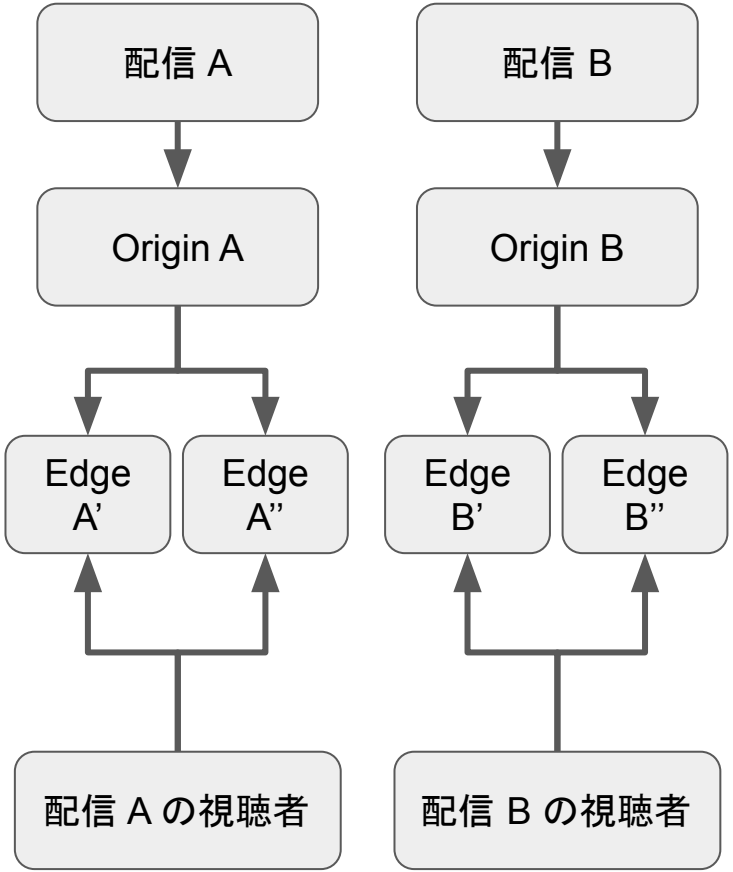
オートスケールこそしないが手動でノード数を増やすのはon-the-fly で可能

配信系のスケーラビリティと可用性の工夫 : Origin - Edge 構成

Origin - Edge の構成: 一般的な配信サーバー

よくあるメディア配信サーバーの構成

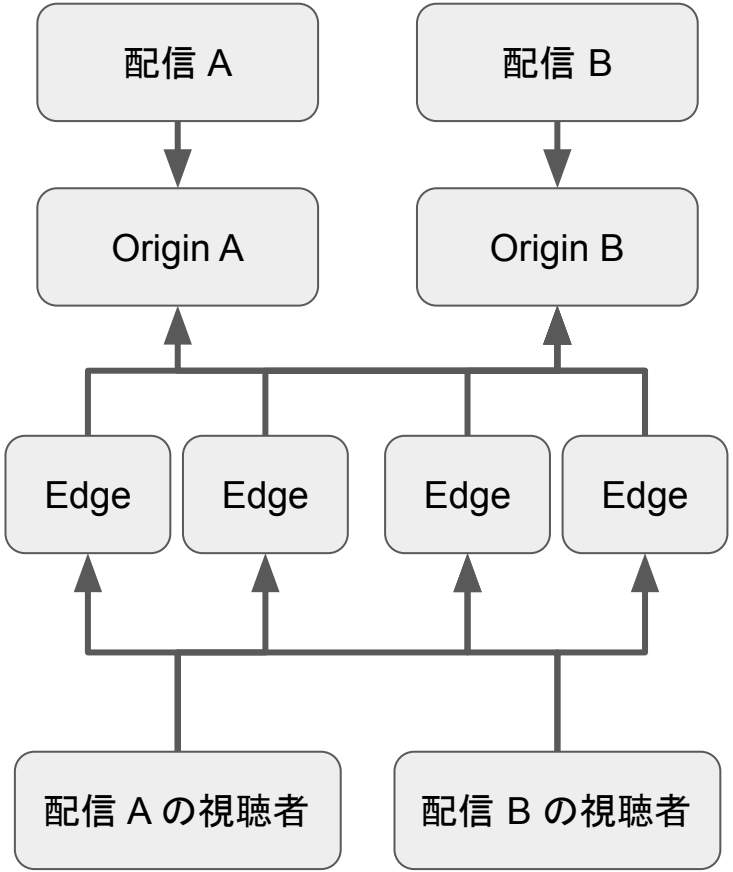
- それぞれの Origin に固定の Edge が複数ぶらさがる形
- 視聴クライアントは、事前にどの Edge にアクセスすべきかを知る必要がある
  - 特殊なプロクシーでの振り分けや、アクセス先の Edge に正確に対応したホスト情報を含む視聴 URL の事前取得など



Origin - Edge の構成: Castify

Castify におけるメディア配信サーバーの構成

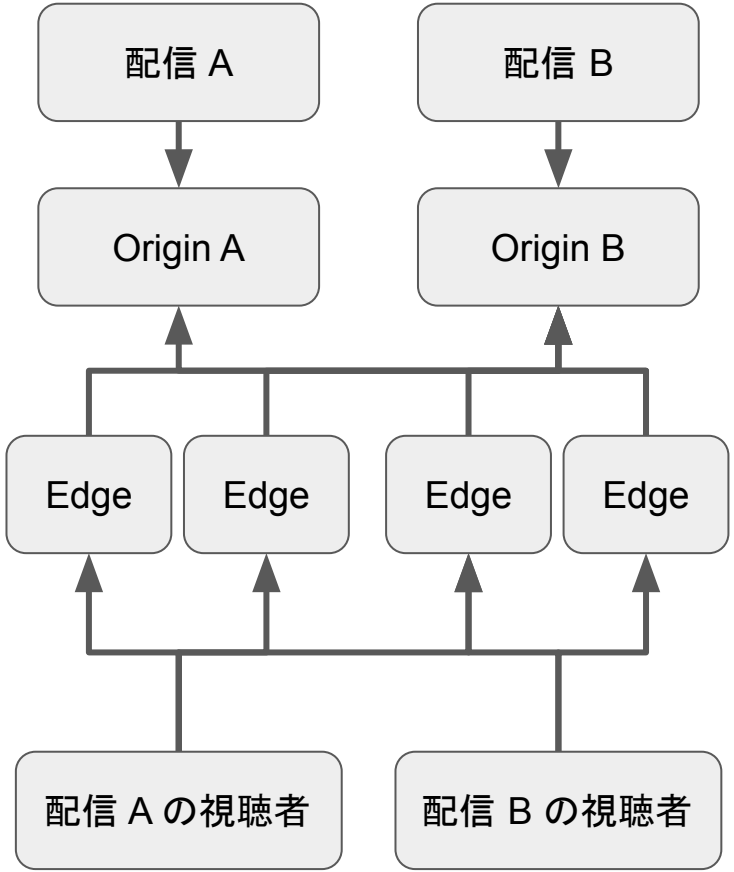
- Edge は全て同じ能力を持つ
  - Origin の指定などを Edge に持たない
  - このため、単純に CDN を経由させることができる



Origin - Edge の構成の工夫: メリット

ステートレスにすることでK8s との相性が良い

1. ある配信の視聴の際に、特定のOrigin に所属する Edge を直前まで知らなくて良い
   - 従来の構成では、ある配信に所属する Edge を事前に知る必要がある
   - Castify では Edge はメディアの所在を API サーバー経由で解決する
2. Edge クラスター全体を K8s Service 化
   - LB による負荷分散を有効に活用でき、特定の配信に属するEdge にアクセスが集中する問題が起きない

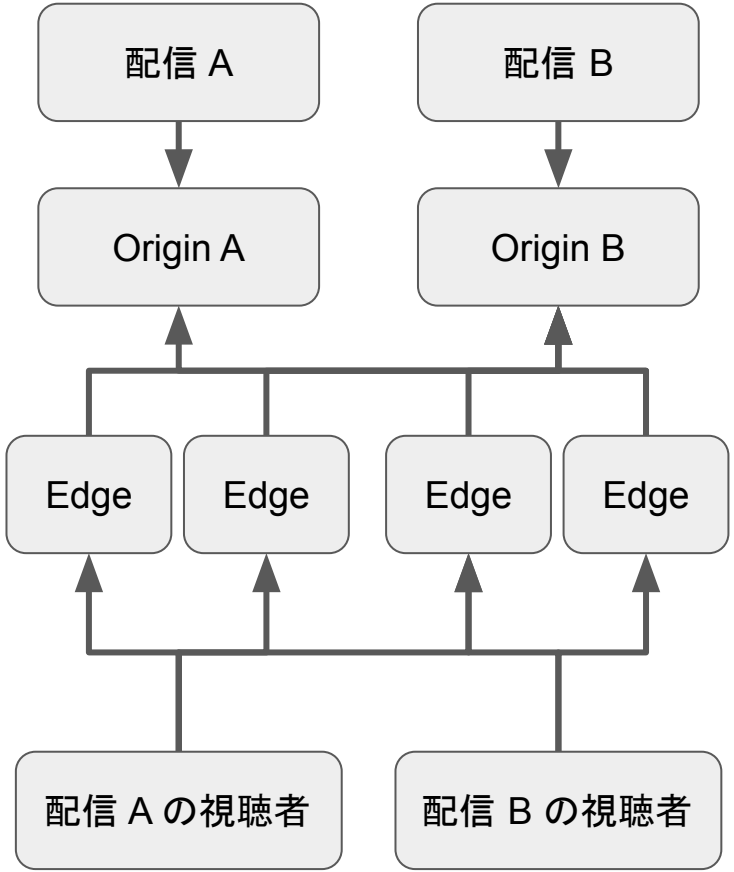


Google Cloud

Origin - Edge の構成の工夫: メリット

この構成による追加のメリット

1. Edge の Pod 数が増加した状況に対応して多階層のレプリケーション構成を行うことでOriginへの負荷を抑えられる
2. 視聴クライアントからEdge に対して過去の時間を指定されたとしても、API サーバーからその時間を保持しているOrigin を指定すれば良い
    a. Castify ではライブ配信中に視聴者がライブ映像を任意に巻き戻して視聴する「タイムシフト機能」を、ライブ視聴と同一の仕組みで提供できている



配信系のスケーラビリティと可用性の工夫 : Transcoder

Transcoder のスケール (1)

基本的には Origin と同じだが、以下のような点が異なる

- 配信一つに対して大きな負荷がかかる
- インスタンスの最低要求スペックが高い
  - Origin は g1-small でかなりの配信数をさばけるがTranscoder は NG

Transcoder のスケール (2)

#1 配信一つに対して大きな負荷がかかる

- 高解像度のエンコードには非常に大きなCPU 負荷がかかる
  - 処理の振り分け先となるPod を適切に選択しないとパンクしてしまう
  - K8s ネイティブの Service によるリクエストの振り分けの仕組みは利用しない

- API サーバーによって以下のような調停を経て適切なPod に処理が振り分ける
  a. エンコード設定の内容から負荷量の見積もりを行う
  b. その負荷を十分に捌けるPod を Transcoder クラスターの中から選択

Transcoder のスケール (3)

#2 インスタンスの要求スペックが高い

自動ダウンスケールによる録画メディアのロストを防ぐためにTranscoder のオートスケールには GKE のオートスケールの仕組みは利用せず、以下のようにGKE の API 経由でノード数を増やす工夫をしている。

1. バッチで定期的にTranscoder クラスター全体の負荷を見積もる
2. 負荷が閾値を超えたらGKE API で Transcoder のノードプールを追加
    - Transcoder は DaemonSet なのでノード増加に応じてPod が増える

Google Cloud

(補足) 既存のノードプールのリサイズではなく**ノードプール自体の追加**を行う理由は、ダウンスケール時にダウンスケールの対象となるノードを正確に把握できるようにするため。これができないと、未回収の録画メディアが残っているノードをダウンスケール対象外にできない。

まとめ

- ライブ配信 SaaS である Castify の概要
- ライブ配信サービスの基本要素
  - Origin, Edge, Transcoder
- 配信系を構成する各要素のスケーラビリティ・可用性の確保の考え方
  - 個々のコンポーネントを Kubernetes ネイティブに設計することで、無理なくスケールできるようになり、可用性も高められた

Thank you