AWS Dev Day Online Japan

JAWS のイベントを支える

AWS Amplify とサーバーレス

2021 9/28 松井英俊

自己紹介

松井 英俊
1990 年 4 月 24 日 生まれ
静岡県立浜松工業高等学校 情報技術科 卒業
製造業、建築土木、スタートアップなど複数業種・職種経験
2017 年ごろより Web 系の開発に携わる
株式会社スタートアップテクノロジー | テックリード
2021 年 6 月 より AWS Serverless Hero

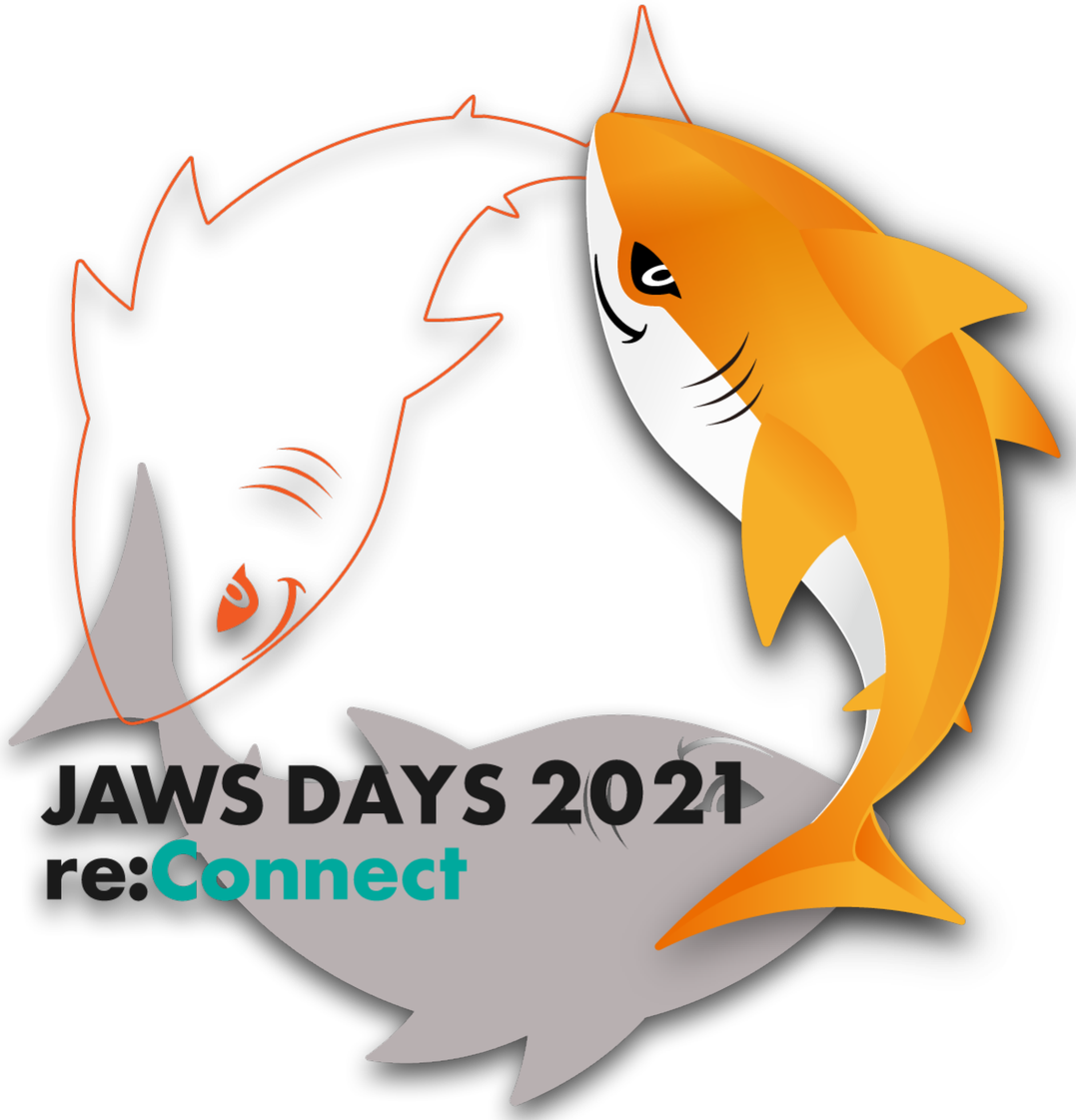
StartupTechnology

JAWS UG との関わり

- 色々あって東京からUターン
- 就職活動がきっかけで地元の技術コミュニティに参加する様になり、JAWS UGと出会う
- ほぼ月1開催の勉強会で何か作って発表するのが習慣に
- builders.flash への記事のオファーをもらい、定期的に寄稿
- コロナウイルス対策のシビックテックに参画
- JAWS DAYS 2021 - re:Connect 実行員に
- AWS Serverless Hero に選ばれる

JAWS DAYS 2021 - re:Connect

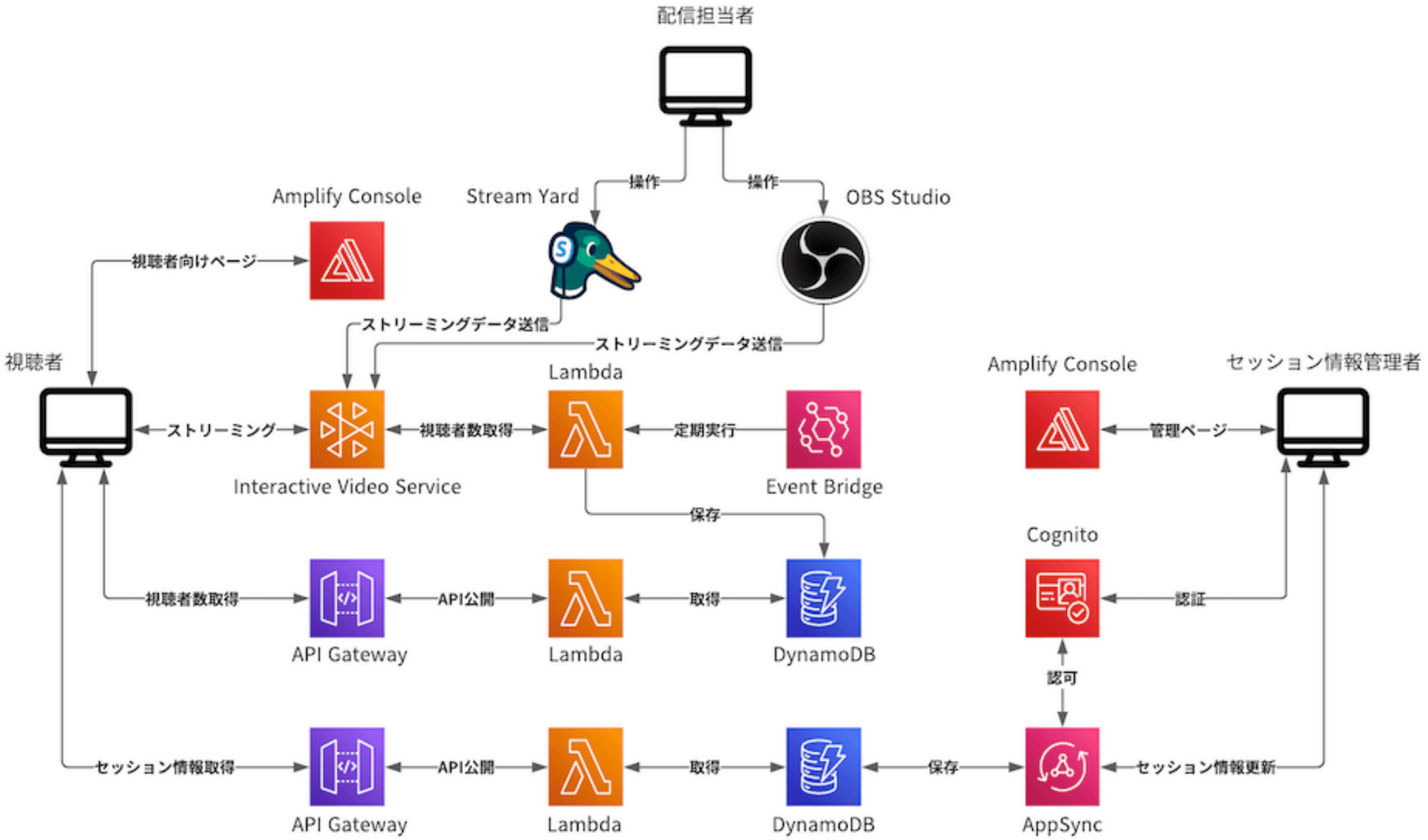
2021 年 3 月 開催
- JAWS-UG 年間を通して最大のイベント
- 過去最高の4000人近い参加申し込み
- コミュニティイベントとして国内最大級
- イベントサイト、配信基盤等は外部委託せず
  全てコミュニティメンバーが構築

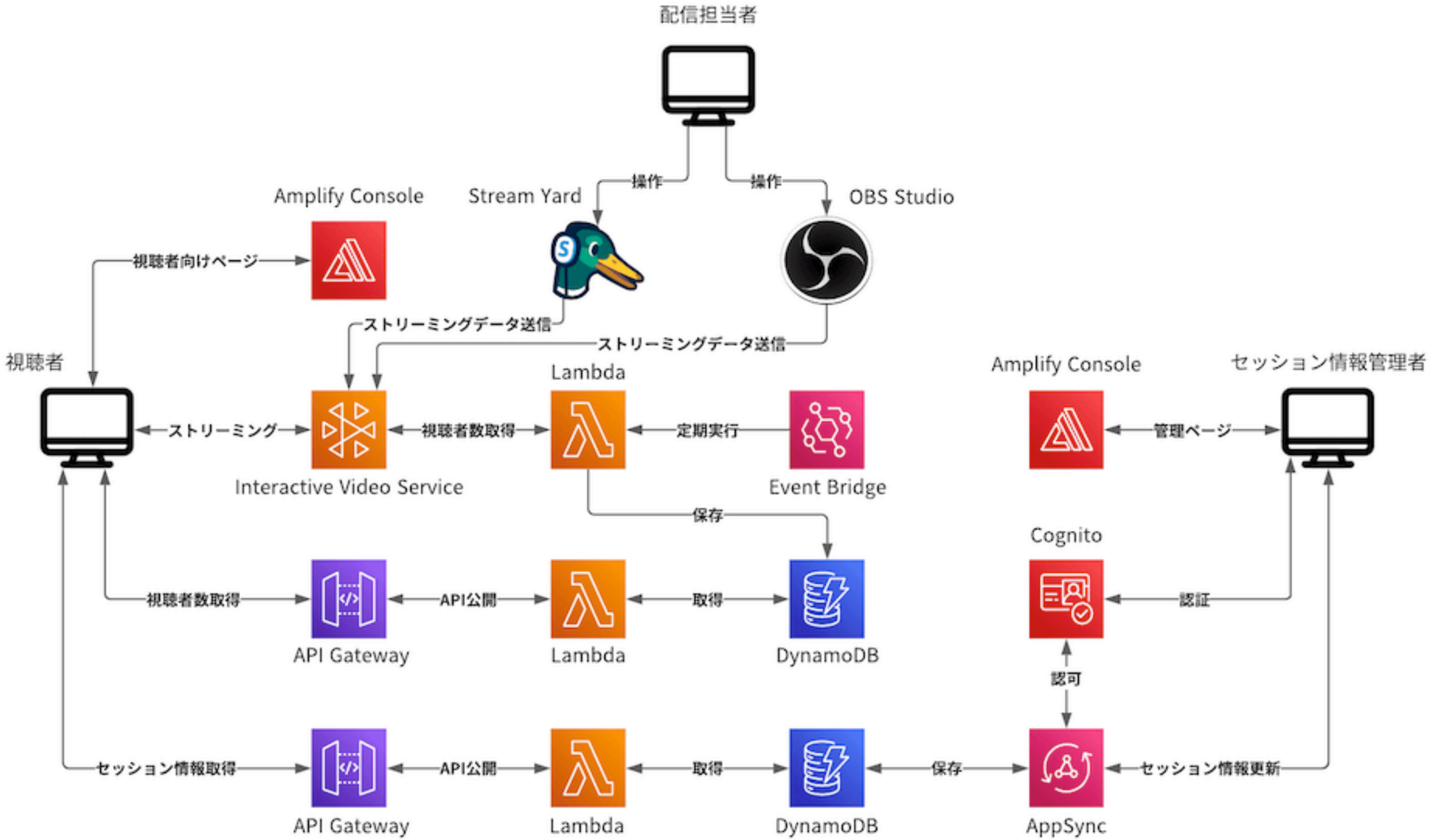


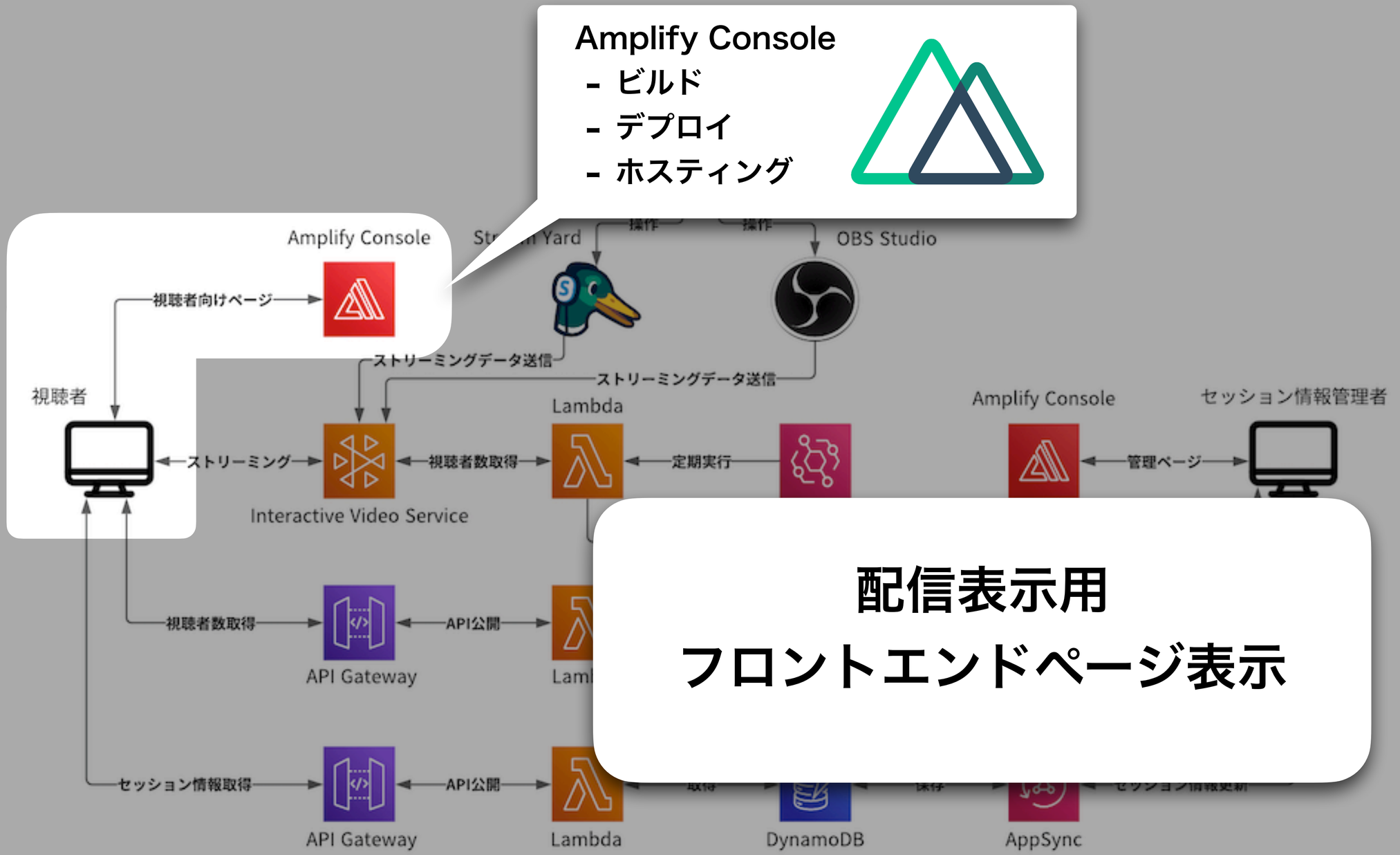
配信サイトの機能要件

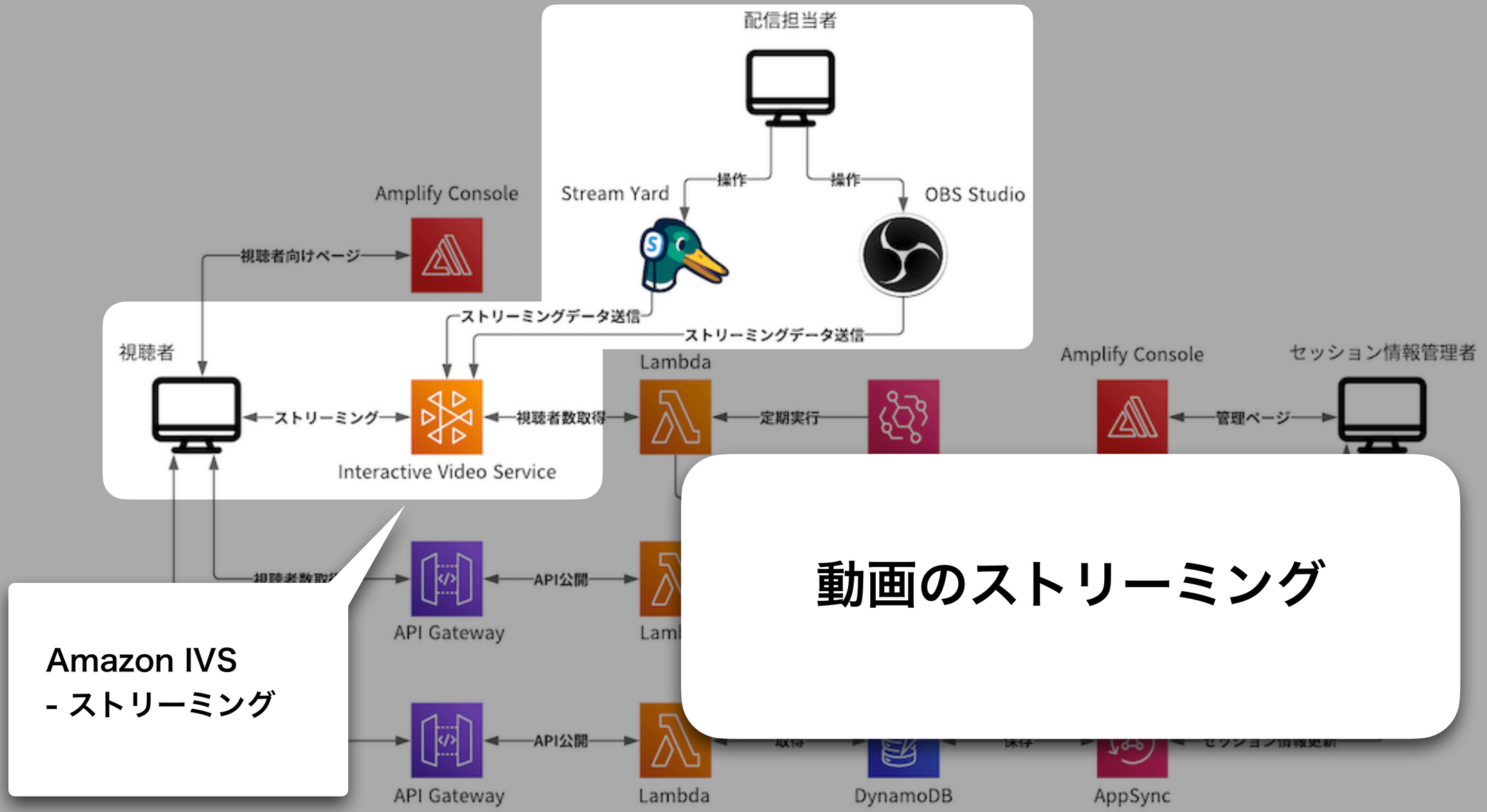
- Amazon Interactive Video Service からの動画ストリーミングを表示
- セッションタイトルや説明文など編集してリアルタイムに表示
- 配信視聴者数をリアルタイムに表示

配信アーキテクチャ ver 1.0

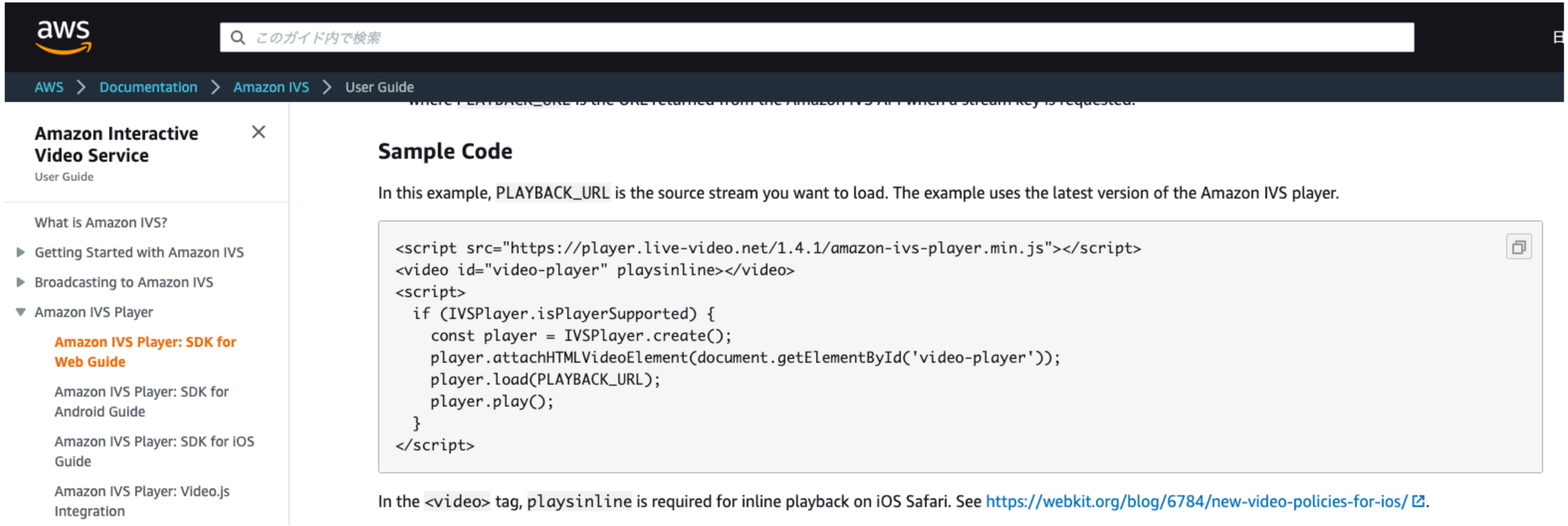



配信担当者
操作
操作
Amplify Console
Stream Yard
OBS Studio
視聴者向けページ
ストリーミングデータ送信
ストリーミングデータ送信
視聴者
Lambda
Amplify Console
セッション情報管理者
ストリーミング
視聴者数取得
定期実行
管理ページ
Interactive Video Service
Event Bridge
保存
Cognito
視聴者数取得
API公開
取得
認証
API Gateway
Lambda
DynamoDB
認可
セッション情報取得
API公開
取得
保存
セッション情報更新
API Gateway
Lambda
DynamoDB
AppSync


Amplify Console
- ビルド
- デプロイ
- ホスティング
Amplify Console
視聴者向けページ
操作
操作
OBS Studio
ストリーミングデータ送信
ストリーミングデータ送信
視聴者
Lambda
Amplify Console
セッション情報管理者
ストリーミング
視聴者数取得
定期実行
管理ページ
Interactive Video Service
視聴者数取得
API公開
API Gateway
セッション情報取得
API公開
API Gateway
Lambda
DynamoDB
AppSync
配信表示用
フロントエンドページ表示


配信担当者
操作
操作
Stream Yard
OBS Studio
Amplify Console
視聴者向けページ
ストリーミングデータ送信
ストリーミングデータ送信
視聴者
ストリーミング
視聴者数取得
Interactive Video Service
Lambda
定期実行
Amplify Console
管理ページ
セッション情報管理者
API公開
API Gateway
API公開
API Gateway
Lambda
DynamoDB
AppSync
Amazon IVS
- ストリーミング
動画のストリーミング

公式サンプル



→ NPM で依存関係が解決できない…

解決策1

```
23  +  methods: {
24  +    startStream (stream) {
25  +      const script = document.createElement('script')
26  +      script.innerHTML = `
27  +        var player = IVSPlayer.create()
28  +        player.attachHTMLVideoElement(document.getElementById('video-player-${stream.name}'))
29  +        player.load("${stream.url}")
30  +        player.play()
31  +      `
32  +      document.body.appendChild(script)
33  +    }
34  +  }
```

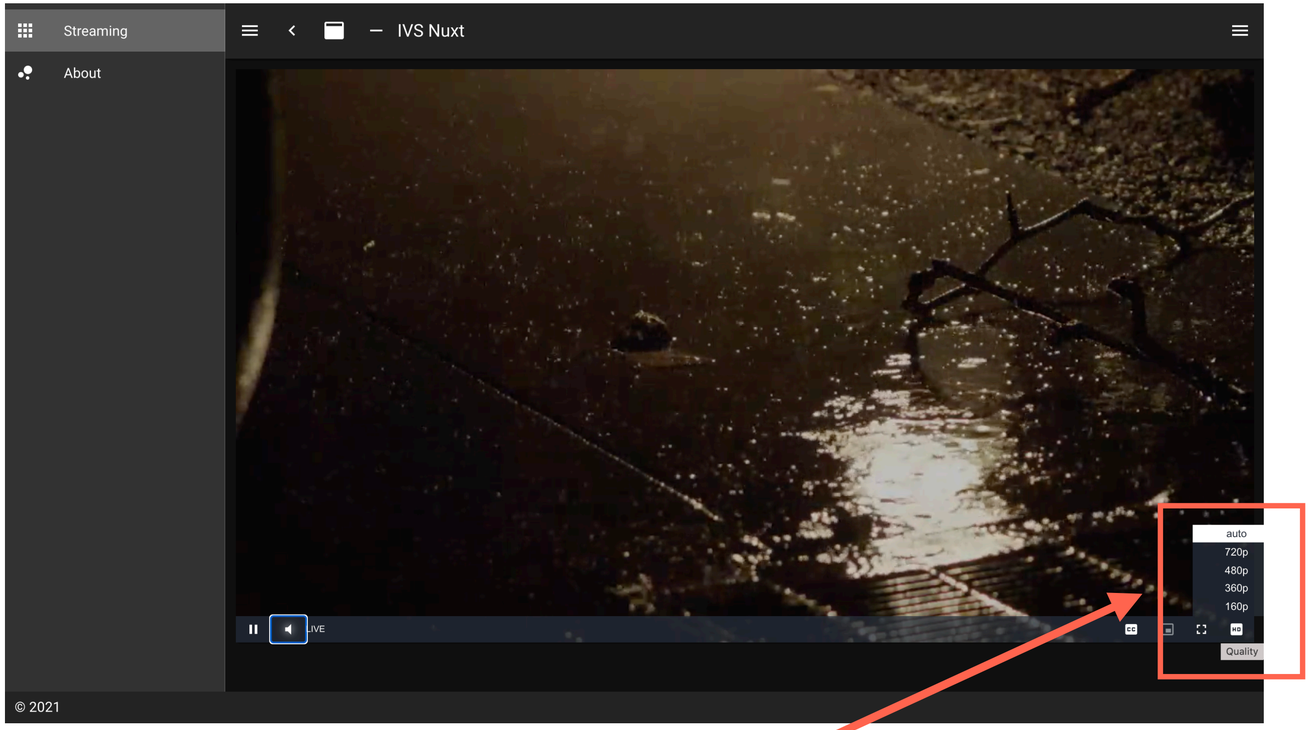
公式のコード

DOM 操作で直接貼り付ける！👊

→ 動くけどダサい…

ABR(Adaptive Bit Rate)追加

```
const script = document.createElement('script')
script.innerHTML = `
  if (typeof player === 'undefined') {
    const player = IVSPlayer.create()
    player.attachHTMLVideoElement(document.getElementById('video-player-${stream.id}'))
    player.load("${stream.url}")
    player.play()
    registerIVSTech(videojs)
    registerIVSQualityPlugin(videojs)
    const player = videojs('video-player-${this.timestamp}', {
        techOrder: ["AmazonIVS"]
    })
    player.enableIVSQualityPlugin()
    player.src("${stream.url}")
  }
`
document.body.appendChild(script)
```



クライアント側でビットレート選択可能に！

→ 動くけど依然ダサい…

解決策2

救世主: Altissie LLC 代表 鈴木 直人さん

- Web や モバイル App の開発に長年携わる
- フロントエンド ゴリゴリマン
- 松井が同じコワーキングスペースで知り合う

解決策2

```
113 +     plugins: [
114 +       new CopyPlugin({
115 +         patterns: [
116 +           {
117 +             from: 'node_modules/amazon-ivs-player/dist/assets/amazon-ivs-wasmworker.min.*',
118 +             to: '[name].[ext]'
119 +           }
120 +         ]
121 +       })
122 +     ]
```

ビルド時に含まれないファイルを
CopyWebpackPlugin でコピー

```
105 +     if (ivs === null) {
106 +       ivs = require('amazon-ivs-player')
107 +       ivs.registerIVSTech(videojs, {
108 +         wasmBinary: '/_nuxt/amazon-ivs-wasmworker.min.wasm',
109 +         wasmWorker: '/_nuxt/amazon-ivs-wasmworker.min.js'
110 +       })
111 +       ivs.registerIVSQualityPlugin(videojs)
112 +     }
113 +
```

依存ファイルをちゃんと持ってきて

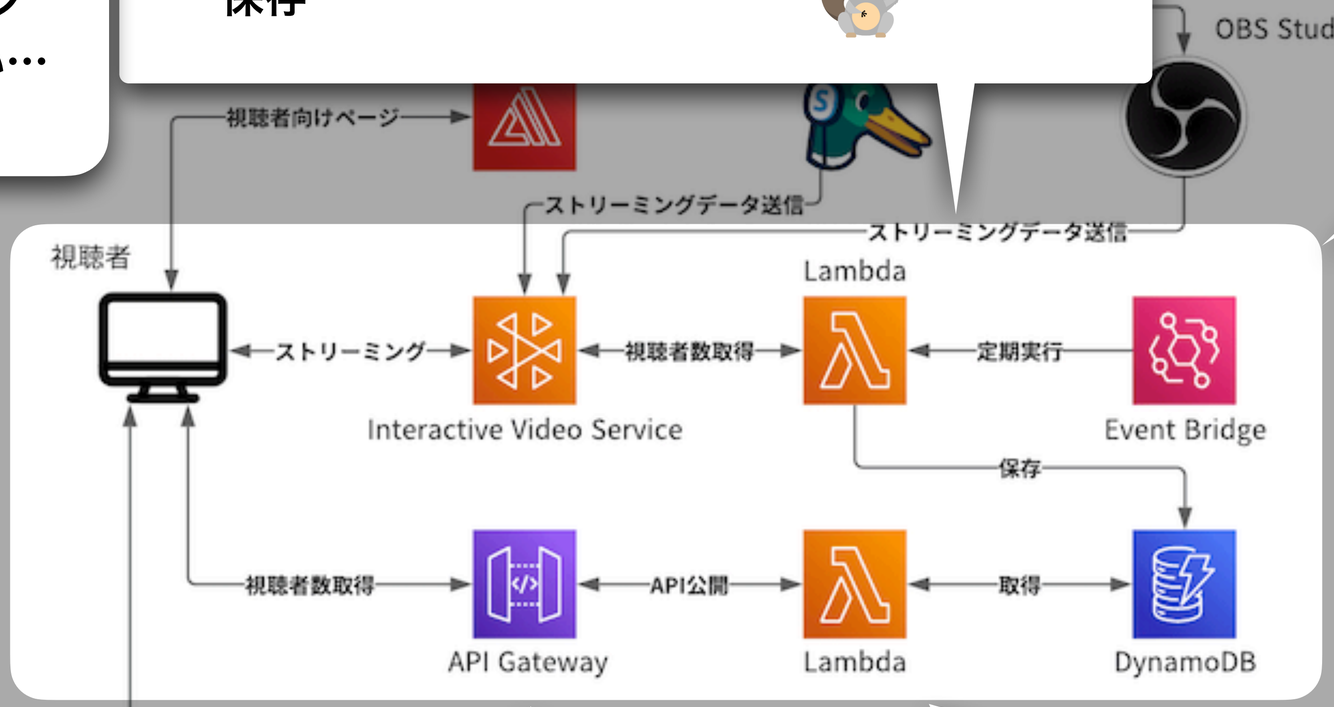
```
118 -       const script = document.createElement('script')
119 -       script.setAttribute('id', 'player-script-' + index)
120 -       script.innerHTML = `
121 -         if (typeof player${index} === 'undefined') {
122 -           registerIVSTech(videojs)
123 -           registerIVSQualityPlugin(videojs)
124 -           const player${index} = videojs('video-player-${index}-${this.timestamp}', {
125 -               techOrder: ["AmazonIVS"]
126 -           })
127 -           player${index}.enableIVSQualityPlugin()
128 -           player${index}.src("${stream.url}")
129 -         }
130 -       `
131 -       document.body.appendChild(script)
    130 +       const player = videojs(`video-player-${index}-${this.timestamp}`, {
    131 +         techOrder: ['AmazonIVS']
    132 +       })
    133 +       player.enableIVSQualityPlugin()
    134 +       player.src(stream.url)
```

強引な DOM 操作を廃止！

AWS Lambda
- Amazon IVS の視聴者数取得
- Amazon DynamoDB に保存

ポーリング
イケてない…

Amazon EventBridge
- Lambda を定期実行



リアルタイム視聴者数取得

Amazon API Gateway
- Lambda へのリクエスト/レスポンス

Lambda
- DynamoDB から視聴者数取得

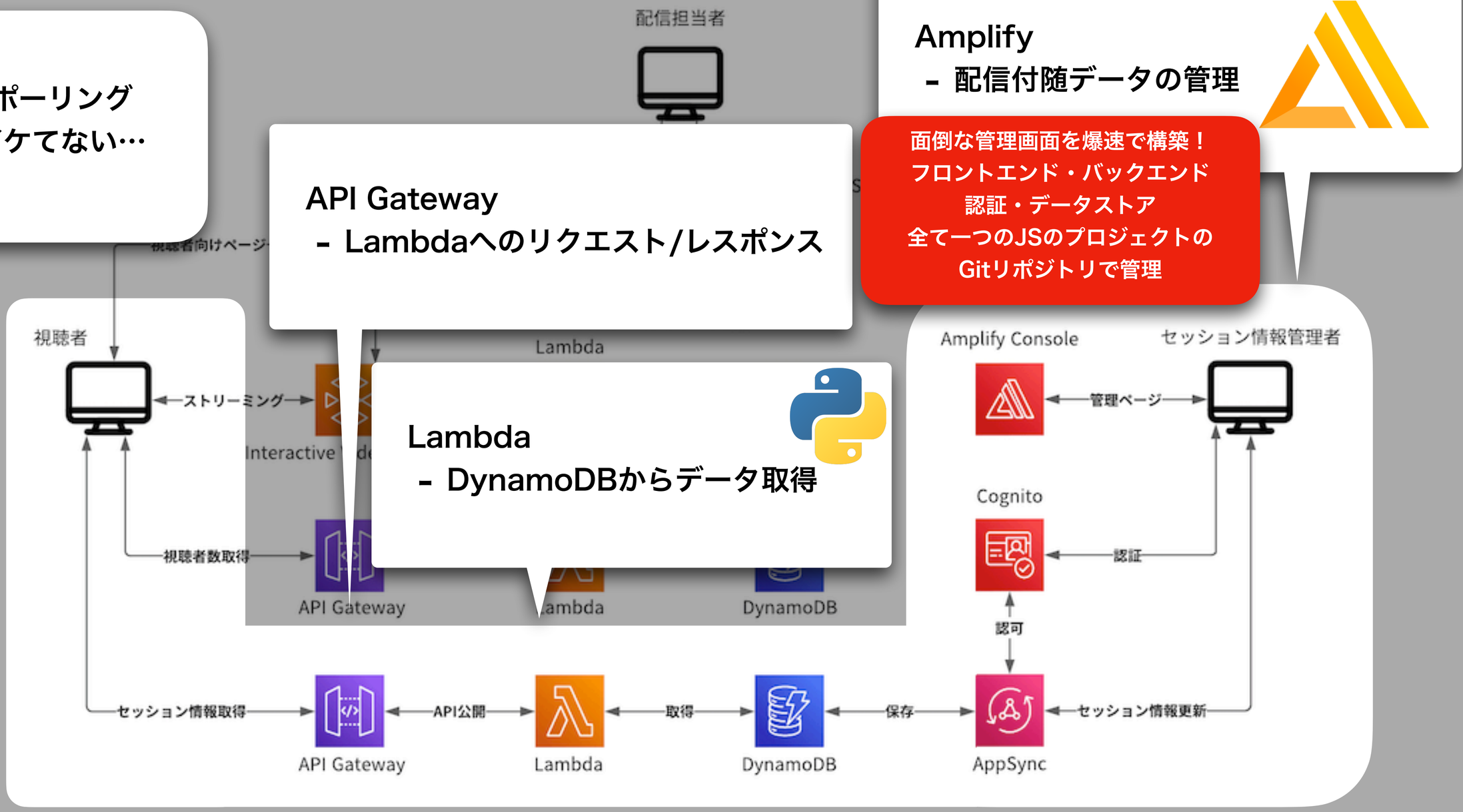
ポーリング
イケてない…

API Gateway
- Lambdaへのリクエスト/レスポンス

Lambda
- DynamoDBからデータ取得

Amplify
- 配信付随データの管理

面倒な管理画面を爆速で構築！
フロントエンド・バックエンド
認証・データストア
全て一つのJSのプロジェクトの
Gitリポジトリで管理



配信サイトの追加の機能要件

アンケート機能を実装したい

配信サイトの非機能要件

- 数千人単位の視聴に耐えたい => Amazon IVS + AmplifyならOKでしょ！

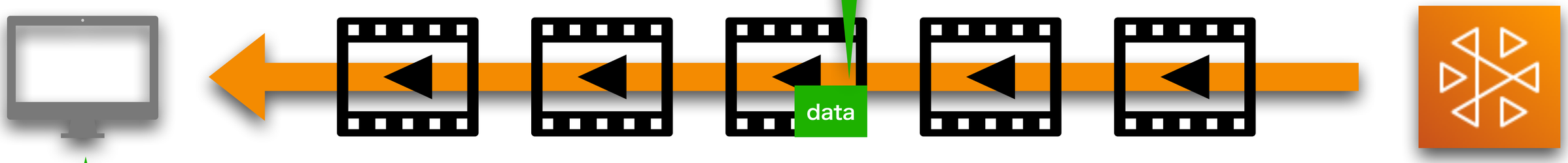
- 諸々のリアルタイムなデータ更新の遅延がない様に => LambdaでOK？

数千クライアントからポーリングくるよ！？

Amazon IVS

Timed Metadata

動画のストリーミングデータに
任意のデータを「相乗り」



JavaScriptのイベントをフック
→ 表示に反映

アンケート機能の実装と
Lambda + API Gatewayの
エンドポイントへのポーリングを
廃止できる！！

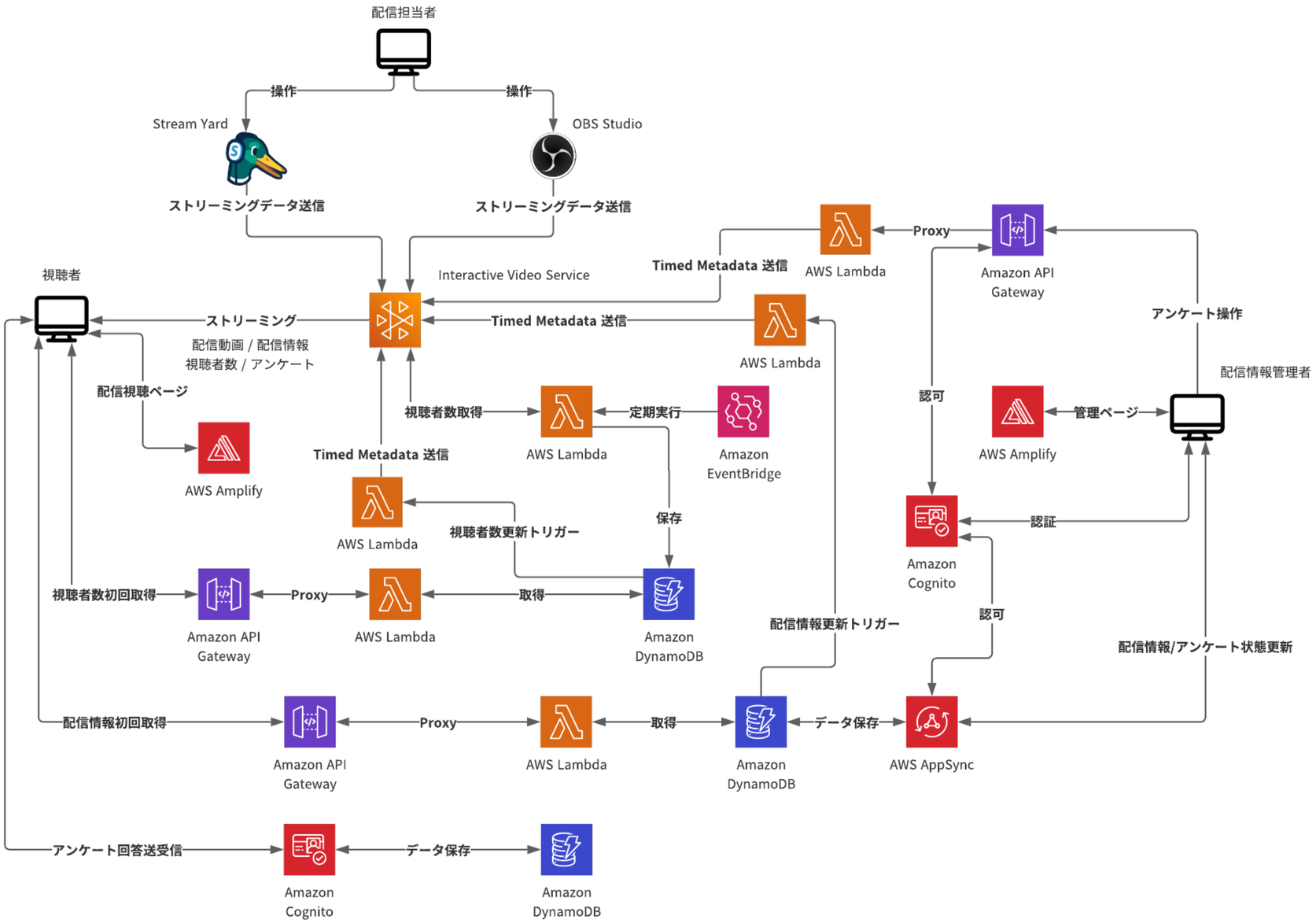
同じチャンネルで
同じテーマでやるならいいが…

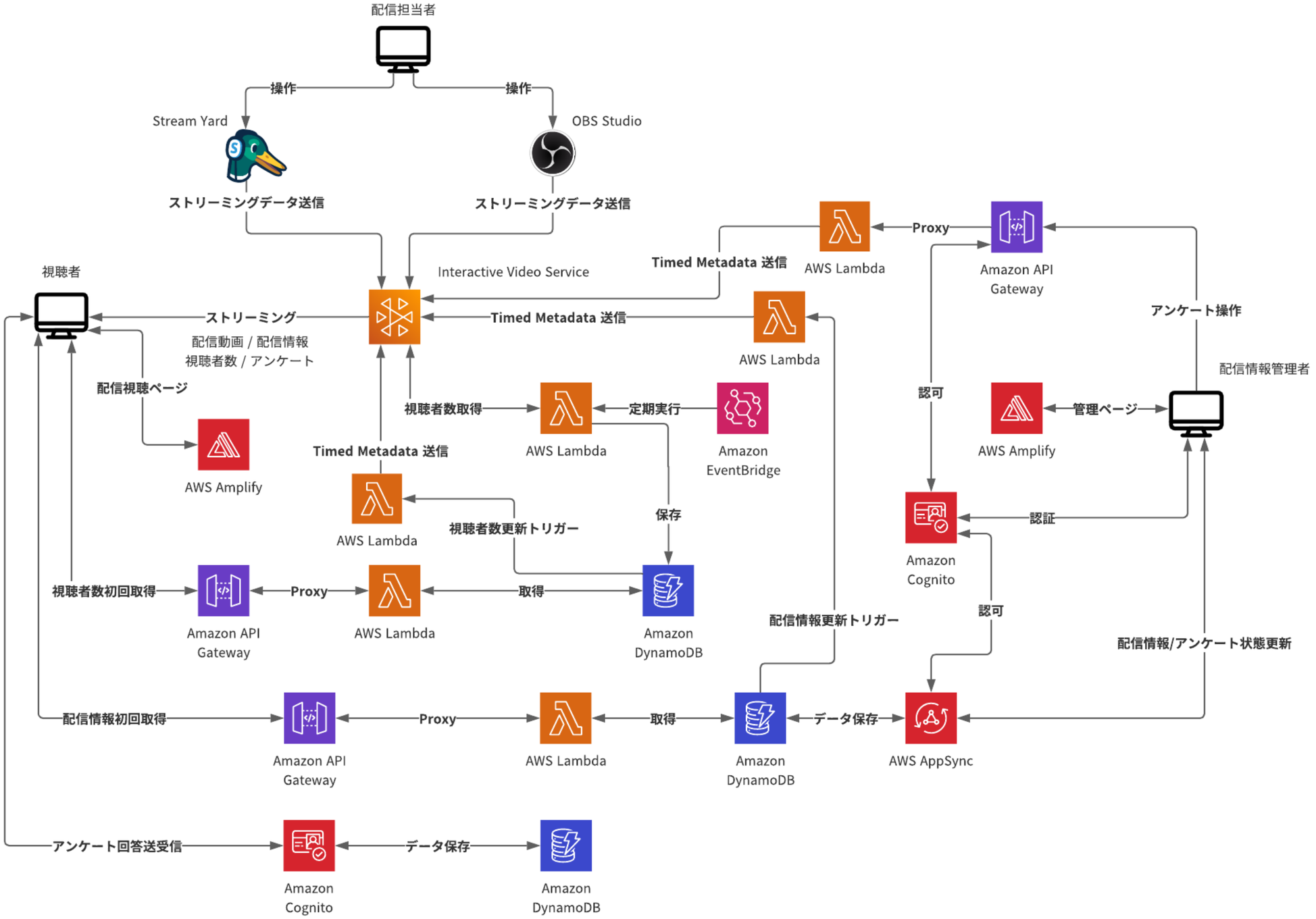
- 時系列的に表示する文字やスタイルを変更したい
- 双方向のインタラクションが欲しい

途端に難しくなる！！

→これをエレガントに解決してくれるのが TimedMetadata

配信アーキテクチャ ver 2.0

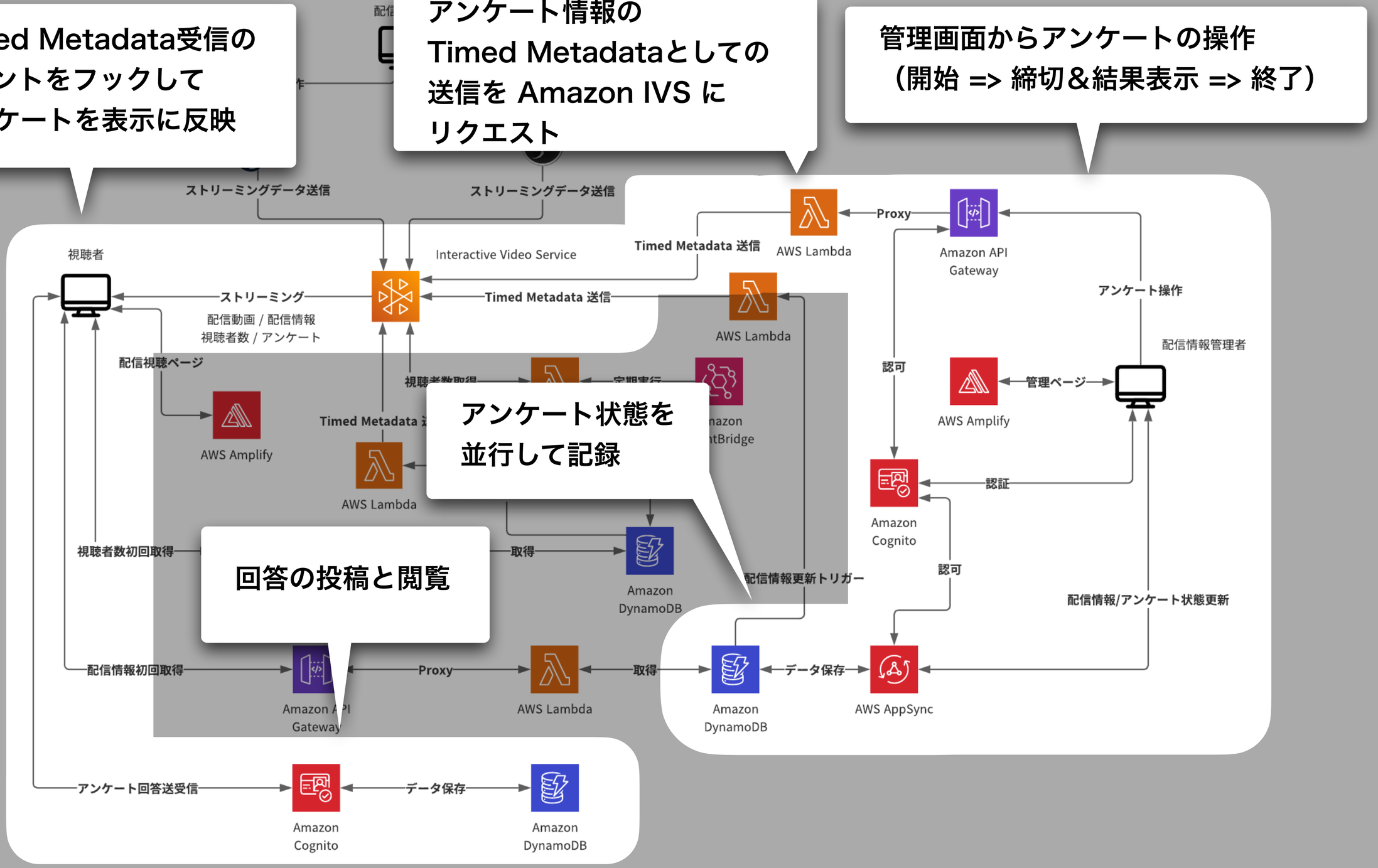


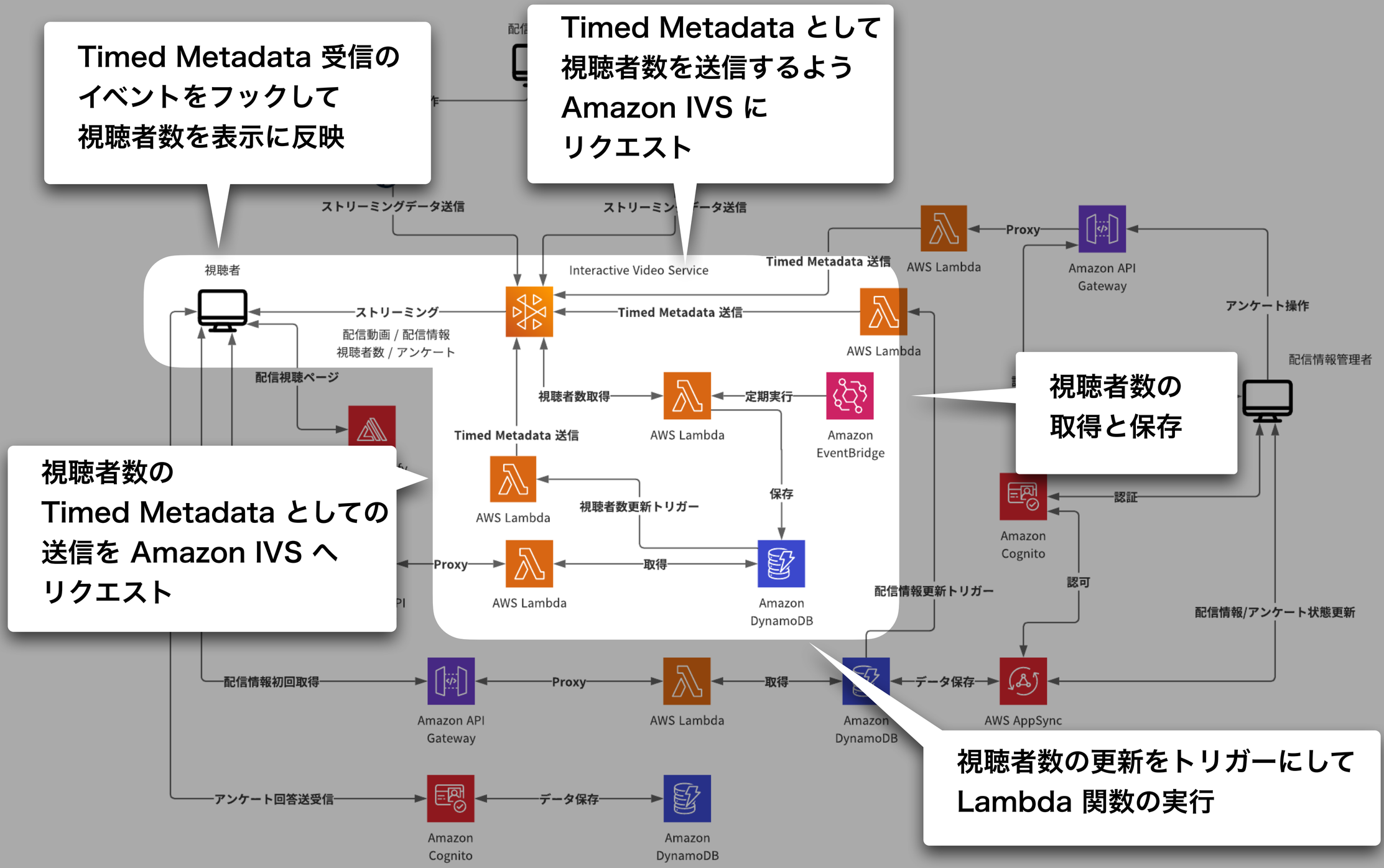

配信担当者
操作
操作
Stream Yard
OBS Studio
ストリーミングデータ送信
ストリーミングデータ送信
視聴者
Interactive Video Service
Timed Metadata 送信
AWS Lambda
Proxy
Amazon API Gateway
ストリーミング
配信動画 / 配信情報
視聴者数 / アンケート
Timed Metadata 送信
AWS Lambda
アンケート操作
配信視聴ページ
視聴者数取得
定期実行
Amazon EventBridge
認可
管理ページ
配信情報管理者
AWS Amplify
AWS Amplify
Timed Metadata 送信
AWS Lambda
視聴者数更新トリガー
保存
認証
Amazon Cognito
視聴者数初回取得
Proxy
取得
Amazon API Gateway
AWS Lambda
Amazon DynamoDB
配信情報更新トリガー
認可
配信情報/アンケート状態更新
配信情報初回取得
Proxy
取得
データ保存
Amazon API Gateway
AWS Lambda
Amazon DynamoDB
AWS AppSync
アンケート回答送受信
データ保存
Amazon Cognito
Amazon DynamoDB

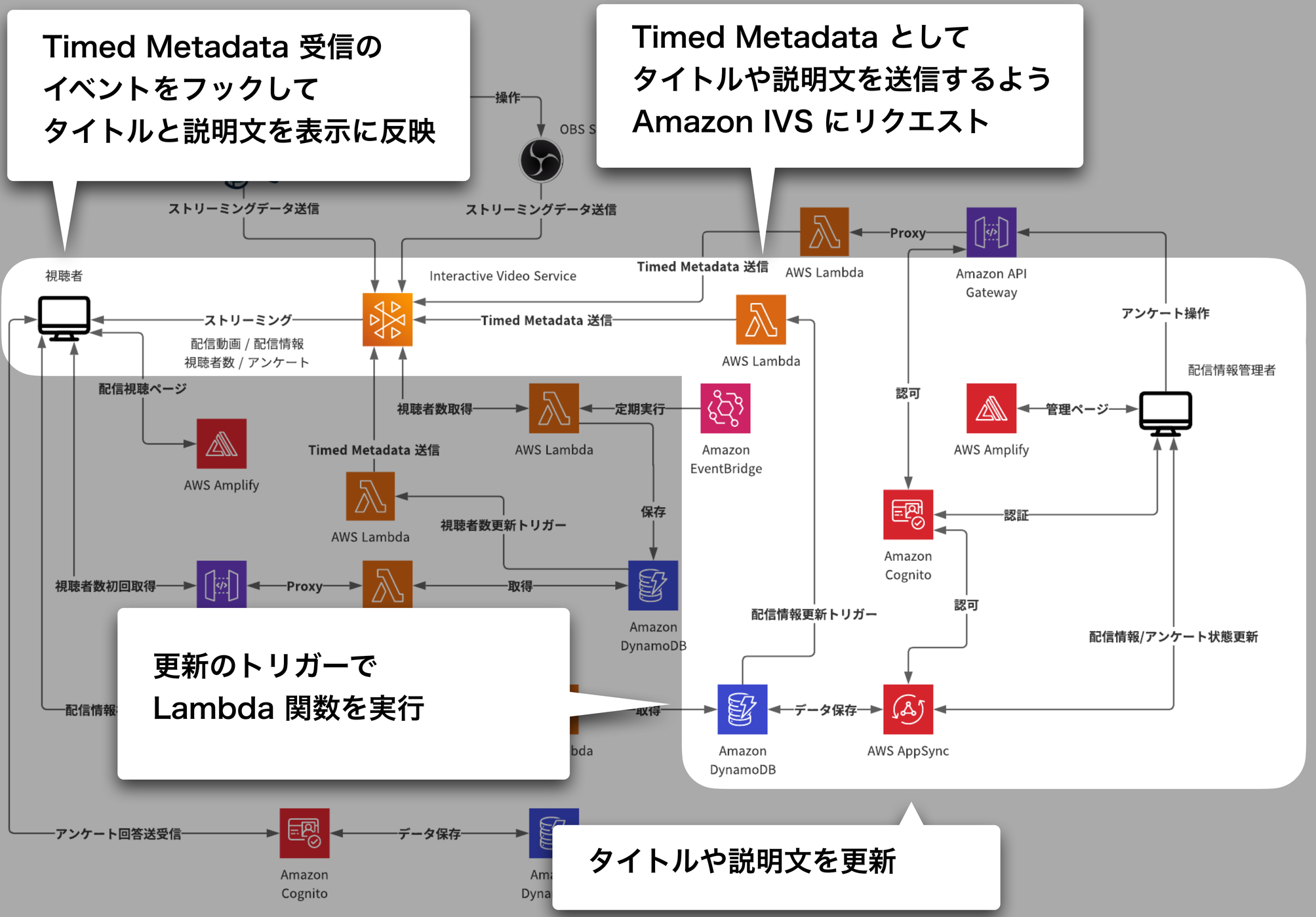
Timed Metadata受信の
イベントをフックして
アンケートを表示に反映

アンケート情報の
Timed Metadataとしての
送信を Amazon IVS に
リクエスト

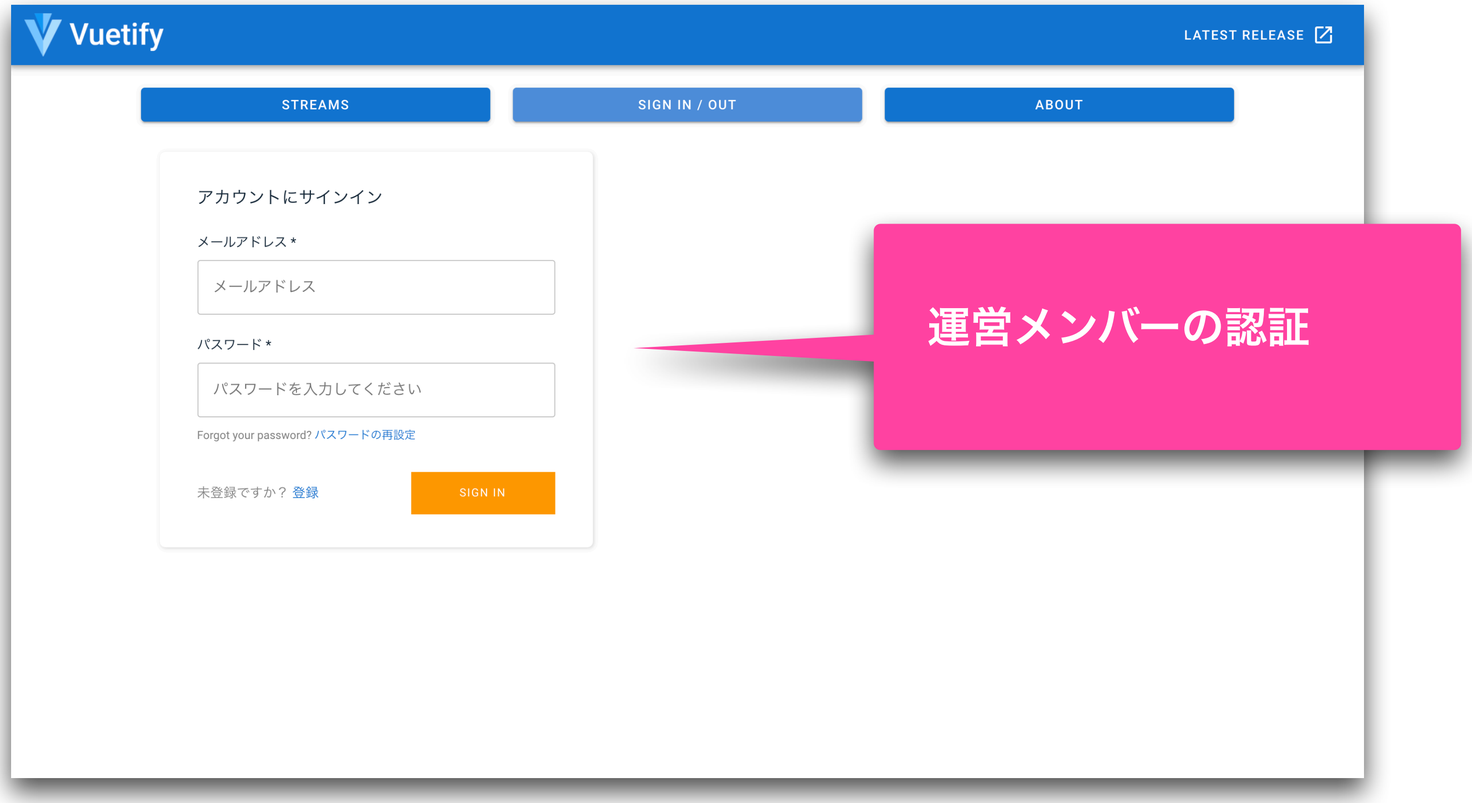
管理画面からアンケートの操作
（開始 => 締切＆結果表示 => 終了）

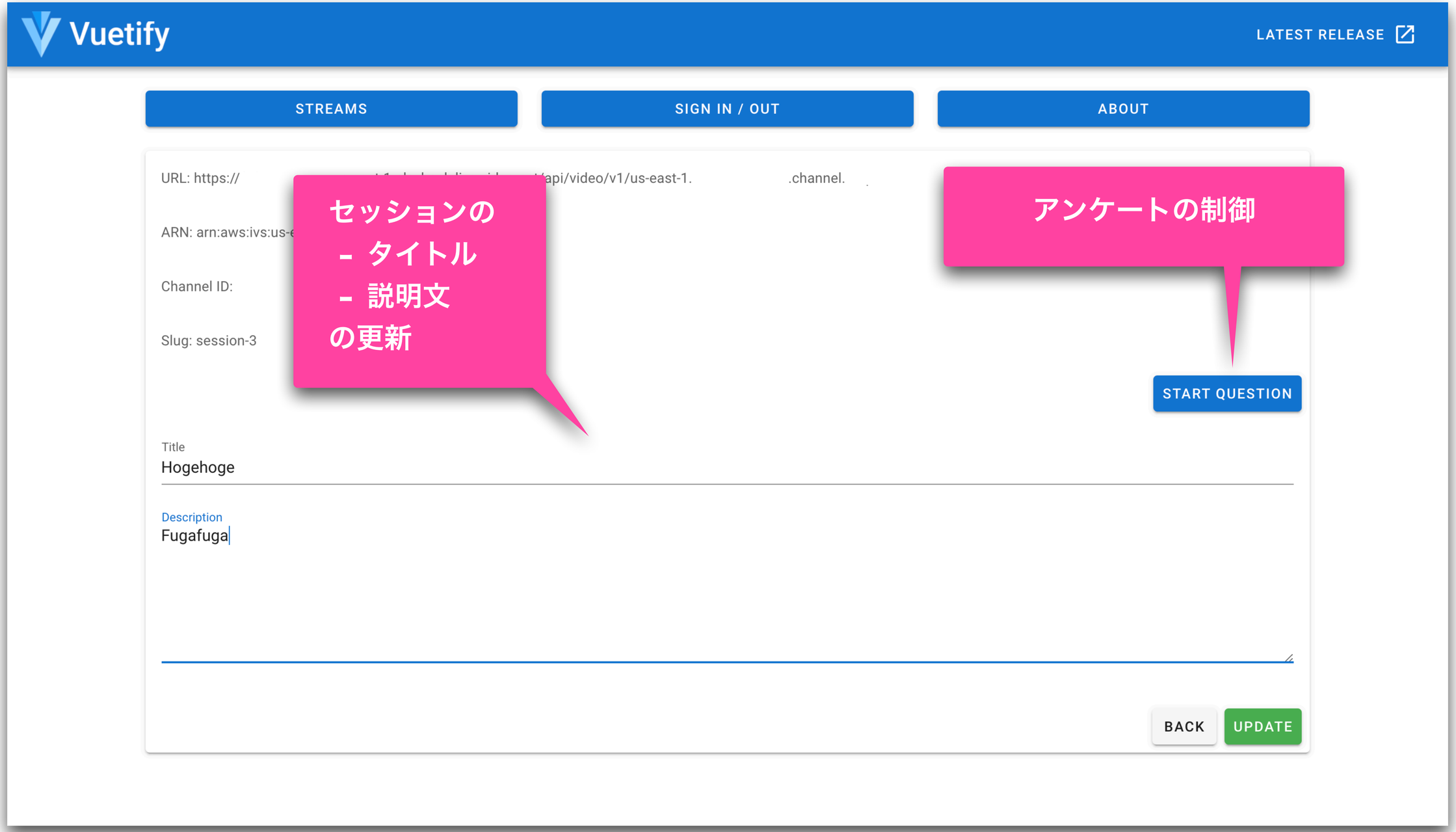



Timed Metadata 受信の
イベントをフックして
視聴者数を表示に反映
Timed Metadata として
視聴者数を送信するよう
Amazon IVS に
リクエスト
視聴者
Interactive Video Service
ストリーミングデータ送信
ストリーミング
配信動画 / 配信情報
視聴者数 / アンケート
Timed Metadata 送信
AWS Lambda
Proxy
Amazon API Gateway
アンケート操作
配信情報管理者
配信視聴ページ
視聴者数取得
定期実行
Amazon EventBridge
保存
視聴者数更新トリガー
取得
Amazon DynamoDB
視聴者数の
取得と保存
Amazon Cognito
認証
認可
配信情報更新トリガー
配信情報/アンケート状態更新
視聴者数の
Timed Metadata としての
送信を Amazon IVS へ
リクエスト
配信情報初回取得
データ保存
AWS AppSync
アンケート回答送受信
視聴者数の更新をトリガーにして
Lambda 関数の実行


Timed Metadata 受信の
イベントをフックして
タイトルと説明文を表示に反映
Timed Metadata として
タイトルや説明文を送信するよう
Amazon IVS にリクエスト
操作
OBS S
ストリーミングデータ送信
ストリーミングデータ送信
視聴者
Interactive Video Service
Timed Metadata 送信
AWS Lambda
Proxy
Amazon API Gateway
ストリーミング
配信動画 / 配信情報
視聴者数 / アンケート
Timed Metadata 送信
AWS Lambda
アンケート操作
配信視聴ページ
視聴者数取得
定期実行
Amazon EventBridge
認可
AWS Amplify
管理ページ
配信情報管理者
AWS Amplify
Timed Metadata 送信
AWS Lambda
保存
視聴者数更新トリガー
Amazon Cognito
認証
視聴者数初回取得
Proxy
取得
Amazon DynamoDB
配信情報更新トリガー
認可
配信情報/アンケート状態更新
更新のトリガーで
Lambda 関数を実行
取得
データ保存
Amazon DynamoDB
AWS AppSync
アンケート回答送受信
データ保存
Amazon Cognito
タイトルや説明文を更新

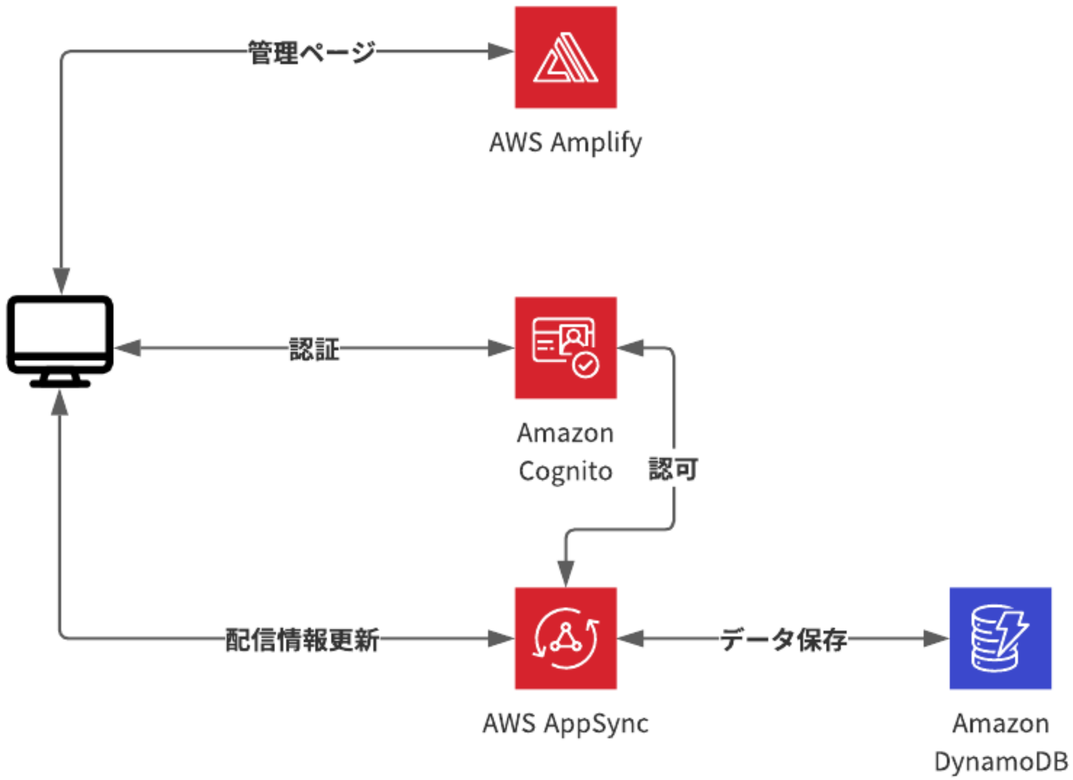
Amplify でバックエンドも
丸っとやった管理画面




Vuetify
LATEST RELEASE
STREAMS
SIGN IN / OUT
ABOUT
URL: https://
api/video/v1/us-east-1.
.channel.
ARN: arn:aws:ivs:us-
Channel ID:
Slug: session-3
セッションの
- タイトル
- 説明文
の更新
アンケートの制御
START QUESTION
Title
Hogehoge
Description
Fugafuga
BACK
UPDATE

```
type Stream @model @auth(rules: [{ allow: private }]) {
  id: ID!
  url: String
  slug: String
  title: String
  description: String
  active: Boolean
  metadataId: String
  state: Int
}
```

IVS ストリーム URL
配信ページの Slug
配信ページのタイトル
配信ページの解説文
有効/無効フラグ
アンケートの ID
アンケートの状態



問題発生

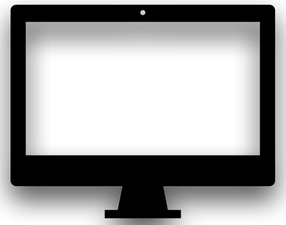
IVS Timed Metadata Quotas

Stream state	PutMetadata	5 TPS per channel

Metadata payload	1 KB	Maximum size of a PutMetadata request payload (Amazon IVS API).

セッション説明文の文字数足らない

分割して送ろう（単純）



4回に分けて送信

4回に分けてリクエスト

原文原文原文…原文

原文原文原文…原文

復元

4分割

長めの説明文に対応！

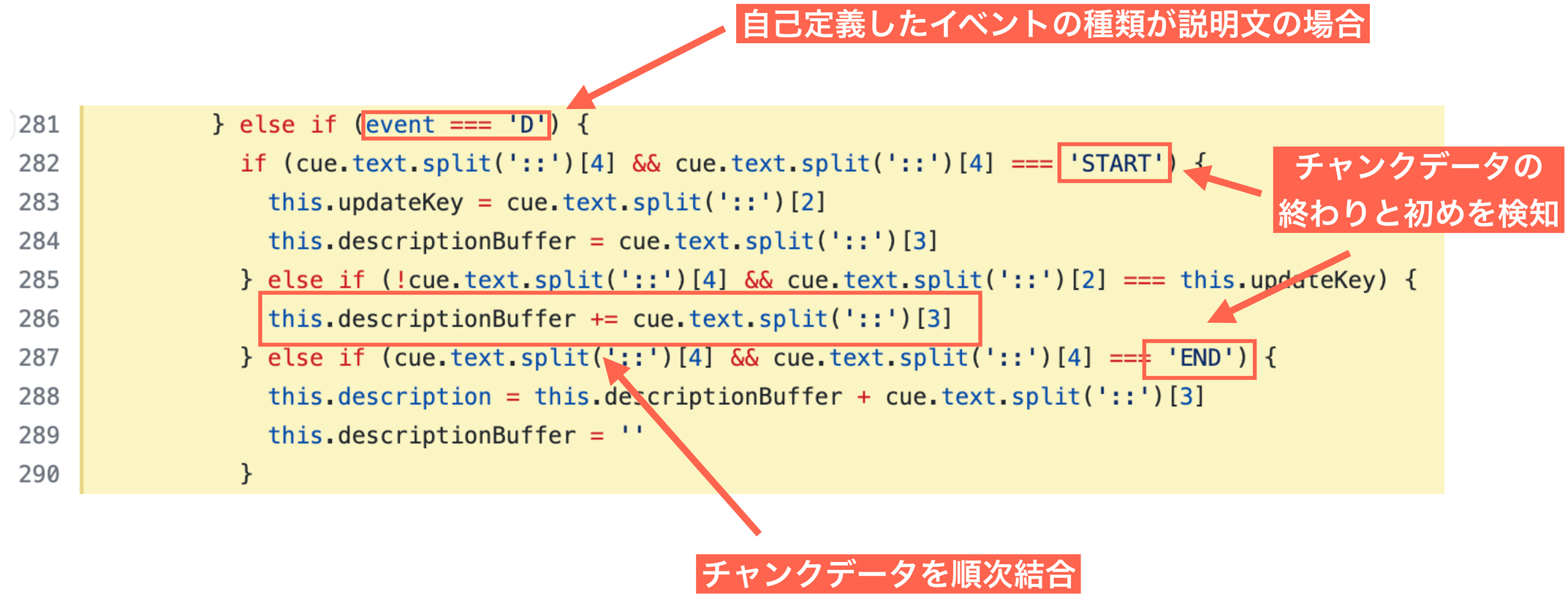
断片1::Key::START
断片2::Key
断片3::Key
断片4::Key::END

断片1::Key::START
断片2::Key
断片3::Key
断片4::Key::END

フロント側のコード解説

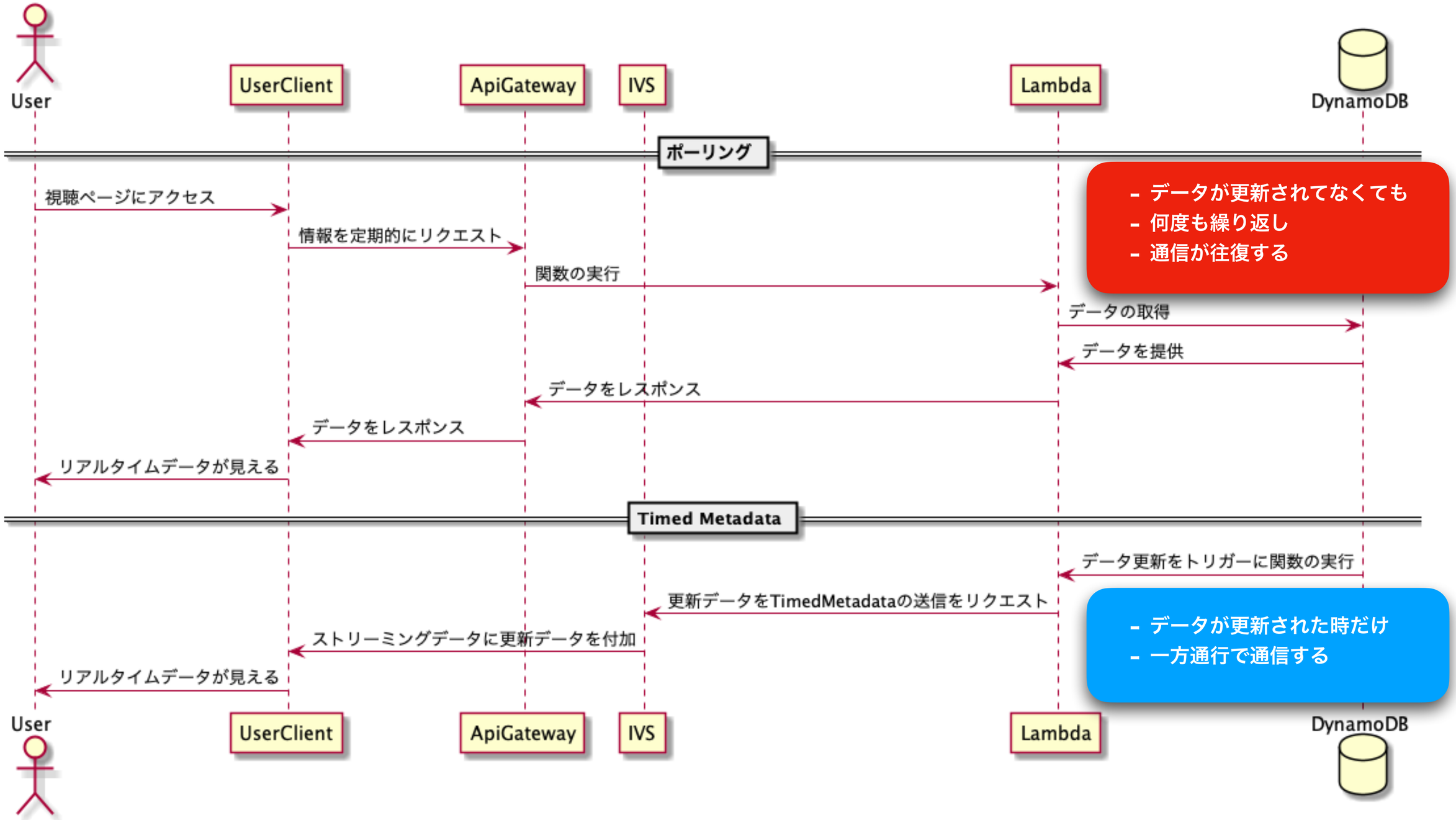

```
235         const playerEvent = this.player.getIVSEvents().PlayerEventType
236         this.player.getIVSPlayer().addEventListener(playerEvent.TEXT_METADATA_CUE, (cue) => {
237           const event = cue.text.split(':')[0]
238           if (event === 'Q') {
239             this.questionId = cue.text.split(':')[1]
240             this.message = '↓下から投票してください!↓'
241             this.snackbar = true
242             this.question = true
243           } else if (event === 'R') {
244             this.questionId = cue.text.split(':')[1]
245             this.question = false
```

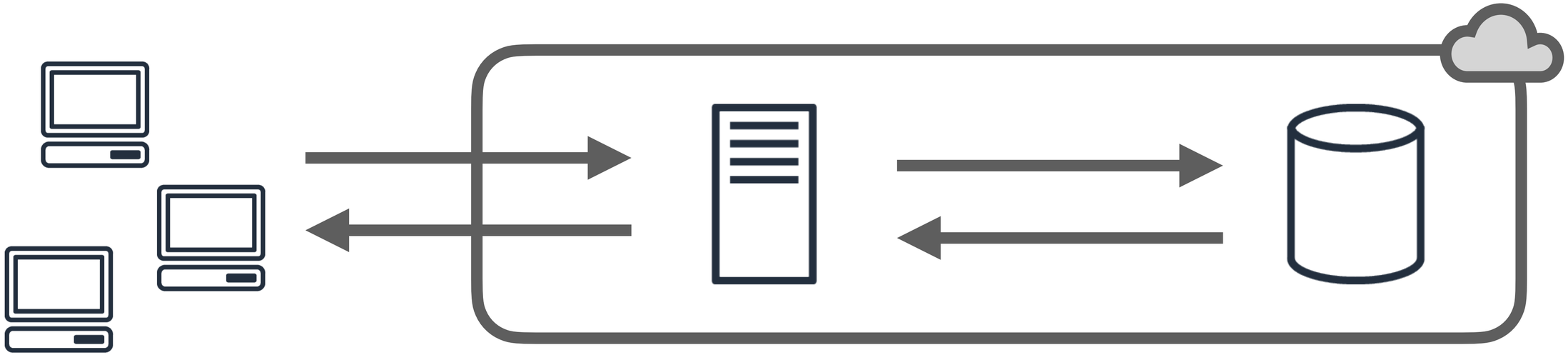
フロント側のコード解説

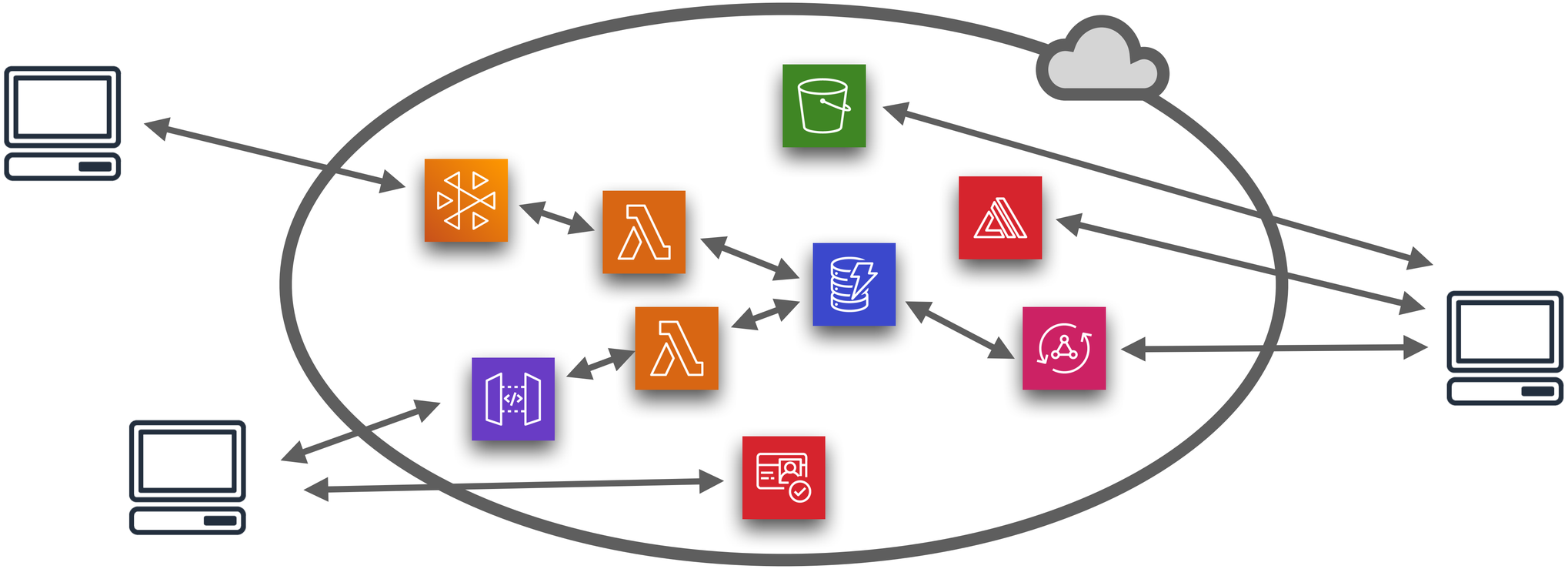


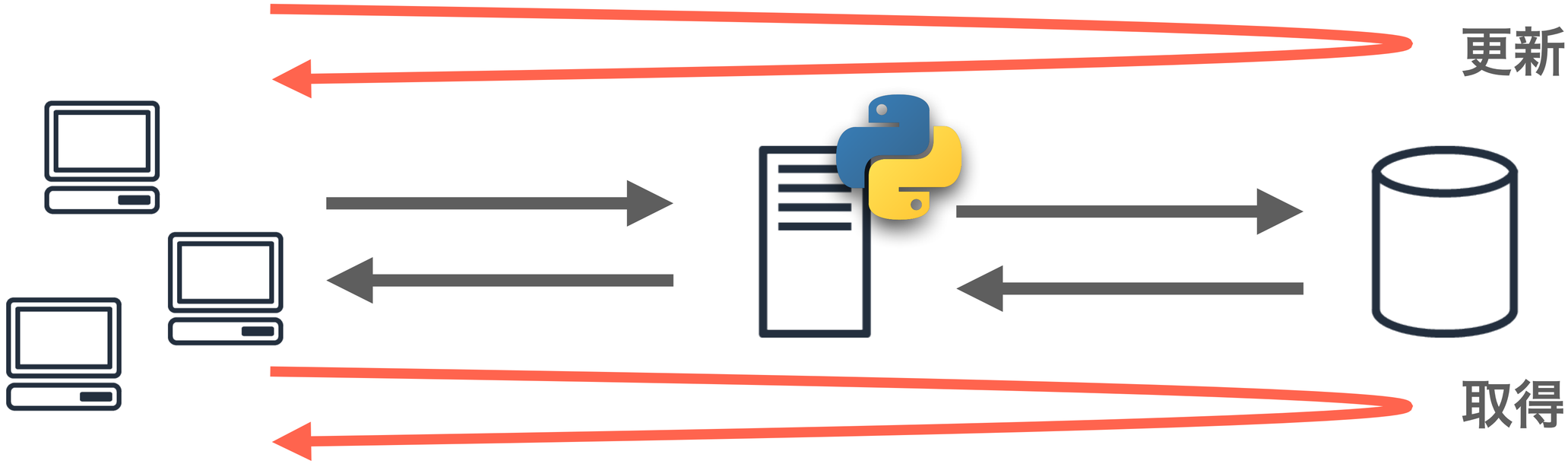

```
281         } else if (event === 'D') {
282           if (cue.text.split('::')[4] && cue.text.split('::')[4] === 'START') {
283             this.updateKey = cue.text.split('::')[2]
284             this.descriptionBuffer = cue.text.split('::')[3]
285           } else if (!cue.text.split('::')[4] && cue.text.split('::')[2] === this.updateKey) {
286             this.descriptionBuffer += cue.text.split('::')[3]
287           } else if (cue.text.split('::')[4] && cue.text.split('::')[4] === 'END') {
288             this.description = this.descriptionBuffer + cue.text.split('::')[3]
289             this.descriptionBuffer = ''
290           }
```

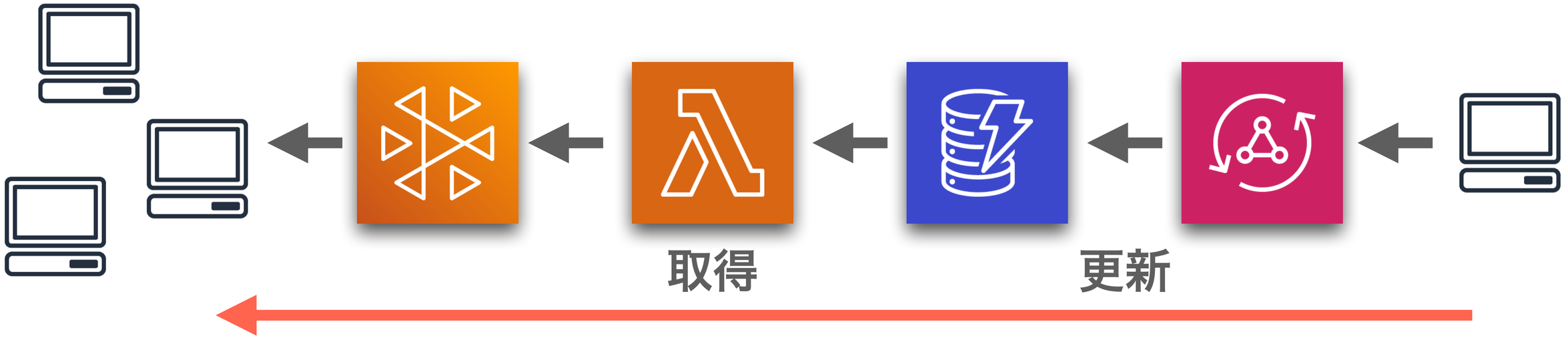
リアルタイムデータ更新方法



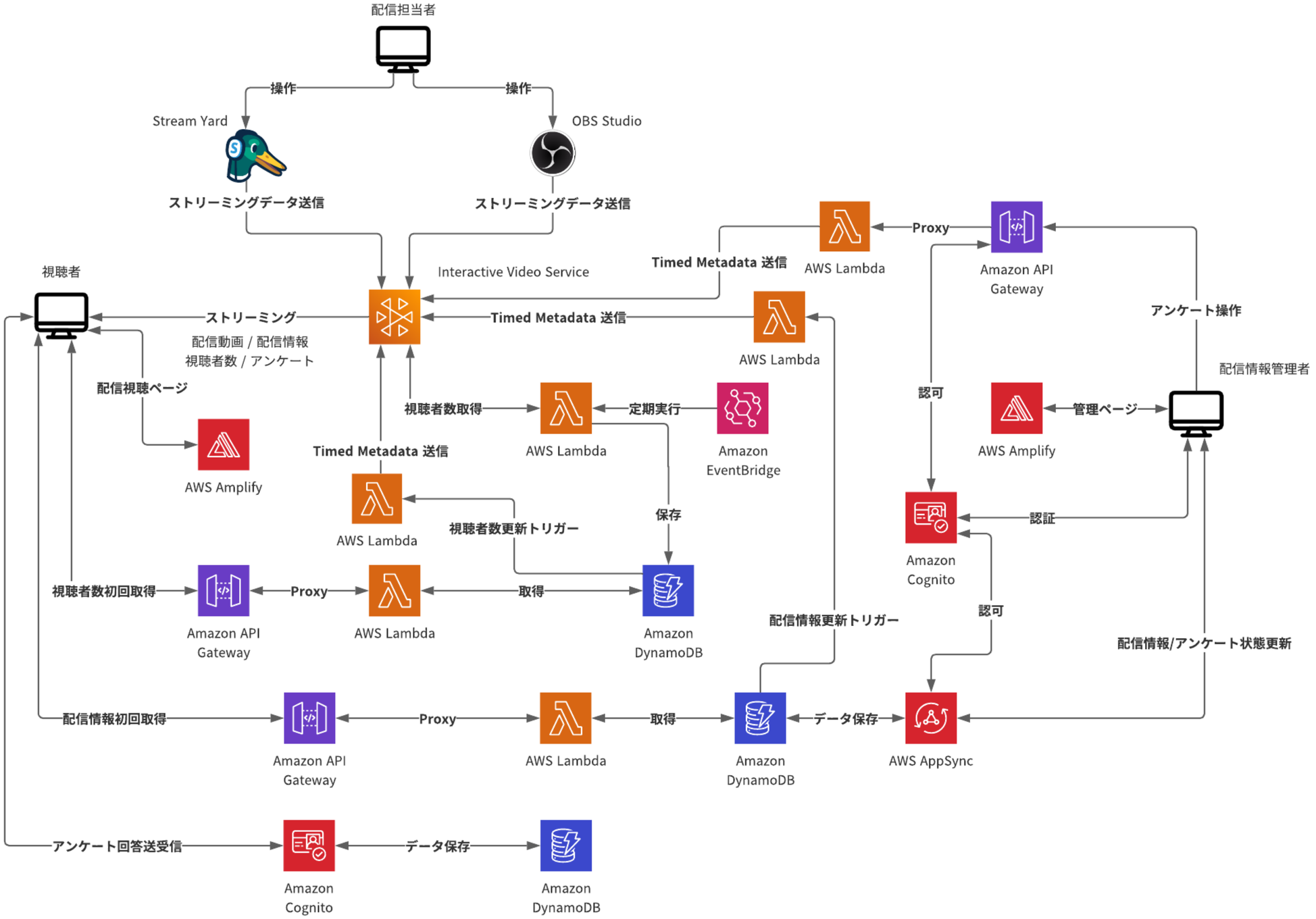





更新
取得


取得
更新

イベントドリブン！


配信担当者
操作
操作
Stream Yard
OBS Studio
ストリーミングデータ送信
ストリーミングデータ送信
視聴者
Interactive Video Service
Timed Metadata 送信
AWS Lambda
Proxy
Amazon API Gateway
ストリーミング
配信動画 / 配信情報
視聴者数 / アンケート
Timed Metadata 送信
AWS Lambda
アンケート操作
配信視聴ページ
視聴者数取得
定期実行
Amazon EventBridge
認可
管理ページ
配信情報管理者
AWS Amplify
AWS Amplify
Timed Metadata 送信
AWS Lambda
視聴者数更新トリガー
保存
認証
Amazon Cognito
視聴者数初回取得
Proxy
取得
Amazon API Gateway
AWS Lambda
Amazon DynamoDB
配信情報更新トリガー
認可
配信情報/アンケート状態更新
配信情報初回取得
Proxy
取得
データ保存
Amazon API Gateway
AWS Lambda
Amazon DynamoDB
AWS AppSync
アンケート回答送受信
データ保存
Amazon Cognito
Amazon DynamoDB

めでたしめでたし

まとめ

IVS

- 1：n のストリーミング配信ができる
- SDKで簡単に利用できる
- スケールする
- ストリーミングついでに任意のデータも送れる

配信サイトの自作に最適！

Amplify: 開発面

- インフラも包括したフレームワークで爆速な開発
- 全部完璧に使う必要はなく必要な部分から使っていける
- 他のサービスを組み合わせたプロトタイピングに最適

Amplify: 運用面

- 機能を実現するコードがそのまま
  インフラ管理のコードに
- マイクロサービス前提なので後から
  部分的に改修してもいい

Amplify: オススメしたい人

- 爆速でサービスリリースしたい
  スタートアップの人
- 仕事でもプライベートでもAWSの
  色々なサービスを使う人

Amplifyは
デベロッパーの
パフォーマンスを
Amplify(増幅)するツール

StartupTechnology

積極採用中です！

- エンジニア
- デザイナー
- CS

おすすめポイント

- 風通しがよく提案しやすい
- 柔軟な働き方
- 新規案件・未経験技術へのチャレンジ
- 社外コミュニティ活動の奨励・サポート

「スタテク」でググってみてください！