鳥人間コンテスト
ライブ配信における**リアルタイム英語字幕**

読売テレビ放送株式会社
クロステック局 放送DXグループ

福岡 一輝

My Background

福岡県 福岡市出身

福岡　一輝

Kazuki Fukuoka

- 新卒でNHKに入社
  - おはよう日本やニュースウオッチ９など生放送番組のSW、Cなど
  - 緊急システムやファイルベースシステムなどの設備整備・保守など
- NHK在籍時にAWSに出会ってしまう⚡
  - 現AWS・S氏 (当時NHK) のサポートの下、開発案件に挑戦！

テレビ業界からも技術からも日本からも4年ほど離れたのち・・・

- 2025年に読売テレビ入社（開発）
- 好きなAWSサービス：AWS SAM

NHK 渋谷放送センター
ニューススタジオにて

Seattle, USA
AWS の本拠地にて

鳥人間コンテストとは

1977年から毎年夏に琵琶湖（滋賀県彦根市）で開催される

自作の人力飛行機による飛行距離や飛行時間を競う大会

公式YouTubeにて大会当日のライブ配信（約10時間 × 2日間）

Source：https://www.ytv.co.jp/birdman/

システム開発の背景

- 数多くの外国人ファンへの対応
- YouTubeライブ標準機能の限界
- 独自システムの開発へ

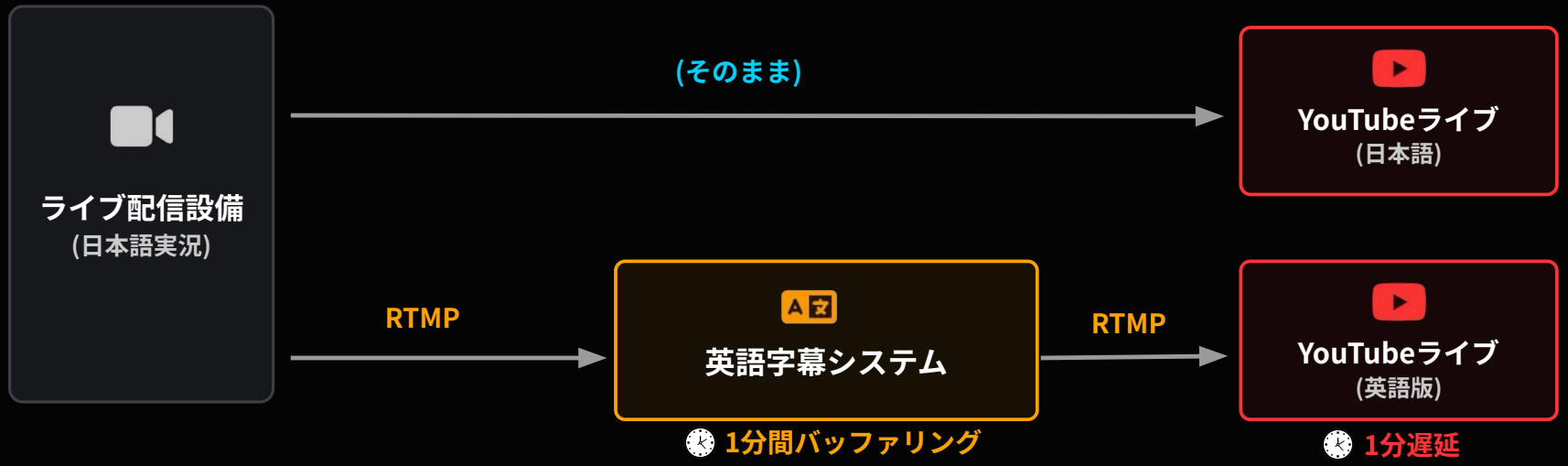
Source：https://www.youtube.com/@ytvbirdman

英語字幕のイメージ

Source：2024年鳥人間コンテストLive配信

アプローチと全体構成

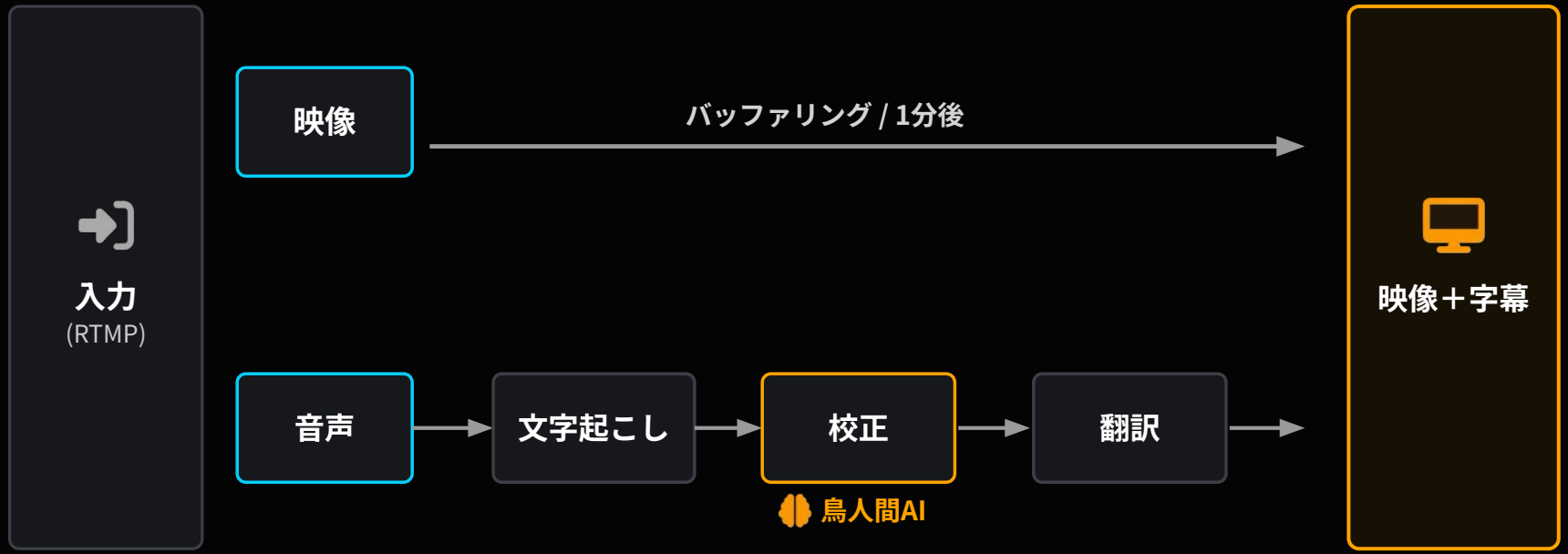
日本語配信と英語配信を別々に分けて配信。
安定した英語字幕の処理のために、英語配信は1分間遅らせる設計を採用。



実際のLive配信映像

Source：2025年鳥人間コンテストLive配信

処理の流れ・パイプライン



鳥人間AIによる校正とは？

文字起こしでは解決できない課題を、AIとロジックで補正

「鳥人間AI」を用いた文章校正と補完

⚠文字起こしミスの修正

例：「今回のフライ (❌フライ → ✅フライト) では」

⚠主語がない表現の補完

「まもなく飛び立ちます」→「(❌I → ✅It) will take off soon」

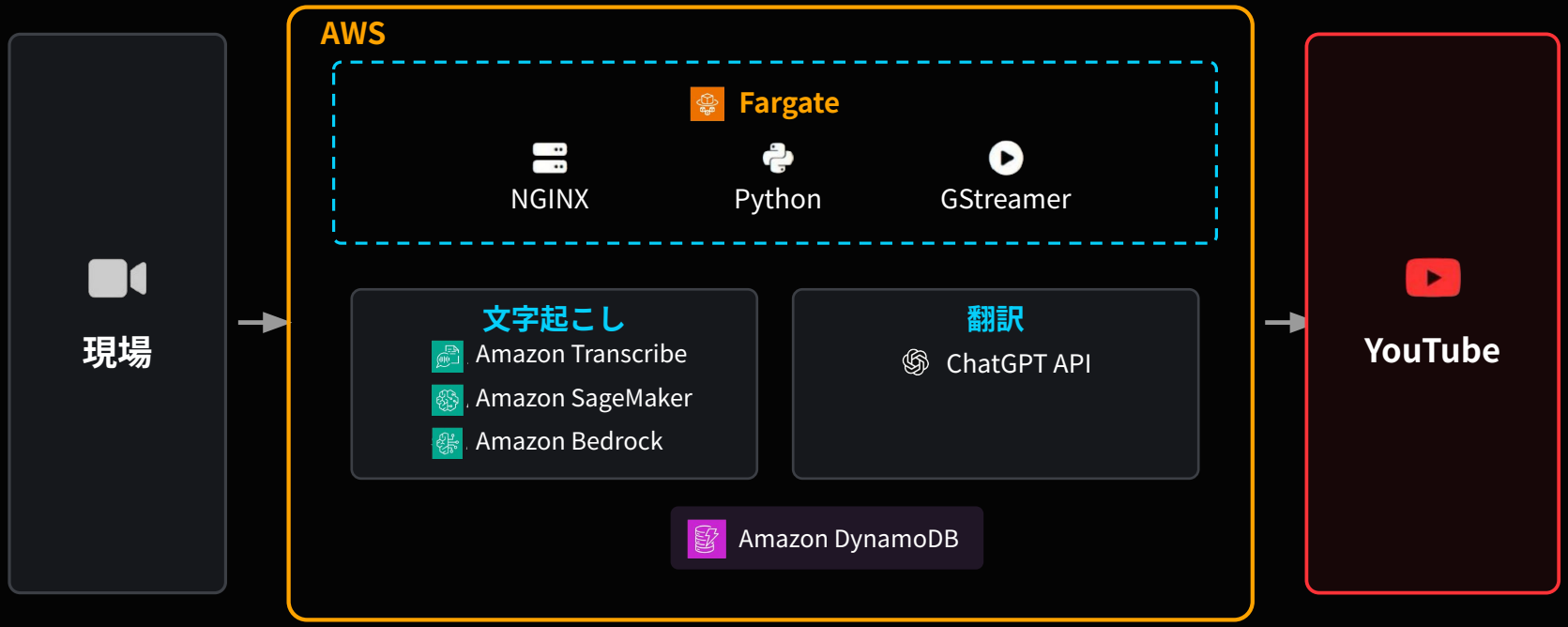
⚠固有名詞と専門用語の誤変換の修正

「❌ウィンドロス」→「✅ Windnauts」、「❌期待」→「✅ 機体」

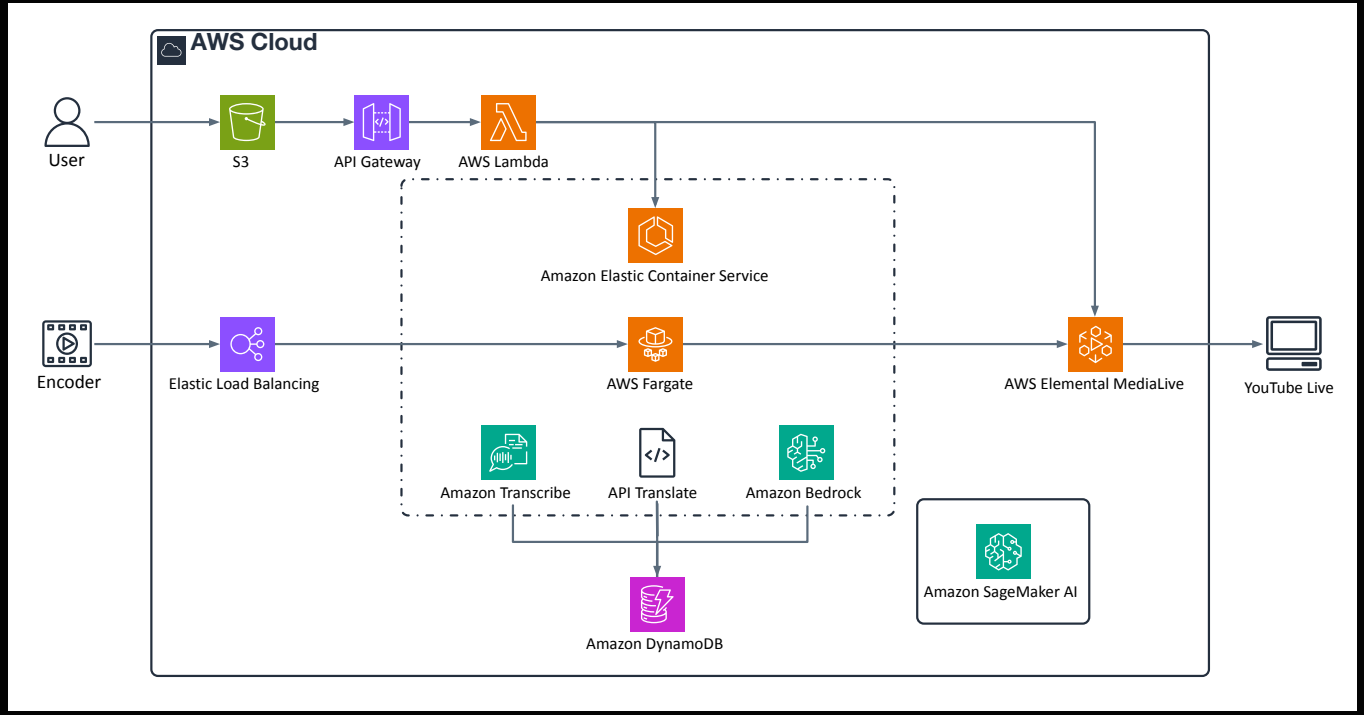
具体的な校正結果の比較

左：❌校正前	右：✅校正後
競合チームエアロセプシーのリーダー、	競合チームAeroscepsyのリーダー、
機期待設計士として六度の優勝。	機体設計士として6度の優勝。
1998年の第22回大会では、	1998年の第22回大会では、
史上初琵琶湖横断飛行を達成し優勝。	史上初の湖面横断飛行を達成し優勝。
そして、	そして、
2006年から鳥人間コンテスト、	2006年から鳥人間コンテスト、
テクニカルアドバイザーを務めています。	テクニカルアドバイザーを務めています。
でま一本当に生きるレジェントなんですけれども、	でま一本当に生きるレジェントなんですけれども、
あのーマさん、	あのMasatoさん、
すでに500人の方がですね、	すでに500人の方がですね、
youtubeを見て、	youtubeを見て、
コメントもどしどし来てくれてます。	コメントもどしどし来てくれてます。
あのスズキさんを応援するコメントがこういっぱいあります。	あのSuzukiさんを応援するコメントがこういっぱいあります。
ありがとうございます。	ありがとうございます。
力になりますもんね。	力になりますもんね。
さあこの後6時からまず昨日フライトができなかった人力プロペラ機部門の東北大学ウィンドロスのフライトをお送りします。	さあこの後6時からまず昨日フライトができなかった人力プロペラ機部門の東北大学Windnautsのフライトをお送りします。

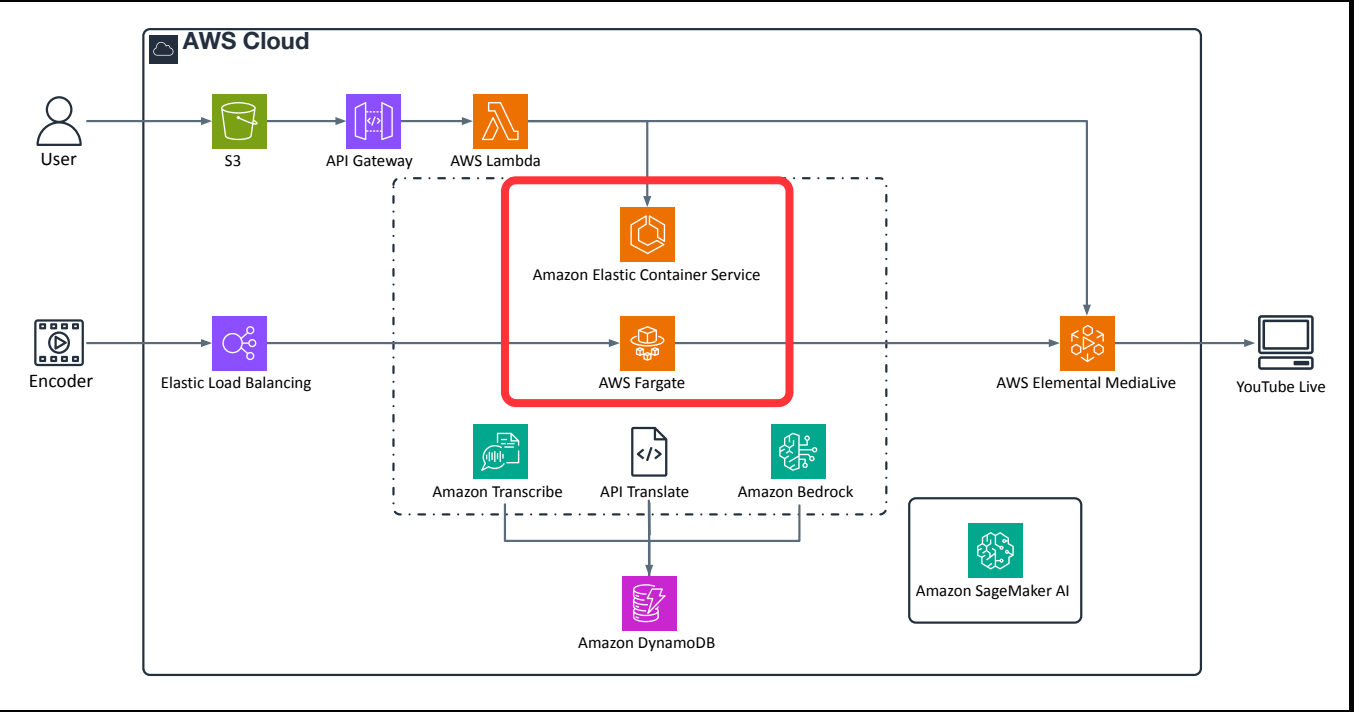
全体システム構成



AWSアーキテクチャ図



システム処理部について



Amazon Elastic Container Service

手元の Docker コンテナを、そのまま AWS へ ― デプロイから運用までを束ねるオーケストレーションツール

① Docker からシームレスにデプロイ

- ▶ 手元の Docker を Amazon ECR にプッシュするだけ
- ▶ 複雑なインフラ構築なしで、そのまま即デプロイ
- ▶ 開発初心者でも挑戦しやすい◎

② マルチコンテナ設計でリスク分散

- ▶ 役割ごとにコンテナを分離して配置できる
- ▶ 片方に不具合が出ても、該当コンテナだけを再起動

✅ 柔軟なシステム構成を実現

AWS Fargate（システム基盤）

サーバー管理はゼロ。コードとコンテナだけに集中できる、フルマネージドな実行基盤

① 完全サーバーレス（管理不要）

- OS 管理・パッチ・初期設定はすべて AWS 側が担当
- インフラに不慣れでも、コードとコンテナに集中できた

② スピード起動 & 従量課金

- EC2 より圧倒的に速いコンテナ起動
- 動いた分だけのクリーンな従量課金でコスト最適
- 異常時はタスクを自動で再起動し可用性を確保

③ 制約と対策 ｜ GPU 非対応という制約は、ハイスペックな CPU でカバー

デメリット

GPU は利用不可。
計算リソースは CPU のみ

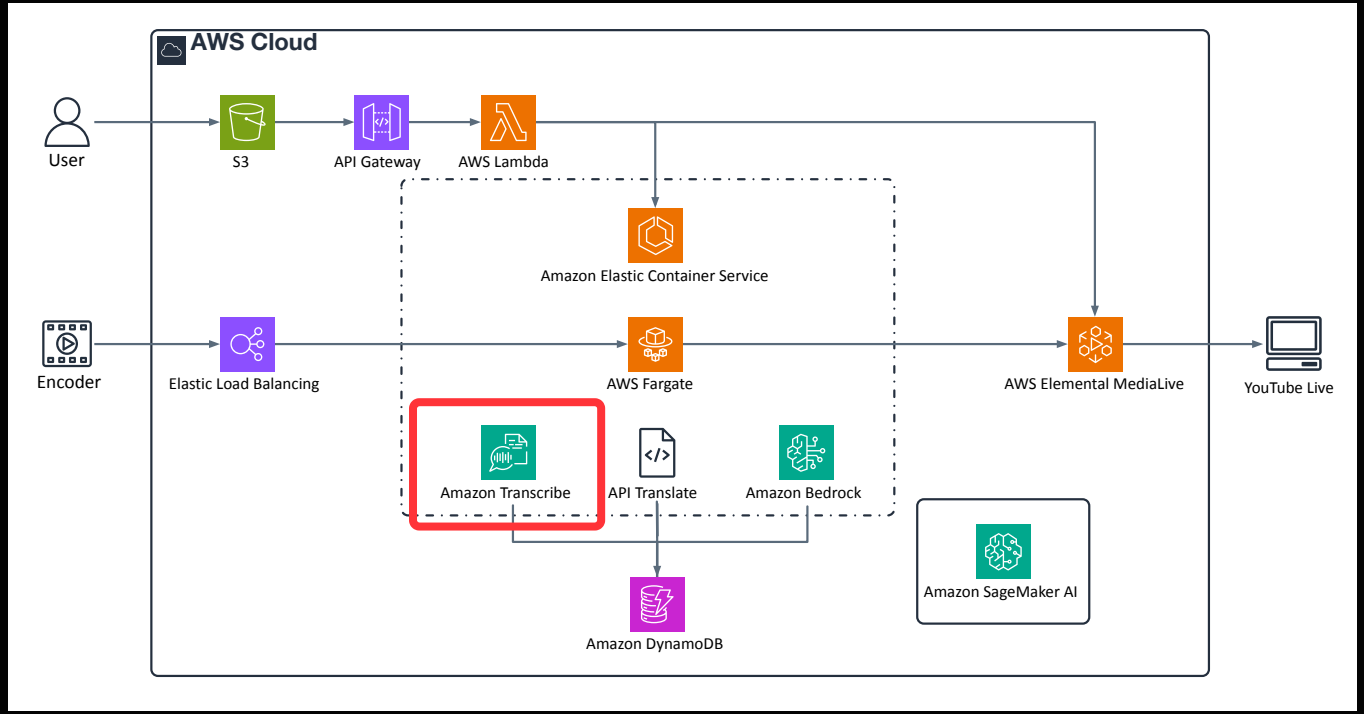
対策 →

対策 ― タスクのスペックを引き上げる

最大 **16 vCPU** / **120 GiB** メモリ

CPU 処理だけでも十分にパワフルに動かせる

文字起こし処理部について



Amazon Transcribe（文字起こし）

音声をリアルタイムで文字起こし

実況音声
マイク入力（ライブ）

→

Amazon Transcribe
リアルタイム文字起こし（ASR）

→

字幕テキスト
リアルタイム出力

フルマネージド・日本語対応・SDK／API で簡単導入（＝選んだ理由）

✓ メリット　— 認識精度を高める3つの機能

カスタムボキャブラリー
Custom Vocabulary
固有名詞・専門用語を登録

✕ ウィンドロス → ✓ Windnauts

カスタム言語モデル（CLM）
Custom Language Model
正解データのテキストから文脈を学習

自前テキスト → 精度UP
※学習データの準備は手間

ボキャブラリーフィルター
Vocabulary Filtering
NGワードを自動で処理

禁止ワード → *** ／削除／タグ

△ デメリット　— 運用上の注意（正直なところ）

精度は音声条件しだい

- 出演者の発音のクリアさ
- 話す内容・文の長さ
- 環境音の有無

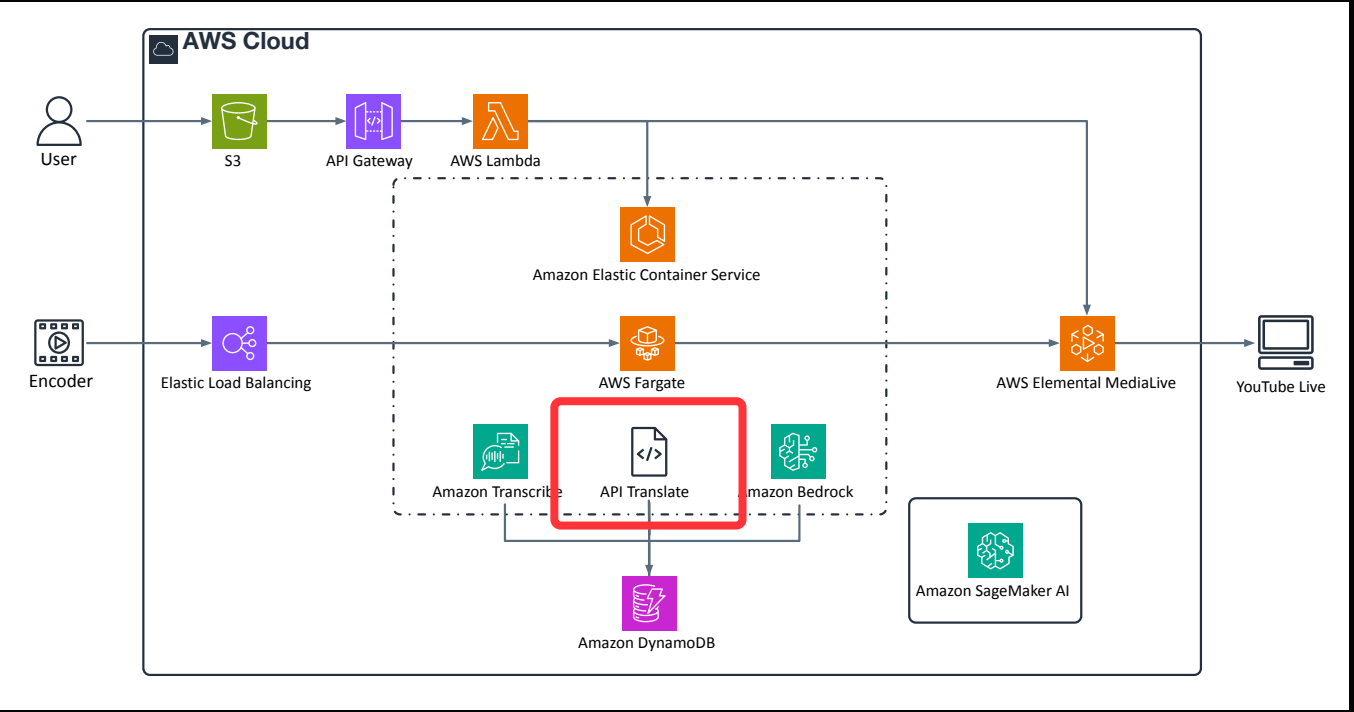
etc.

ストリーミングは再接続が必要

～4時間　再接続→　～4時間　再接続→　～4時間

最大セッション長で切断 → 新セッションで再接続＋字幕を結合する実装が必須

翻訳処理部について



Amazon Translate（翻訳）

翻訳特化のAWSサービス ― 決め手は“自由なカスタマイズ”

決め手　翻訳を“自由にカスタマイズ”したい（文脈理解・文章の長さ調整・スタイル制御）　→　**GPT API を採用**

Amazon Translate
翻訳専用サービス（NMT）

見送り

多言語を高速・大量に翻訳する翻訳特化AI

- 高速・大量翻訳に強い
- Custom Terminology（カスタム用語）
  固有名詞・専門用語の訳を統一できる
- Parallel Data・ACT（Active Custom Translation）
  対訳例を参照して翻訳スタイルを調整

※ モデル学習ではなく、対訳例で出力を調整する機能

GPT API で翻訳
LLM を API 経由で呼び出して翻訳

✓採用

採用理由：翻訳を自由にカスタマイズしたかった

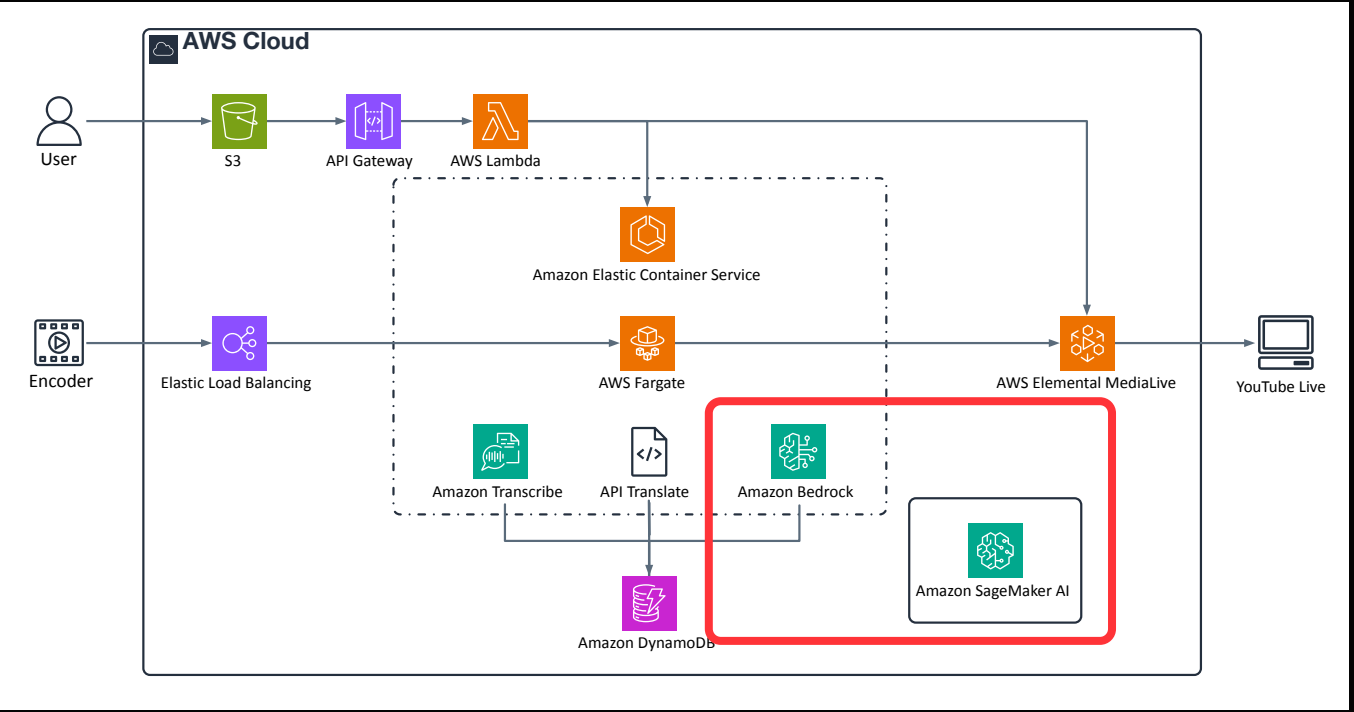
- プロンプトで柔軟にカスタマイズ
  文脈理解の自然な翻訳・文章の長さ調整・スタイル制御
- 使うAIモデル（翻訳モデル）を選択・最適化

プロンプト＝翻訳ルールを自在に記述できる

採用（自由度を最優先）

※ API 利用の注意　レート上限（リクエスト／トークン数）超過時の処理 ― リトライ／指数バックオフの実装が必要

AIを用いた校正処理部について



Amazon Sagemaker

既存LLMを“鳥人間専用AI”へ育てる（ファインチューニング） ― 文字起こしの精度限界を補正するため

選んだ理由 ― Amazon Transcribeでは拾いきれない固有名詞・実況表現を補正する“鳥人間AI”を自前で作るため（💡Sagemakerを使ってみたかった）

✓メリット

- 実例・語彙で“専用AI”にできる
- JumpStartで初学者でも着手しやすい

△デメリット

- 最低限のML知識が必要
- 学習データ作成の負荷が大きい

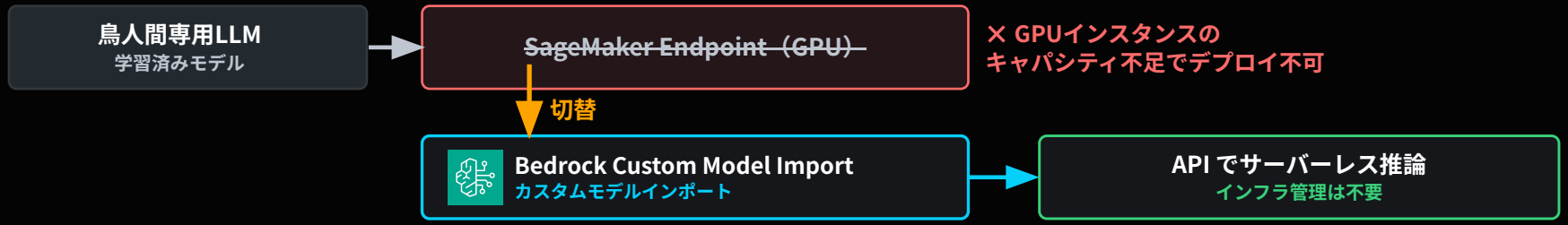
ファインチューニングの仕組み（LoRA）

巨大な“土台”に、薄い層を重ねるだけ

土台は変えず、薄い層（LoRA）を重ねる＝軽量・低コスト

Amazon Bedrock（基盤）

作った“鳥人間AI”を実際に動かす — サーバーレスでAPIデプロイ



選んだ理由 — GPU枠が取れずSageMakerに乗らなかったモデルを、Bedrockがサーバーレスで“そのまま”API提供

✓ メリット

▸ 自動スケール・サーバ管理不要

アクセス状況に応じてモデルが自動でスケール。
サーバ管理不要。

▸ 未使用時のコストを抑制

5分間アクセスが無いと自動でゼロに縮小。
未使用時は課金されない。

△ デメリット　コールドスタート

▸ コールドスタートが発生

停止後の初回リクエストは起動待ちが生じる
（数十秒〜・モデルサイズ次第）

▸ リトライ実装が必要

ModelNotReadyException に備え、リトライ／指数バックオフを実装

まとめ

AWS の 既存サービス　＋　AWSで作ったカスタムAI　＝　放送品質のリアルタイム英語字幕を実現

① 開発してみて

— 良かったこと

- “土台”を簡単に構築
- リアルタイム処理にも対応
- コンテナ開発がおすすめ
  ECR→ECS→Fargate が簡単・快適

② 学び・知見

— 参加者へのシェア

- “組み合わせ”だけでは不足
- IaC で全部コード化が便利

ネットワーク含め CDK／CloudFormation 化で
コマンド1つで一気にデプロイ

③ AWSへの期待

— こんなことできたらいいな

- Fargate の GPU 対応に期待
  現状はGPU部分を ECS on EC2 で対応

- Transcribe の精度向上に期待
  音声・AI系サービスの品質に期待

ご清聴ありがとうございました。