

# クラウドコンピューティング 5 班発表スライド



# 流れ

1. 制作物の紹介
2. 要件
3. システム設計(AWSアーキテクチャダイアグラム)
4. 各部分の説明
5. デモンストレーション
6. まとめ・感想



## SNSでバズっている飲食店のご飯を食べてみたい

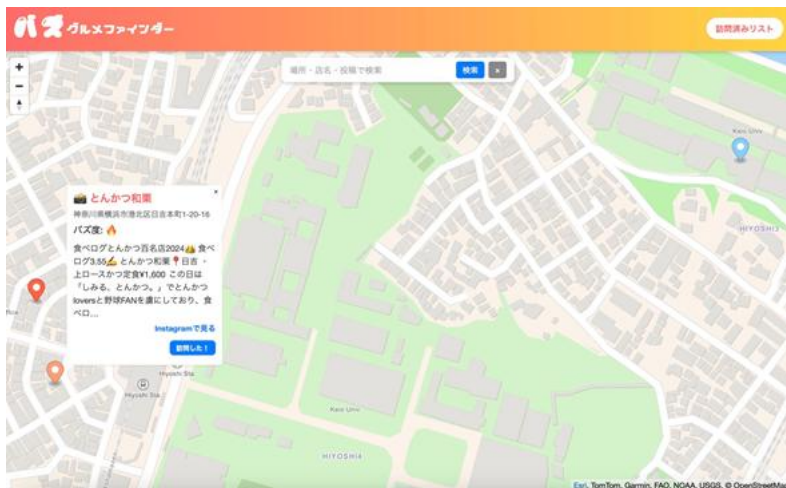
でも、その場で現在地に近いバズっている飲食店を

SNSから直接探すのは至難の業

→SNSでバズったおいしい飲食店で、現在地に近い飲食店を  
見つけて地図上に表示するアプリが欲しい

# 1. 制作物

## SNSでバズったおいしい飲食店を表示するWebアプリケーション





## 2. 要件

位置情報センサで位置を取得して、おすすめの飲食店を地図に提示する

### <メイン>

- ・ SNSでは特にグルメ投稿が盛んなYoutubeとInstagramを想定する
- ・ 最も近い飲食店は強調しつつ、他にも探せる状況を保持したい
- ・ 外部でもある程度速い接続ができるようにしたい

### <サブ>

- ・ 負の方向でバズっている情報はいらないので、表示しない 等





## 4. 各部分の説明

## 部分①SNSからのデータ抽出、バズ度を算出

- ・ InstagramやYouTubeのAPIを利用して、投稿名や投稿文、いいね数等を取得し、各SNSにて異なる項目や式からバズ度(Buzz)を算出

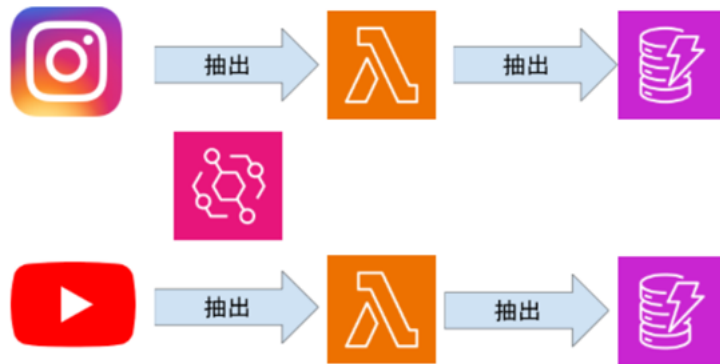
→数値の範囲が5段階評価で同じような分布になるように調整

- ・ 各データベースに保存

→プライマリーキーを投稿IDとして設定して、  
同一のバズった情報がDBに追加されることを防ぐ

- ・ 上記の内容をEventBridgeでスケジュール実行

→7:00, 11:00, 18:00に実行





## 部分①-a. Instagramからのデータ抽出、バズ度を算出

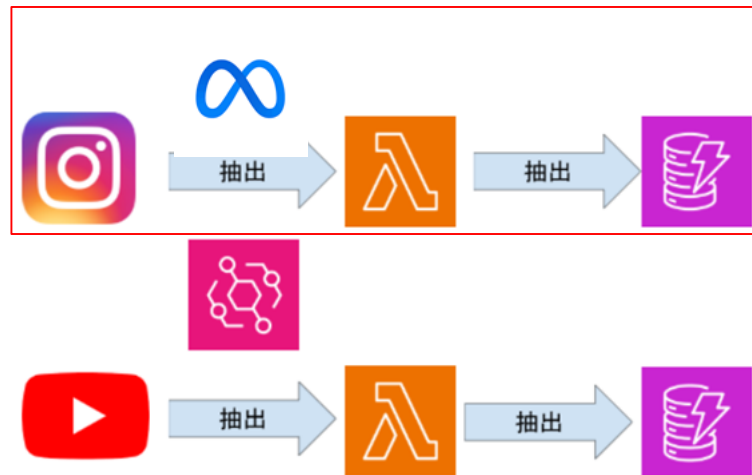
- ・ Meta for Developers のAPIを利用して、投稿名や投稿文、いいね数等を取得
- ・ Instagramのバズ度を求める式：

$$\text{Buzz} = \text{いいね数} / \text{経過時間(h)} / 100$$

(Min, Max等で1~5に補完)



$$\text{buzz\_score} = \max(1, \min(5, \text{buzz\_score}))$$



## 部分①-b. YouTubeからのデータ抽出、バズ度を算出

- ・ YouTube Data API v3 を利用して、高評価数、再生回数、登録者等のjsonデータを取得

→google cloud console上で1日10000リソースの制限あり

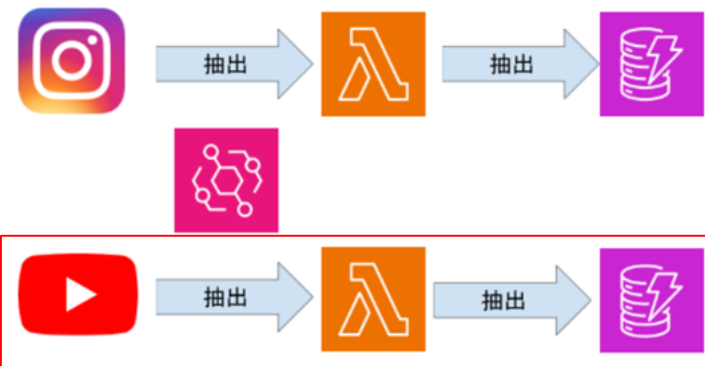
- ・ Youtubeのバズ度は数値的に求める

→ $\text{Buzz} = 100 * \text{再生回数} / (\text{登録者数} * \text{経過時間(h)})$

- ・ videoDuration="shorts"でバズったshorts動画に限定

→ $\text{buzz} > 0.01$ でフィルタリング

→ $\text{buzz\_score} = \max(1, \min(5, \text{buzz\_score}))$



## 部分①-b. YouTubeからのデータ抽出、バズ度を算出

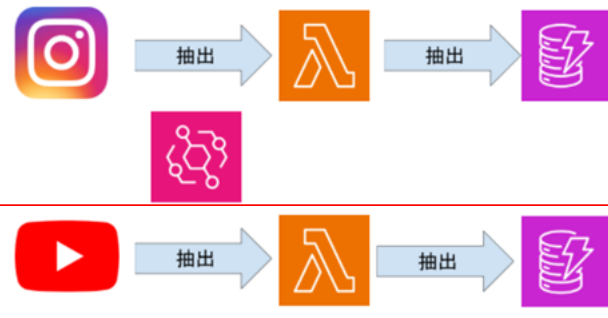
### できなかったこと

- ・バズっている動画のデータは取得できるが、負の方向のバズも混ざる（例：接客が悪い、汚い、まずい）

- ・ネット上の口コミを参照してフィルタリングすることで、良好な飲食店のみをDBに入りたい

→外部APIで使えるものが少ない（規約が厳しい、有料）

→住所や店名等の必須情報が揃っている動画は良好な傾向があり、今回の実装では問題が見られなかった



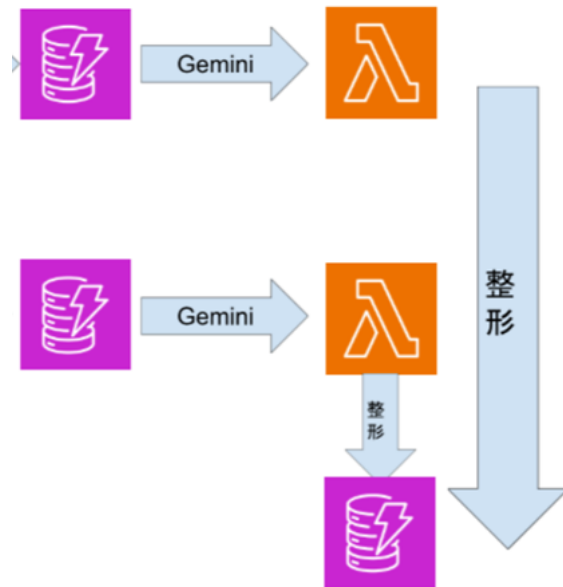
## 部分②抽出したデータから更に店名と住所を抽出

Gemini で文から店名と住所を抽出

"あなたは、Youtube / Instagramのグルメキャプションから  
店舗情報をJSON形式で抽出する専門家です。"

必須情報が見つからない → Null

(以降のパイプラインに参与しない)

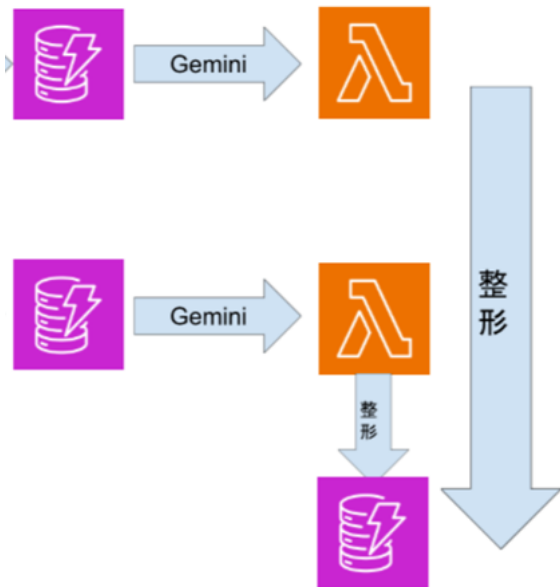


## 部分②抽出したデータから更に店名と住所を抽出

lambdaとDBはまとめることもできたが、わざと分けた

→Gemini のRequests Per Minuteの制限を避ける

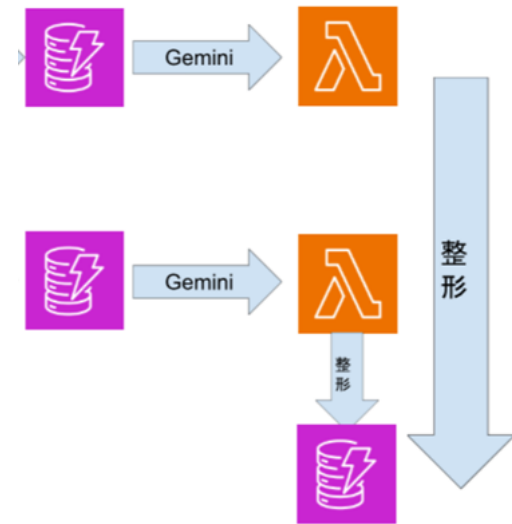
→制限にかかると、パイプラインが滞るため



## 部分②抽出したデータから更に店名と住所を抽出

### できなかったこと

- ・各SNSから同じ店に関する同じクリエイターの動画が入ってしまいう可能性を排除できなかった  
(発生する可能性は低い)



## 部分③住所から緯度経度を算出して保存

Amazon Location Serviceで、住所から緯度経度を算出し、新しいDBに保存

- ・ Amazon Location Service以外でもライブラリや他のAPIでも可能だが、精度が悪く、理想的な店舗の位置の表示にはならなかった
- ・ Web側での呼び出し時に変換すると、ピンの表示までに時間がかかり使いにくい

→全体のアーキテクチャを通して、重たい処理はバッチ処理でDBへの書き込み時に保存するように設計した

ここで、バックエンド側の操作は終了する

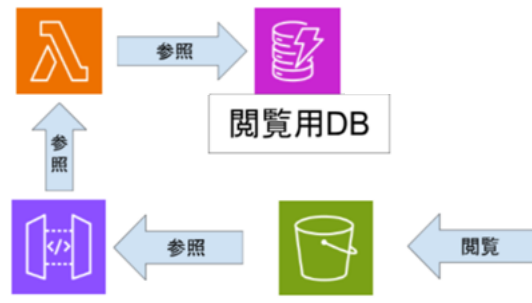


## 部分④Web側

- ・ S3を使って、webページを公開
- ・ ページを読み込んだ際にAPI Gatewayを通して、Lambda関数を使い整形されたデータベースにアクセスし、店の情報を得る

- ・ 現在の位置情報を、ブラウザが提供する

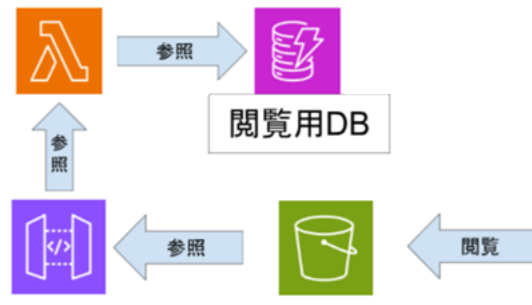
JavaScript APIのGeolocation APIから  
読み取って、緯度と経度から店との距離を計算し、  
1番近いバズった店をポップアップ表示





## 部分④Web側

- ・ バズ度ごとにピンの色を変える
- ・ 店名、住所、投稿から店を検索し、表示できるようにする
- ・ ある程度(現在の設定では半径約3km以内)店に近づくと、  
訪問記録をつけることができるようになる
- ・ 訪問した店のリストは、右側の訪問済み  
リストの欄から見られるようになる
- ・ リストの店をクリックしても、その店に飛ぶことができる





## 6. まとめ・感想

AWSに加えてSNSやGemini のAPIなどを活用しておおよそ要件に沿った制作物が完成した。各トークンの制限を考えて作成するのが難しかった。

あまり「エッジの情報を活用する」ようなシステムにはできなかった。画像とかもう少しデータを活用できたらもっとよかったと思う

色々試行錯誤が必要で大変だったが、結果的に作りたいものを作ることができて良かった。マップの表示の部分と、データベースからのデータの取得には特に苦労した。

外部サービスから得たデータの管理と解読が大変だった。



ご清聴ありがとうございました