
インフラとバックエンド
とフロントエンドをくま
なく調べて遅いアプリを
早くした件
Optimizing a Slow App: A Deep Dive into Infrastructure,
Backend, and Frontend
YU OTSUBO

初手謝罪

原因はAWSではありませんでした...。

NW-JAWSなのにすみません。

00 自己紹介

名前	Yu Otsubo
所属	KDDIアジャイル開発センター(KAG)
職種	ゆるふわエンジニア
SNS	@meitante1conan
その他	妻と二人暮らしでしたが 最近犬をお迎えしました！ なので最近はエンジニアリングのつぶやき少なめ

本日のアジェンダ

1. どんな問題が発生したの？
2. CloudFrontが返す謎のエラー
3. CloudWatch Application Signalsでボトルネックを調べる
4. ごん...おまえだったのか🦊
5. 最後に

01

どんな問題が発生したの？

ファイルのアップロードに失敗するよ....?

01 どんな問題が発生したの？

どんなアプリを開発しているの？

PDFとかパワポのファイルを**アップロードして**、
非常に重たい処理並列で実行、結果を画面に表示させる
業務アプリケーションです。

この場で詳細話せないのが悔しい...。

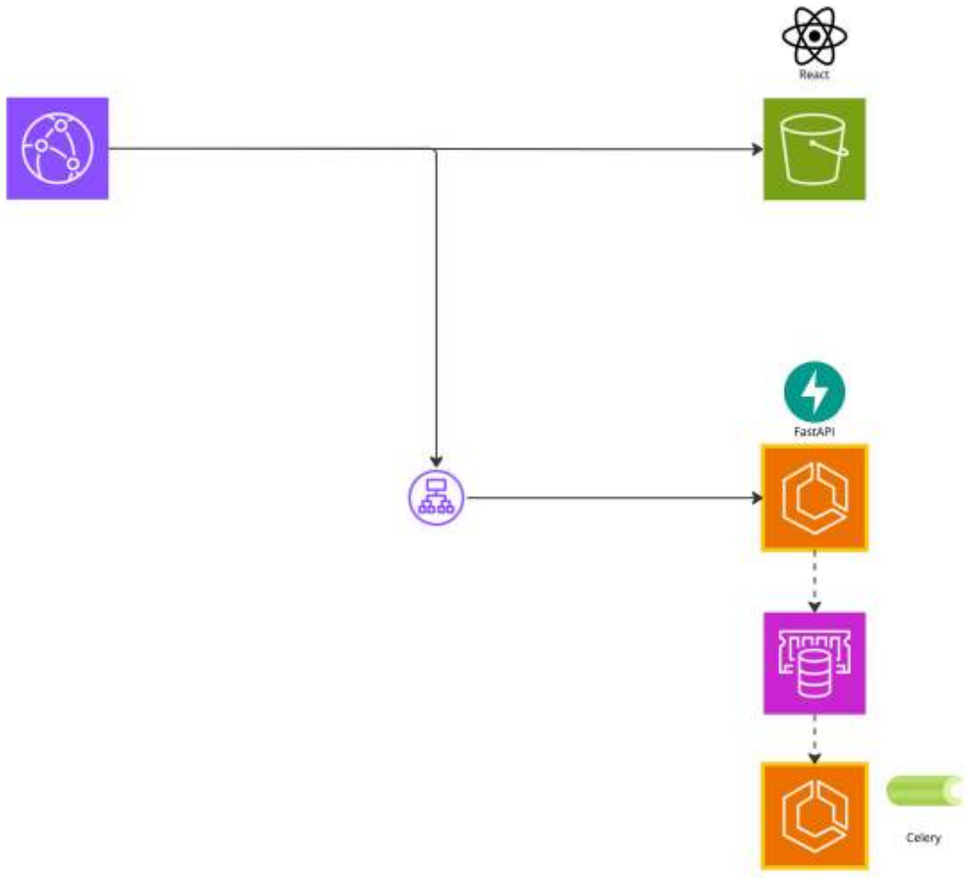
01 どんな問題が発生したの？

ファイルアップロードが
待っても完了しないよ！

開発チームで確認したところ確かにAWS環境で再現。

ただし、**ローカルでは発生しない**。

「ペロッこれはインフラ起因！」



01 どんな問題が発生したの？

簡単な構成をおさらいしましょう

ちょっと複雑な構成ですが、追って行きましょう！

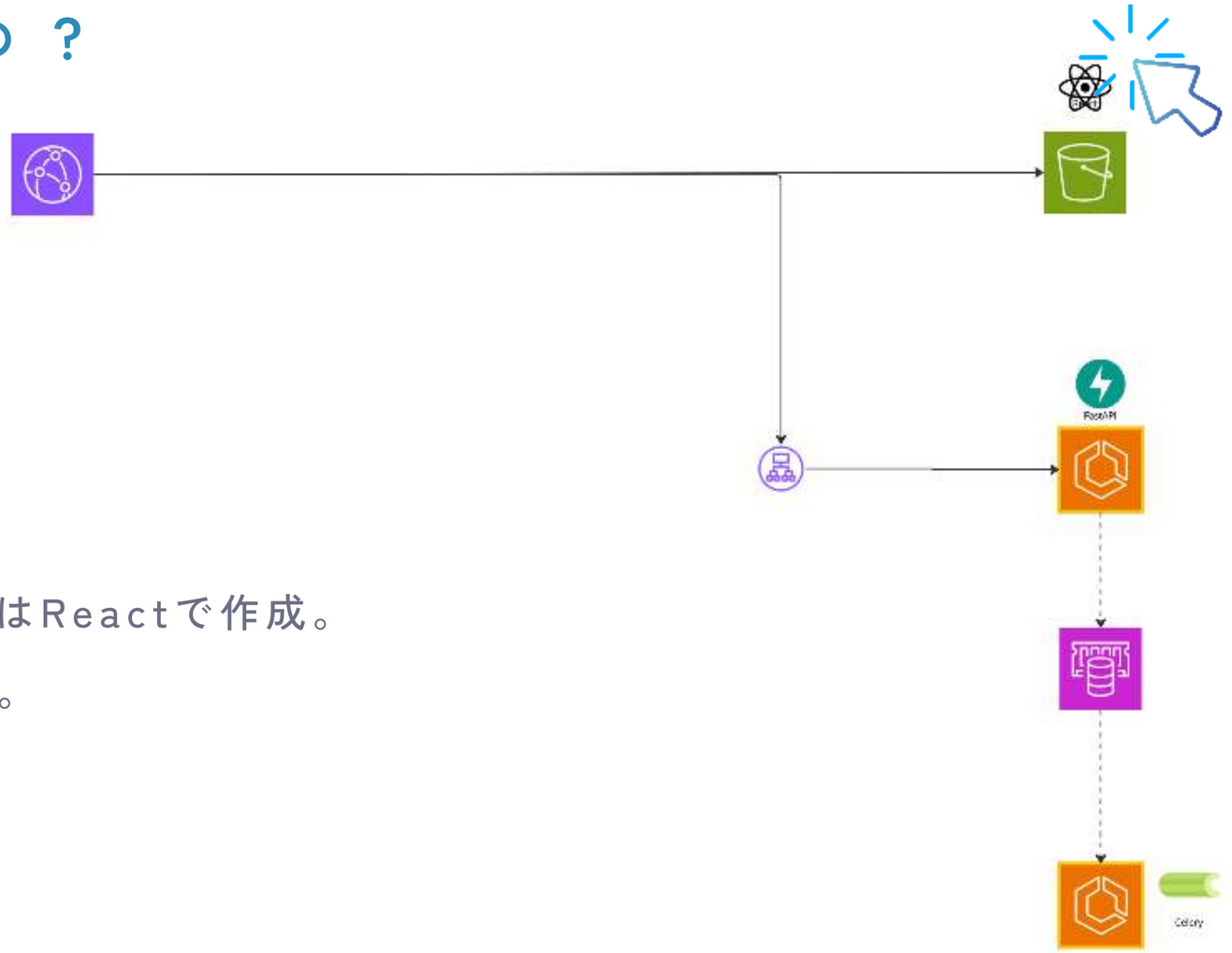
みなさんもどこが原因か想像しながらお聞きください！

01 どんな問題が発生したの？

CloudFrontでフロントエンドアプリケーションを配信

ビヘイビアでバックエンドAPIにルーティングしている

01 どんな問題が発生したの？

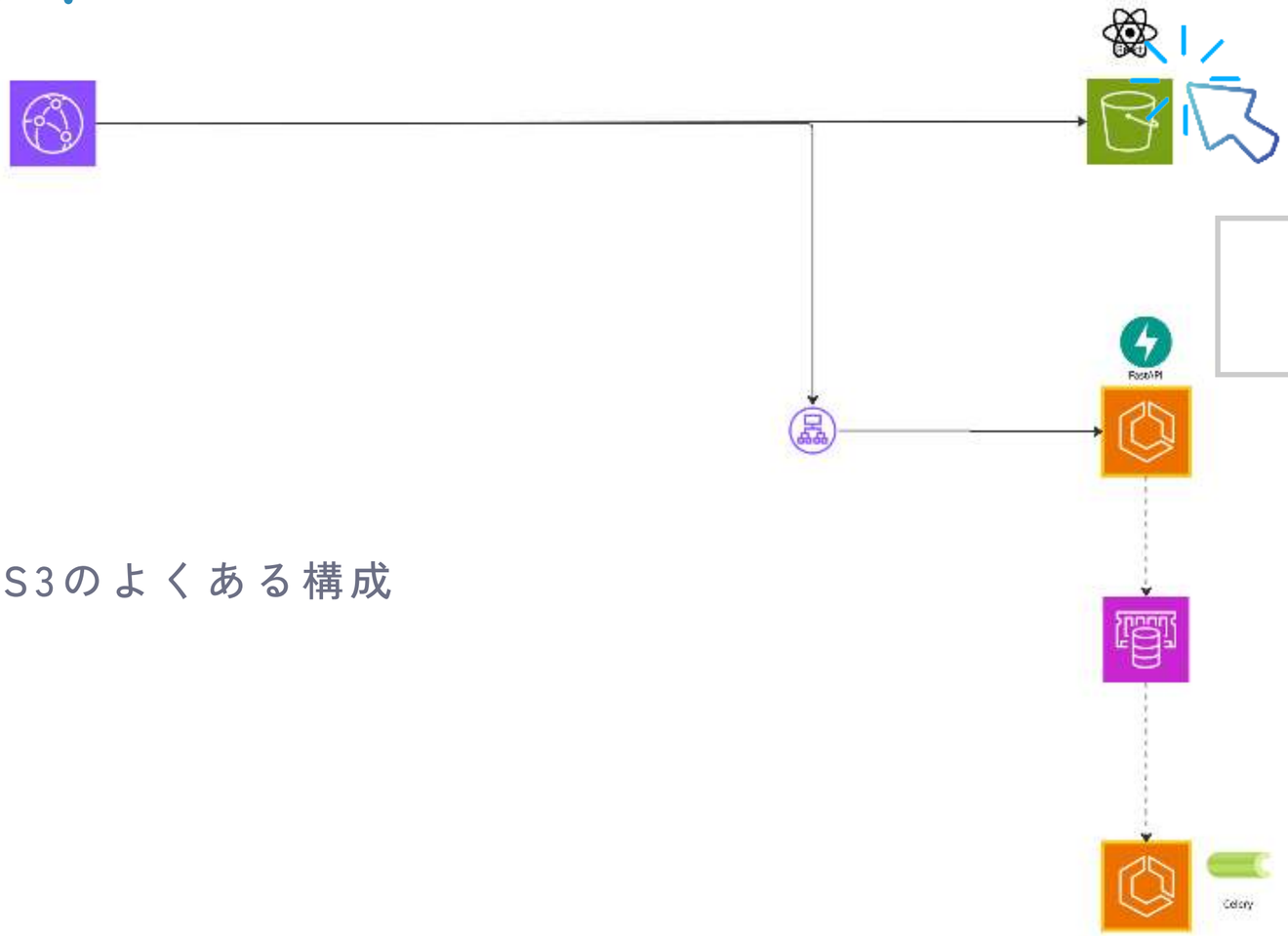


フロントはReact

フロントエンドはReactで作成。

SPAで動きます。

01 どんな問題が発生したの？



S3でホスティング

CloudFront ＋ S3のよくある構成

01 どんな問題が発生したの？

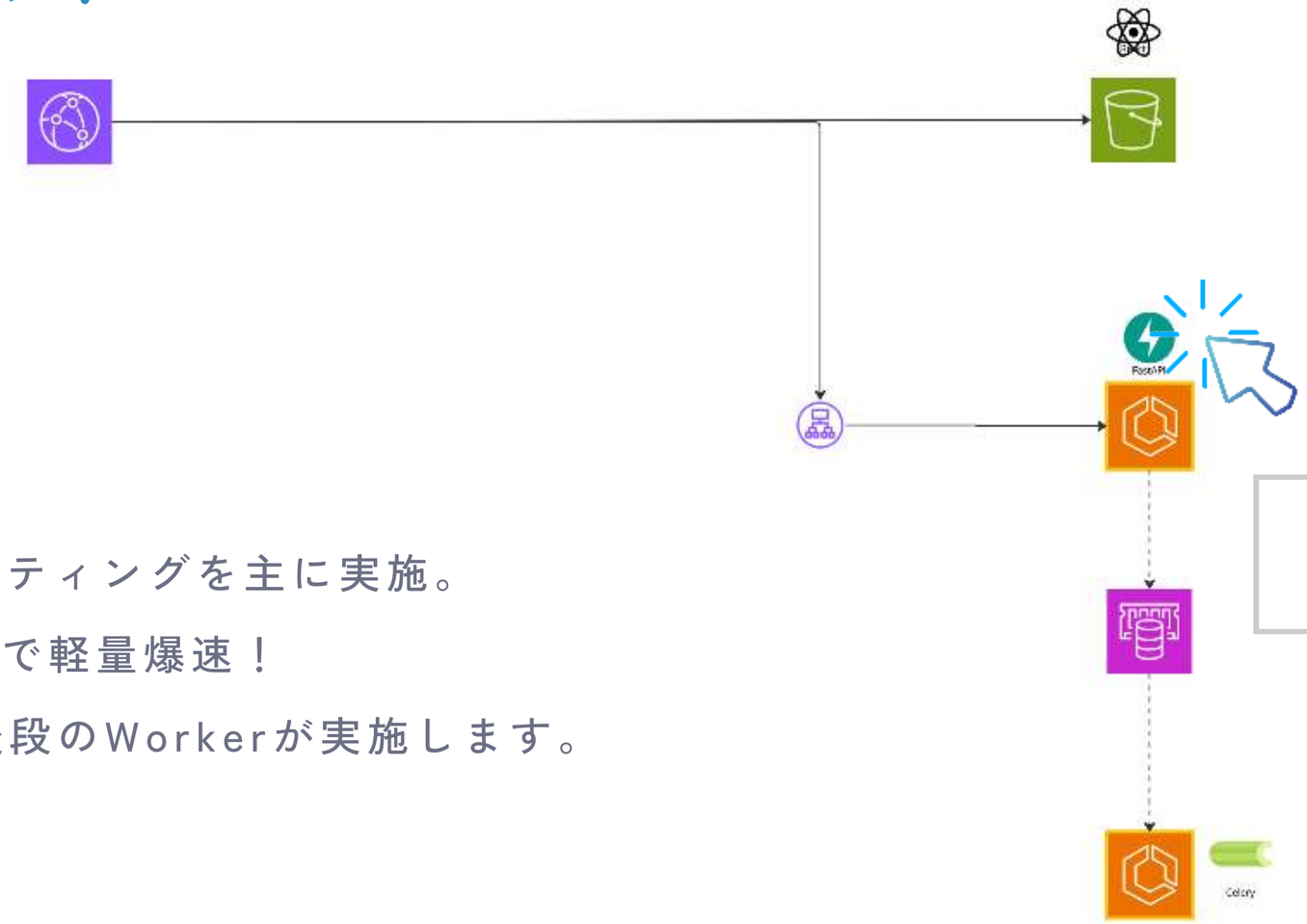
FastAPIへのアクセスのフロント面です。

カスタムヘッダー＋

マネージドプレフィックスリストでセキュアに

ALB

01 どんな問題が発生したの？



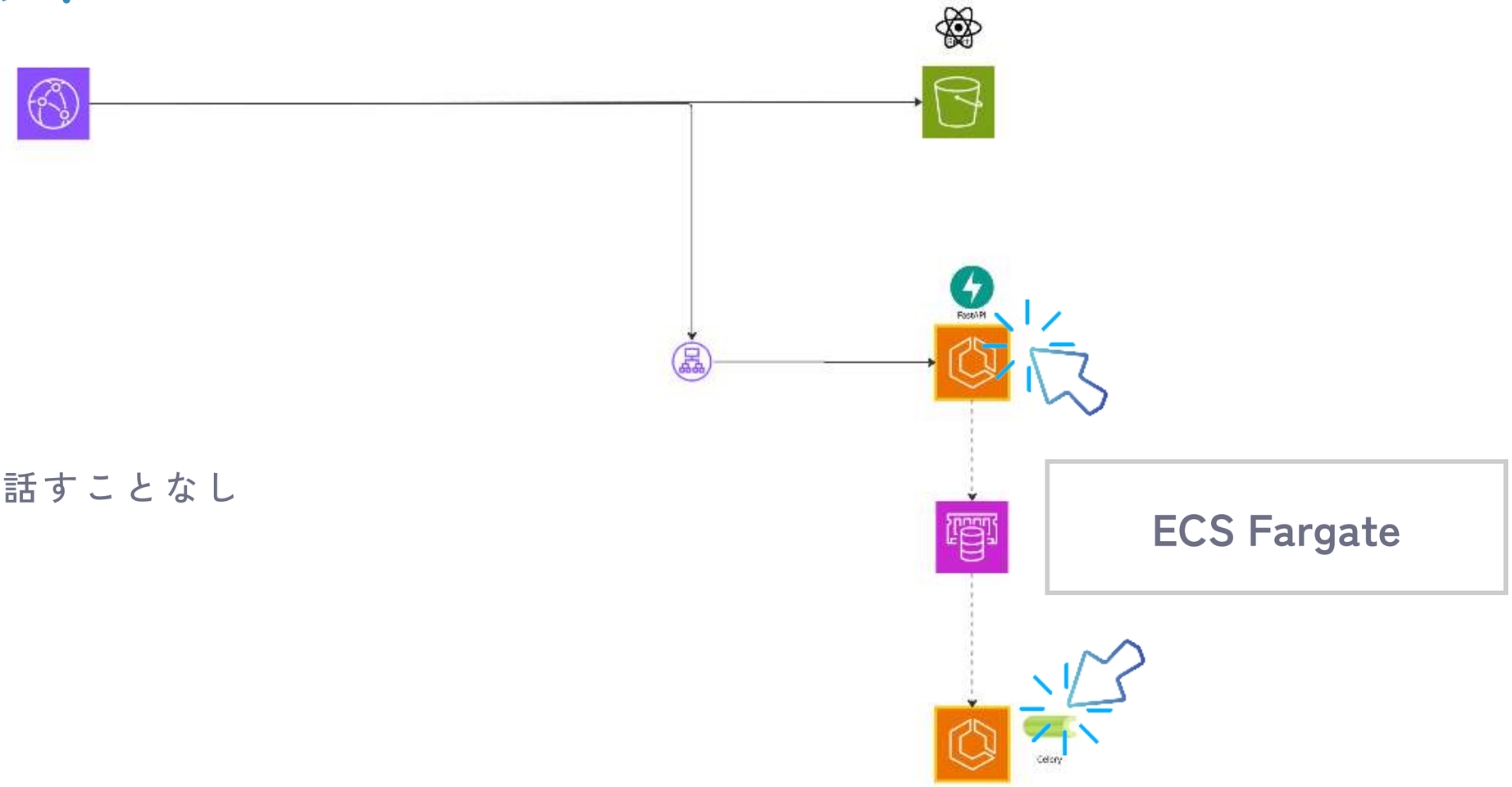
バックエンドは
FastAPI

FastAPIはルーティングを主に実施。

Uvicornのお陰で軽量爆速！

重たい処理は後段のWorkerが実施します。

01 どんな問題が発生したの？



特に取り立てて話すことなし

01 どんな問題が発生したの？

CeleryのBroker/Result Backendとして利用しています。

Result Backendを兼ねることができるので地味に便利。

RedisでWorkerへ
Job登録

01 どんな問題が発生したの？

Celeryのコーディング体験半端なくいいですよ！！！

普通のPython書いたら即Worker化できます。おすすめ。

一つあたりのWorker処理はとても小さいのでThread Poolを

Geventで設定

CeleryでWorker構築
重たい処理はこちらで

02

CloudFrontが返す謎のエラー

どうやらCloudFrontのバックエンドがタイムアウトしているらしい...。つまり...?

02 CloudFrontが返す謎のエラー

ファイルが重たいから...?
コンテナが小さいから...?

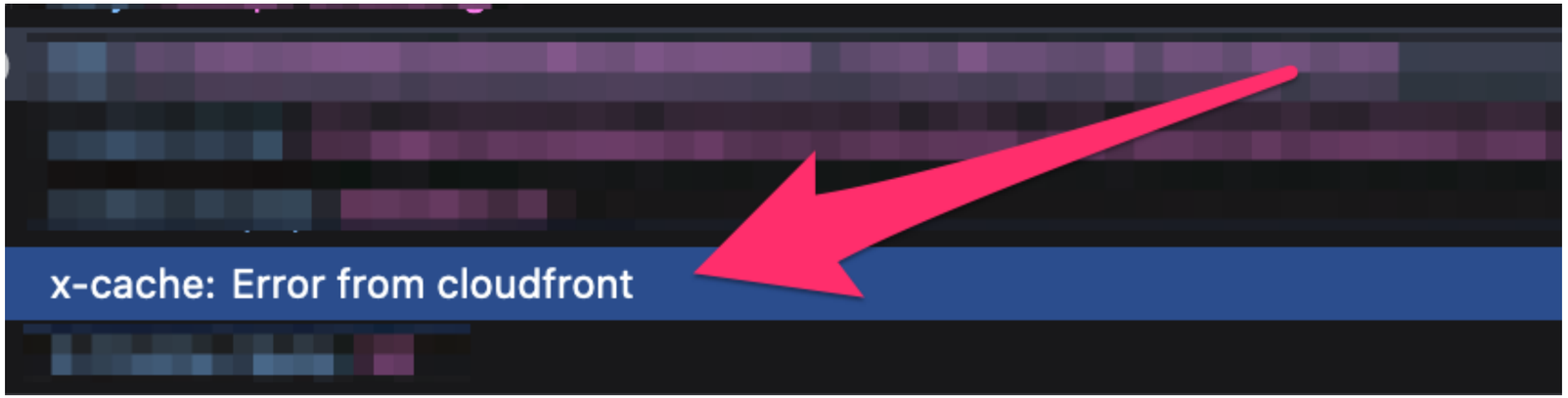
DevtoolsでAPIのレスポンスを見てみよう...

「ん！？」

02 CloudFrontが返す謎のエラー

CloudFrontがエラーを吐いている！

Error from cloudfrontが出ている！！！！



02 CloudFrontが返す謎のエラー

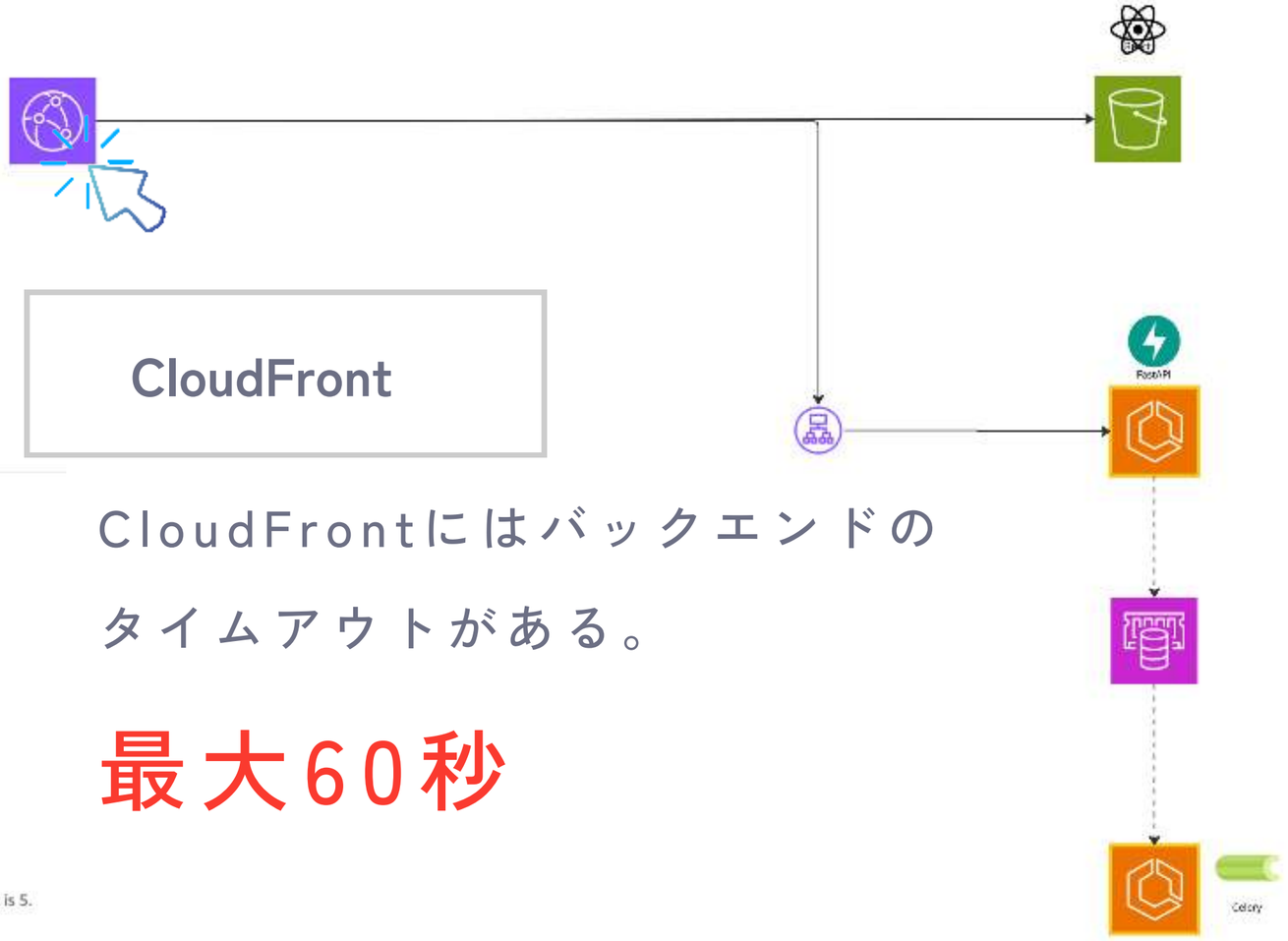
CloudFrontがエラーを吐いている！

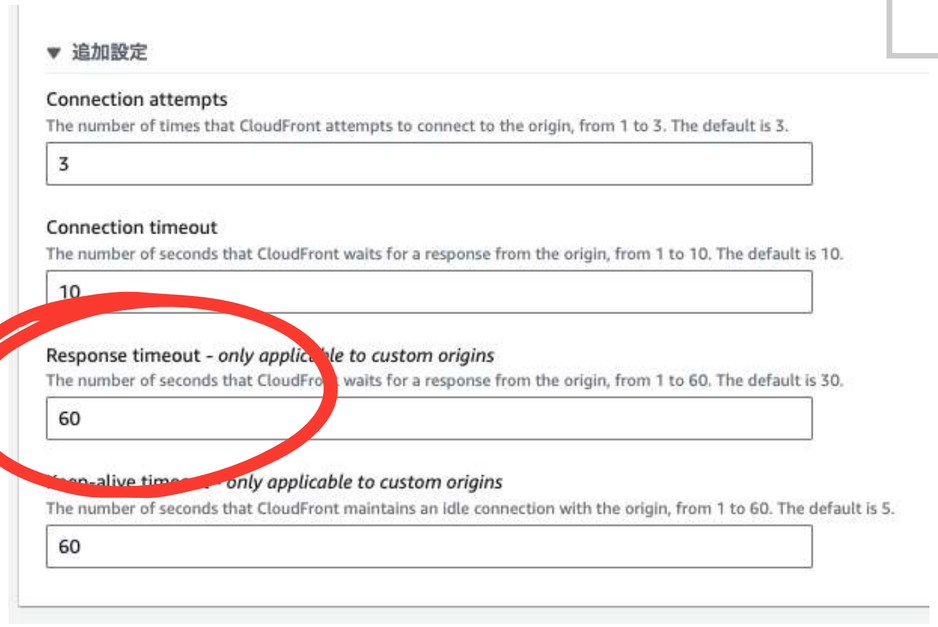
でも特にアプリのログにエラーは出てない。。。

なにかタイムアウトに引っかかっている？

インフラ臭いですね。

02 CloudFrontが返す謎のエラー





CloudFrontにはバックエンドのタイムアウトがある。

最大60秒

これですね....。

02 CloudFrontが返す謎のエラー

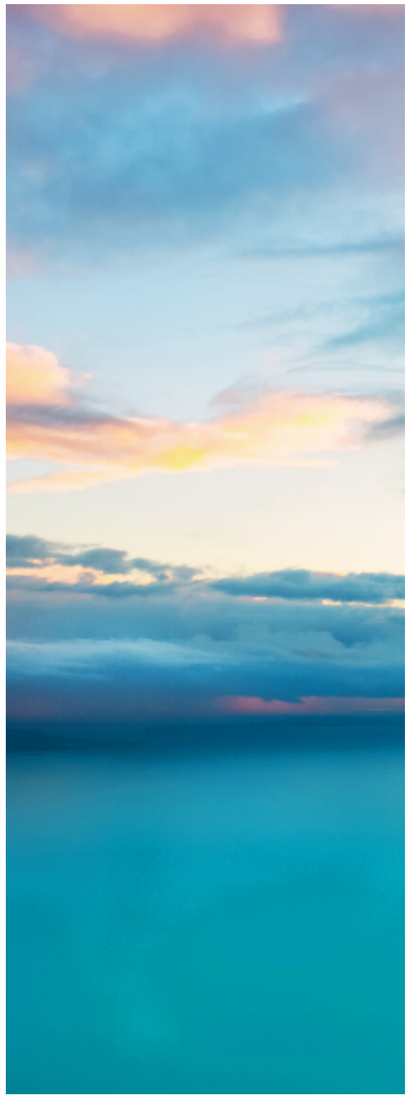
ALBの設定を見直してみる

接続アイドルタイムアウト
60 秒

一方、ALBにもタイムアウトはあるが、

最大4000秒に伸ばしても変わらず。

ALBのヘルスチェックも問題ない...



02 CloudFrontが返す謎のエラー

何かがとても遅くて
CloudFrontで60秒タイムアウト

ボトルネック調査！！

私はストレートネック

03

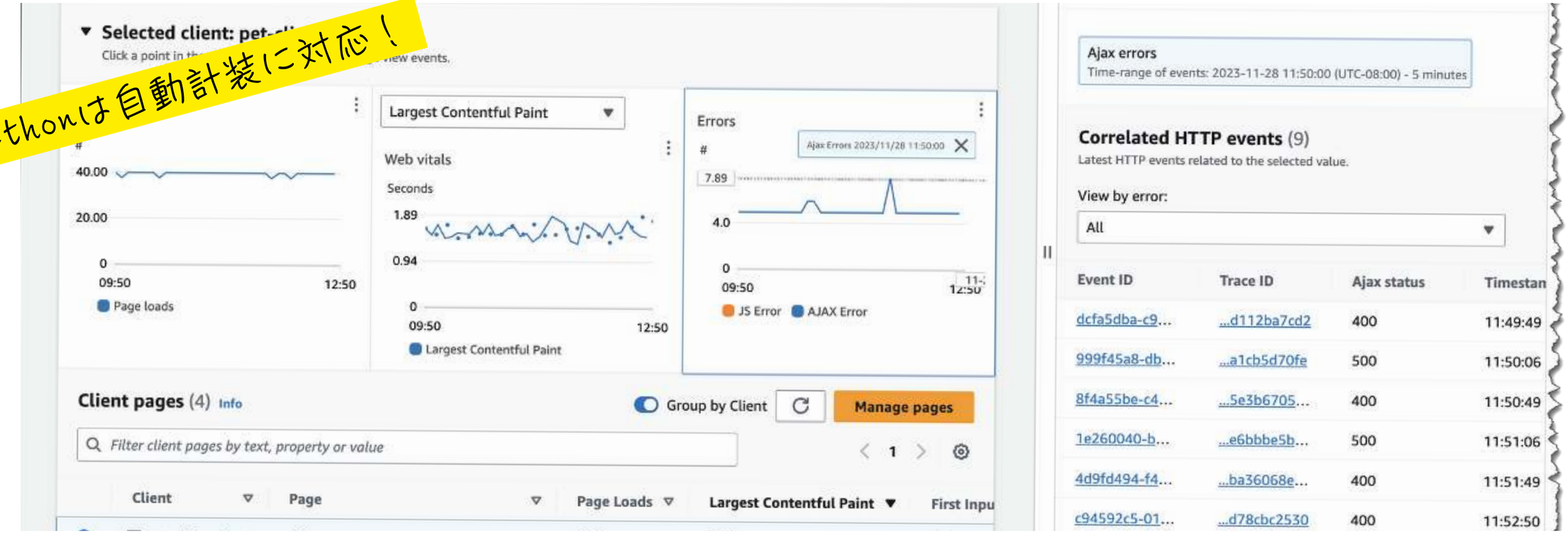
CloudWatch Application Signalsでボトルネックを調べる

おや...?他のAPIも遅いぞ...? つまり...?

03 CloudWatch Application Signalsでボトルネックを調べる

アプリケーション全体のトレースするときに便利

OpenTelemetry(OTEL)を活用してAWSのアプリケーションモニタリングを自動化する新しいサービス



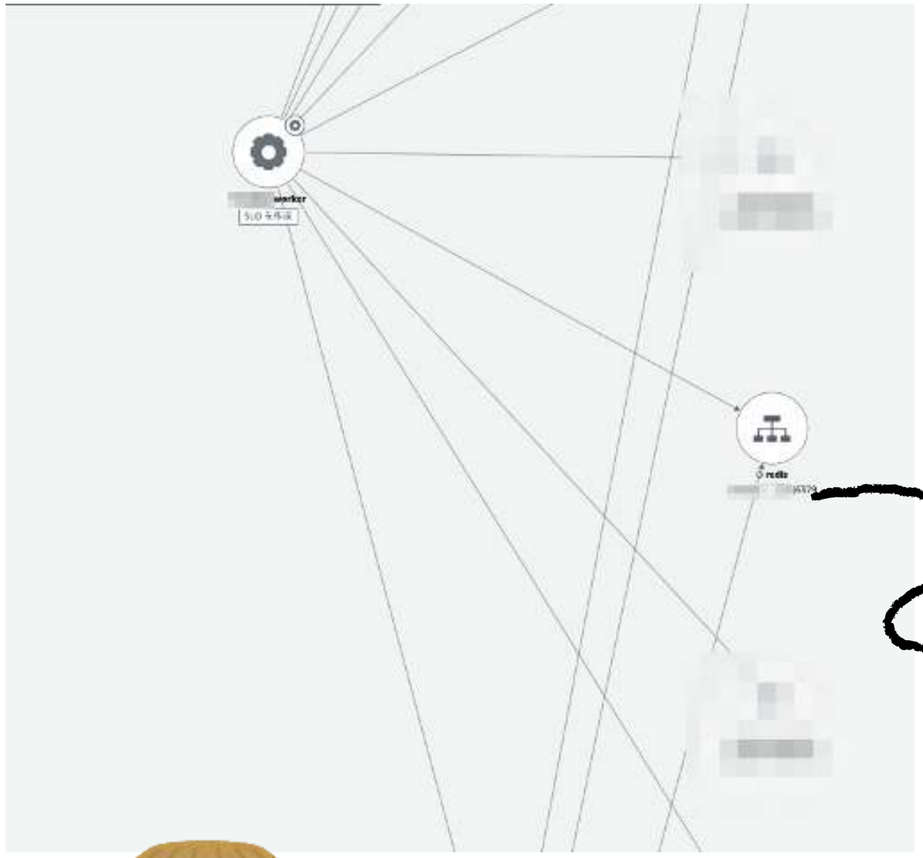
03 CloudWatch Application Signalsでボトルネックを調べる

アップロードだけがタイムアウトすると思ったら全APIが遅い...!!!

	名前	SLI ステータス	依存関係	レイテンシ...
◉	POST /api/[redacted]	SLO を作成	-	180.0K ms
○	POST /api/[redacted]	SLO を作成	6	156.7K ms
○	[redacted]	SLO を作成	2	-
○	[redacted]	SLO を作成	14	[redacted]
○	[redacted]	SLO を作成	3	[redacted]
○	[redacted]	SLO を作成	-	
○	GET /api/healthz	SLO を作成	-	7.6K ms
○	[redacted]	SLO を作成	-	[redacted]

明らかに遅い。

単純なstatusしか返さないhealthcheckのAPIですら数秒かかっている。



03 CloudWatch Application Signalsでボトルネックを調べる

逆にredis/workerのスピードは早い。

ということは被疑箇所はFastAPIだ！

こうして調査の旅はアプリケーション内部へ進んだ...

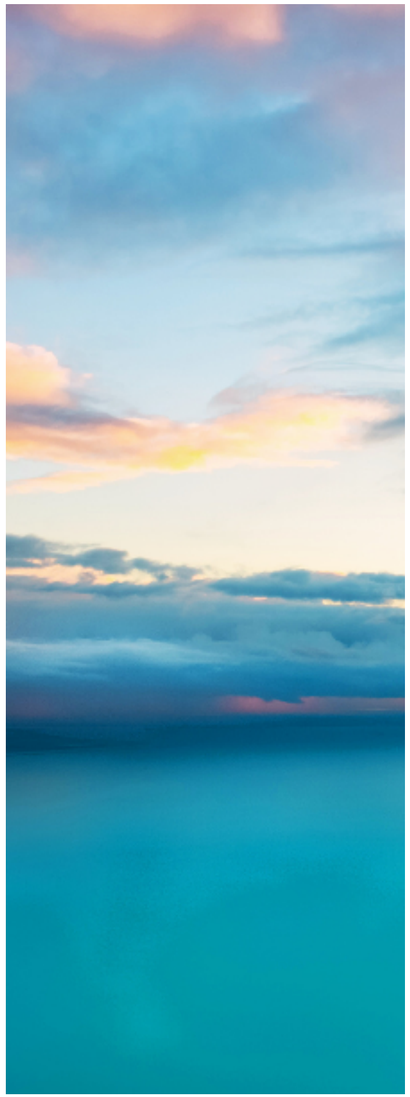
	依存関係	リモートオペレーション	目標	レイテンシ...	レイテンシ...	レイテンシ...	リクエ...
◉	redis	GET	celery-backend.y	12.1 ms	1.4 ms	0.9 ms	2.4K
○	redis	SET	-celery-backend.y	1.6 ms	1.6 ms	0.9 ms	2
○	redis	UnknownRemoteOperation	-celery-backend.y	13.3 ms	1.6 ms	1 ms	7.9K

04

ごん...おまえだったのか🦊

私のごんは栗を持ってこず、バグを持ってきました。


改めて謝罪
原因はAWSではありませんでした...。
NW-JAWSなのにすみません。



04 ごん...おまえだったのか🦊

ジーイベント

Gevent、お前だったのか

gevent/**gevent**

Coroutine-based concurrency library for Python

100 Contributors　133k Used by　6k Stars　938 Forks

04 ごん...おまえだったのか🦊

ジーイベント
Geventとはなにか？

Geventは、Pythonで非同期プログラミングを実現するためのネットワークライブラリで、
Greenletという軽量な協調的マルチスレッド(コルーチン)を使用します。

I/O待ちの多い処理では効率的に処理をこなせます。(CPU負荷の低いやつ)
GeventはCeleryのスレッドプールで選択可能です。

04 ごん...おまえだったのか🦊

Geventとはなにか？

geventは、Pythonで非同期プログラミングを実現するためのネットワークライブラリで、
greenletという軽量な協調的マルチスレッド(コルーチン)を使用します。

要は小さい処理なら
Celery Workerを高密度にできますよ
というもの

04 ごん...おまえだったのか🦊

Geventとはなにか？

geventは、Pythonで非同期プログラミングを実現するためのネットワークライブラリで、
greenletという軽量な協調的マルチスレッド（コルーチン）を使用します。

コスト削減のため導入してました

要は重たい処理なら
Celery Workerを高密度にできますよ
というもの

04 ごん...おまえだったのか🦊

Seems to be in conflict with gevent monkey.patch_all() #4873

Closed | 9 tasks done | GoGsxl opened this issue on May 8, 2022 · 3 comments

Celery Workerの高密度化のために導入したGevent

04 ごん...おまえだったのか🦊

Seems to be in conflict with gevent monkey.patch_all() #4873

Closed | 9 tasks done | GoGsxl opened this issue on May 8, 2022 · 3 comments

Celery Workerの高密度化のために導入したGevent

↓

FastAPIのコンテナにもpipでGeventインストール

04 ごん...おまえだったのか🦊

Seems to be in conflict with gevent monkey.patch_all() #4873

Closed | 9 tasks done | GoGsxl opened this issue on May 8, 2022 · 3 comments

Celery Workerの高密度化のために導入したGevent

↓

FastAPIのコンテナにもpipでGeventインストール

↓

Geventのmonkey.patch_allがライブラリ起因で適用

非同期・Threadingの標準ライブラリがgreenlet処理に置き換え

04 ごん...おまえだったのか🦊

Seems to be in conflict with gevent monkey.patch_all() #4873

Closed | 9 tasks done | GoGsxl opened this issue on May 8, 2022 · 3 comments

Celery Workerの高密度化のために導入したGevent

↓

FastAPIのコンテナにもpipでGeventインストール

↓

Geventのmonkey.patch_allがライブラリ起因で適用

非同期・Threadingの標準ライブラリがgreenlet処理に置き換え

↓

Uvicornと競合！！　イベントループに変な形で割り込み！

04 ごん...おまえだったのか🦊

Seems to be in conflict with gevent monkey.patch_all() #4873

Closed 9

Issueも上がっていました。

Celery Workerの高密度化のために導入したGevent

↓

FastAPIのコンテナにもpipでGeventインストール

↓

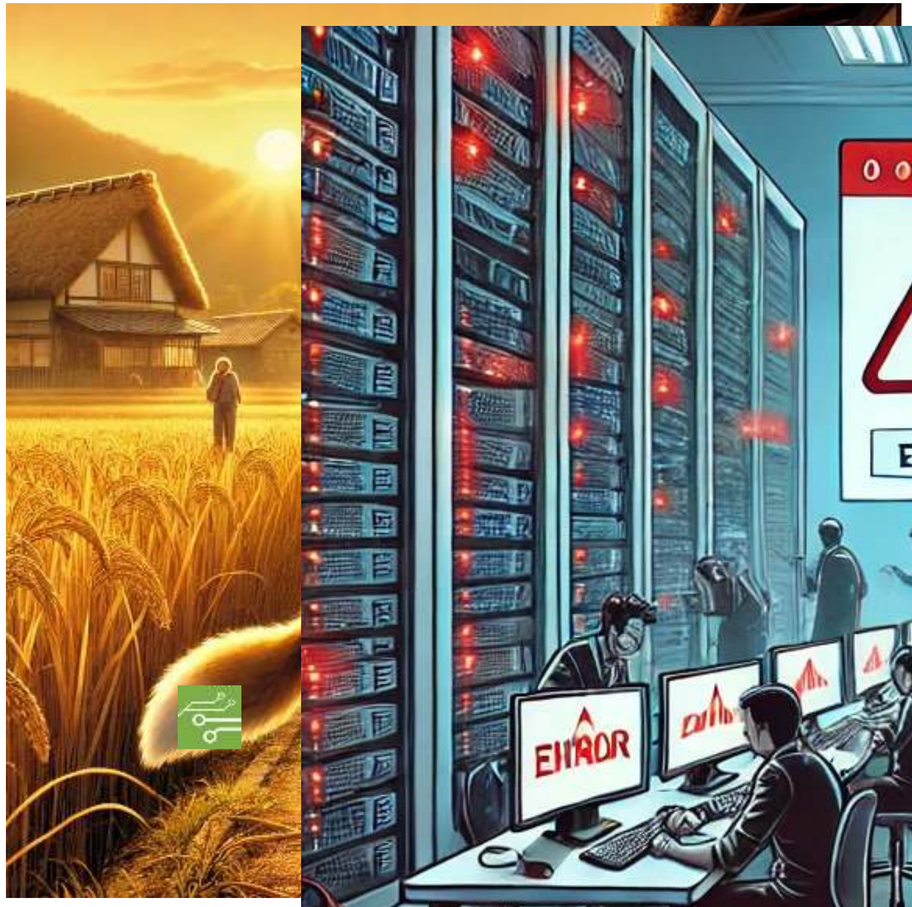
Geventのmonkey.patch_allがライブラリ起因で適用

非同期・Threadingの標準ライブラリがgreenlet処理に置き換え

↓

Uvicornと競合！！　イベントループに変な形で割り込み！

04 ごん...おまえだったのか🦊



ch_all() #4873

Gevent

ストール

適用

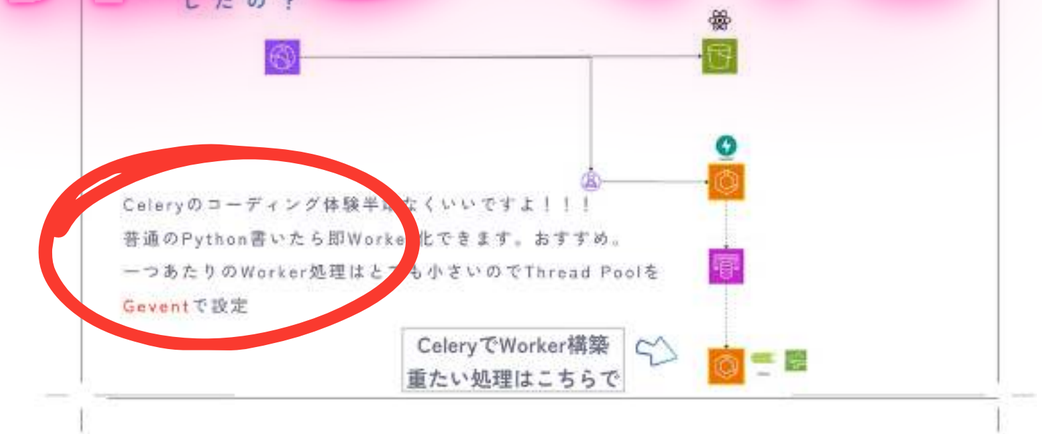
処理に置き換え

刈り込み！

04 ごん...おまえだったのか🦊

04 ごん...おまえだったのか🦊

めっちゃ推してた



04 ごん...おまえだったのか🦊

解決方法

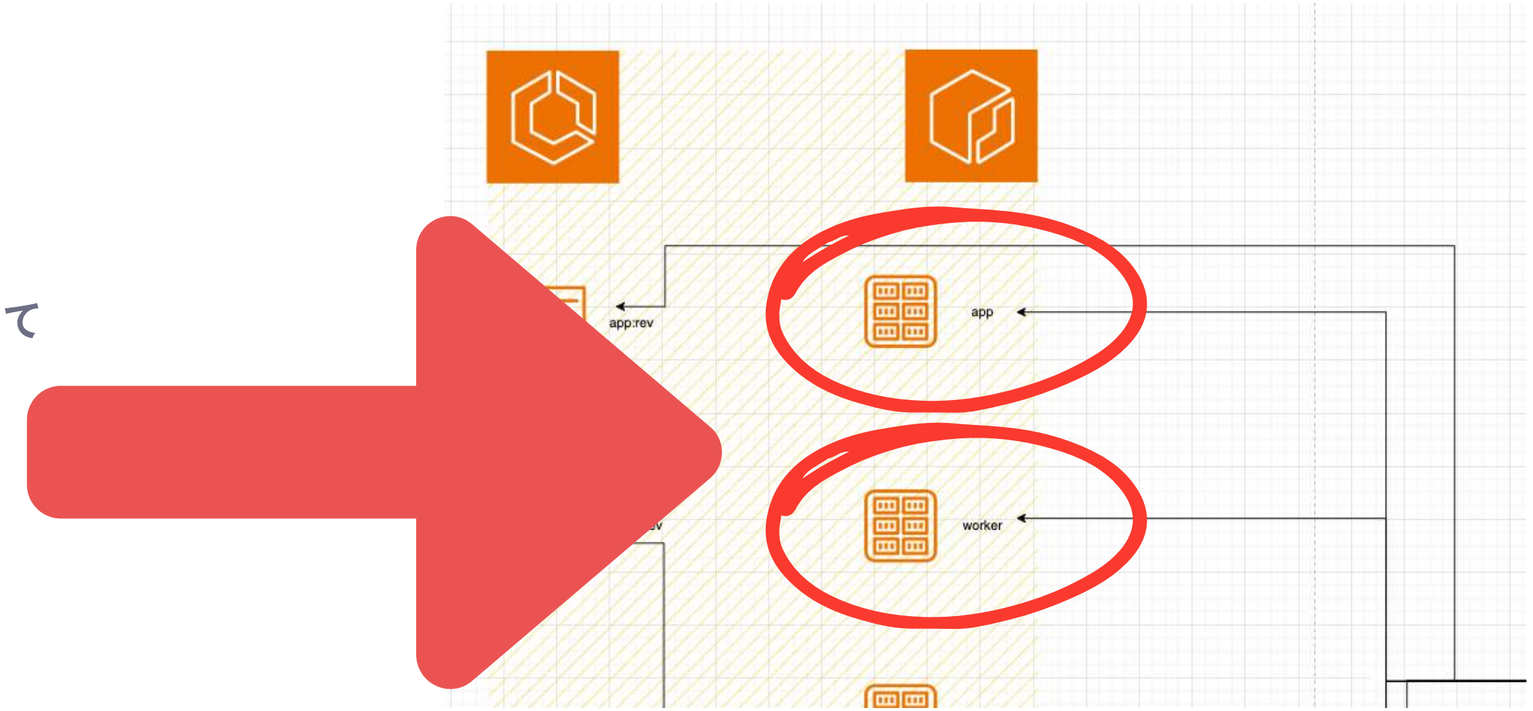
Dockerfileを２つに分けて

それぞれ

- geventあり
- geventなし

で作成した。

たったこれだけ...。



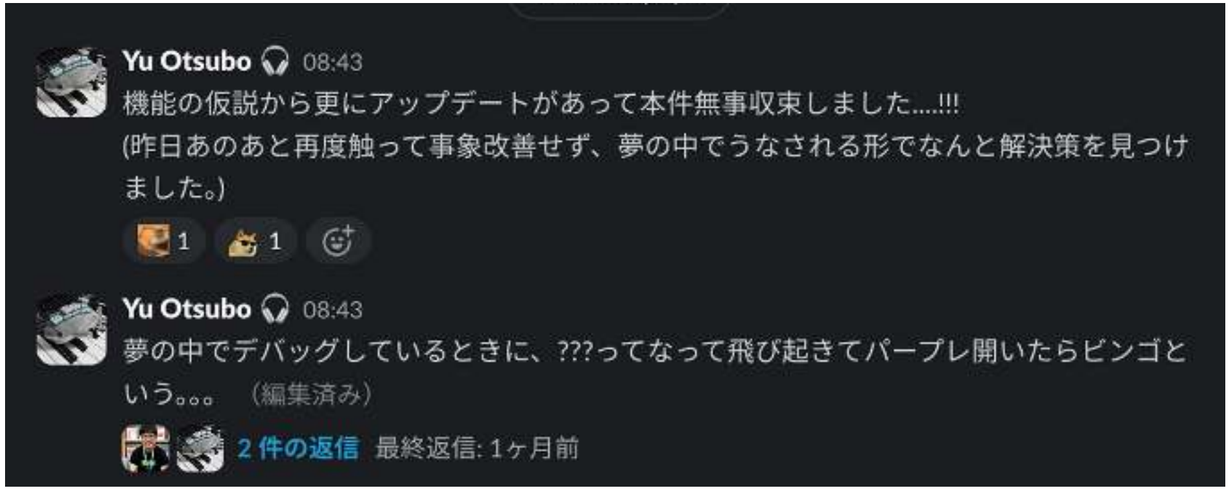
※細かい話ですが、MacだとGreenletの動きがまたちょっと動きが違うようで、ローカルだと再現しなかったようです。

04 ごん...おまえだったのか🦊

原因わからず
夢でデバッグ

夜遅くまで調査して、
それでもわからなくて、
悔しくて、
寝ながらデバッグして、
朝起きてperplexityに聞いて
解決。

喜びすぎて変なタイポしてますね。



05 最後に

難しいトラブルシュートは、アプリインフラ両方を行ったり来たりして調べるべし。

それぞれに原因がありそうで、調査投げてしまうと時間がかかってしまう！！

05 最後に

難しいトラブルシュートは、アプリインフラ両方を
行ったり来たりして調べるべし。

それぞれに原因がありそうで、調査投げてしまうと
時間がかかってしまう！！

05 最後に

難しいトラブルシュートは、アプリインフラ両方を
行ったり来たりして調べるべし。

それぞれに原因がありそうで、調査投げてしまうと
時間がかかってしまう！！

なろう！フルスタックエンジニア

私も精進します。。。

05 最後に

インフラとバックエンドとフロントエンドをくまなく調べて遅いアプリを早くした件

05 最後に

だいぶ

ちょっと

ちょっと

インフラとバックエンドと
フロントエンドを
調べて遅いアプリを早くし
た件

ちょっと

多少


THANK YOU
ご清聴ありがとうございました。