LANGFUSE

Langfuse v3 使ってますか？

Langfuse
Night #1

2025

YU OTSUBO

ABOUT ME

Yu Otsubo

ゆるふわエンジニア

KDDIアジャイル開発センター所属

生成AIを活用したプロジェクトでゆるふわエンジニア

妻と犬と細々暮らしている

ちなみに、志田未来さんと神木隆之介さん
それぞれが生まれた誕生日の間に生まれている

ところで

……

Langfuse v3

使っている？

Langfuse Cloud版

使っている人

挙手お願いします🙋‍♂️🙋‍♀️

Langfuseセルフホステッド版

使っている人

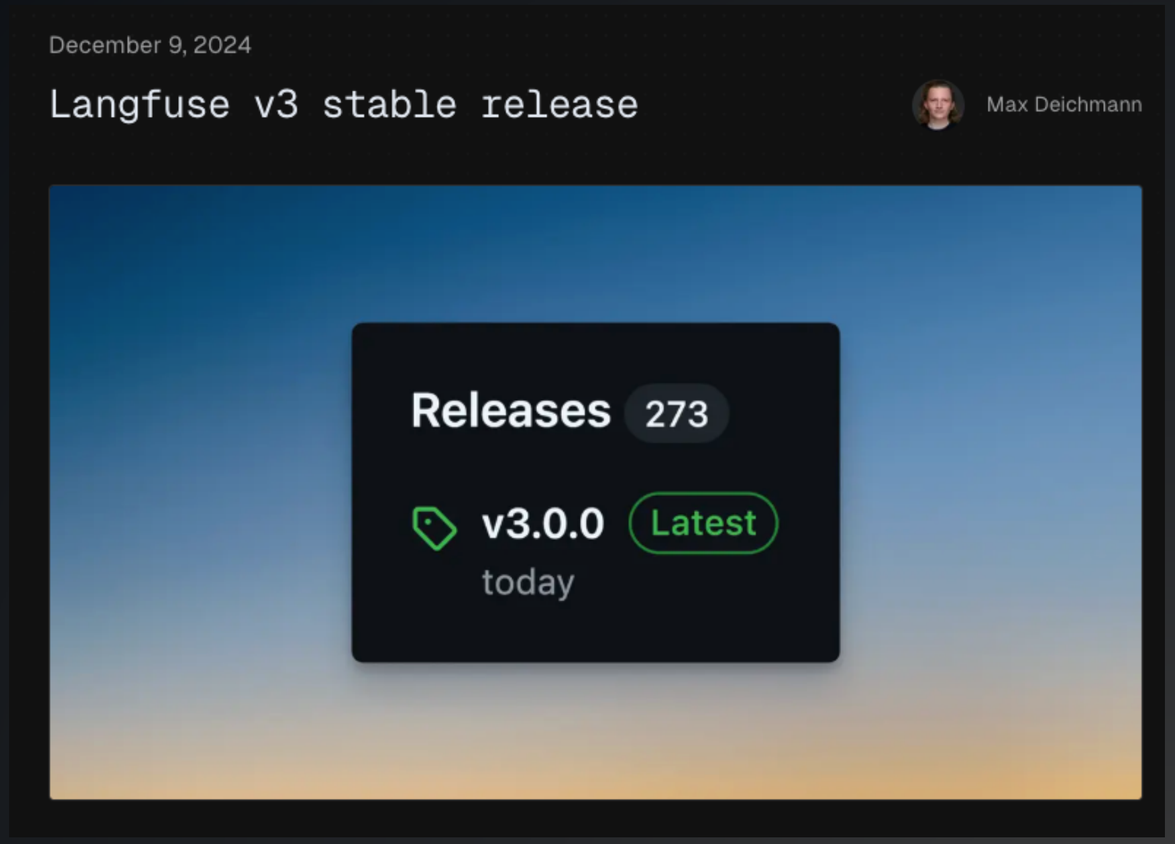
挙手お願いします🙋‍♂️🙋‍♀️

Langfuseセルフホステッド版でV3

使っている人

挙手お願いします🙋‍♂️🙋‍♀️

Langfuse v3



STABLE RELEASE ON 12/9/2024.

.....

大幅にアーキテクチャが変わったことで話題でした。
マイグレーションにチャレンジした方も多いのではないでしょうか?

V3 どんなイメージですか...?

やたらコンポーネント多いな...

コンテナ増えた...

Clickhouseってなに...?

開発者にメリットあるの...?

今日のLTを見て

すぐにアップデートしましょう～

……

その前にLangfuse V3の特徴

アップデートでどんなことが
変わったのでしょうか?

Async Worker

Webサーバーの
重たい処理を
非同期で処理

OLAP(Clickhouse)

分散トレーシングという
重たい分析
ワークロードを解決

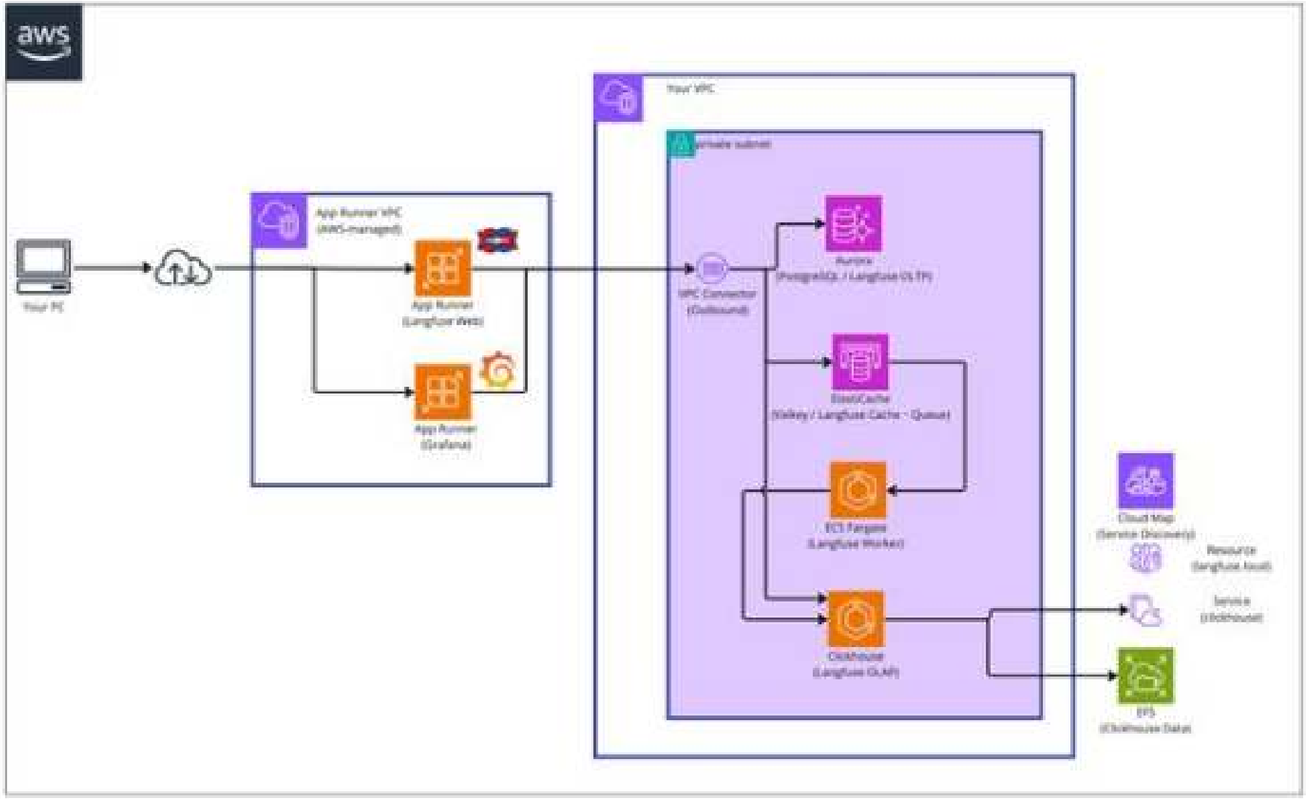
Queuing

Async Workerと
Webサーバーの間を
安全に受け持つ

その前にLangfuse V3の特徴

アップデートでどんなことが
変わったのでしょうか?

詳しくはBlogを読んでね！



Langfuse v3はv2からどのように変わったのかを噛み締めながらAWSマネージドサービスでLangfuse v3を作りきる

AWSはマネージドサービスがやっぱりいいですよね。 Table of Contents Langfuse v3がついにGAしました だいぶ長い記事なので先に結論 Langfuseとは 我々のLangfuse v2アーキテクチャ Langfuse v...

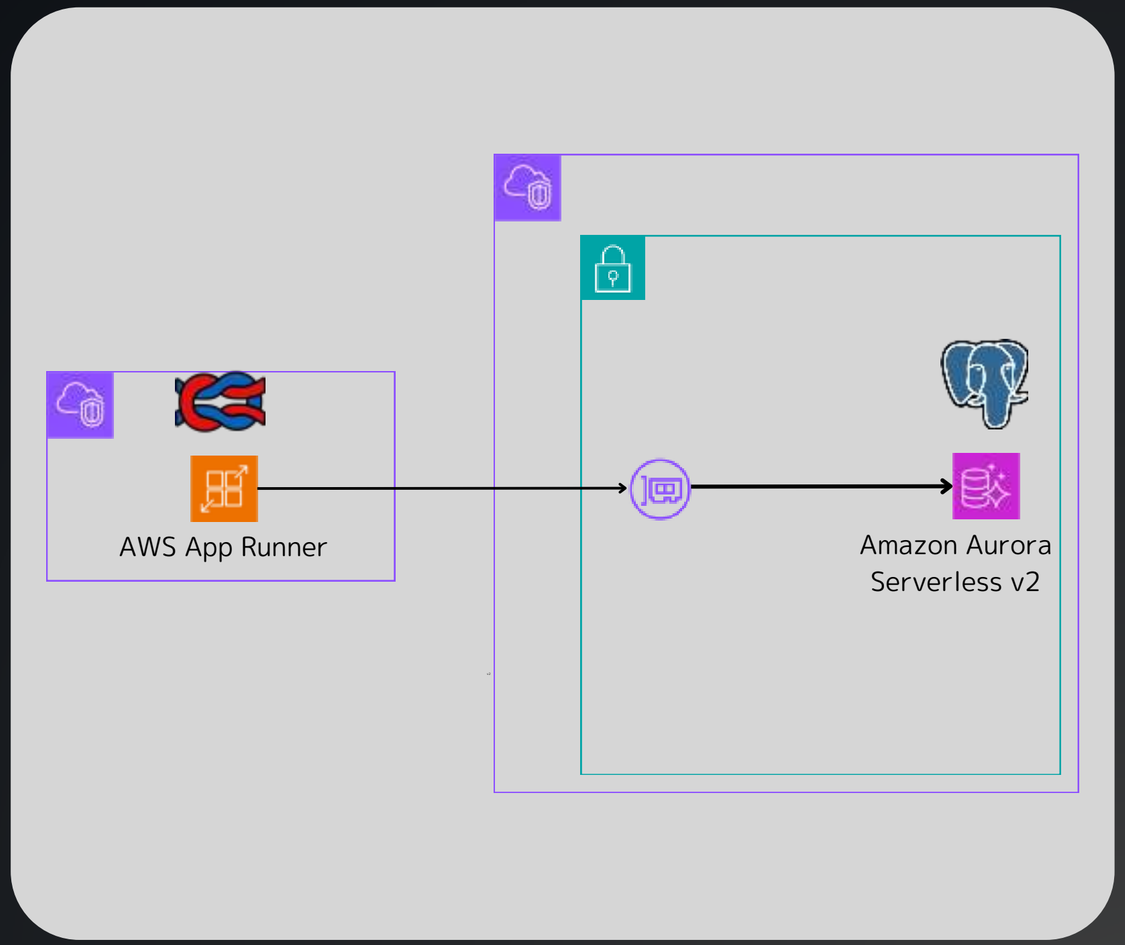
tubone-project24.xyz

今日はどのように

V2→V3マイグレーションを実施するか実践的な話をします

……

まずはじめに



我々のV2構成はこれだ！

.....

AWS App RunnerとAurora構成
LangfuseはNext.jsで動いているので
App Runnerでリクエストベースでの
課金が実現できます。

シンプルさ

可用性

インフラコスト

v3に移行するにあたって

右の4つを重要項目として
V3切り替えを目指します。

会社のMission/Vision/Value
みたい...

Keep Data

データは今運用しているものを
消さずに

Minimum Downtime

できるだけダウンタイムは少なくしたい

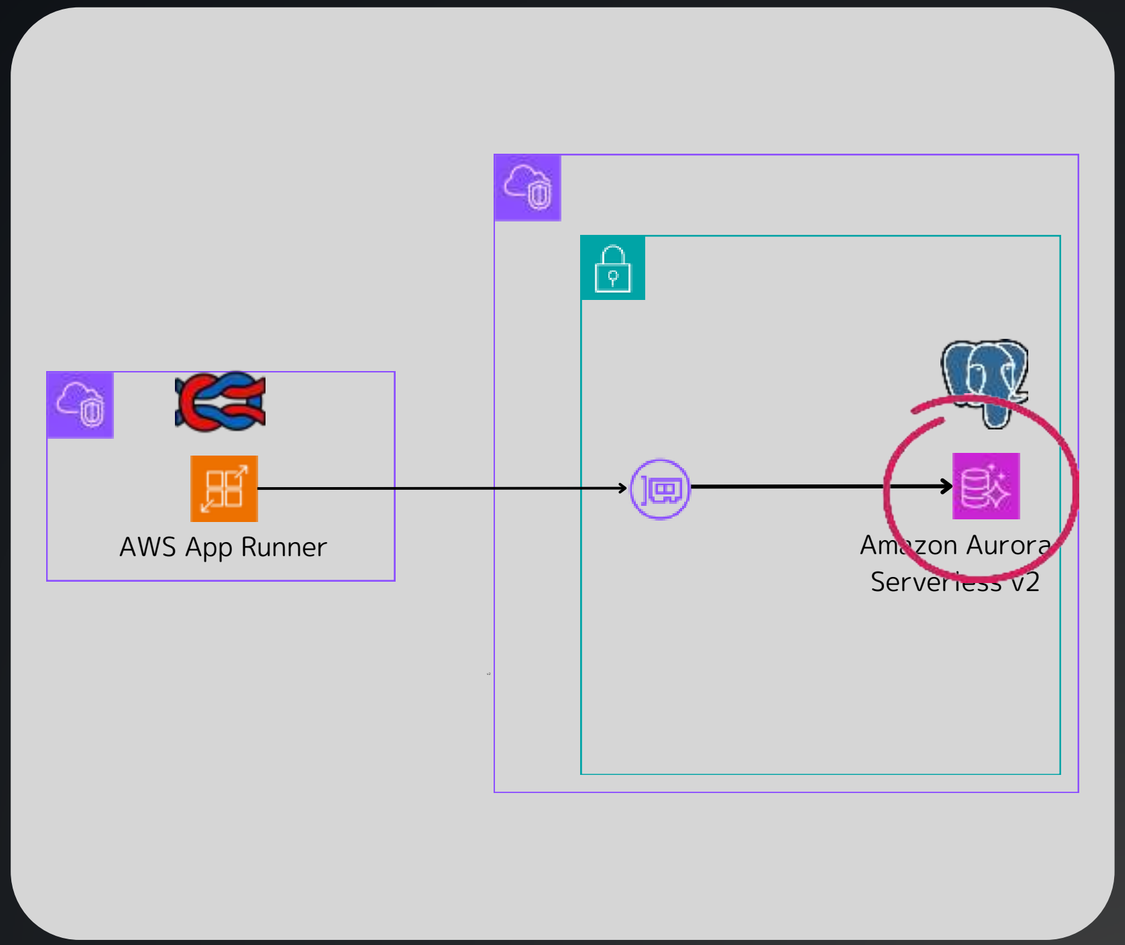
Reduce the Cost

V3になったからと言って大幅にコスト
アップは許されない

Easy Deploy

面倒な手順書は作成したくない
IaCでパッと華麗に

Keep Data

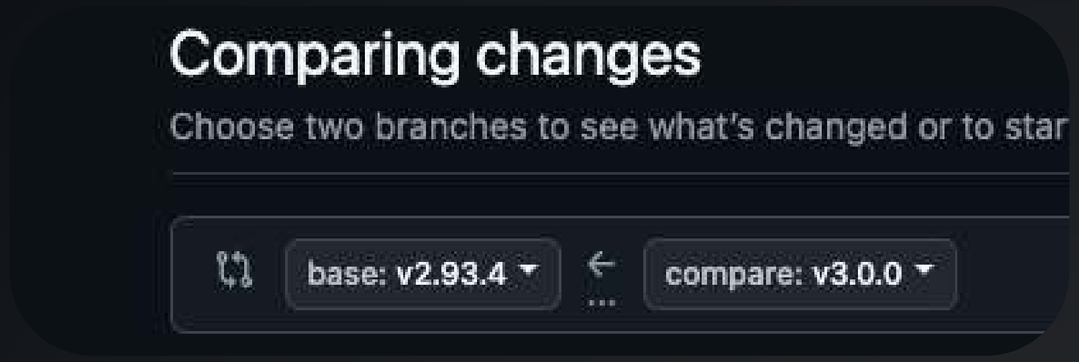


PostgreSQLのインスタンスは
そのままに

・・・・・

現行動いている構成のAuroraは消したり、
移行したりせずそのまま放置する。

念の為作業前にスナップショットを取って
おく。(保険は重要)




```
1  + -- AlterTable
2  + ALTER TABLE "projects" ADD COLUMN "deleted_at" TIMESTAMP(3);
```

```
1  + -- DropForeignKey
2  + ALTER TABLE "job_executions" DROP CONSTRAINT "job_executions_job_output_score_id_fkey";
3  +
4  + -- DropForeignKey
5  + ALTER TABLE "traces" DROP CONSTRAINT "traces_session_id_project_id_fkey";
```

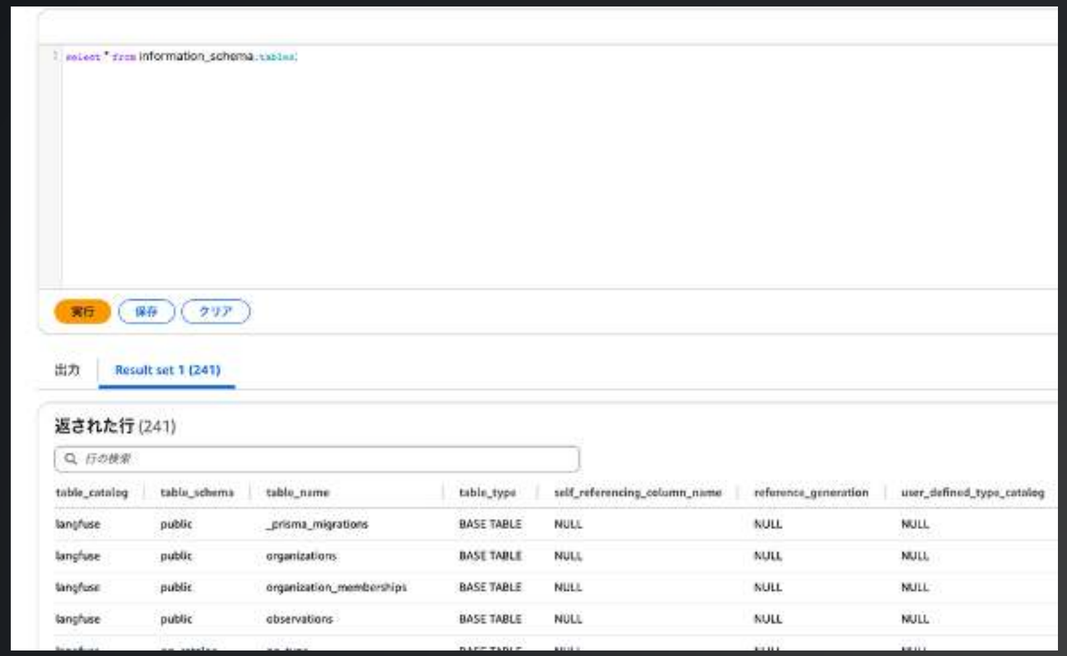
Keep Data

DBマイグレーションに注意

.....

万が一のv3→v2切り戻しに備えて、
バージョンアップで走るPostgreSQLに対してのPrisma Migrationについて注意。

down.sqlはLangfuseプロジェクトで未作成なので、
万が一のときは手動でダウンマイグレーションを実行する必要がある。



Keep Data

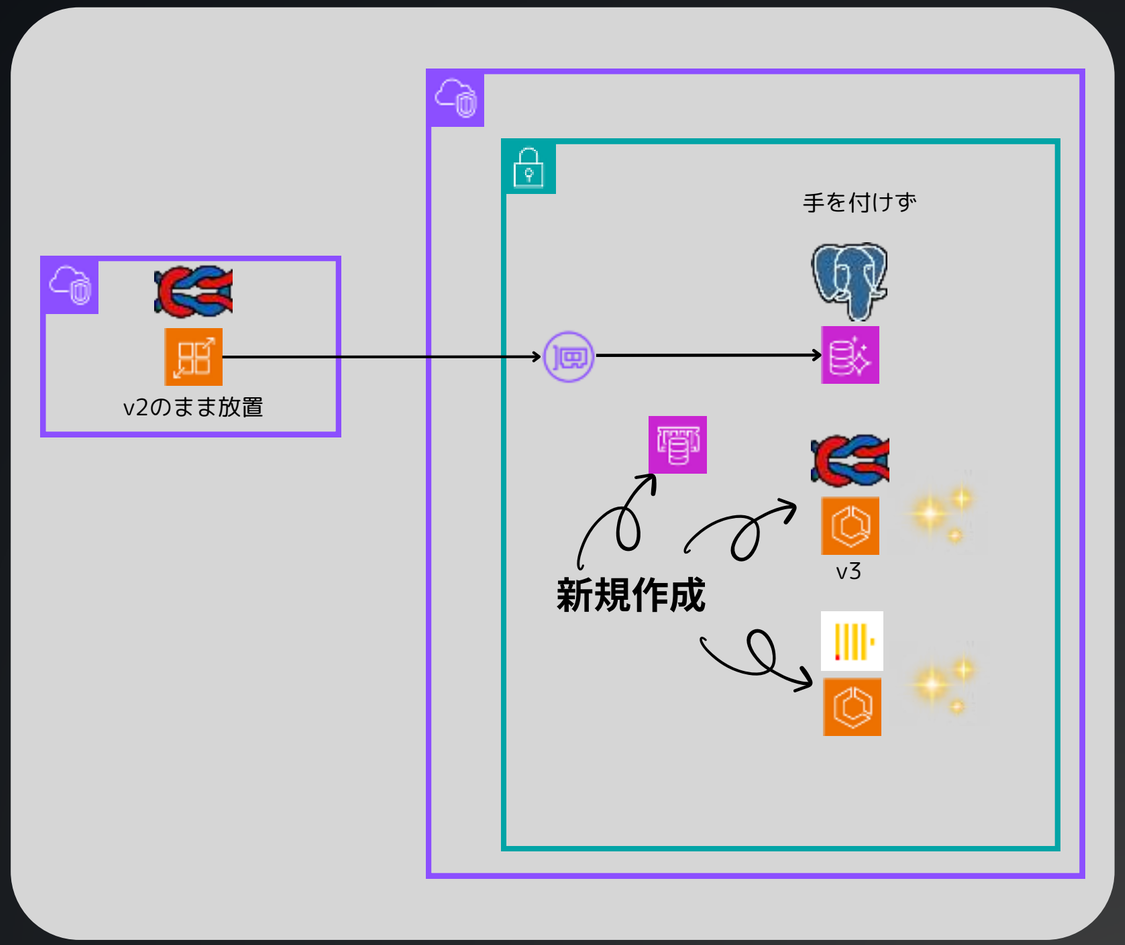
Auroraのクエリエディタを有効化しておく

.....

あまりおすすめされないが、万が一のときはPostgreSQLのテーブルに対して操作が必要なため

- ダウンマイグレーションの際
- Clickhouseへのバックグラウンドマイグレーションを強制実行したい場合

Minimum Downtime



Webサーバー以外を
すべて新規作成し、
最後にv3に上げる

.....

Async worker単体で動くことはないので
(Queueにジョブ登録されるまで動かない)
一気に裏側を作成し、
最後にWeb Serverのv3バージョンアップ
を実施しv3に変更する。

App Runnerの自動デプロイを賢く使って
公式のMigration Guideを少しだけ簡易化
させた方法です。

Minimum Downtime

それでも不安なら
夜間メンテにしてしまう

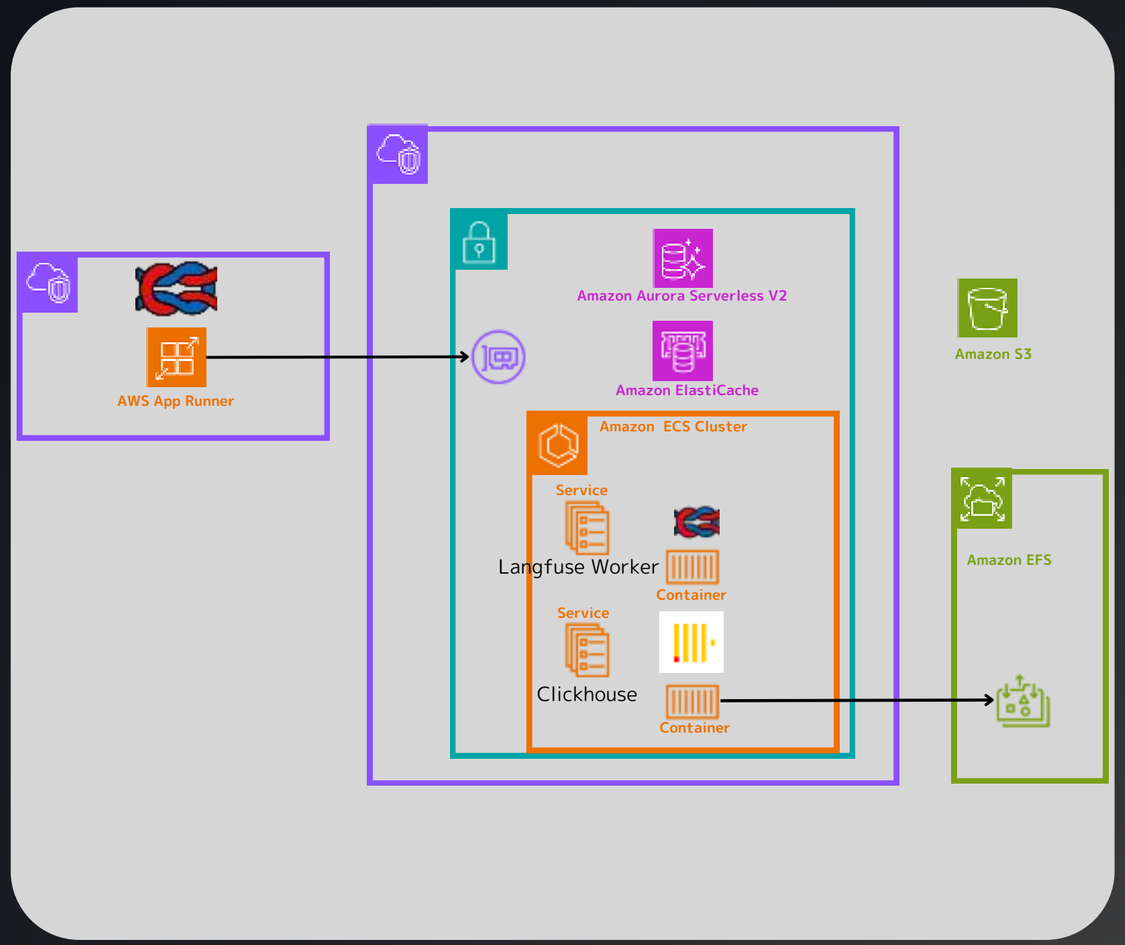
……

不測のことはつきものなのでアプリケーション全体をメンテナンスにしてしまう。

ビジネス部門との調整に難航する場合は、v3化によるスケーラビリティを説明するとよい。

Reduce the Cost

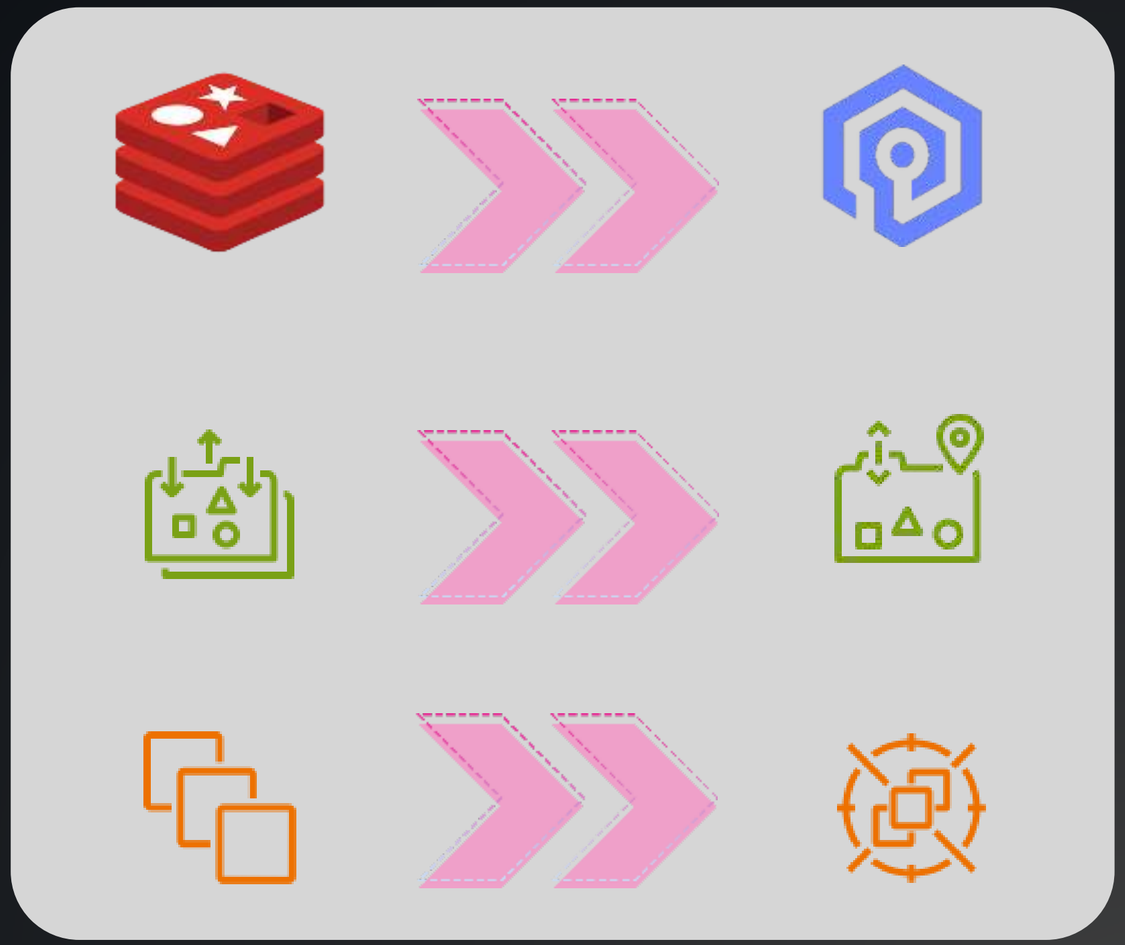
可能な限りマネージドに



……

単純なAWS費用だけでなく、
運用コスト含めて考えるとできるだけマネージドな
サービスを活用するとよい。

- Queue → ElastiCache(Redis)
- Async Worker / Clickhouse→ ECS (Fargate)
- Clickhouseの永続化 → EFS
- Blob Storage → S3



Reduce the Cost

Fargate SpotやValkey、EFS one-zoneのことも考えてみる

.....

さらにコストを抑えたい場合、

- Fargate SpotでAsync Workerを実行する(※1)
- Redisの代わりにValkeyを使う
- EFSのone-zoneにする(※2)
- (開発環境)Scheduleで落とす

などが検討できる

※1: 最悪RedisにQueueueが貯まるので、Langfuse画面上に反映が遅延することも
※2: ClickhouseのコンテナのAZと揃える必要がある。揃えないで立ち上げるとProvisioningで落ちる

V2

約 $60

App Runner
(1 vCPU & 2 GB)
で日勤帯のみアプリ利用想定

Aurora Serveless v2
(0.5ACUで稼働想定)

V3 お得構成

約 $100

v2構成に加えて
ElastiCache
(Valkey cache.t2.micro)
EFS One Zone
Fargate Spot2台

V3 ちゃんとした構成

約 $196

Fargate Spot→Fargate

EFSをStandardに

ちゃんと作るならこっち

どれくらいコスト変わるのか

v2と比べ+$40は痛いですが、抑えようと思えば$100くらいは圧縮できる

Easy Deploy

すべてはTerraform applyで

.....

ここまで説明してきたことを、
Terraformで一発で適用できたら嬉しい。

そんな願いを叶えるモジュールがある。

無事に
移行できました

なのでご安心を

……

どんなイメージですか...?

やたらコンポーネント多いな...

コンテナ増えた...

Clickhouseってなに...?

開発者にメリットあるの...?

どんなイメージですか...?

AWSで簡単に作れるなぁ..!

マネージドで安心...!

コストも工夫次第でなんとかなりそう

移行も簡単だぞ！

最後に

Langfuseセルフホステッド版と向き合うには

OSSという特性を活かして
Just Read it!!!

OSSのコードを読むことでトラブルシュートもできるし、
趣深い改修が入っていることもわかる。

コードを読もう！

最高のドキュメントはコードです！(持論)

でも一人で読むのはツライので
皆さん情報共有しましょう！ﾀｽｹﾃ.....

THANK YOU

FOR WATCHING