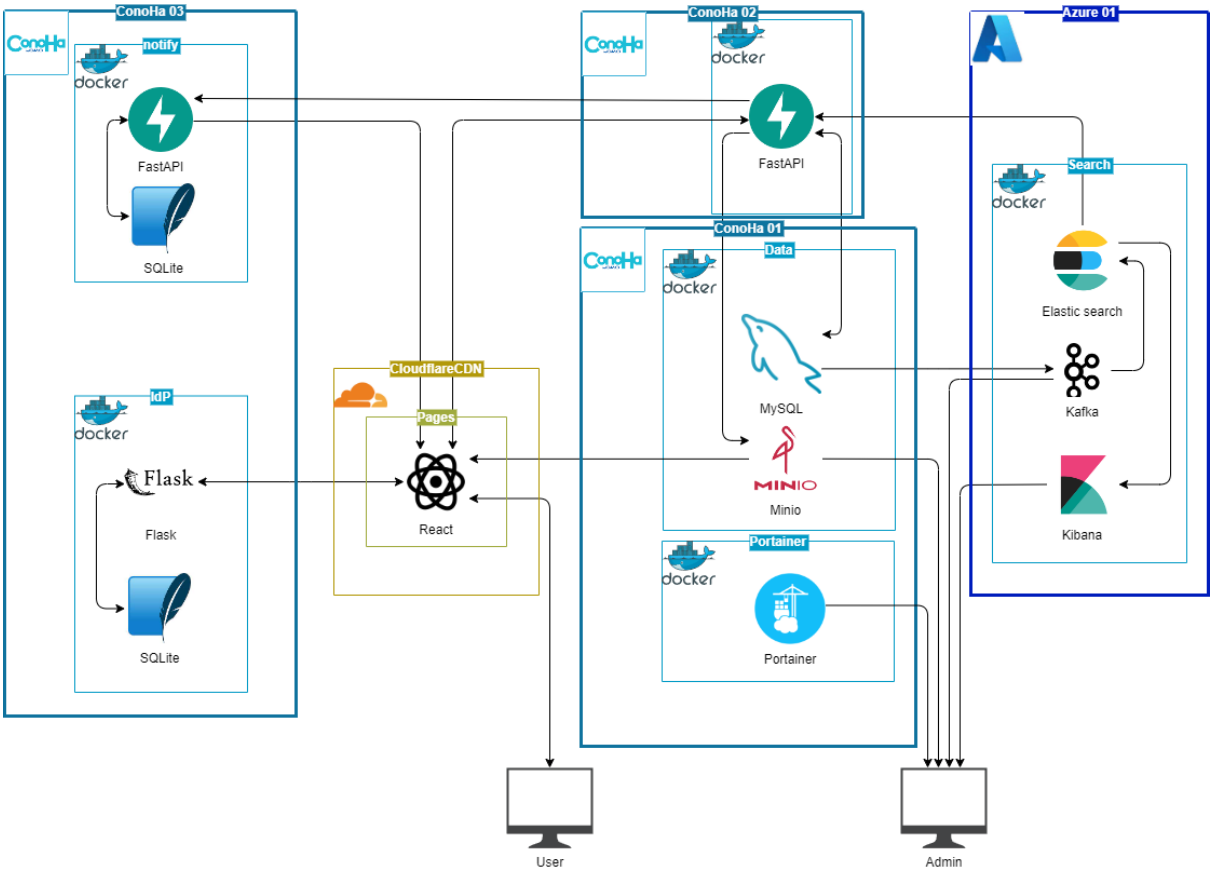
ドキュメント

アーキテクチャ



基本的なサーバはConoHaを利用しリソースが必要なところは学校のAzureをお借りしています。

フロントからアクセスするものはすべてCloudflareを通しておりキャッシュやDoS軽減を行っています

すべてDockerを利用しており、Portainerですべてのインスタンスを管理できます

ConoHa 01

これはユーザのデータを保存するためのインスタンスとして利用しています

またConoHaのインスタンスでは一番リソースが多いため、管理用コンソールもこちらでホスティングしています

- MySQL
  - メインのDBです
  - サービスを利用するにあたって必要なデータはすべてここに格納されます
- Minio
  - S3互換のオブジェクトストレージです
  - ユーザがアップロードした画像や動画などはすべてここに保存されます
  - ファイルを直接配信するだけでなく、管理コンソールも提供します
- Portainer
  - Docker管理用コンソールを提供します
  - このPortainerからすべてのインスタンスのDockerを管理できます

ConoHa 02

バックエンドサーバです

一番トラフィックが多いインスタンスなので、単一でホスティングされています

- FastAPI
  - メインのバックエンドサーバです
  - 認可フロー以外はステートレスに実装されています

ConoHa 03

特殊な機能を支えるインスタンスです

最初の要件にはなかった機能拡張のためのサーバをホストします

- Notify
  - FastAPI
    - 通知を行うサーバです
    - ユーザは定期的にアクセスを行い通知を取得します
  - SQLite

        - 通知を保存するために利用しています
        - ユーザの認証情報のキャッシュも行っています
- IdP
    - Flask
        - OAuth 2.0および、OpenID Conect1.0に対応した IDプロバイダです
        - パスワード認証とパスキーに対応しています
        - クライアントを発行すれば他のサービスでも問題なく利用できます
    - SQLite
        - IdPで作成されたユーザー情報を保存しています
        - メインのサービスのデータは持ちません

Azure 01

学校にお借りしているインスタンスです

この中では一番リソースが多く様々なサービスがホストされています

- Elastic search
    - ユーザ投稿のSearchAPIを提供します
    - Kuromojiを利用し形態素解析を実施し、中精度の検索をおこないます
        - データが蓄積したうえで、チューニングをすれば、高精度の検索が提供できます
    - Kafkaにより投稿データのリアルタイム更新を行っています
- Kibana
    - Elastic searchの管理及びデータ ビジュアライザを提供します
    - 展示時は投稿されたものの中から多く含まれる単語を抽出し、Tag Cloudとして表示します
- Kafka
    - MySQLに蓄えられたデータをストレージに蓄積し、Elastic searchに注入します
    - 分散ストレージとしても使えるため将来的な拡張性も提供します

- 複数のインスタンスからなります
  - Kafka
    - データを蓄積するDBです
  - zookeeper
    - Kafkaのストレージの管理を行っています
  - Kafka Connecter
    - MySQLにデータが更新されたのを検知しデータを取得、Kafkaにデータを挿入します
    - Kafkaのデータが更新されたのを検知しElastic searchにデータを挿入します
  - Kafka UI
    - Kafkaに蓄えられたデータの閲覧、管理を行えるUIを提供します

Cloudflare

皆さんご存じのCDNです

- Pages
  - React+Viteで実装されたフロントをホスティングしています
  - ステートレスな実装になっています
- CDN
  - フロントから発行される通信はすべてCloudflareを経由し必要に当たってキャッシュされます

こだわり

かなりこだわりを持って開発を行ったためいろんな観点から記載します

技術選定におけるこだわり

言語は授業でふれたものを利用しましたが、採用する技術はチームメンバがほぼ・まったく触れたことがないものを選定しました

これは私個人の考えですがHEWを単なる課題としてとらえるのではなく挑戦し成長する機会としてとらえているためです。

なぜかといえば、学校から開発に使えるリソースや時間を提供され、発表の機会をもらえるのですから活用しない手はありません。

このような機会をもらったのであれば、学生なのですから、開発手法や、効率、マーケティングにとらわれず様々なものを試し、失敗を積み重ね、手探りでベストを見つける経験をするべきです。

これは他から**作るもの**や**作り方の制限**を受ける社会では経験できないでしょう。

だからこそ完成度を高めるのものではなく自分たちが一番成長でき、将来性を高めるものを選定しました。

開発におけるこだわり

一般的に使われているツールを使用し触れたことのない人には学習の機会としてもらいました

またGitHubCopailotやChatGPT、DALL-EなどAIを最大限活用するようにしていました

認証、認可等の複雑な部分はSaaSを利用するのではなく、独自で実装を行い学習の機会としました

可能な限りお金がかからないように気を付けました

実装におけるこだわり

ステートレスに実装を行うことにこだわり実装を行いました

セッション管理をせずOpenID Conectを利用しIDトークンを用いて認証しています

そのためバックエンドを複数立てることができ、ロードバランサ等で接続先が変更されてもユーザは認識することはありませんし、DBもKafkaを用いることで分散が可能です

これらのことから非常にスケーラビリティに優れるサービスを構築できました