Latona 株式会社 DevOps ・ CI/CD Engineer Technical Test

- アプローチ
  - 継続インテグレーション、継続的デリバリーを実現するために、Github と CircleCI サービスを連携させる。Github への push をトリガーとして、CircleCI 上でビルドとテストを駆動させる。テストを通ったシステムを、運用環境(Heroku)にデプロイする。
  - Git ではマイクロサービスとそのテスト、および Circle CI のワークフローを管理する。
  - マイクロサービスは、気象情報 API をコールし、正常レスポンスであれば、気象情報データを処理して、現在時刻を名前とした CSV ファイルを特定のディレクトリに出力する。
  - また、マイクロサービスは、Web アプリケーションとし、HTTP メソッドでリクエストを受け取ることで、気象情報を取得し CSV ファイルを出力する。また、API のように HTTP リクエストへのレスポンスでも気象情報を返すようにする。
  - コーディングスタイルは PEP8 に準拠する

- 重要な概念

  DevOps として、開発したソフトウェアを定期的に統合し、テスト、を実施する。そして、運用環境へのリリースサイクルを早くする。

- 直面している可能性のある問題
  - ビルド、テストは成功しているが、運用環境 Heroku 上でうまく動いていない。
  - 外部 API 仕様の変更により、運用中サービスが動かなくなる可能性
  - 通信異常などにより、外部 API が使用できない可能性
- 解決策
  - Docker のコンテナ型仮想化により、開発、テスト、運用の環境構築することで、環境上の差異をなくす。また、Web サービスとしてのテストができていないため、テスト環境へデプロイし、HTTP リクエストを送信させるシステムテスト工程を増やす
  - ソースコード変更による push 時のみでなく、定期的にビルドとテストを実施し、外部環境の変更による不具合を早期に検出する
  - マイクロサービス側では、気象情報の取得に異常が発生した場合、Failsafe アクションとして、エラー情報の保存、異常を通知する機能を実装する

- Git リポジトリおよび Circle CI へのログイン方法

  ・Git リポジトリ URL：https://github.com/u-masato/micro-weather-service

  ・Circle CI へのログイン ：ログインした状態で

  https://circleci.com/gh/u-masato/micro-weather-service にアクセス

インテグレーション方法：.env ファイルに OpenWeatherMap の API_KEY を記載し、

> gunicorn microweather.app:app または python app.py を実行する