CI&CD

ゴール

- なぜCI&CDをするのかの概観を感じる
  - 動機/理由となるところ
  - 実行環境について
- 今度、身につけていくことのボリューム感を感じる

なぜCI&CDをやる?

動機/理由となるところ

- 品質の高いコードが書きたい
- 安心したい
- 楽をしたい

どんな安心?

- 開発が着実に進んでいることの確証を得たい
- さらに、定量的な評価での確証を得たい

どうして楽をしたい?

- 開発における確認作業は大変！（Pull Requestのレビュー）
- なるだけコーディングに関するクリエイティビティに時間を使いたい

→ CI&CDをすると、安心と楽を得つつ、品質の高いコードになっていく

CI&CDって？

CI（Continuous Integration）: 継続的インテグレーション

コードの変更を起点にコードの静的分析、ビルド、テスト、成果物の生成などの実行を自動化する手法

CD（Continuous Delivery）: 継続的デリバリー

CIで検証・テストされたコードや成果物を目的の環境に自動でデプロイする手法

例えば、Pull Requestレビュー時の安心

- タイポがないか、コーディング規則にあっているか
  - → コードの静的分析
- ちゃんとIssue解決となっているか
- 変更点によって、他に意図しない影響を与えていないか
  - → 自動化されたビルド、テスト、デプロイの成功

→ これらの確証が取れている状態であることがわかっていれば、安心して、確認の負担を減らせる

CI&CDの実施ついて

sfzでの具体的な案件・場面

- https://github.com/sforzando/marziale
- https://github.com/hacking-papa/nuricame-web
- https://github.com/dentsudigital/mis

以下のようなツールを使うことで、CI&CDを実行している

CI&CDの実行環境

- ローカルの開発環境
- CIツール/プラットフォーム
    - GitHub Actions
    - CircleCI
    - etc.

コードの静的分析（主にローカルにて）

- Lintの実行
    - Pylint
    - ESLint
    - etc.

ビルド

- プログラムの実行環境の構築
    - ライブラリのインストール
    - コンパイル

テスト

- Unitテスト
- E2Eテスト
- テスト結果の集計/表示
    - Codecov

成果物の生成

- ドキュメントの生成
- etc.

CI&CDの実行結果の通知

- GitHub連携
- Slack連携