CI/CDツール導入検討

アジェンダ

1. CI/CD is 何？
2. 何がおいしいの？
3. どのツールを使うべきか？

CI/CD is 何？

CI/CD

- 一般に、継続的インテグレーションと、継続的デリバリーまたは継続的デプロイメントの組み合わせのプラクティスを指します。(wiki(原文英語))

なんか分からん単語がようさんあるやんけ

継続的インテグレーション

継続的デリバリー

継続的デプロイメント

やな

継続的インテグレーション

- CI（英: continuos integration）とは、すべての開発者の作業コピーを1日に数回、共有されたメインラインにマージすることである。グラディ・ブーチは1991年のメソッドでCIという用語を最初に提案したが、彼は1日に数回の統合を提唱していなかった。エクストリームプログラミング(XP)ではCIの概念を採用し、1日に1回以上、おそらく1日に何十回も統合することを提唱した。 (wiki)

ああ、なるほど。CIは日常的にmasterにマージしろってことね。

じゃあCDはcontinuos deliveryかdeploysで日常的に本番環境にデプロイやデリバリー作業しろってことね

確かに日常的にソースや本番環境を新しいものにしておく
CI/CDの思想はGoodだしアジャイル開発とかなり親和性
が高そうだが

これらに時間がかかって実現が厳しい

ユニットテストや統合テスト

静的解析(コーディングルール遵守の確認とか)

レビュー

資料おこし

ビルドとか適用作業

そこでCI/CDツール

こいつらを自動化してもらおう

⭕ ユニットテストや統合テスト -> 得意

⭕ 静的解析(コーディングルール遵守の確認とか) -> 得意

❌ レビュー -> こいつは人力かなあ

⭕ 資料おこし -> sphinxとかと組み合わせればできるはず

⭕ ビルドとか適用作業 -> 得意

ふーん…。で結局何がおいしいんだっけ？

ツールを導入するメリット

- ソースや環境を常に新しく保つためにかかる人的工数を削減、最小化できる
- 誰でもマージ、デプロイ、デリバリーができるようになる(権限とかは設けるべきだが)
- 安定的なリリースに寄与できる

悪い大人じゃないのでデメリットもあげます

ツールを導入するデメリット

- 作業工数自体がなくなるわけではない。(あくまで作業者がフリーになるだけ)
- 利用料金がかかる場合がある
- オンプレ環境だと
  - 設定、保守が属人化しやすい
  - 変な運用をすると使い物にならなかったり保守地獄が発生する
- ツールによっては学習コストがかかる
- プロジェクトの規模によっては自動化の恩恵が少ない

デメリットもあるがメリットが強力

補足

- 当たり前だけどCI/CDツールはテスト等自動化してくれるけどテスト自体は自分で用意しなければならないです

CI/CDについては大体分かったが

何を導入したらいいんだろうか

この辺りが有名どころ。他にも色々ある

- Jenkins
- circleci
- Travis CI
- GitHub Actions
- Drone.io

多すぎて分からん、選定基準が欲しい

デメリットに文句がなければ基本的にJenkinsを選ぶべき

メリット

- 無料
- プラグインが豊富 / 自由度が高い
- 日本語のナレッジが多くある

デメリット

- オンプレ(場合によってはメリット)
- 習得コストがかかる
- インフラ管理にコストがかかる / 技術が属人化しやすい

個人的にはjenkinsは前職で管理地獄をみたためで好きではない

- 管理していた方が退職して構成等がブラックボックス化
- SRE部署の勝手なスクリプトやパイプラインの仕様変更による事故
- 利用しているスレイブのサーバーが別部署のものになって急に使えなくなる

運用がしっかりしていれば回避できるはずだが組織の規模が大きくなるとそうもいかないのだろうか...

Jenkinsに辛い思いをされた方用の選定基準

- 利用料金(費用対効果的にどうか)
- 誰でも使える / 仕組みに透明性があるか
- GitHub / Docker / AWS 等との連携性の高さ
- CIの実行速度

そう考えると以下2つが気になる

- Travis CI
- GitHub Actions

この2つについては実際に利用したのち結果をまとめて報告する