はてなにおける fujiwara-wareの活用や ecspressoのCI/CD構成

id:cohalz / @cohalz

2025/1/17 Fujiwara Tech Conference 2025

自己紹介

- こはる(@cohalz)
- 株式会社はてな SRE
- 趣味でfujiwara-wareにコントリビュート

Hatena

今日話すこと

- 社内で利用しているfujiwara-wareの紹介
- ecspressoのCI/CD構成

fujiwara-wareの活用

色々使ってます

- 扱ってる技術が近いのもあり日々便利に使ってます
  - ECSやAWS全般、Mackerel
  - PerlやGo

ecspresso / lambroll

- ほぼ全てでecspressoを採用
  - リリース手順が違うなどはある
- lambrollは一部利用
  - あまり更新のないものはCDKのままにすることも

maprobe

- Mackerelのメトリックをいい感じに取ったり集計したりするツール
  - AWSのマネージドサービスにプラグイン実行したり
  - ホストメトリックを集計してサービスメトリックにしたり
- 最近はOTel形式に変換する機能なども

Redis::Setlock

- Redisで排他制御をするためのPerlモジュール
  - Redisを使って排他制御するwrapperコマンド Redis-Setlock をPerlとGoで書いた - 酒日記 はてな支店
- EventBridge経由のバッチで利用
  - at-least-onceの多重実行を防ぐ

AWS::XRay

- PerlでX-Rayにトレースを送るモジュール
  - 当然公式にX-RayのSDKはないため
  - Perlでも分散トレーシングしたい！AWS::XRay による解析とその実装 / YAPC::Tokyo 2019 - Speaker Deck

fujiwara-wareではないが...

- カヤック社のOSSも利用し始めています
  - mirage-ecs
  - shimesaba
  - prepalert
- [カヤック発OSSカタログ - KAYAC Engineers' Blog]

どうやって探してる？

- ブログやGitHubから他のものをざっと見る
  - 好みのOSSを作ってるなら他も似た課題を持ってるはず
- 個人や会社の他のOSSもdigって見ると良いかも

ecspressoを
よりよく使う

ecspressoを本番運用するにあたって

- イメージの更新どうやる？
- 設定ファイル共通化したい！
- どこからデプロイする？
- レビューやリリースのプロセスは？

イメージの更新どうやる？

- イメージの確認・更新が悩ましい
  - タスク定義は巨大になりがちで見にくい
  - コンテナ定義内のimage差し替えも割と面倒

改善策

- デプロイ時に環境変数などで渡す
- 別のファイルから読み込む
- jsonnetでやる方法の紹介

jsonnetの工夫

::を使うと隠しフィールドに

```
{
  params:: {
    env: '',
  },
  name: $.params.env,
}
// => { "name": "" }
```

ファイルの読み込みと更新

```
import 'base-task-def.libsonnet' {
  params+: {
    env: 'production',
  },
  cpu: '2048',
}
// => { "name": "production", "cpu": "2048" }
```

実際に使ってる機能は？

- コメント、値の上書き、ファイル分割程度
  - 各環境の共通部分を分離
- 複雑なことをしすぎない方がわかりやすい
  - 関数、if、ループなども避ける

事故を防ぐ工夫

よくある(？)事故

- デプロイするバージョンを間違える!
- デプロイするサービスを間違える!!
- ECRのイメージが消えてタスクが立たない!!!

デプロイを間違える対策案

- 手元からデプロイしない(大事!!)
- デプロイするバージョンやサービスを
  レビューできる

ECRのイメージが消える対策案

- いい感じのライフサイクルポリシーを決める
  - 本番用は個数、それ以外は日数がおすすめ
- デプロイ不可な状態に早めに気付く仕組み
  - 防ぐことはできないがリスクは緩和

それを踏まえてCI/CDはどうするか

- SSOTで宣言的な構成にしたい
- 自動化したい
- 複数のサービスを同じ方法で扱いたい
  - 一つ一つ設定していくのは手間
  - はてなだとチームで見るECSサービスも多い

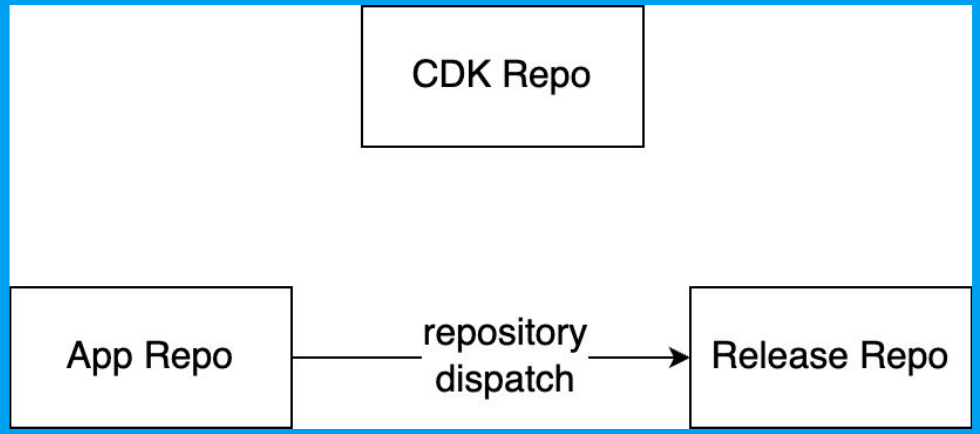
ecspressoのCI/CDを作っていく

CI/CDの構成を考える

- 社内にEKS+ArgoCDのCI/CDが既にあった
- これを元に手作り
  - GitOpsぽい形に

デプロイ用のリポジトリを作る

- コードともCDKとも違うリポジトリを作るように
- このリポジトリに全サービスのECS設定を置く



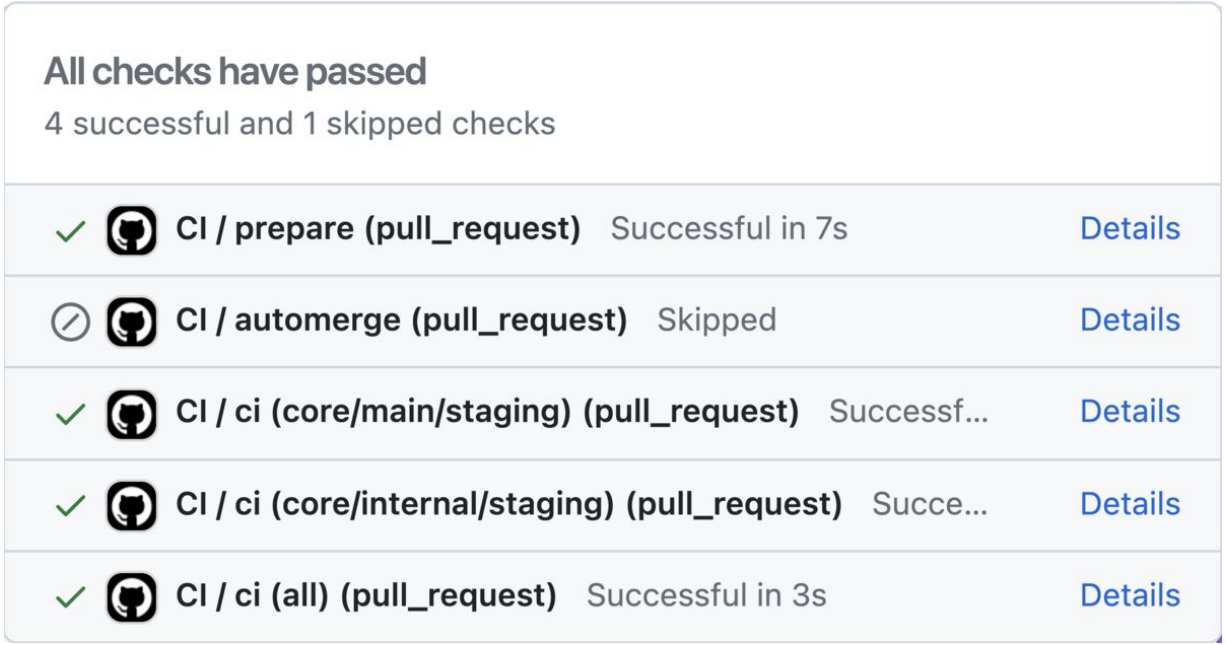
2 blog/_images/nginx.libsonnet

```
...  ...   @@ -1,5 +1,5 @@
 1    1    {
 2    2      "name": "nginx",
 3    3      "repository": "[redacted].dkr.ecr.ap-northeast-1.amazonaws.com/blog-team/blog/proxy-nginx",
 4         -  "tag": "0a4995b"
      4    +  "tag": "02539fd"
 5    5    }
```

アプリリポジトリの方からJSONを送って更新

PRが作られると

- 変更されたファイルからデプロイ対象を特定
  - どのサービスがデプロイされるのかの確認もできる
- そのECSサービスのみverifyとdiffを実行
  - チェックが通らないとデプロイできない



デプロイされる対象のみチェックする

mainにマージすると

- GitHub Actionsでecspresso deployを実行
  - CIでテストしたサービスのみが自動でデプロイ
- ロールバックしたい時はrevertするだけ
  - リポジトリとECSの状態は常に同じ

複数サービスも同じリポジトリに

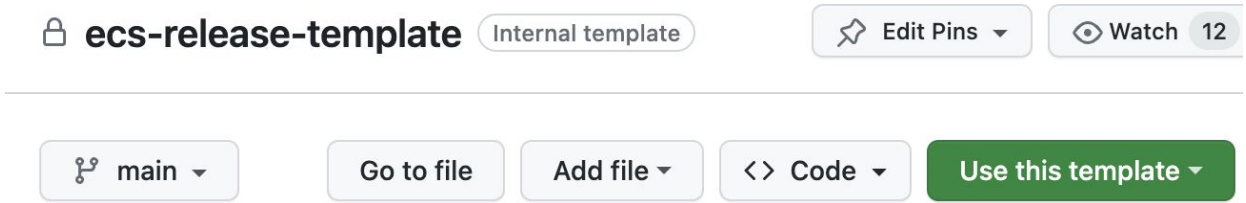
- チームでメンテする全てのサービスを集約
  - これによりチーム内では1つのデプロイ方法だけ覚えれば良くなる

さらに利用を広げたい

- 別チームでも同じ構成を使えるようにしたい
- 新規サービスを追加しやすくしたい

リリース用リポジトリのテンプレート

- 必要なワークフローなど揃えたテンプレート
- Cloneしてecspressoのファイルを置く
  - 既に構築したCI/CDをすぐ使える



ファイル分割ツールも用意

- 既存のECSのサービスをこの構成のecspressoファイルを書き出してくれる
  - 分割だけでなく環境変数のパラメータ化なども

Actionの再利用

- 社内向けComposite Action / Reusable Workflowにする
  - 各チームはそれを利用する形
  - アップデートはRenovateで拾う

chore(deps): update hatena/actions-collection action to v0.0.6

Open **renovate** wants to merge 1 commit into `main` from `renovate/hatena-actions-collection-0.x`

Conversation 0 | Commits 1 | Checks 2 | Files changed 3

renovate bot commented 3 minutes ago

MENDRenovate

This PR contains the following updates:

Package	Type	Update	Change
hatena/actions-collection	action	patch	`v0.0.5` -> `v0.0.6`

Release Notes

▶ hatena/actions-collection (hatena/actions-collection)

Renovateで社内のActionを更新
リリースノートも確認できる

リリースのフロー

普段の開発のフロー

- PRをmainから作ってマージする
  - GitHub Flowに近いスタイル
- リリース手順が書かれたissueが作られる
  - stagingが更新されてるので確認

リリース待ち一覧 (CountAPI) #230

Closed

github-actions opened on Aug 30, 2023

What's Changed

- chore(deps): update aws-actions/configure-aws-credentials action to v3 by **@renovate** in chore(deps): update aws-actions/configure-aws-credentials action to v3 #229

Full Changelog: `release-20230706T161342...fd7587a`

リリース手順が書かれたissueが作られる
内容と手順を確認してclose

Issueを閉じるとリリース

- タグが打たれリリースリポジトリに通知
  - それをマージすればデプロイ完了
  - ECRにエイリアスタグを作るだけでビルド待ちもなし
- 問題があった時は
  - アプリ/リリースどちらかのリポジトリで戻す

GitHub Actionsで実行しているもの紹介

ECRイメージの存在確認

- 定期的に全サービスにverifyを実行
  - 失敗したものをSlackへ通知
- initしてverifyならecspresso管理してなくても検証できる

デプロイ設定のファイルを用意

- デプロイ時に動作をカスタマイズしたい
  - 別のECSサービスも同時/順番にデプロイしたい
  - Mackerelにアノテーションしたい
- 別途JSONファイルを独自に利用して実現
- ActionsがそのJSONの値を見て処理する
  - デフォルトでjqが同梱されてるので扱いやすい

ステージング環境を夜間休日落とす

- 開発しない時間に止める
  - 落とす対象を決めてスケジュール実行
- 落とす時はscale —tasks=0
- 戻す時はdeployし直すだけ

リリースの履歴を残す

- GitHubのリリースにも変更一覧を出す
  - github-scriptでgithub.rest.repos.createRelease
  - generate_release_notes: trueするだけ
- タイミングをMackerelにアノテーション
  - cohalz/post-mackerel-annotation

ecspressoのバージョン固定&更新

- asdfやGitHub Actionsでecspressoのバージョンを固定しRenovateで更新する - Re:cohalz

色々やってきた結果

現状

- 社内で多くのサービスがこの構成になった
  - チームメンバーが異動してもすぐ構成を理解できる
- 事故も少なく平和に過ごせている

課題もある

- どうしても手作りが多くて複雑
  - 単一のECSサービスの利用には大袈裟
  - 使うだけならいいが中身を理解しようとすると大変
- Lambdaのデプロイどうするか
  - ecspressoのリポジトリと同居する？

おわり

- fujiwara-wareやecspressoの活用事例を紹介しました
- こうしてるよとかの話を聞きたい！