Treasure 2020

Infrastructure Part

syota だよ

大塚 翔太 (shota ohtsuka)

2019 年入社

fluct SRE で一年過ごして、今は zucks affiliate 所属

普段はいろんな事をしてます。

- 開発環境作ったり
- CI / CD の調整
- 監視周り、アラート対応
- 周辺ツールの version 上げ等

tomokatsu だよ

新保 智喝 (tomokatsu shimbo)

2017 年入社

システム本部で、主にメディア系サービスが使っているインフラ（主にAWS）を管理したり、全社で使っていたデータセンタの管理・撤退をやってました

G Suite, コーポレートサイト, 問い合わせフォームの運用などコーポレートエンジニアな働き方もしていました

こないだの7月からデジコ（https://digi-co.net/）でアプリケーションエンジニアやってます！職業プログラマ歴1ヶ月です！お願いします！

やること (2h)

- 事前課題について
- グループワーク環境について
- CI / CD について
- 監視について
- DevOps について

持って帰ってもらいたい事

- 今回の構成に使用している AWS のサービス理解
- インフラに対する新しい視点
- チームで開発を進める上で大事にしたい事

お願い事

今年はオンラインなのもあり

反応が無いと緊張するので

適度にリアクションください

 # infra まで mm

事前課題について

事前課題について

- Issue

  https://github.com/VG-Tech-Dojo/treasure-app/issues/18

- 解答例 PR

  https://github.com/VG-Tech-Dojo/treasure-app/pull/17

事前課題について (質問タイム)

グループワーク環境について

手を動かそう!!

treasure-2020-x から自分のチームの aws console にスイッチロールしよう

treasure-2020-x console にログイン
https://treasure-2020-x.signin.aws.amazon.com/console

手を動かそう!!

treasure-2020-x から自分のチームの aws console にスイッチロールしよう

スイッチロール画面を開く
x を自分のチームに置き換えてね
ex) a -> treasure-2020-a

手を動かそう!!

treasure-2020-x から自分のチームの aws console にスイッチロールしよう

確認しよう。
みんなできた？

これであなた達は **admin** です。

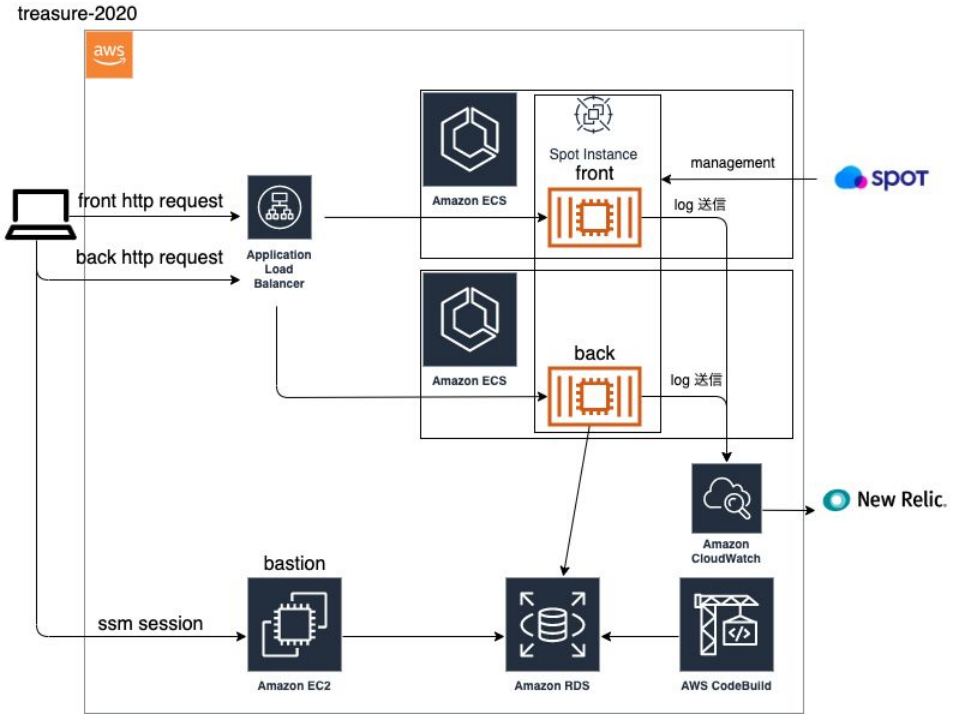
admin って？

なんでもできる。
必要がある場合はどんどん挑戦してください。
困ったら助けます、相談乗ります！

ただし！！

- 機密情報はしっかり自己管理しよう
- 共有してもいい範囲をしっかり考えてから共有しよう
- コストも考えよう、実現に必要なコストを意識しよう
- いたずらは絶対にしない事 ()

AWS 環境を見ていこう
~全体~



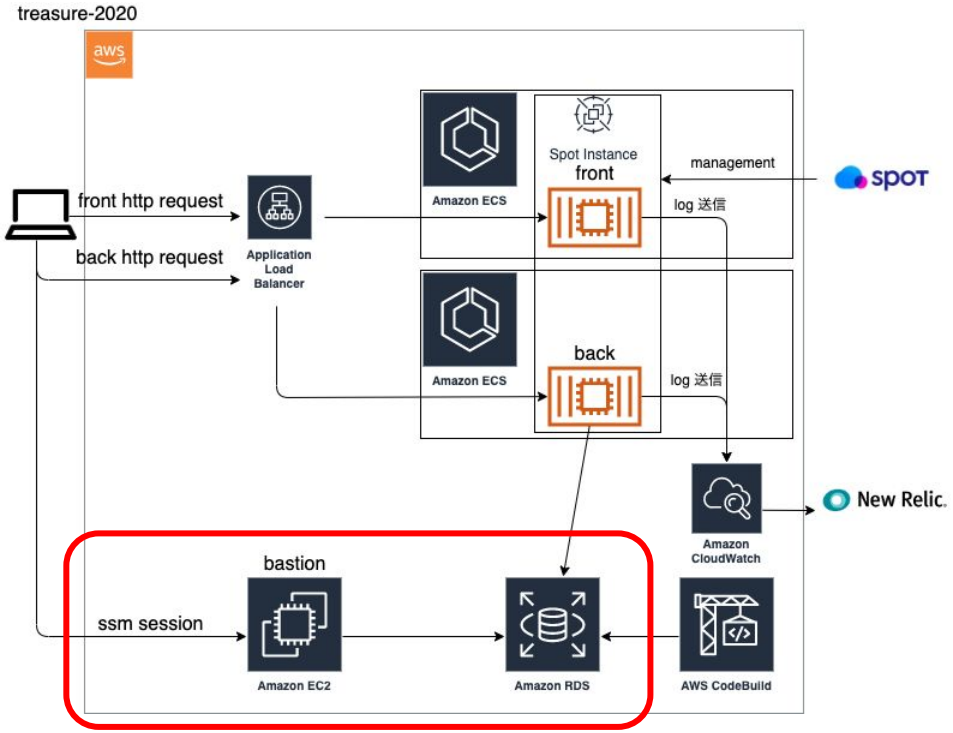
AWS 環境を見ていこう
~Relational Database Service~

RDS = RDB サービス

今回は Mysql 8.0 を動かして
treasure-app の DB として使っている

Mysql server に login したい場合は
bastion サーバー (EC2) 経由で RDS に
接続できる

RDS Doc

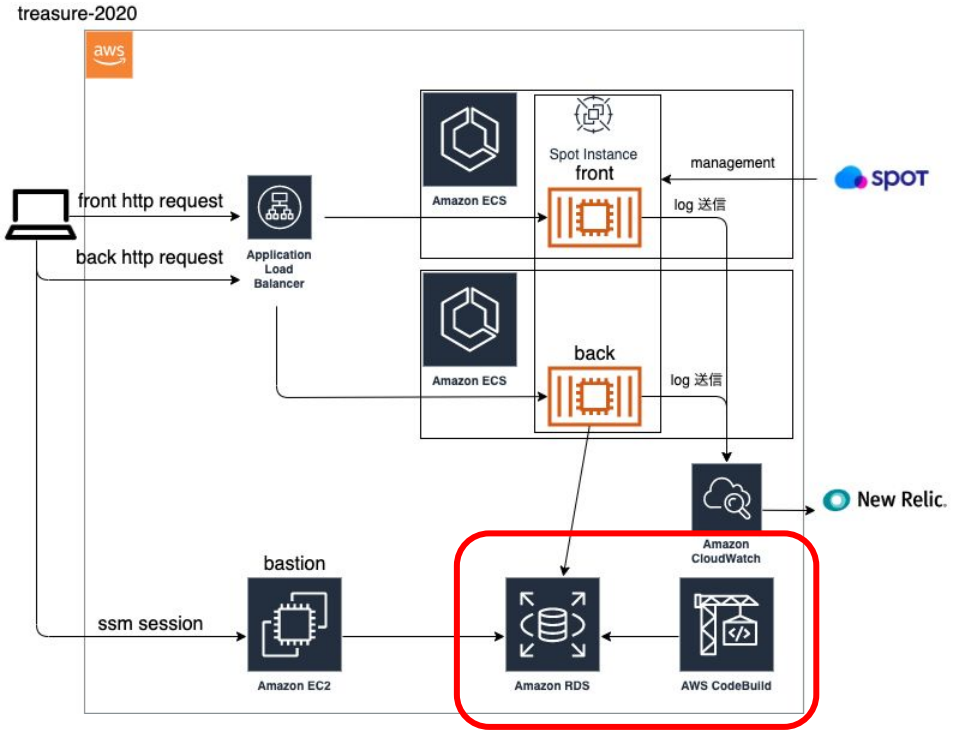


AWS 環境を見ていこう
~CodeBuild~

CodeBuild = ソースコードをコンパイル、テスト、成果物生成までの一連の処理を実行できるサービス

今回の Treasure では RDS に flyway migrate を実行する為に使用してます。

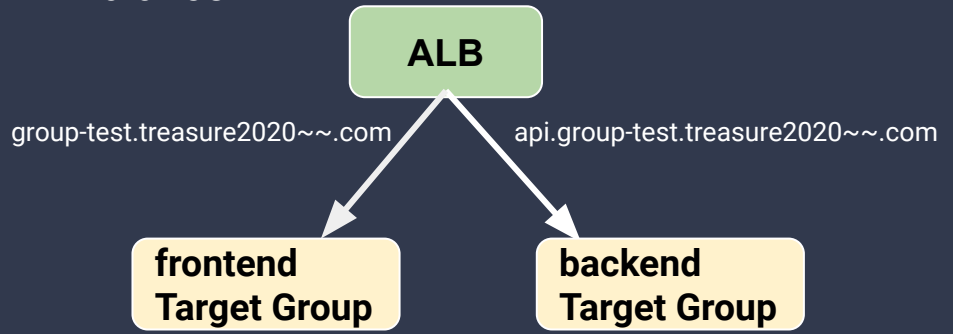
CodeBuild Doc



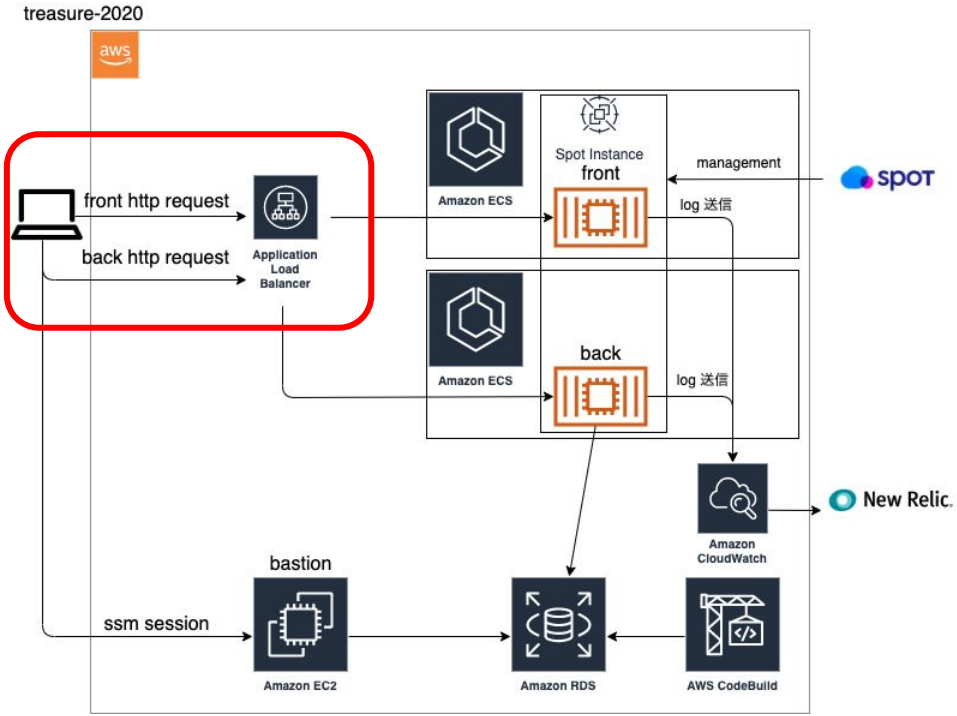
AWS 環境を見ていこう
~Load Balancer/Target Group~

LB/TargetGroup = リクエストを分散、ルーティングするサービス

ALB = LB の中の Application Load Balancer



[LB Doc]
[TargetGroup Doc]



AWS 環境を見ていこう
~Elastic Container Service~

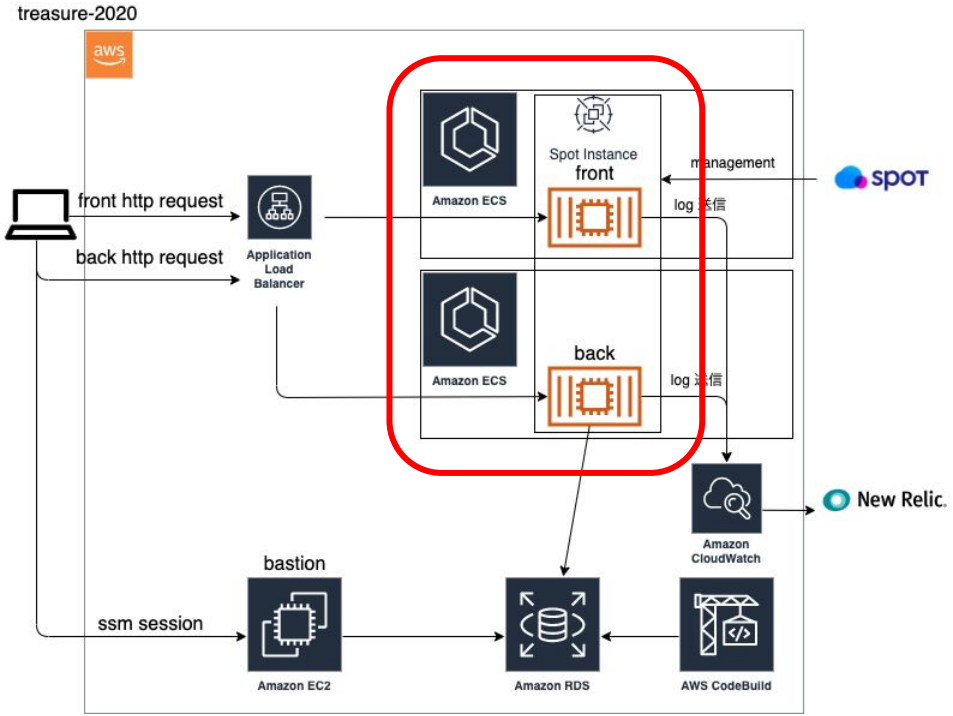
ECS = Docker Container を AWS 上で簡単に実行、停止、管理できるサービス

frontend, backend docker はこのサービスの上で動いています。

起動後に前スライドで説明した Target Group に登録されてリクエストを受けれるようになっている

起動時に使う docker iamge は ECR (Elastic Container Registry) に置いている

ECS Doc



AWS 環境を見ていこう
~Spot Instance/Spot~

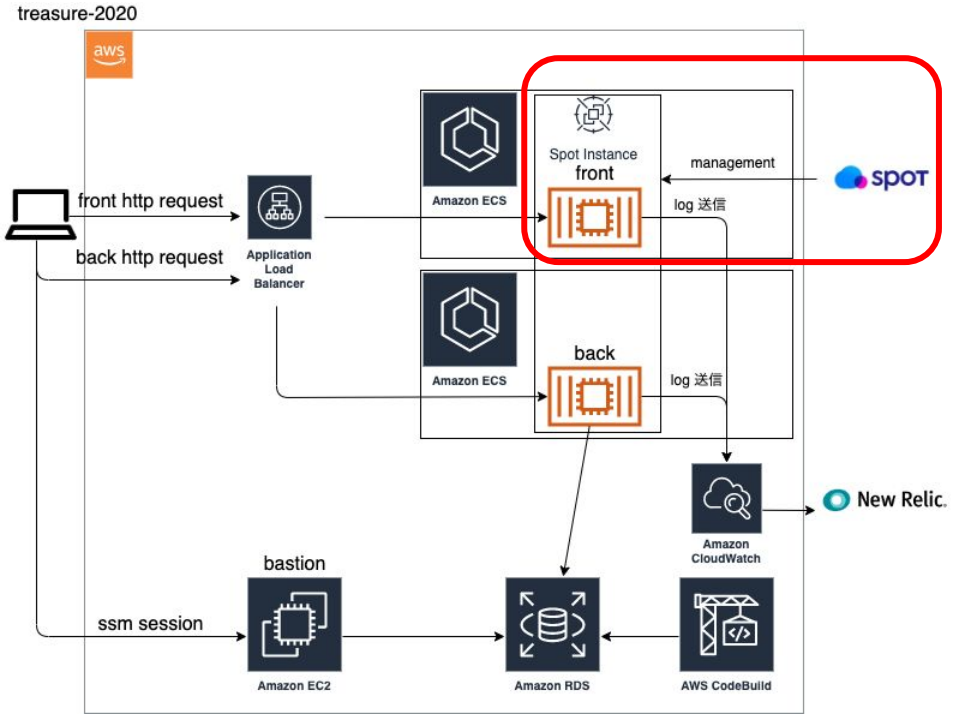
Spot Instance = お安い EC2 instance

制約はあるけど、AWS 上で余っている EC2 Instance リソースを安く使えるサービス

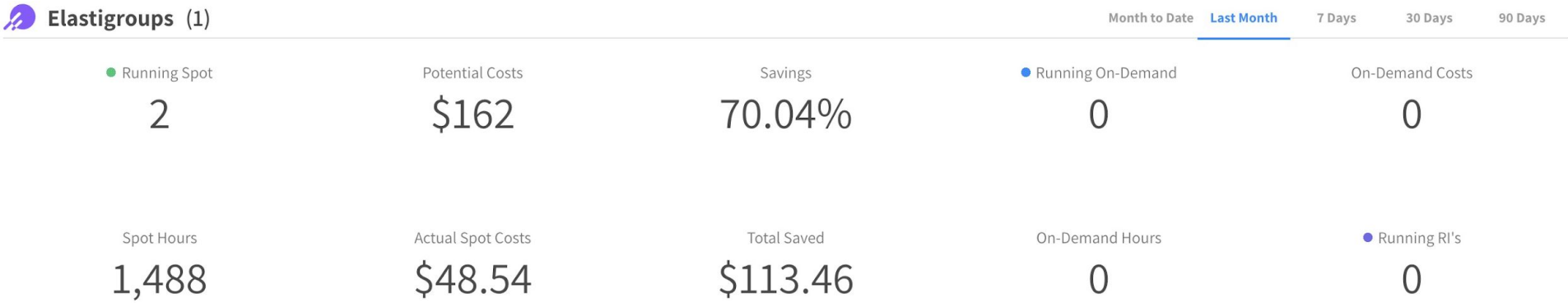
Spot = Spot instance をマネジメントしてくれる外部サービス (旧名 Spotinst

今年の Treasure の ECS は Spot で立ち上げた Spot Instance 上で動いている

Spot Instance Doc
Spot Doc



Spot Console



AWS については
まだまだ説明仕切れてない部分があり、
興味がある方は @syota, @tomokatsu まで
質問してもらえると喜びます！w

個人チャットでも、**#infra** でも！

CI / CD について

そもそも CI/CD ってなに？

CI / CD 知ってる人いますか？

そもそも CI/CD ってなに？

- **CI 継続的インテグレーション**

  自動ビルド・自動テストが用意された環境で、
  頻繁にコードをマージする**習慣**

- **CD 継続的{デリバリー|デプロイメント}**

  CIの拡張
  検証環境・本番環境へ、**頻繁**にデプロイメントする**習慣**

なんで CI が必要？

本番環境には **正常なコード** しか
リリースしたく無い

なんで CI が必要？

正常なコードとは

- テストが通っている
- ビルド(コンパイル)できる

なんで CI が必要？

PR を作る度に**正常なコード**だと証明するのは大変

!! CI の導入 !!

なんで CD が必要？

本番環境に手作業でリリースしていると...

- エンジニア毎にリリース方法が統一されない
- 手順書必須
- リリースに時間がかかる
- めんどくさくなって後でまとめてやりたくなる

なんで CD が必要？

もし、
まとめてリリースした時にバグが含まれていると...

あああああああああああああああああああああああああああああああ
あああああ～～直ぐに戻せない... orz

なんで CD が必要？

みなさん、オワりたいですか？

!! CD の導入 !!

CI/CD にはどんなツールを使えばいい？

CI ツールによって得意不得意はあります。
興味があればいろんな CI ツールを触ってみよう！

- Jenkins
- CircleCI
- Travis CI
- Concourse CI
- Github Actions

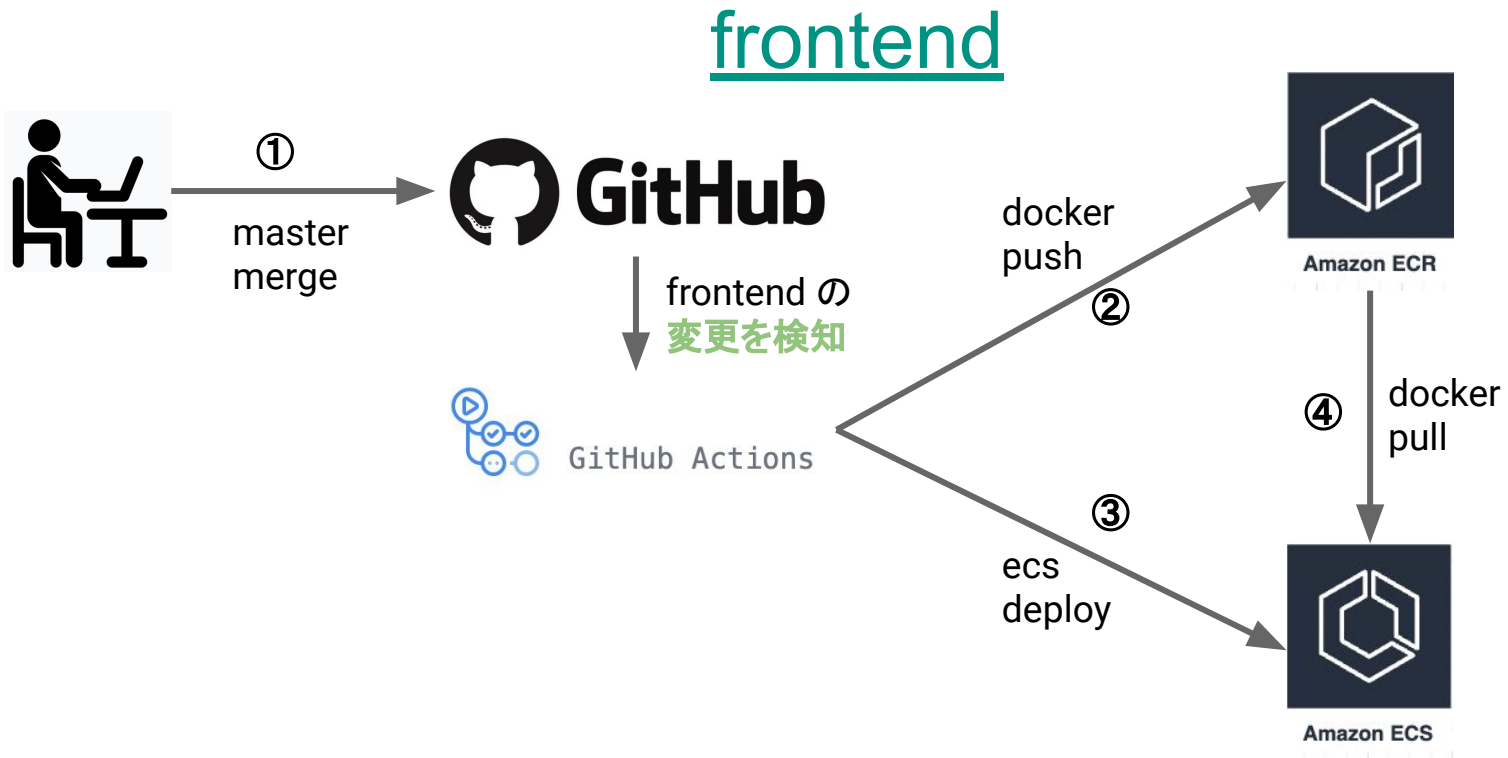
Github Actions

作りがシンプルだったので今回 Treasure で採用しました
開発者でも簡単に workflow 手順を変更できる
Github UI に統合されている

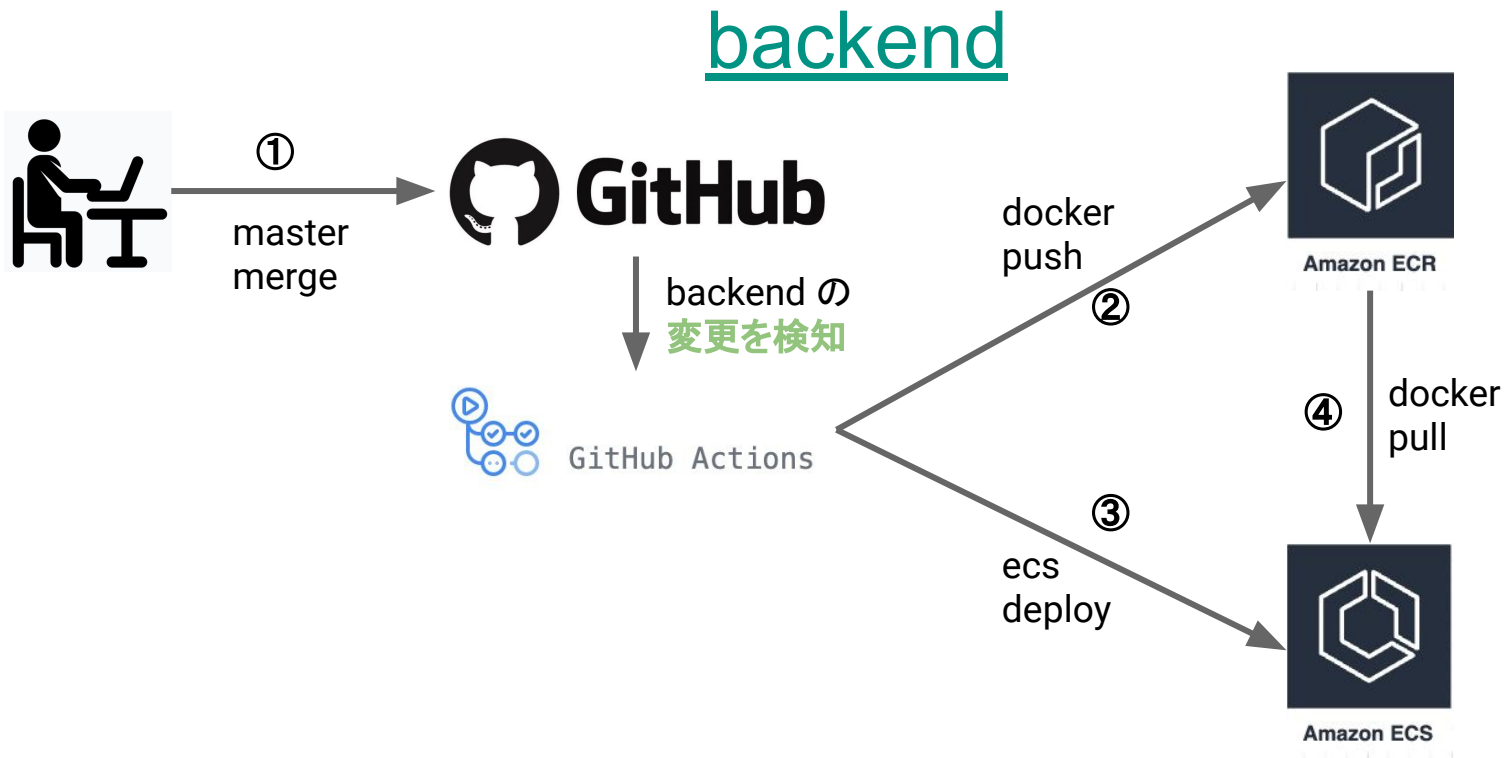
改めて

https://github.co.jp/features/actions

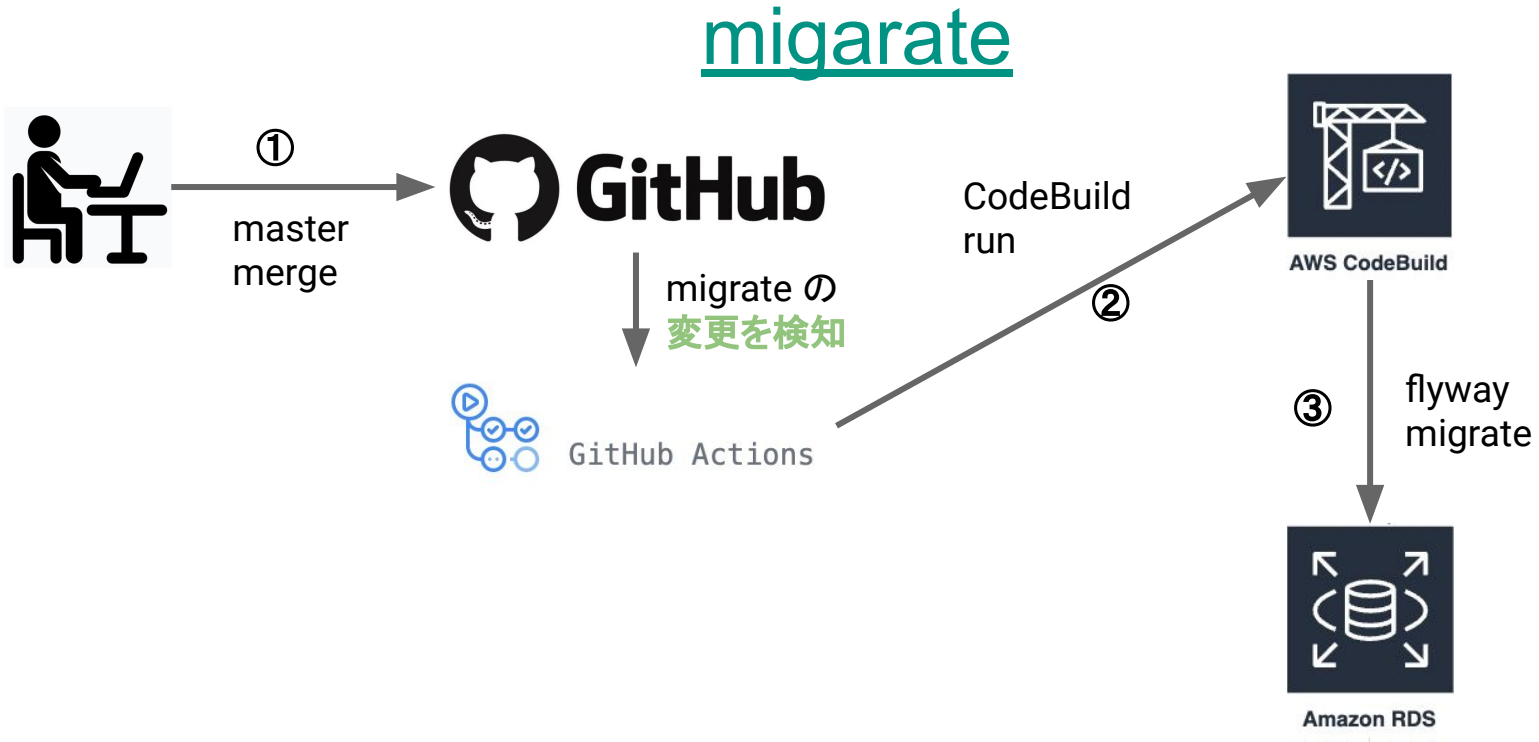
どんな CI / CD を用意したの？(frontend)



どんな CI / CD を用意したの？(backend)



どんな CI / CD を用意したの？(migrate)



手を動かそう!!

チーム で試しに CI / CD を回してみよう

チーム毎に別れて CI / CD を実際に体験してもらいます

今からやる事

- チームの中で誰か1人決める
- 決まった人は画面共有する
- 各自のチームリポジトリを clone してくる
  Github リポジトリは treasure-2020-{a-f}
- frontend ログインリンクの文言を変更する PR を作る -> サンプル PR
- チームのみんなに PR をマージしていいか確認して、マージボタンをポチる
- Github Actions を観察したり、終わったらデプロイされているか確認する

手を動かそう!!

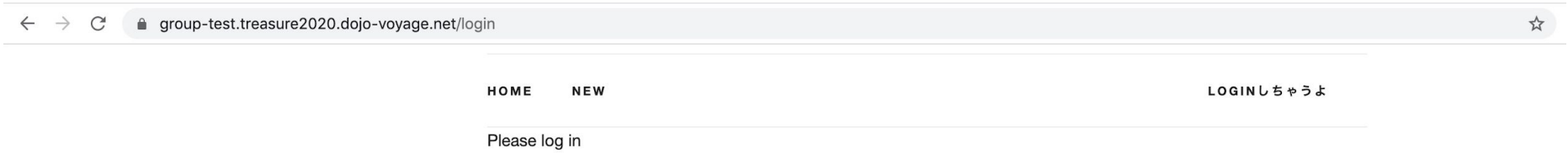
チーム で試しに CI / CD を回してみよう

デプロイが終わったら以下のURLで変更されているか確認

https://group-{team-name}.treasure2020.dojo-voyage.net/login

手を動かそう!!

チーム で試しに CI / CD を回してみよう



グループワーク中…

グループワーク終了！！

監視について

なぜ監視をするの？

サービスが健康であることを把握するため

サービスが健康？

サービスが想定通りに稼働している状態を健康と表現する

不健康になる要因とは

要因は様々、障害は起こるもの
なので気付けるようにする必要がある

監視って具体的にナニをするの？

サービスが健康であることを把握するために（自動で）情報収集する

どういう情報があれば健康状態が把握できるでしょう

- 外形監視（ページ見えてる？）
- リクエスト数（リクエスト流れてきてる？）
- アクセスログ（5xx エラーたくさん出てない？）
- アプリケーションログ（なんかエラー出てない？）
- AWSの健康状態（https://status.aws.amazon.com/）
- リリースタイミング（直前のリリースに影響してない？）
- etc ...

どうやって監視するの？

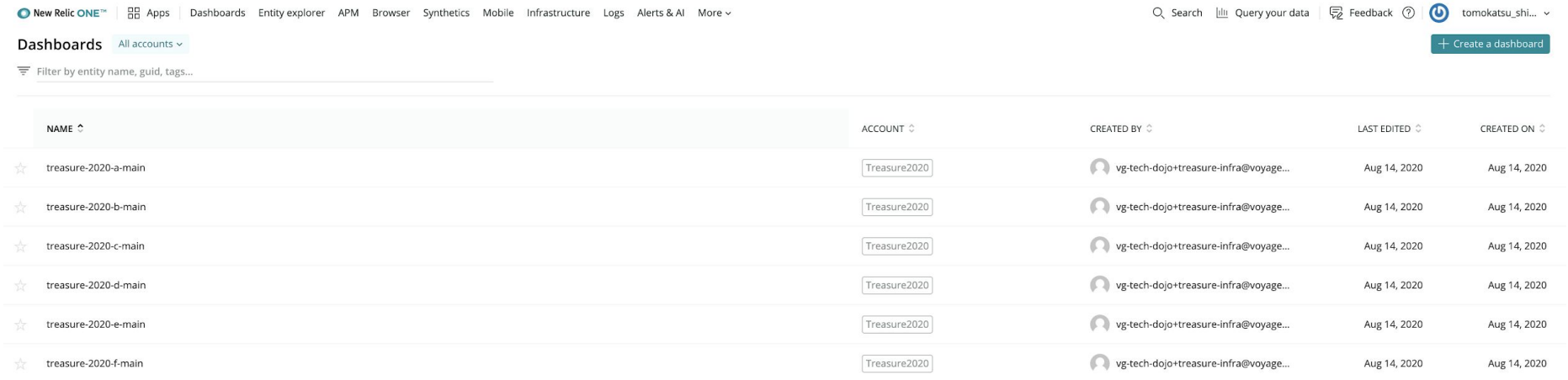
今回はアプリケーションの健康状態を見る為に

を用意しました

アカウントはすでに配ったので Let’s 監視！！

NewRelic Dashboards の使い方

https://one.newrelic.com/launcher/dashboards.launcher

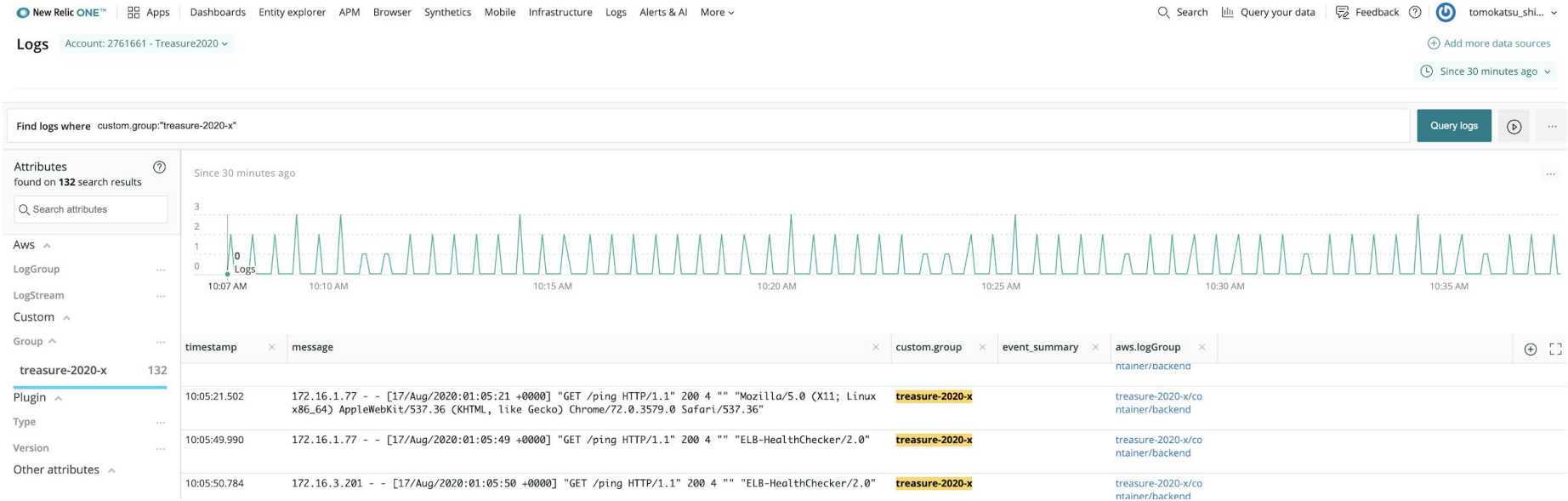


各チームにメインのダッシュボードを1つ作ってあります

外形監視の履歴、リクエスト数、HTTPステータスコード、ECSのコンテナ数、CI/CD実行履歴、etc...

NewRelic Logs の使い方

https://one.newrelic.com/launcher/logger.log-launcher



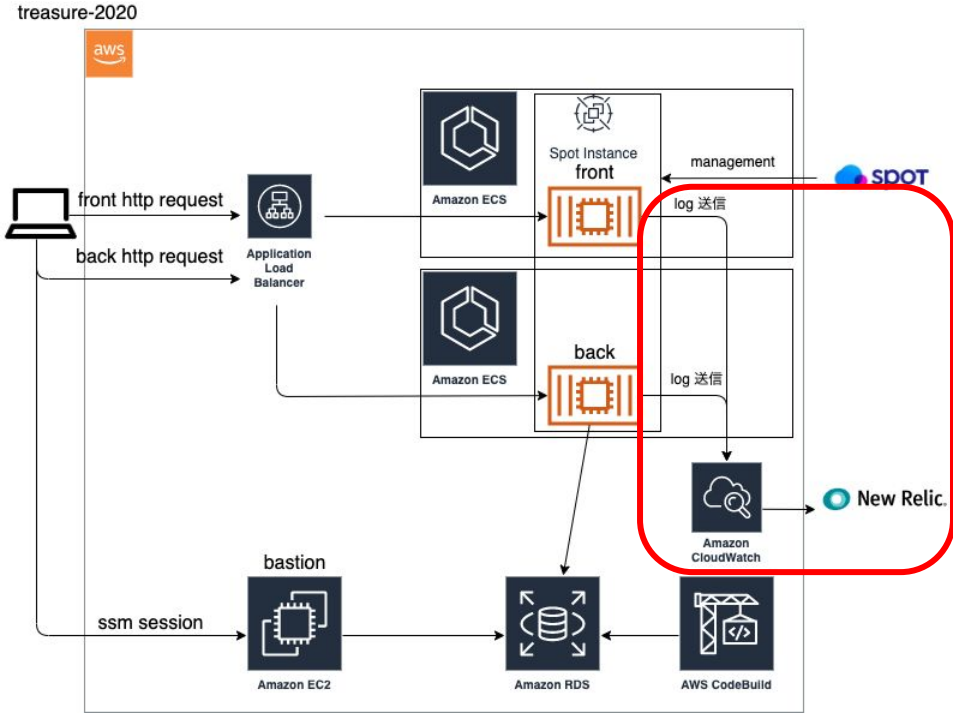
Query: custom.group:"treasure-2020-x" -"ELB-HealthChecker/2.0"

忘れてたわけじゃないです。

この部分は監視の為のものです

ECS log
⬇️
CloudWatch Logs
⬇️
NewRelic

に流して見やすくしています



監視についてより詳しく知りたい！

成長を続ける広告配信プラットフォームのモニタリングを改善してきた話

Speaker: fluct SRE みっさん

https://speakerdeck.com/larufa/cheng-chang-wosok-keruguang-gao-pei-xin-puratutohuomufalsemonitaringuwogai-shan-sitekitahua

入門 監視 ――モダンなモニタリングのためのデザインパターン

https://www.oreilly.co.jp/books/9784873118642/

DevOps について

DevOps って？

DevOps とは、
アイデア (新しいソフトウェア機能、拡張リクエスト、バグ修正など) を
開発からプロダクションへと進め、実際にユーザーに価値提供できるようにする
までのプロセスを迅速化するためのアプローチを表します。

https://www.redhat.com/ja/topics/devops

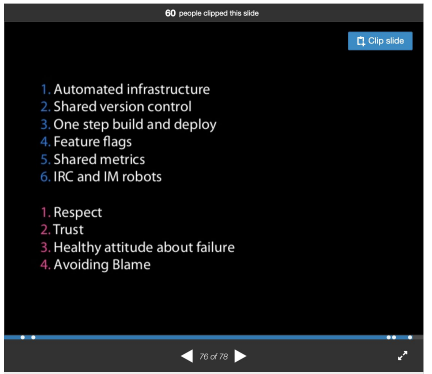
どんなアプローチが必要？

ツール（ 実践出来るものもある ）

1. 自動化されたインフラストラクチャ (Automated infrastructure)
2. バージョン管理システムの共有 (Shared version control)
3. ワンステップによるビルドとデプロイ (One step build and deploy)
4. フィーチャーフラグ (Feature flags)
5. メトリクスの共有 (Shared metrics)
6. IRCとインスタントメッセンジャーのBot (IRC and IM robots)

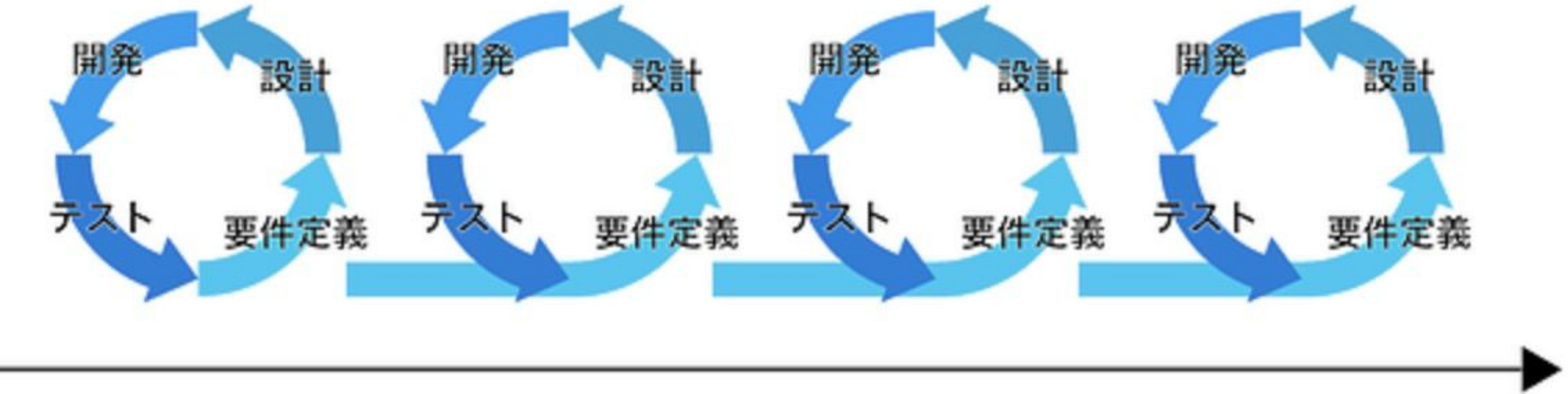
組織（ すぐ**実践**出来る ）

1. お互いを尊重する (Respect)
2. お互いを信頼する (Trust)
3. 失敗に対して健全な態度を取る (Healthy attitude about failure)
4. 相手を非難しない (Avoiding Blame)



https://www.slideshare.net/jallspaw/10-deploys-per-day-dev-and-ops-cooperation-at-flickr/76

小さく早くたくさん



いよいよ
午後からグループワークがはじまります
@ syota, @ tomokatsu は
全力で チーム開発をサポートするので
いつでも相談してください！！

ありがとうございました