FiNCを支えるインフラ技術
～　ECSとDevOps　～

FiNC

自己紹介

- 名前： 中村　郷史（なかむら　さとし）
- 所属： プロダクト本部 技術開発部 SREグループ
- 担当： Technical Lead in SRE
- 社歴： 2017年1月ジョイン。約1年半。
- よく使っているAWSのサービス

  ECS、CloudFormation

FiNC

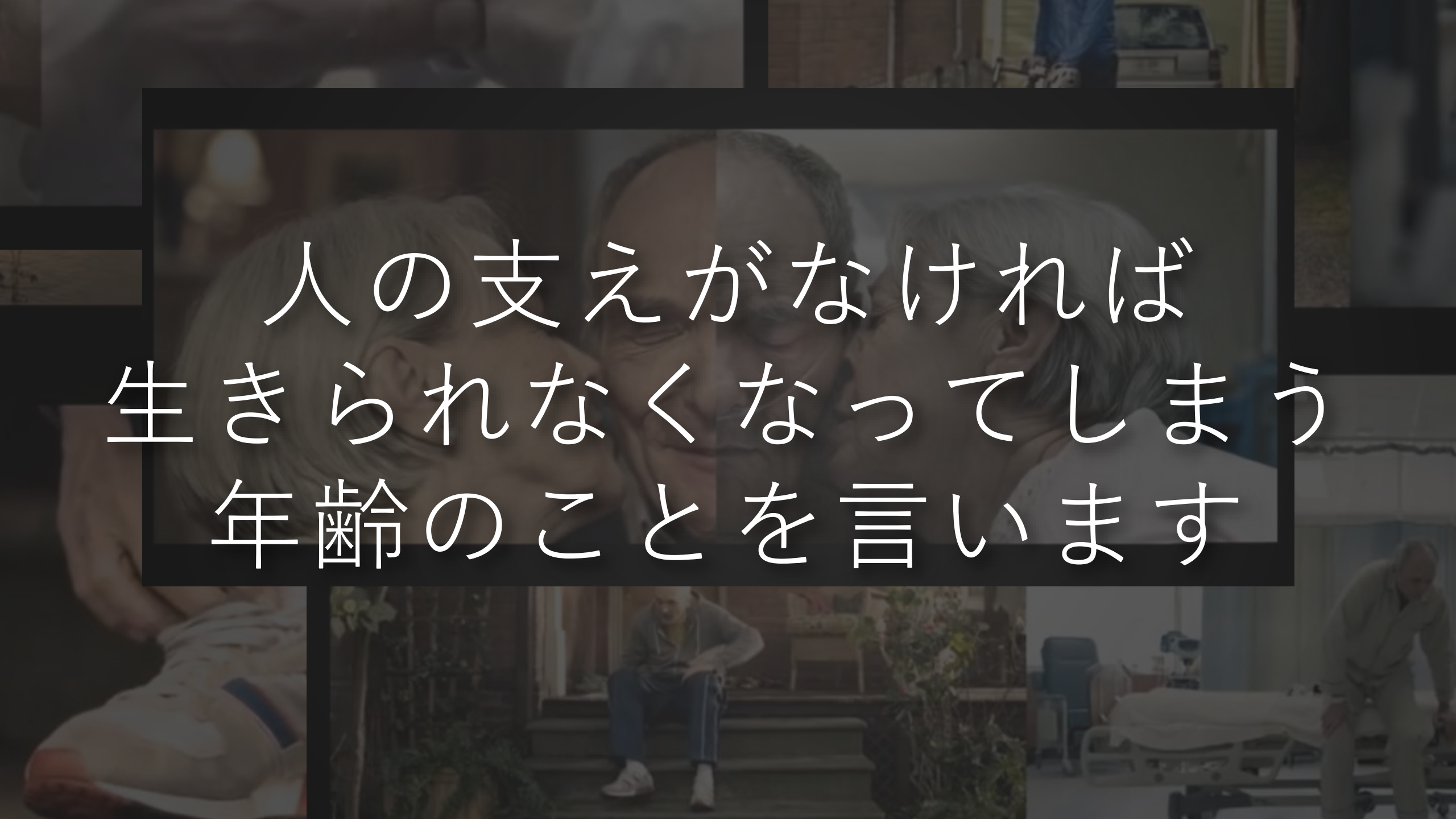
FiNC

人の行動を変え、健康を実現する
ヘルスケアカンパニー

< Our Vision >

一生に一度のかけがえのない
人生の成功をサポートする

「健康寿命」という言葉はご存知ですか？

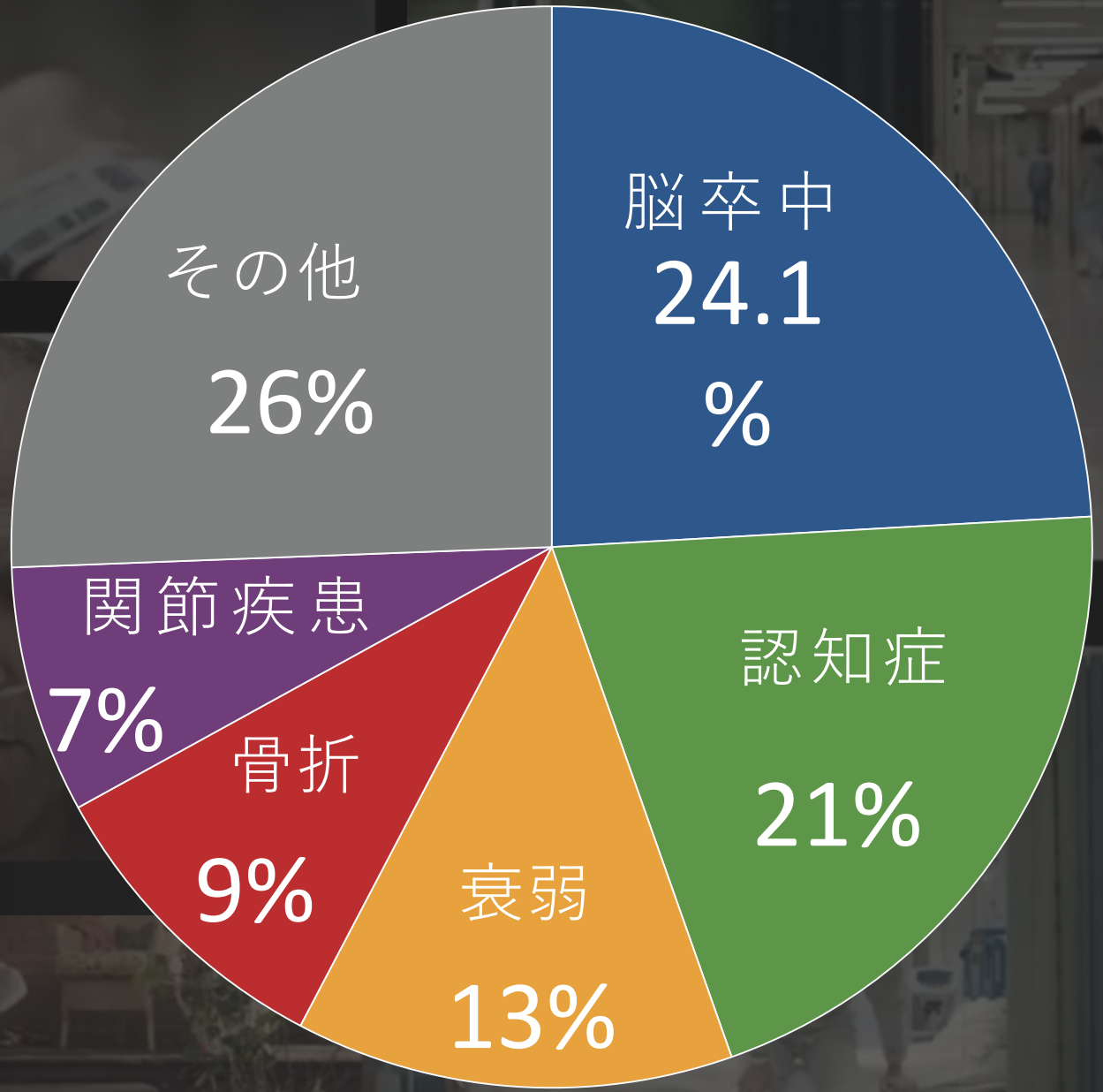


人の支えがなければ
生きられなくなってしまう
年齢のことを言います


(年)
60歳
65歳
70歳
75歳
80歳
85歳
平均寿命
健康寿命
約11年

寝たきりの原因

厚生労働省　国民基礎調査より



「生活習慣の改善」に必要な三要素

栄養　**運動**　**休養**

1. 何をすべきか分からない
2. 継続できない

Mission

すべての人にパーソナルコーチを

FiNCのサービス

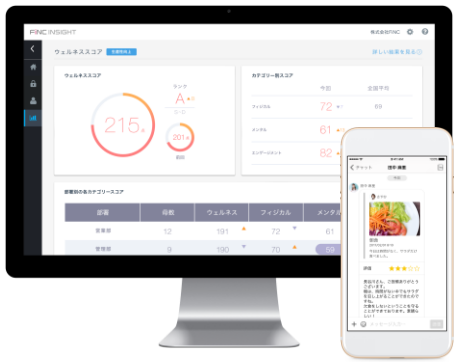
広告	有料課金	EC

FiNC インタラクティブコミュニケーション

FiNC プレミアム

FiNC ダイエット家庭教師

FiNC for ビジネス BUSINESS



FiNC FIT

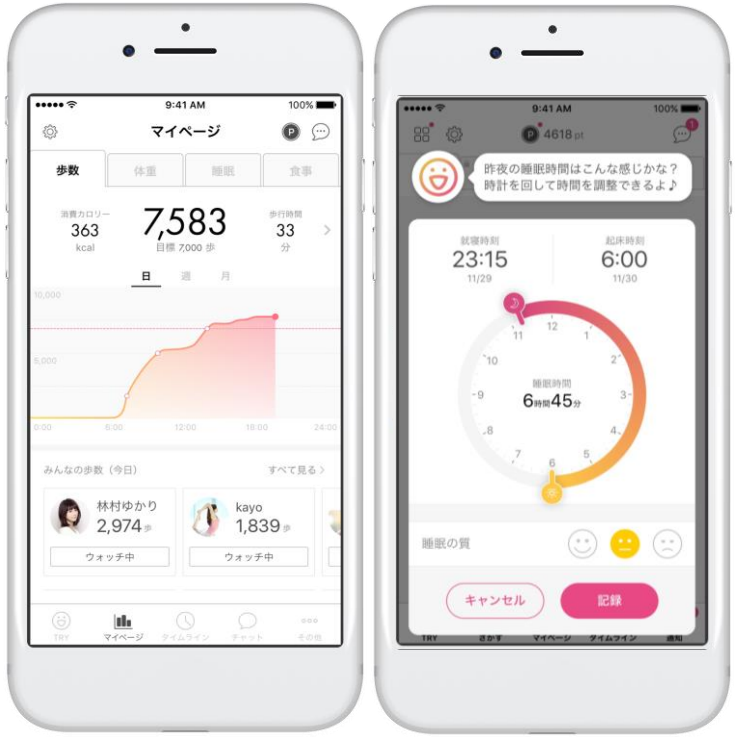
FiNC MALL

FiNC

ライフログがたまるほど、あなたにフィットする
パーソナルトレーナーAIアプリ

FiNCアプリ

ライフログの記録



歩数・睡眠時間が自動で記録
体重・食事・生理日なども

動画コンテンツ

モデルやアスリート、
栄養士/トレーナー等による
ヘルシーレシピ・フィットネス等の動画

SNS

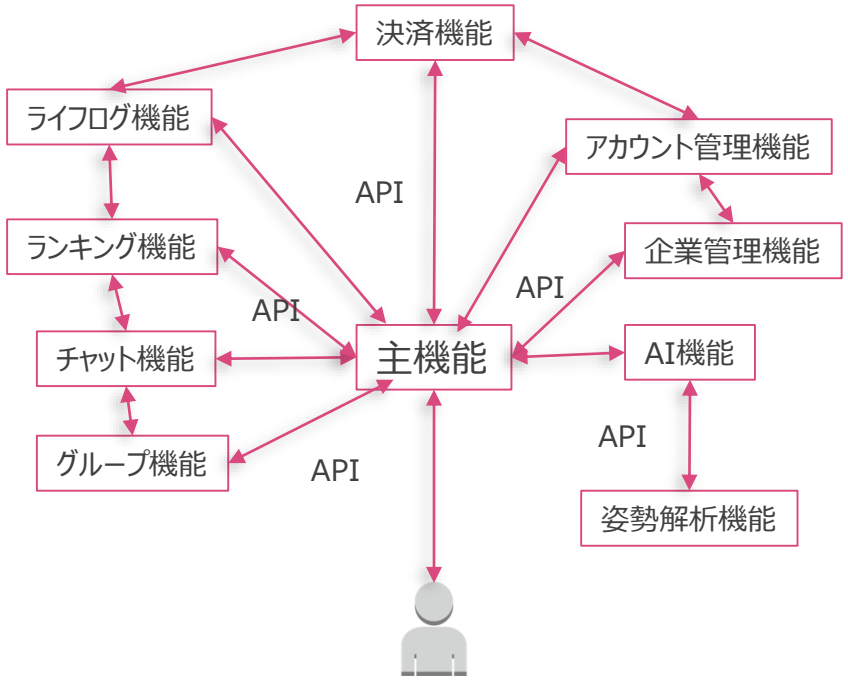
家族や友達、著名人・専門家と
楽しくコミュニケーション

FiNCのシステム

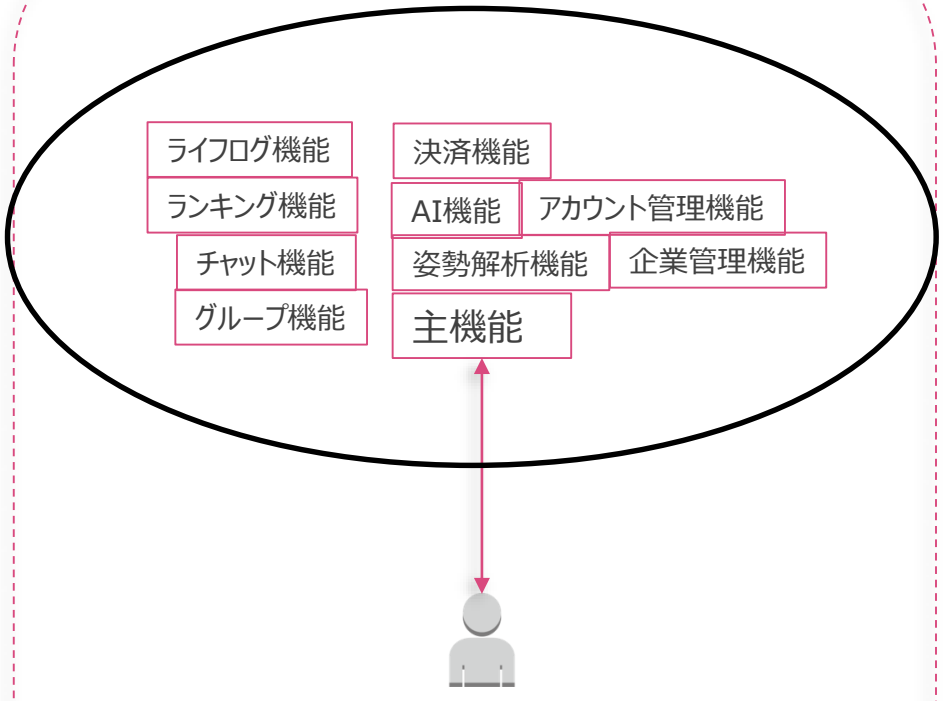
マイクロサービス・アーキテクチャ

FiNC

マイクロサービスとは



マイクロサービス・アーキテクチャ



モノリシック・アーキテクチャ

FiNC

なぜマイクロサービスか？

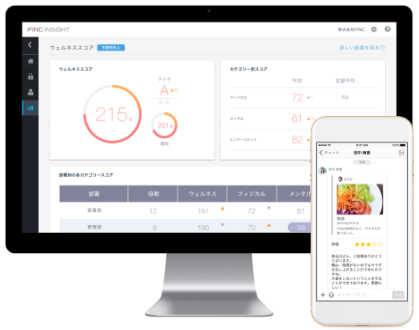
ヘルスケアの中で様々なサービスを提供するという
ビジネス構想があったから

FiNC インタラクティブ
コミュニケーション

FiNC プレミアム

FiNC
ダイエット家庭教師

FiNC for ビジネス
BUSINESS



FiNC FIT

FiNC MALL

アジェンダ

- なぜECSにしたのか
- ECS Tips
- ECS 事件簿

FiNC

アジェンダ

- なぜECSにしたのか
- ECS Tips
- ECS 事件簿

FiNC

ECS以前の構成の振り返り

- EC2
- Ansible
- Capistrano

FiNC

EC2時代の問題点

- スケール性能

  サーバの起動、構築からサービスに組み込まれる時間

- 余剰リソース

  余剰CPU、メモリの有効活用

- 多様な構成管理

  OS、言語、フレームワーク、バージョン等

- 異なる設定管理

  非同期処理、DB設定、環境設定

FiNC

なぜECSにしたのか

- スケール性能

  コンテナ（サーバ）増強の速さ

- リソース最適化

  1台のサーバリソースを使いきれる

  （最大70コンテナ稼動）

- 多様な構成管理、異なる設定管理

  SREが中央管理するのではなく、開発側に委譲

FiNC

なぜECSにしたのか

- スケール性能

  コンテナ（サーバ）増強の速さ

- リソース最適化

  1台のサーバリソースを使いきれる

  （最大70コンテナ稼動）

- 多様な構成管理、異なる設定管理

  SREが中央管理するのではなく、開発側に委譲

FiNC

構成管理ファイル

- 多様な構成
  - OS、言語、フレームワーク、バージョン等
    - 初期：Ansible（Ops）
    - 移行期：Dockerfile（Ops）
    - 現在：Dockerfile（Dev）
- 異なる設定
  - 非同期処理、DB設定、環境設定
    - 初期：Capistrano（Ops）
    - 移行期：Envfile（Ops）
    - 現在：Envfile（Dev）

※Envfile・・・環境変数によってデプロイフロー使い分けているファイル

FiNC

ECS移行期間、方法

- 移行期間

  約半年

- 移行方法

  Route53 RoutingPolicy: Weighted

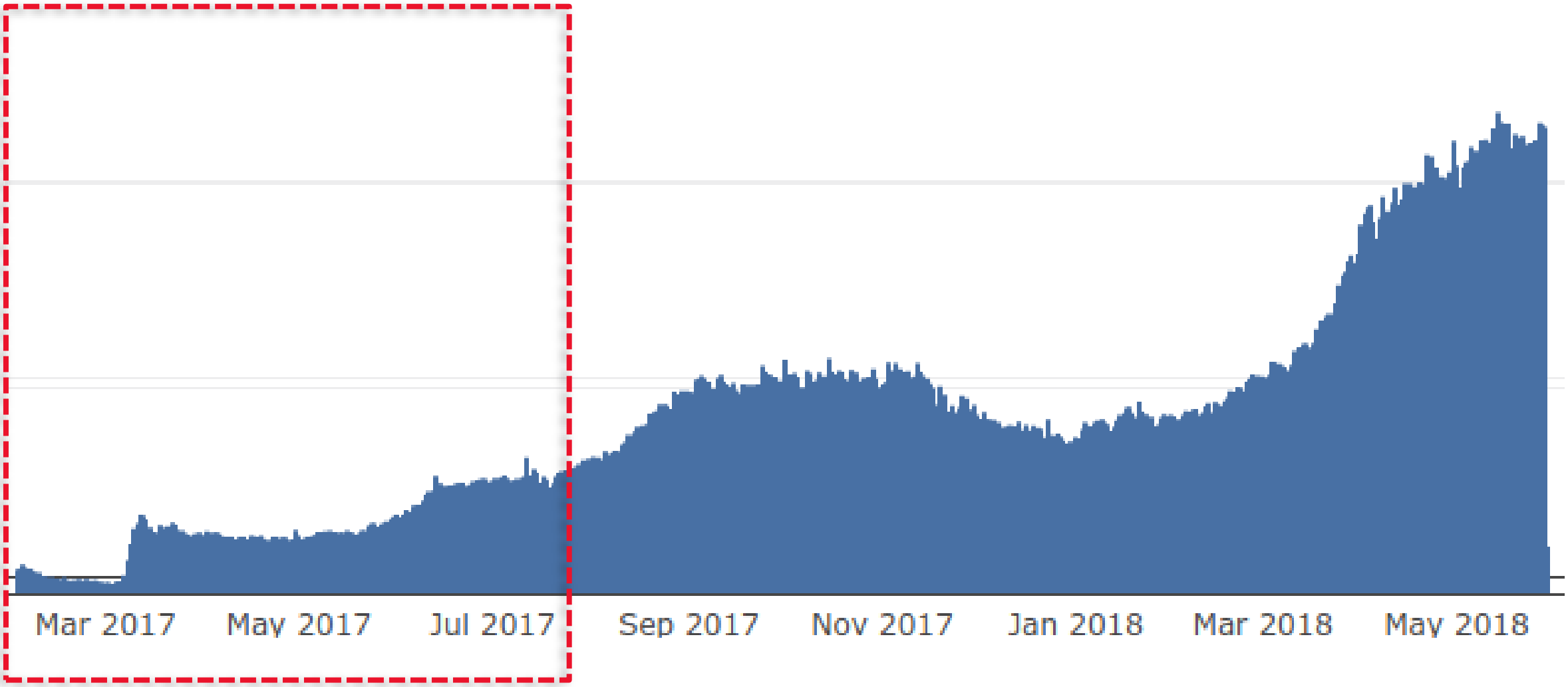
  EC2 99%

  ECS 1% => 5% => 30% => 100%

  ・・・1日放置 ⇄ エラー、リソース確認

FiNC

トラフィック推移


Mar 2017
May 2017
Jul 2017
Sep 2017
Nov 2017
Jan 2018
Mar 2018
May 2018

ECSに移行した結果

- スケール性能

  急激なトラフィックに耐えられるようになった

- 余剰リソース

  集約率向上 ＝> 1コンテナインスタンスに平均約20タスク

- 多様な構成

  開発者がSREに依頼することなく変更が可能

- 異なる設定

  開発者がSREに依頼することなく変更が可能

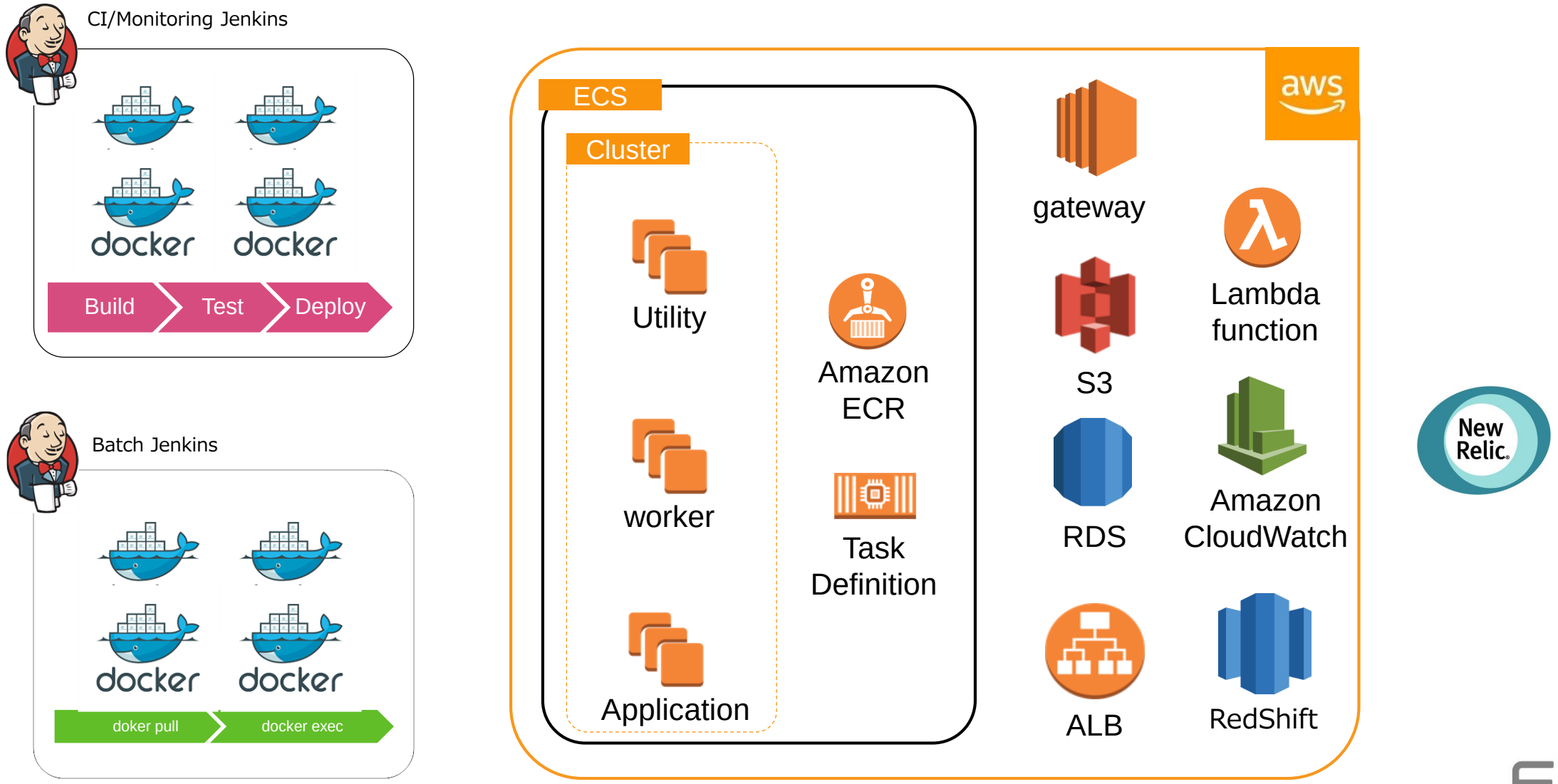
FiNC

アジェンダ

- なぜECSにしたのか
- ECS Tips
- ECS 事件簿

FiNC

発表内容の全体像


CI/Monitoring Jenkins
docker
docker
Build
Test
Deploy
Batch Jenkins
docker
docker
doker pull
docker exec
ECS
Cluster
Utility
worker
Application
Amazon ECR
Task Definition
aws
gateway
S3
RDS
ALB
Lambda function
Amazon CloudWatch
RedShift
New Relic

Tips1

DBマイグレーション実行方法

FiNC

デプロイフロー

Build → **Test** → **Deploy**

- docker run
- package install
- db create

- test code

- （image upload）
- migration
- ECS TaskDefinition update
- ECS Service update

FiNC

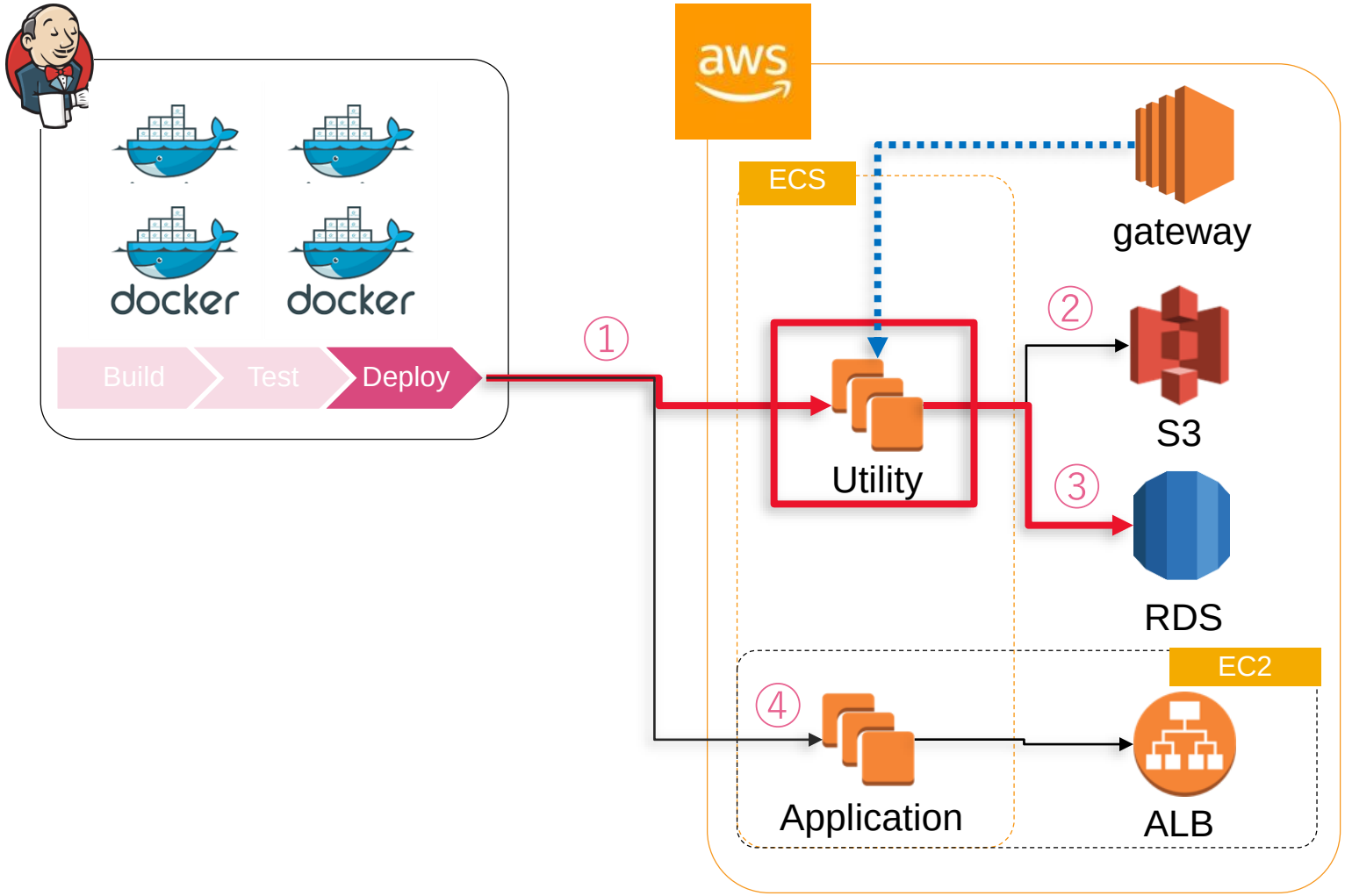
どうしたか？

アプリケーションクラスターと

同じクラスターを作る

FiNC

デプロイフロー



デプロイフロー

1. utilityクラスターのサービス更新
2. 画像アップロード
3. migration
4. Applicationクラスターのサービス更新

rails consoleの実行

Tips2

テスト高速化

FiNC

ある日・・・

テストめちゃくちゃ遅いんですけど

FiNC

デプロイフロー（再掲）

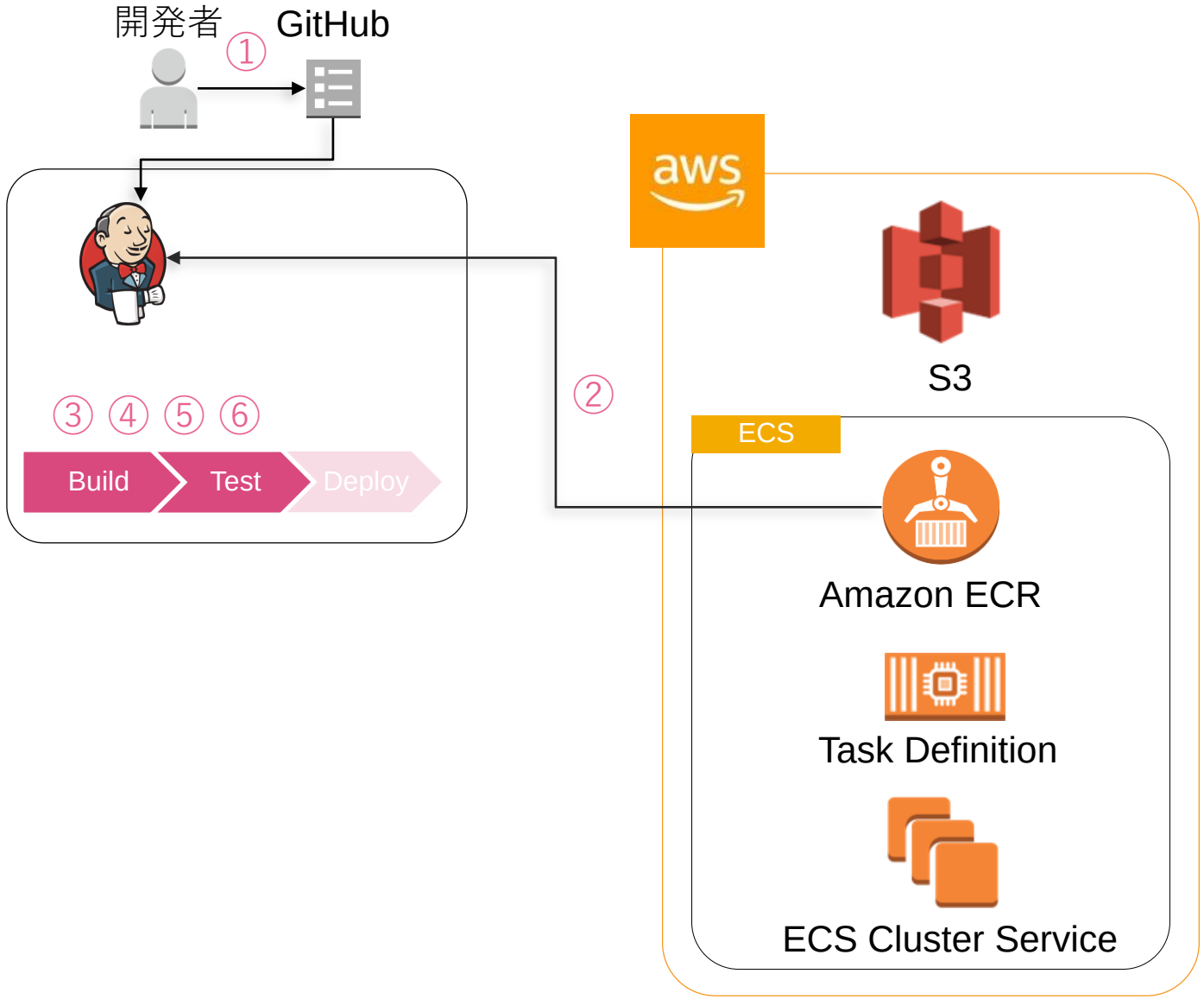
- docker run
- package install
- db create

- test code

- （image upload）
- migration
- ECS TaskDefinition update
- ECS Service update

FiNC

テストフロー



テストフロー

1. 開発者がGitHubにpush/merge
2. イメージをpull
3. コンテナを起動
4. 必要パッケージをインストール
5. db create
6. テストコード実行

どうしたか？

環境に変更がない場合はビルド処理をスキップ

FiNC

環境構成チェック方法

対象ファイル、ディレクトリをSHA256に変換

- インストール

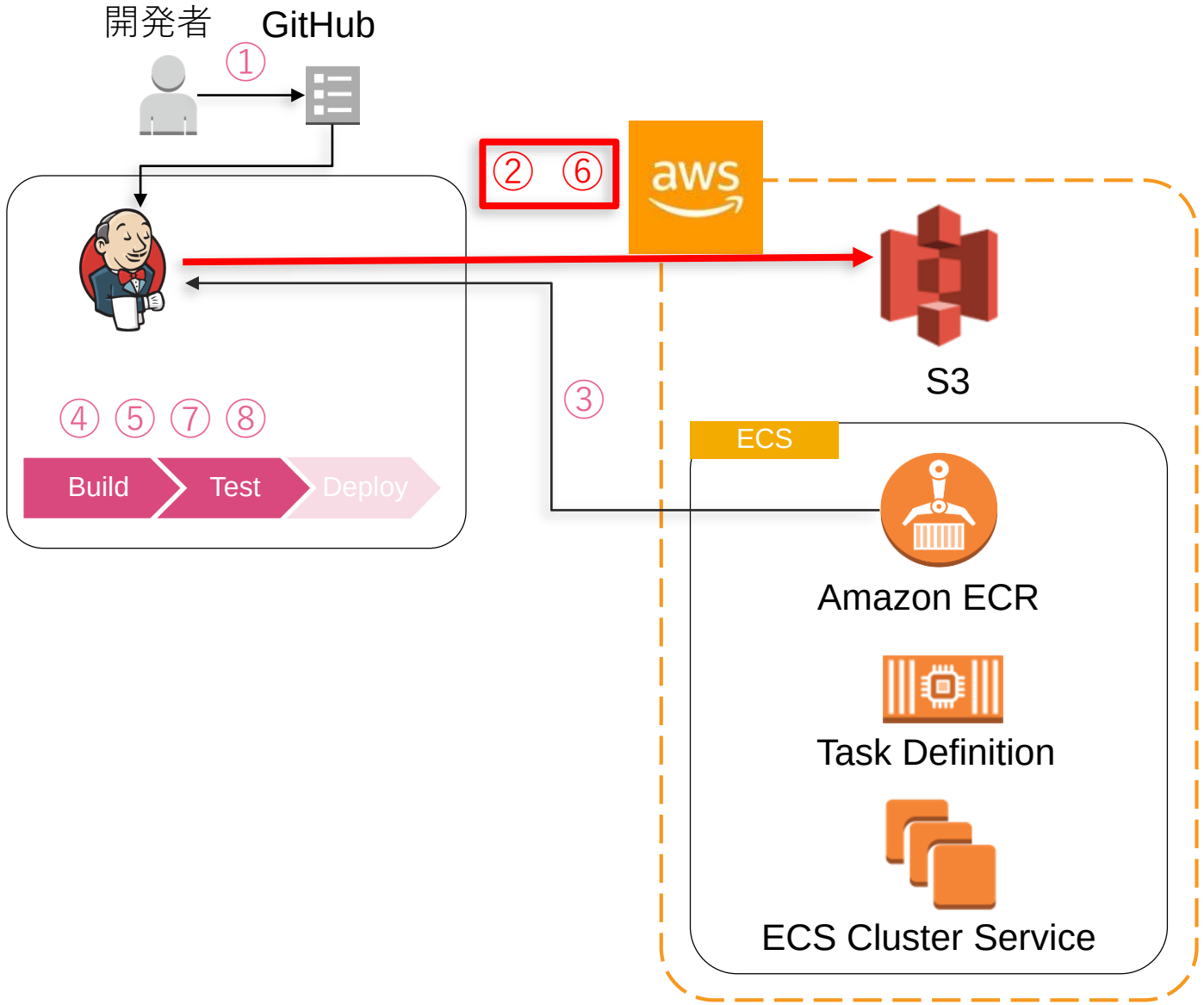
  Dockerfile、Gemfile

- マイグレーション

  db/migrate

FiNC

テストフロー（改善版）



テストフロー

1. 開発者がGitHubにpush/merge
2. config check
3. ECRからインストール済みイメージをpull
4. コンテナを起動
5. 必要パッケージをインストール
6. config check
7. db create
8. テストコード実行

FiNC

実測値

15分 -> 5分

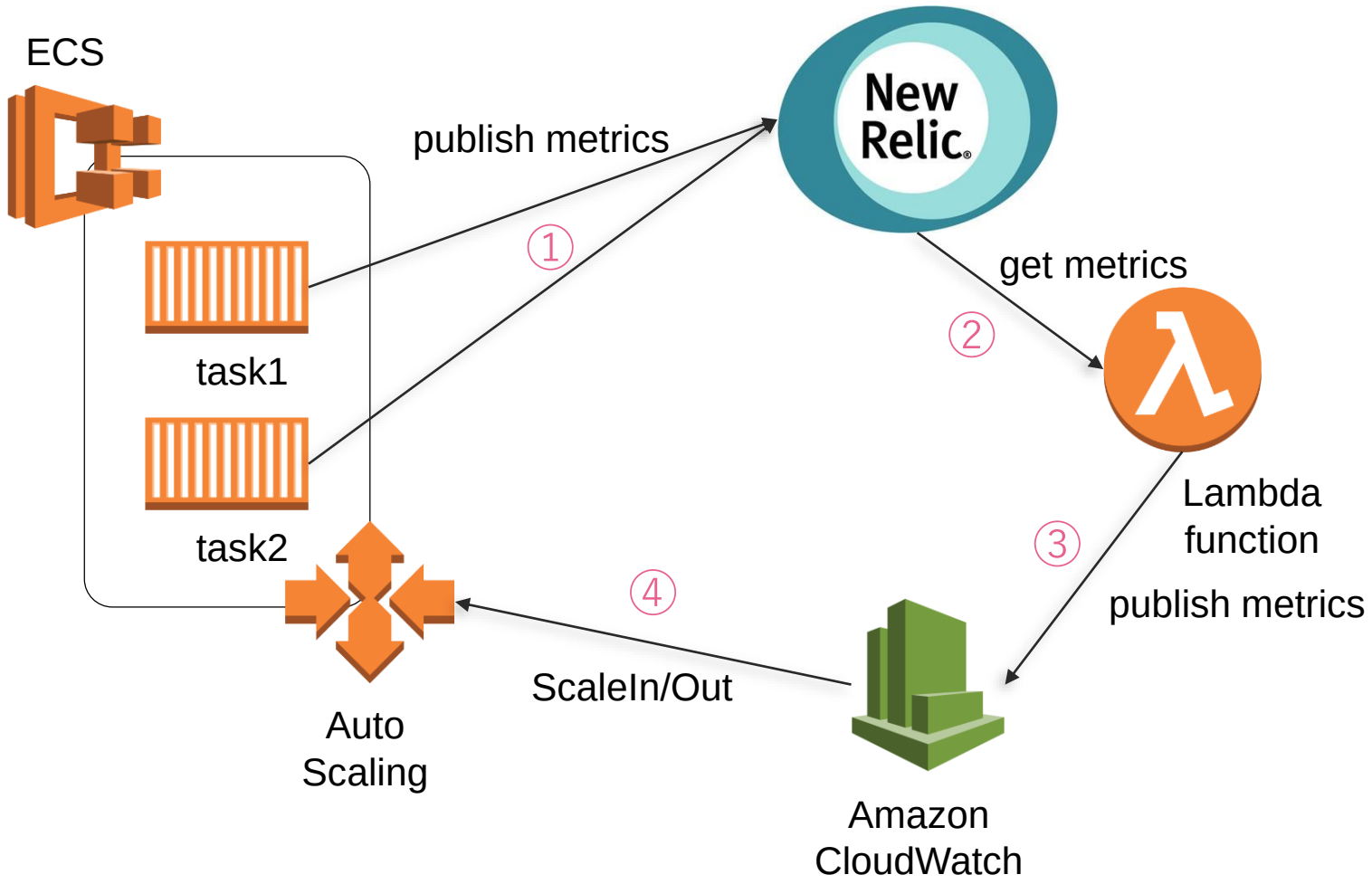
FiNC

Tips3

オートスケール方法

FiNC

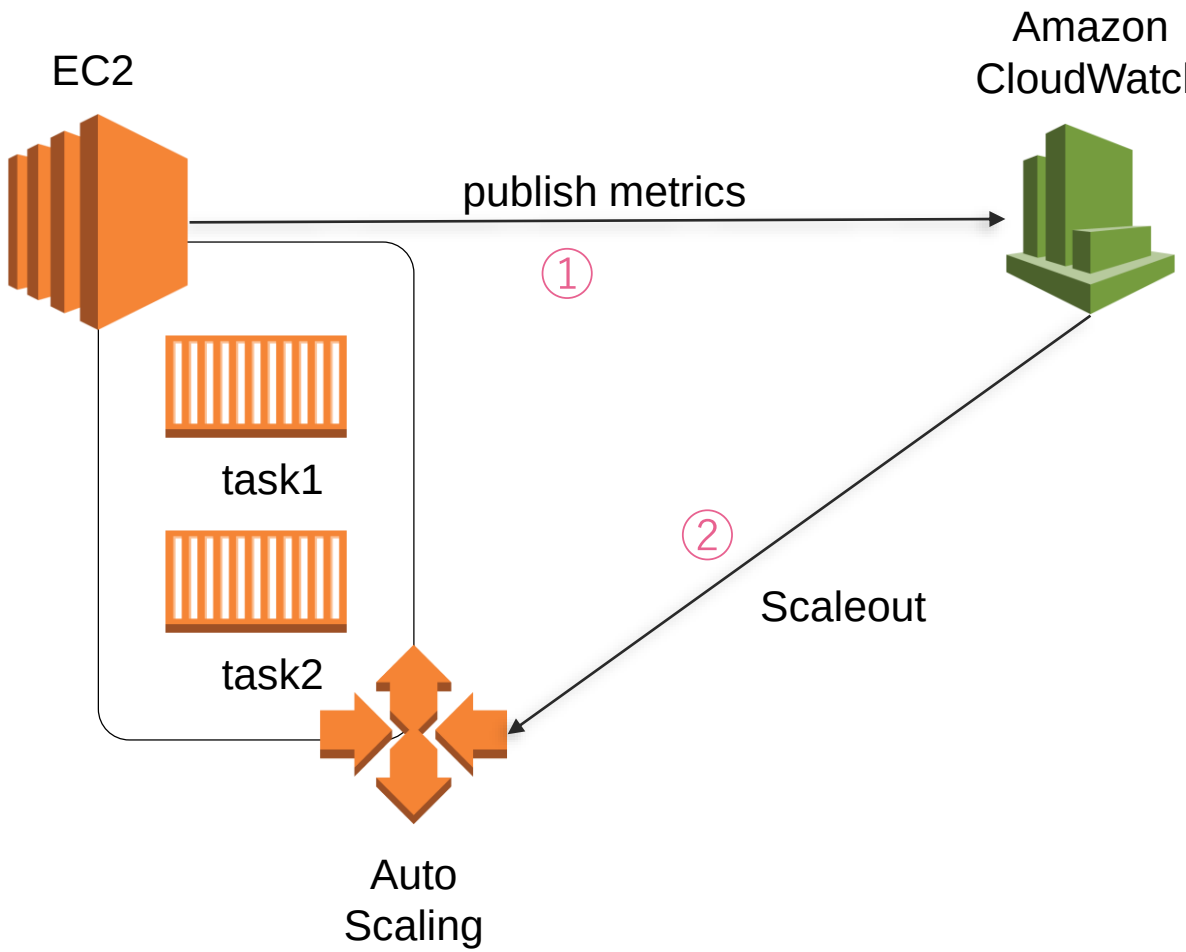
スケールイン/アウト（ECS）



スケールイン/アウト

1. unicorn worker num
2. lambda function
3. 1.をCloudWatchに送る
4. トリガー発動

スケールアウト（EC2）

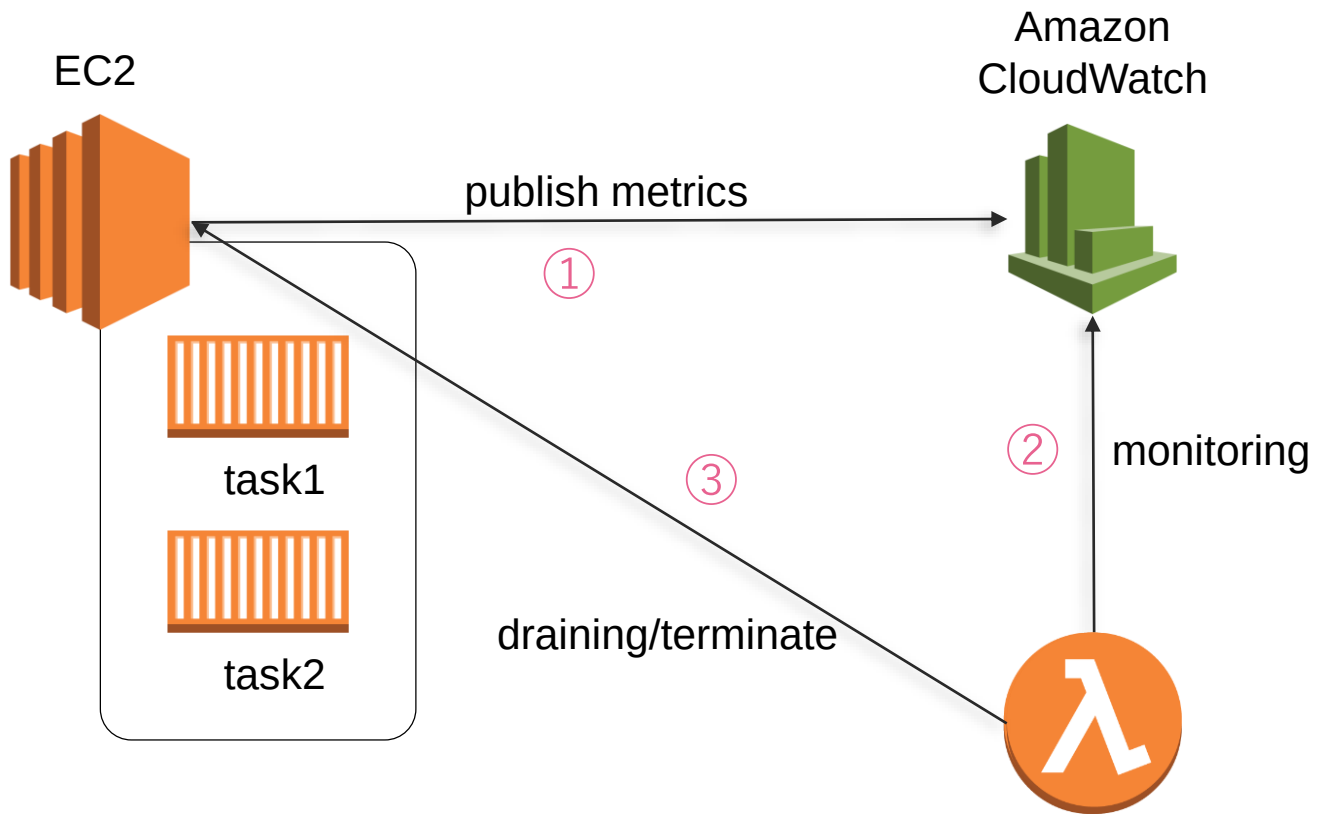


スケールアウト

1. CPU Utilization
2. Scale Out

FiNC

スケールアウト（EC2）



スケールアウト

1. CPU Utilization
2. monitoring
3. draining/terminate

Tips4

バッチ処理実行方法

FiNC

EC2時代

cronで実行

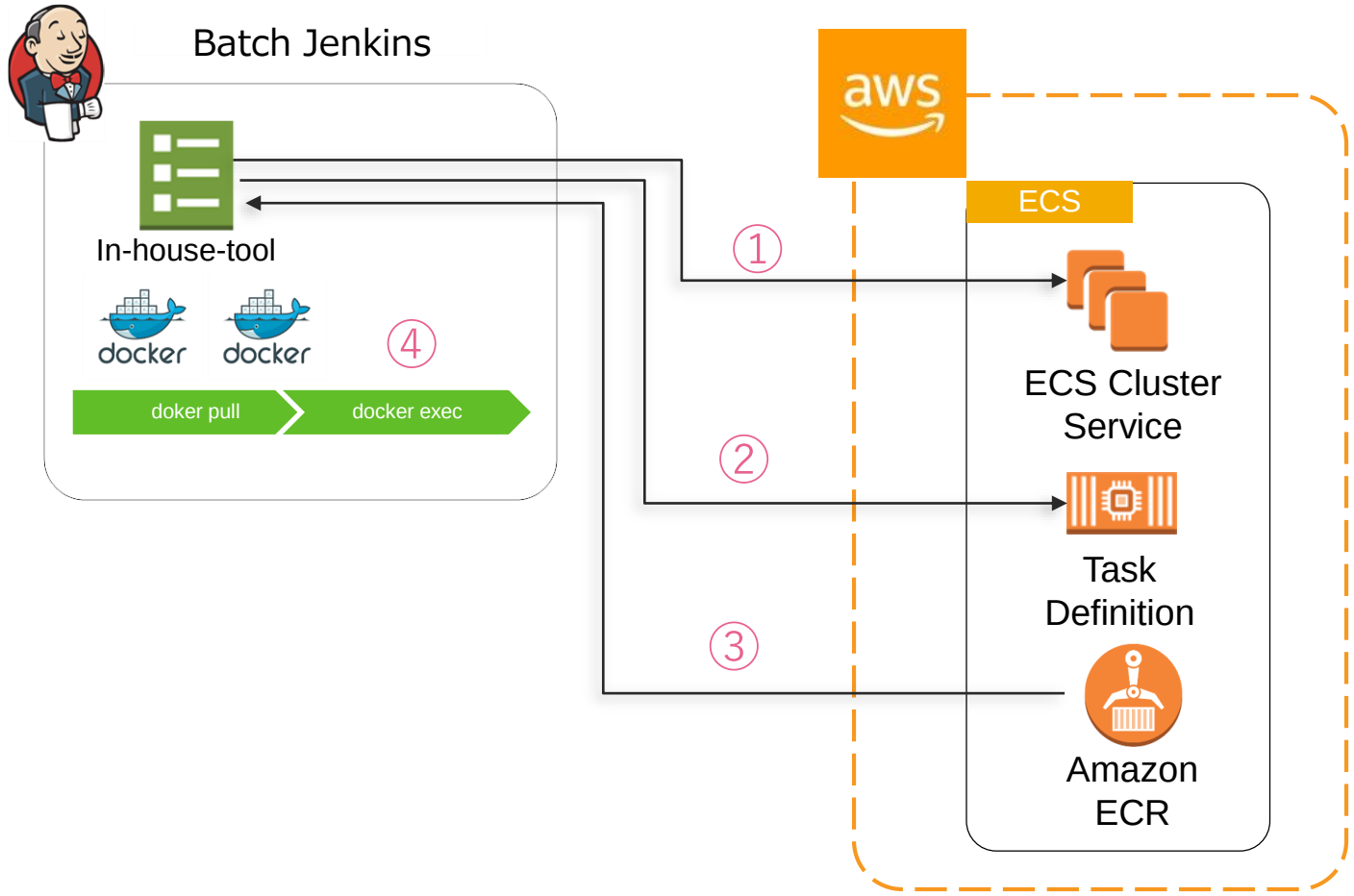
FiNC

どうしたか？

バッチ用のJenkins

FiNC

バッチ用Jenkins



ジョブ例

ServieName=AAA

Container=rails

batch-tool bundle exec rake XXX

バッチ実行方法

1. タスク定義を検索
2. リポジトリとタグを検索
3. docker pull
4. コマンド実行

FiNC

Tips5

Dockerfileの管理

FiNC

構成管理ファイル

- 多様な構成

  OS、言語、フレームワーク、バージョン等

  Dockerfile

- 異なる設定

  非同期処理、DB設定、環境設定

  Envfile

FiNC

Devが管理

- 管理しているもの

  Dockerfile

  Envfile（環境変数）

  ・・・徐々に移行

- 管理していないもの

  オートスケールの設定

FiNC

アジェンダ

- なぜECSにしたのか
- ECS Tips
- ECS 事件簿

FiNC

問題1

fluentdサーバSPOF問題

FiNC

fluentdを使っている理由

- awslogs(当時)

  30GB/日 x 15サービス=450GB

  450GB x $0.76=$342/日 x 30day = $10,260/月

・・・fluentdを使った

FiNC

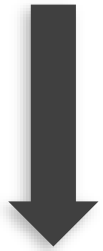
ある日・・・

なぜかコンテナが起動しない

FiNC

原因

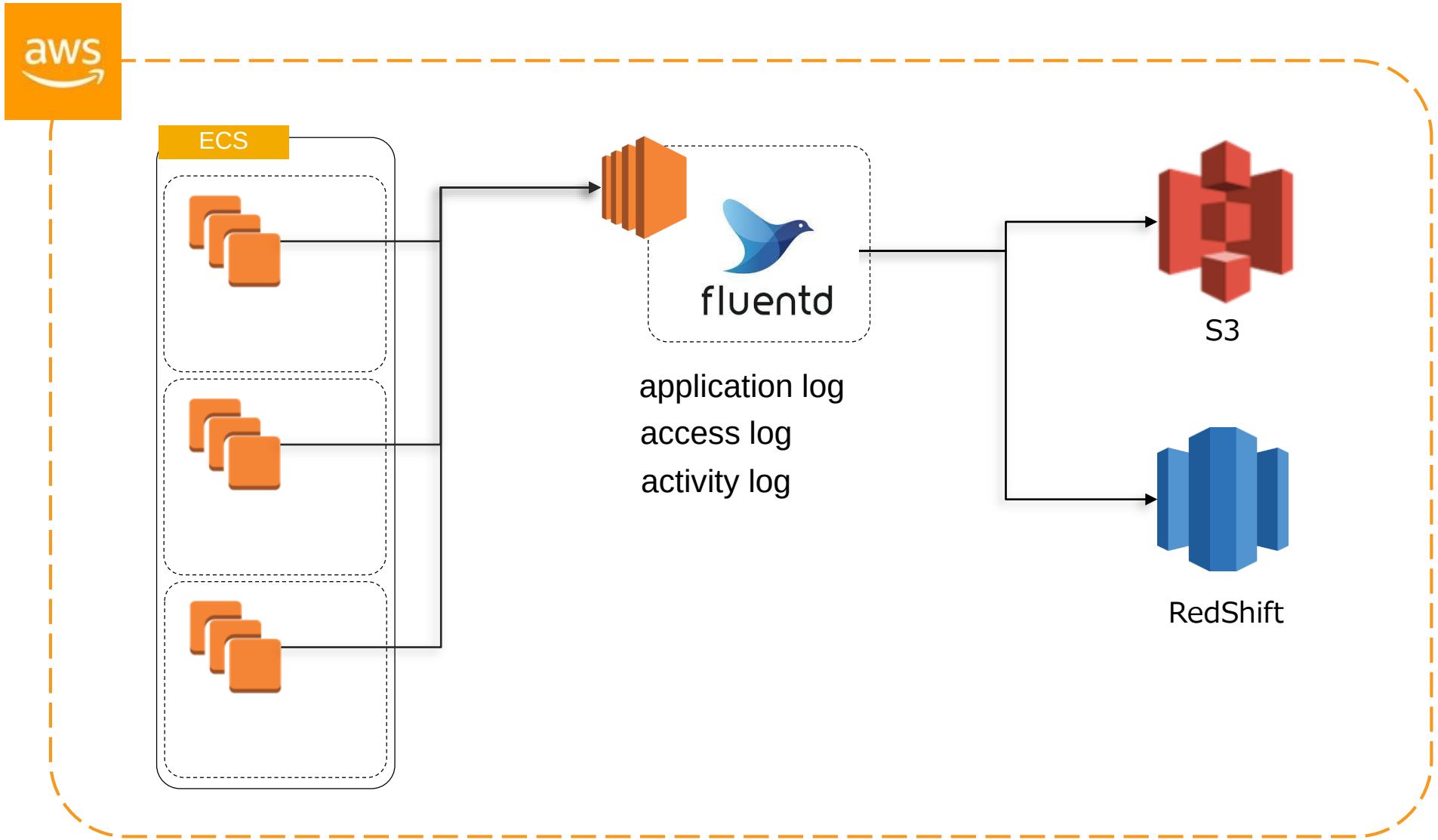
fluentdサーバに障害



コンテナが上がらない

FiNC

fluentdの設定



FiNC

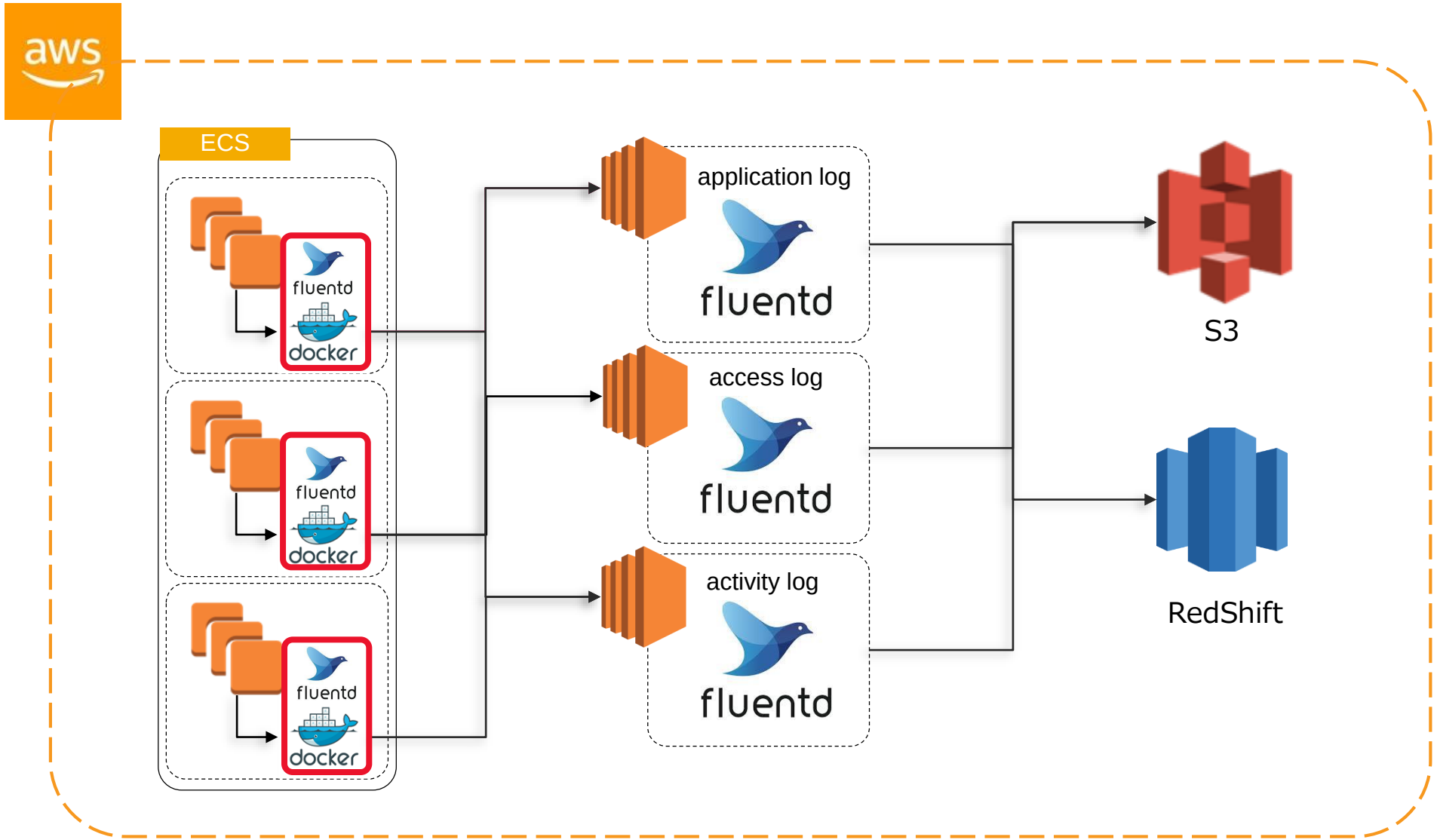
どうしたか？

コンテナホストに

fluentdの中継サーバーを立てる

FiNC

fluentdの中継



FiNC

問題2 （ラスト）

push通知重複問題

FiNC

ある日・・・

１人に同じプッシュ通知が重複して届く

原因

非同期処理ワーカーが処理中にコンテナが停止してしまい、最初から処理を再開

- ECSタスクの停止動作

$ kill -SIGTERM (kill -15)

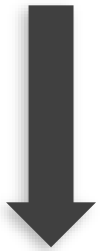
$ kill -SIGKILL (kill -9)

FiNC

どうしたか？

$ kill -USR1

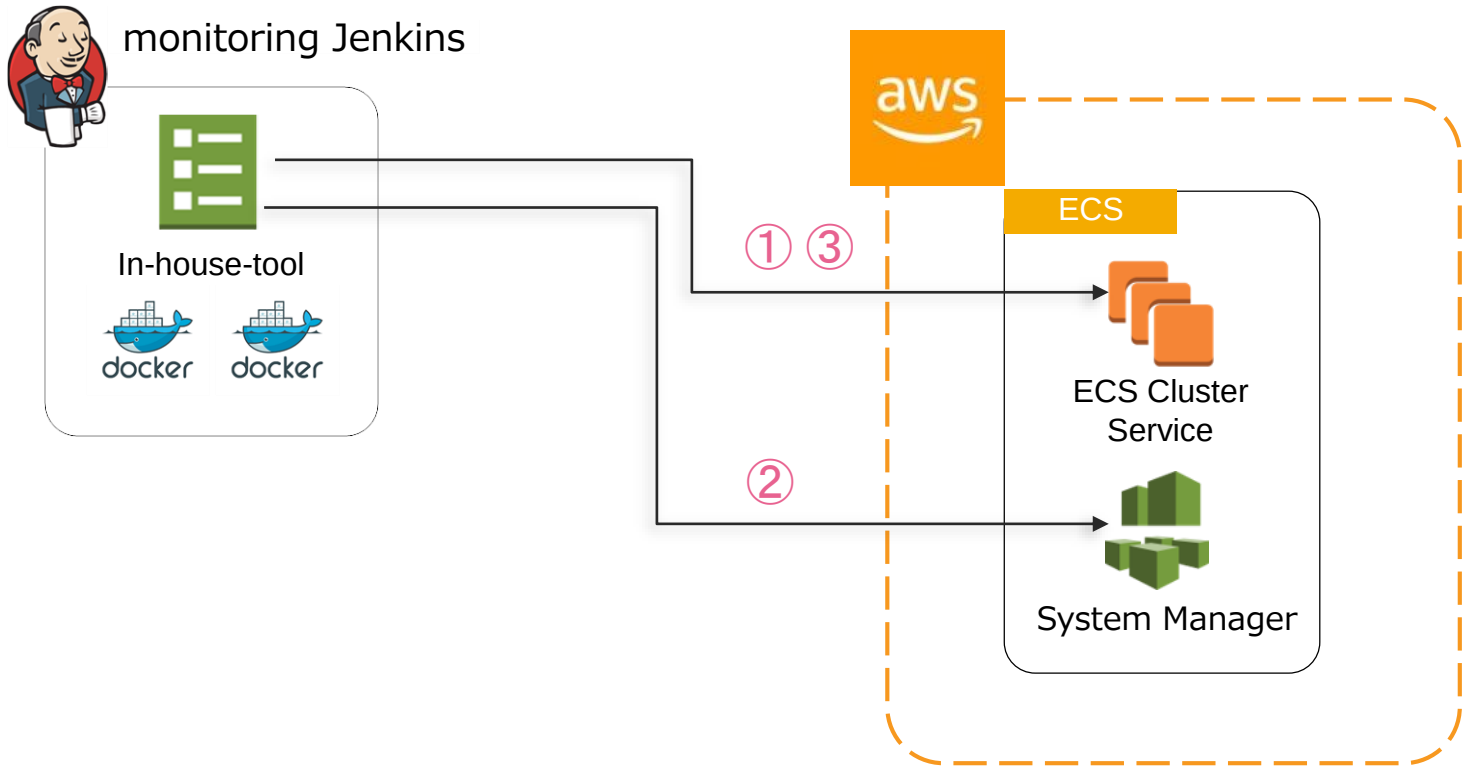
停止シグナルを送ってから、停止するまで待つ



コンテナ監視を社内で構築

FiNC

ワーカー監視方法



1. 稼働中のタスクの最大メモリとワーカー数を確認
2. SystemManagerにメモリとワーカー数を問い合わせ
3. 最大メモリ値を超えたタスクに対してユーザシグナルを送りタスクを終了

FiNC

FiNC

ライフログがたまるほど、あなたにフィットする
パーソナルトレーナーAIアプリ

採用

エンジニア採用してます

- SiteReliabilityエンジニア
- サーバサイドエンジニア（Rails）
- iOSエンジニア
- Androidエンジニア

https://www.wantedly.com/companies/finc/projects

FiNC

エンジニアブログ

- Redshiftディスク使用量削減大作戦
- 細部カイゼン合宿が思ったより有意義だった件
- マイクロサービス指向 Rails API開発ガイド

・・・この後、AWS Summit Tokyo

の話しもアップします

https://medium.com/finc-engineering

FiNC

Thank you