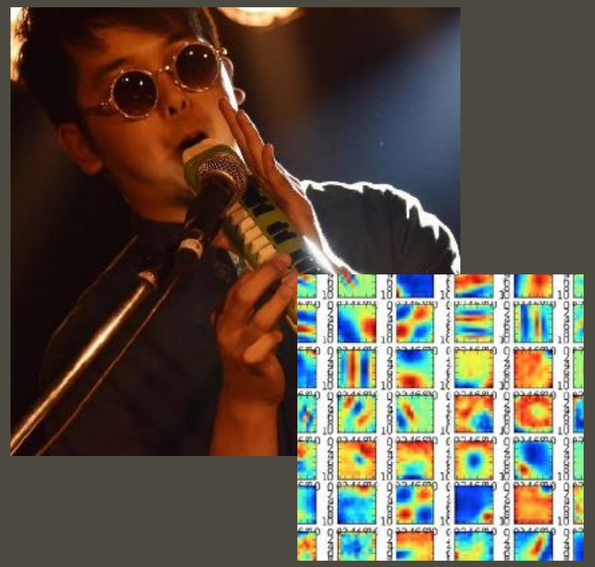
クックパッドの機械学習を支える基盤のつくりかた

クックパッド株式会社
染谷 悠一郎　星 北斗

2017/06/01 AWS Summit Tokyo



github.com/ayemos

twitter.com/ayemos_y

www.ayemos.me

自己紹介

- 染谷 悠一郎 [Yuichiro Someya]
- クックパッド株式会社
  研究開発部 エンジニア

  # 2016年新卒入社

aws CERTIFIED

Solutions Architect - Associate

毎日の料理を楽しみに cookpad 265万レシピ

料理名・食材名 × 目的・用途 レシピ検索

たけのこ ランチ 筍ご飯 うどん そうめん

MYフォルダ

レシピ関連サービス

- みんなのレシピ
- プロのレシピ
- 献立
- 料理動画
- 話題のキッチン

プレミアムサービス

- 人気順検索
- レシピランキング
- 殿堂入りレシピ
- 専門家厳選レシピ
- プレミアム献立

もっと見る

生活関連サービス

- 料理教室
- ベビー

4月24日のおすすめ

5分de しっとり鶏そぼろ丼 coffeedge

クックパッドからのお知らせ

1歳未満の乳児に蜂蜜を与えないでください

ニュース

にんじんの「おかかマヨ和え」

家にあるカツオ節とマヨネーズでササッと作れるので、あ

間違いなしの
鉄板レシピをご紹介！

レシピ数 260万品 以上

月次利用者数 6,000万人 以上

（2017年3月末時点）


cookpad
Ingredient name, dish...
Search Recipes
broccoli salad teriyaki bbq pulled pork baked beans latest
Create Recipe
3 ing. pancakes
Quick & Simple Frittata
Pumpkin soup
Vietnamese Grilled Pork (Thit Nuong)
Honey-Glazed Carrots
Coconut Curry Shrimp
Spinach & Artichoke dip pasta with grilled chicken
Dorito baked pie
كوكباد
إبحث عن وصفة
أضف وصفة
معروك هش وطري وقطني بحشوة النوتيلا و الزبيب
هاد فطورنا اليوم
مكرونة الفوتشيني
مكرونة بالجبن والجمبري
عجينة فطائر
حلا سهل
فتوش
اصابع البطاطا بالجبنة (طبق جانبي)
بيتزا الخضار
cookpad
ค้นหาสูตรอาหาร
เขียนสูตรอาหารที่นี่


cookpad
Ingredient name, dish...
Search Recipes
broccoli salad teriyaki bbq pulled pork baked beans latest
Create Recipe
كوكباد
ابحث عن وصفة
أضف وصفة
cookpad
ค้นหาสูตรอาหาร
เขียนสูตรอาหารที่นี่
17言語に対応
62カ国に展開
（2017年3月末時点）

クックパッドとAWS

- 2011年に DC から完全移行
- ap-northeast-1 および us-east-1 を主に利用
  # グローバルサービスは主に US で稼働

Amazon EC2 インスタンス

700 ~ 1,400台程度

Amazon S3 オブジェクト

1億以上

ピーク時同時リクエスト数

15,000 req/s 以上

利用しているサービス

- 今回のお話に関係するサービス

  # Amazon EC2, Amazon VPC, AWS Lambda
  （以降 EC2, VPC, Lambda と表記）

  # Amazon SQS, Amazon SNS（以降 SQS, SNS と表記）

  # Amazon S3, Redshift（以降 S3, Redshift と表記）

  # AWS Config, Amazon CloudWatch, AWS CloudTrail, AWS IAM
  （以降 Config, CloudWatch, CloudTrail, IAM と表記）

＝ テーマ ＝

クックパッドの機械学習基盤

クックパッドの研究開発部について

- 2016年7月に発足
  - 研究開発チーム(2015年6月~)、研究開発室 (2016年4月~) を経て
    現在は正社員5名 + アルバイト／インターン4名の組織に
- 機械学習を中心に、食文化系の研究における実績も
  - クックパッド江戸ご飯
  - 第31回 人工知能学会 全国大会
- **料理きろく**の公開 (2016年12月)

料理きろく

スマートフォンの写真の内、

料理写真を自動的に収集/記録

畳み込みニューラルネットワーク

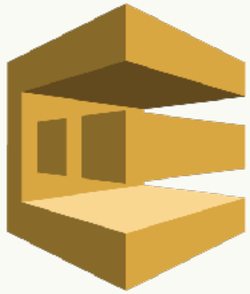
による料理画像の自動認識

料理きろくのアーキテクチャ

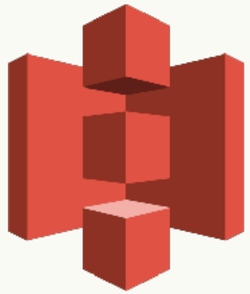
- 画像のアップロード、判定処理には時間がかかる（~数百ms）
  - APIサーバーと統合して同期処理するのは非現実的
  - 非同期な料理／非料理の判定処理が必要
- 必要な計算機環境がアプリケーションサーバーと大きく異なる
  - Ruby ⇔ Python , Rails ⇔ Chainer, CPU ⇔ GPU
  - 判定処理とアプリケーションを疎結合にするのがベター

料理きろくのアーキテクチャ

S3, SQSを介した非同期で疎結合な写真判定フローの実装



SQS



S3

料理きろくのアーキテクチャ

- APIサーバーでS3 Pre-Signed URLを生成
  - # 生成されたURLを利用してクライアント(iOS, Andoird)から判定用に縮小された写真をS3に直接アップロード

料理きろくのアーキテクチャ

- APIサーバーでS3 Pre-Signed URLを生成
  - # 生成されたURLを利用してクライアント(iOS, Andoird)から判定用に縮小された写真をS3に直接アップロード

料理きろくのアーキテクチャ

- S3 Event Notificationで写真のアップロードイベントをSQSにEnqueue

1. Pre-Signed URLの発行

2. 判定用画像のアップロード

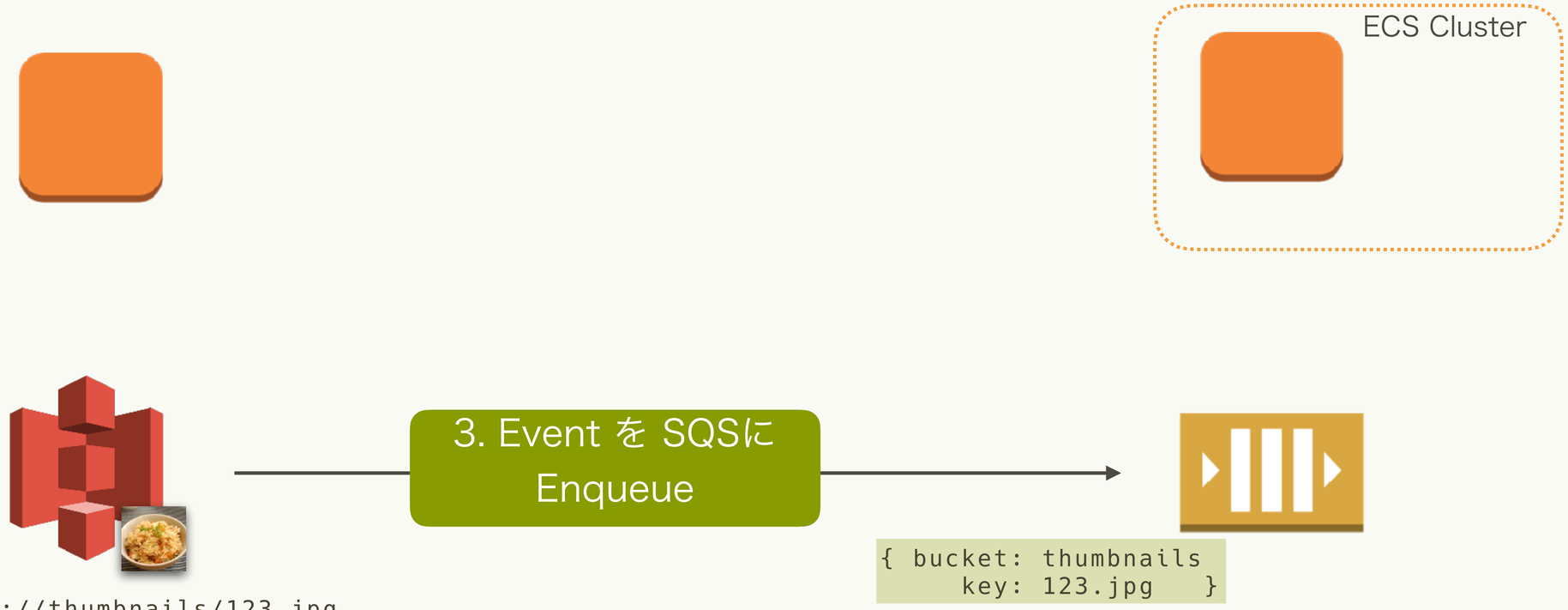
s3://thumbnails/123.jpg

3. Event を SQSにEnqueue

```
{ bucket: thumbnails
     key: 123.jpg   }
```

料理きろくのアーキテクチャ

- 判定処理workerがSQSからメッセージをDequeueして処理
  - # メッセージに含まれる写真のS3上のkey情報を利用して写真をダウンロード
  - # keyに含まれる写真のidを利用してAPI越しに判定結果を通達

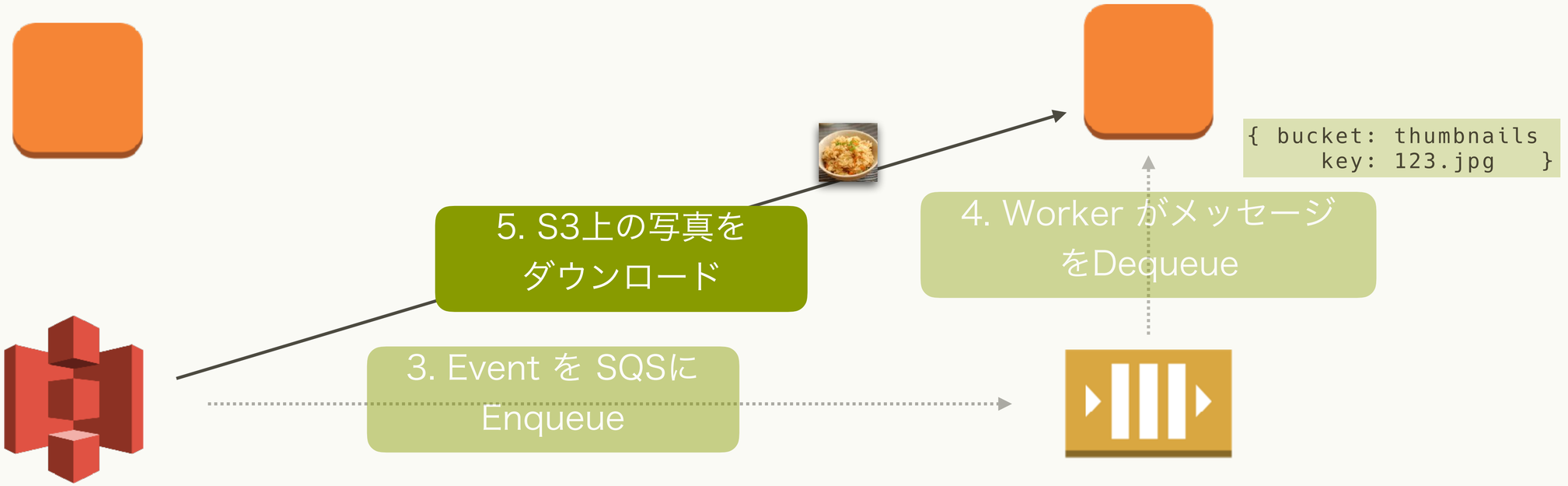


料理きろくのアーキテクチャ

- 判定処理workerがSQSからメッセージをDequeueして処理
  # メッセージに含まれる写真のS3上のkey情報を利用して写真をダウンロード
  # keyに含まれる写真のidを利用してAPI越しに判定結果を伝える

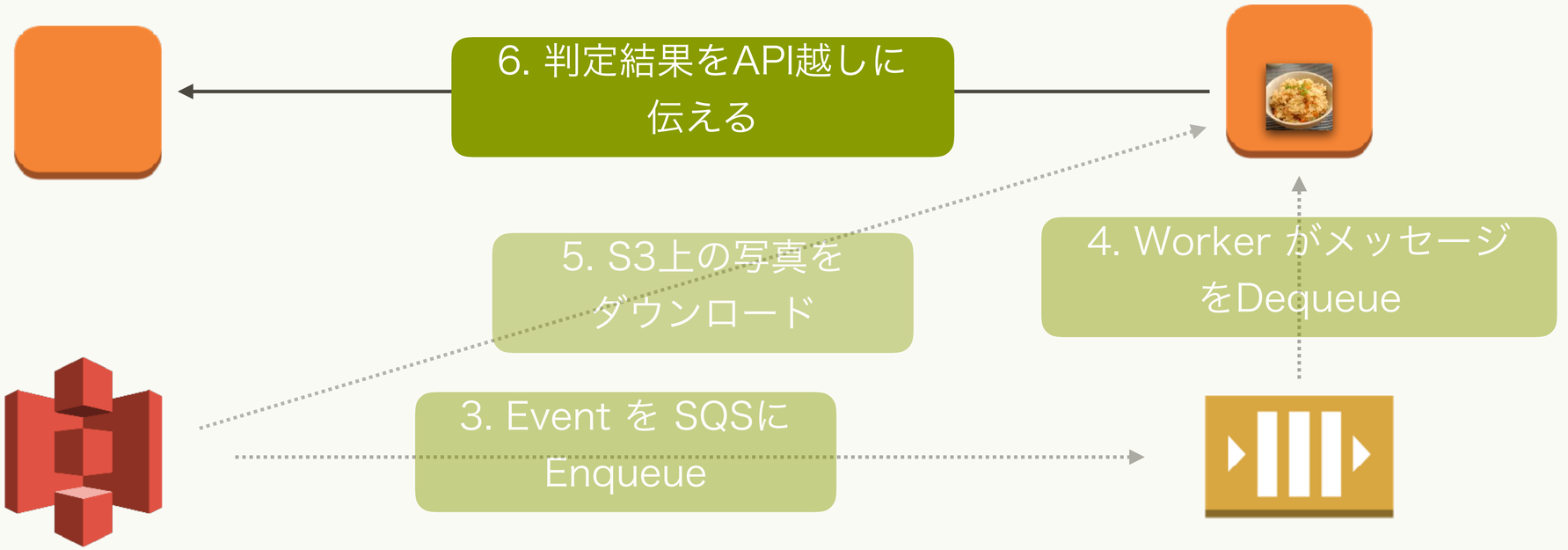
料理きろくのアーキテクチャ

- 判定処理workerがSQSからメッセージをDequeueして処理
  # メッセージに含まれる写真のS3上のkey情報を利用して写真をダウンロード
  # keyに含まれる写真のidを利用してAPI越しに判定結果を伝える



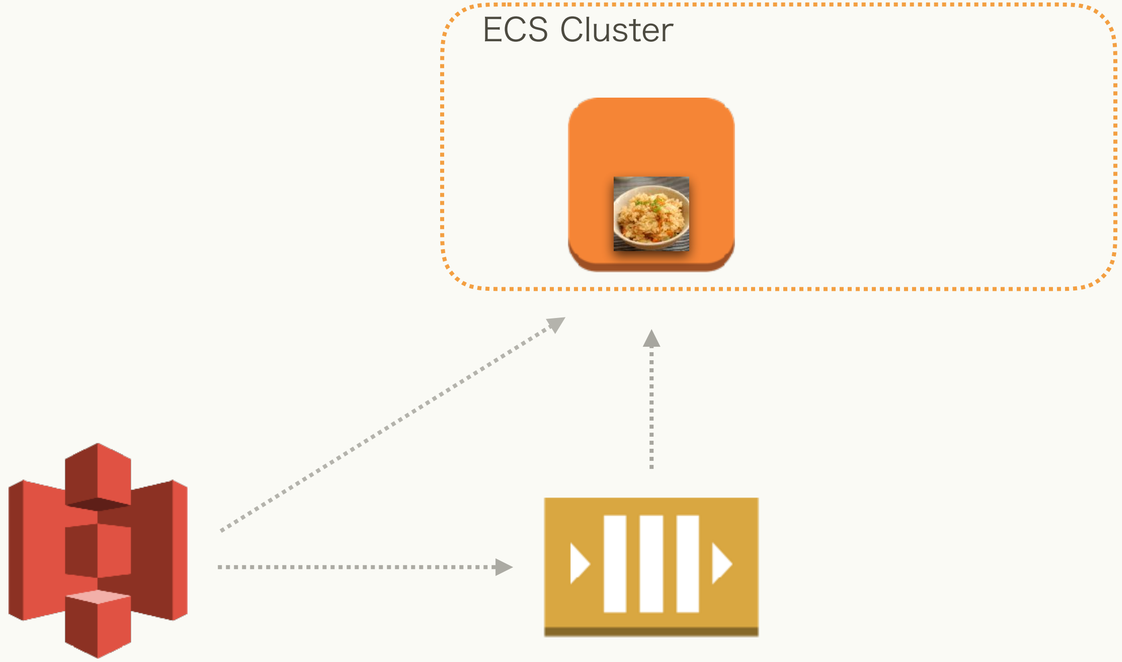
料理きろくのアーキテクチャ

- 判定処理workerがSQSからメッセージをDequeueして処理
  - # メッセージに含まれる写真のS3上のkey情報を利用して写真をダウンロード
  - # keyに含まれる写真のidを利用してAPI越しに判定結果を伝える



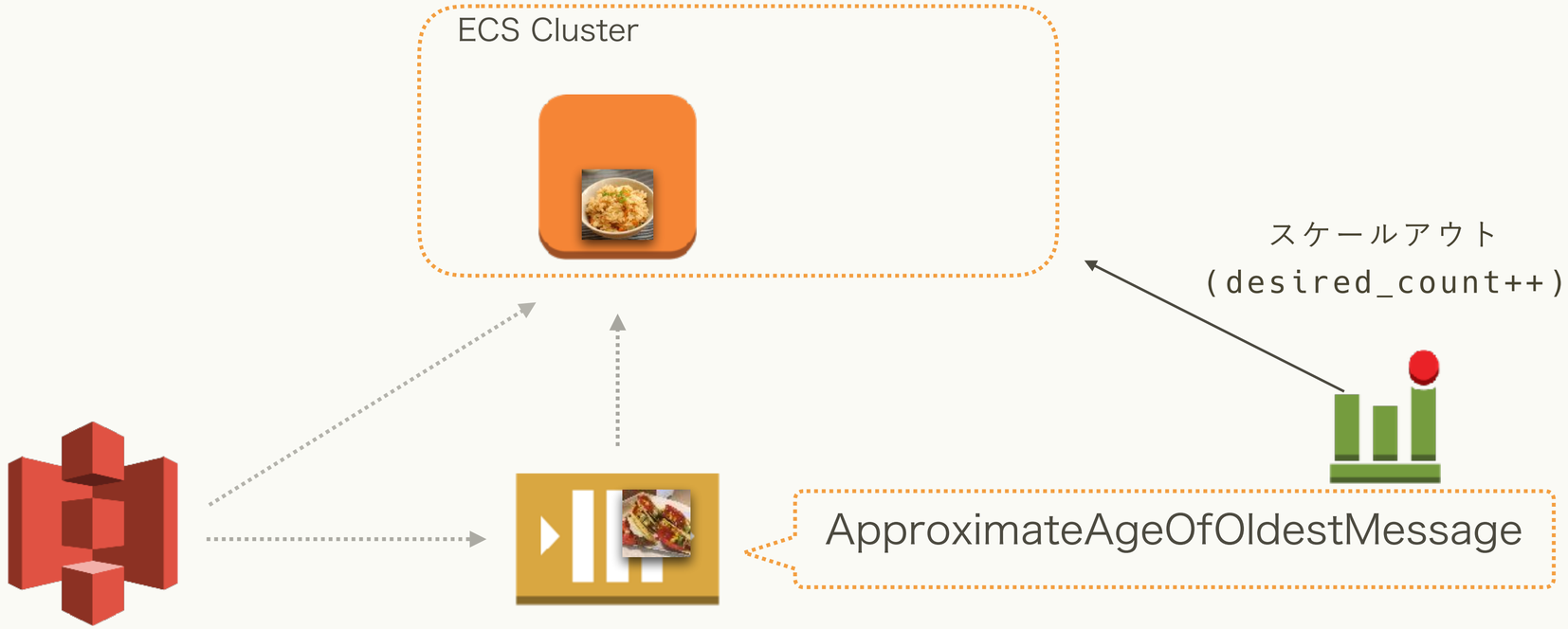
料理きろくのアーキテクチャ

- SQS の CloudWatch メトリクスに基づいた Alarm で worker をオートスケール
  - アプリケーションの性能要求を満たすように
    （例えば、新しくアップロードされた写真は1時間以内に判定されて欲しい）
  - ApproximateAgeOfOldestMessage が利用できる



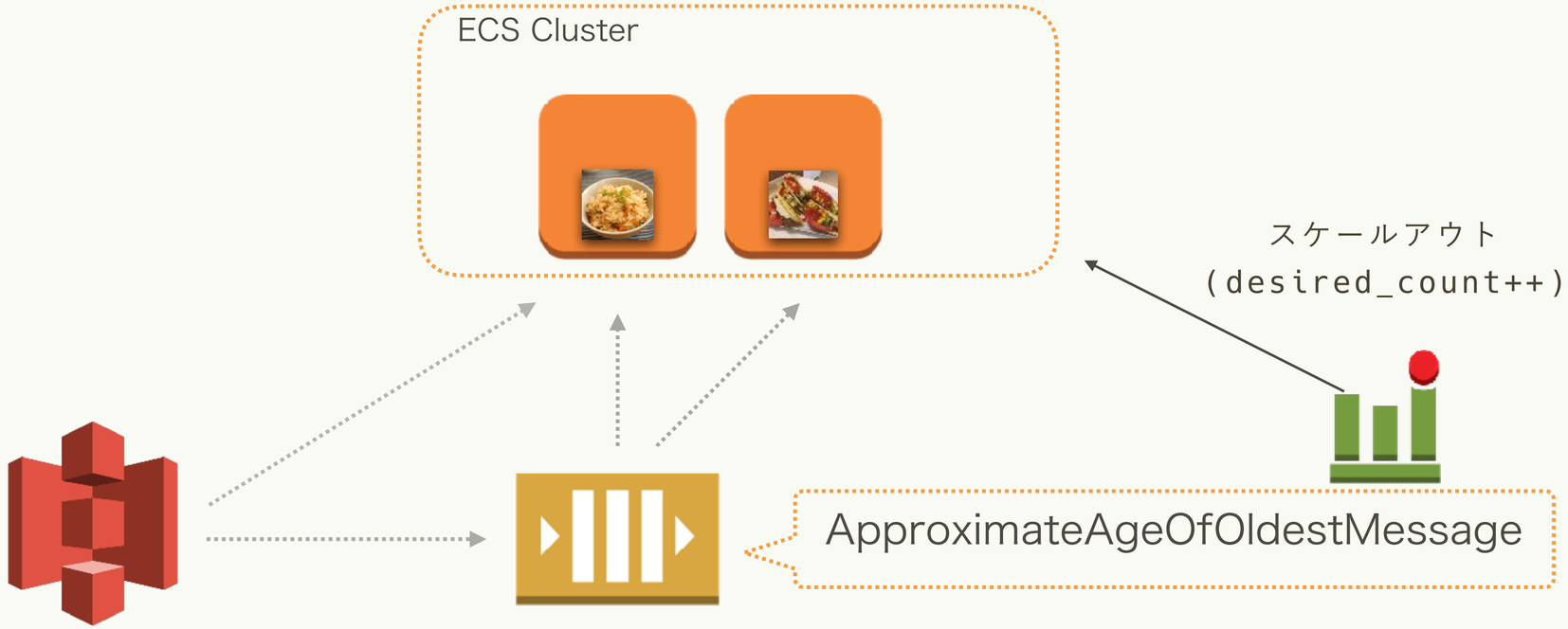
料理きろくのアーキテクチャ

- SQS の CloudWatch メトリクスに基づいた Alarm で worker をオートスケール
  - # アプリケーションの性能要求を満たすように
    （例えば、新しくアップロードされた写真は1時間以内に判定されて欲しい）
  - # ApproximateAgeOfOldestMessage が利用できる



料理きろくのアーキテクチャ

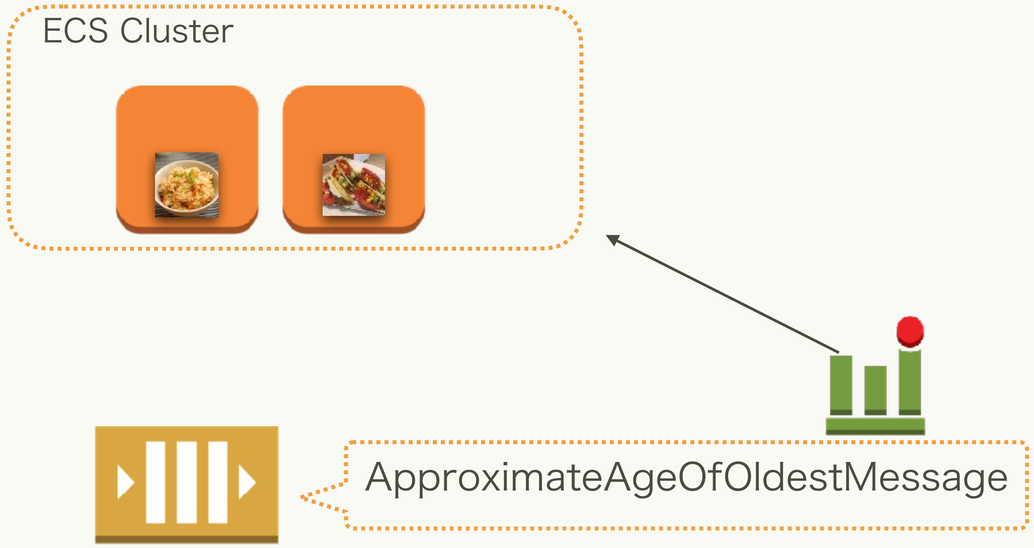
- SQS の CloudWatch メトリクスに基づいた Alarm で worker をオートスケール
  - # アプリケーションの性能要求を満たすように
    （例えば、新しくアップロードされた写真は1時間以内に判定されて欲しい）
  - # ApproximateAgeOfOldestMessage が利用できる



ECS上での写真判定

- eagletmt/hako (*1) を利用
  - # yaml形式のファイルで判定処理worker を定義、CLIコマンドで立ち上げ
    - # オートスケールの設定もhakoで行える

```
scheduler:
  cluster: hako-production-g2
  desired_count: 1
  autoscaling:
    min_capacity: 4
    max_capacity: 10
    policies:
      - alarms: [cookpadnet-worker-busy]
        cooldown: 300
        adjustment_type: ChangeInCapacity
        scaling_adjustment: 1
app:
  image: cookpadnet-worker
  cpu: 128
  memory: 1024
  env:
    AWS_REGION: ap-northeast-1
    COOKPADNET_ENV: production
   ...
```



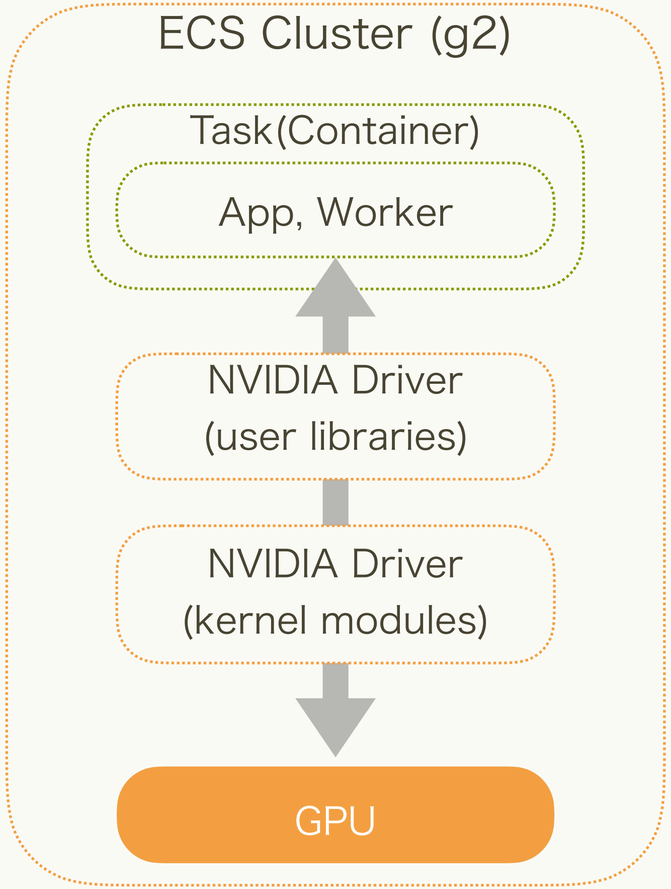
1: https://github.com/eagletmt/hako

ECS上での写真判定

- 写真判定にGPUを使用している
  - # 判定処理にGPUアクセラレーションが利用できる
  - # 同価格帯のCPUインスタンスと比べて4~5倍の性能差
- g2インスタンスで写真判定用のECS Clusterを構成

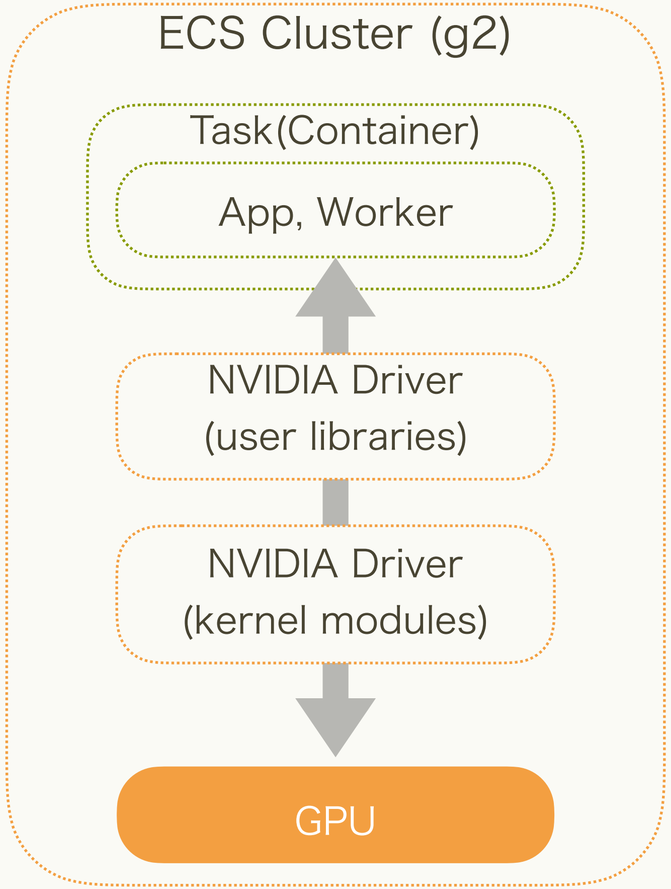
ECS上での写真判定

- ContainerからGPUを操作する為に必要な設定
  - # NVIDIA Driverのインストール
    -> Clusterにインストールし、Taskからはvolumeで見る
  - # 適切なLinux Capabilityの付与
    -> privileged オプションを利用



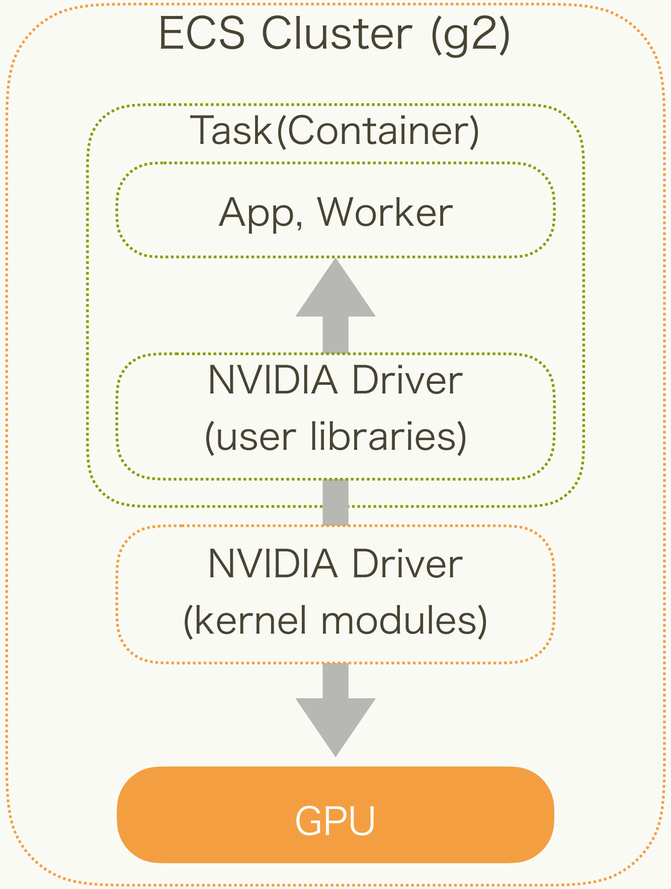
ECS上での写真判定

- privilegedなTask(のrootユーザー)は、ECSクラスタに対してrootユーザーと同様な権限をもつ
  - # クラスタに影響を与えるような操作が可能 (mlock, kill, utime, …)
  - # ECSで `--device` オプションが使えると…
- Taskで非rootなユーザーを利用する (DockerFile内で `USER worker` 等)
  - # Cluster上のNVIDIA Driver (user libraries)が見えなくなる



ECS上での写真判定

- Clusterと同じバージョンのuser librariesをContainerにインストール
  - # ContainerがClusterにバージョン依存する（別のClusterでは動かないかもしれない）



運用実績（2017/05/28現在）

100,000 人以上
のユーザーが利用

4,200,000 枚以上
の料理写真

- 画像認識モデルを一度アップデート
- アップデートも乗り越え、大きな問題もなく運用できている

機械学習基盤について

星 北斗（ほし ほくと）/ @kani_b

- クックパッド株式会社
  インフラストラクチャー部 部長
  - # 2013年新卒入社
- SRE, セキュリティエンジニア
  - # 設計構築運用, IDS, WAF, 脆弱性診断, ISMS, etc
- AWS 認定 SA, DevOps エンジニア (Professional)

機械学習基盤とは

- 研究開発部のエンジニアが自由に手を動かせる場所
  - 本番環境とは切り離されている
- 自由度が高く本番の運用に左右されない

研究開発部設立当初

- 必要なインスタンス類は全てインフラ部が準本番環境に用意
  - # 依頼を受け、それをもとに必要なリソースを準備する
- 本番環境と同様の運用
  - # コードをベースにしたプロビジョニング、監視、etc

研究開発部設立当時の AWS 管理体制

- インフラ部の運用担当者のみが全権限を持つ
  - 新しいリソースの準備は基本的に
    インフラ部が行う
  - コード化されている箇所も “実行” は
    インフラ部が担当

- AWS に関係する全てのことをインフラ部に集約
  - 設計、運用、問い合わせ、etc

起きた問題

- リソースの用意が追いつかない
  - # サービスの運用も行うため優先度が下がりやすい
  - # 必要なリソースが多く、人員増加とともに加速
- 本番環境で行っていた運用に合わない
  - # 知見があまりなく、ミドルウェア設定の段階から色々手を動かす
  - # 安定したプロビジョニングより試行錯誤による結果

中央管理の限界

- 規模拡大に伴いインフラ部自身がボトルネックになることが増えた
  - 新リソースの作成, 設計などの相談, 問い合わせ など
- 我々が得意とするような運用を求められないものが出てきた (研究開発)
- 「そもそも AWS の利点を活かせていないのでは」という疑問
  - セルフマネージできることが良さの一つであるはず
- 本来やるべきことに時間を使えるようにしたい
  - “AWS の検証や管理” ではなく “ユーザにサービスを届けること” が仕事

管理方針の転換

- 権限と責任をインフラ部から各開発者へ移譲する方向にシフト
- ただ放り投げるのでなく管理すべき部分をおさえて移譲していく
  - そのための仕組みの多くを AWS が提供していくれている
- 機械学習基盤はその初期の取り組み
  - 基盤整備担当を研究開発部からアサイン

自由な環境のために必要だったもの

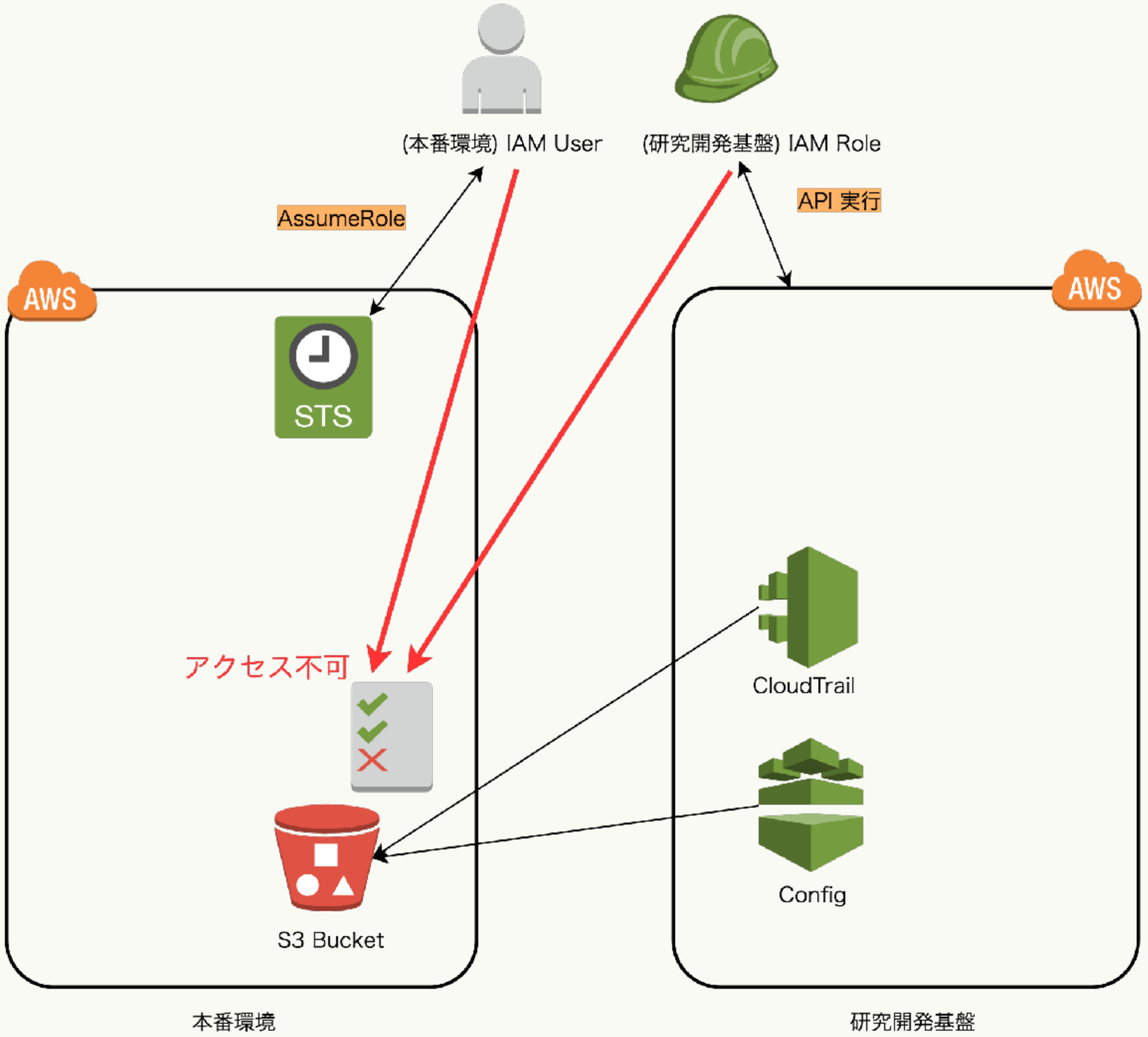
- 部のメンバーが自由に触れられる AWS 環境の用意
- AWS 環境を安全に使えるセキュリティ
- 予算管理
- AWS に関する相談やノウハウなどの移譲

アカウントの分離

- とにかく手を動かすフェーズにおいては本番環境ほど
  管理されている必要がない
- 本番環境とリソースが混じらない環境が欲しい
- 研究開発用の AWS アカウントを作成してそちらを使うように
- 支払いは本番環境のアカウントにまとめる (Consolidated Billing)

権限管理

- 本番環境で使う IAM User (個人ごと発行) から AssumeRole
  - # 必要なサービスの Admin 権限を広く付与
- CloudTrail や Config を有効にし output 先を本番アカウントの S3 へ
  - # 渡す Role からは権限を抜いておく
- 開発者に権限を渡しつつ改竄不能な形でトレースできる環境に


(本番環境) IAM User
(研究開発基盤) IAM Role
AssumeRole
API 実行
AWS
AWS
STS
アクセス不可
CloudTrail
Config
S3 Bucket
本番環境
研究開発基盤

自由な環境のために必要だったもの

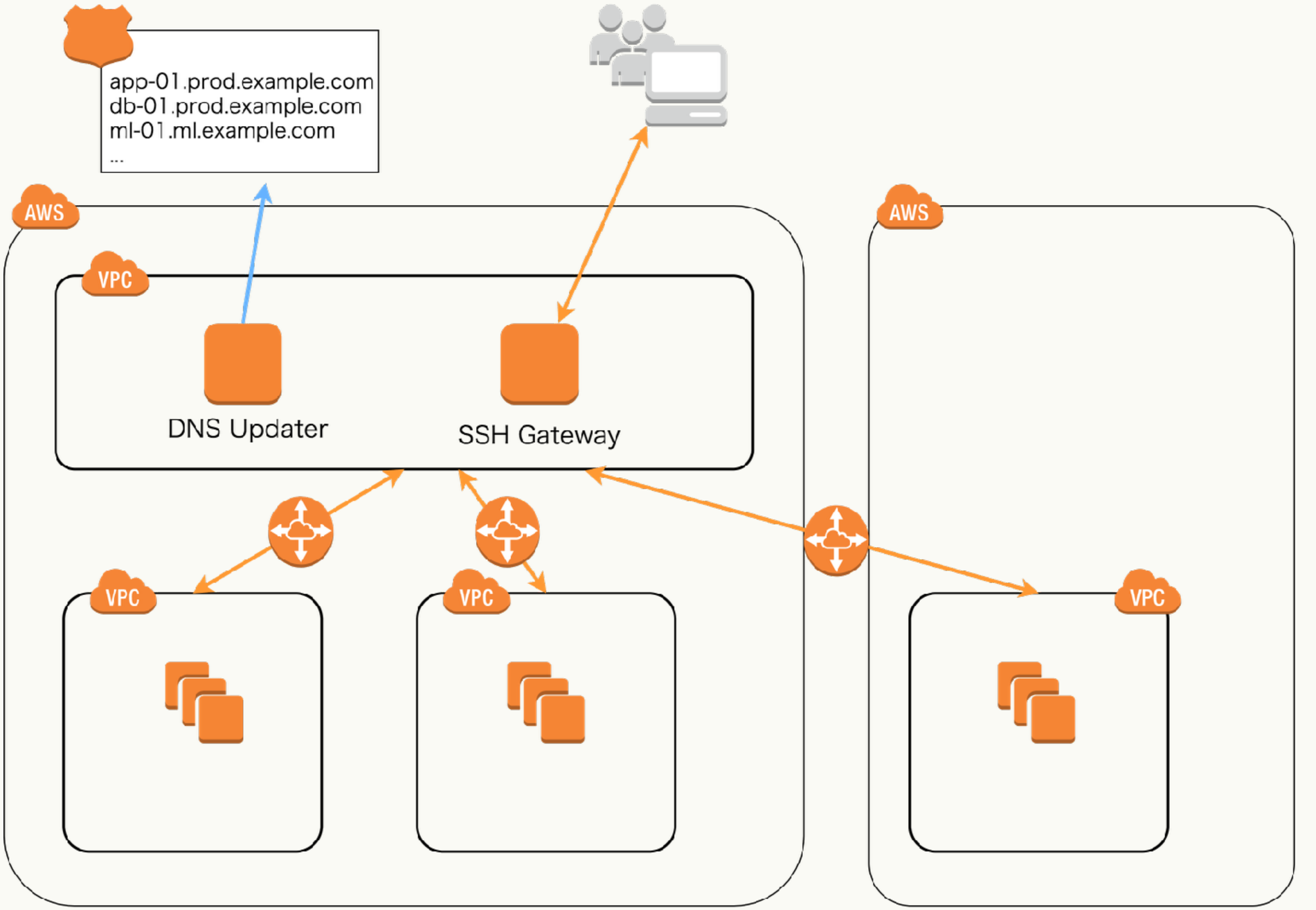
- 部のメンバーが自由に触れられる AWS 環境の用意 ✅
- AWS 環境を安全に使えるセキュリティ
- 予算管理
- AWS に関する相談やノウハウなどの移譲

ネットワークの構成

- 踏み台 SSH サーバがある VPC (本番アカウント, インフラ部管理) からVPC Peering 経由で接続できるようにする

  # 踏み台をインフラ部管理下のサーバに集約できる

- Name タグを使った正引き、逆引きを提供してアクセスしやすく


app-01.prod.example.com
db-01.prod.example.com
ml-01.ml.example.com
...
AWS
VPC
DNS Updater
SSH Gateway
VPC
VPC
AWS
VPC
本番環境 (*.prod.example.com)
研究開発基盤 (*.ml.example.com)

設定状況の監視

- AWS Config Rules でセキュリティグループのインターネット解放をチェック
  - 設定変更をトリガとして lambda function でチェック可能
  - AWS 提供の function レポジトリ: awslabs/aws-config-rules
- 細かい API コールは CloudTrail からチェック

自由な環境のために必要だったもの

- 部のメンバーが自由に触れられる AWS 環境の用意 ✅
- AWS 環境を安全に使えるセキュリティ ✅
- 予算管理
- AWS に関する相談やノウハウなどの移譲

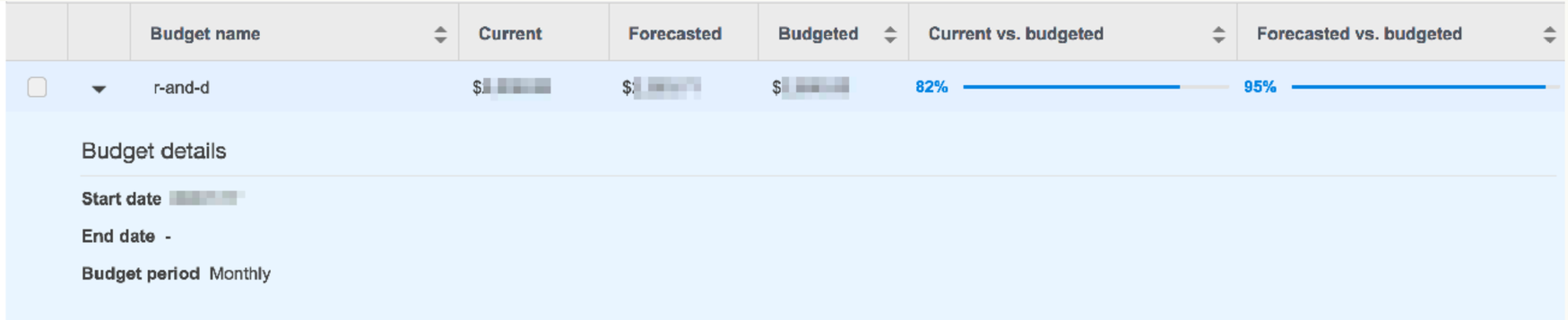
予算管理

- 細かい管理は要求されていないものの、ある程度は把握すべき

  # 落とし忘れや侵入などのアラートにもなりうる

- Billing Console で提供されている Budgets を利用

- 「今月は予算を超えそう (予測)」な場合や「予算のn割を使った (実際)」に SNS Notification を受け取り必要があれば対応



自由な環境のために必要だったもの

- 部のメンバーが自由に触れられる AWS 環境の用意 ✅
- AWS 環境を安全に使えるセキュリティ ✅
- 予算管理 ✅
- AWS に関する相談やノウハウなどの移譲

AWS 相談，ノウハウ

- AWS に関するノウハウは基本的にインフラ部が持っていた
  - # 相談もほぼインフラ部が受ける
- サポートプランはデベロッパー (2011年の移行以来)
  - # ケースの応答にも時間がかかるため基本的に自分たちで調査
  - # AWS への問い合わせ代行のようになっていたこともあった
- 本来自分たちが解かなくても良い問題に時間がかかっていた

エンタープライズサポートへの移行

- 2011年の移行当初から利用していたデベロッパーから切り替え
- 1契約で Consolidated Billing 配下の複数アカウントに適用可能
- 価格はそれなりにする（当時 $49/mo => **最低** $15,000）
- TAM アサイン, ケース対応を考慮しエンタープライズを選択
  - 時間と知識を買うことが主な目的

サポートケース対応はどうなったか

- 応答の品質が上がり、解決までの時間は確実に短くなった
  - # 「サポートに聞いてみよう」となる心理コストがとても下がった
  - # 必要ならサービスチームにエスカレーションもしてくれる
  - # Aurora など勢いのあるプロダクトを使う時は地味に効いてくる
- 各開発者が勝手にケースを作成して解決できるようになった 🚀
  - # 不具合だけでなく機能への質問、要望なども多く上がる

その他

- TAM アサインによりコンテキスト共有がしやすく
  - # ケースの取りまとめや料金レポートの作成もお願いしている
- Limit Increase の処理が爆速に
  - # コンシェルジュがアサインされて対応してくれる（最短3,4分）
- AWS 管理運用で地味にかかっていたコストがかなり下がった
  - # AWS への依存度が高い組織にとっては特に有用だと思う

自由な環境のために必要だったもの

- 部のメンバーが自由に触れられる AWS 環境の用意 ✅
- AWS 環境を安全に使えるセキュリティ ✅
- 予算管理 ✅
- AWS に関する相談やノウハウなどの移譲 ✅

開発者アカウント

- 研究開発用アカウントと同様の開発者アカウントも用意
  - 全開発者が自由に触ることのできる AWS アカウント
- ネットワークや管理系の実装は研究開発用アカウントとほぼ同じ
- 色々なサービスを自由に検証できる（会社持ちで！）
  - Kinesis, Lambda, API Gateway, X-Ray, Lex, Pinpoint などで利用されている

機械学習基盤の現状

- 意図通り活発に利用されている 🎉
  - 新しい機械学習モデルのテスト、モデル改善
  - 新サービス開発
- 開発者の手で積極的な改善が行われている（このあと紹介します）

  # インフラ部は必要な時に協力

(GPU)計算機環境

GPUインスタンス(g2, p2)の利用

- 深層学習系のタスクではGPUインスタンスを利用したい

GPUインスタンス(g2, p2)の利用

- プロビジョニングツール (Itamae等) での自動環境構築を検討
  - NVIDIA Driver(*1), CUDA(*2), cuDNN(*3)などはインストールに時間がかかる
- Packer(*4)を利用してバージョン毎にプロビジョニング済みAMIを作成／管理
  - インスタンスが立ち上がってすぐGPU環境が利用できる
  - us-east-1 ではDeep Learning AMIも試用／検討している

1: http://www.nvidia.com/Download/index.aspx?lang=en-us 2: http://www.nvidia.co.jp/object/cuda-parallel-computing-platform-jp.html
3: https://developer.nvidia.com/cudnn 4: https://www.packer.io/

インスタンスの利用方法

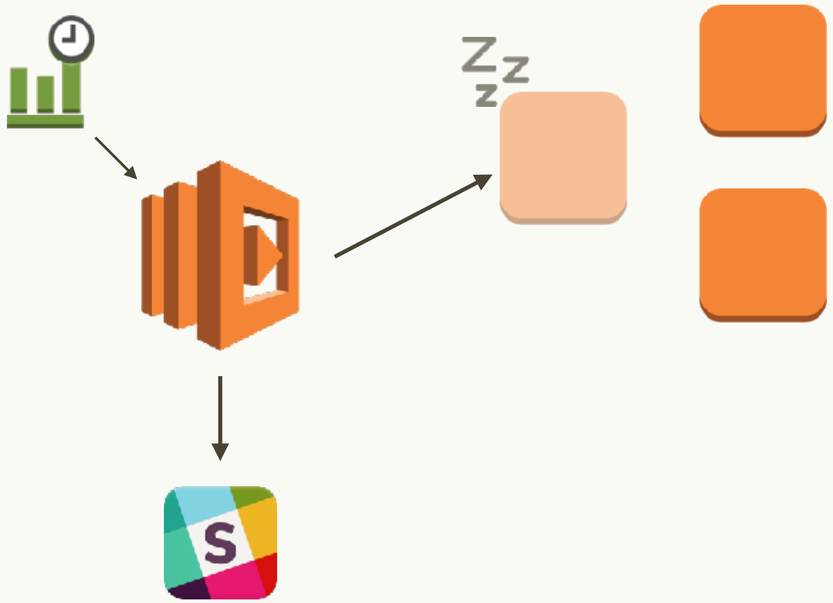
- GPU タスクは基本的に排他
  - # 他の開発者と同時には利用しづらい
- 利便性を重視し、個人毎に専有インスタンスを用意
  - # 運用上の課題の様子見をしたいというのもあった
  - # 研究開発目的では、雑に sudo 出来る独立した環境が便利

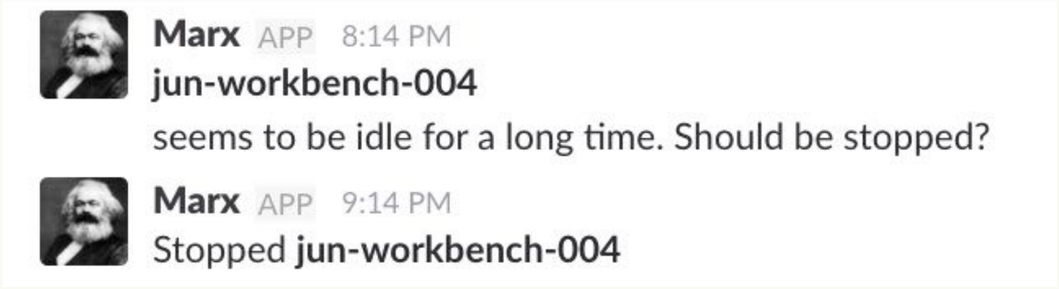
アイドルインスタンスの監視

- 使っていないインスタンスの落とし忘れなどが発生
  - # 費用面で大きな問題ではないが、近い将来組織拡大の障壁になりかねない
- 利用単価の高いインスタンスについて、アイドル状態を監視

アイドルインスタンスの監視

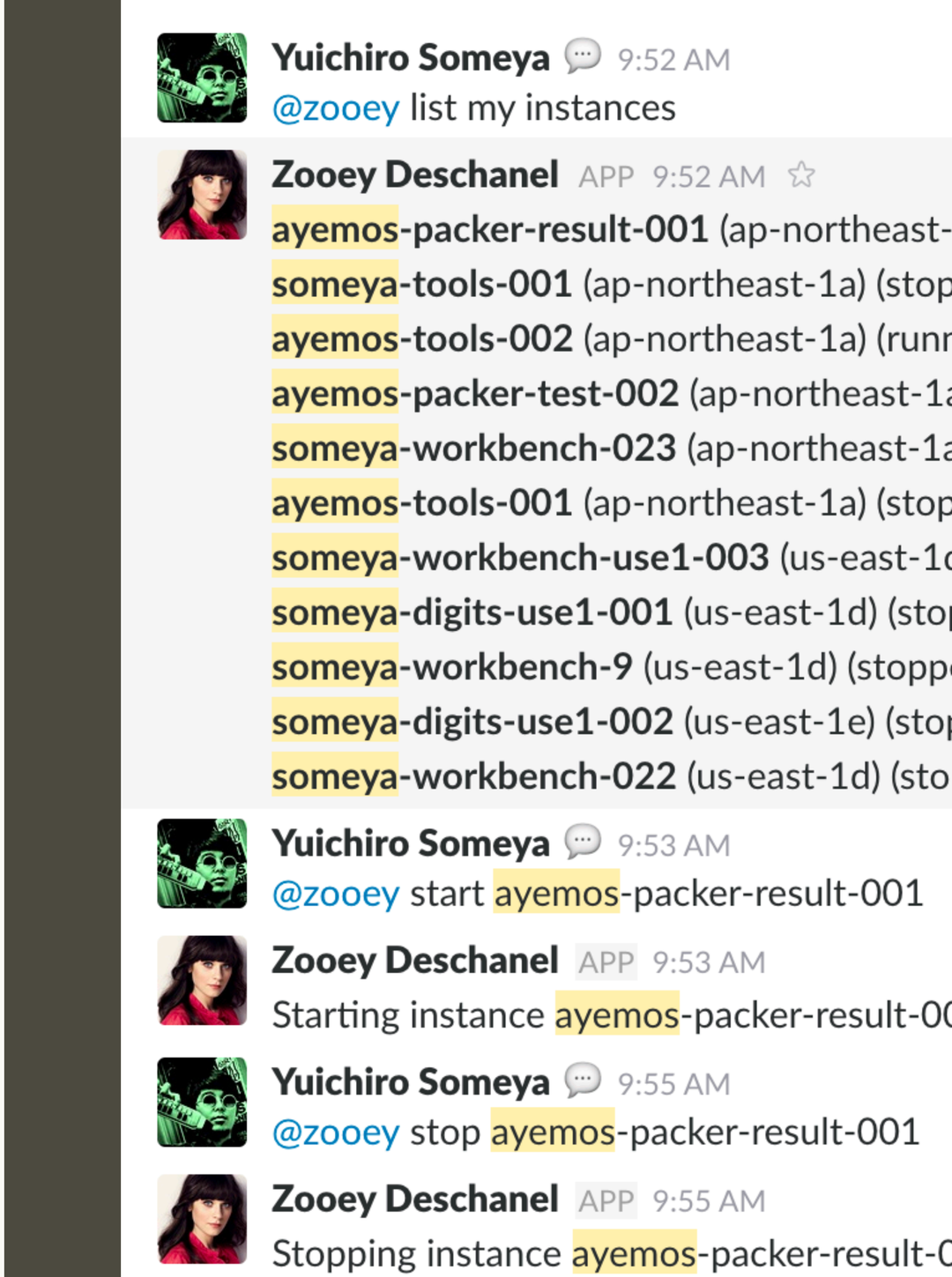
- Lambda を毎時で呼び出し、インスタンスを監視
  - 3時間アイドルなインスタンスについてはSlackに警告を投げる
  - さらに1時間アイドルだった場合はストップ





ChatBotによる
インスタンスの操作

- 落ちたインスタンスを立ち上げるのが手間になる
  # Slackから研究用インスタンスの検索、Start/Stopが出来るBotを用意



まとめ

すばやく手を動かしサービスをつくるために

セルフサービス化

全てを管理するのでなく
トレースできる仕組みづくり


CloudTrail


Config

ノウハウや相談先の移譲

エンタープライズサポートの活用

GPU計算機環境

Packerを利用した
GPU AMIの作成と運用

AMI

インスタンスの
落とし忘れ監視

Lambda

CloudWatch

インスタンス管理 ChatBot

料理きろくの公開

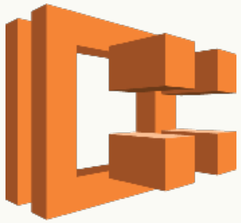
非同期で疎結合な
写真判定フロー

SQS

S3

ECS GPU クラスター

Hako によるデプロイと
オートスケール



ECS

PR

We're Hiring!

新卒採用

中途採用

第二新卒採用

アルバイト採用

https://info.cookpad.com/careers

Thank you!