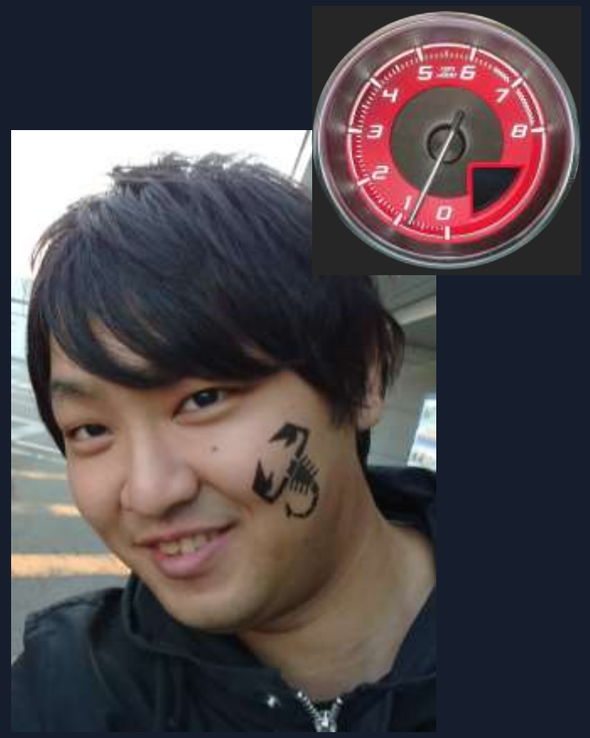
CUS-21

顧客最適な機械学習モデルを提供する対話エンジンサービスとAmazon SageMakerの活用事例

白木義彦
株式会社PKSHA Technology

三好良和
株式会社BEDORE

スピーカー



白木 義彦 (Yoshihiko Shiraki)

株式会社PKSHA Technology

Software Engineer

三好 良和 (Miyoshi Yoshikazu)

株式会社BEDORE

Engineering Manager

目次

BEDOREの紹介

BEDOREの紹介

PKSHA TECHNOLOGY

2012年 東京大学松尾研究室の卒業生で創業し、
アルゴリズムエンジニアや機械学習リサーチャーを含め、
現在グループ240名程度

機械学習技術をベースにしたエンタープライズ向け
アルゴリズムサプライヤー

動画解析、予測最適化、言語処理など
幅広いアルゴリズムソリューションを提供

2017年9月、東証マザーズに上場

トヨタ、NTTドコモ等と資本業務提携

BEDORE

PKSHA Technologyの自然言語処理部門として設立し、
100%子会社として独立

対話エンジンサービスを中心に
自然言語処理ソリューションをエンタープライズ企業様に
提供

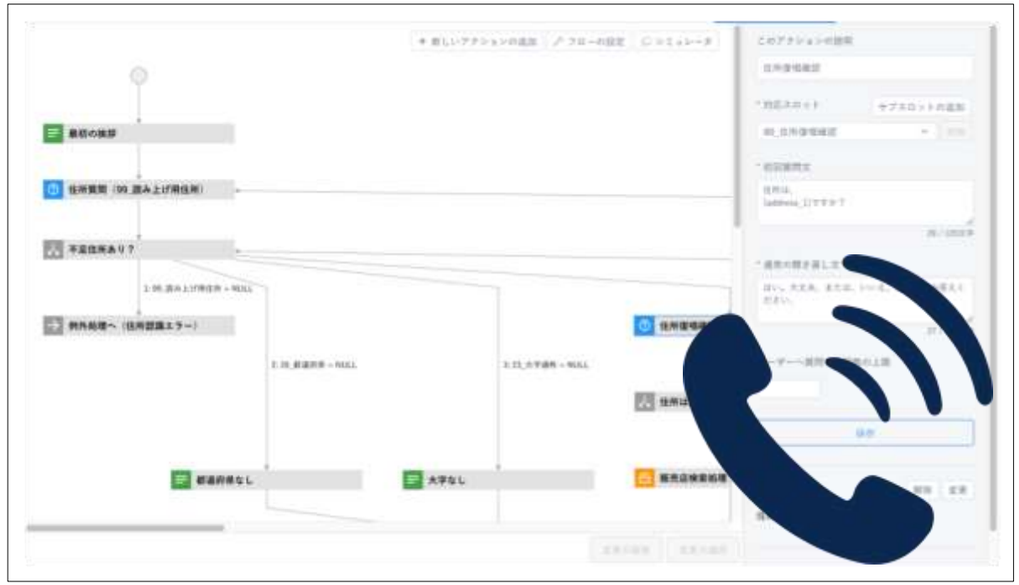
日本語の言語処理に特化した、AI型 対話ソリューション

BEDORE

Conversation
/ Workplace

BEDORE

Voice Conversation



エンタープライズ企業を中心に100社以上の導入実績
累計1億回以上の対話を提供

BEDOREにおける機械学習利用

1. 回答不足改善

- 対話エンジンをお客様が運用する上で、対話例が不足していること
  (改善すれば自動で対話する例が増える) を伝え、改善を促す機能

2. 個人情報マスク

- 対話の中で個人情報を扱うケースあり(住所、氏名、メールアドレス・・・)
- 個人情報対象となる対話の一部分にマスク[＊＊＊＊＊]をかける

3. 回答分類

- プロダクトの中心機能
- 入力された会話に対して適切に回答する

BEDOREの「回答分類」の特徴

1. クライアント毎に学習済み機械学習モデルを持つ（マルチモデル）

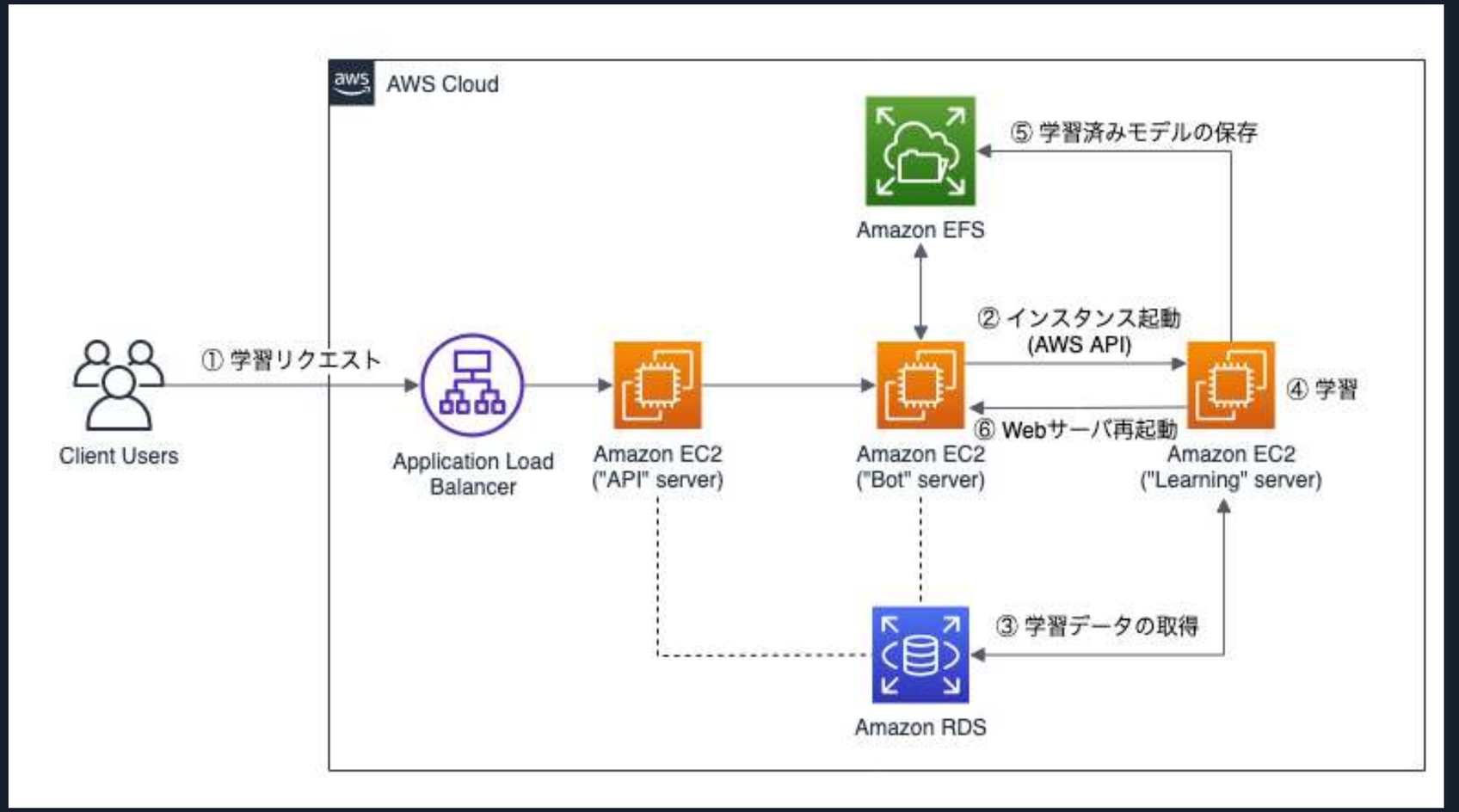
- より高い精度を求めて
- クライアント間での知識ドメインが大きく異なる

2. クライアント担当者が任意のタイミングで学習を開始できる

- 学習をする = チャットエンジンのアップデート
- 新たに自動回答できる幅を広げられる
- すぐにエンドユーザへ価値提供できるようにするため

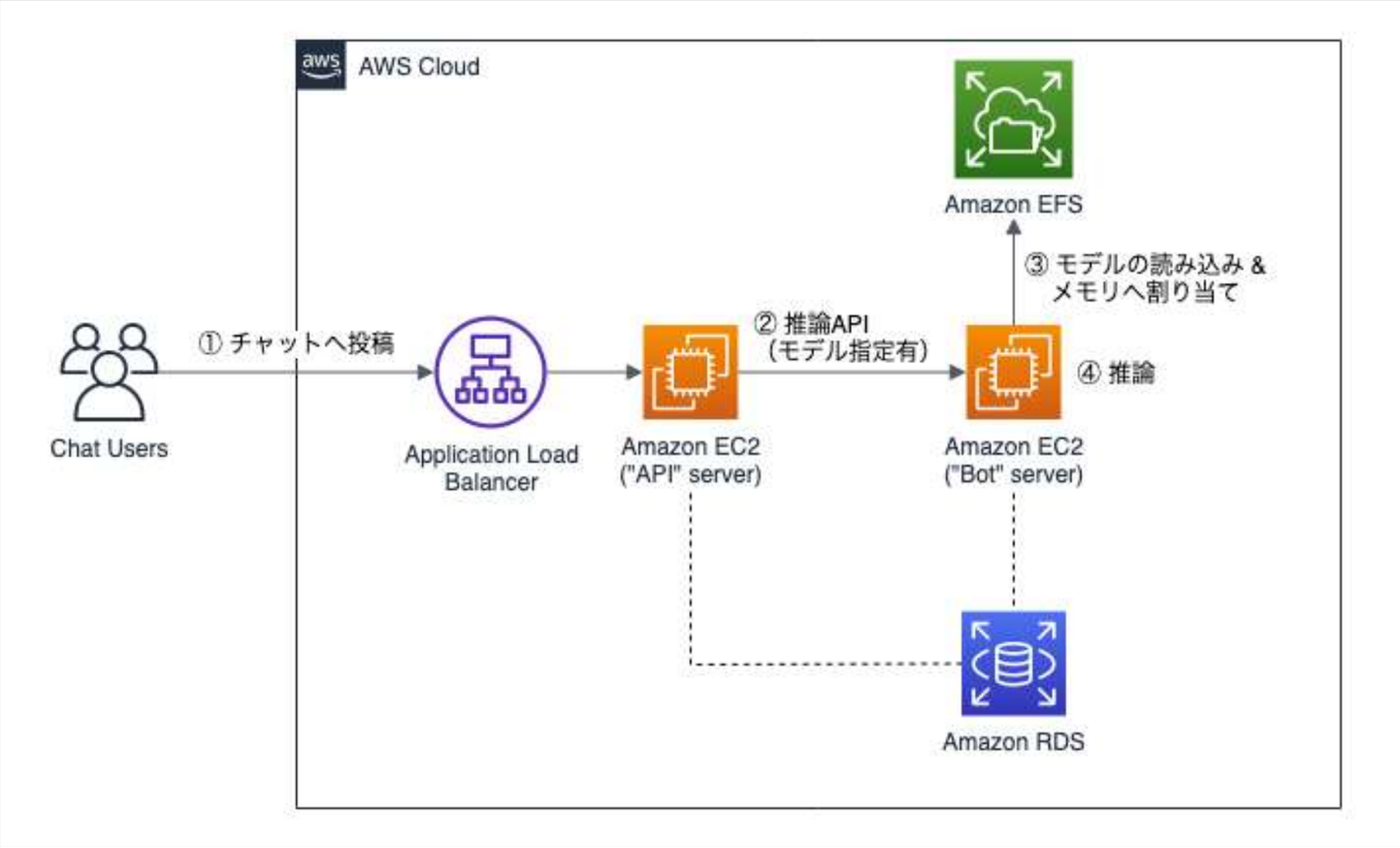
「回答分類」関連の過去のアーキテクチャ【学習】

Amazon Elastic Compute Cloud（Amazon EC2）の学習サーバを起動する



「回答分類」関連の過去のアーキテクチャ【推論】

リクエストに応じて、Amazon Elastic File System (Amazon EFS) から
モデルを読みメモリに割り当てる

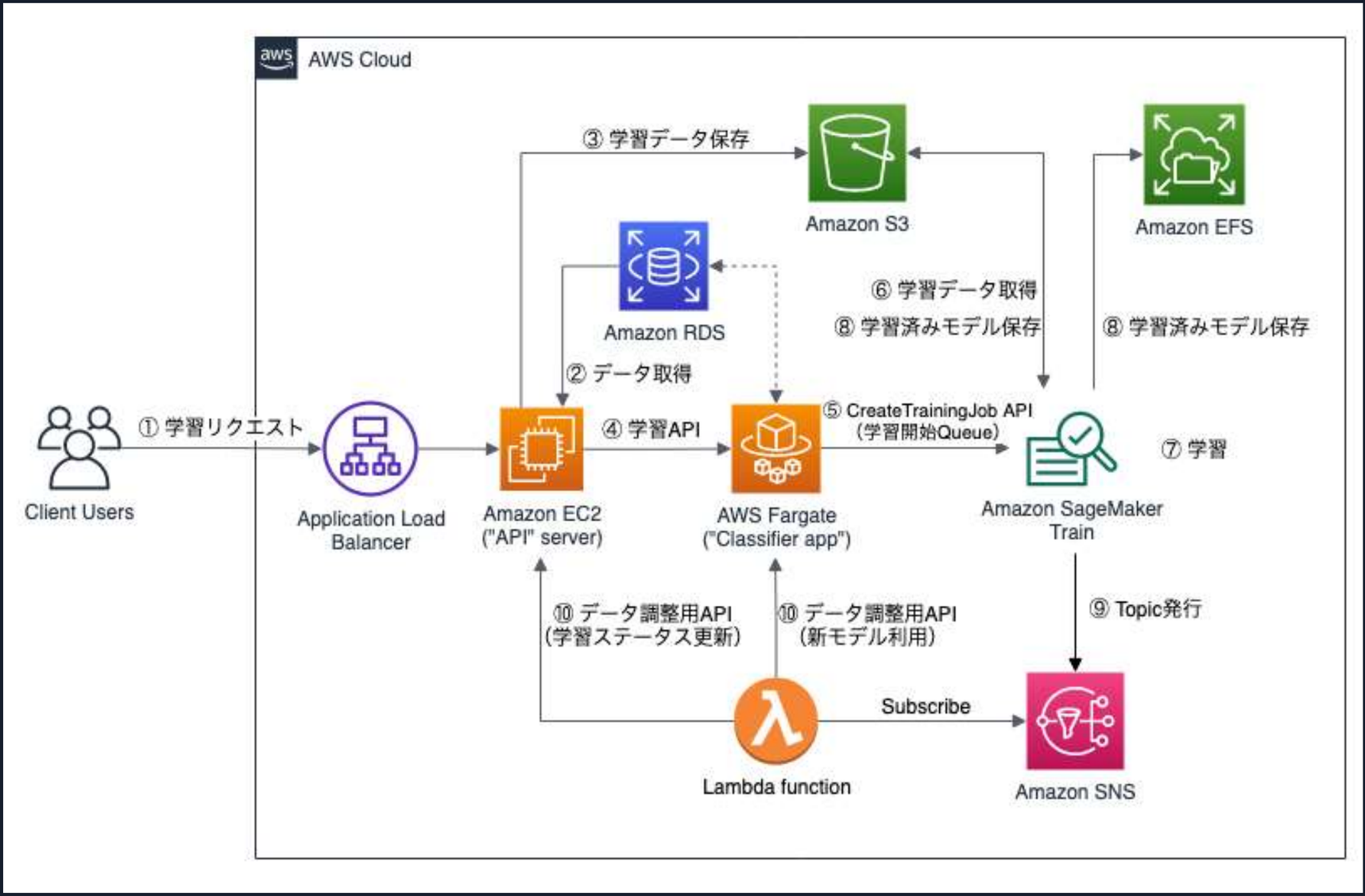


過去アーキテクチャの課題

1. 学習全体の高速化
   - Amazon EC2ではOSから起動するため、学習全体に速度改善の余地がある
2. コンテナの未利用
   - 環境差異・エンジニアの担当範囲適正化
3. Amazon EC2 の起動トラブル
   - あるバージョンのAMIで起動時のインスタンスステータスチェックに失敗する事象に遭遇
   - 最終的に原因不明、ベースイメージのバージョンを上げることで発生しなくなった
4. 「Bot」のWebサーバ再起動
   - 学習後に古いバージョンのモデルを使わないようメモリに載ったモデルをクリアする目的
   - 上記課題からインスタンスを起動することを忌避するようになったためWebサーバ再起動
   - 学習済みモデルをファイルシステムに持たせる => ステートフルなインスタンス

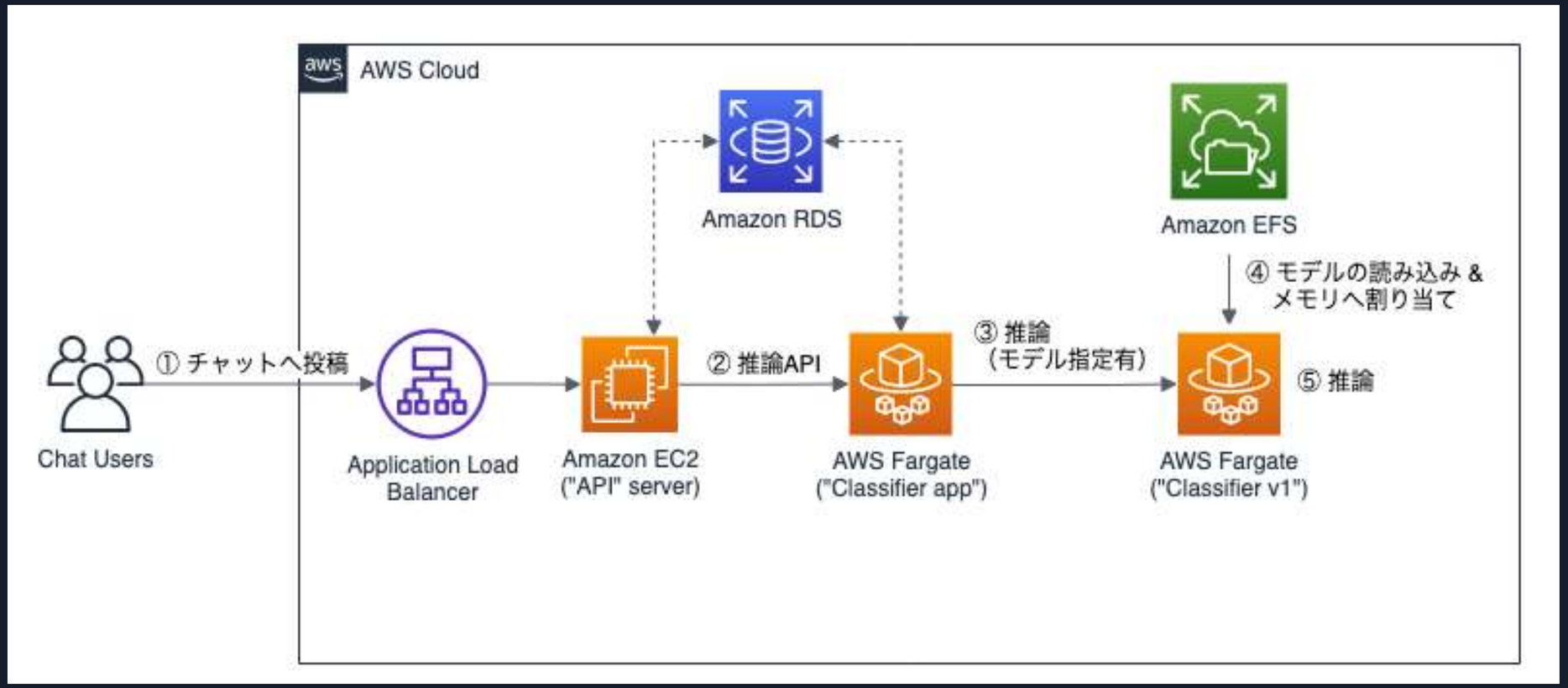
Amazon SageMaker を軸とした
リアーキテクチャを実施

Amazon SageMaker を利用した現在のアーキテクチャ【学習時】



Amazon SageMaker を利用した現在のアーキテクチャ【推論時】

- コンテナベースに切り替え
- 推論側では Amazon SageMaker の利用は見送り（後述）



学習基盤にAmazon SageMaker Trainを選択した理由

『APIリクエスト1つで学習が動くフルマネージドな学習環境が得られる』

- 学習コードを同梱したDockerイメージを用意するだけでOK
- 学習基盤としてあると嬉しい機能を用意してくれている
  1. Amazon Simple Storage Service (Amazon S3) とのSync
  2. 学習フェーズに応じた通知
  3. 追いやすい各種メトリクス
- Webサーバ + キューイングサービス を用意しなくてOK
  - キューイングサービスは特に嬉しい

⇒ 管理するリソースが減り、運用面での貢献が期待できる

Amazon SageMaker を導入した効果①

移行後のAmazon SageMaker基盤での学習は**無事故**

※Amazon SageMakerにおいてAWSがマネージしている点において

- 期間：2020年08月～
- 学習回数：約7800回

実際に運用業務を行っているインフラエンジニアからの声もGood

- 「OS・ミドルウェアのメンテナンスが必要なくなって、フルマネージドの旨味がある」
- 「かつてAMIを作成するのにPacker+Ansibleを使っていた時のデバックの苦労を思うと、これは助かる」

Amazon SageMaker を導入した効果②

学習に要する時間が 57.4% 減少した ※ BEDOREの学習サービスにおいて

- 学習開始APIのリクエスト～新モデル利用開始

  移行前：平均35分程度
  移行後：平均15分程度

- コンテナ起動がAmazon EC2起動よりも早いことが主たる要因

Amazon SageMaker を導入した効果③

クライアント毎にインスタンスサイズを最適化できる

- Amazon SageMakerのモデル学習では学習実行リクエスト時にインスタンスサイズの指定が可能
- AWS Fargate等ではタスク定義を変更する必要がある
  - クライアント毎にタスク定義を持つのは現実的ではない
- マルチモデルを扱うサービスにとってAPIで指定できるとインフラ側の変更無しに扱いやすい

Amazon SageMaker > Amazon Sagemaker API Reference

```
"ResourceConfig": {
   "InstanceCount": number,
   "InstanceType": "string",
   "VolumeKmsKeyId": "string",
   "VolumeSizeInGB": number
},
```

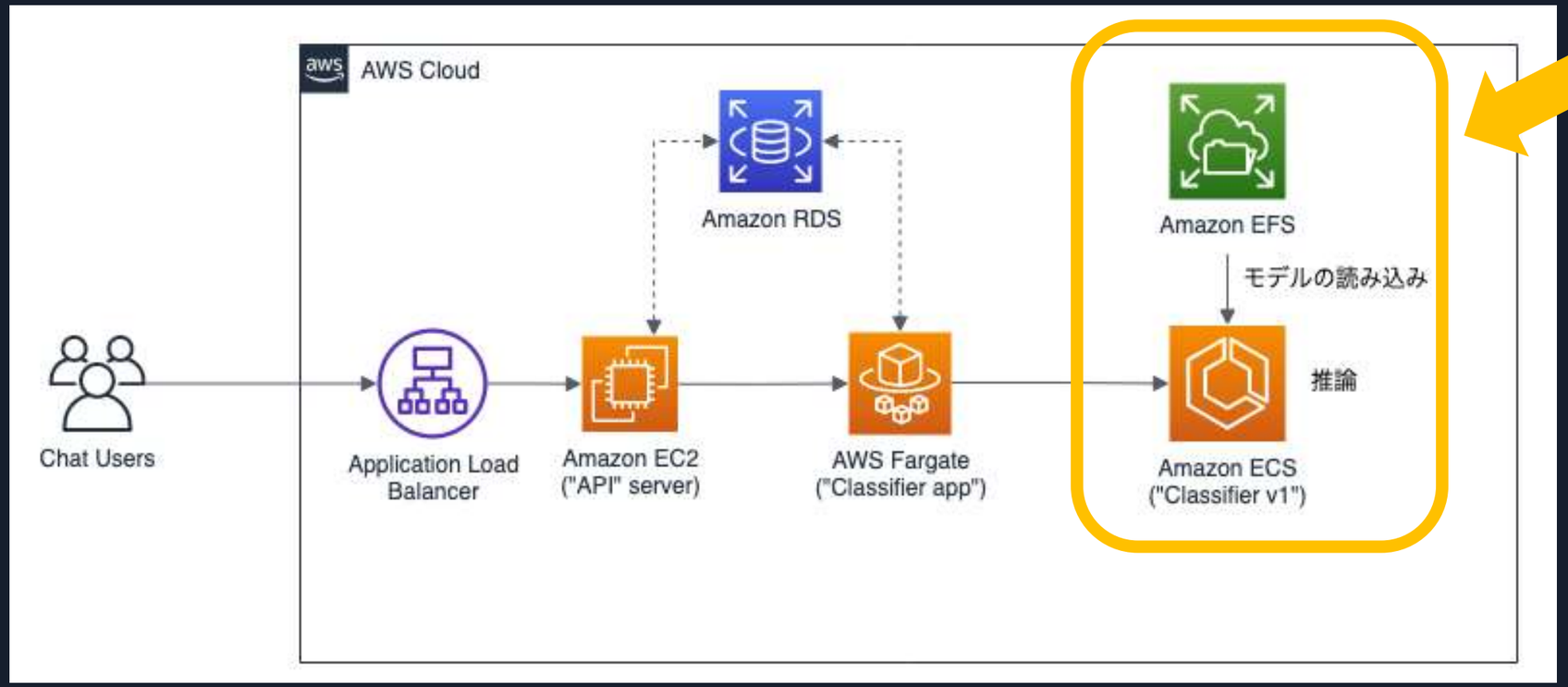
https://docs.aws.amazon.com/sagemaker/latest/APIReference/API_CreateTrainingJob.html

本構成で生じた課題

1. Amazon Elastic File System (Amazon EFS) のRead遅延
2. Amazon SageMaker Multi-Model Endpoints の未導入

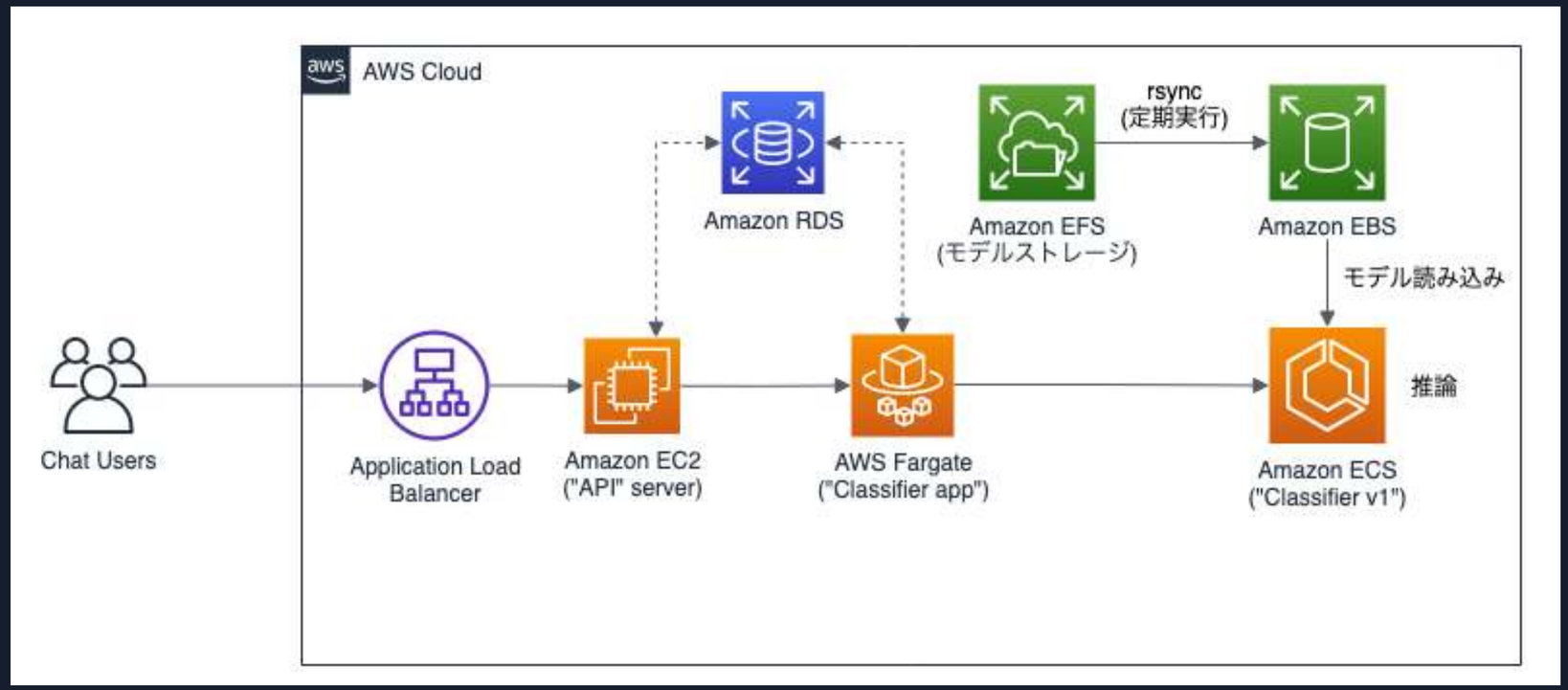
Amazon EFSのRead遅延

- 当初回答分類の推論に AWS Fargate + Amazon EFSを利用していた
- EFS上の同一ファイルへ複数クライアントからアクセスする際の競合READによりレイテンシーが低下する事象が発生



Amazon EFSのRead遅延の解決

- コンテナからローカルのファイルシステム上のモデルファイルを読むように変更
  - Amazon EFSから直接読まない
- それに伴いAWS Fargateから、Amazon EC2ベースのAmazon ECSに変更
- 定期的にAmazon EFS内のモデルファイルをAmazon EBSにsyncする

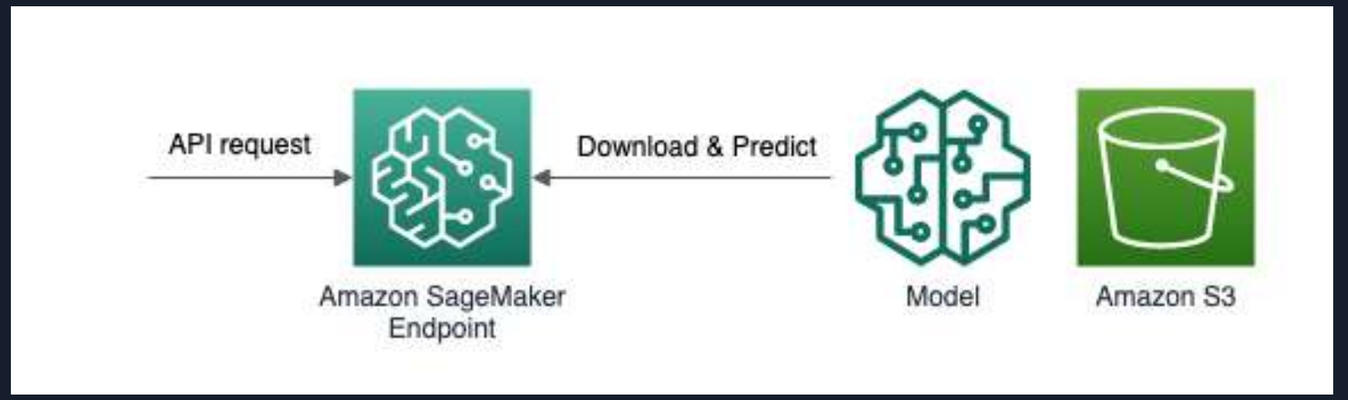


本構成で生じた課題（再掲）

1. Amazon Elastic File System (Amazon EFS) のRead遅延
2. Amazon SageMaker Multi-Model Endpoints の未導入

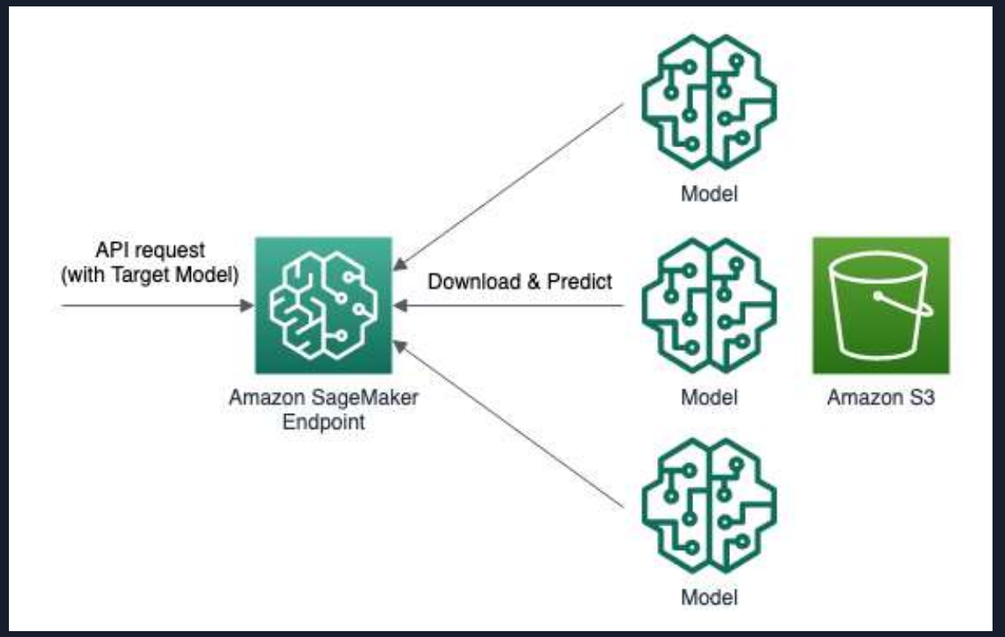
Amazon SageMaker Multi-Model Endpoints とは

- 単一Model Endpoint
  - APIリクエスト一つでAmazon S3に保存した学習済みモデルを使った推論しその結果が得られるホスティングサービス
  - 1つのEndpointに1つの学習済みモデルが割当可能
  - コンテナで動作



Amazon SageMaker Multi-Model Endpoints とは

- Multi-Model Endpoints
  - 単一Model Endpointを複数の学習済みモデルに拡張
  - APIリクエスト側でモデルを指定し、共有サービングコンテナが指定モデルを使い応答する
  - 1モデル1コンテナの動作でなくなるため、ホスティングコストを大きく削減できる



Amazon SageMaker Multi-Model Endpoints の未導入

- Amazon SageMaker Multi-Model Endpoints の導入で、
  よりシンプルに、かつマネージドリソースの活用を進められる

- BEDOREのようなマルチモデルをサービングするサービスにはもってこい
  - 自前API・Webサーバを実装せずに済む
  - モデルの保存・読み込み・メモリへの割当も自前実装のため、そこも省略可能
  - Amazon ECSのクラスタ管理から解放される

Amazon SageMaker Multi-Model Endpoints 導入への課題

- 初回リクエスト時に、Amazon S3からモデルをダウンロードした後に推論を行うためレイテンシー増が存在
  - BEDOREはリアルタイムに推論リクエストが送られる
  - レイテンシーはユーザ体験のためなるべく低くしたい
- 改善があれば、BEDOREでも利用可能になり更なる効率化が期待できる

モデルA（約400KB）での初回リクエスト

	現構成	Multi-Model Endpoints
Req-time [sec]	0.301	1.065

モデルB（約40MB）での初回リクエスト

	現構成	Multi-Model Endpoints
Req-time [sec]	0.695	1.448

学習側のパイプラインでは
Amazon SageMaker の利用によって効率化が実現

推論側でも Multi-Model Endpointsの
レイテンシ改善により
フルマネージドな機械学習のパイプライン実現に期待

まとめ

- BEDOREはクライアント毎に学習済みモデルを持ち、
  ユーザが任意で学習させられる
- 過去のAmazon EC2ベースからAmazon SageMakerベースにリアーキテクチャ
- Amazon SageMakerを学習基盤に用いることでフルマネージドの恩恵大
- マルチモデルサービス・リアルタイム推論においてMulti-Model Endpointsの
  今後に期待

一つのマルチモデルサービングサービスの事例として参考になれば。

最後に

- 日頃からTAMの方はじめ、AWSのサポートに感謝しております
- この場でもって言葉に

いつもありがとうございます。

Thank you!

Yoshihiko Shiraki
Software Engineer,
PKSHA Technology Inc.

Yoshikazu Miyoshi
Engineering Manager,
BEDORE Inc.