BETTER TOGETHER WITH intel

aws DEV DAY

LOCATION | JANUARY 15, 2021

A-5

[AWS Startup ゼミ]

よくある課題を一気に解説！
～御社の技術レベルがアップする 2021 秋期講習～

Kazuki Matsuda @mats16k
Startup Solutions Architect

Yuichiro Saito @koemu
Startup Solutions Architect

About us

松田 和樹　　@mats16k

スタートアップ ソリューションアーキテクト

創業期のスタートアップに２人目のエンジニア として入社し、上場前まで幅広い業務（SRE、データエンジニア、アプリ開発、情シス、採用、監査対応）に従事。現在は、スタートアップのお客様の支援をしながら 次のキャリアを模索中。

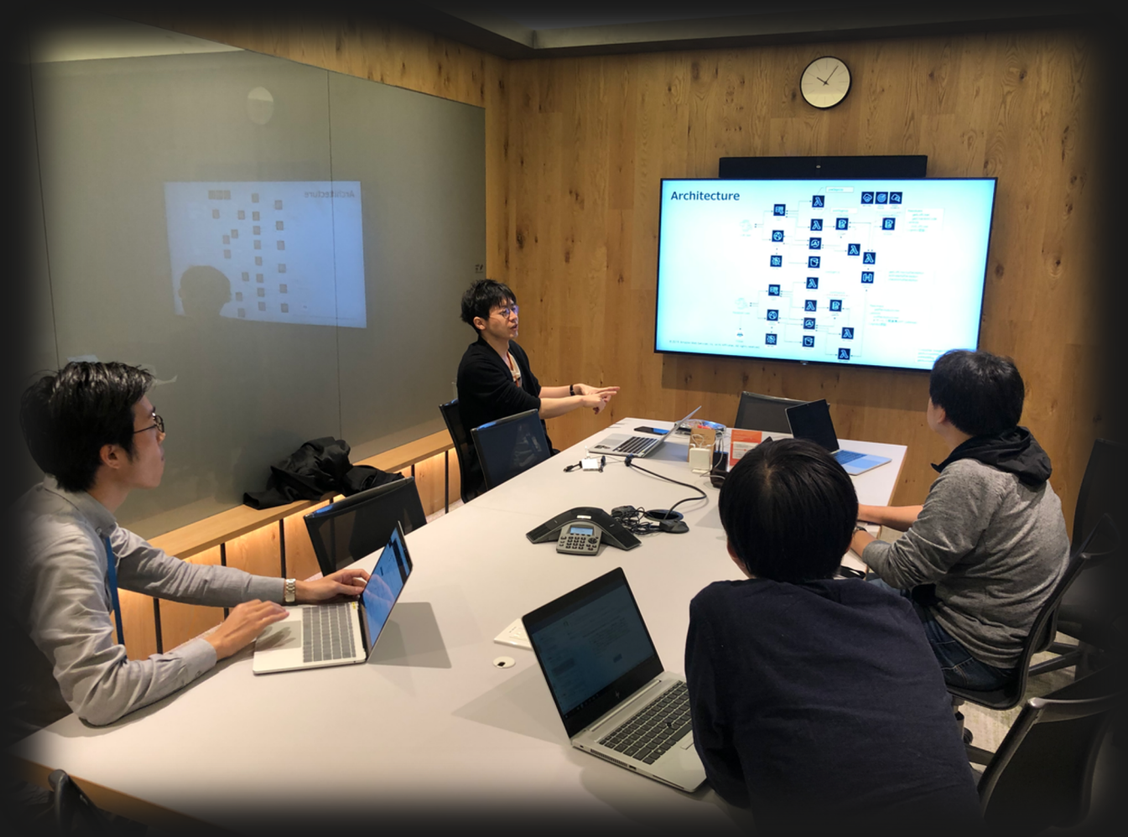
齋藤 祐一郎　　@koemu

スタートアップ ソリューションアーキテクト

バックエンドのソフトウェアエンジニアとして、数々のスタートアップ企業に勤務。直近では C to C サービス、FinTech (決済)の開発・運用に携わり、IPO などを経験。現在は、創業間もないスタートアップ企業様、FinTech 企業様の技術支援を担当。

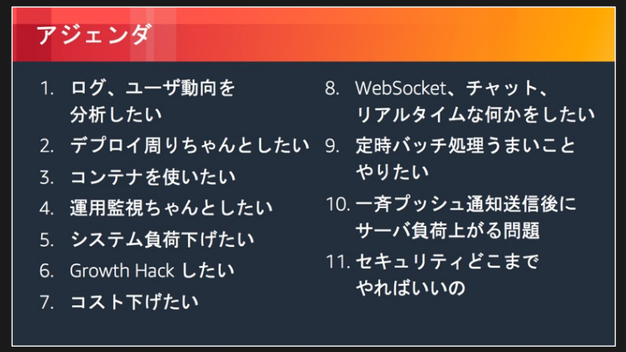
[AWS Startup ゼミ] よくある課題シリーズ

- AWS のスタートアップチームは、日々受けている技術相談からスタートアップにおける技術的な課題と傾向を把握
- このスライドは 2021年秋時点での「あるある」な課題をまとめたもの
- 詳細な解説を全てこのセッション・資料で行うのではなく、「こうしたい」に対して、「どの様に考えていけばよいか」を示す逆引き辞典となっている

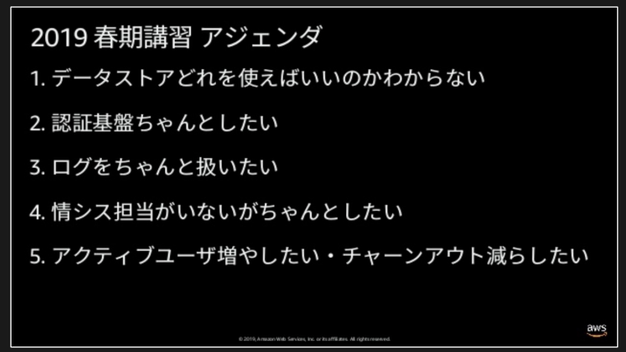


[AWS Startup ゼミ] これまでの発表

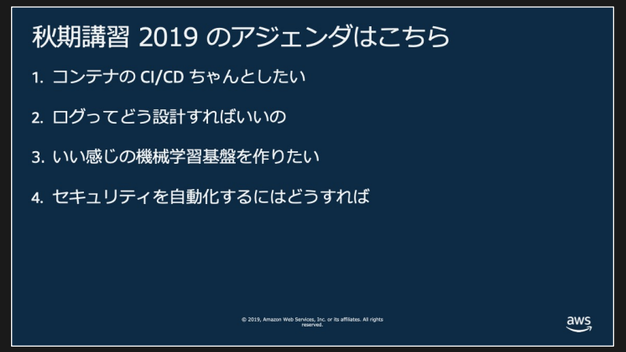
2018 秋期講習



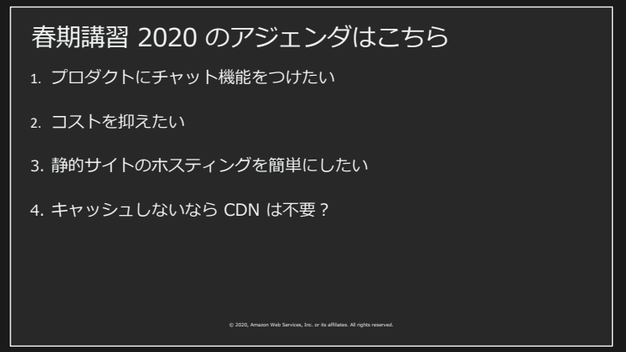
2019 春期講習



2019 秋期講習



2021 春期講習



2021秋期講習のアジェンダはこちら

1. 単一の AWS アカウントで開発してきたけど、そろそろ分けたい

2. CI/CD 簡単にやりたい

3. エンタープライズ企業からの要望でオンプレミスへの納品がしたい

4. とりあえず Amazon RDS/Aurora をスケールアップする運用をやめたい

1. 単一の AWS アカウントで
開発してきたけど、そろそろ分けたい

単一の AWS アカウントで開発してきたけど、そろそろ分けたい

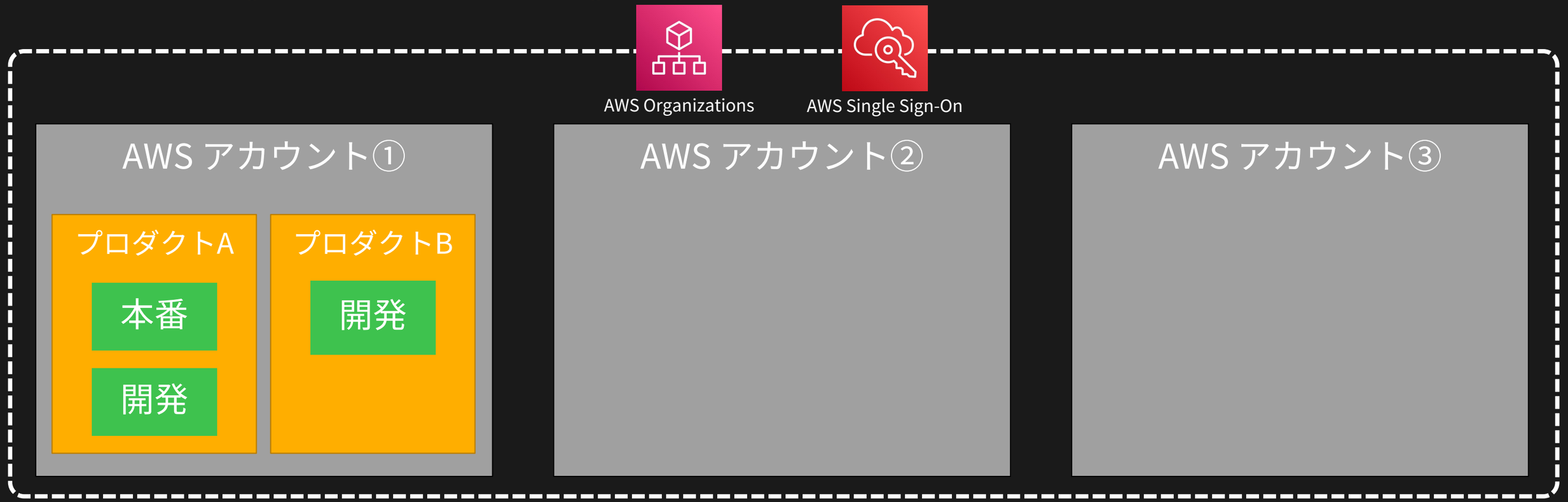
- よくいただく課題 😂
  - 「最初に作った AWS アカウントに本番環境も開発環境も入っていて、分けたい。。」
  - 「AWS アカウント分割してみたけど、アカウント増えてむしろ管理が大変になった。。」
- 本当にしたいことは何？ 😳
  - 「既存環境に影響を与えること無く、ガバナンスや運用を改善したい」
  - 「今後の AWS アカウント増にも備えたい」
- 思考フロー 😌
  1. 移管先の AWS アカウントを用意しよう
  2. 本番環境への影響度が低いものから移管しよう
  3. ネットワークの疎通が必要な場合は VPC 同士をつなごう

1. 移管先の AWS アカウントを用意しよう

- リソース移管先の AWS アカウントを用意する

1. 移管先の AWS アカウントを用意しよう

- リソース移管先の AWS アカウントを用意する
- 単純に増やすと管理が煩雑になるため、AWS Control Tower を利用して
  マルチアカウント環境を整備することを推奨
  - （AWS Organizations や AWS Single Sign-On の設定を簡素化することが出来ます）

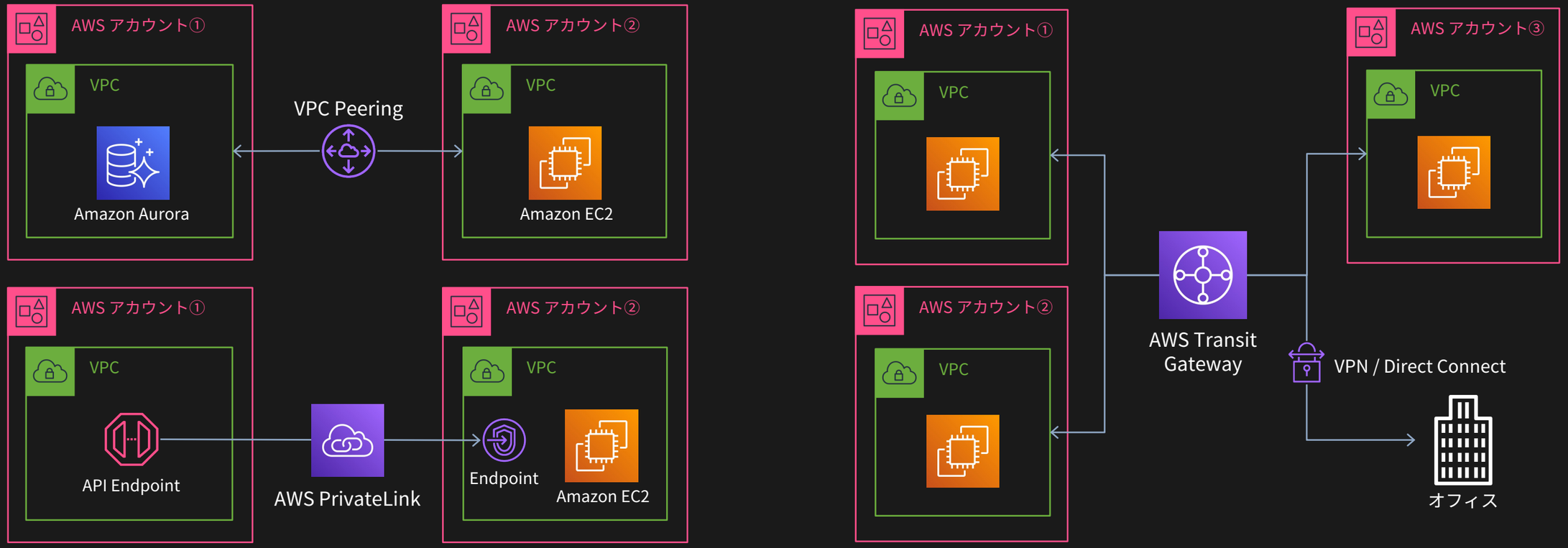


2. 本番環境への影響度が低いものから移管しよう

- 本番環境への影響度が低い環境から移管する
  - Infrastructure as Code が実践できている場合は移行が容易
  - データについては別途移管が必要な点に注意
- 本番環境の整理を目的に、別アカウントに再構築するのも選択肢のひとつ

3. ネットワークの疎通が必要な場合は VPC 同士をつなごう

- 必要に応じて VPC Peering や Transit Gateway 等で VPC 間をつなぐ
  - IP アドレスの範囲が被らないように注意する
  - 恒久対応 or 暫定対応 なのかを明確にする
- 通信が片方向の場合は PrivateLink の利用も検討する



実装時の参考資料と事例集

- AWS サービス別資料集
  - AWS アカウント シングルサインオンの設計と運用 (資料 | 動画)
  - Amazon VPC (資料 | 動画)
  - AWS Transit Gateway (資料 | 動画)
- AWS Summit / AWS Dev Day / その他イベント
  - マルチアカウント運用での権限移譲と統制の両立 - AWS Summit Tokyo 2019 (資料 | 動画)
  - マルチアカウント管理の基本 - AWS Expert Online 2021 (資料)
  - 増加するシステムをマルチアカウントで効率よく管理する - AWS Innovate 2020 (資料)
  - Using AWS Control Tower to govern multi-account AWS environments at scale - AWS re:Inforce 2019 (資料 | 動画)
  - Managing Multi-Account AWS Environments Using AWS Organizations - AWS re:Inforce 2019 (資料 | 動画)
- その他の資料 / Blog記事
  - スタートアップにおけるマルチアカウントの考え方と AWS Control Tower のすゝめ
  - AWS Control Towerを利用したマルチアカウント管理とセキュリティ統制 - JAWS DAYS 2021

2. CI/CD を簡単にやりたい

CI/CD を簡単にやりたい

- よくいただく課題 😂
  - 「CI/CD を整備したいけど、よくわからん…」
  - 「既存環境へのつなぎ込みが難しい…」
- 本当にしたいことは何？ 😳
  - 「デプロイを自動化したい」
  - 「CI/CD を通してプロダクトの品質を上げたい、開発効率を上げたい」
- 思考フロー 😌
  1. 自動デプロイが組み込み済みのプロダクトを使おう
  2. **ツールチェーンが提供するデプロイパイプラインを利用しよう**
  3. プロダクトとツールの特性を理解し、自分で実装しよう

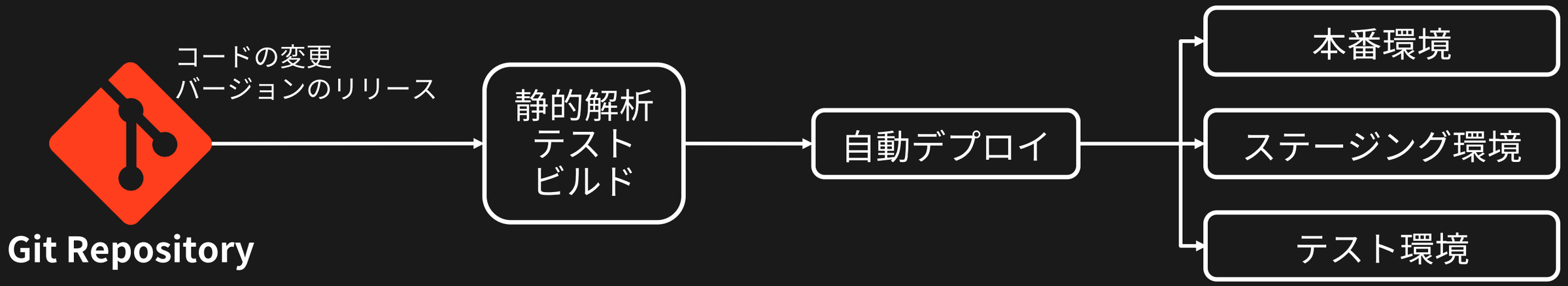
0. CI/CD とは

「 Continuous Integration / Continuous Delivery 」の略

※ 直訳すると、継続的インティグレーション / 継続的デリバリー

CI/CD は、アプリケーション開発を効率化する仕組みや開発手法のことで、単一の技術を指すものではない

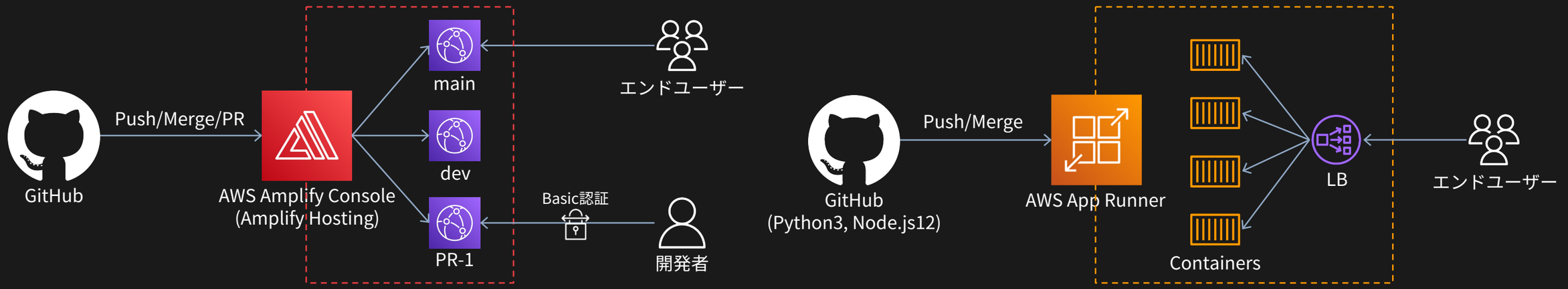
CI/CD パイプラインのイメージ



1. 自動デプロイが組み込み済みのプロダクトを使おう

AWS には GitHub と接続するだけでデプロイ可能なサービスがある

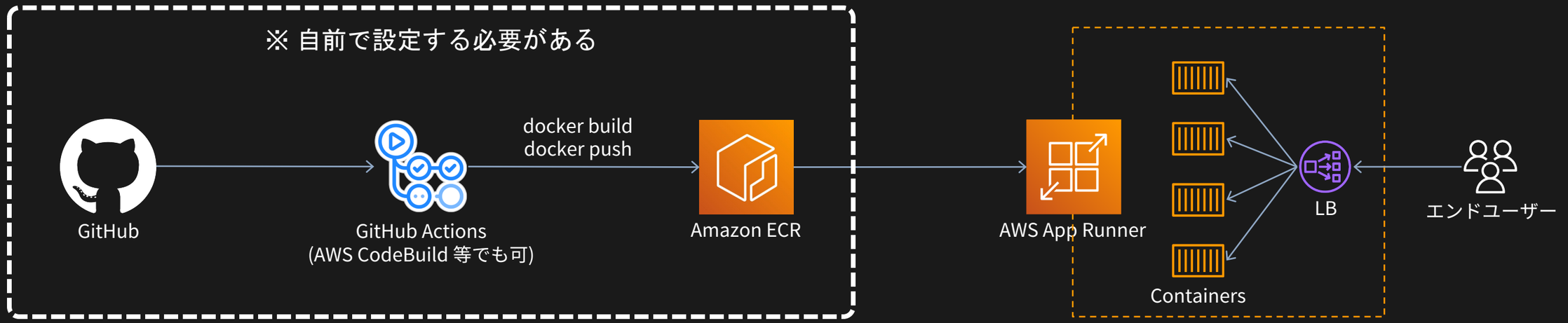
- **AWS Amplify Console** : 静的サイト、SPA のフロントエンド
  - ブランチごとのホスティング、プルリクエストのプレビュー、Basic認証
  - （バックエンドの API 等も Amplify CLI で構成していれば可能だが、今回は割愛）
- **AWS App Runner** : Python3, Node.js12 製のアプリケーション



1. 自動デプロイが組み込み済みのプロダクトを使おう

AWS には GitHub と接続するだけでデプロイ可能なサービスがある

- **AWS Amplify Console** : 静的サイト、SPA のフロントエンド
  - ブランチごとのホスティング、プルリクエストのプレビュー、Basic認証
  - （バックエンドの API 等も Amplify CLI で構成していれば可能だが、今回は割愛）
- **AWS App Runner** : Python3, Node.js12 製のアプリケーション
  - その他のランタイムもコンテナイメージを持ち込むことで対応可能



2. ツールチェーンが提供するデプロイパイプラインを利用しよう

AWS が提供するツールや OSS によっては、デプロイパイプラインを
簡単に構築できるサブコマンドやオプションが提供されている。

- **AWS Copilot CLI**
  - Amazon ECS や AWS App Runner を利用したコンテナアプリケーション
- **AWS SAM Pipelines** (Serverless Application Model)
  - サーバーレスアプリケーション
  - AWS CodePipeline の他、Jenkins, GitLab CI/CD, GitHub Actions 向けの雛形も生成可能
- **AWS CDK Pipelines** (Cloud Development Kit)
  - AWS CloudFormation で構築可能なあらゆるアプリケーション
  - 任意のプログラミング言語で記述可能

3. プロダクトとツールの特性を理解し、自分で実装しよう

実環境が複雑、あるいは仮想マシンのリソースが多いなど抽象化することが難しい場合は、その環境に合ったデプロイパイプラインを個別に実装してあげる必要がある。

AWS の SA に
相談しよう！

CI/CDをちゃんとしたい

- よくいただく課題😂
  - 「今はSSHで入ってgit pullでデプロイしてるんですが」
  - 「AWSのベストプラクティス知りたい」
- 本当にしたいことは何？🤔
  - 自動化されたリリースプロセスを構築して楽したい
  - まず何をすればいいのか分からないので知りたい！
- 思考フロー🤨
  1. サーバをステートレスにしよう
  2. Pull型のデプロイ手段を使おう
  3. CI/CDパイプラインを構築したくなったら？

#awsstartup amazon web services

コンテナの CI/CD ちゃんとしたい

- よくいただく課題 😂
  - 「 コンテナ使いたいんだけど、Build とかあって逆にめんどくさくなってる気がする」
  - 「 自動テスト、自動デプロイみたいなのやりたいけどみんなどうしてるの？」
- 本当にしたいことは何？ 😳
  - 開発速度を上げ、プロダクトの価値を向上させたい
  - テストをちゃんと行い、プロダクトのクオリティを上げたい
  - 自動化しつつも状況の把握やロールバック等のハンドリングを適切にやりたい
- 思考フロー 😌
  1. 手作業でのデプロイと決別する
  2. リビジョンとデプロイメントを連動させることを考える
  3. マネージドサービスを中心にパイプラインを構成する

aws

実装時の参考資料と事例集

- AWS サービス別資料集 / ドキュメント
  - AWS Amplify (資料 | 動画), AWS App Runner (ドキュメント), AWS Copilot CLI (ドキュメント)
- AWS Summit / AWS Dev Day / その他イベント
  - AWSの 継続的インテグレーション／デリバリー総まとめ！モダンアプリケーション構築のためのCI / CD ベストプラクティス！ - AWS Summit 2021 (資料 | 動画)
  - Code シリーズを利用したパイプラインの自動化入門 - AWS Summit 2018 (資料 | 動画)
- その他の資料 / Blog記事
  - AWS Amplify でのフルスタックアプリケーションの CI/CD パイプラインの構築
  - AWS Copilot によるコンテナアプリケーションの自動デプロイ
  - AWS Copilot CLI を使用した永続性を持つ AWS App Runner サービスの継続的ワークフローの実現
  - Introducing AWS SAM Pipelines: Automatically generate deployment pipelines for serverless applications
  - AWS SAM Pipelines - Serverless Land
  - CDK Pipelines: AWS CDK アプリケーションの継続的デリバリ
  - Copilot Primer Workshop - AWS workshop studio

3. エンタープライズ企業からの要望で オンプレミスへの納品がしたい

エンタープライズ企業からの要望でオンプレミスへの納品がしたい

- よくいただく課題 😂
  - 「エンタープライズ企業から "オンプレミス版の提供" について要望を頂いているが、対応方針に悩んでいる」
- 本当にしたいことは何？ 😳
  - 「エンタープライズ側のセキュリティ基準を守りながら、安心してサービスを提供したい」
  - 「サービス提供側からアプリケーションを管理可能にし、俊敏性を担保したい」
- 思考フロー 😌
  1. **社外に説明できるように、自社の運用体制を整えよう**
  2. **閉域網で接続可能なサービスの利用を検討してみよう**
  3. サービス提供者側からメンテナンスできる仕組みを考えよう

0. エンタープライズ側が、どのような懸念を持っているか確認しよう

- 「オンプレミス」という言葉の背景にある懸念について、エンタープライズの方にじっくり伺ってみましょう。悩みに寄り添うことで、オンプレミス版を追加開発をすることなく、クラウドで実現できる別の方法が見つかるかもしれません。

- こんな質問を投げかけてみると良いでしょう。
  - 「現在の私たちのサービスの、どの部分にどのようなご懸念をお持ちでしょうか。」
  - 「御社内でクラウドサービスを利用する際に、規定されたルールはありますか。」
    - 外部発注する際の規定
    - 使用可能なAWSのサービスが定められている場合はその内容
    - 保存しているデータとその管理方法
  - 「現在、他のクラウド版のSaaSをご利用いただいていますでしょうか。もし利用されている場合、どのような形でしょうか。」

1. 社外に説明できるように、自社の運用体制を整えよう

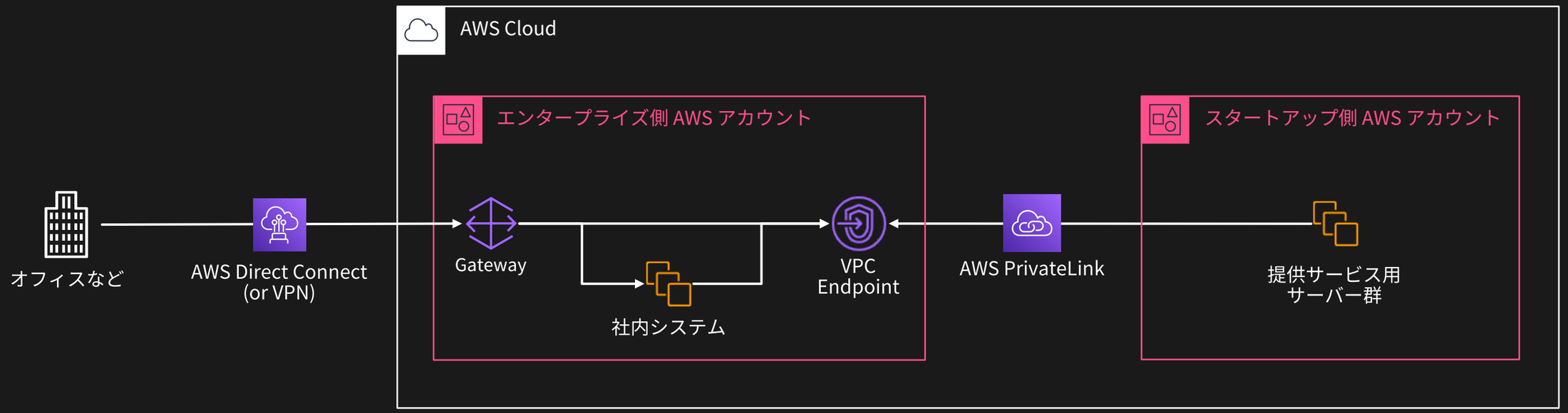
- 「スタートアップ企業は内部統制が発展途上であり、組織として適切なデータの取り扱う仕組みが十分ではない」と先入観で思われていることがあります。この懸念を払拭できる説明ができることも大切です。

- **AWS Well-Architected Framework**
  - クラウド設計・運用のベストプラクティス集
  - これをもとに現状をレビューし、必要に応じ改善を踏まえ、自社のシステムの状態を説明可能にします。

2. 閉域網で接続可能なサービスの利用を検討してみよう

- パブリックネットワーク（インターネット）を直接経由することなく、AWS 上にあるサービスを提供することができます。

- **AWS PrivateLink** : VPC エンドポイントとして AWS アカウントを跨いでサービスを提供可能

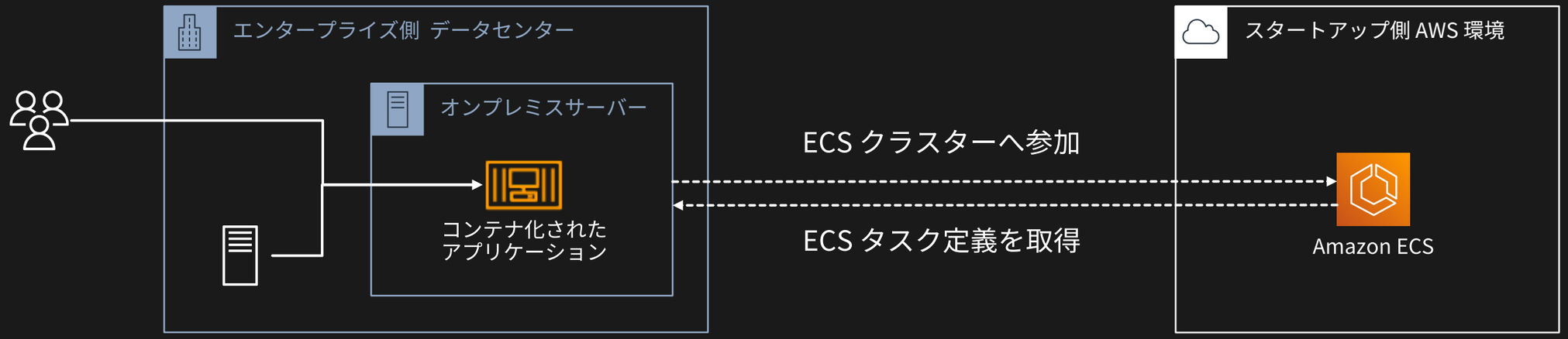


3. サービス提供者側からメンテナンスできる仕組みを考えよう

エンタープライズ側のシステムにカスタマイズして導入する場合でも、
AWS を通じアプリケーションを管理・運用する方法があります。

<u>たとえば、</u>

- Amazon ECS Anywhare : お客様自身で管理している IT インフラ上の仮想マシンを Amazon ECS クラスタに参加させることが可能



実装時の参考資料と事例集

- AWS サービス別資料
  - AWS Well-Architected Framework (資料 | 動画 | Blog)
  - Amazon Virtual Private Cloud (VPC) (資料 | 動画 | Blog)
  - AWS Direct Connect (資料 | 動画 | Blog)
- ホワイトペーパー
  - Securely Access Services Over AWS PrivateLink
- その他の資料 / Blog記事
  - Startup.fm 新春特別企画 – AWS セキュリティワークショップ
  - Building an Amazon ECS Anywhere home lab with Amazon VPC network connectivity
  - Securing access to AMIs in AWS Marketplace

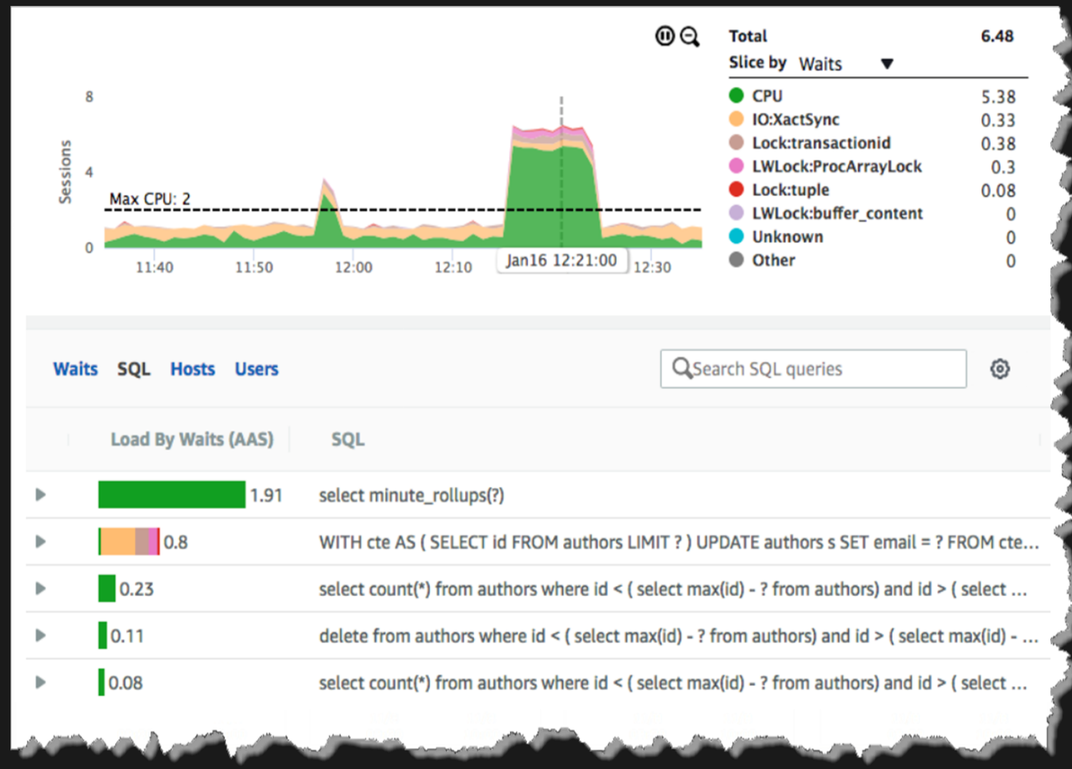
4. とりあえず Amazon RDS/Aurora をスケールアップする運用をやめたい

とりあえず Amazon RDS/Aurora をスケールアップする運用をやめたい

- よくいただく課題 😂
  - 「事業の伸びに合わせてスケールアップを繰り返しているが、コストが嵩んできている…」
  - 「突然 DB からの応答がなくなったり遅くなったりし、ひどい時は再起動しかできなくなる」
- 本当にしたいことは何？ 😳
  - 「事業の成長に対応できる、スケーラブルなプロダクトにしたい」
- 思考フロー 😌
  1. リードレプリカを利用して、参照をスケールアウトさせよう
  2. キャッシュを活用して、参照負荷を軽減しよう
  3. キューを活用して、書き込み負荷を軽減しよう
  4. 即時性が求められる書き込みが多量に発生するテーブルは NoSQL への移行を検討しよう

0.「推測するな、計測せよ。」

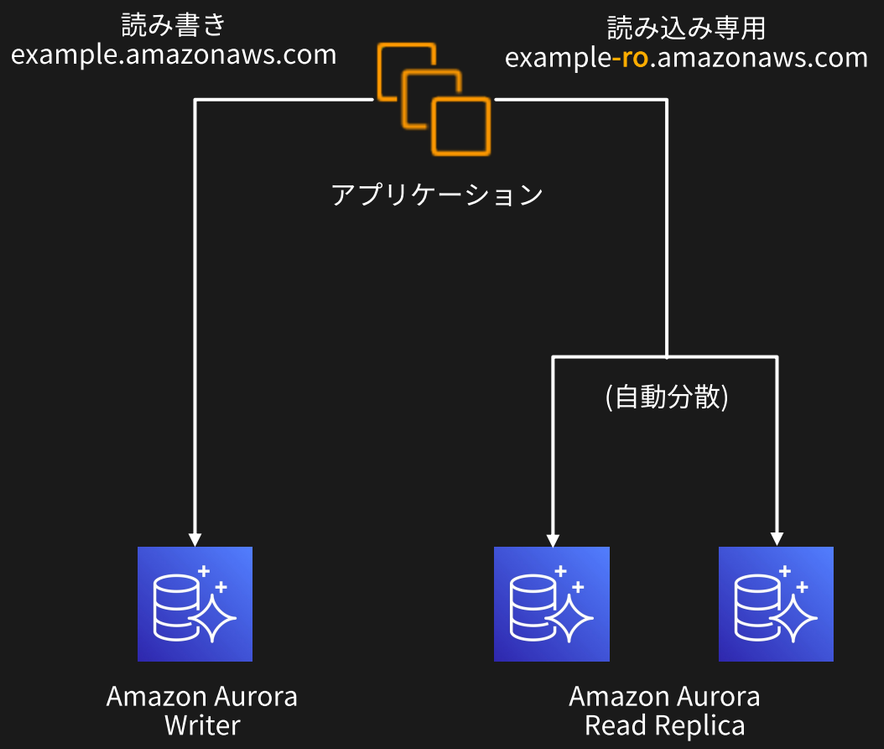
- Amazon RDS/Aurora には、
  RDS Performance Insights という、
  パフォーマンスの統計を取るサービス
  があります。
  問題のあるクエリの調査が可能。



1. Index が効いておらず参照コストが高い SQL
2. アプリケーション側から N+1 の問い合わせが
   あり結果として時間がかかるもの
3. 集計関数を用いており1回あたりに取り出す
   レコード数が多くなりDBサーバに負荷をかけて
   いるもの

1. リードレプリカを利用して、参照をスケールアウトさせよう

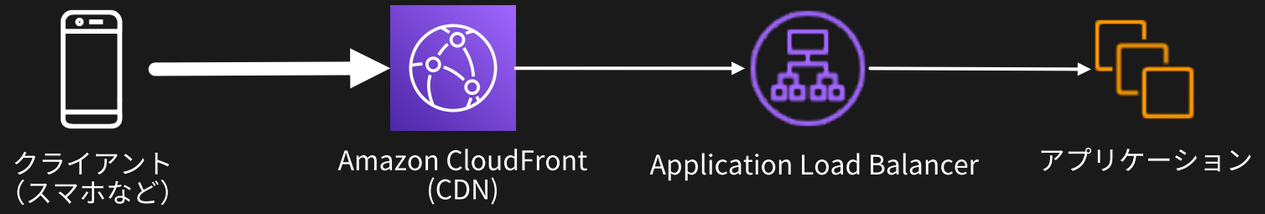
- Amazon RDS/Aurora はリードレプリカを増やすことが可能。
  - 参照専用のインスタンスを用意することで、ライターインスタンスの負荷を軽減することができます。
- リードレプリカを利用にあたっては、アプリケーション側で参照時に専用のエンドポイントを利用するよう実装する必要がある。



2. キャッシュを活用して、参照負荷を軽減しよう

- RDBMSの問い合わせ結果をキャッシュし、参照負荷を軽減します。

- Amazon ElastiCache: 情報をメモリ上に保持することで、問い合わせ速度を上げられます。
  - 一度取得した情報を ElastiCache にキャッシュする。なお、取得した情報が RDBMS 上で更新された場合、そのキャッシュを破棄しなければデータの不整合が発生する。
- Amazon CloudFront: Web ページ・API レスポンスをキャッシュします
  - 更新頻度が高くない/ユーザー毎にカスタマイズ不要なWebページや、マスタデータに効果的。



3. キューを活用して、書き込み負荷を軽減しよう

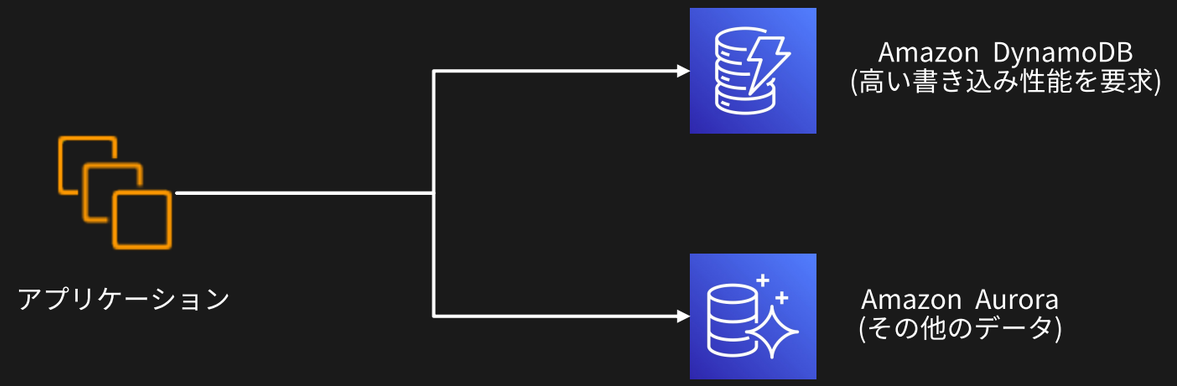
- 書き込み内容をキューに溜め、処理を1〜数個に直列化し、負荷を軽減します。

- Amazon SQS: 非同期処理を実現するメッセージキューイングサービス
  - 書き込む情報を SQS に溜め込み、別のプログラムが非同期に順次 RDBMS に書き込むなどの処理を実現できます。

4. 即時性が求められる書き込みが多量に発生するテーブルは NoSQL への移行を検討しよう

- RDBMS は、ライターインスタンスは原則1つのみで、アプリケーション改修なしでの書込み負荷対策にはスケールアップ以外にはありません。

- Amazon DynamoDB: 低レイテンシで、読み書き共にスケールする NoSQL データベースサービス
  - 書込み処理自体を DBMS 側でスケールでき、根本的な対応が可能です。ただしアプリケーションの改修が必要です。



実装時の参考資料と事例集

- AWS サービス別資料
  - Amazon Aurora MySQL Compatible Edition ユースケース毎のスケーリング手法 (資料 | 動画 | Blog)
  - Amazon DynamoDB Advanced Design Pattern (資料 | 動画 | Blog)
  - Amazon DynamoDB (資料), Amazon ElastiCache (資料), Amazon SQS (資料 | 動画)
- AWS Blog / ハンズオン
  - Amazon Aurora Serverless と Amazon ElastiCache を使用してリアルタイムなリーダーボードをビルドする
  - Performance Insights を使用して Amazon Aurora の MySQL のワークロードを分析する
  - Level Up Your Games with Amazon Aurora
- その他の資料
  - Active Record で複数のデータベース利用 (Ruby on Rails)
  - Laravel 8.x データベース：準備 (PHP)
  - AWS モダンアプリケーション開発 ホワイトペーパー -コマンドクエリ責務分離 (CQRS)

まとめ

まとめ

2021 秋期講習のアジェンダは以下のとおりでした

1. 単一の AWS アカウントで開発してきたけど、そろそろ分けたい
2. CI/CD 簡単にやりたい
3. エンタープライズ企業からの要望でオンプレミスへの納品がしたい
4. とりあえず Amazon RDS/Aurora をスケールアップする運用をやめたい

Thank you!

Kazuki Matsuda (@mats16k)
Startup Solutions Architect

Yuichiro Saito (@koemu)
Startup Solutions Architect