
aws
SUMMIT
ONLINE
JAPAN

暗号資産(仮想通貨)取引所を支えるサーバーレスログ基盤の裏側

Kaori Yatsu

システム部

ビットバンク 株式会社

aws SUMMIT ONLINE

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

自己紹介

名前

谷津 香 (やつ かおり)

所属

ビットバンク株式会社

システム部 インフラチーム

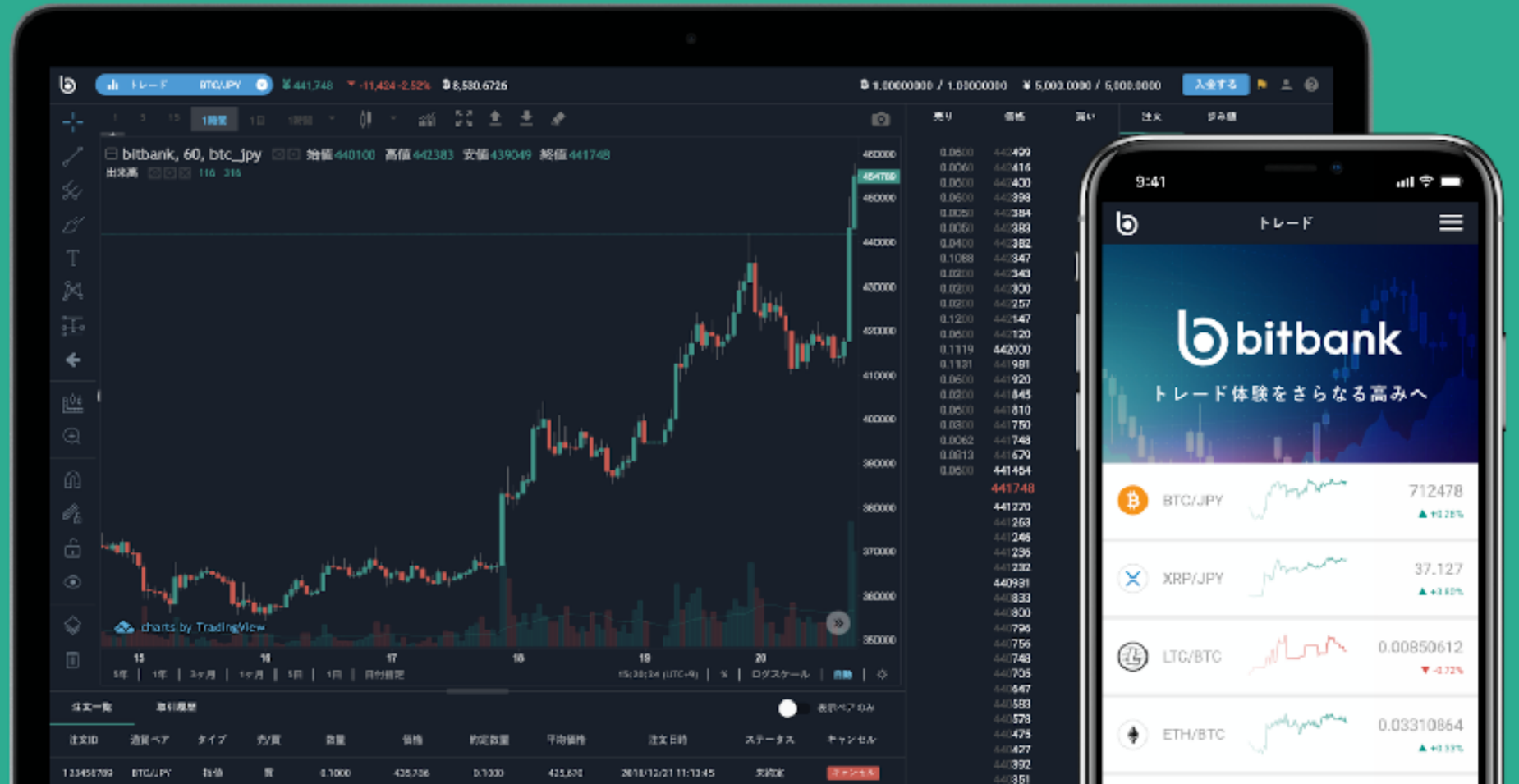
経歴

小売系のシステム会社でポイントシステムの
開発・運用を担当

暗号資産（仮想通貨）取引所 - bitbank.cc

ビットコイン・アルトコインの板取引が可能な暗号資産（仮想通貨）取引所

2017年3月よりサービスを開始し、国内の暗号資産（仮想通貨）取引所では最大級の規模まで発展



アジェンダ

- ログ基盤ができる前
- 課題
- どのように解決していったか
- アーキテクチャ
- 新たに見えてきた課題
- 今後の取り組み
- まとめ

今回話すこと

アマゾン ウェブ サービス (AWS)のマネージドサービスを活用してログ基盤を構築した当社の事例についてお話しします。

組織上の制約とログ基盤に求めたものに対して、
Amazon Kinesis Familyを使いどう解決したかをご紹介します。

ログ基盤ができる前

aws SUMMIT ONLINE

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

前提

ビットバンクでは全サービスをAWS上に構築しログを取得している

アプリケーションログ

- AWS Fargate, AWS ElasticBeanstalk, AWS Lambda, etc…
- Amazon CloudWatch Logsに集約

セキュリティログ

- AWS CloudTrail
- Amazon Aurora監査ログ

マルチアカウント管理

前提

ビットバンクでは全サービスをAWS上に構築しログを取得している

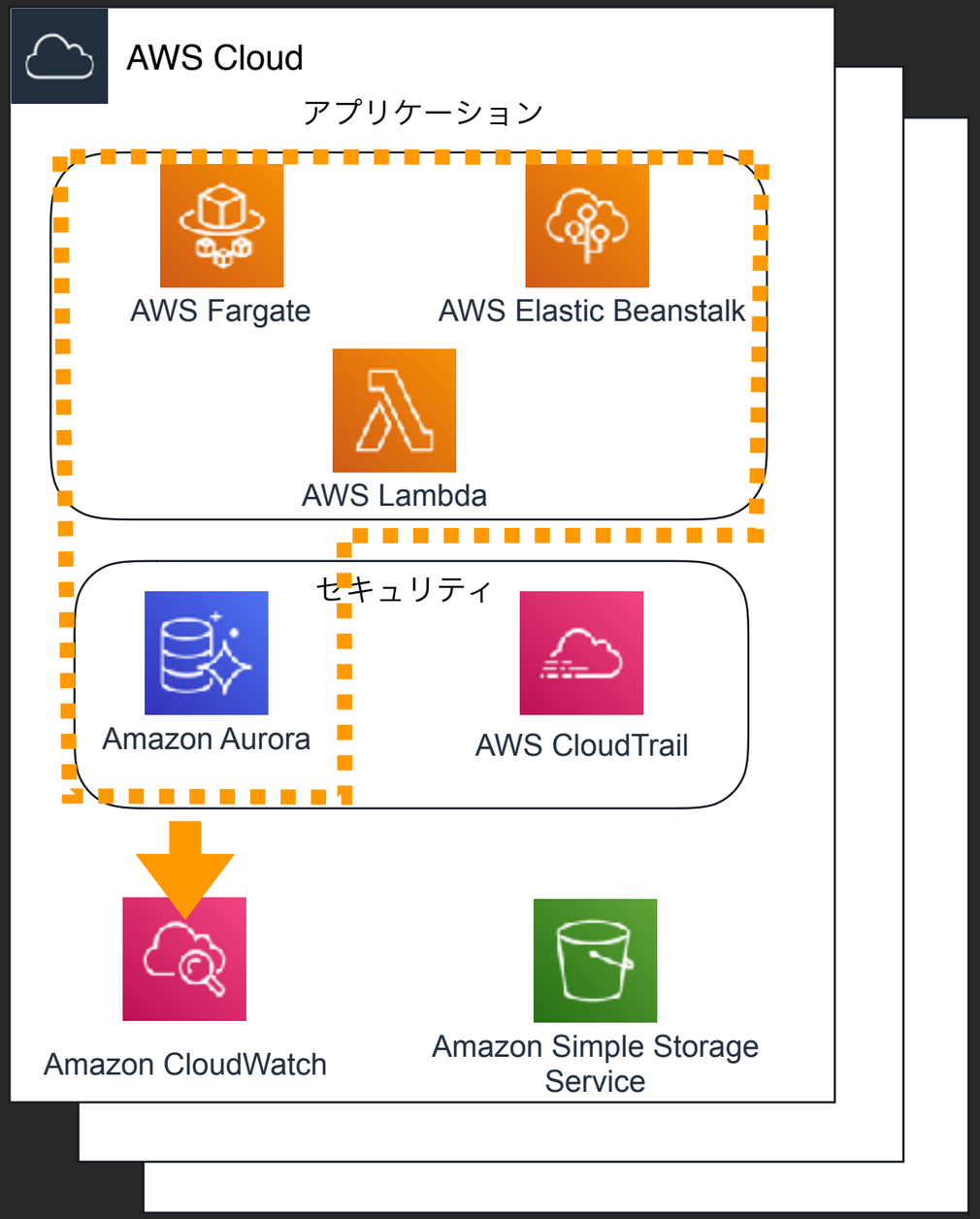
アプリケーションログ

- AWS Fargate, AWS ElasticBeanstalk, AWS Lambda, etc…
- Amazon CloudWatch Logsに集約

セキュリティログ

- AWS CloudTrail
- Amazon Aurora監査ログ

マルチアカウント管理



AWSアカウント管理

- AWS Organizations
- マルチアカウント構成

以前の構成

アプリケーションログ

- CloudWatch Logsに送信しLambdaでSlackにエラー通知
- エラーがあったらCloudWatchを直接見に行き調査をしていた

セキュリティ・監査ログ

- AuroraのDBユーザーのクエリログを取得
- CloudWatch Logsを目検していた

課題

- ログ分析ができてない
- 監査要件でのログ抽出に手間がかかっていた
- 高度なリアルタイムアラートができてない

ログ基盤に求められたもの

運用

・負荷を最小限に

・NoOps化

分析・通知

・分析しやすいログ形式で保存

・リアルタイム性

・高度な分析ができる

再現性

・既存の環境を壊さない

・構成をコードで管理

どのように解決していったか

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

設計の考え方

設計するにあたって次の点を指針とした

NoOps 化

- 人手が必要な作業が最小限になること

ポータビリティ

- 新しいアプリケーションログを迅速にログ連携できること
- 他(Dev / Prod / Stg)の環境でも容易にログ連携できること

既存環境への影響を最小限に

- すでに稼働しているアプリケーションに影響が出ないこと
- 問題があった際の切り戻しが容易であること

今回利用したKinesis Family

Amazon Kinesis Data Streams

データストリームを収集・保存・処理

Amazon Kinesis Data Firehose

データストリームをデータストアに配信

Amazon Kinesis Data Analytics

データストリームをSQLで分析

Amazon Kinesisを採用した理由①

Amazon Kinesis Data Streams

- CloudWatch Logsをソースにしたクロスアカウント連携に対応
- スケールできるため管理が容易
- 他のKinesisと組み合わせて使うときの入口として利用

Amazon Kinesisを採用した理由②

Amazon Kinesis Data Firehose

- Amazon S3にデータを時系列順に置ける
- データをLambdaで加工ができるので分析しやすい形に置ける
- ログ分析ツールとの親和性が高い

Amazon Kinesisを採用した理由③

Amazon Kinesis Data Analytics

- Lambdaと組み合わせて利用
- リアルタイムに分析が可能
- SQLを用いてデータのクエリを柔軟に設定できる
- 送信先にLambdaを指定可能

Amazon SQSを使わずにAmazon Kinesisにした理由

SQSを使った場合

- ログが急増した場合処理が遅延してしまう
- 既存環境を壊さないように設計すると構成が複雑になり開発コストが上がる
    - Lambdaでメインロジックを複数持つ
        - Lambdaの実装が肥大化
            - Slack通知、フィルターの実装、ログの加工、Amazon S3に時系列に配信など
- LambdaがQueueに入れる時ログの順番の保証ができない

Amazon SQSを使わずにAmazon Kinesisにした理由

SQSを使った場合

- ログが急増した場合処理が遅延してしまう
- 既存環境を壊さないように設計すると構成が複雑になり開発コストが上がる
  - Lambdaでメインロジックを複数持つ
    - Lambdaの実装が肥大化
      - Slack通知、フィルターの実装、ログの加工、Amazon S3に時系列に配信など
- LambdaがQueueに入れる時ログの順番の保証ができない

Lambdaによる実装範囲と管理対象をできるだけ減らしたかった

Amazon SQSを使わずにAmazon Kinesisにした理由

SQSを使った場合

・ログが急増した場合処理が遅延してしまう
・既存環境を壊さないように設計すると構成が複雑になり開発コストが上がる
　・Lambdaでメインロジックを複数持つ
　　・Lambdaの実装が肥大化
　　　・Slack通知、フィルターの実装、ログの加工、Amazon S3に時系列に配信など
・LambdaがQueueに入れる時ログの順番の保証ができない

これらの問題を解決してくれた

Lambdaによる実装範囲と管理対象をできるだけ減らしたかった

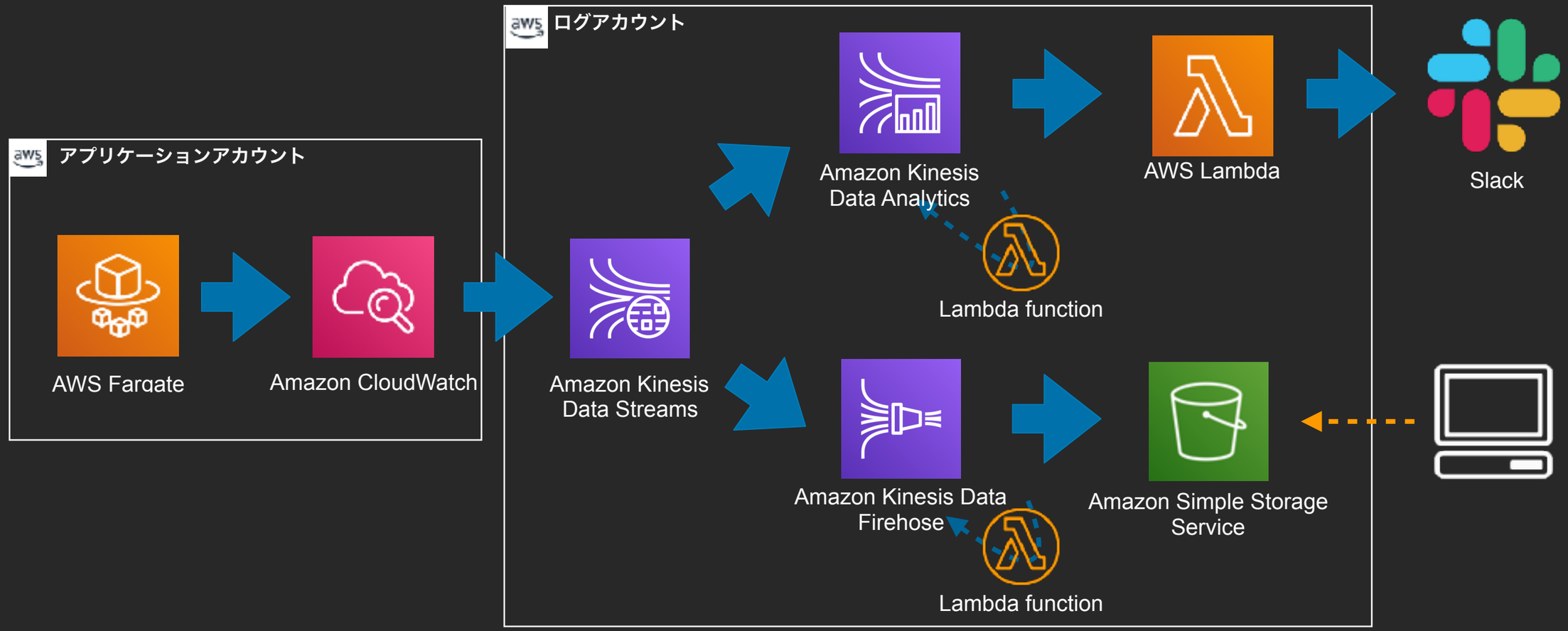
アーキテクチャ

aws SUMMIT ONLINE

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

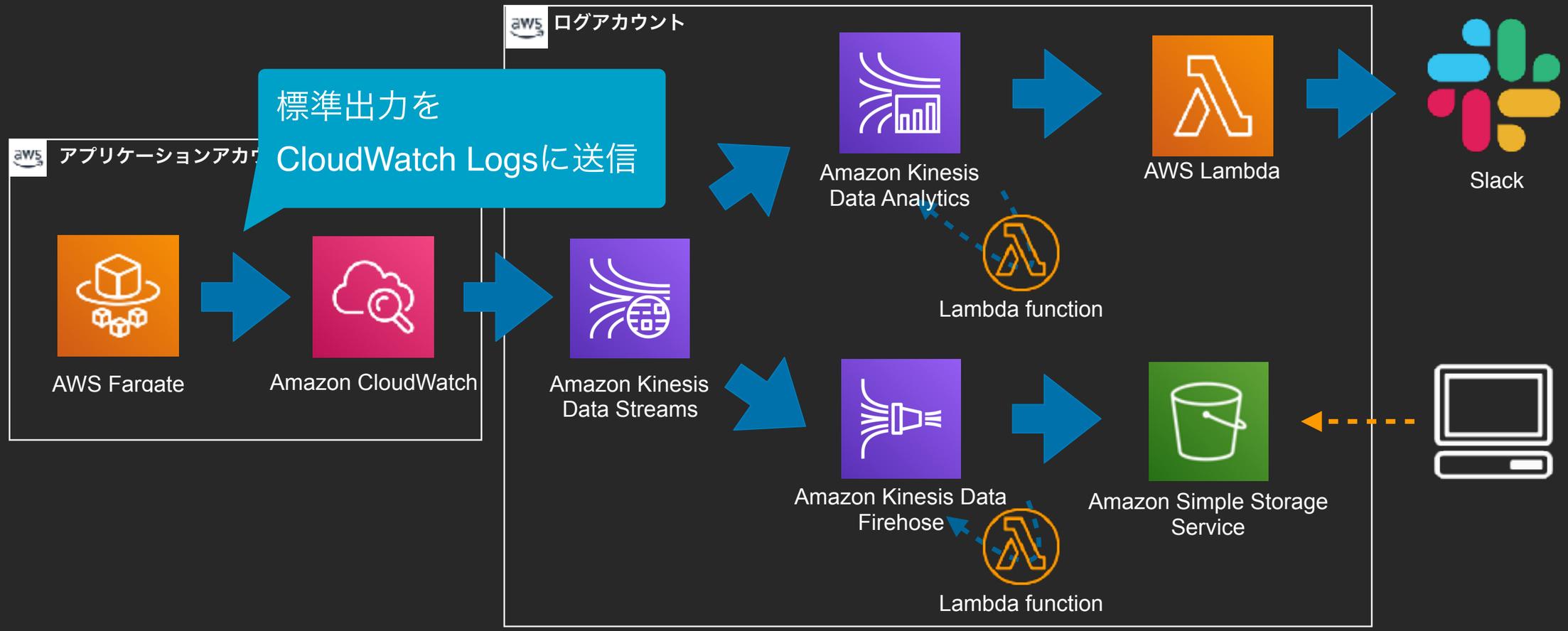
Fargateのパターン

- awslogsログドライバを活用
- 標準出力をCloudWatch Logsに送信



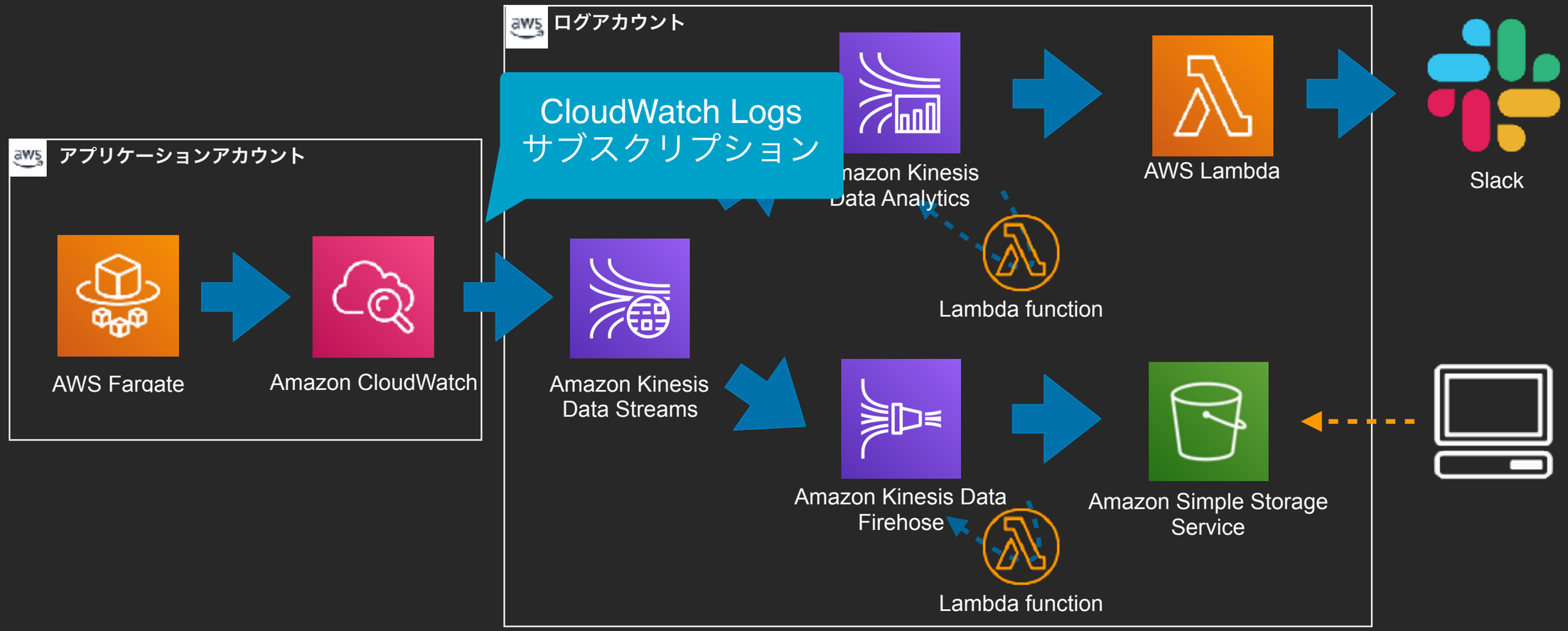
Fargateのパターン

- awslogsログドライバを活用
- 標準出力をCloudWatch Logsに送信



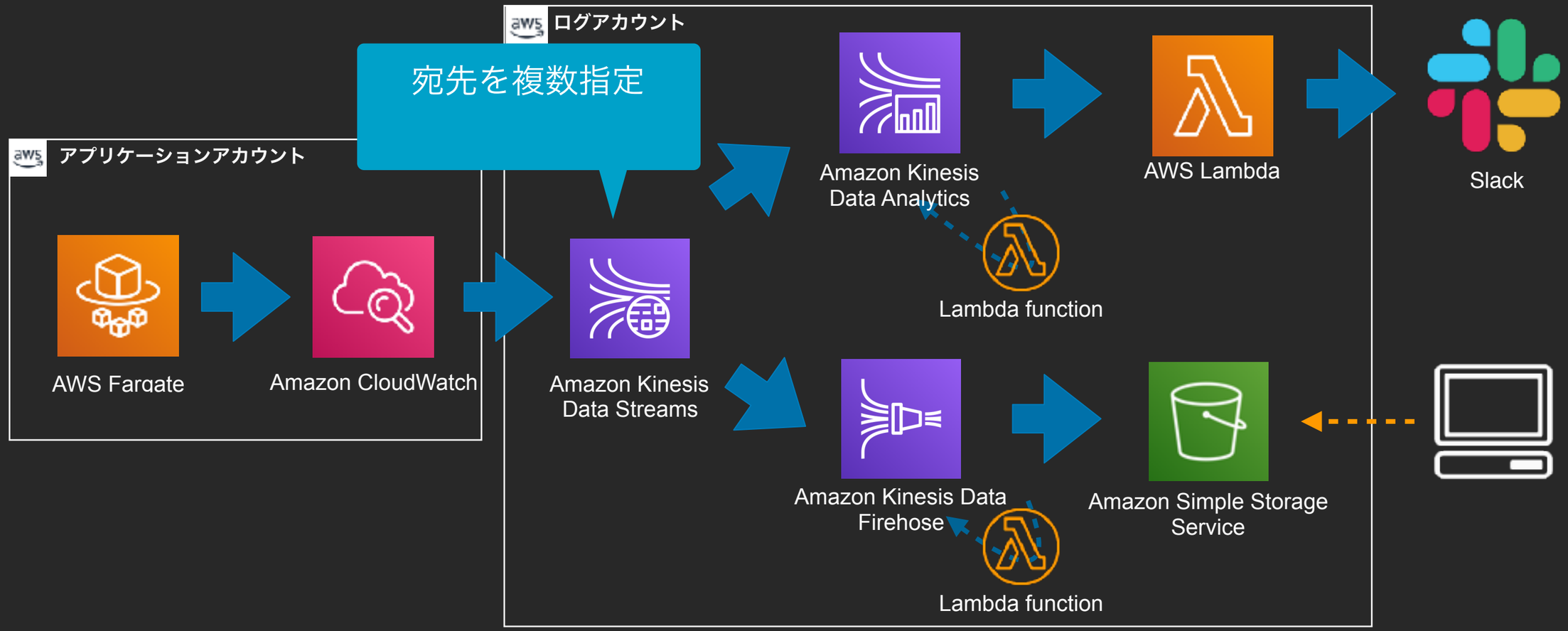
Fargateのパターン

- awslogsログドライバを活用
- 標準出力をCloudWatch Logsに送信



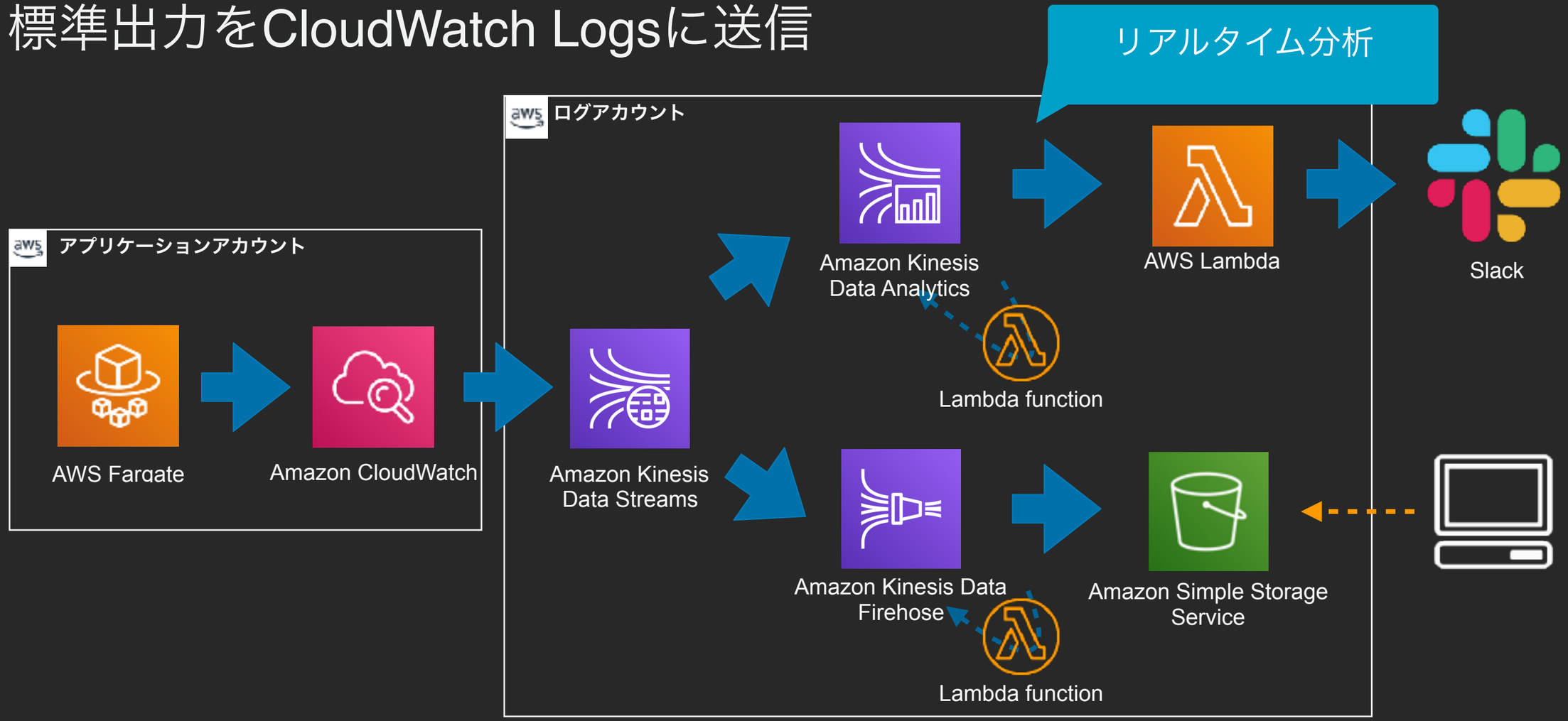
Fargateのパターン

- awslogsログドライバを活用
- 標準出力をCloudWatch Logsに送信



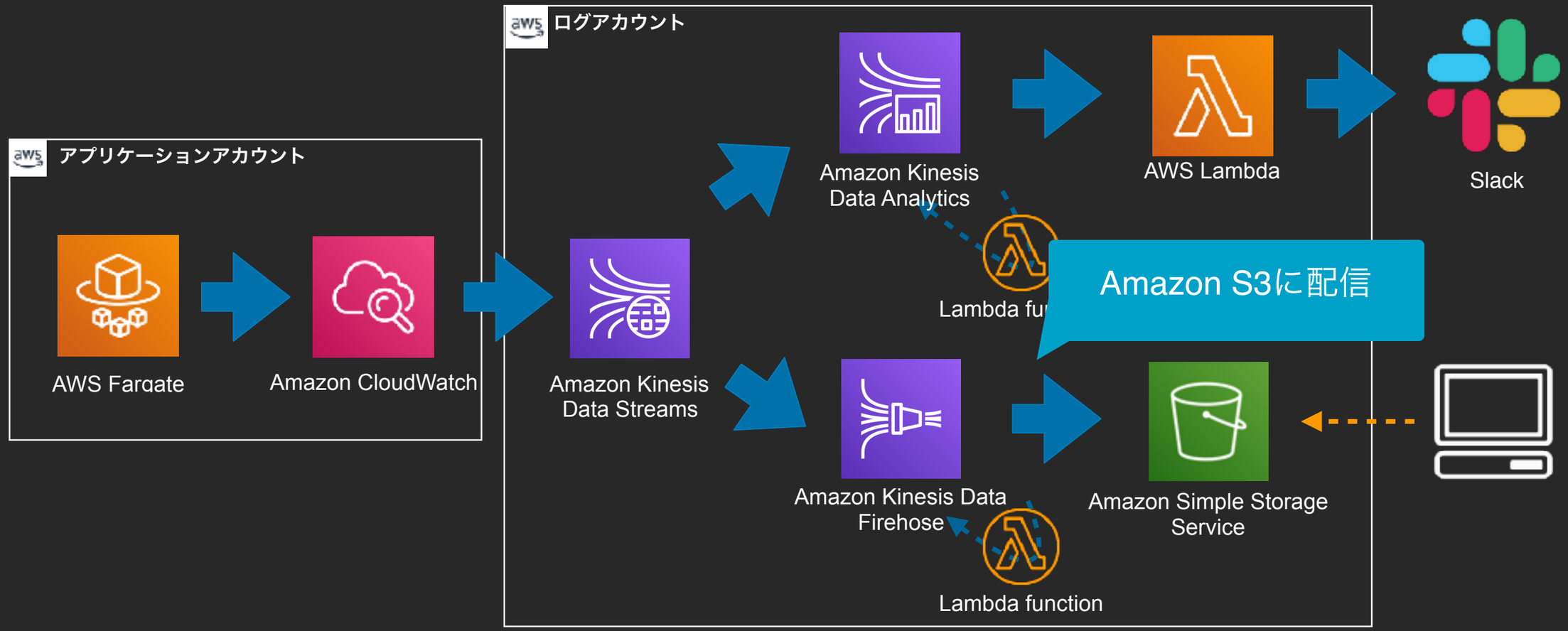
Fargateのパターン

- awslogsログドライバを活用
- 標準出力をCloudWatch Logsに送信



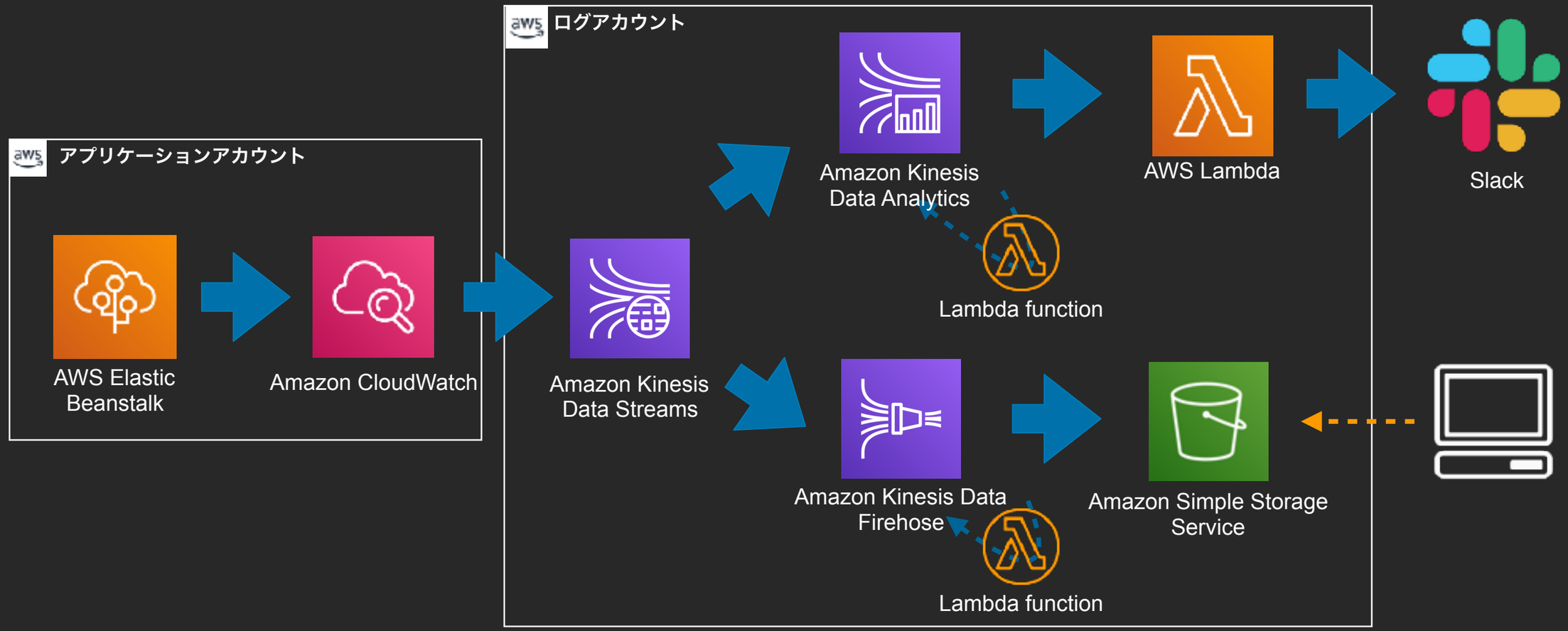
Fargateのパターン

- awslogsログドライバを活用
- 標準出力をCloudWatch Logsに送信



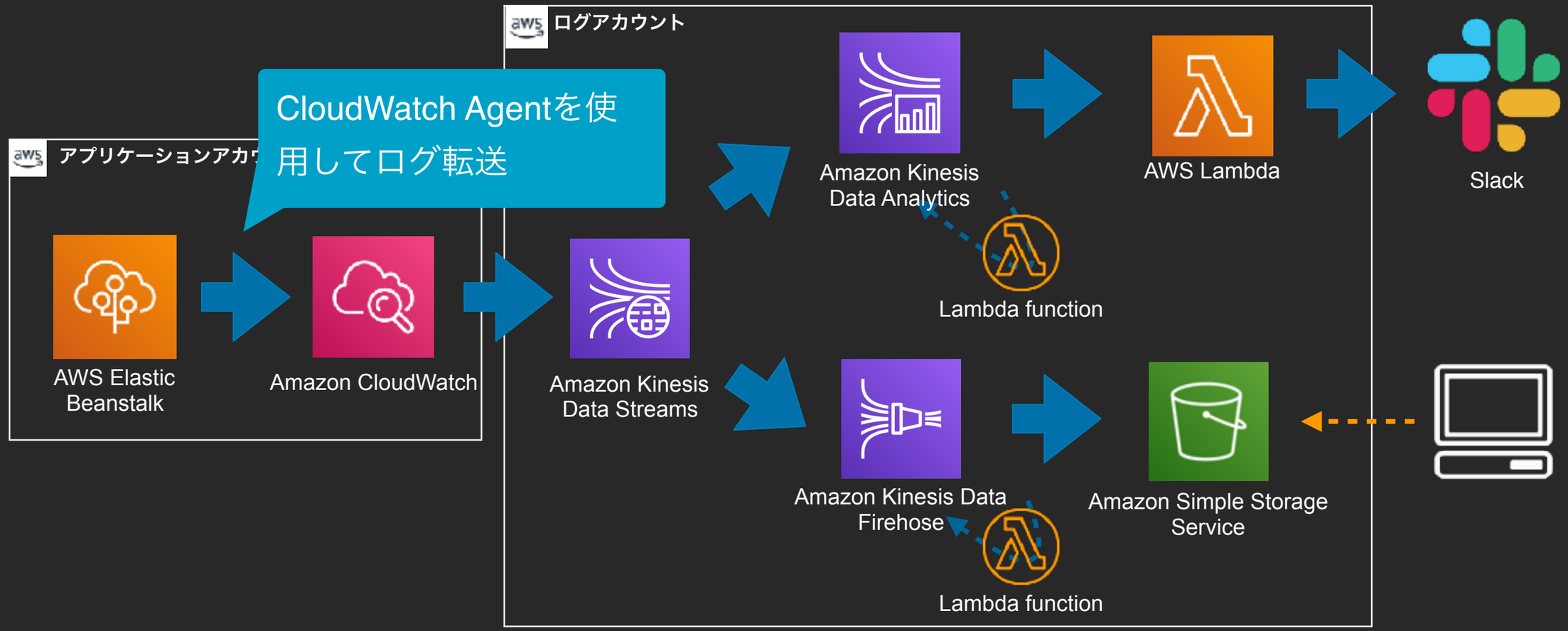
ElasticBeanstalkのパターン

- CloudWatchAgentを使う



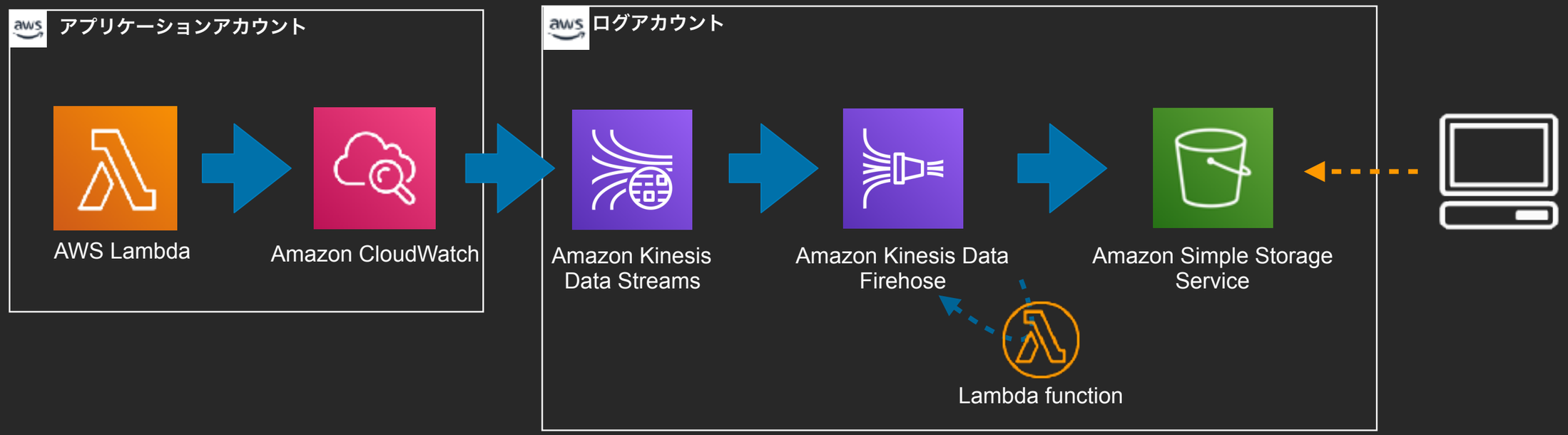
ElasticBeanstalkのパターン

- CloudWatchAgentを使う



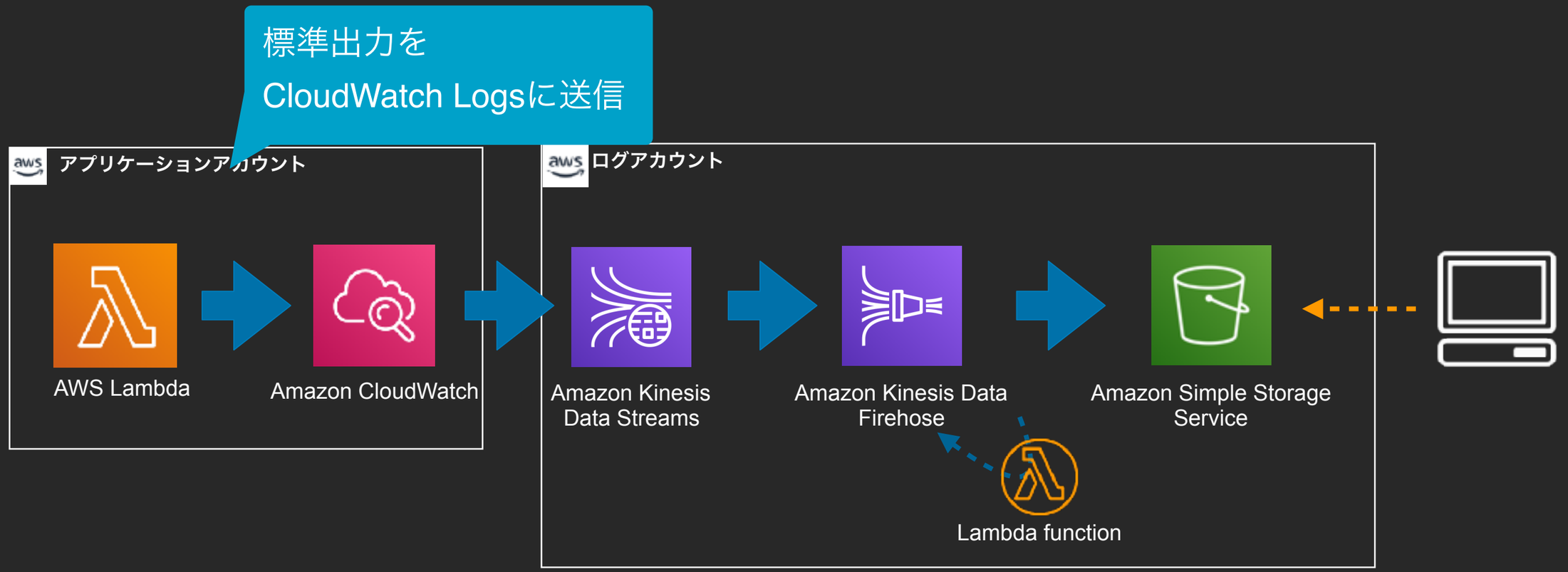
Lambdaのパターン

- Lambdaの標準出力はCloudWatch Logs



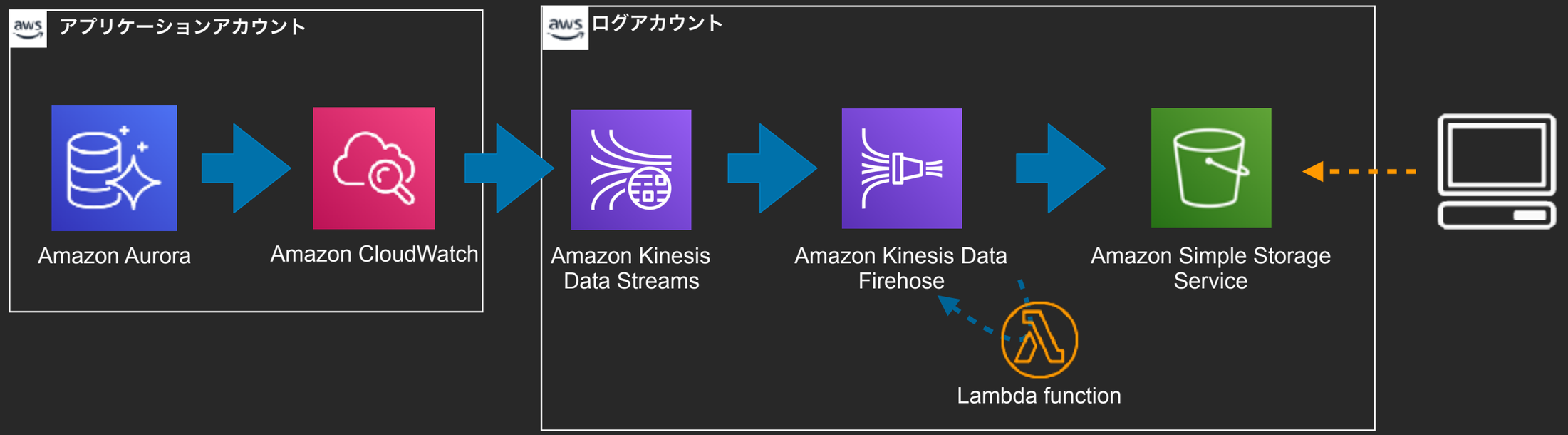
Lambdaのパターン

- Lambdaの標準出力はCloudWatch Logs



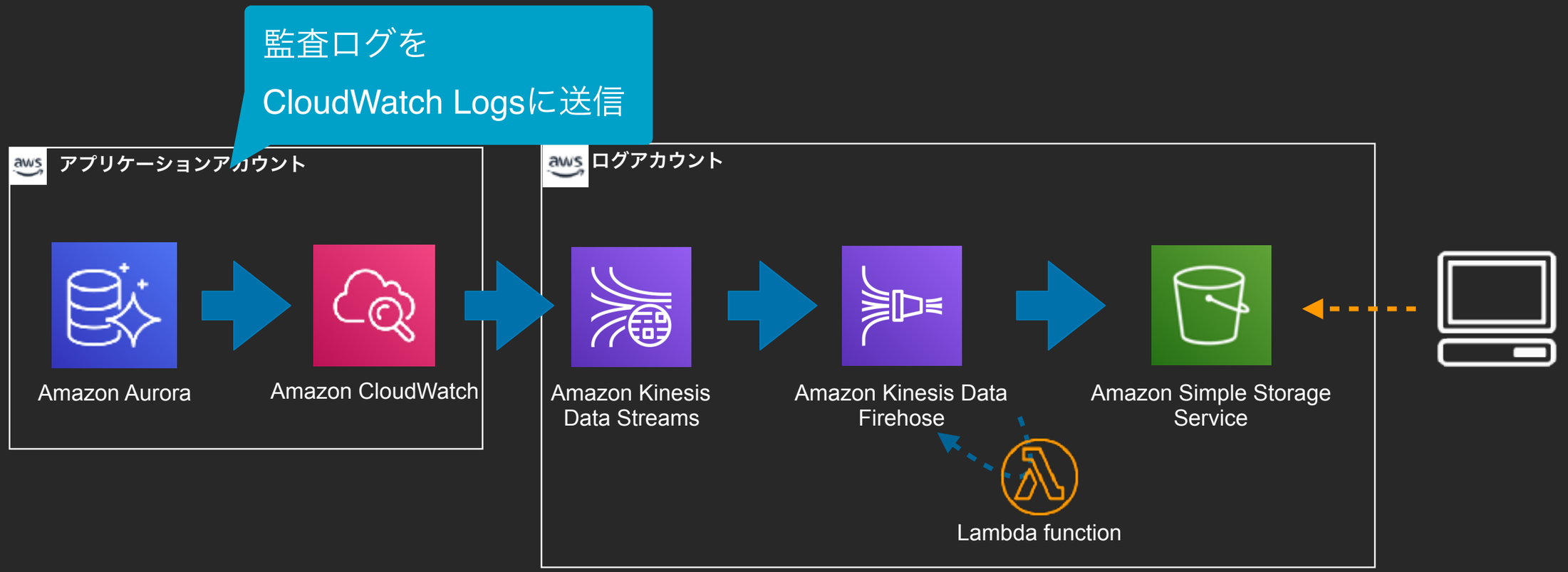
Auroraのパターン

- 監査ログをONにするとCloudWatch Logsに出力される
- Kinesis Data Firehose内でLambdaを呼び出しJSONにパース



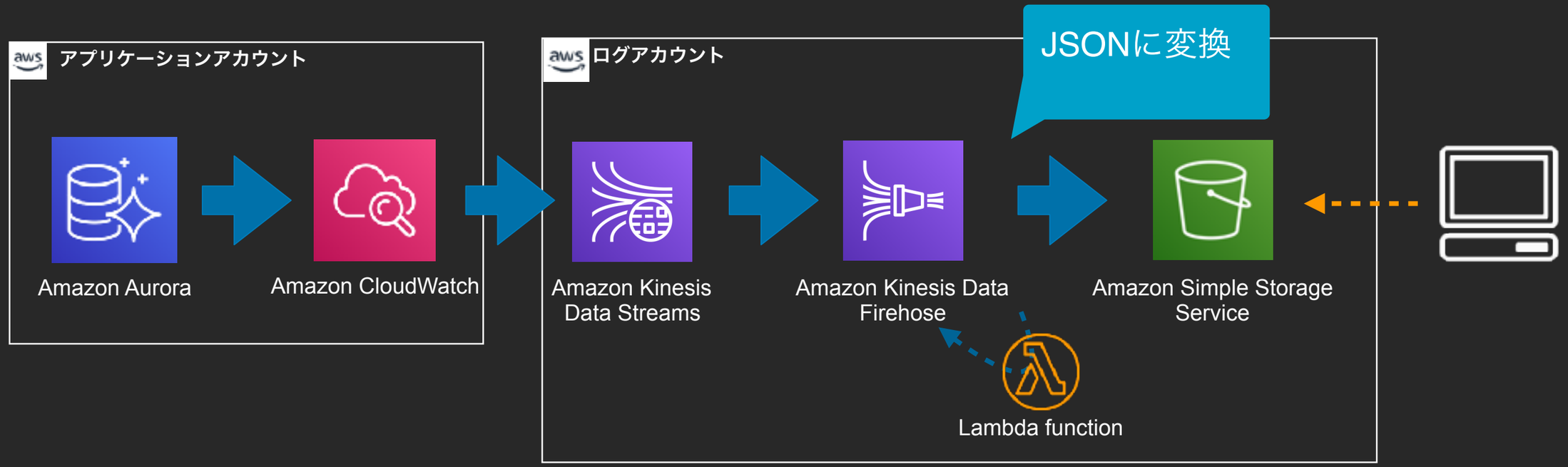
Auroraのパターン

- 監査ログをONにするとCloudWatch Logsに出力される
- Kinesis Data Firehose内でLambdaを呼び出しJSONにパース



Auroraのパターン

- 監査ログをONにするとCloudWatch Logsに出力される
- Kinesis Data Firehose内でLambdaを呼び出しJSONにパース



CloudWatch Logs から Kinesis 連携時のTips

- ログイベントがCloudWatch Logsから転送される際まとめられるので必要に応じて分解する
- データがbase64エンコードとgzip圧縮されている
- LambdaのBluePrintが活用できる

CloudWatch Logs から Kinesis 連携時のTips

- ログイベントがCloudWatch Logsから転送される際まとめられるので必要に応じて分解する
- データがbase64エンコードとgzip圧縮されている
- LambdaのBluePrintが活用できる

ログイベントに複数入ってくる

AWS CloudFormationによる環境構築

- Infrastructure as a Code(IaC)の実現
- 環境の再現性
  - マルチアカウントへのデプロイの柔軟さ
  - 管理のしやすさ
  - 変更のトラッキング
  - 運用負担の軽減
  - ヒューマンエラーの回避

AWS CloudFormationによる環境構築

- Infrastructure as a Code(IaC)の実現
- 環境の再現性
  - マルチアカウントへのデプロイの柔軟さ
  - 管理のしやすさ
  - 変更のトラッキング
  - 運用負担の軽減
  - ヒューマンエラーの回避

NoOps化！

CloudFormationがなかった場合・・・

CloudFormationがなかった場合・・・

構築に時間がかかる・・・

CloudFormationがなかった場合・・・

構築に時間がかかる・・・

全部手作業・・・

CloudFormationがなかった場合・・・

構築に時間がかかる・・・

全部手作業・・・

より詳細な手順書が
必要
ドキュメントも大量に
必要・・・

CloudFormationがなかった場合・・・

構築に時間がかかる・・・

全部手作業・・・

より詳細な手順書が
必要
ドキュメントも大量に
必要・・・

構成管理
できない・・・

CloudFormationで全部解決！

構築に時間がかる・・・

全部手・・・

より詳細な申請書が

ドキュメントも大量に

必要・・・

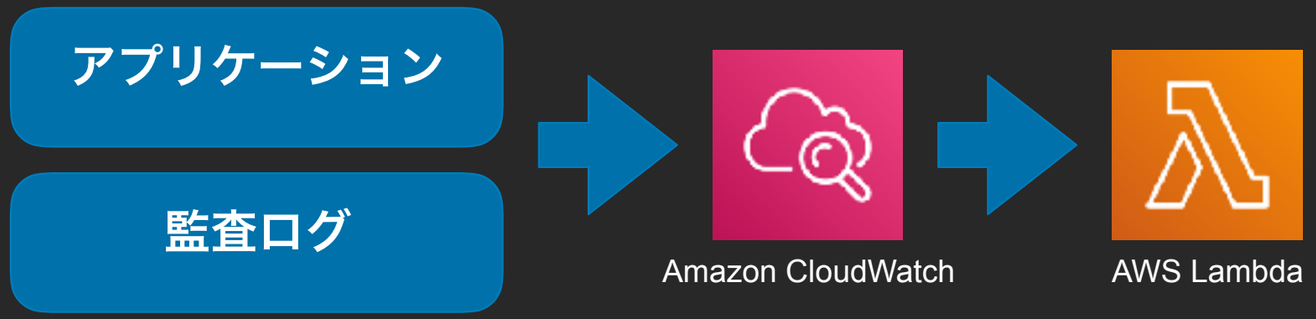
権限管理

できな・・・

移行時

- CloudFormationを利用することで手作業を最小限にした
- サブスクリプションフィルターを切り替えるだけでリリースできたのでスムーズに移行ができた
- 本番影響が無かった

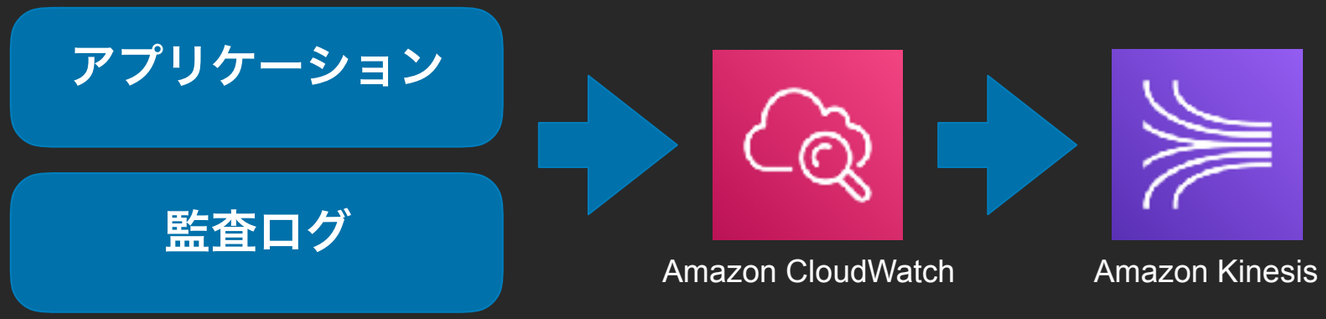
[Before]



移行時

- CloudFormationを利用することで手作業を最小限にした
- サブスクリプションフィルターを切り替えるだけでリリースできたのでスムーズに移行ができた
- 本番影響が無かった

[After]



ログ基盤の導入により改善されたこと

Amazon S3にログを保管しログ分析が可能に

- アプリケーション性能の調査
- エラー発生時の原因ログ調査
- ログを必要とする問い合わせ時の調査

監査要件への対応

- 監査業務にかかる時間が短縮できた

高度なリアルタイムアラートが可能に

- SIEMサービスと連携ができ、リアルタイムで検知ができるようになった

+α NoOpsの実現

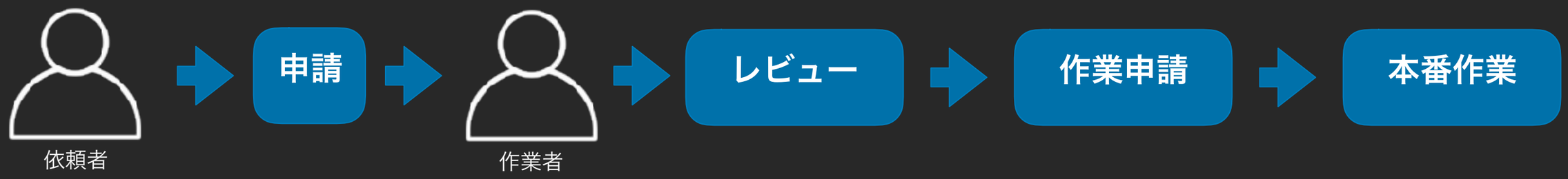
- マネージドサービスとサーバーレス化による運用軽減

新しく見えてきた課題

CloudFormationのデプロイは人間が行う

- ログの連携まで多少の待ちが発生してしまう
- 人間の本番作業をできるだけ減らしたい

現在の申請ステップ



新しく見えてきた課題

CloudFormationのデプロイは人が行う

- ログの連携まで多少の待ちが発生してしまう
- 人間の本番作業をできるだけ減らしたい

🚀🚀🚀 爆速にデリバリーしたい🚀🚀🚀

新しく見えてきた課題

CloudFormationのデプロイは人が行う

- ログの連携まで多少の待ちが発生してしまう
- 人間の本番作業をできるだけ減らしたい

🚀🚀🚀 **よろしい、ならば自動化だ** 🚀🚀🚀

今後の取り組み

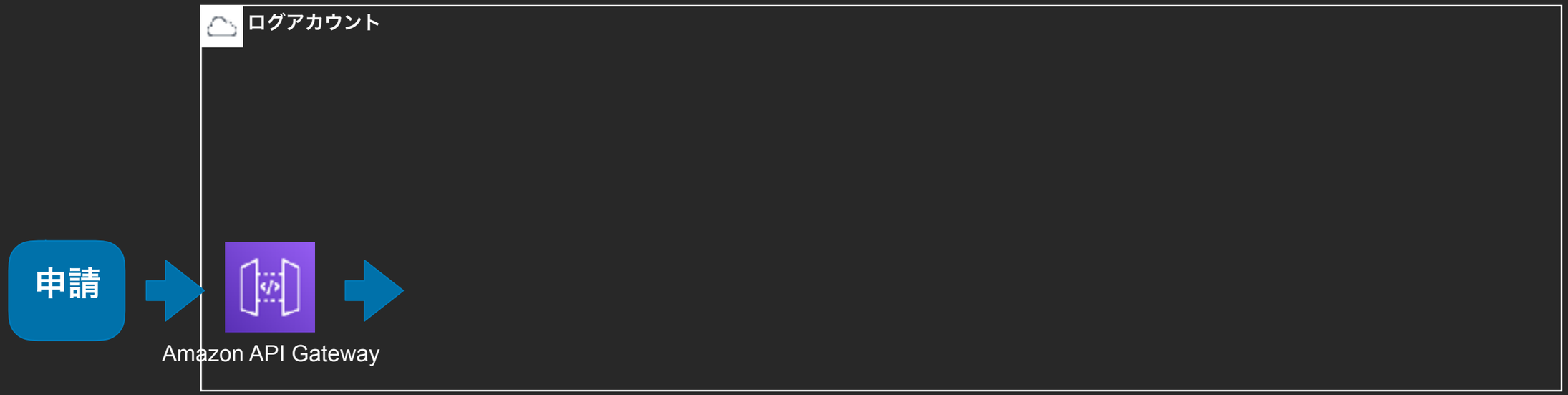
連携の仕組みを自動化し、CloudFormationのデプロイに人を介さない

今後の取り組み

連携の仕組みを自動化し、CloudFormationのデプロイに人を介さない

今後の取り組み

連携の仕組みを自動化し、CloudFormationのデプロイに人を介さない

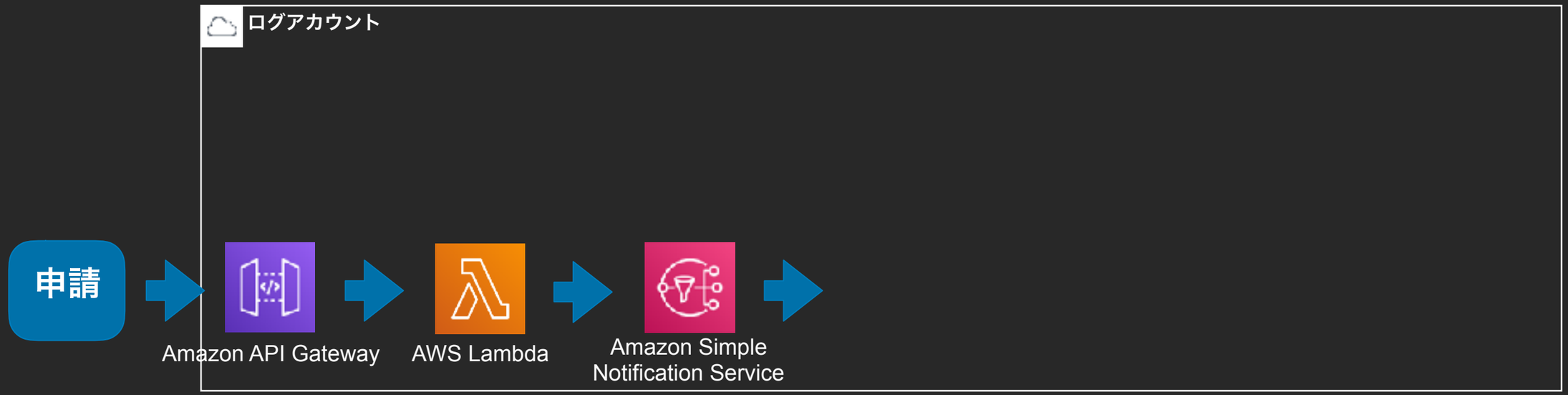


今後の取り組み

連携の仕組みを自動化し、CloudFormationのデプロイに人を介さない

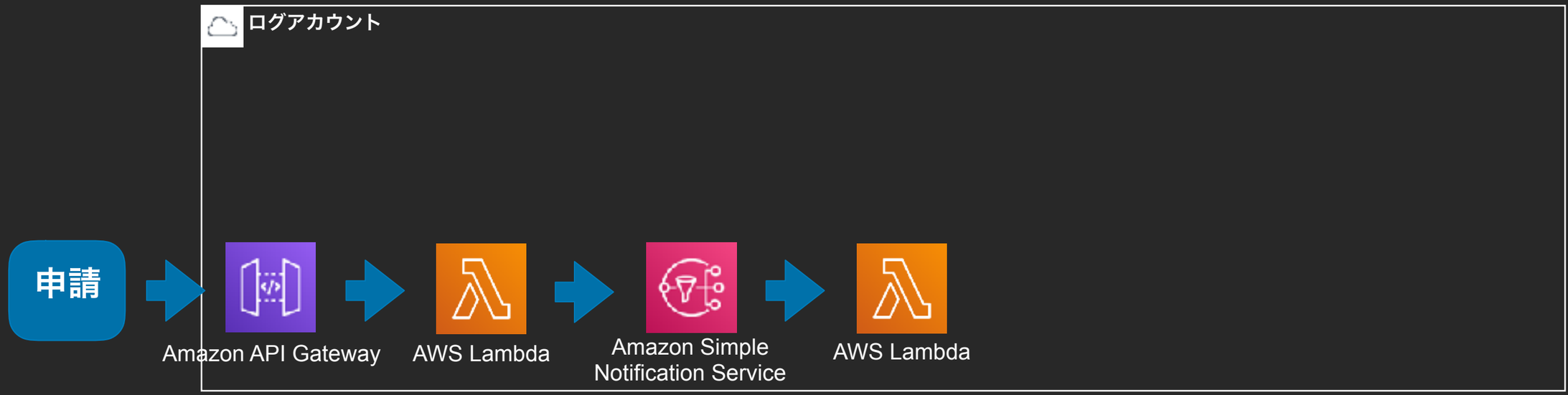
今後の取り組み

連携の仕組みを自動化し、CloudFormationのデプロイに人を介さない



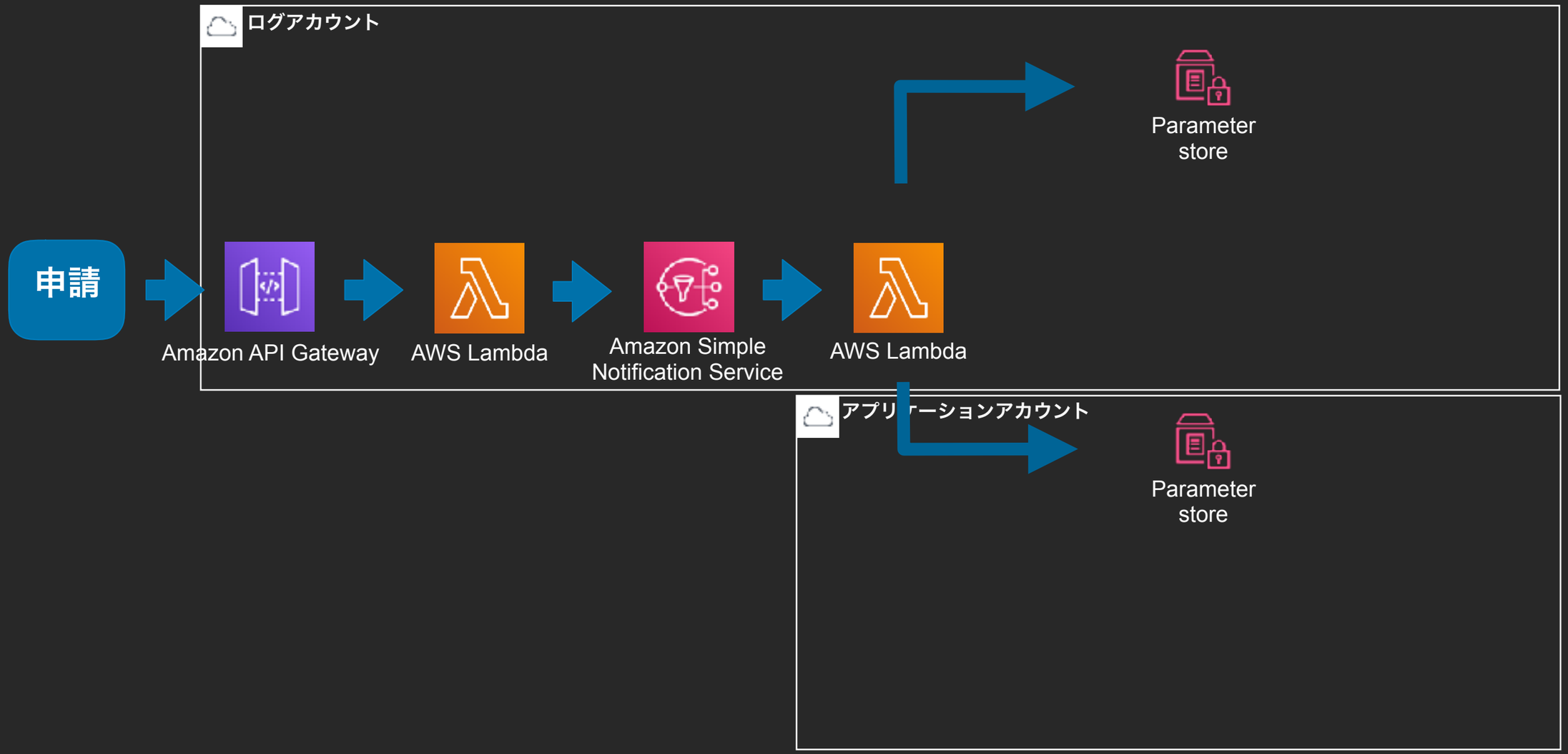
今後の取り組み

連携の仕組みを自動化し、CloudFormationのデプロイに人を介さない



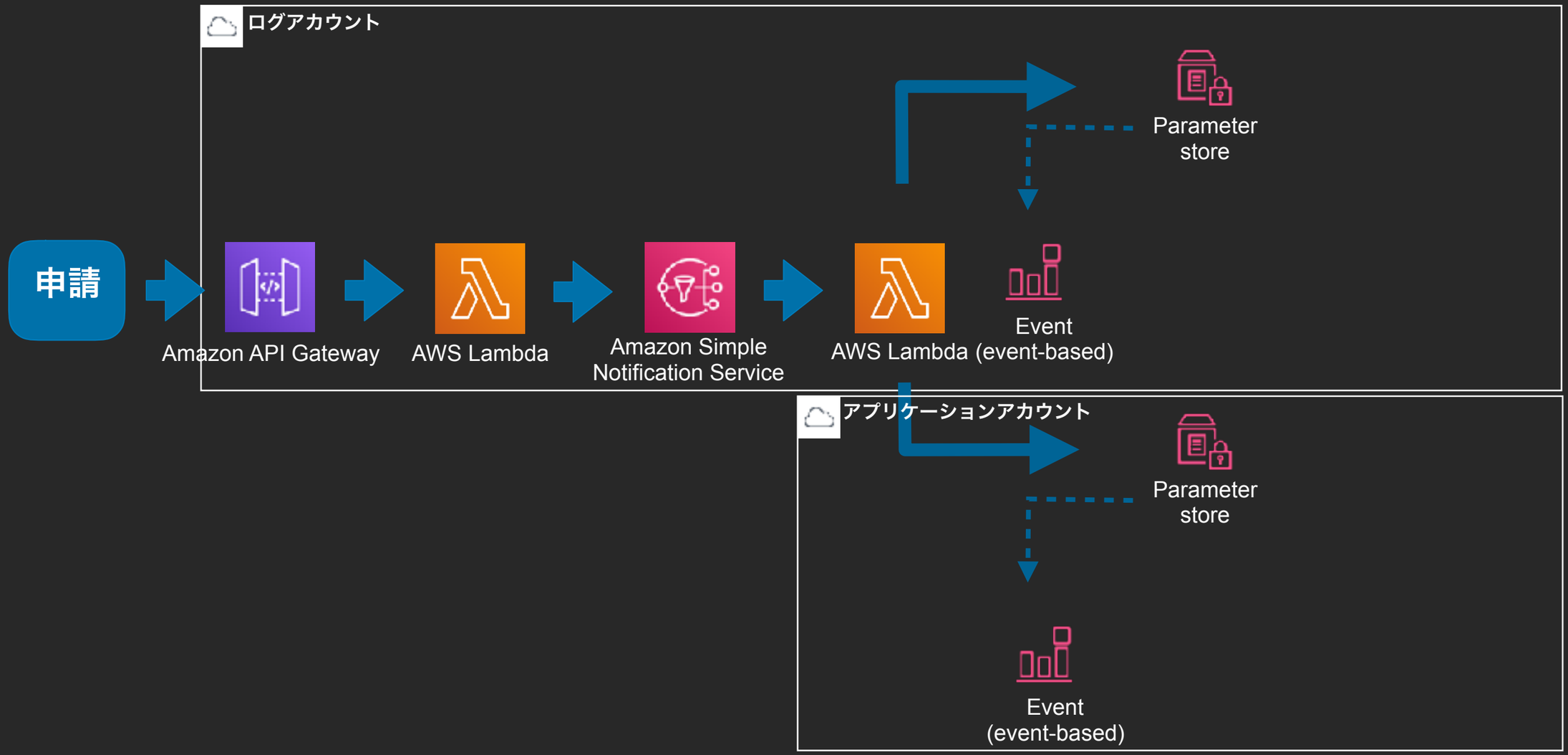
今後の取り組み

連携の仕組みを自動化し、CloudFormationのデプロイに人を介さない



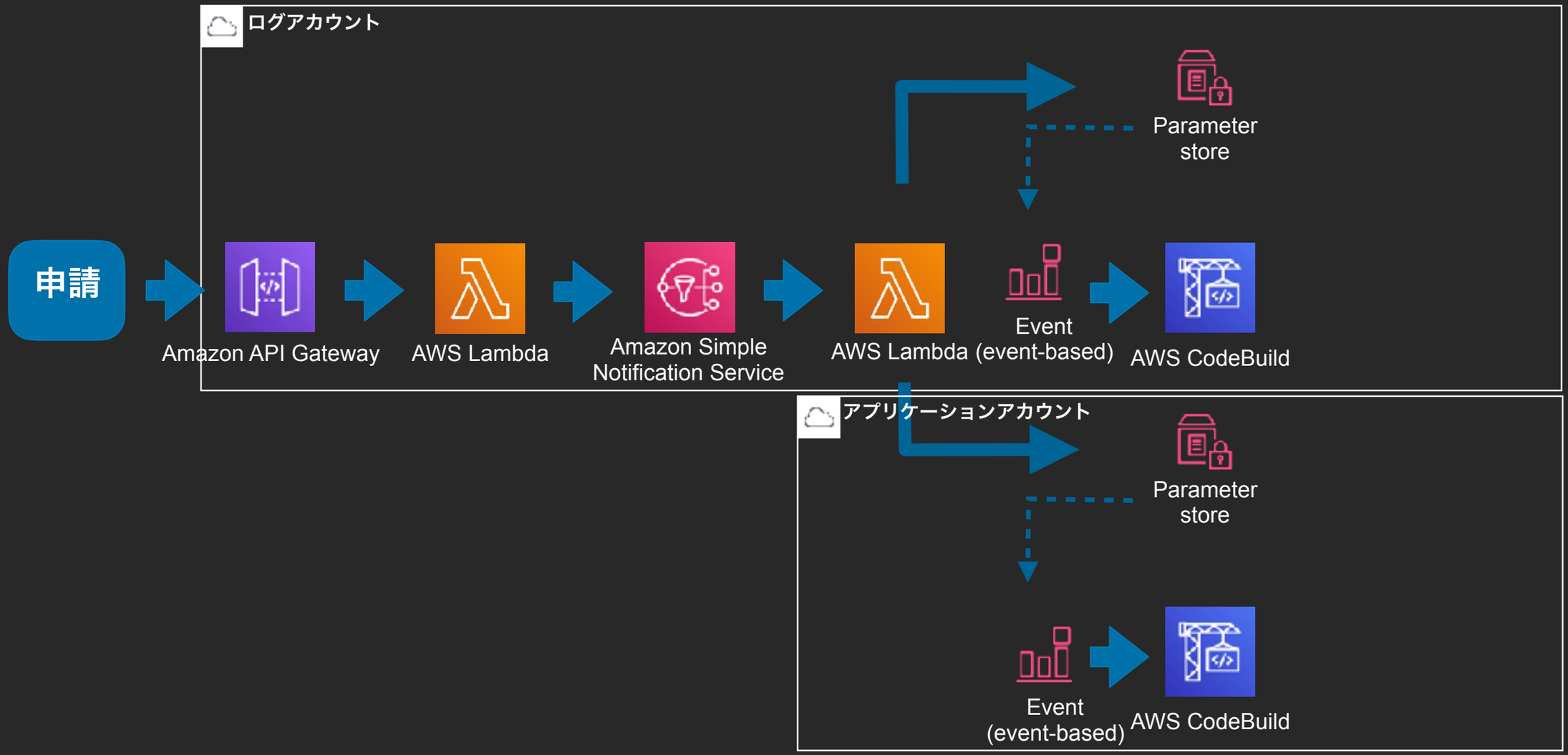
今後の取り組み

連携の仕組みを自動化し、CloudFormationのデプロイに人を介さない



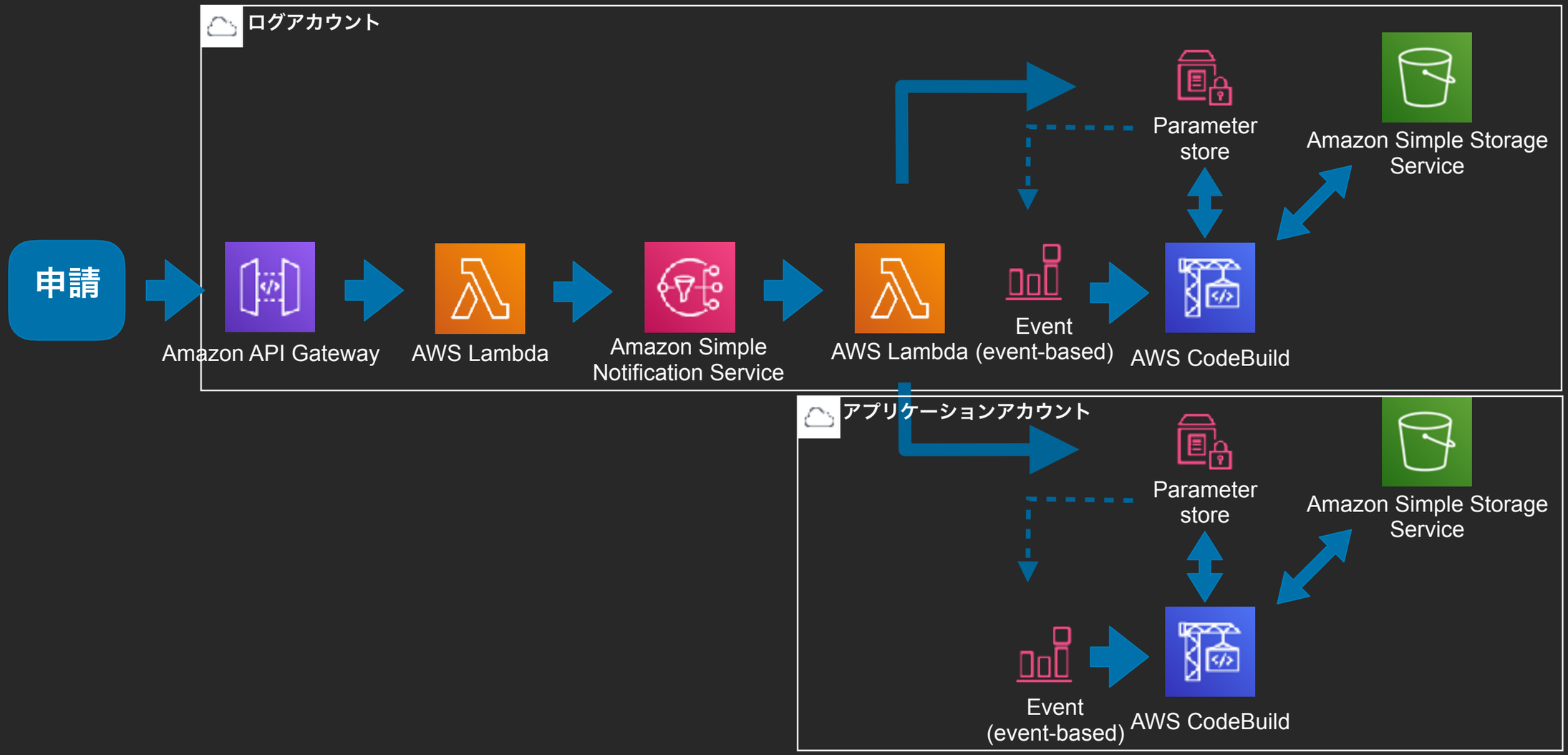
今後の取り組み

連携の仕組みを自動化し、CloudFormationのデプロイに人を介さない



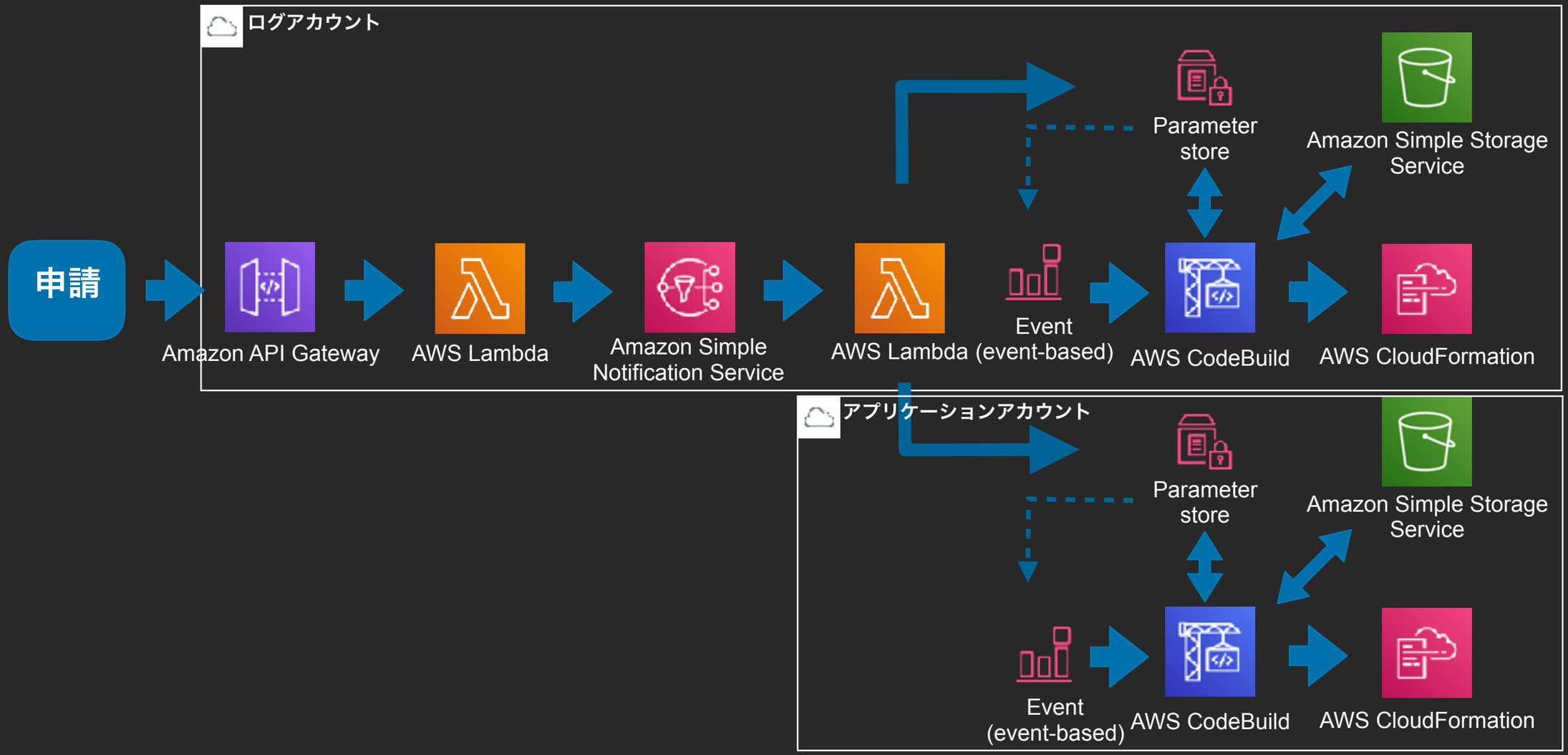
今後の取り組み

連携の仕組みを自動化し、CloudFormationのデプロイに人を介さない



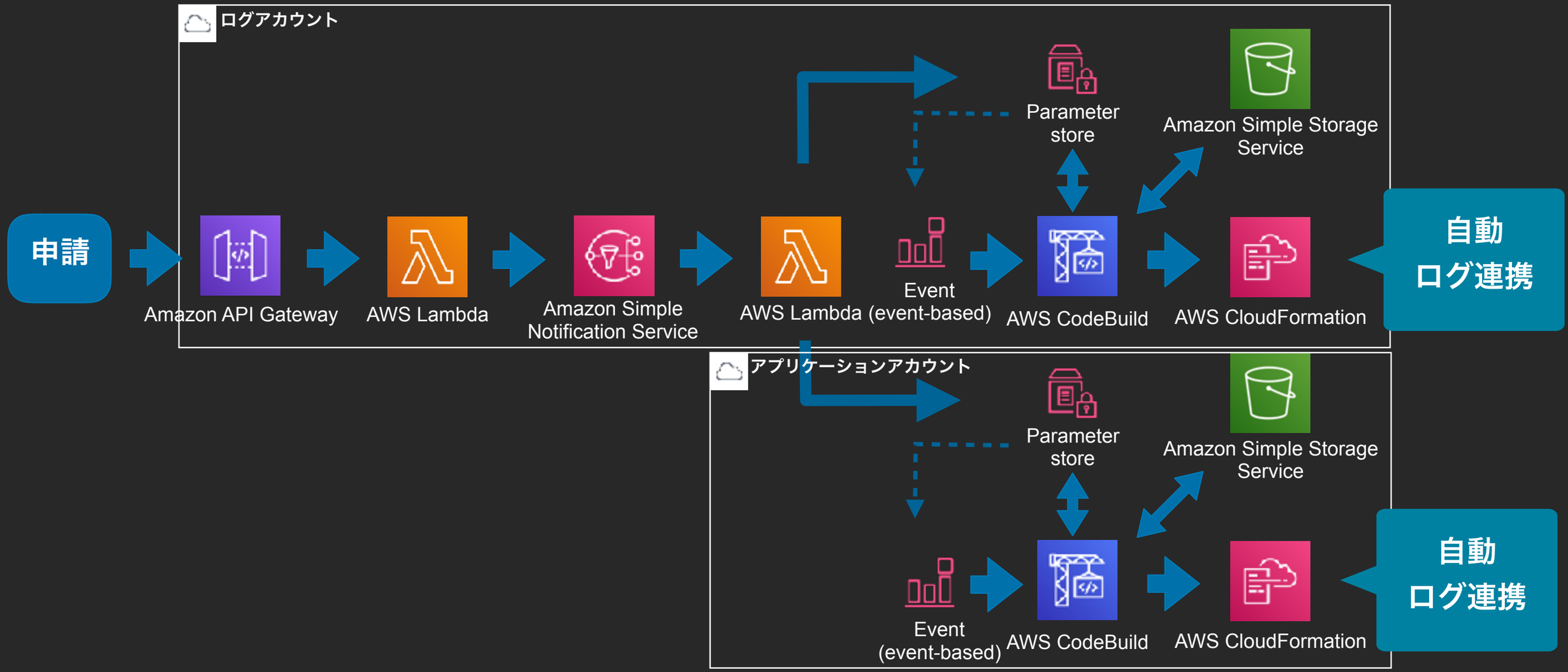
今後の取り組み

連携の仕組みを自動化し、CloudFormationのデプロイに人を介さない



今後の取り組み

連携の仕組みを自動化し、CloudFormationのデプロイに人を介さない



今後の取り組み

連携の仕組みを自動化し、CloudFormationのデプロイに人を介さない

まとめ

Kinesis

- 基盤にKinesis+Lambdaを活用
  - 既存の環境を壊さず移行できた
  - 拡張性が広がった

IaCの話

- CloudFormationを使うことで運用負荷を下げた
- NoOps化の実現

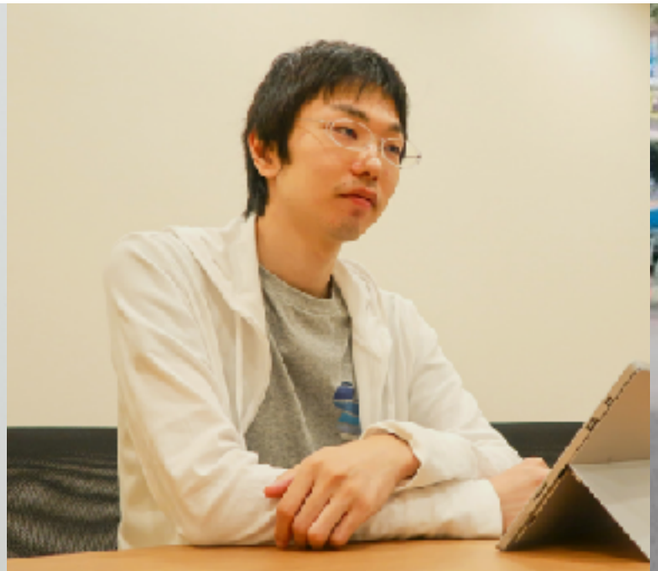
移行による新しい課題と今後の取り組み
についてお話ししました。



We Are Hiring!

ビットバンク 採用

検 索


Thank you!

ビットバンク株式会社 谷津 香

@mdps513

aws SUMMIT ONLINE

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.