ビットバンクにおける Amazon Redshift の活用事例 ~ Amazon SageMaker と Redshift Data API を添えて ~

ビットバンク株式会社
Platform部 エンジニア 谷津 香
Platform部 エンジニア 加藤 雅行
Platform部 エンジニア 長尾 康志

bitbank

会社紹介

事業概要

ビットバンク株式会社は日本でTop3（預り資産）に入る暗号資産取引所を運営している暗号資産交換業者

（関東財務局長：登録番号第00004号）

主要事業

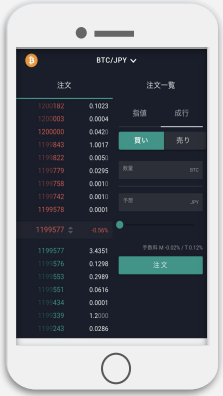
bitbank.cc
暗号資産取引

bitbank MARKETS
暗号資産マーケットの情報サイト

「bitbank.cc」のサービス概要

bitbank

- 暗号資産取引所
- 暗号資産販売所
- 貸して増やす（レンディング）



取り扱い通貨（27銘柄）

BTC, LTC, XRP, ETH, MONA, XLM, BCC (BCH), BAT, QTUM, OMG, XYM, LINK, MKR, BOBA, ENJ, MATIC, DOGE, DOT, ASTR, ADA, AVAX, AXS, FLR, SAND

本日の内容

1. Amazon Redshiftを用いたデータの活用戦略について
2. Amazon Redshiftを利用したAmazon SageMaker環境構築
3. Amazon Redshift Data APIを利用した顧客へのデータ提供

Amazon Redshiftを用いたデータの活用戦略について

自己紹介

谷津香(やつ かおり)

ビットバンク株式会社

システム部門 プラットフォーム部

データチーム エンジニア

@mdps513

去年、この場でこんな話をしました

なぜビットバンクは
Amazon Redshiftを
選んだのか？

ビットバンク株式会社
システム部門 プラットフォーム部 谷津 香

あれから一年・・・
実際のところどうだったの？

Redshiftを利用開始して良かった点と微妙だった点が見えてきた

良かった点

- クエリ速度が劇的に改善した
- チューニングが今の所不要
- DataSharingが利用しやすい
- Redshiftの新機能が継続的に登場
  - Redshift Serverlessなど

微妙だった点

- Aurora MySQLとの互換性や仕様の問題に引っかかる
  - FederatedQuery
  - DMS
- 割と高頻度にメンテナンスがある

良かった点

クエリ速度が劇的に改善した

- 1781万行の取得にかかった時間が約2分
- 以前の場合...
  - 数時間〜十数時間程度かかっていた

良かった点

クエリ速度が劇的に改善した

- 1781万行の取得にかかった時間が約2分
- 以前の場合...
  - 数時間〜十数時間程度かかっていた

微妙な点 ~Aurora MySQLとの互換性や仕様の問題~

HLL(History List Length)が発生

- Aurora mysql(InnoDBのエンジン)はインスタンス単位ではなくクラスタ単位で共有ストレージを持っている性質上、レコード数が巨大なテーブルを全件スキャンするような操作が走るとCPU負荷が少しずつ上昇する現象
- Aurora MySQL の RollbackSegmentHistoryListLength(undo ログレコード)の上昇により検知ができる

微妙な点 ~Aurora MySQLとの互換性や仕様の問題~

HLL(History List Length)が発生

- Aurora mysql(InnoDBのエンジン)はインスタンス単位ではなくクラスタ単位で共有ストレージを持っている性質上、レコード数が巨大なテーブルを全件スキャンするような操作が走るとCPU負荷が少しずつ上昇する現象
- Aurora MySQL の RollbackSegmentHistoryListLength(undo ログレコード)の上昇により検知ができる

→FederatedQueryやDMSにてAuroraに対して大規模SELECTを行なうと発生する可能性があるため対策が別途必要

微妙な点

割と高頻度にメンテナンスがある

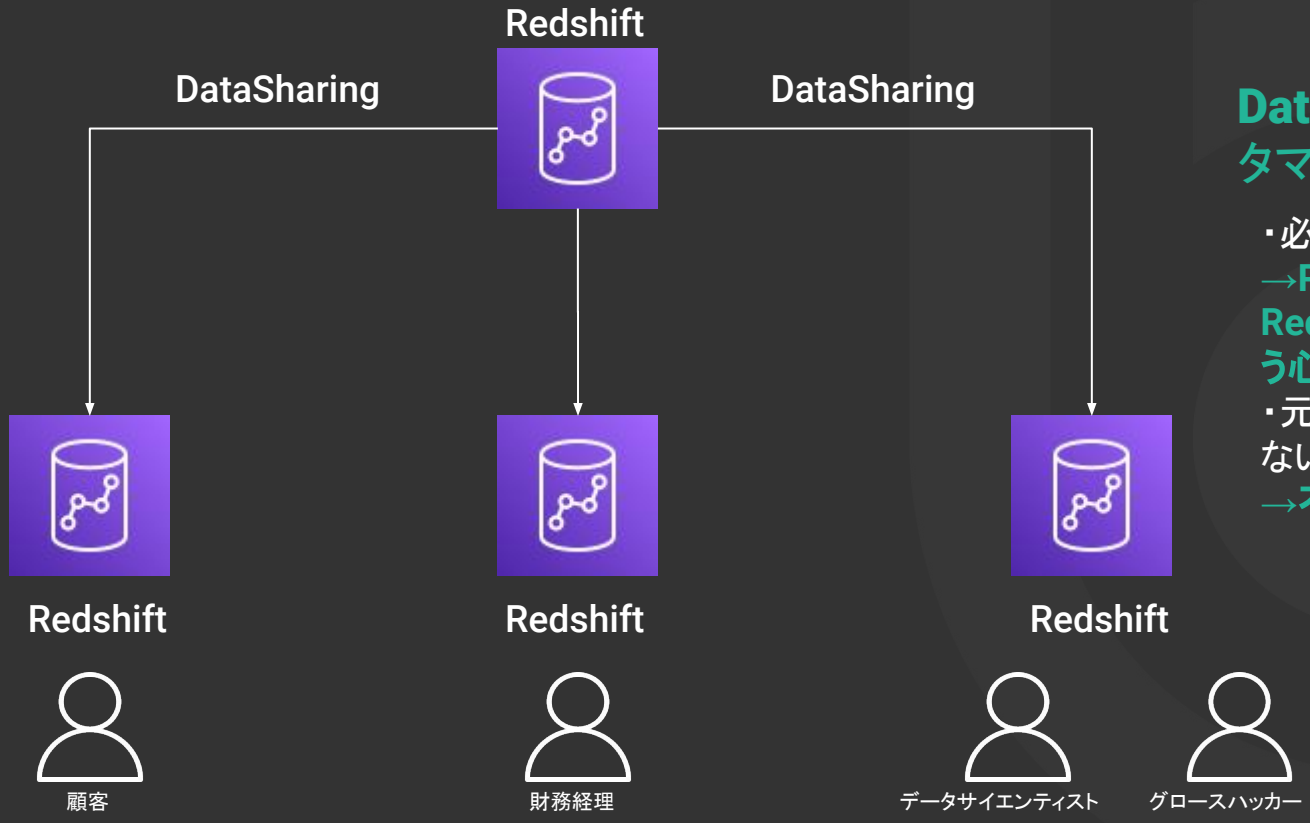
- 事前のイベント通知&メンテナンスの延期で予期しないダウンタイムを回避
  - 月一でまとめてメンテナンス対応
- 今後serverless化余地があるかも検討

検知&メンテの仕組み

結論: 割とイケてるが、改善の余地あり
(AWSさんよろしくお願いします🙏)

実際どんな感じに活用してるの？

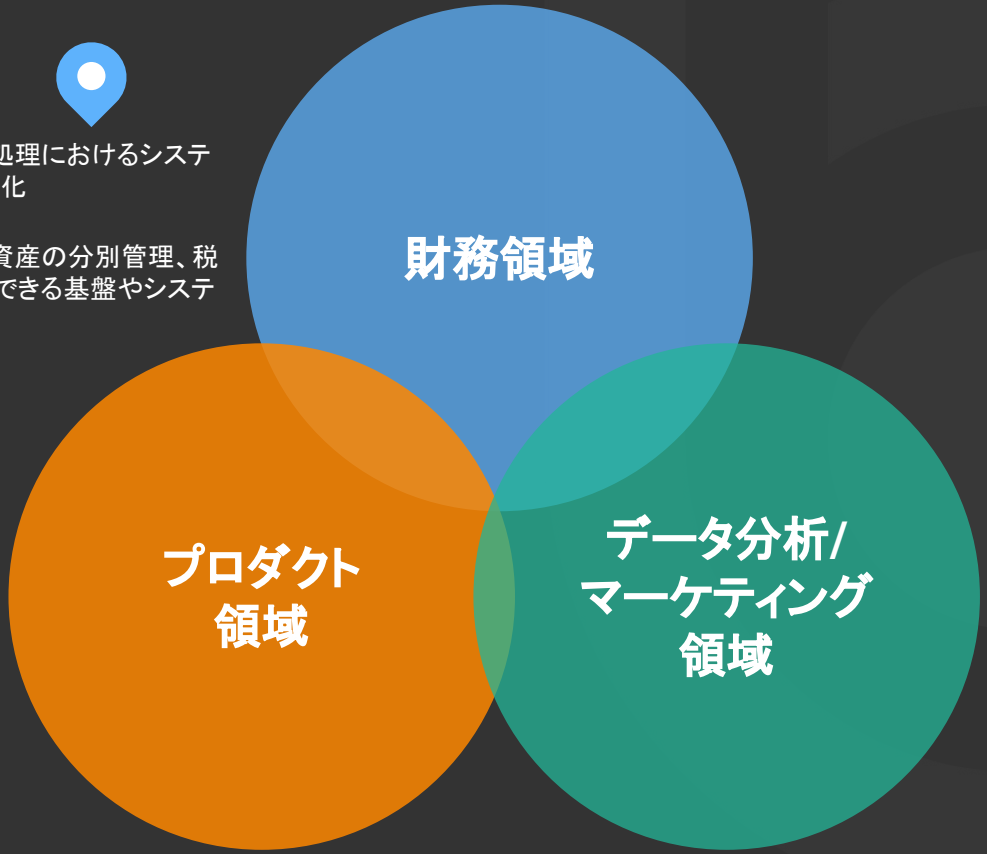
bitbankにおけるRedshiftの活用



Datasharing機能をデータマート的に活用

・必要なデータのみ共有できる
→PIIなどのデータが仮に大元のRedshiftに連携されても見えてしまう心配がない
・元データを書き換えられる心配がない
→不正が起きにくい

bitbankにおけるRedshiftを中心としたデータ活用の軸



財務領域

目的

・暗号資産の財務処理におけるシステム整備や業務効率化

取り組み

・財務処理や暗号資産の分別管理、税制対応などに対応できる基盤やシステムの構築

プロダクト領域

目的

・データ基盤を用いた顧客への機能開発や運用

取り組み

・取引履歴及び約定履歴のデータ抽出機能の提供

データ分析/マーケティング領域

目的

・データサイエンティストやマーケターが利用するデータ基盤やデータの整備

取り組み

・データサイエンティストが利用する高度な分析およびML基盤の提供

・データサイエンティスト、グロースハッカーが利用するデータの整備

bitbankにおけるRedshiftを中心としたデータ活用の軸

財務領域
目的
・暗号資産の財務処理におけるシステム整備や業務効率化
取り組み
・財務処理や暗号資産の分別管理、税制対応などに対応できる基盤やシステムの構築

データ分析/マーケティング領域
目的
・データサイエンティストやマーケターが利用するデータ基盤やデータの整備
取り組み
・データサイエンティストが利用する高度な分析およびML基盤の提供
・データサイエンティスト、グロースハッカーが利用するデータの整備

プロダクト領域
目的
・データ基盤を用いた顧客への機能開発や運用
取り組み
・取引履歴及び約定履歴のデータ抽出機能の提供

bitbankにおけるRedshiftを中心としたデータ活用の軸

財務領域
目的
・暗号資産の財務処理におけるシステム整備や業務効率化
取り組み
・財務処理や暗号資産の分別管理、税制対応などに対応できる基盤やシステムの構築

データ分析/マーケティング領域
目的
・データサイエンティストやマーケターが利用するデータ基盤やデータの整備
取り組み
・データサイエンティストが利用する高度な分析およびML基盤の提供
・データサイエンティスト、グロースハッカーが利用するデータの整備

プロダクト領域
目的
・データ基盤を用いた顧客への機能開発や運用
取り組み
・取引履歴及び約定履歴のデータ抽出機能の提供

bitbankにおけるRedshiftを中心としたデータ活用の軸

財務領域
目的
・暗号資産の財務処理におけるシステム整備や業務効率化
取り組み
・財務処理や暗号資産の分別管理、税制対応などに対応できる基盤やシステムの構築

データ分析/マーケティング領域
目的
・データサイエンティストやマーケターが利用するデータ基盤やデータの整備
取り組み
・データサイエンティストが利用する高度な分析およびML基盤の提供
・データサイエンティスト、グロースハッカーが利用するデータの整備

プロダクト領域
目的
・データ基盤を用いた顧客への機能開発や運用
取り組み
・取引履歴及び約定履歴のデータ抽出機能の提供

bitbankにおけるRedshiftを中心としたデータ活用の軸

財務領域
目的
・暗号資産の財務処理におけるシステム整備や業務効率化
取り組み
・財務処理や暗号資産の分別管理、税制対応などに対応できる基盤やシステムの構築

Amazon Redshiftを利用したAmazon SageMaker環境構築

データ分析/マーケティング領域
目的
・データサイエンティストやマーケターが利用するデータ基盤やデータの整備
取り組み
・データサイエンティストが利用する高度な分析およびML基盤の提供
・データサイエンティスト、グロースハッカーが利用するデータの整備

プロダクト領域
目的
・データ基盤を用いた顧客への機能開発や運用
取り組み
・取引履歴及び約定履歴のデータ抽出機能の提供

bitbankにおけるRedshiftを中心としたデータ活用の軸

財務領域
目的
・暗号資産の財務処理におけるシステム整備や業務効率化
取り組み
・財務処理や暗号資産の分別管理、税制対応などに対応できる基盤やシステムの構築

Amazon Redshiftを利用したAmazon SageMaker環境構築

データ分析/マーケティング領域
目的
・データサイエンティストやマーケターが利用するデータ基盤やデータの整備
取り組み
・データサイエンティストが利用する高度な分析およびML基盤の提供
・データサイエンティスト、グロースハッカーが利用するデータの整備

プロダクト領域
目的
・データ基盤を用いた顧客への機能開発や運用
取り組み
・取引履歴及び約定履歴のデータ抽出機能の提供

bitbankにおけるRedshiftを中心としたデータ活用の軸

財務領域
目的
・暗号資産の財務処理におけるシステム整備や業務効率化
取り組み
・財務処理や暗号資産の分別管理、税[illegible]できる基盤やシステ

Amazon Redshift Data APIを利用した顧客へのデータ提供

プロダクト領域
目的
・データ基盤を用いた顧客への機能開発や運用
取り組み
・取引履歴及び約定履歴のデータ抽出機能の提供

Amazon Redshiftを利用したAmazon SageMaker環境構築

データ分析/マーケティング領域
目的
・データサイエンティストやマーケターが利用するデータ基盤やデータの整備
取り組み
・データサイエンティストが利用する高度な分析およびML基盤の提供
・データサイエンティスト、グロースハッカーが利用するデータの整備

Amazon Redshiftを利用した Amazon SageMaker環境構築

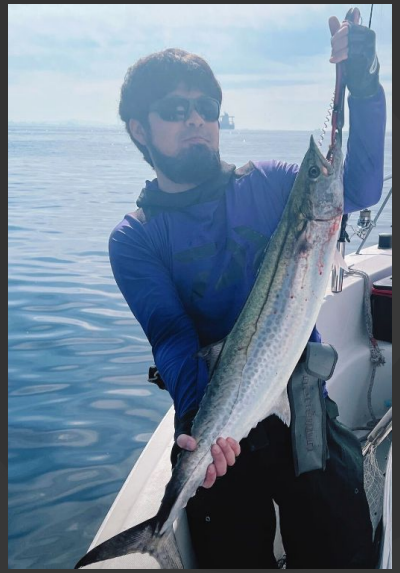
自己紹介

加藤 雅行 (かとう まさゆき)

ビットバンク株式会社

システム部門 プラットフォーム部

データチーム エンジニア



ここからの内容

話すこと

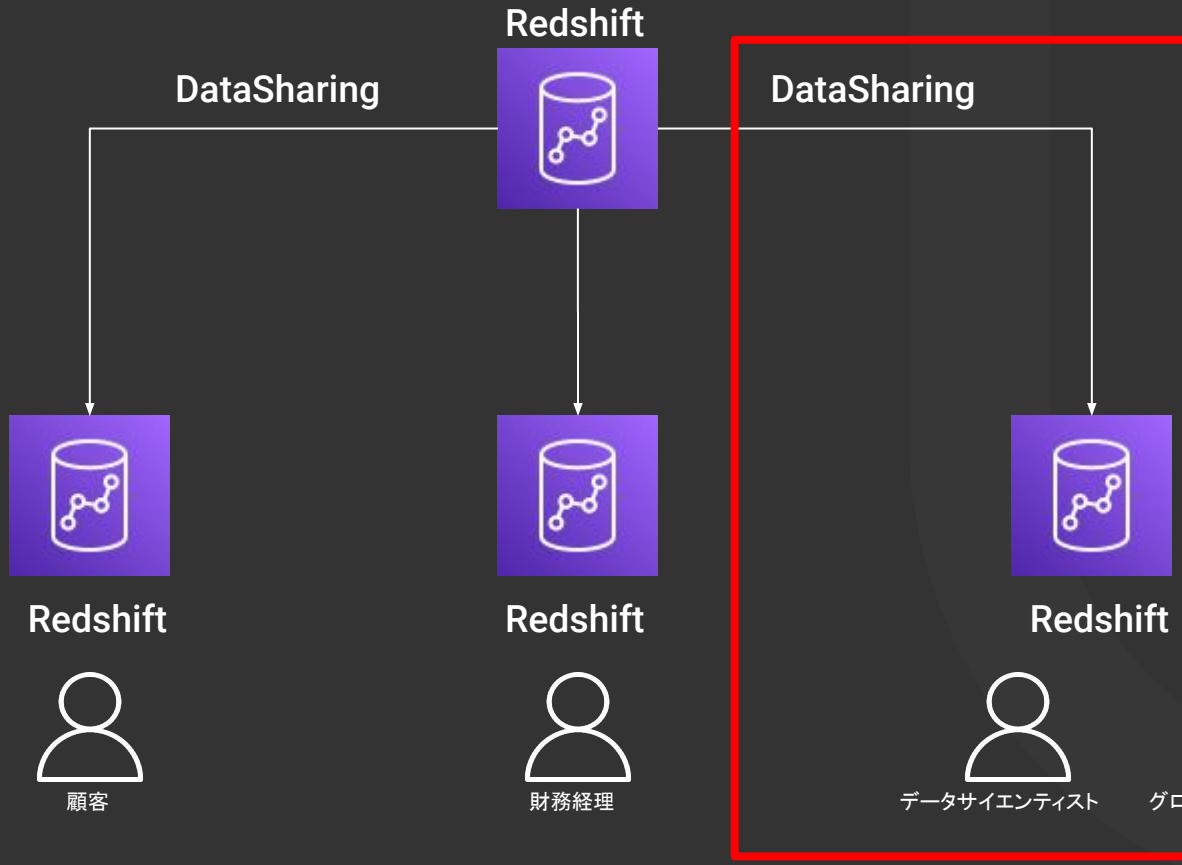
・Amazon Redshiftを利用したAmazon SageMaker環境構築の流れ

話さないこと

・ビットバンクが行なっている分析業務

ビットバンクのデータサイエンティストが
利用している分析環境についてお話しします

bitbankにおけるRedshiftの活用（再掲）

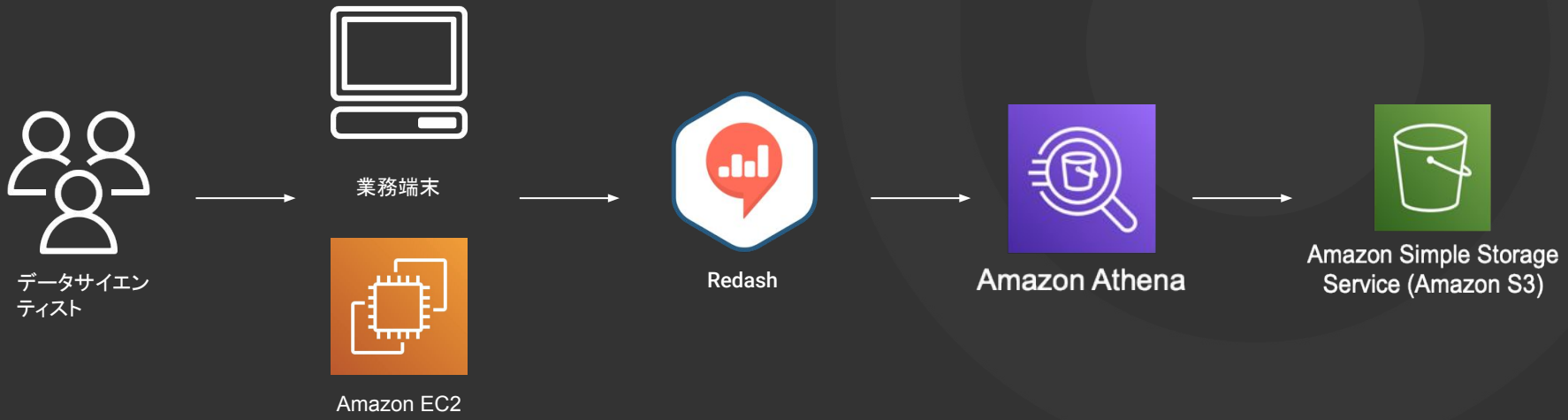


Datasharing機能をデータマート的に活用

・必要なデータのみ共有できる
→PIIなどのデータが仮に大元のRedshiftに連携されても見えてしまう心配がない
・元データを書き換えられる心配がない
→不正が起きにくい

従前のデータサイエンティストの分析環境

- 分析環境はローカル端末 or EC2でJupyter Notebookを起動していた
- 人によって自由にライブラリをカスタマイズしていた
- 大量データを扱うのにRedashからデータを取得していた



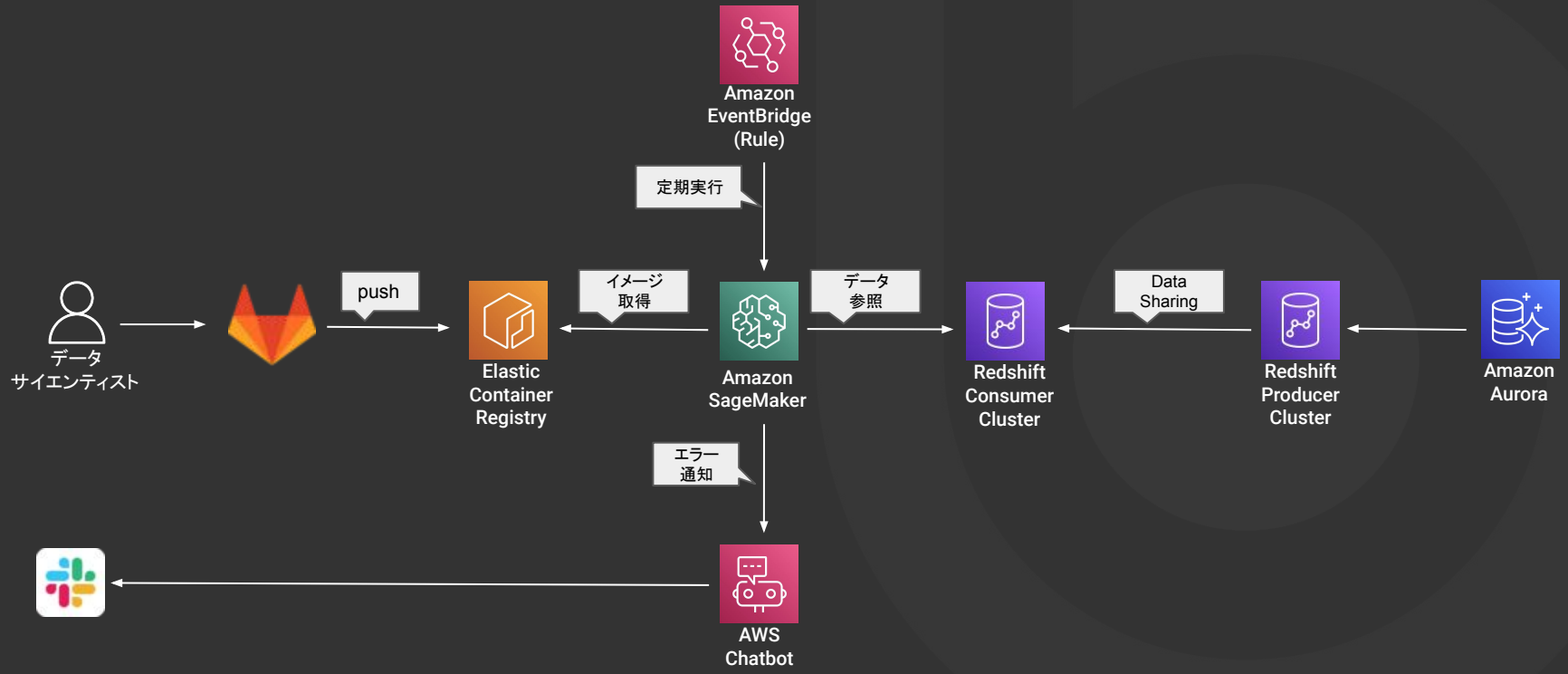
従前のデータサイエンティストの分析環境の課題

- 分析環境はローカル端末 or EC2でJupyter Notebookを起動していた
  - 分析環境が乱立しているため管理しきれなくなってきていた
- 人によって自由にライブラリをカスタマイズしていた
  - ある環境で動作している処理が別環境では動作しないといった事象が発生しチーム開発の阻害となっていた
- 大量データを扱うのにRedashからデータを取得していた
  - クエリが遅い
    - 社内からたくさんのリクエストがある
      - Athena経由でS3アクセスする際にS3のAPI制限に引っかかってクエリが失敗する
  - 一度に300万件以上のデータを取得できない Redashの制約がある

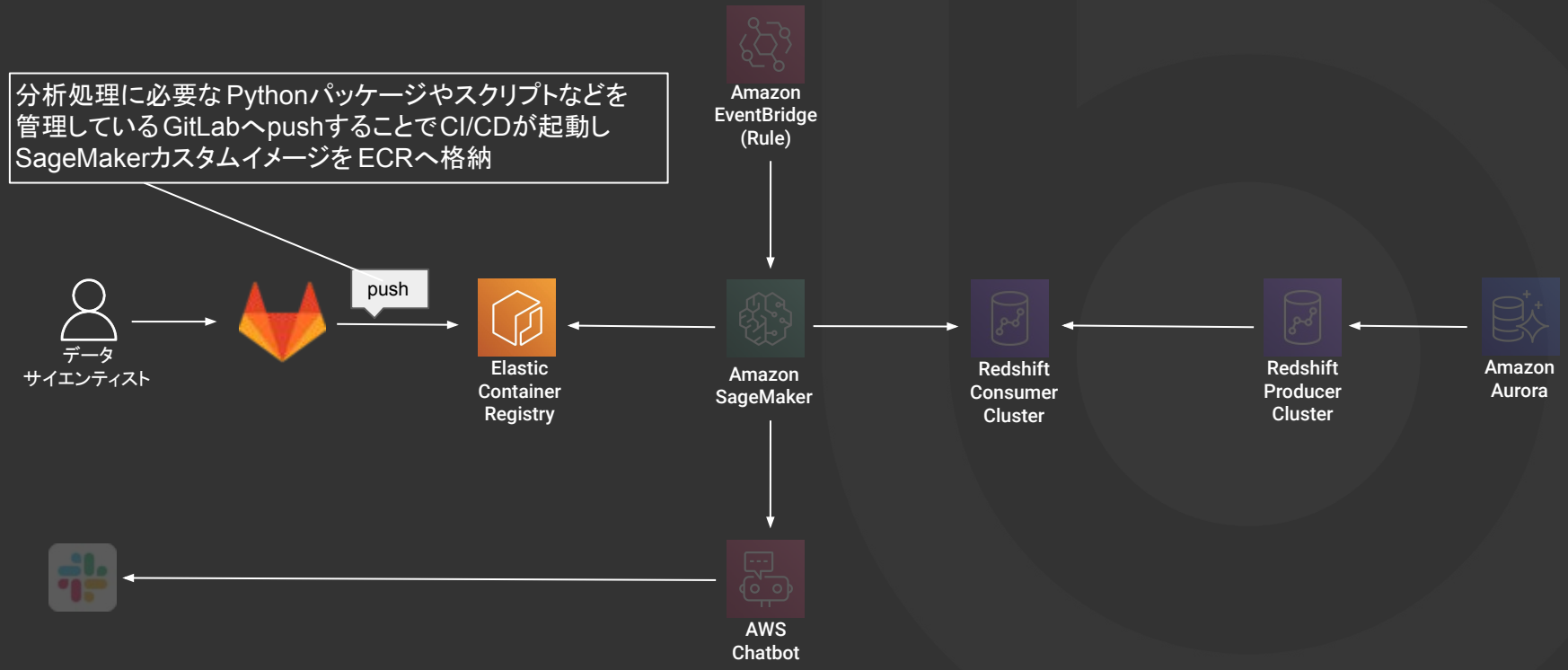
従前の分析環境にある課題の解決

- 分析環境はローカル端末 or EC2でJupyter Notebookを起動していた
  - **Amazon SageMakerに統一することで管理負担を下げる**
- 人によって自由にライブラリをカスタマイズしていた
  - **Amazon SageMakerに分析者共通で利用するカスタムイメージを用意**
- 大量データを扱うのにRedashからデータを取得していた
  - **データウェアハウス（Redshift）からデータを取得**

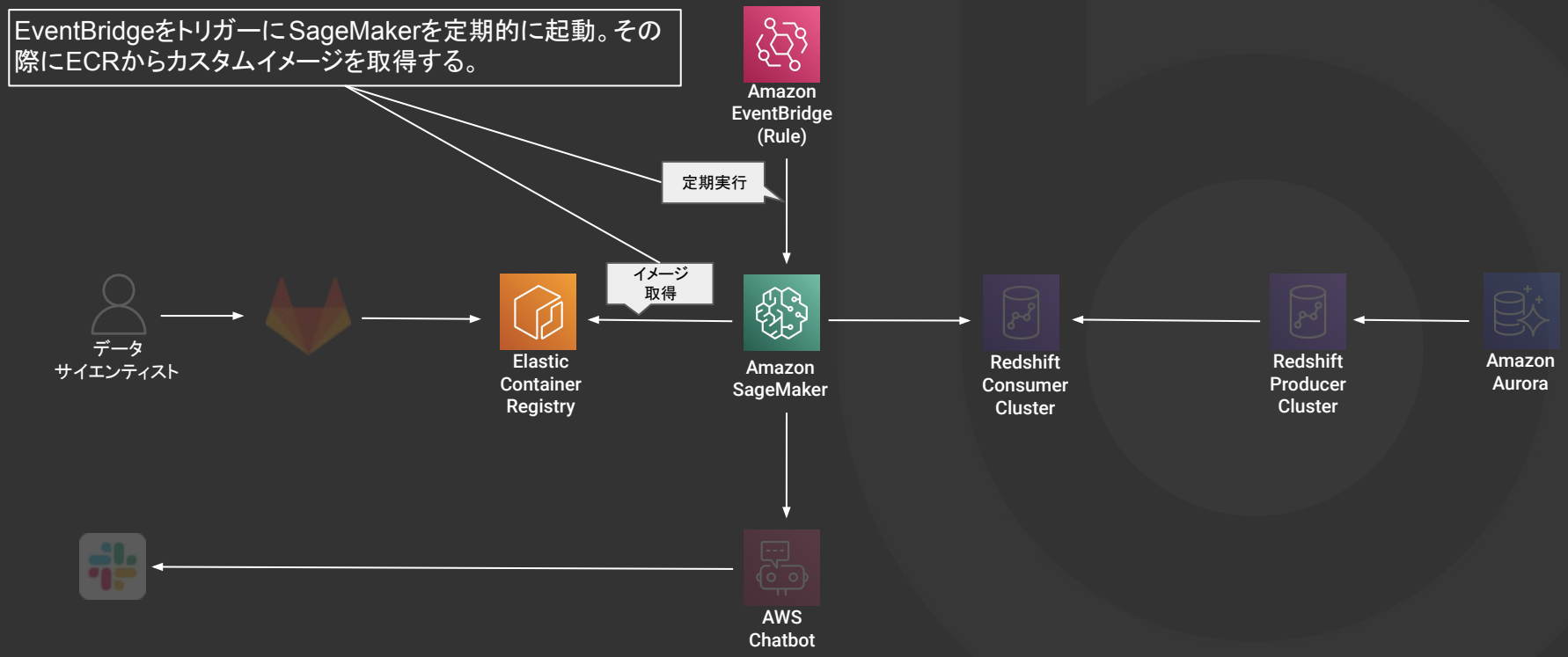
Amazon Redshiftを利用したAmazon SageMaker環境 - 全体構成図



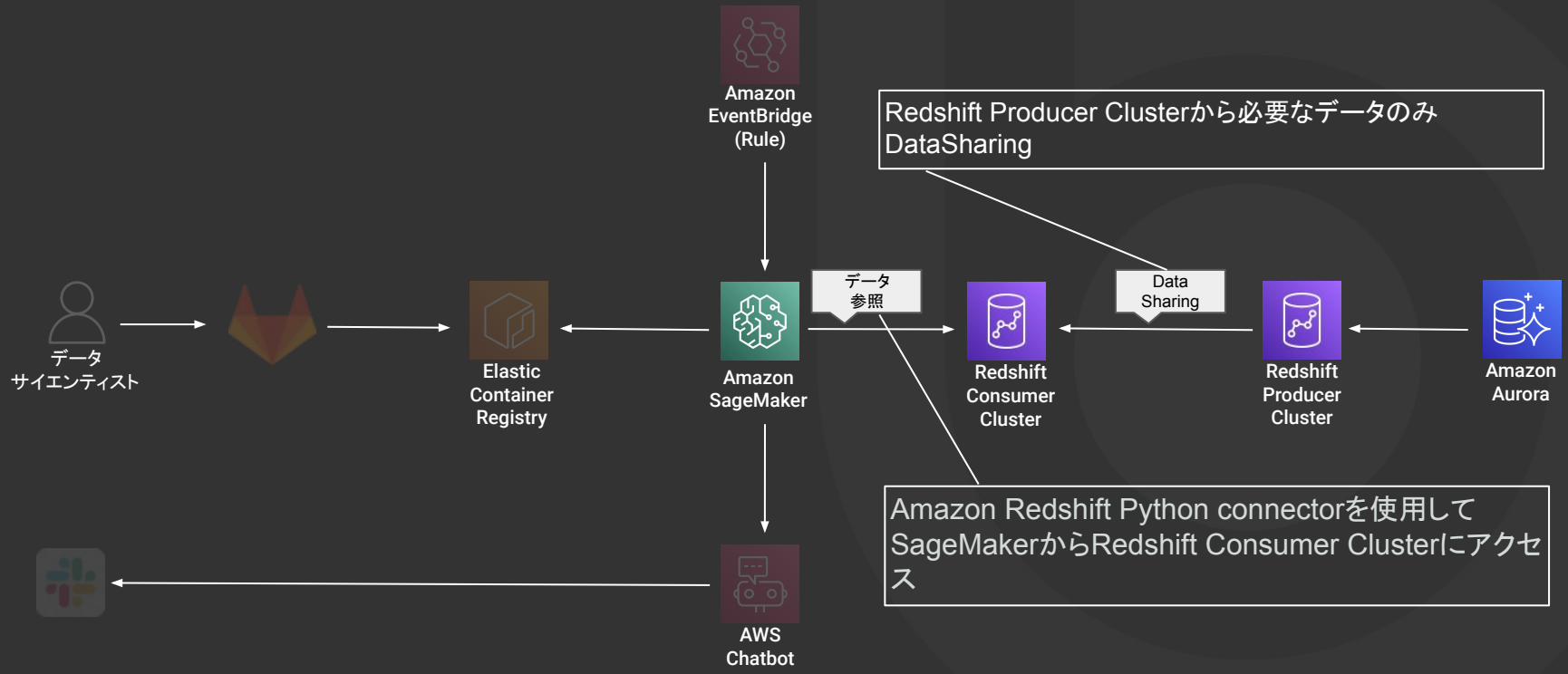
Amazon Redshiftを利用したAmazon SageMaker環境 - カスタムイメージの利用



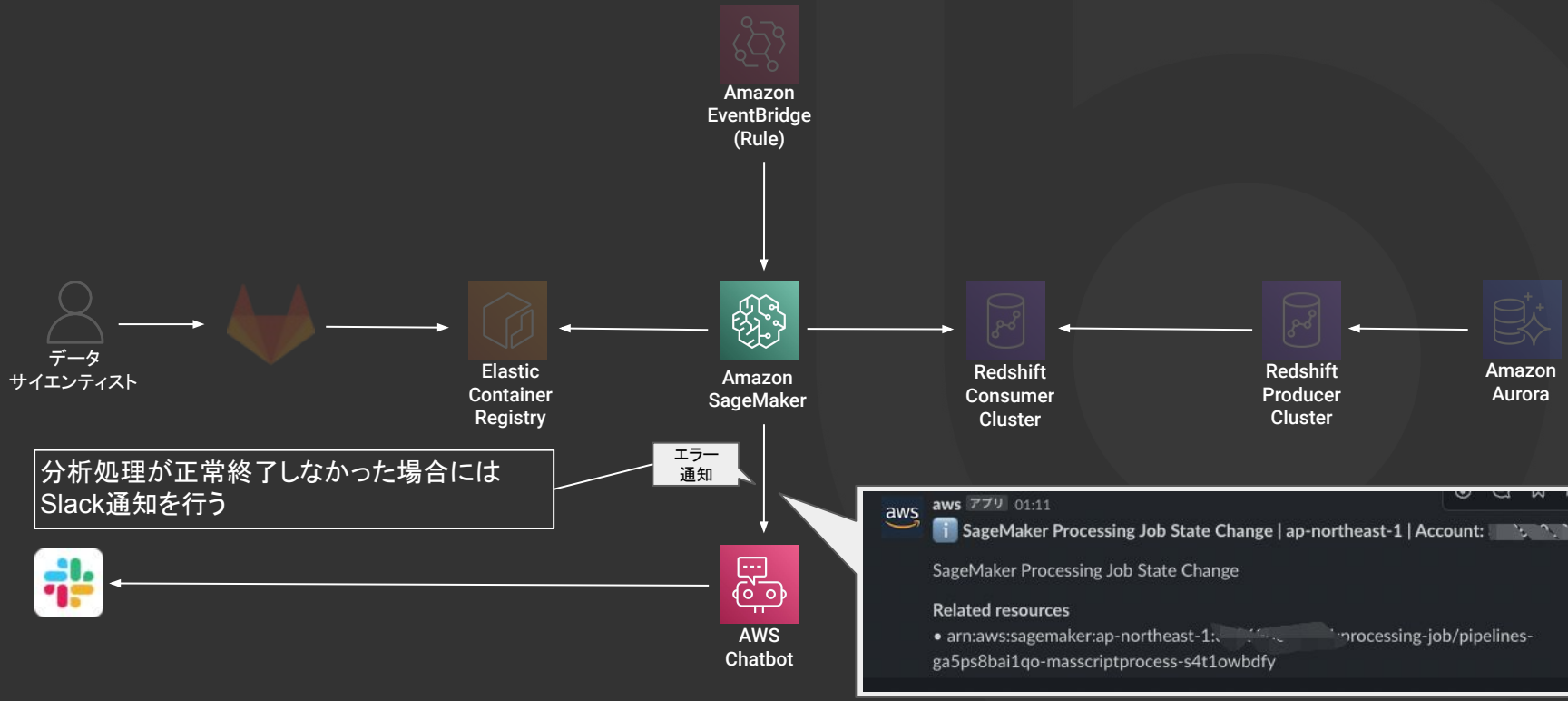
Amazon Redshiftを利用したAmazon SageMaker環境 - 分析処理



Amazon Redshiftを利用したAmazon SageMaker環境 - データソース



Amazon Redshiftを利用したAmazon SageMaker環境 - エラー通知



従前の分析環境にある課題解決の結果

- 分析環境はローカル端末 or EC2でJupyter Notebookを起動している
  - **Amazon SageMakerに統一することで管理負担を下げる**
- 人によって自由にライブラリをカスタマイズしている(属人的になっている)
  - **Amazon SageMakerに分析者共通で利用するカスタムイメージを用意**

結果

属人化の解消

- SageMakerカスタムイメージとすることで属人化の解消

分析環境のコード&IaC化

- 以前までは必要なパッケージなどが管理されておらず、別の分析環境を作る際に一から構成する必要があったが、カスタムイメージとすることで Gitの管理もできるようになった
- 分析環境構築もIaC化されたため、異なる分析用途の環境構築の横展開も可能になった

従前の分析環境にある課題解決の結果

- 大量データを扱うのにRedashからデータを取得している
  - **データウェアハウス（Redshift）からデータを取得**

結果

データ取得件数の制約解消

- データ参照先をRedashからRedshiftに変更することでRedashの制約から解放された

クエリ速度改善

- データ格納処理とRedashからのクエリが同じs3バケットにアクセスしている現状から分析用の Redshiftへの読み込みアクセスのみになった
- 1781万行の取得時間が数時間〜十数時間 → 2分

まとめ

Amazon Redshiftを利用したAmazon SageMaker環境を構築して

1. 必要なデータのみを分析環境RedshiftにDataSharingすることでサイロ化を防ぐことができる（有用なデータは全てデータウェアハウスへ）
2. 分析環境が統一されたことでデータサイエンティストが分析業務に注力できるようになった

今後の展望としてはデータ分析環境を展開していく予定

データ分析環境を展開していく

- データサイエンティストにはユーザビリティを下げずに分析業務を行なってほしいが、分析内容によっては高レベルのセキュリティを求められるケースがある
- 一つの分析環境で分析業務を行う場合、その環境のセキュリティを高レベルにしなければならない
- 一つの分析環境にまとめるよりも統制の効いた環境を個別に作成した方が分析用途によってセキュリティレベルを変えることができる

→ データ分析環境をマルチアカウント展開するのが良さそう

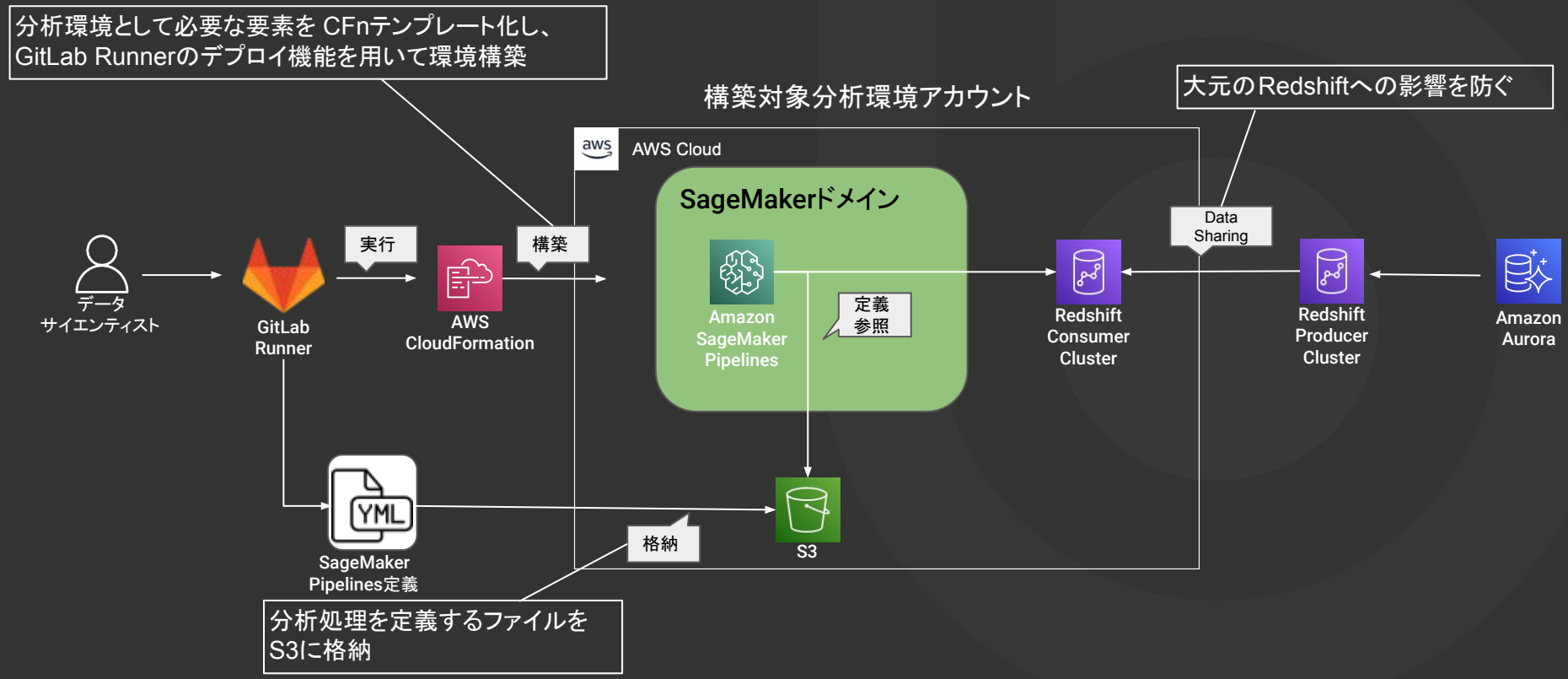
データ分析環境を展開していく

- データサイエンティストにはユーザビリティを下げずに分析業務を行なってほしいが、分析内容によっては高レベルのセキュリティを求められるケースがある
- 一つの分析環境で分析業務を行う場合、その環境のセキュリティを高レベルにしなければならない
- 一つの分析環境にまとめるよりも統制の効いた環境を個別に作成した方が分析用途によってセキュリティレベルを変えることができる

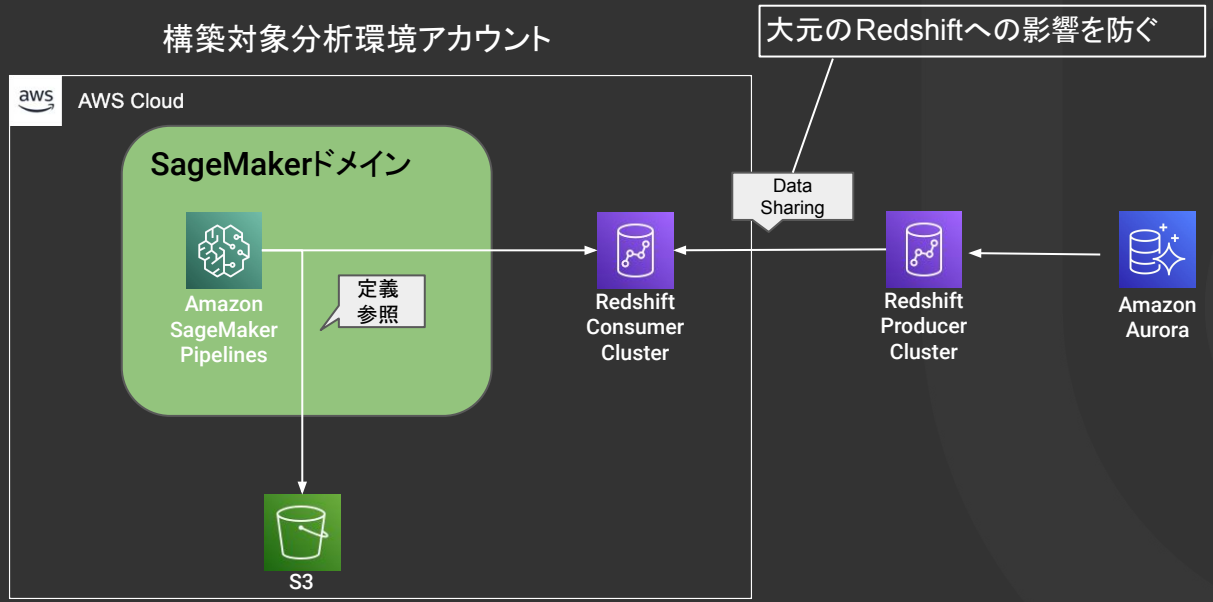
→ データ分析環境をマルチアカウント展開するのが良さそう

とはいえできるだけ構築を簡略化したい

Amazon Redshiftを利用したAmazon SageMaker環境 - 汎用化版全体図



Amazon Redshiftを利用したAmazon SageMaker環境 - 汎用化版全体図



メリット

物理的ではなく論理的にデータ連携

・データのコピー/移動することなくデータ参照が可能

必要なデータのみ連携

・分析業務に必要なデータのみ連携することでデータへのアクセス権限を絞ることができる

DataSharing先のRedshiftへの書き込み不可

・大元のRedshift(データウェアハウス)へ書き込みができないので分析環境が乱立してもデータ汚染の心配がない

Amazon Redshift Data APIを利用した顧客へのデータ提供

自己紹介

長尾 康志 (ながお やすし)

ビットバンク株式会社

システム部門 プラットフォーム部

インフラチーム エンジニア

ここからの内容

話すこと

・**Amazon Redshift**を利用した顧客へのデータ提供

・**Redshift Data API**の活用事例

話さないこと

・暗号資産取引所のデータの中身

ビットバンクが顧客に提供するデータ

・取引所の注文履歴

・取引所の約定履歴

・販売所の売買履歴

ビットバンクが顧客に提供するデータ

- **取引所の注文履歴**
- **取引所の約定履歴**
- 販売所の売買履歴

今回はこのふたつについてお話しします

ビットバンクが顧客に提供するデータ

・取引所の注文履歴

期間や通貨ペアを指定して

暗号資産の注文履歴を表示できる機能

・取引所の約定履歴

期間や通貨ペアを指定して

売買が成立した履歴を取得できる機能

Amazon Redshift導入前の
顧客へのデータ提供についてお話しします

従来の顧客へのデータ提供

従来の顧客へのデータ提供

Auroraから顧客にデータを提供する問題点

①データ抽出の負荷がサービスに影響し得る

従来の顧客へのデータ提供

Auroraから顧客にデータを提供する問題点

①データ抽出の負荷がサービスに影響し得る

②Auroraへの負荷を軽減するため、抽出上限を1リクエストあたり500件に

従来の顧客へのデータ提供

Auroraから顧客にデータを提供する問題点

①データ抽出の負荷がサービスに影響し得る

②Auroraへの負荷を軽減するため、抽出上限を1リクエストあたり500件に

③Auroraのデータが増え続け、クエリの性能が劣化していく

Amazon Redshiftが導入された今、
従来の全ての問題が解決できる

Amazon Redshiftを利用した顧客へのデータ提供

Auroraから顧客にデータを提供する問題点

①データ抽出の負荷がサービスに影響し得る

②Auroraへの負荷を軽減するため、抽出上限を1リクエストあたり500件に

③Auroraのデータが増え続け、クエリの性能が劣化していく

Amazon Redshiftを利用した顧客へのデータ提供

Auroraから顧客にデータを提供する問題点

①データ抽出の負荷がサービスに影響し得る

→ サービスと分離したデータ基盤としてAmazon Redshiftを利用

②Auroraへの負荷を軽減するため、抽出上限を1リクエストあたり500件に

③Auroraのデータが増え続け、クエリの性能が劣化していく

Amazon Redshiftを利用した顧客へのデータ提供

Auroraから顧客にデータを提供する問題点

①データ抽出の負荷がサービスに影響し得る

→ サービスと分離したデータ基盤としてAmazon Redshiftを利用

②Auroraへの負荷を軽減するため、抽出上限を1リクエストあたり500件に

→ サービスへの影響が無いため、抽出上限を解除

③Auroraのデータが増え続け、クエリの性能が劣化していく

Amazon Redshiftを利用した顧客へのデータ提供

Auroraから顧客にデータを提供する問題点

①データ抽出の負荷がサービスに影響し得る

→ サービスと分離したデータ基盤としてAmazon Redshiftを利用

②Auroraへの負荷を軽減するため、抽出上限を1リクエストあたり500件に

→ サービスへの影響が無いため、抽出上限を解除

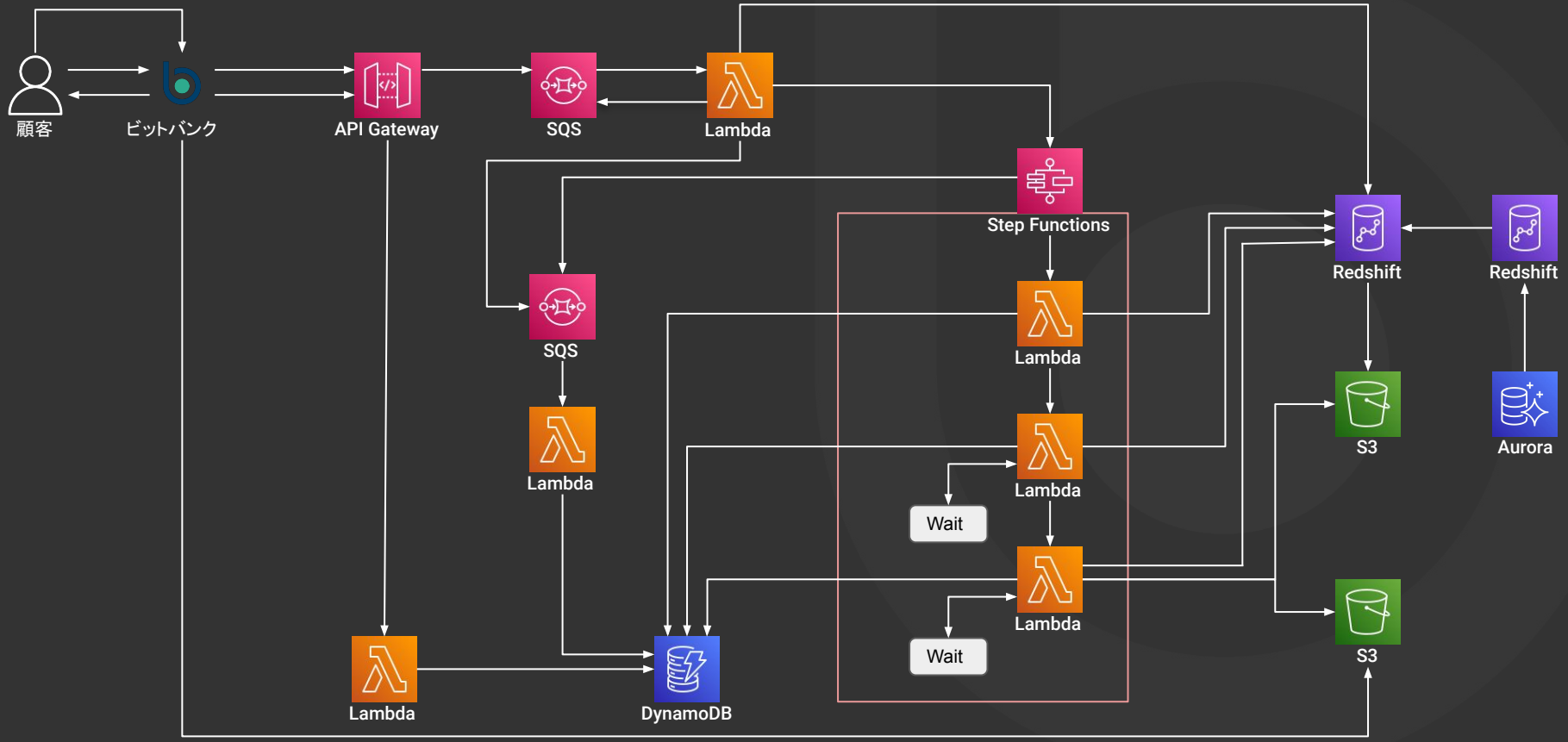
③Auroraのデータが増え続け、クエリの性能が劣化していく

→ 提供するデータがRedshiftにあるため、Auroraからデータをパージできる

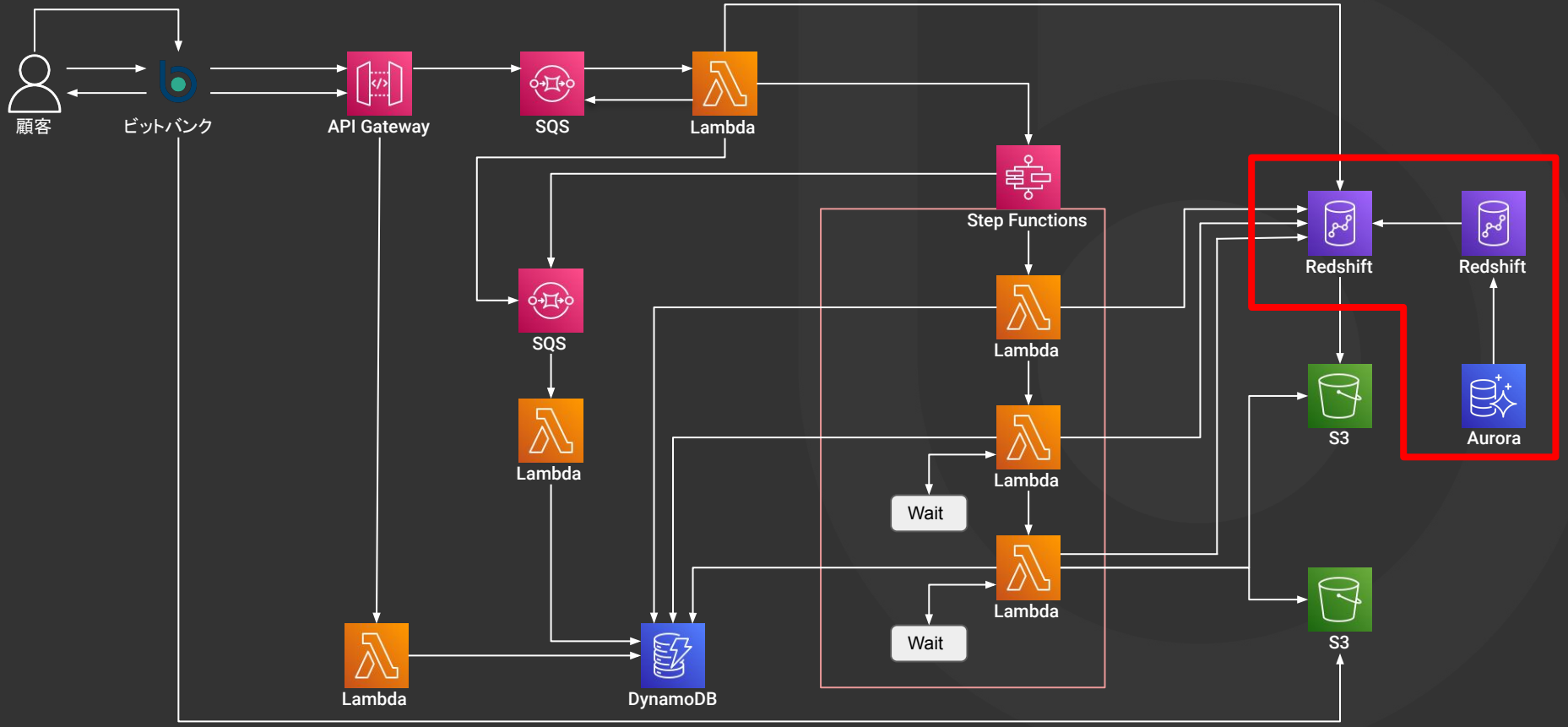
Amazon Redshiftを利用した顧客へのデータ提供

構成図

Amazon Redshiftを利用した顧客へのデータ提供 - 全体構成図

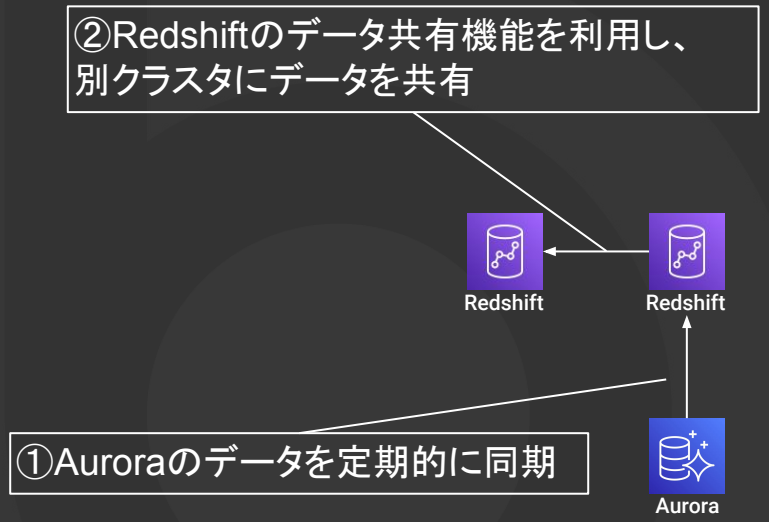


Amazon Redshiftを利用した顧客へのデータ提供 - データソース



Amazon Redshiftを利用した顧客へのデータ提供 - データソース

Amazon Redshiftを利用した顧客へのデータ提供 - データソース



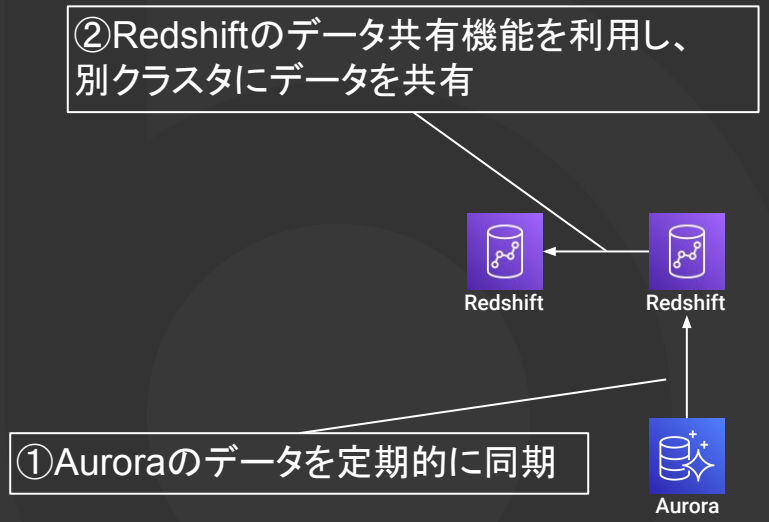
Amazon Redshiftを利用した顧客へのデータ提供 - データソース

データ共有機能を利用するメリット

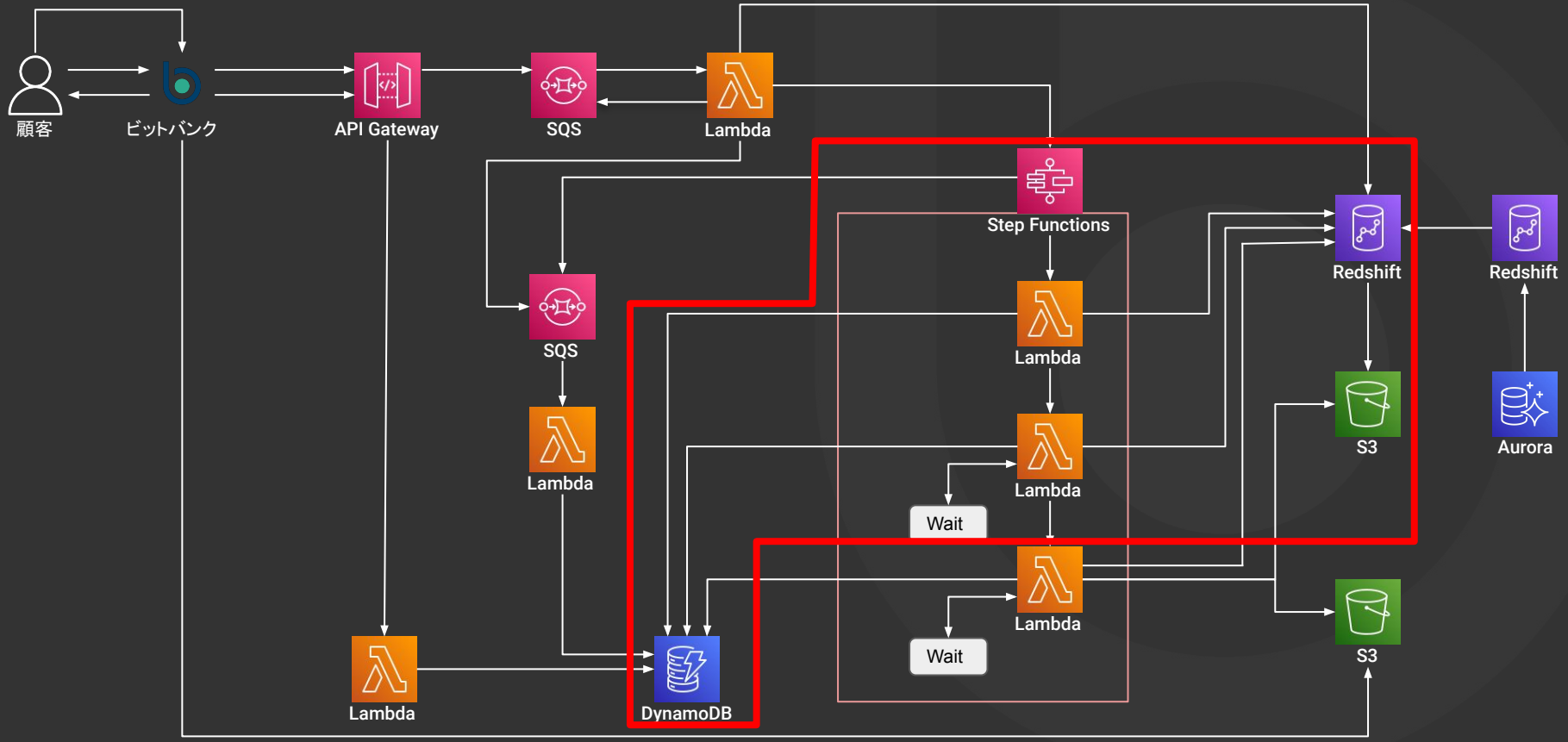
・特定のテーブルのみ共有できる

・クラスタを分けることでデータ抽出の負荷のみに

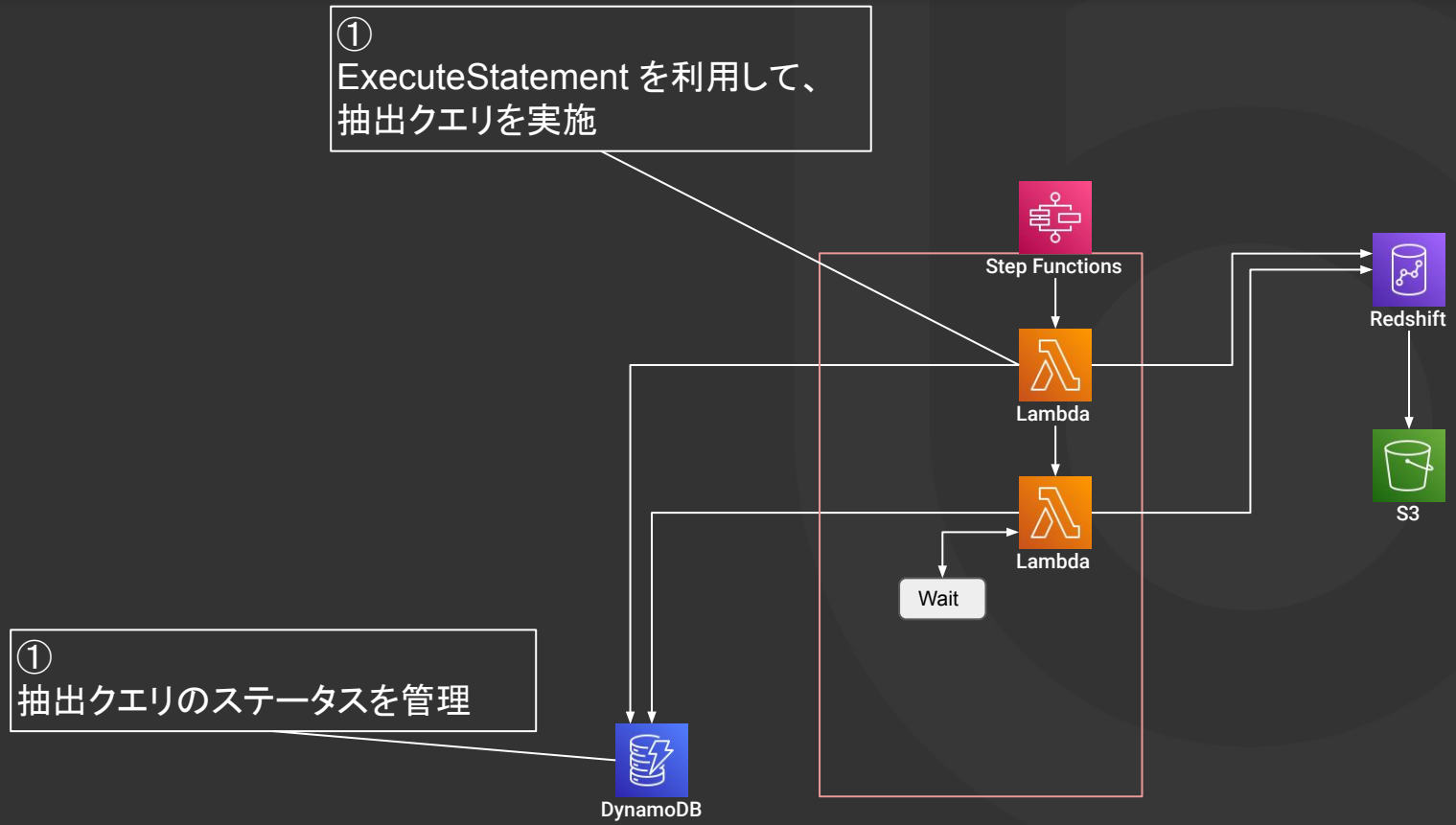
・Readのみの権限で利用可能



Amazon Redshiftを利用した顧客へのデータ提供 - データ抽出機構



Amazon Redshiftを利用した顧客へのデータ提供 - データ抽出機構



Amazon Redshiftを利用した顧客へのデータ提供 - データ抽出機構

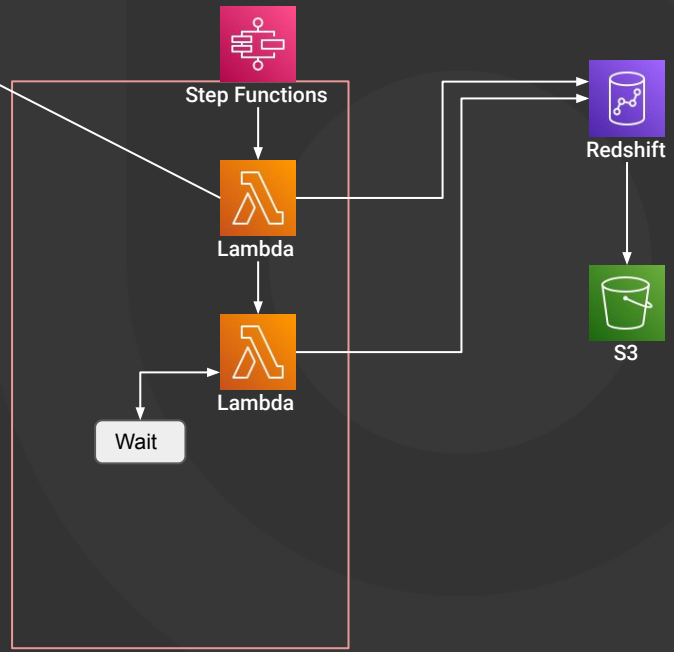
①
ExecuteStatement を利用して、
抽出クエリを実施

ExecuteStatement:

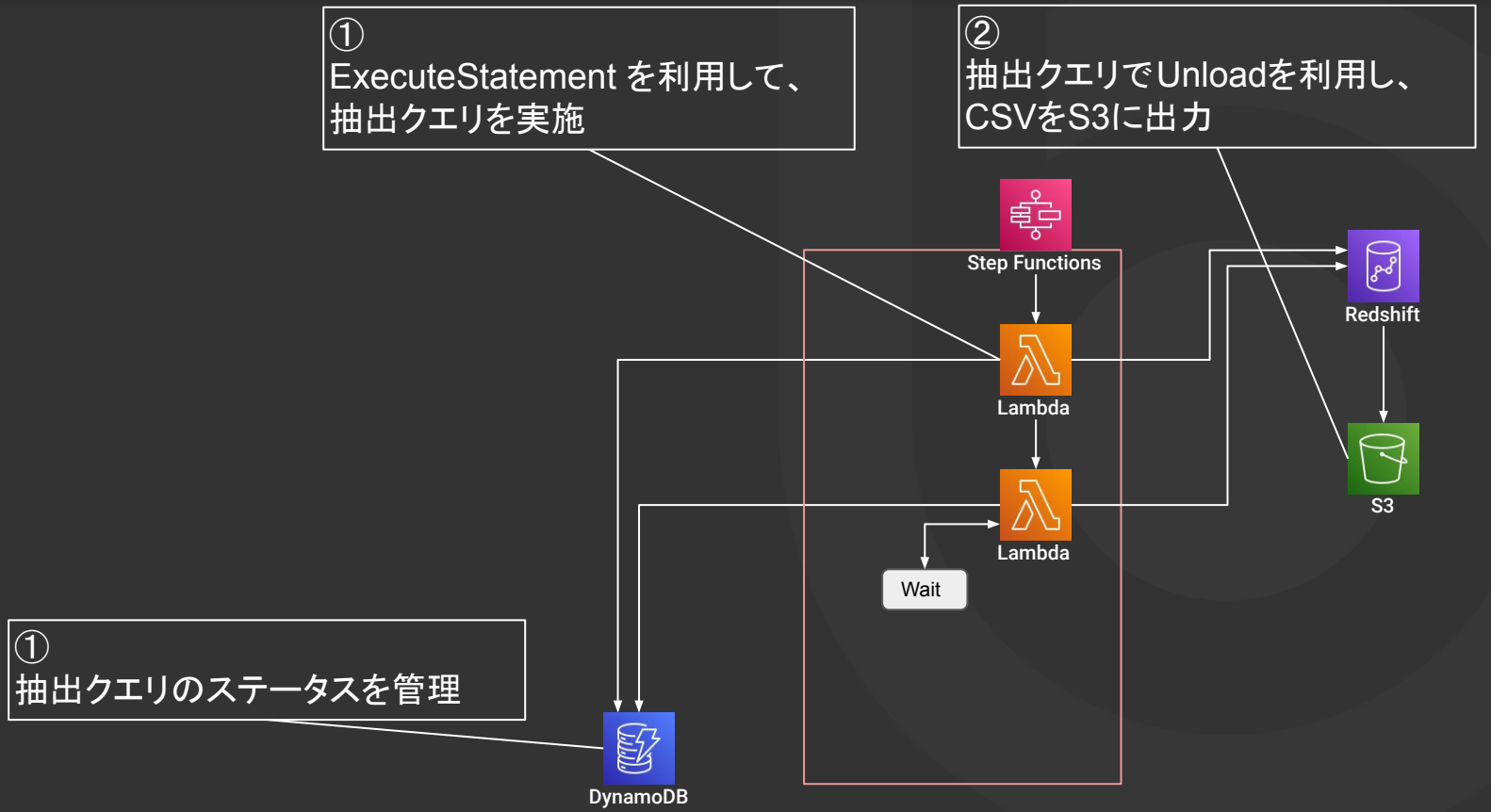
・Redshiftへのクエリ実行が可能

・Unloadを利用したクエリを実行することで、
RedshiftからS3に直接出力できる

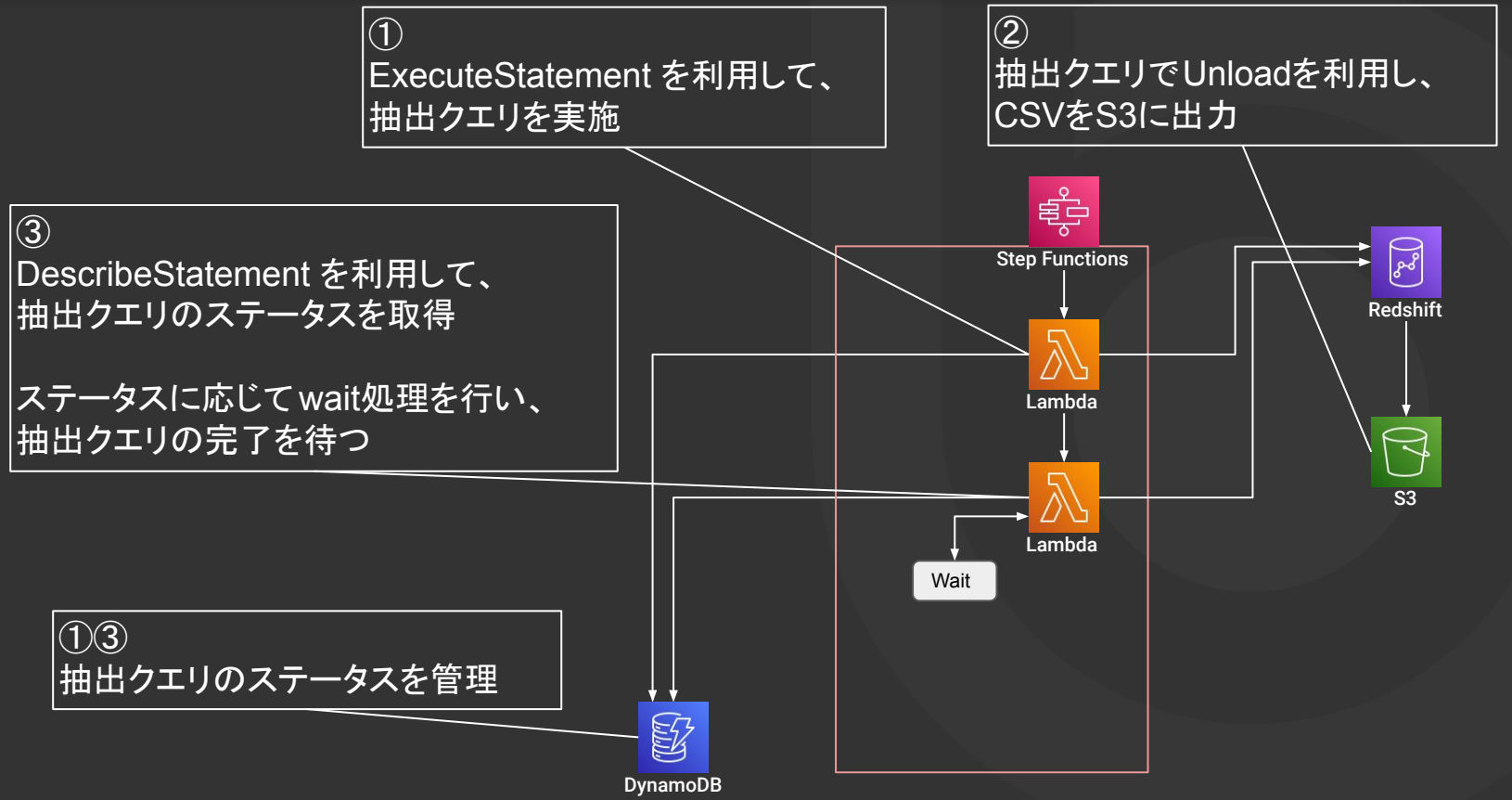
・実行を開始するだけ
結果の取得やステータスの確認は出来ない



Amazon Redshiftを利用した顧客へのデータ提供 - データ抽出機構



Amazon Redshiftを利用した顧客へのデータ提供 - データ抽出機構



Amazon Redshiftを利用した顧客へのデータ提供 - データ抽出機構

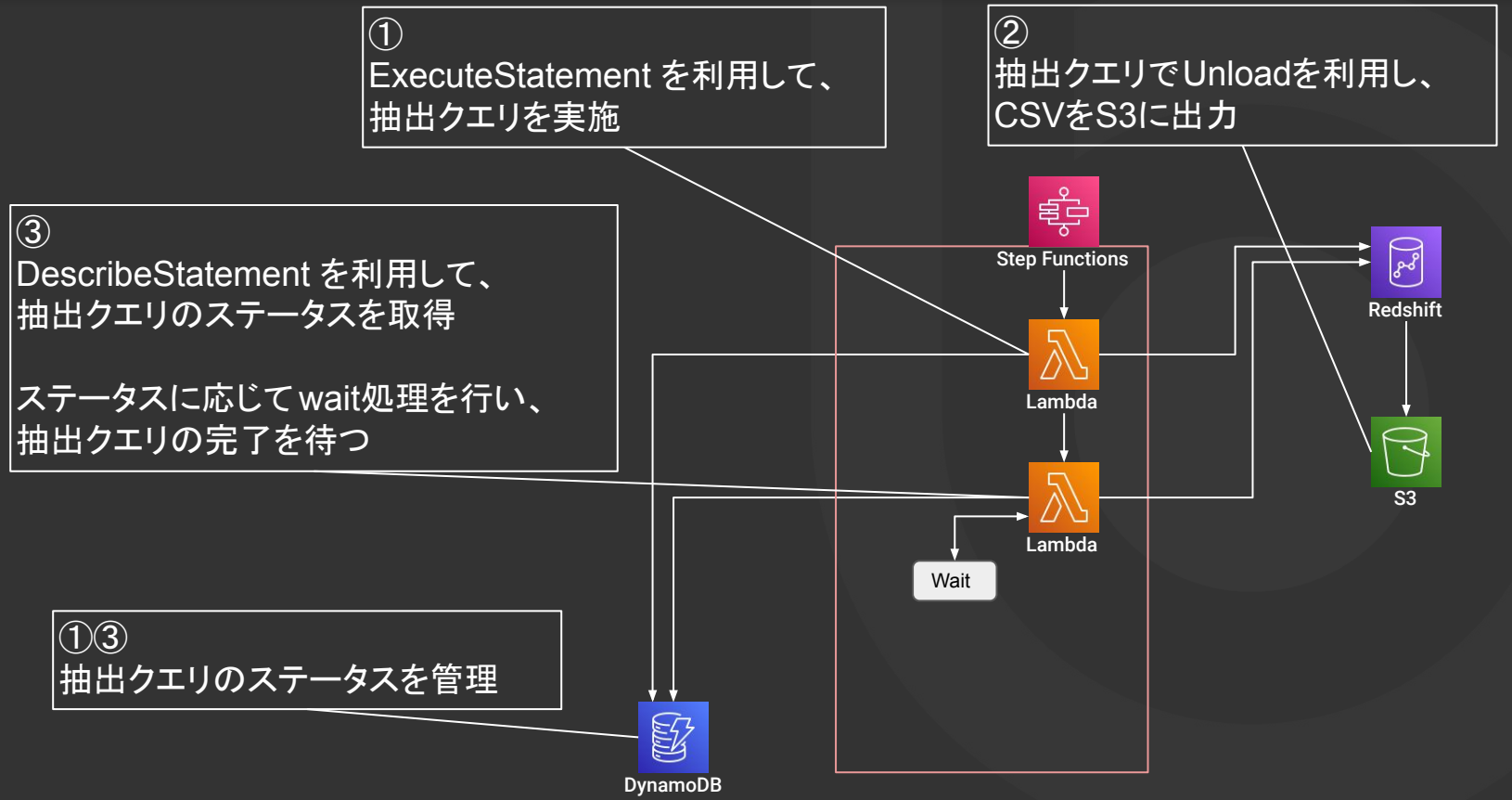
③
DescribeStatement を利用して、
抽出クエリのステータスを取得

ステータスに応じてwait処理を行い、
抽出クエリの完了を待つ

DescribeStatement:

- ExecuteStatementで実行したクエリのIDを利用
  クエリのステータスを取得することができる
- Step Functions内に組み込むことで、
  クエリのステータスによる分岐が可能に

Amazon Redshiftを利用した顧客へのデータ提供 - データ抽出機構



Redshift Data APIの説明 - Redshift Data APIの注意点

実際にあった問題

・ExecuteStatementで実行したクエリのステータスが、
　DescribeStatementで取得できない

Redshift Data APIの説明 - Redshift Data APIの注意点

実際にあった問題

・ExecuteStatementで実行したクエリのステータスが、
DescribeStatementで取得できない

→ 実行時と取得時で同一のIAM Role
または同一のIAMユーザを使用する必要がある

Redshift Data APIを使うメリット

・Redshiftへの接続を維持し続けなくて良い

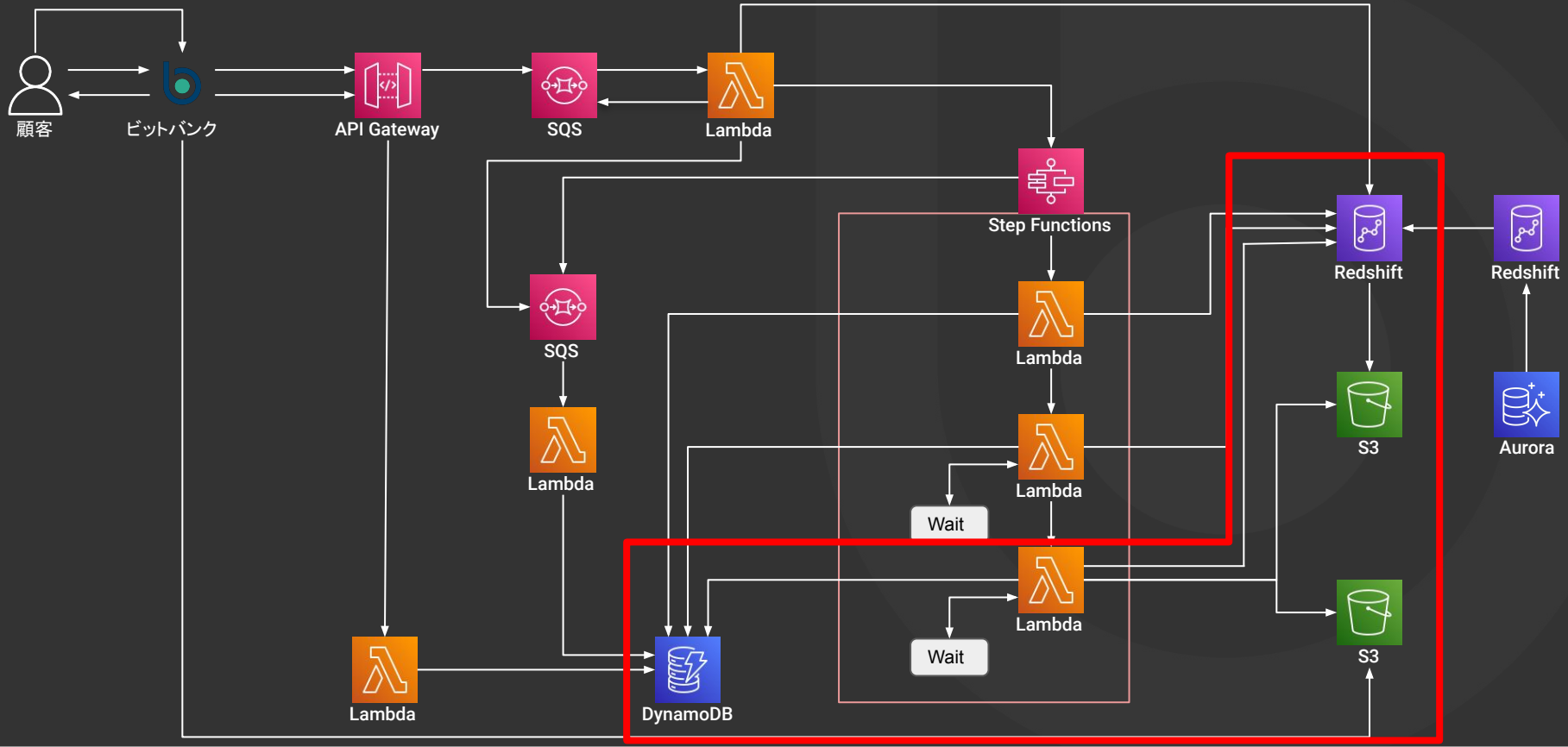
・SDKが提供されており開発しやすく、認証にパスワードを利用しなくても良い

・長時間動くクエリでも、非同期で実行できるため問題ない(最大24時間まで)

・Data APIのクエリ結果の最大サイズは 100 MBだが、Unloadを使えば回避できる

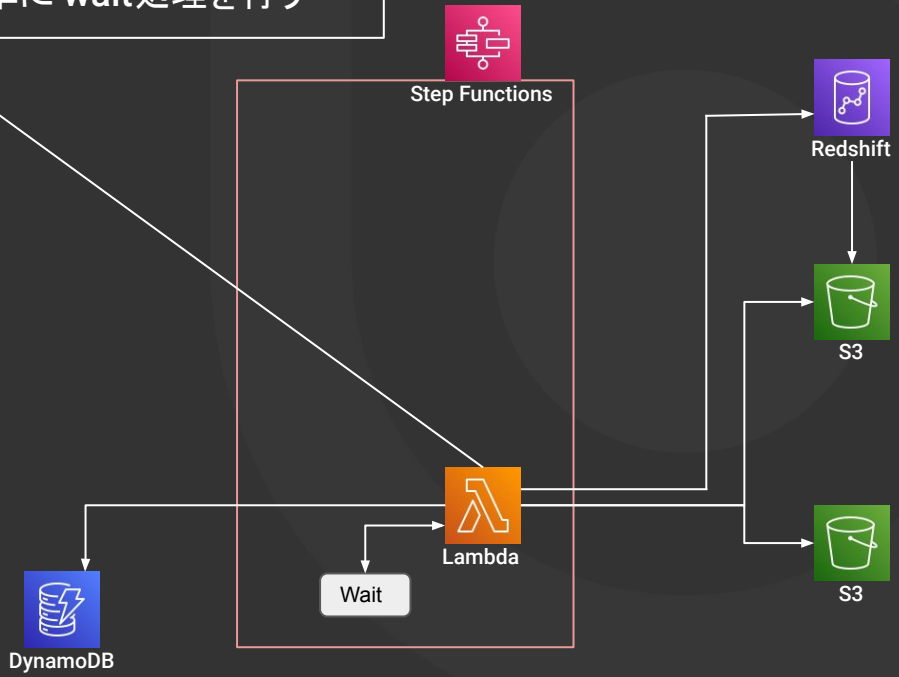
Unload: クエリの結果をS3に出力できる

Amazon Redshiftを利用した顧客へのデータ提供 - 抽出データ統合



Amazon Redshiftを利用した顧客へのデータ提供 - 抽出データ統合

①
ListStatementsを実行
現在実行中のクエリ数を取得
Redshiftのクエリ同時実行上限を基準に wait処理を行う



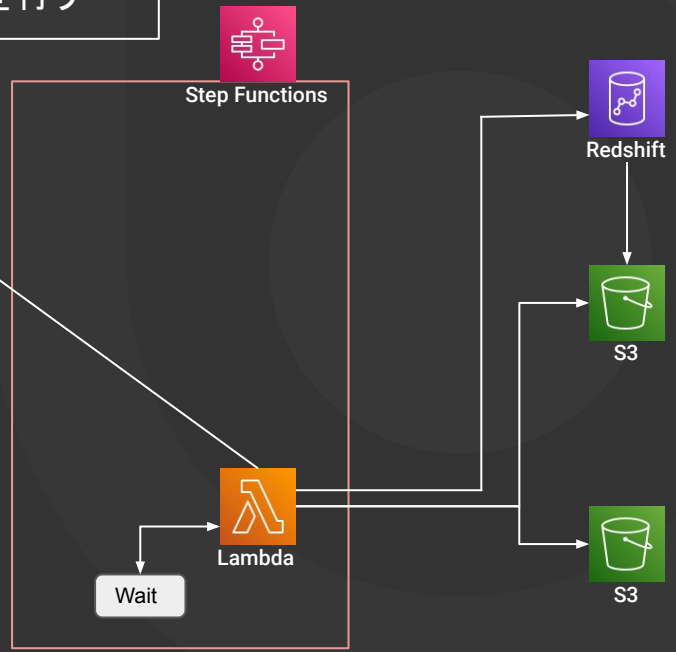
Amazon Redshiftを利用した顧客へのデータ提供 - 抽出データ統合

①
ListStatementsを実行
現在実行中のクエリ数を取得
Redshiftのクエリ同時実行上限を基準に wait処理を行う

ListStatements:

・Redshiftで実行中のクエリー覧が取得できる

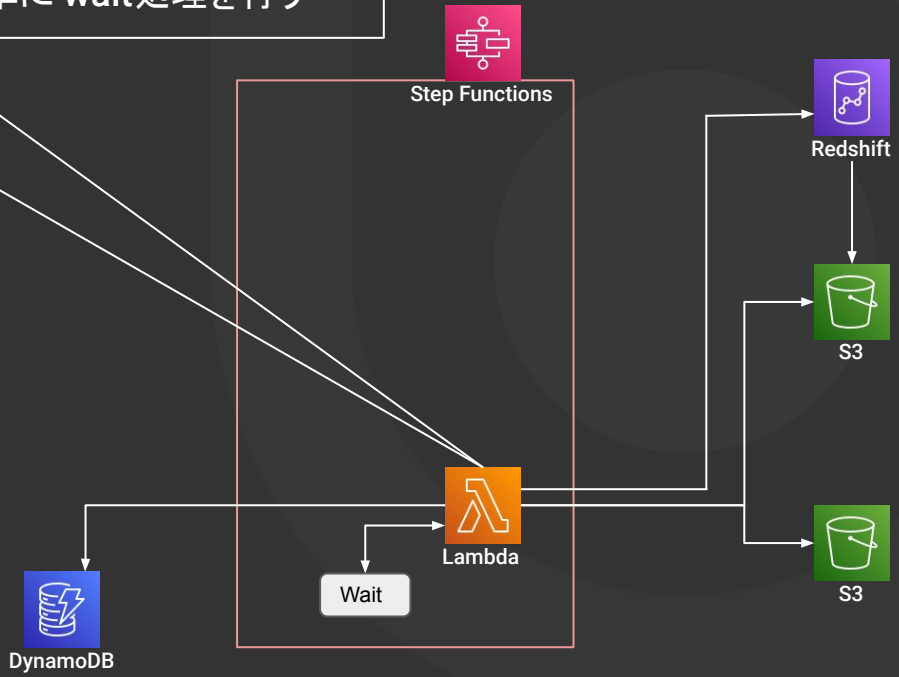
・クエリ同時実行数を調整したい場合に便利



Amazon Redshiftを利用した顧客へのデータ提供 - 抽出データ統合

①
ListStatementsを実行
現在実行中のクエリ数を取得
Redshiftのクエリ同時実行上限を基準に wait処理を行う

②
S3からS3へファイルを圧縮しながら転送



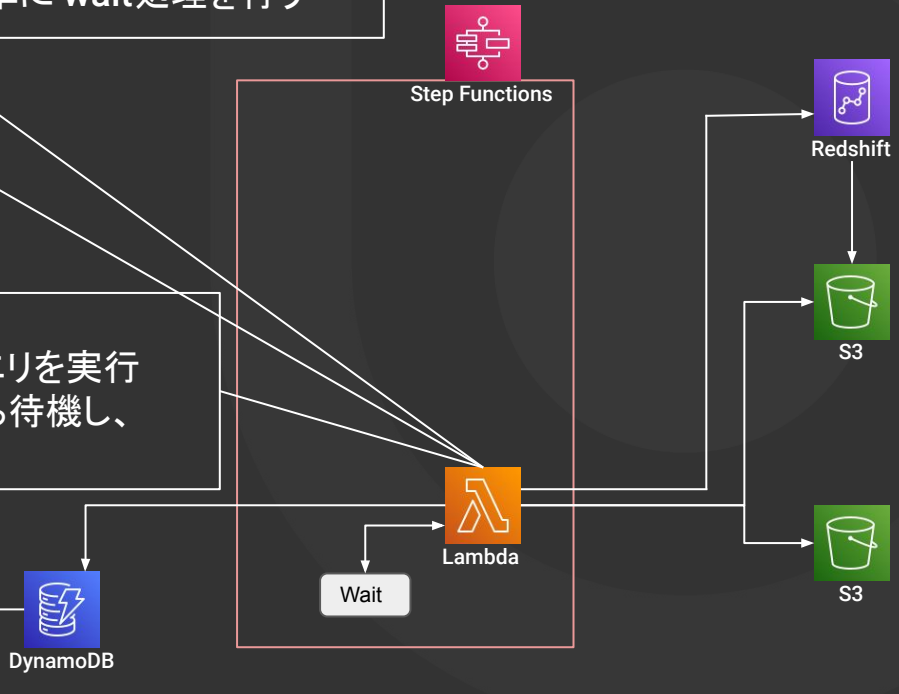
Amazon Redshiftを利用した顧客へのデータ提供 - 抽出データ統合

①
ListStatementsを実行
現在実行中のクエリ数を取得
Redshiftのクエリ同時実行上限を基準に wait処理を行う

②
S3からS3へファイルを圧縮しながら転送

③
ExecuteStatementを利用して抽出行数取得クエリを実行
Lambda内でDescribeStatementを実行しながら待機し、
GetStatementResultで抽出行数を取得

③
抽出クエリのステータスを管理
抽出クエリの 抽出行数を保管

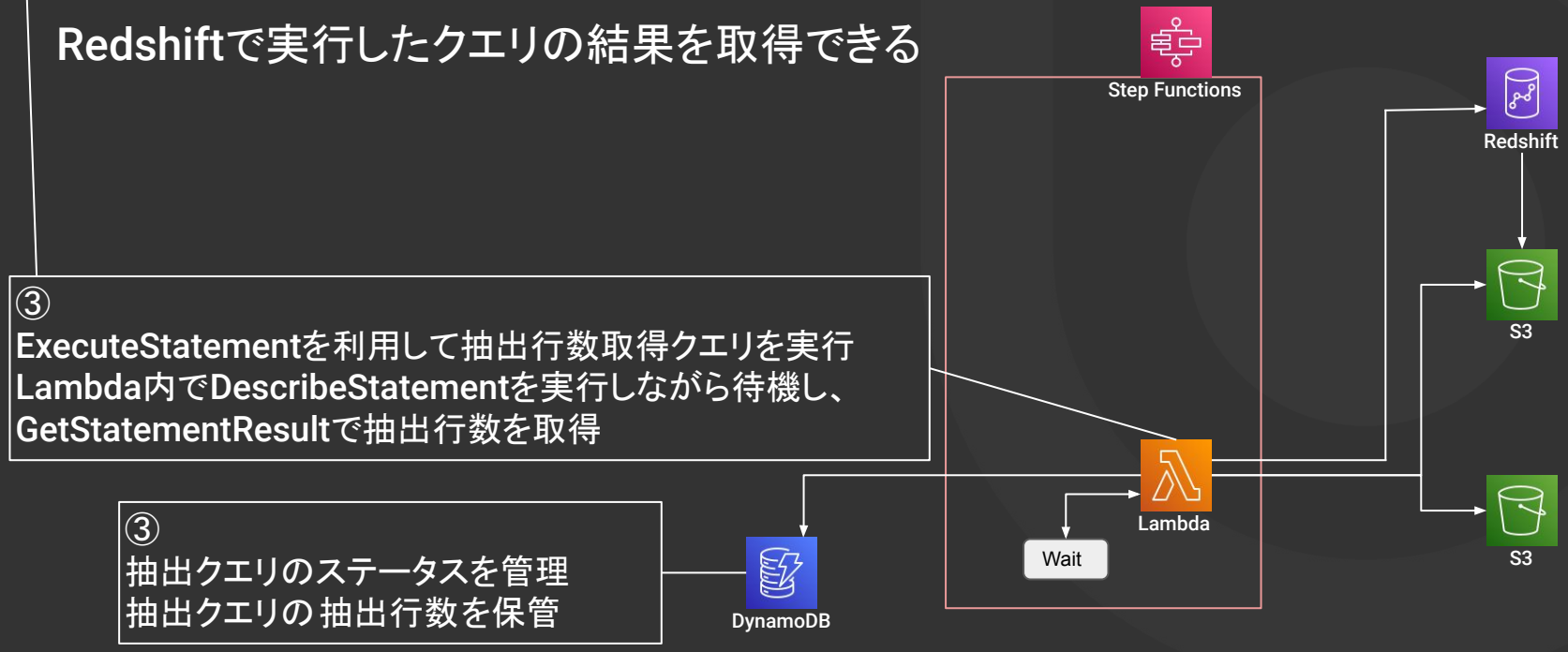


Amazon Redshiftを利用した顧客へのデータ提供 - 抽出データ統合

GetStatementResult:

・ExecuteStatementで実行したクエリのIDを利用

Redshiftで実行したクエリの結果を取得できる



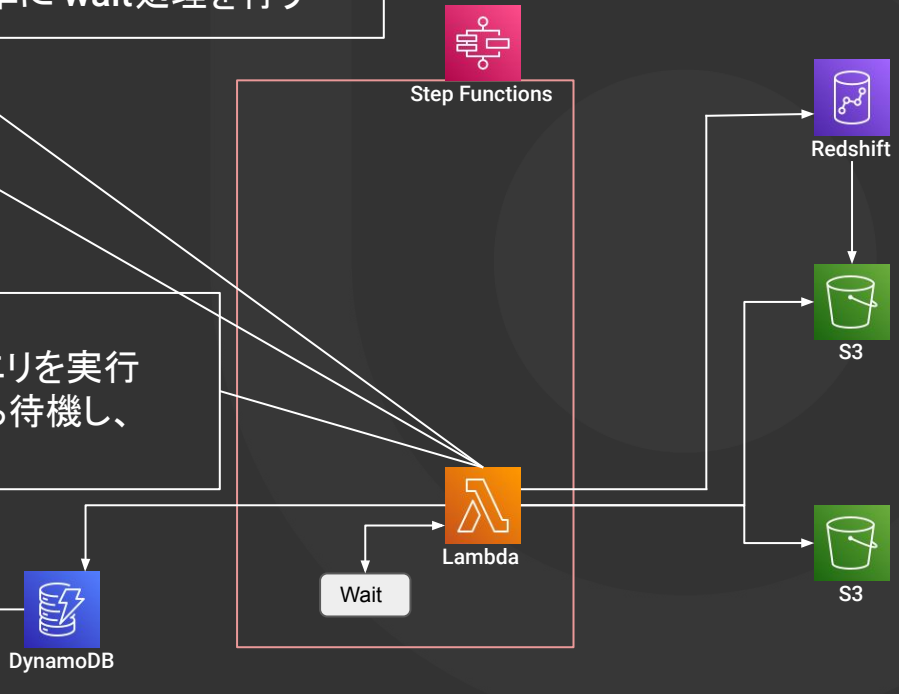
Amazon Redshiftを利用した顧客へのデータ提供 - 抽出データ統合

①
ListStatementsを実行
現在実行中のクエリ数を取得
Redshiftのクエリ同時実行上限を基準に wait処理を行う

②
S3からS3へファイルを圧縮しながら転送

③
ExecuteStatementを利用して抽出行数取得クエリを実行
Lambda内でDescribeStatementを実行しながら待機し、
GetStatementResultで抽出行数を取得

③
抽出クエリのステータスを管理
抽出クエリの 抽出行数を保管



Redshift Data APIの説明 - Redshift Data APIの注意点

実際にあった問題

・ExecuteStatementで実行したクエリの結果が、
GetStatementResultで取得できない

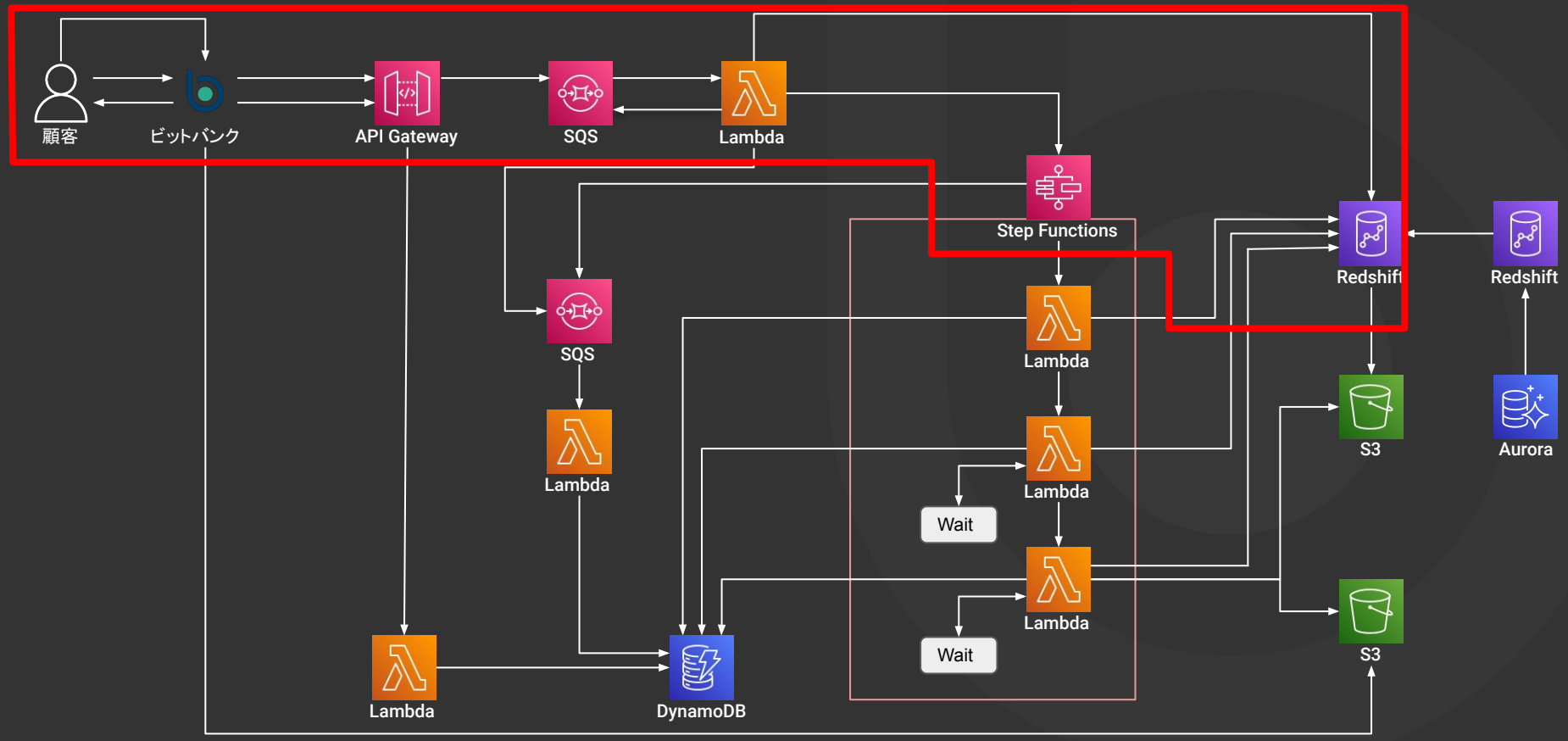
Redshift Data APIの説明 - Redshift Data APIの注意点

実際にあった問題

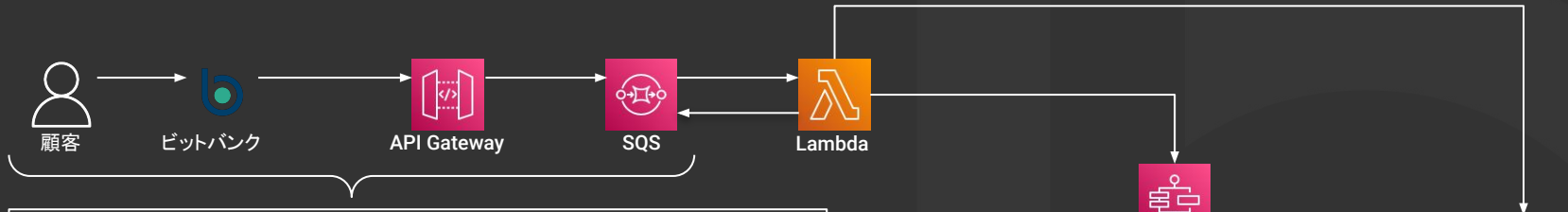
・ExecuteStatementで実行したクエリの結果が、
GetStatementResultで取得できない

→ 実行時と取得時で同一のIAM Role
または同一のIAMユーザを使用する必要がある

Amazon Redshiftを利用した顧客へのデータ提供 - データ抽出実行

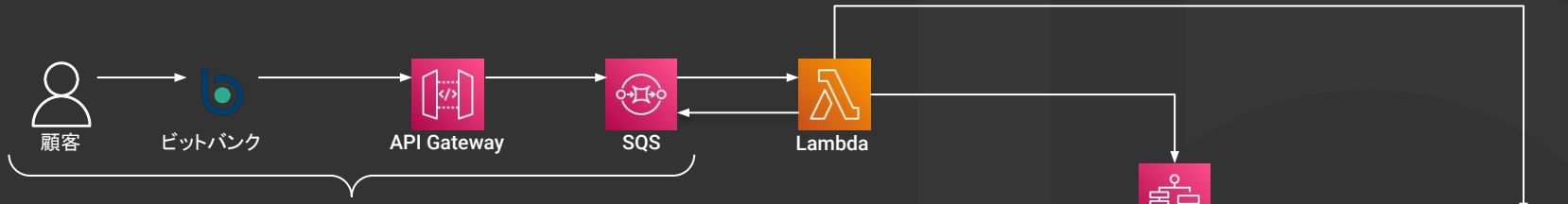


Amazon Redshiftを利用した顧客へのデータ提供 - データ抽出実行



①
ユーザはアプリケーションから抽出リクエスト
抽出リクエストはAPI GatewayからSQSに送られる

Amazon Redshiftを利用した顧客へのデータ提供 - データ抽出実行



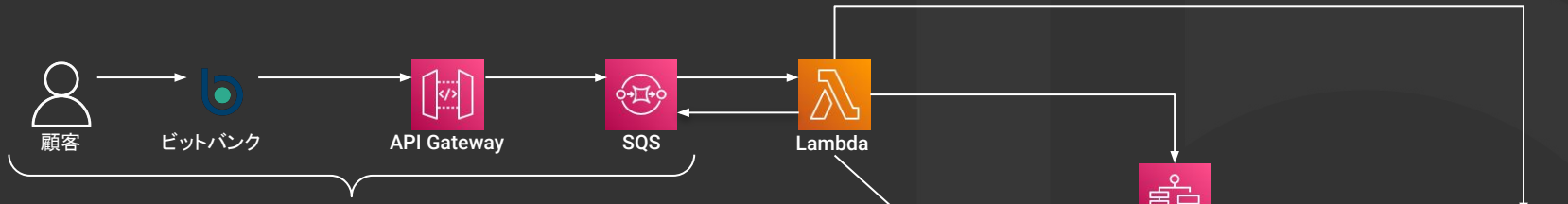
①
ユーザはアプリケーションから抽出リクエスト
抽出リクエストはAPI GatewayからSQSに送られる

SQSを利用するメリット

・SQSにリクエストが貯まる仕組みのため、

顧客利用に影響なくRedshiftのメンテナンスが可能

Amazon Redshiftを利用した顧客へのデータ提供 - データ抽出実行

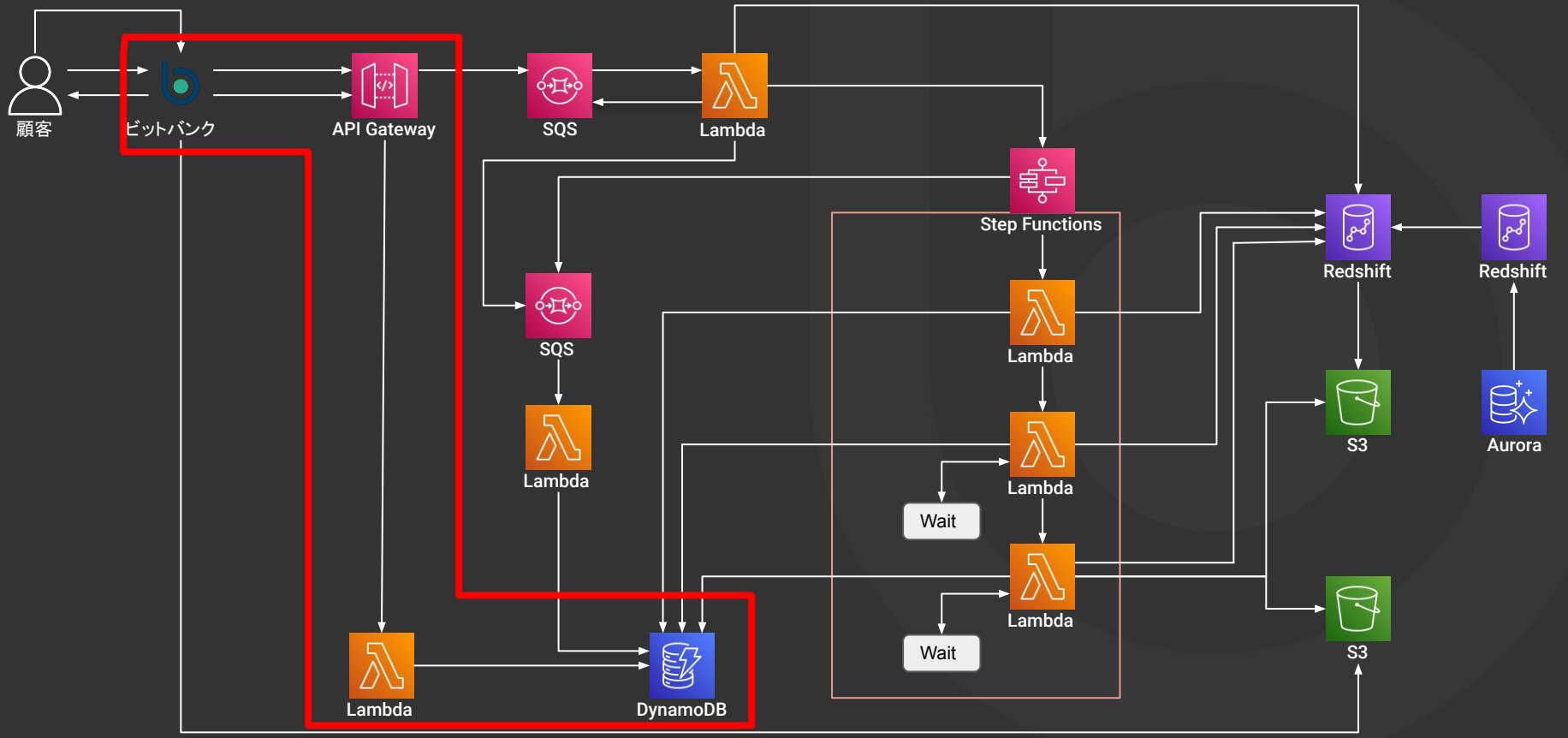


①
ユーザはアプリケーションから抽出リクエスト
抽出リクエストはAPI GatewayからSQSに送られる

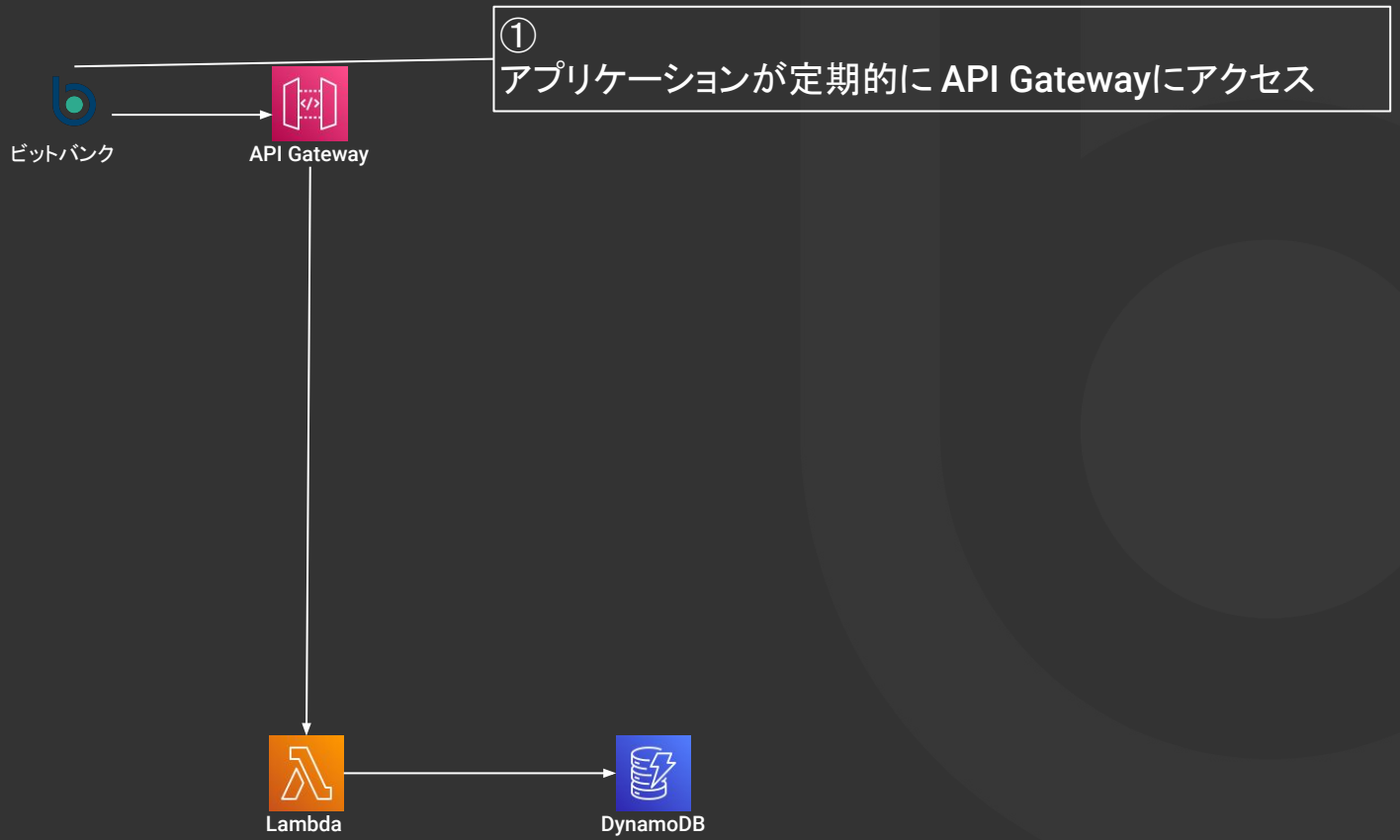
②
SQSに格納されたリクエストにより、Lambdaが起動

Redshiftのクエリ同時実行数を確認し、
・クエリ同時実行制限に該当する場合は SQSに戻す
・問題がなければStep Functionsを起動する

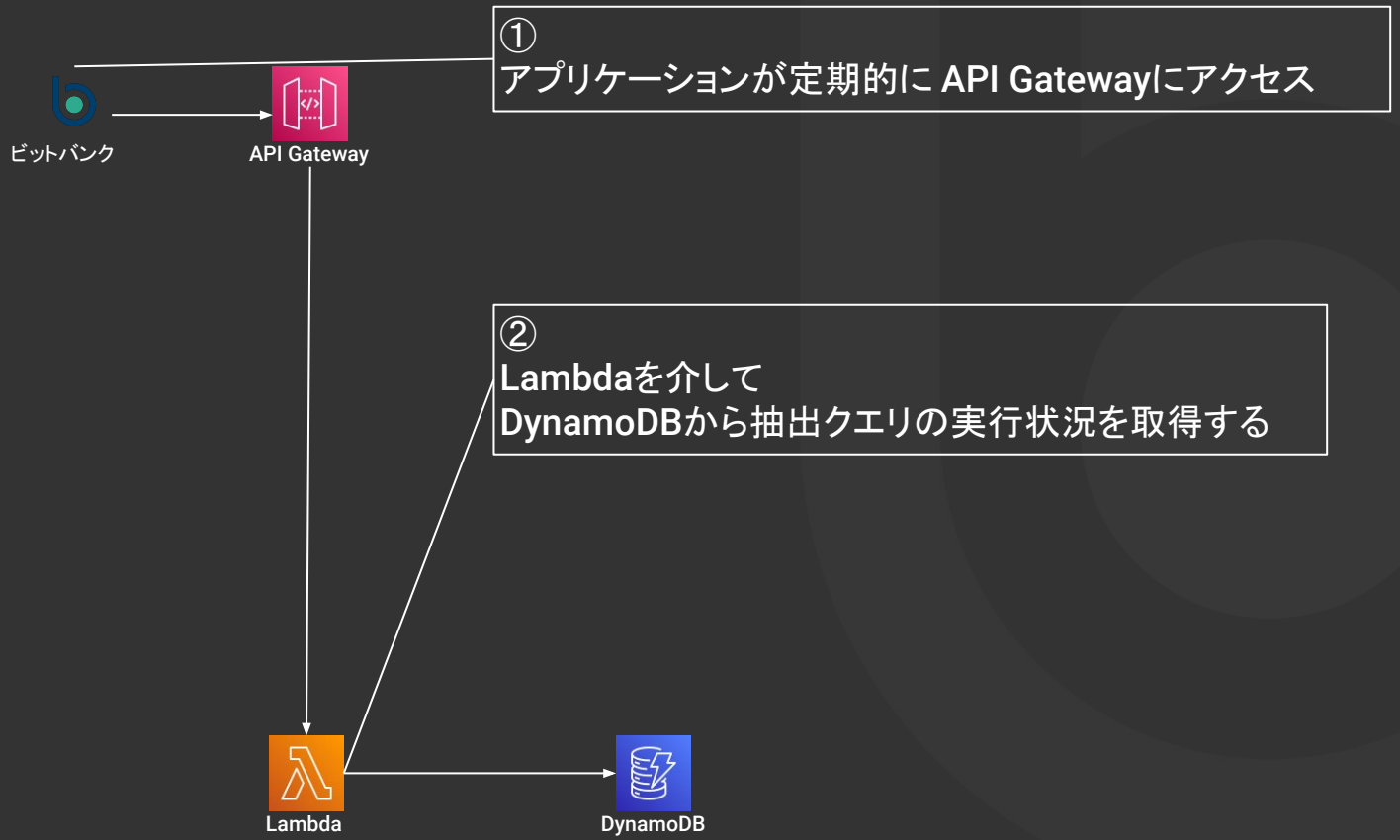
Amazon Redshiftを利用した顧客へのデータ提供 - データ抽出状況確認



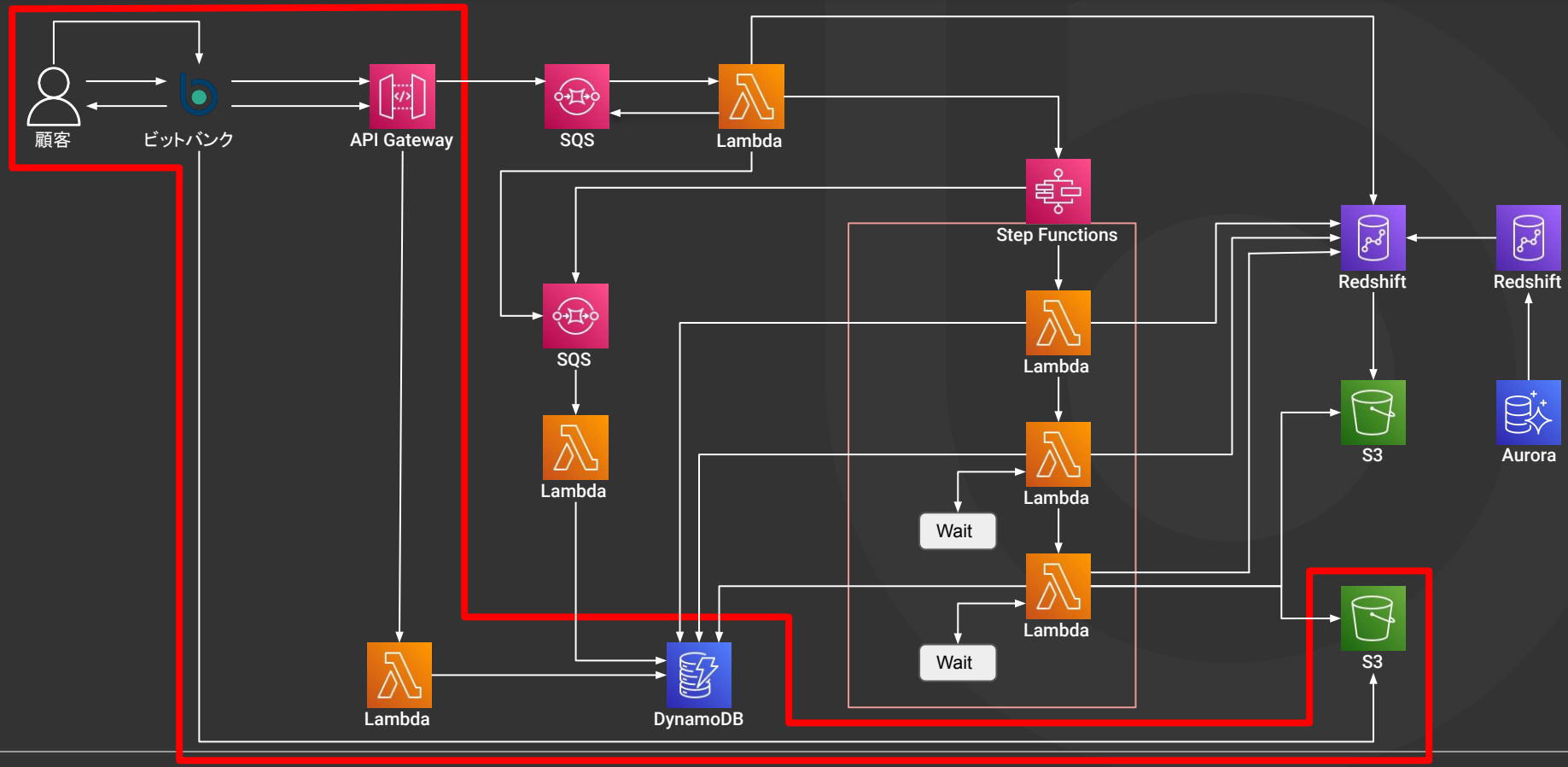
Amazon Redshiftを利用した顧客へのデータ提供 - データ抽出状況確認



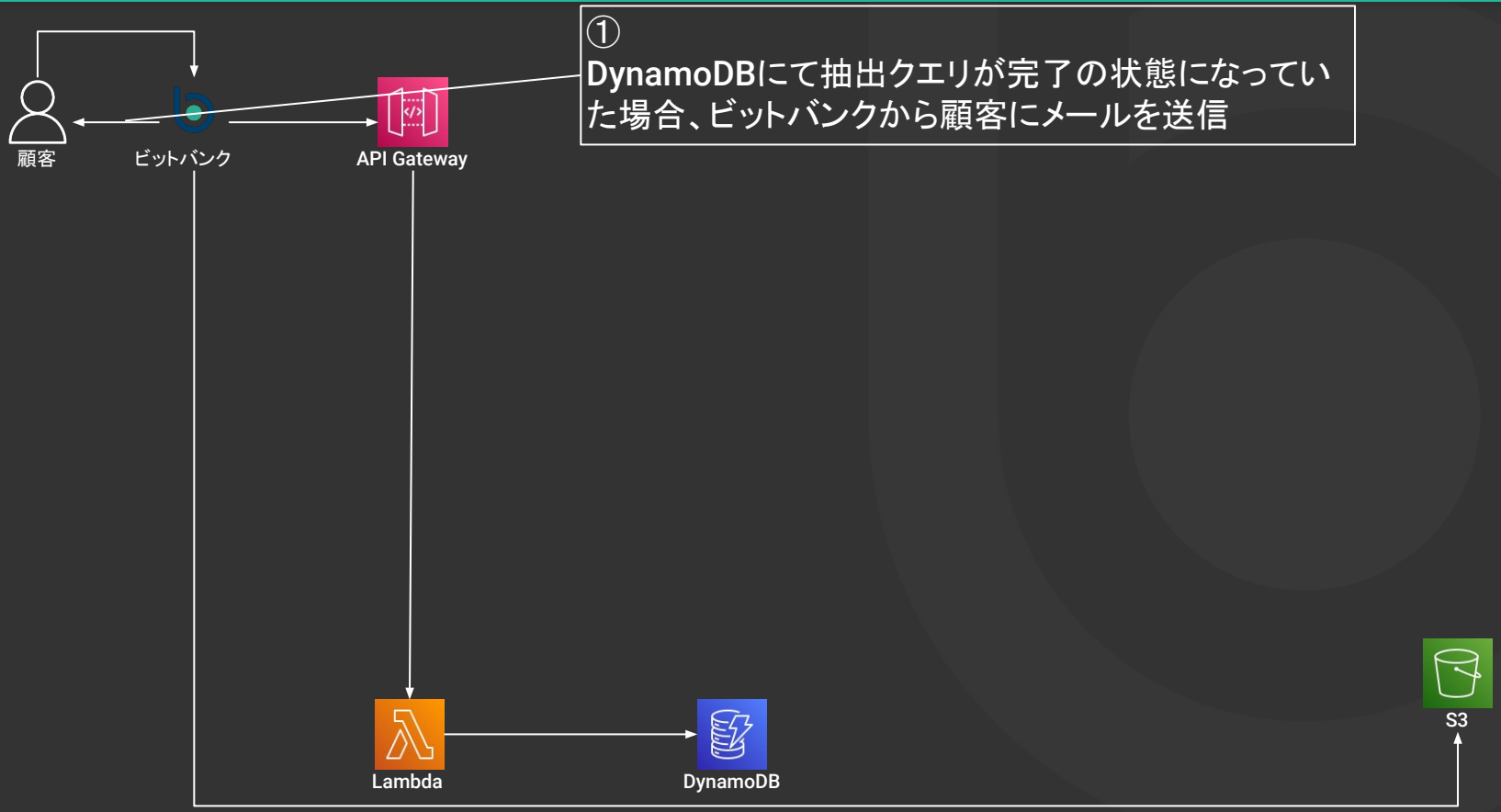
Amazon Redshiftを利用した顧客へのデータ提供 - データ抽出状況確認



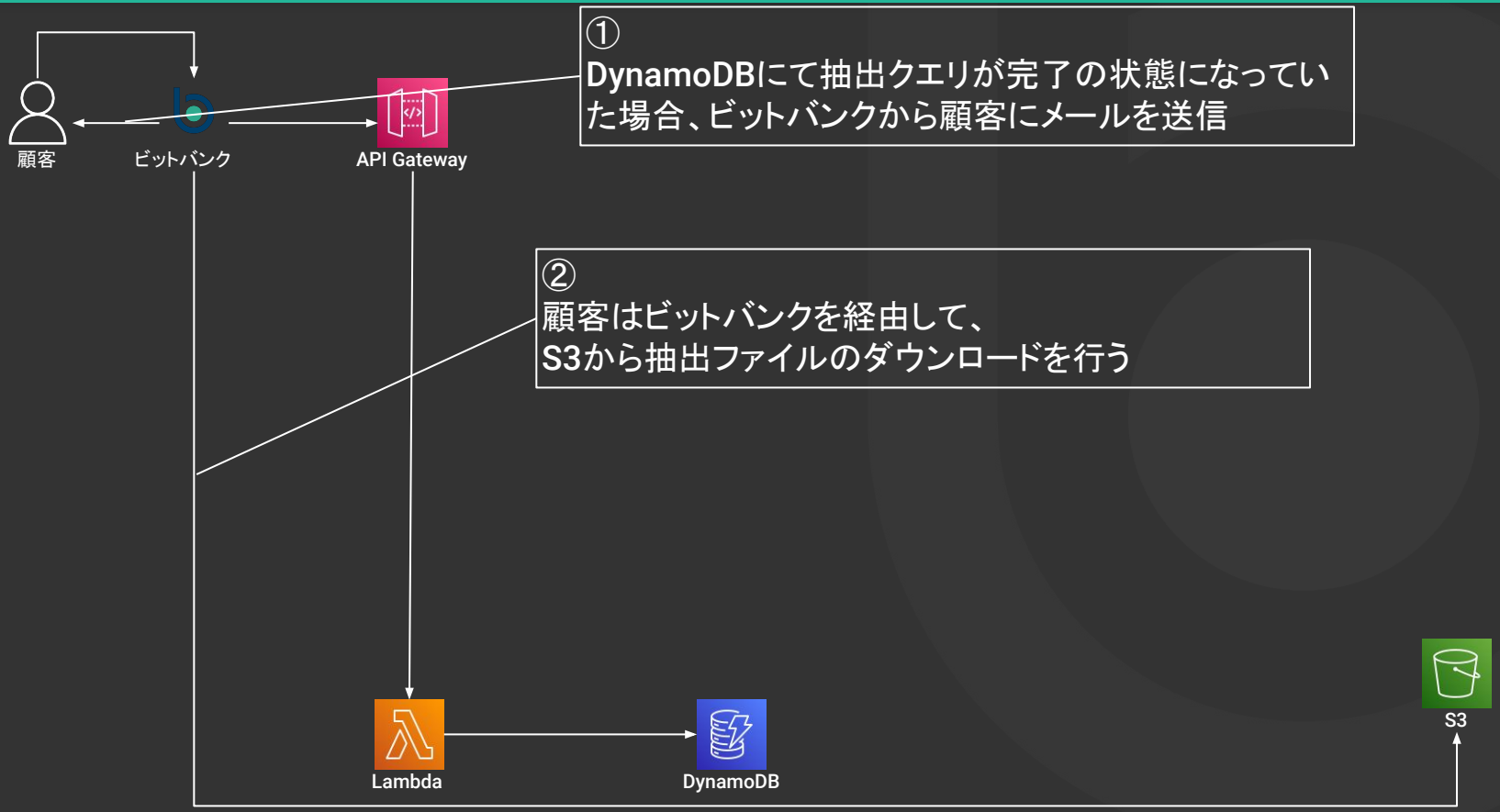
Amazon Redshiftを利用した顧客へのデータ提供 - データダウンロード



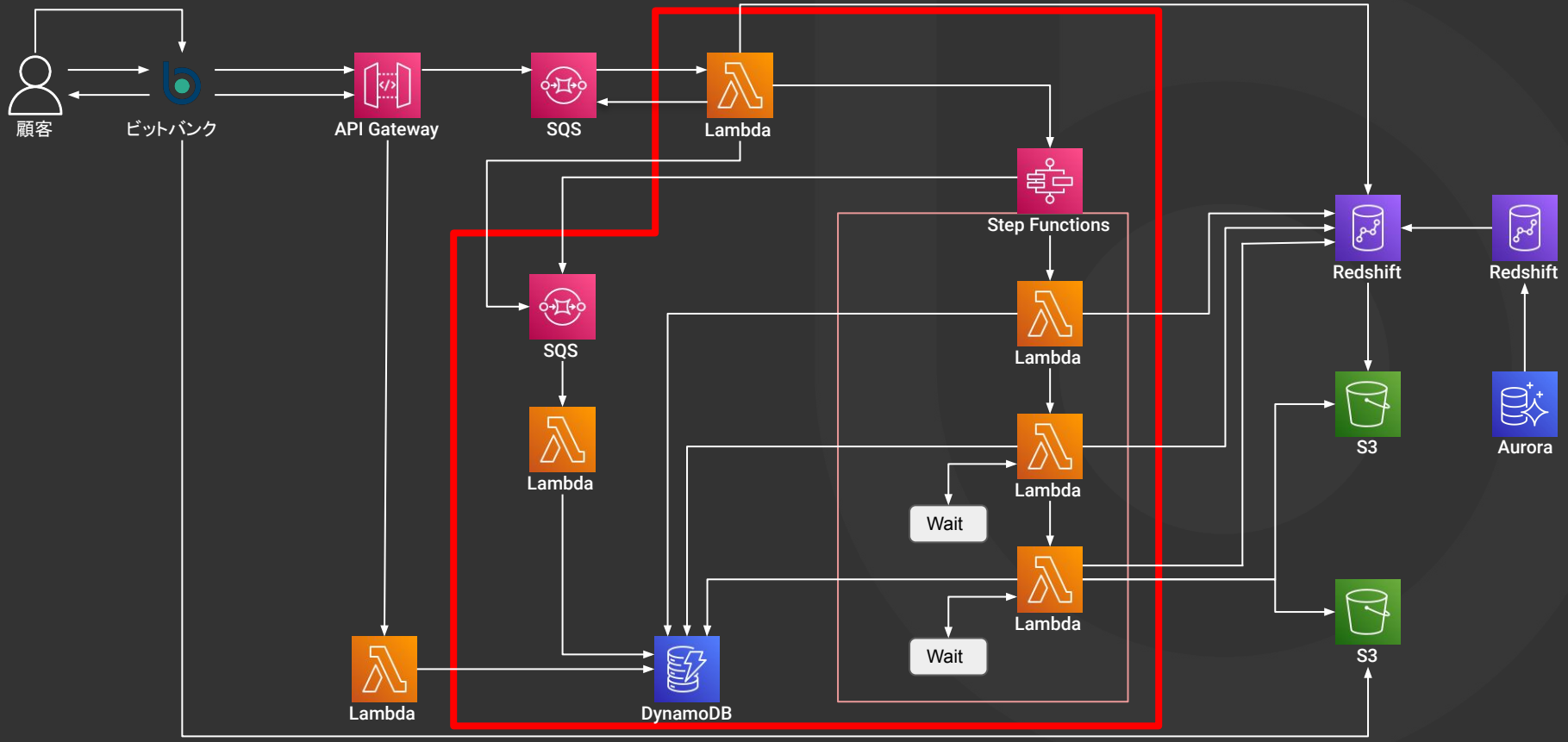
Amazon Redshiftを利用した顧客へのデータ提供 - データダウンロード



Amazon Redshiftを利用した顧客へのデータ提供 - データダウンロード

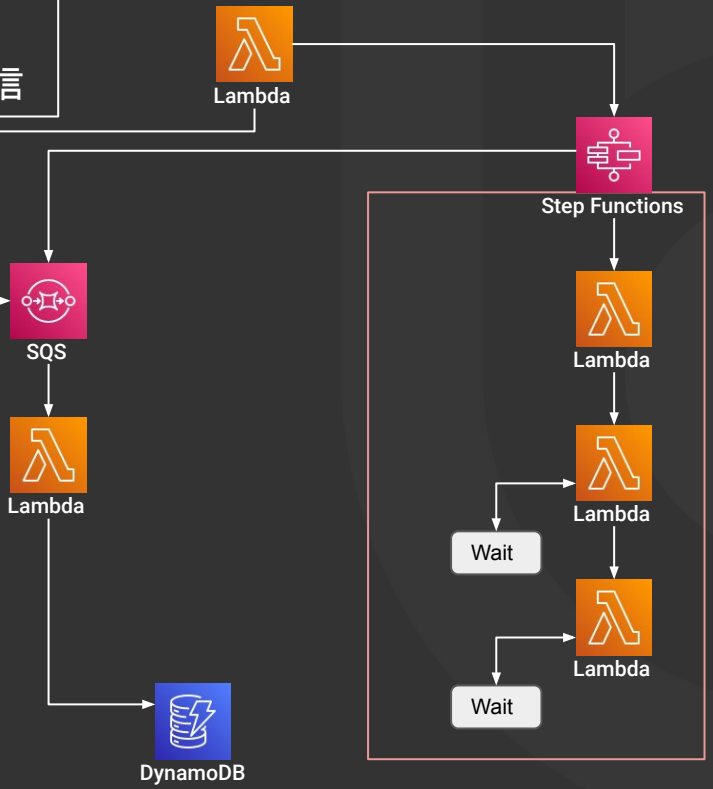


Amazon Redshiftを利用した顧客へのデータ提供 - エラーハンドリング



Amazon Redshiftを利用した顧客へのデータ提供 - エラーハンドリング

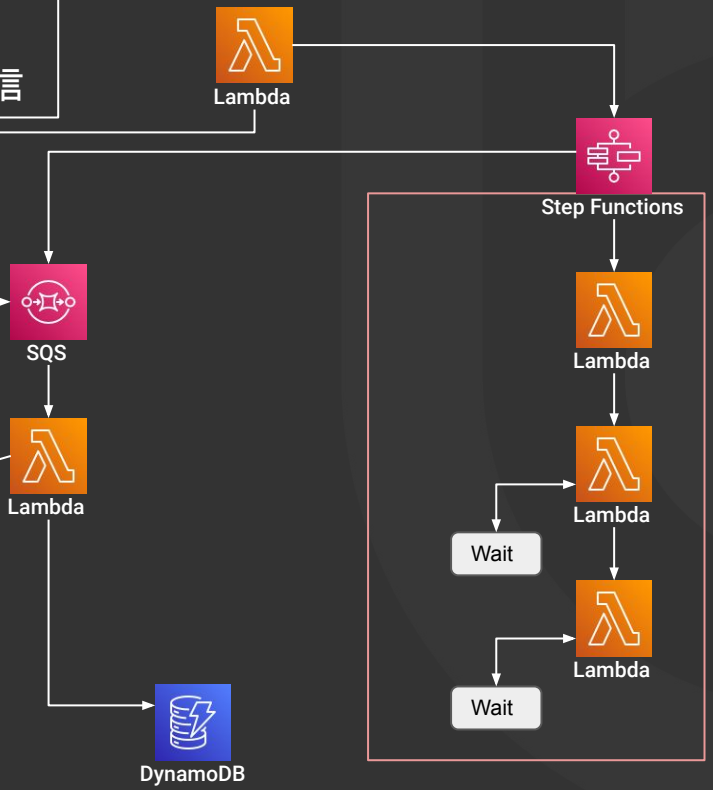
①
異常なリクエストを検知した場合、
Step Functionsを起動せずにSQSに送信



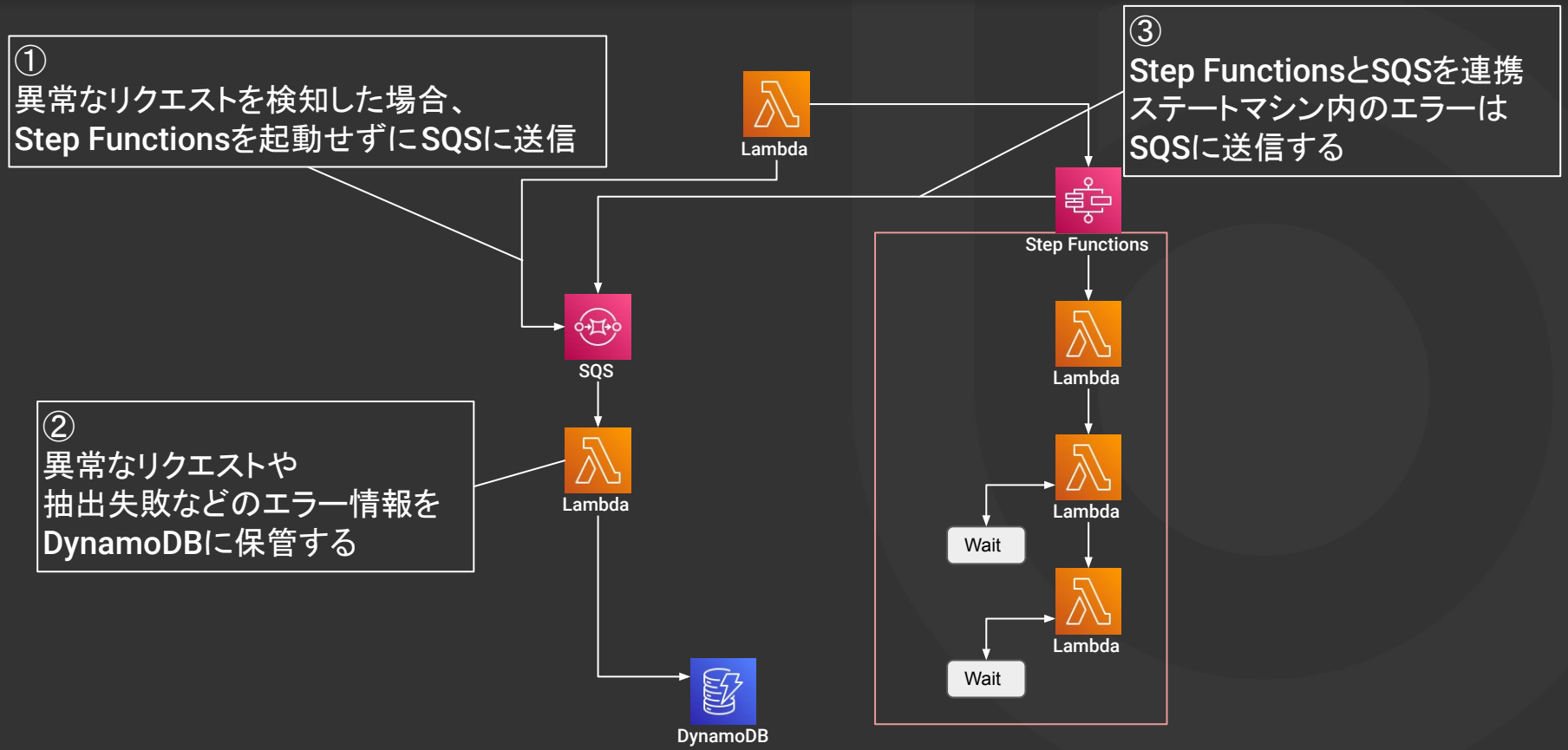
Amazon Redshiftを利用した顧客へのデータ提供 - エラーハンドリング

①
異常なリクエストを検知した場合、
Step Functionsを起動せずにSQSに送信

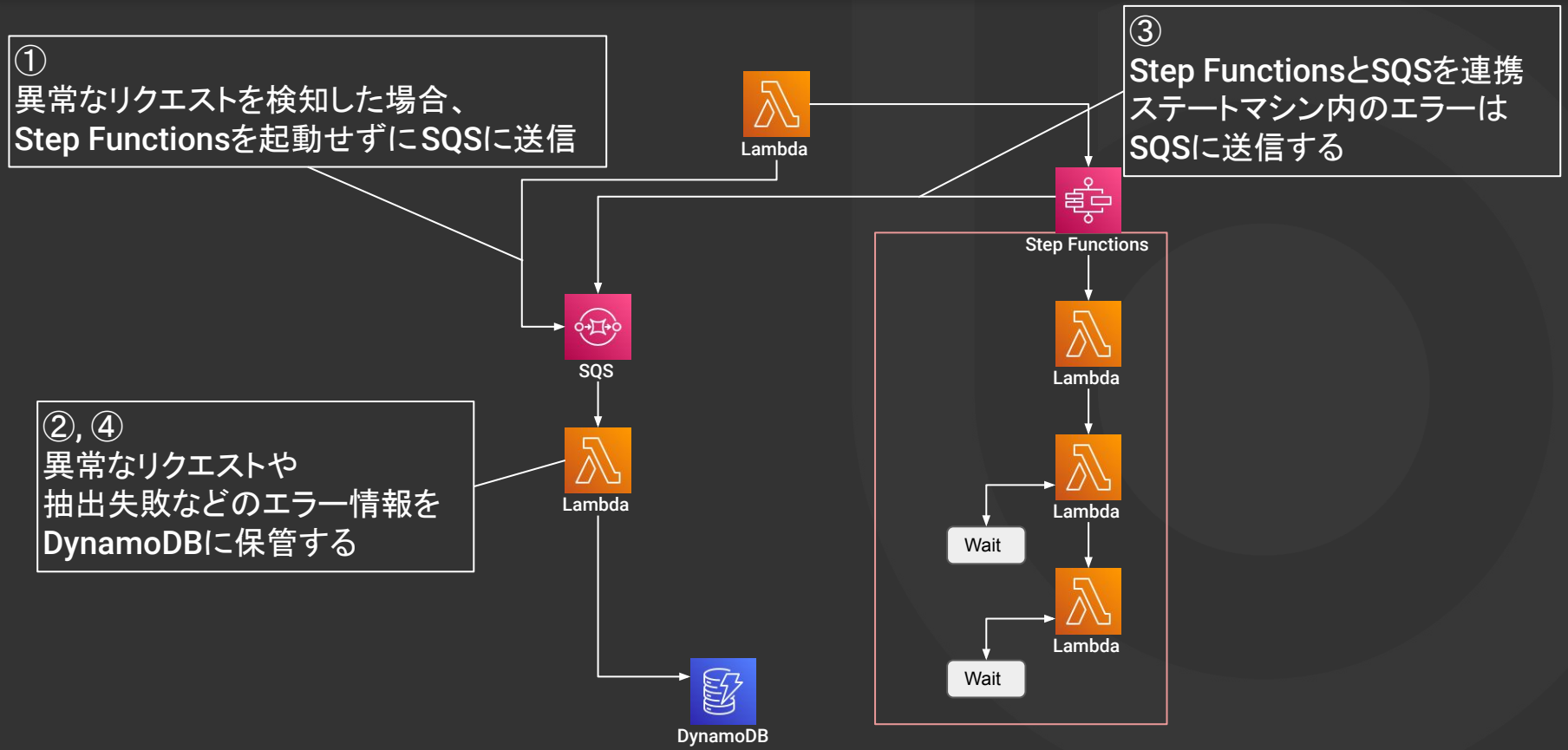
②
異常なリクエストや
抽出失敗などのエラー情報を
DynamoDBに保管する



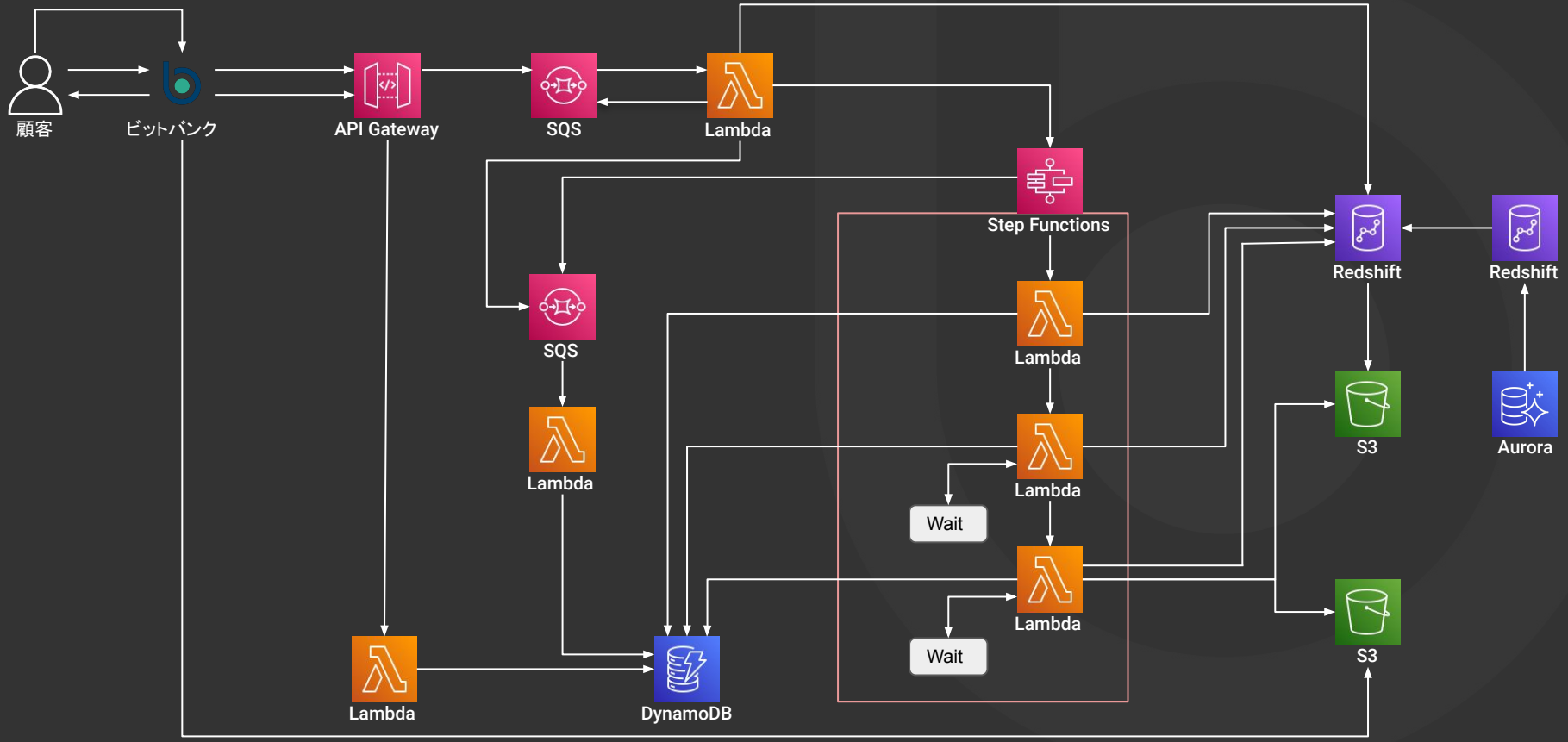
Amazon Redshiftを利用した顧客へのデータ提供 - エラーハンドリング



Amazon Redshiftを利用した顧客へのデータ提供 - エラーハンドリング



Amazon Redshiftを利用した顧客へのデータ提供 - 全体構成図



Amazon Redshiftを利用した顧客へのデータ提供

・抽出リクエストは通貨ペアの指定が可能

・抽出完了時にメールが届くためリクエストを送った後はメールを待つだけ

・抽出されたデータは1週間ダウンロード可能

Amazon Redshiftを利用した顧客へのデータ提供

現在の課題

・月1程度のAmazon Redshiftのパッチ提供への対応が必要

→ Amazon RedshiftのServerless化で解決できるかもしれない

・定期的に同期されるデータを利用しており、最新データの提供が出来ない

→ Federated Queryの利用を検討中

Redshiftを経由してAuroraからも情報を取得することで最新データまで提供

Amazon Redshiftを利用することで改善した点

1. データ抽出の負荷がサービスに影響を与えない
2. リクエストあたりの抽出上限の撤廃
3. サービスに利用中のAuroraからデータのパージが可能な状態に

まとめ

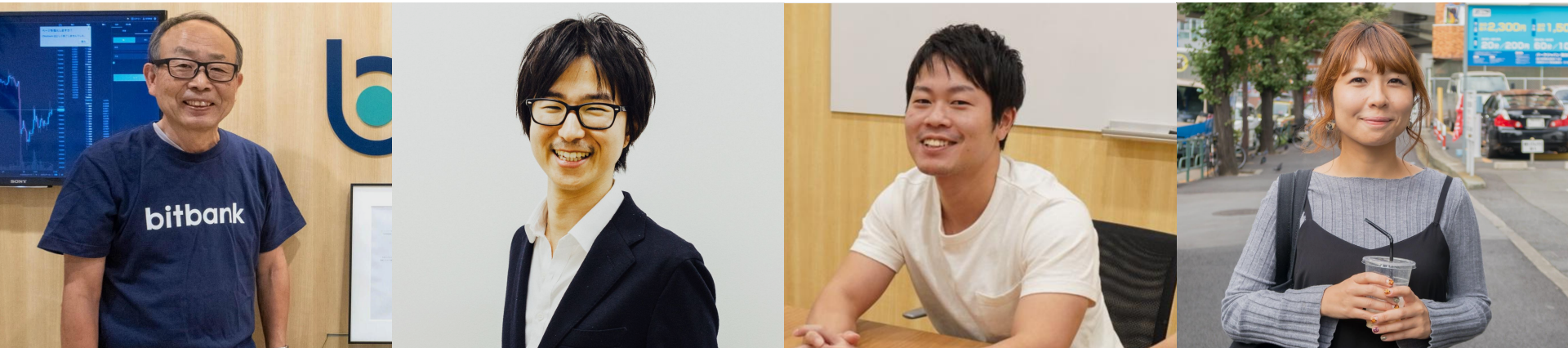
Redshift Data APIを使ってみて・・・

1. Redshift Data APIとStep Functionsは相性が抜群
2. 非同期的に大きなデータを容易に扱える点が魅力
3. クエリ結果が大きくなる場合はUnloadが便利

We Are Hiring!

ビットバンク 採用

検索



ビットコインの技術で
世界中にあらゆる価値を流通させる