10000店舗の自動発注システム構築

ゲンキー株式会社

2024年10月12日

10000店舗の自動発注システム構築

・自己紹介
・課題と解決策
　・当時の話
　・現状と理想、そして危機感から奇跡
・コンサル始動
・REAL始動
　・始動と３Ｋ
　（苦悩、苦労、苦渋）
　・利用した技術
　・RedShiftをシフト
　・現在進行中
　・課題
　・REAL開発まだまだ先は長い

昔話、
どうでもいい話・・・

技術系
結構？ちゃんとしてます。

10000店舗の自動発注システム構築

・自己紹介

ゲンキー株式会社

【戦略本部】

〒910-0332 福井県坂井市丸岡町下久米田38-33

ＩＴ本部ＳＥリーダー

はた　ひさゆき

畑　尚由記

最近の趣味は登山（もっぱら日帰り）

ジョギング※フルマラソン挑戦したい！

ＩＴ本部　REAL担当

おがわ　せいや

小川　誠也

趣味：将棋、麻雀

（最近はほぼ見る専）

10000店舗の
自動発注システム構築

10000店舗の自動発注システム構築

順位	チェーン名	2023年1月	2024年1月
1位	ウエルシア	3,533	3,677
2位	ドラッグスギ	2,282	2,467
3位	ツルハドラッグ	1,644	1,791
4位	クスリのアオキ	1,380	1,496
5位	ディスカウントドラッグコスモス	1,344	1,493

↑ドラックッストア　↓コンビニ

順位	チェーン名	2023年1月	2024年1月
1位	セブン・イレブン	21,111	21,248
2位	ファミリーマート	16,314	16,047
3位	ローソン	13,735	13,779

国内だけで、１万店舗

https://diamond-rm.net/management/224175/

10000店舗の自動発注システム構築

・課題と解決策

・当時の話

・現状と理想、そして危機感から奇跡

R(レギュラー)店

ランチェスタードミナント戦略

社会のインフラとチェーンストア企業への挑戦

10000店舗の自動発注システム構築

システムの老朽化
オーバーヒート

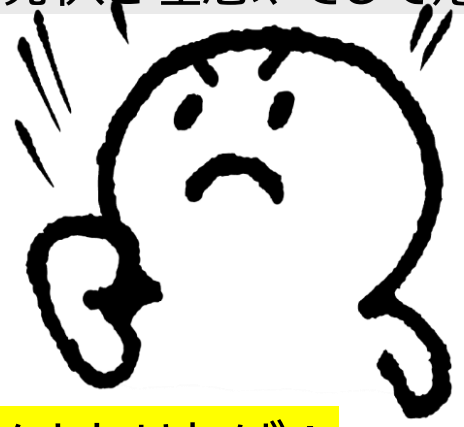
・課題と解決策
　・当時の話
　・現状と理想、そして危機感から奇跡

10000店舗の自動発注システム構築

・課題と解決策

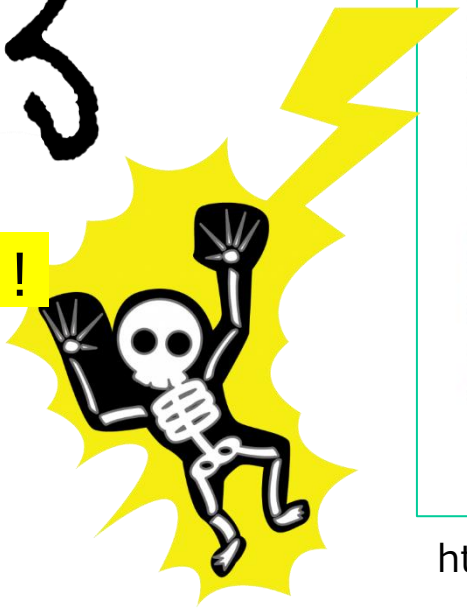
・当時の話

・現状と理想、そして危機感から奇跡



なんとかせねば！

ピンチをチャンスに！



ダイソー快進撃を支える「毎晩105億件
ータ処理」する需要予測システムはどう生
まれたか

AWS Summit 2019　アマゾン・ウェブ・サービス（AWS）　ダイソー

伊藤 有/Tamotsu Ito［編集部］
Jun. 17, 2019, 05:10 AM | ビジネス　58,152

Shutterstock

小売業にとって、在庫管理は店舗ごとの売り上げを最大化させるための重要な業務だ。

https://www.businessinsider.jp/post-192845

10000店舗の自動発注システム構築

・コンサル始動

前進あるのみ！

達成感！！

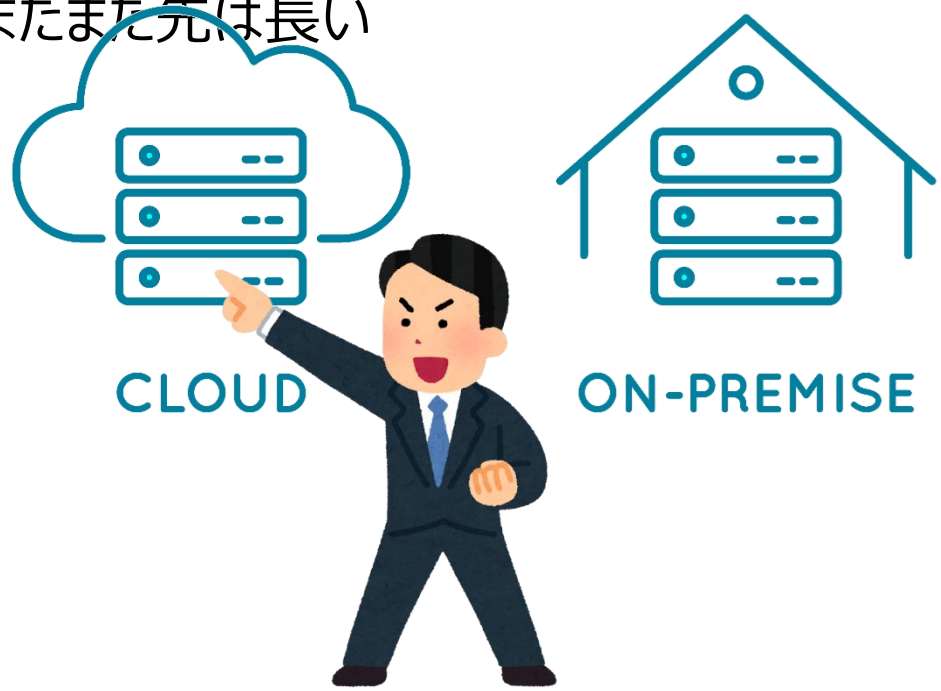
ご縁に感謝！

10000店舗の自動発注システム構築

・REAL始動
　・始動と３Ｋ（苦悩、苦
　・利用した技術
　・現在進行中
　・課題
　・REAL開発

10000店舗の自動発注システム構築

・REAL始動
  ・始動と３Ｋ（苦悩、苦労、苦渋）
  ・利用した技術
  ・課題
  ・現在進行中
  ・REAL開発まだまだ先は長い



10000店舗の自動発注システム構築

・REAL始動
  ・始動と３K（苦悩、苦労、苦渋）
  ・利用した技術
  ・課題
  ・現在進行中
  ・REAL開発まだまだ先は長い

10000店舗の自動発注システム構築

利用した技術

・**RedShift**
計算エンジンとして使用
・**Lambda**
変数の設定やファイル転送など
・**StepFunctions**
ジョブフローとして使用
・**DynamoDB**
Sfn制御のための変数などを格納
・**Glue**
SQLファイルを読み込んでクエリの実行
・（**CircleCI**）
GitHubと連動して自動デプロイ

・**S3**
バックアップデータ、スクリプト、
連携ファイルなどを格納
・**EventBridge**
Lambdaなどの定刻起動
・**Athena**
spectrumデータの分析
・**CloudFormation**
Sfn、Lambdaなどのリソースのデプロイ
・**Cloud9**
開発環境。デプロイにも使用
・**Secret Manager**
各DBユーザのパスワード管理

10000店舗の自動発注システム構築

・なぜRedShift？

Redshiftを選んだ理由 ⇒ **大量データの処理が早い**

データ件数	RDBMS 行指向	Amazon Redshift 列指向
１万	０．０６秒	３秒
１００万	２秒	４秒
１億	３６秒	５秒
１００億	２，０４０秒	６秒

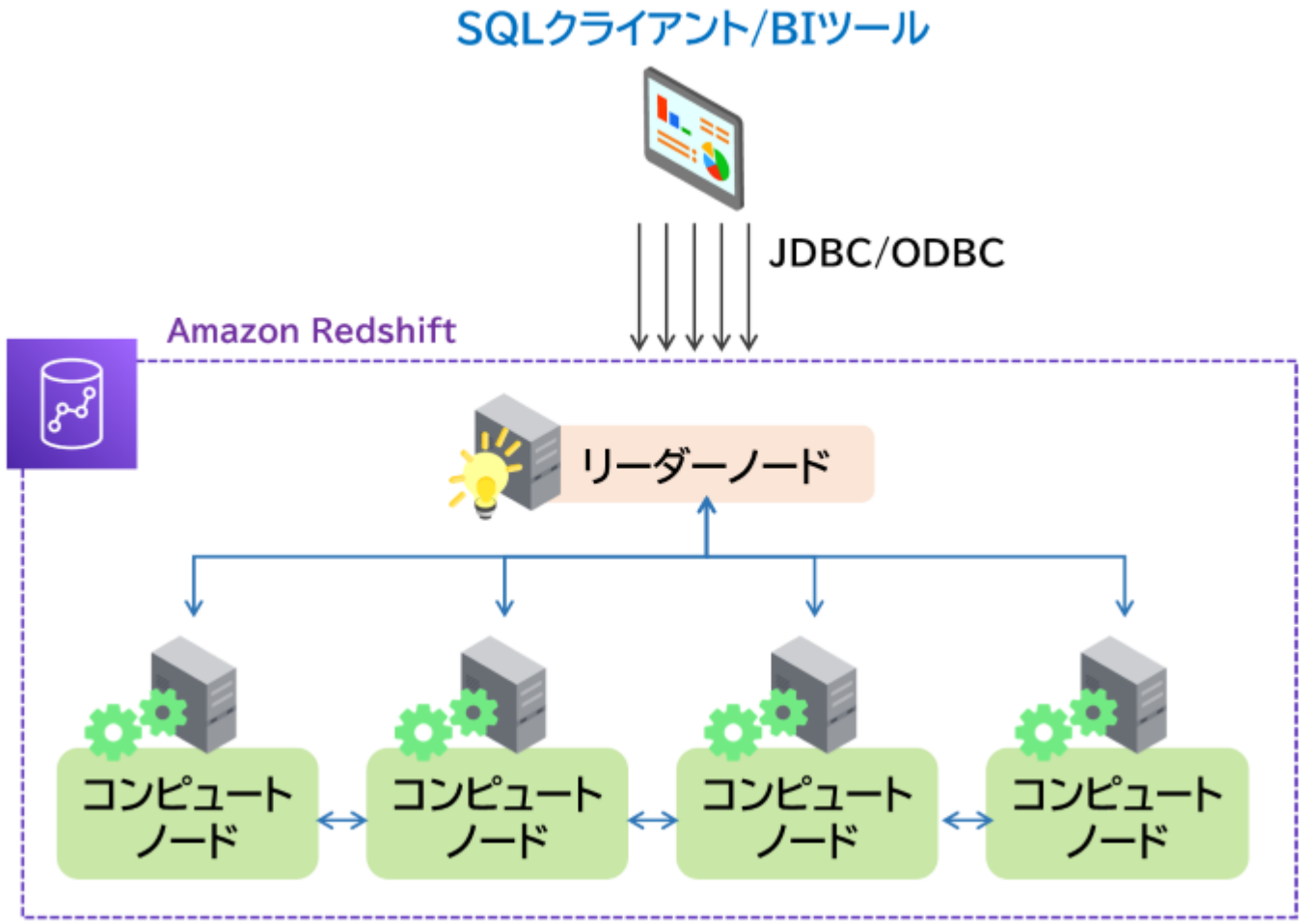
10000店舗の自動発注システム構築

・なぜRedShift？

比較	RDBMS 行指向	Amazon Redshift 列指向	Amazon DynamoDB Key-value
扱いやすさ	○ 設計が簡単	▲ 設計にコツがいる	× できることが限定
処理 (データ小)	○ 早い	▲ 普通	× (処理するものでない)
処理 (データ大)	× 遅い	○ 早い	× (処理するものでない)
拡張性 スケールアウト	× できない	▲ 準備が必要	○ スパイクに対応可能
並列アクセス	▲ 普通	× 苦手	○ アクセス

10000店舗の自動発注システム構築

・RedShiftの仕組み



https://www.ashisuto.co.jp/db_blog/article/aws-redshift.html

10000店舗の自動発注システム構築

・REAL始動
  ・始動と３Ｋ（苦悩、苦労、苦渋）
  ・利用した技術
  ・課題
  ・現在進行中
  ・REAL開発まだまだ先は長い

10000店舗の自動発注システム構築

課題

・障害対策

発注システムのため、”障害＝発注なし”となって大損害となる可能性。

それを回避するための方策は？

10000店舗の自動発注システム構築

障害対策

① 予備発注システムの整備

前日のうちに翌日の発注データを作成しておく。

精度は部門にもよるが80~90%くらい

② 障害が起きない仕組みへ

機能的なバグを除けば、RedShift固有の仕様に悩まされたことが多い。

実行計画が急に変わって特定のクエリが終わらずに止まってしまう、

Diskを大量に食うような処理に変わってDiskFullが起こってしまう、など。

→対策は、チューニング以外にない

10000店舗の自動発注システム構築

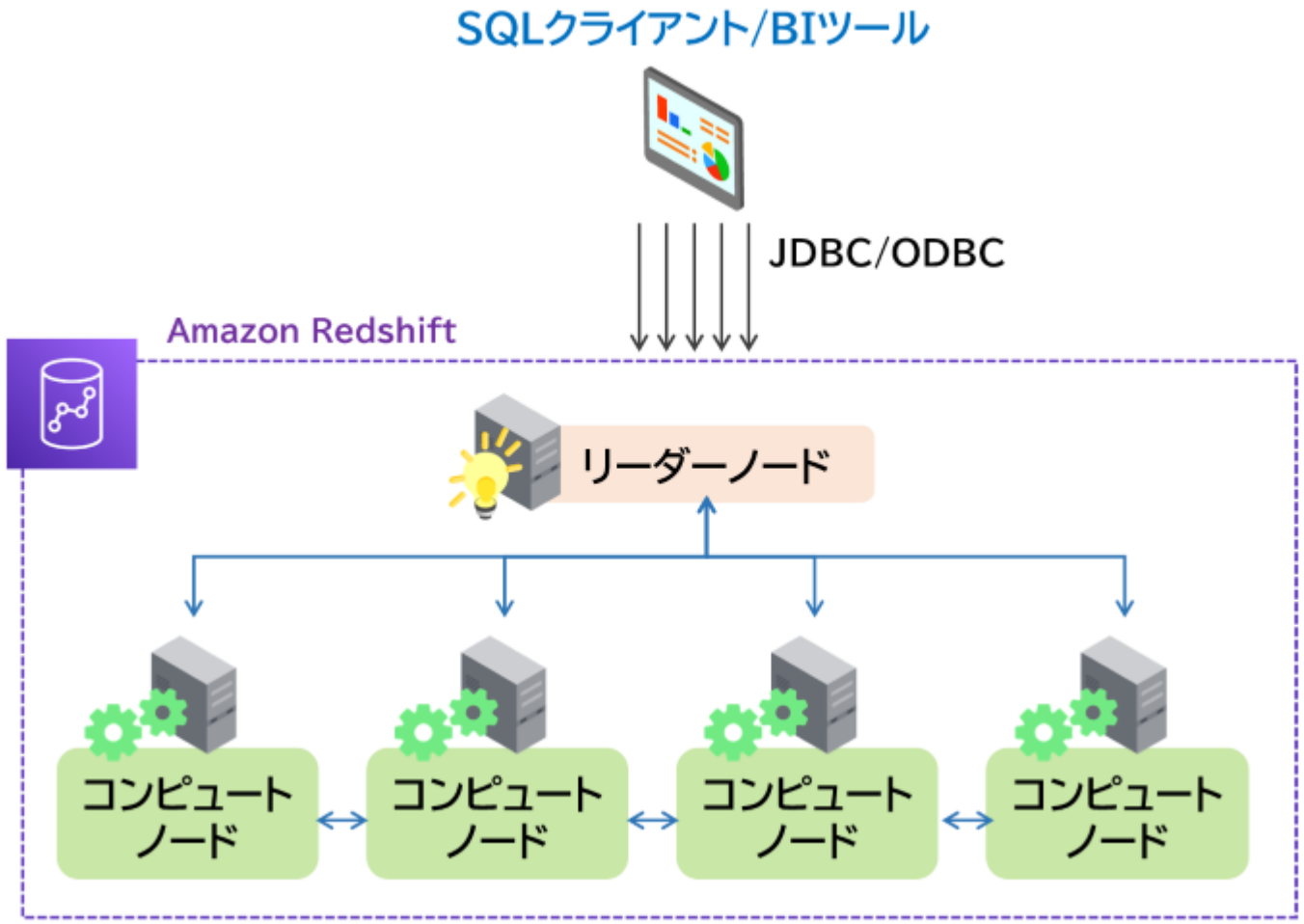
障害対策-チューニング

前提：RedShiftの実行計画はオプティマイザが自動で判断しているため、
ユーザ側で制御できるものではない。

その上で、ユーザ側ができる対策は大きく分けて2つ
①Dist KeyとSort Keyを結合するテーブル間でそろえる
②サブクエリを大量につなげているクエリでは、サブクエリを実テーブルに展開する

10000店舗の自動発注システム構築

・RedShiftの仕組み（再掲）



https://www.ashisuto.co.jp/db_blog/article/aws-redshift.html

10000店舗の自動発注システム構築

①Dist KeyとSort Keyを結合するテーブル間でそろえる

Real設計方針「ソートキーをなるべく合わせる」

最速の結合方法であるMERGE JOINが選択されるようになる。
なるべく早く、店舗商品に分解するように中間テーブルを設計する。

ソートは MERGE JOIN を有効にしますか？

以下の条件をすべて満たすとき、
最も高速な結合操作である
MERGE JOIN が行われる

- **2つのテーブルで同じソートキーが指定され、同じ列で分散されている**
- どちらのテーブルも80%以上ソートされている
- 2つのテーブルがJOIN条件でDISTKEY列とSORTKEY列の両方を使用して結合されている

https://www.slideshare.net/AmazonWebServicesJapan/aws-black-belt-online-seminar-2017-amazon-redshift

→最速のMERGE JOINにするためにテーブル設計を行う必要あり

10000店舗の自動発注システム構築

②サブクエリを大量につなげている場合は、サブクエリを実テーブルに展開する

サブクエリを大量につなげたクエリでは、ワーク領域を大量に消費してしまうため、
RedShiftの処理を圧迫してしまう可能性がある。

→原因がよくわからないときに有効。
実テーブルではKeyを明示的に指定できるので、
Joinがしやすくなっているのかもしれません。

あとは、（私の感覚上での話ですが）サブクエリの中でもSortをしておくと、
クエリパフォーマンスがよくなる気がします。

10000店舗の自動発注システム構築

・REAL始動
  ・始動と３Ｋ（苦悩、苦労、苦渋）
  ・利用した技術
  ・課題
  ・現在進行中
  ・REAL開発まだまだ先は長い

10000店舗の自動発注システム構築

現在進行中

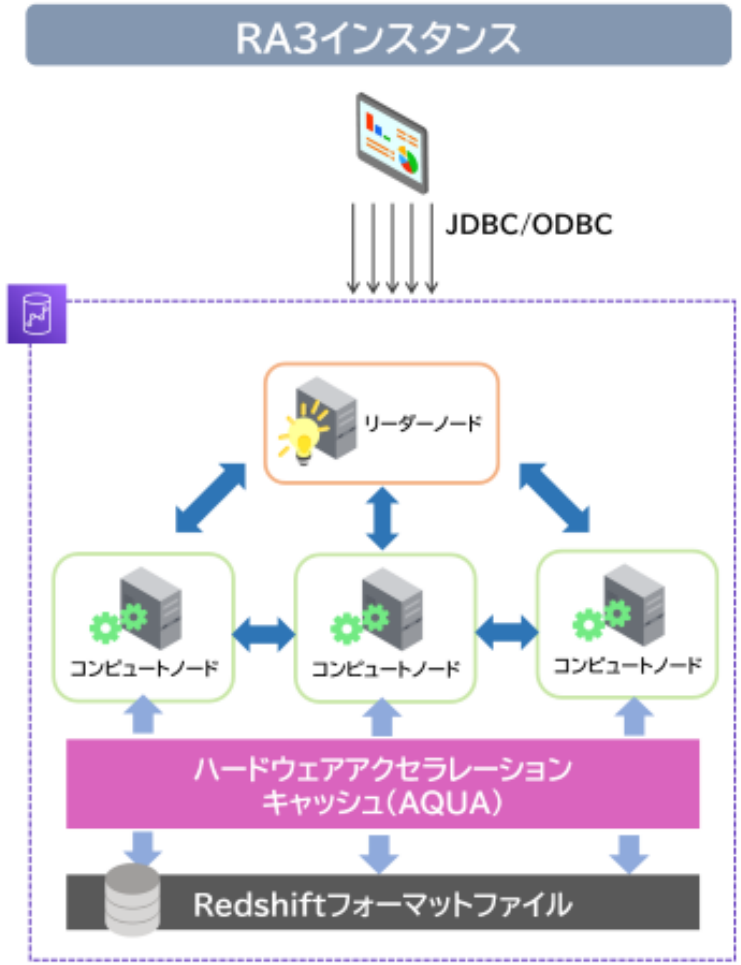
・DC2インスタンスが2024年末で廃止！？

・RedShiftの定期メンテナンスでバージョンが上がると、
クエリの再コンパイルが走って処理が遅延してしまう。
普段と比べて1時間ほど合計で遅れてしまう
2週間に1度ほどのペースでこの事象が起こっていたため、早急に対策が必要

・コスト
→これらを解決する手段として
①RA3インスタンスへの移行・・・性能を上げて処理遅延を軽減
②Serverlessへの移行・・・バージョンアップの影響を受けにくくする
を検討。

10000店舗の自動発注システム構築

RA3インスタンスとは



DC2インスタンスは各コンピュートノード内にSSDストレージを持っているが、
RA3インスタンスはS3のデータ領域を利用。
必要な処理能力に応じてスケールアウトが可能。

https://www.ashisuto.co.jp/db_blog/article/aws-redshift.html

10000店舗の自動発注システム構築

Serverlessとは？

ノード数やインスタンスタイプを綿密に設定しなくても、
実行時に自動でスケールしてくれる、というサービス。
名前空間とワークグループというコンポーネントにより管理される。

名前空間　　・・・DB名、DB管理者パスワード、
IAMロール、暗号化とセキュリティなど
ワークグループ・・・ネットワークとセキュリティ

RPUという処理ユニットの容量を設定するだけで、自動で実行時にスケールしてくれる。簡単に使いやすいのが特徴。

10000店舗の自動発注システム構築

各インスタンス性能、コスト比較

	1日あたり実行時間	1日あたりコスト
dc2.large ノード6	4時間10分	10$
RA3.xplus ノード3	3時間50分	20$
Serverless　RPU8	3時間20分	15$
Serverless　RPU16	2時間30分	20$

※あくまで弊社環境での感想です

・RA3インスタンス▲

→そんなに早くならないわりにコストが高い。バージョンアップの影響も受ける。

・Serverless◎

→何回か回すごとに早くなっていく。バージョンアップの影響もあまり受けない

→Serverlessへ移管

10000店舗の自動発注システム構築

Serverless移管について

〇データ移行コスト（工数）

DC2、RA3のインスタンスのスナップショットから復元できるため、

データの規模にもよるが30分程度で移行可能。

〇注意点

・Serverless は3AZのサブネットが必要なので、2AZ構成でクラスタを使用していた場合は新たに異なるAZのサブネットを用意する必要がある

・Serverlessでは参照できるシステムテーブルが違うので、システムテーブルを参照する処理があった場合は、変更する必要がある

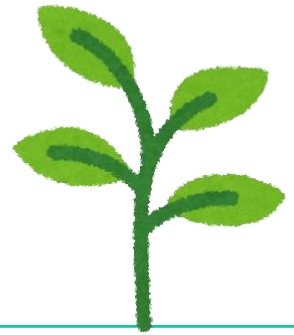
例）Serverless だと stl_load_errors にアクセスできない

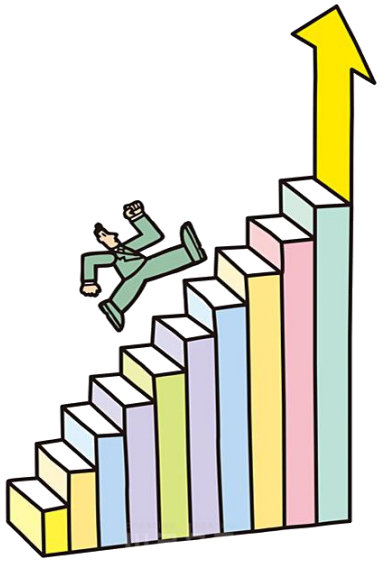
・長時間稼働がない際、再度使用する際に接続エラーが出る可能性がある

コンピューティングリソースは、アクティビティがない場合はバックグラウンドで自動的にシャットダウンし、データをロードしているときやクエリを受信すると再開します。

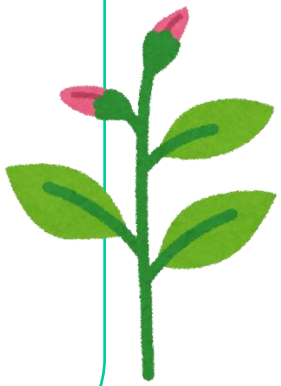
10000店舗の自動発注システム構築

・REAL始動
  ・始動と３Ｋ（苦悩、苦労、苦渋）
  ・利用した技術
  ・課題
  ・現在進行中
  ・REAL開発まだまだ先は長い





予測精度の向上
AIの活用
最適在庫の計算
などなど



10000店舗の自動発注システム構築

最後に

ご清聴ありがとうございました。

E n d