Amazon Redshiftを
1年運用して感じた教訓とその対策方針

2022/03/24 15:05 - 15:35
AWS で実践！Analytics Modernization ～事例祭り編～

面白法人カヤック　池田将士　@mashiike

アジェンダ

- 自己紹介・会社紹介・事業紹介
- Redshift導入背景と採用理由
- 1年運用して感じた教訓とその対策
  - Ep1: 開発環境に関して (教訓1)
  - Ep2: Redshiftの新機能を試したい (教訓2)
  - Ep3: Clusterのお引越し (教訓3 ~ 5)
- まとめ

自己紹介

池田 将士 (@mashiike)

面白法人カヤック 技術部 SREチーム所属

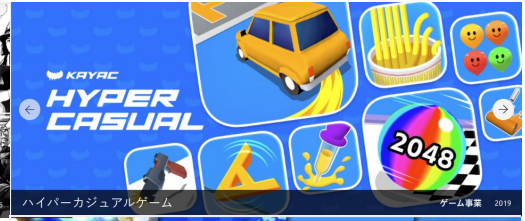
データにまつわる技術を担当

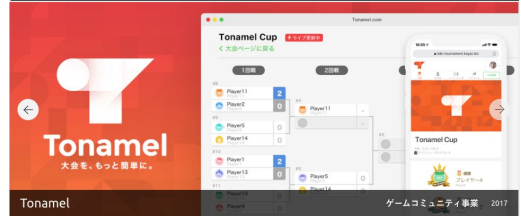
2016年末に新卒社員として入社
現在は『Tonamel』のデータ基盤を開発・運用

好きなAWSサービス

- Amazon S3
- Amazon Kinesis Data Firehose
- Amazon Redshift

会社紹介





面白法人カヤック

鎌倉の地にて、主にWeb技術を用いて
人の印象に深く残るような面白コンテンツを作る会社

ゲームからWebサービス、ミュージアムetc…
様々なことに挑戦

Tonamel

事業紹介

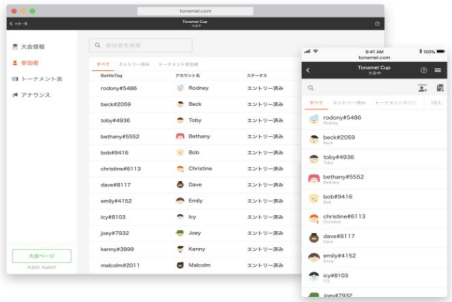
大会プラットフォームサービス『Tonamel』

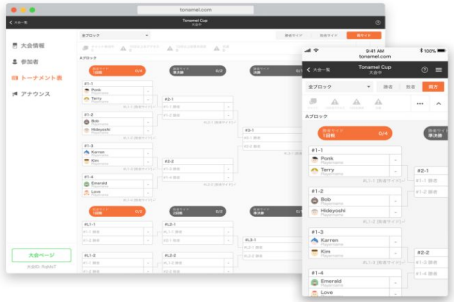
2017年7月7日:
『Lobi tournament』としてリリース
2020年4月1日:
『Tonamel』にサービス名変更

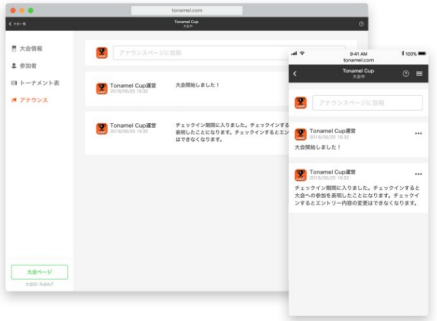
2021年2月: Redshift 導入

大会主催者の皆さん、
お待たせしました！

Tonamelを使えば、誰でも簡単に大会を開催・運営できます。エントリーから大会終了までに必要な機能が揃っています。







Redshift導入背景と採用理由

導入背景

- 本番Amazon Aurora MySQL＋Redash分析環境
- プロダクトアプリケーションがマイクロサービス化

採用理由

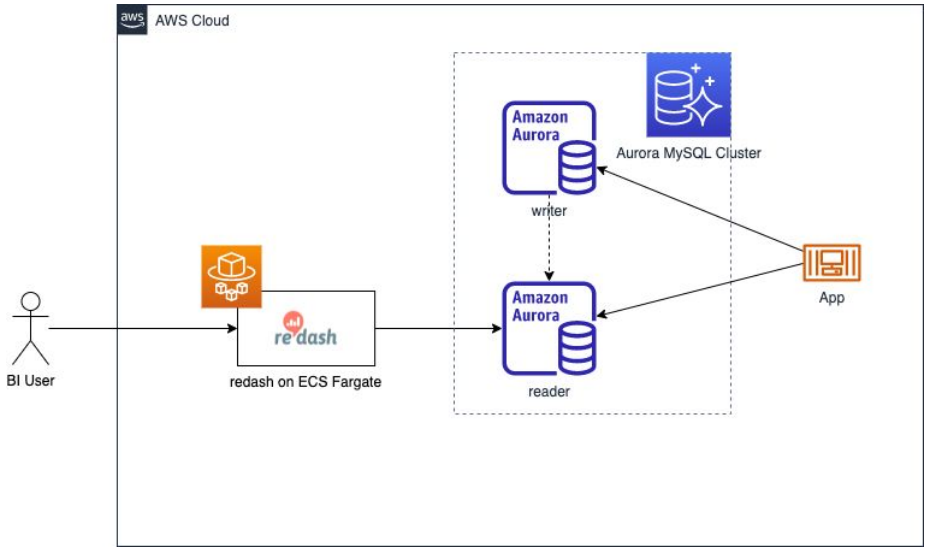
- SUPER型による半構造化データの取り扱い
- Redshift Federated Query for MySQL compatible
- 社内導入実績

Tonamel

導入背景: 従来の分析環境

モジュラモノリスなアプリケーションが使用しているAurora MySQLのReaderへ直接BIツールが接続

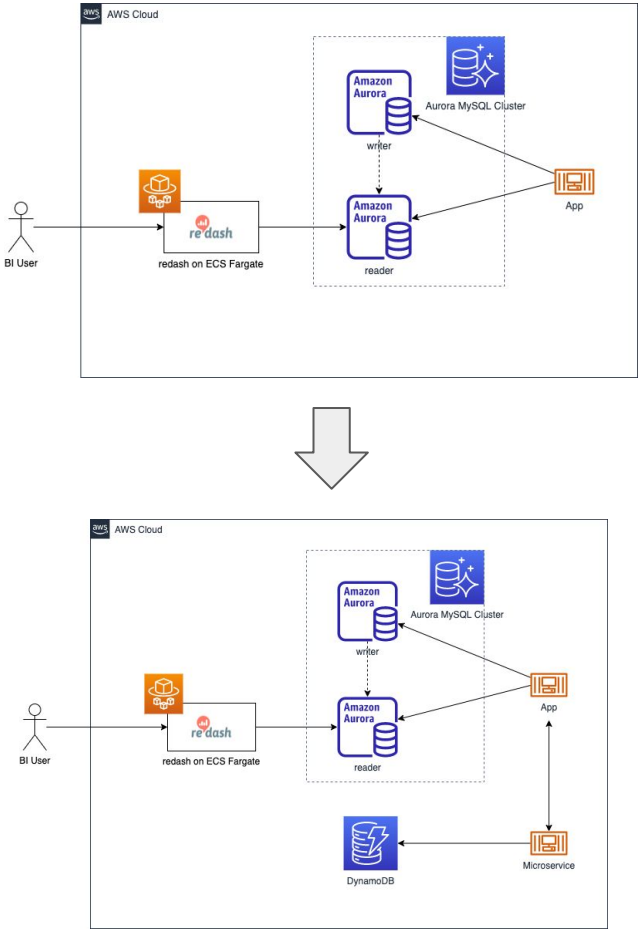
Window関数やWITH句は使えず、分析の複雑化への対応が困難



導入背景: アプリケーションのマイクロサービス化

- アプリケーションの一部機能が
  マイクロサービス化
- DynamoDBの導入により
  データのサイロ化が加速
- 今後はマイクロサービスを
  増やしていく方針

本格的なデータ基盤の必要性



近代的なデータ基盤の必要性とその中核の選択肢


Amazon Athena


Amazon S3


Amazon Redshift


Google BigQuery

近代的なデータ基盤の必要性とその中核の選択肢


Amazon Athena


Amazon S3

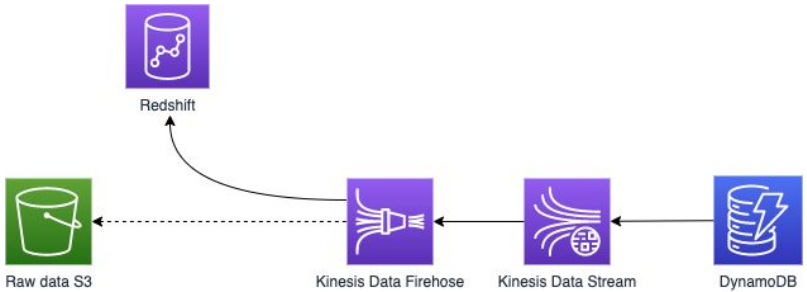

Amazon Redshift

主に3つの理由により採用


Google BigQuery

採用理由: SUPER型

- DynamoDBの変更情報は
  Kinesis Data Stream Adapterで
  容易に収集可能
- Kinesis Data FirehoseでSUPER型を
  用いたテーブルに直接コピーすればとて
  も取り扱いやすい




```
CREATE TABLE "source__ddb"."stream_event" (
  "awsRegion"     varchar ENCODE ZSTD,
  "dynamodb"      super   ENCODE ZSTD,
  "eventID"       varchar ENCODE ZSTD,
  "eventName"     varchar ENCODE ZSTD,
  "userIdentity"  varchar ENCODE ZSTD,
  "recordFormat"  varchar ENCODE ZSTD,
  "tableName"     varchar ENCODE ZSTD,
  "eventSource"   varchar ENCODE ZSTD
)
DISTSTYLE AUTO SORTKEY AUTO ENCODE AUTO;
```

採用理由: SUPER型

- Dyn
  Kine
  容易
- Kine
  用い
  も取

PartiQL言語によるSQL互換の直感的なデータアクセスが可能

```
1 select
2     "eventID" as event_id
3     ,"eventName" as event_name
4     ,'epoch'::timestamp +
5         ("dynamodb"."ApproximateCreationDateTime"::bigint)/1000 *
6         interval '1 second' as event_at
7     ,"dynamodb"."Keys"."HashKey"."S"::varchar as hash_key
8     ,"dynamodb"."Keys"."SortKey"."S"::varchar as sort_key
9     ,"dynamodb"."NewImage" as new_image
10    ,"dynamodb"."OldImage" as old_image
11 from source__ddb.stream_event
```

Result 1 (100)

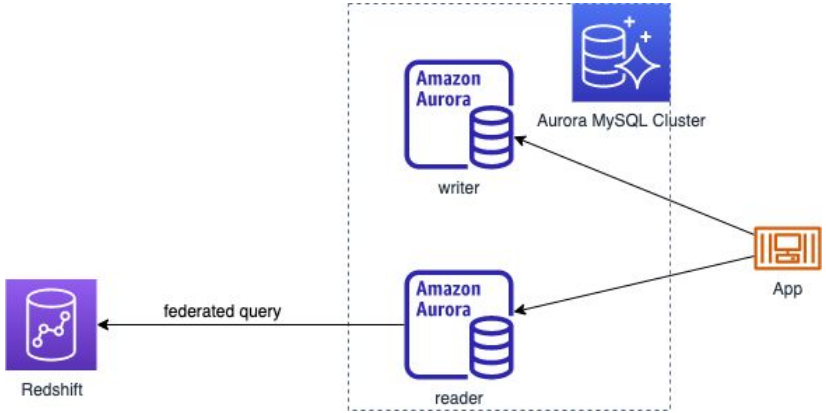
ev...	event_name	event_at	h.	s.	new_image	old_image
046c...	MODIFY	2022-01-14 02:02:11	h...	C...	{"Settings":{"M":{"Bat...	{"Settings":{"M":{"Battl...
074e...	INSERT	2022-01-14 02:02:04	h...	M...	{"ProcessStartEpoch...	NULL
30cf4...	MODIFY	2022-01-14 02:02:11	h...	P...	{"LatestMatchupID":{...	{"LatestMatchupID":{"S...
3c8b...	MODIFY	2022-01-14 02:02:05	h...	M...	{"Deadline":{"N":"0"},...	{"Deadline":{"N":"1642...
430fc...	INSERT	2022-01-14 02:02:05	h...	S...	{"BlockID":{"S":"0636...	NULL
4830...	MODIFY	2022-01-14 02:02:06	h...	C...	{"Settings":{"M":{"Bat...	{"Settings":{"M":{"Battl...
52f82...	INSERT	2022-01-14 02:02:01	h...	C...	{"Settings":{"M":{"Bat...	NULL
a563...	MODIFY	2022-01-14 02:02:11	h...	M...	{"Status":{"N":"0"},"C...	{"Status":{"N":"0"},"Co...

DynamoDB

Tonamel

採用理由: Federated Query

- Aurora MySQLへの負荷は問題なし
  （現行BIが直接接続してるため）
- 今後のマイクロサービスの増加で
  Aurora MySQLのデータソースが増える
  可能性が濃厚
- Federated Queryを使用することで、
  ETLパイプラインの簡略化を期待




```
1 CREATE EXTERNAL SCHEMA federated_for_mysql
2 FROM MYSQL
3 DATABASE 'main'
4 URI 'app.cluster-ro-xxxxxxxxxxxx.ap-northeast-1.rds.amazonaws.com'
5 IAM_ROLE 'arn:aws:iam::000000000000:role/redshift-federated'
6 SECRET_ARN 'arn:aws:secretsmanager:ap-northeast-1:000000000000:secret:prod/rds/redshift-federated-XXXXXX';
```

採用理由: 社内導入実績とAWS親和性

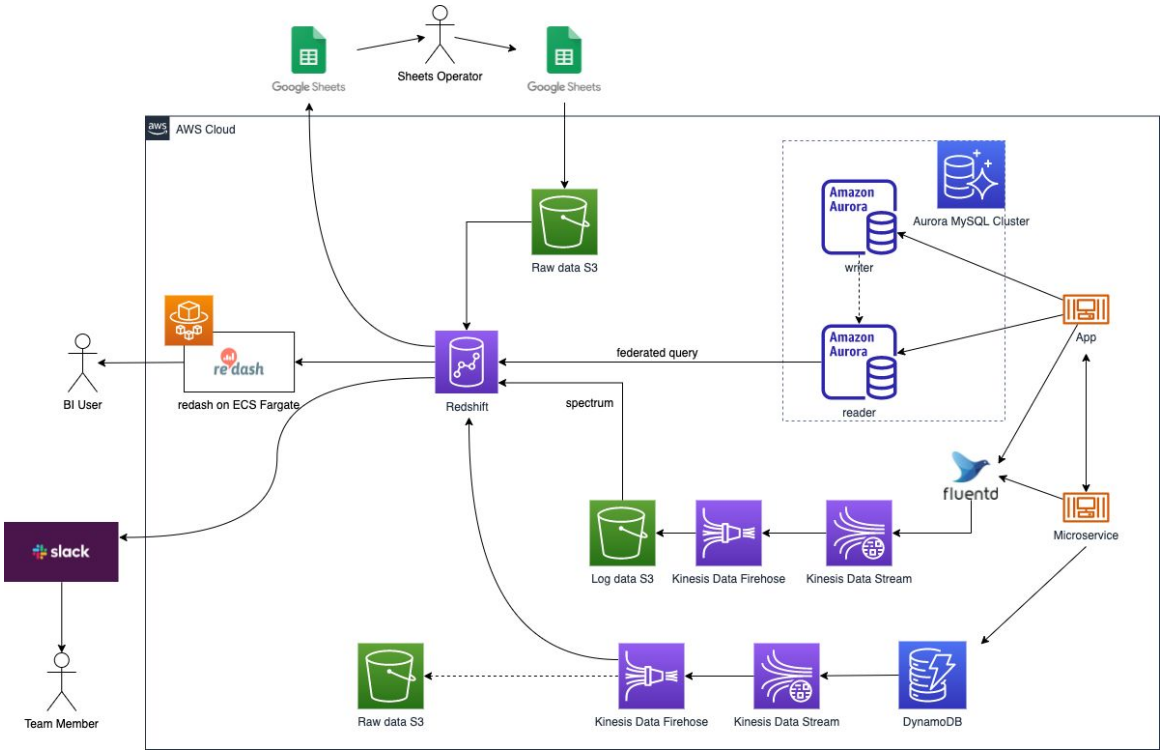
- 社内の他プロダクトで導入実績あり
- 社内知見
- アプリケーションがAWSにあるので他のAWSサービスとの親和性が高い

近代的なデータ基盤の必要性とその中核の選択肢

	半構造データ	ETLの簡略化 FederatedQuery	AWS親和性 と導入実績
 Amazon Athena ＋  Amazon S3	△ ※事前定義が必要	○	△
 Amazon Redshift	○	○	○
 Google BigQuery	○	△ ※レイテンシーの懸念	△

1年運用して感じた教訓とその対策方針

2021年2月に導入してから約1年、色々なことが発生
発生したことを踏まえた教訓と対策方針を共有

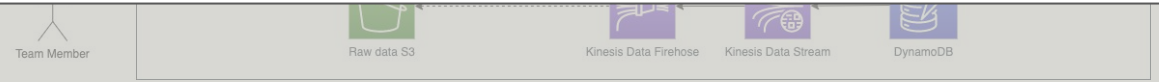


1年運用して感じた教訓とその対策方針

2021年2月に導入してから約1年、色々なことが発生
発生し

3つのエピソード と 5つの教訓

- Ep1: 開発環境に関して
  - 教訓1: ???
- Ep2: Redshiftの新機能を試したい
  - 教訓2: ???
- Ep3: Clusterのお引越し
  - 教訓3: ???
  - 教訓4: ???
  - 教訓5: ???



Ep1: 開発環境に関して

育ってくると、開発環境が欲しくなりますよね？

EP1:開発環境について

スモールスタート
dc2.Large を 2ノードで開始

開発環境用のClusterを
用意する余力はない

1つのClusterで
Databaseを分けて同居

現行世代

リージョン: アジアパシフィック (東京)

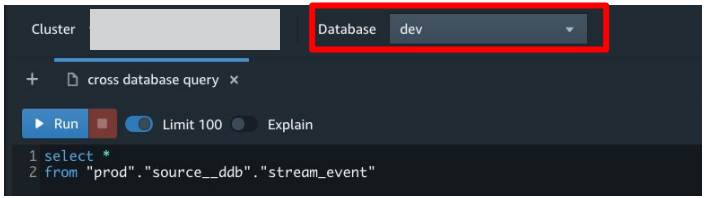
	vCPU	メモリ	対応ストレージ容量	I/O
高密度コンピューティング DC2				
dc2.large	2	15 GiB	0.16TB SSD	0.60 GB/s
dc2.8xlarge	32	244 GiB	2.56TB SSD	7.50 GB/s
高密度ストレージ DS2				
ds2.xlarge	4	31 GiB	2TB HDD	0.40 GB/s
ds2.8xlarge	36	244 GiB	16TB HDD	3.30 GB/s
Redshift マネージドストレージを備えた RA3*				
ra3.xlplus	4	32 GiB	32TB RMS	0.65 GB/s
ra3.4xlarge	12	96 GiB	128 TB RMS	2.00 GB/s
ra3.16xlarge	48	384 GiB	128 TB RMS	8.00 GB/s

*各 RA3 ノード付属マネージドストレージの対応ストレージ容量の合計

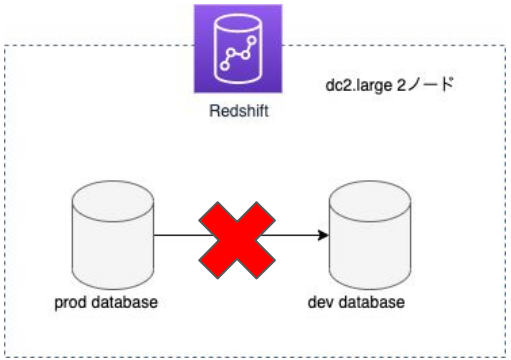
https://aws.amazon.com/jp/redshift/pricing/?nc=sn&loc=3

Tonamel

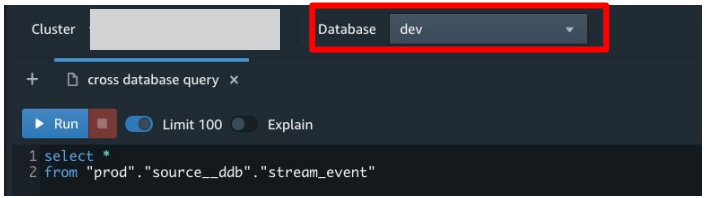
しかし、本番環境のデータで変換を試したい時が・・・



例えば、アプリケーションの開発環境のデータでは量が少ないので、本番環境のデータで新しいデータ変換を試したい時など



しかし、本番環境のデータで変換を試したい時が・・・



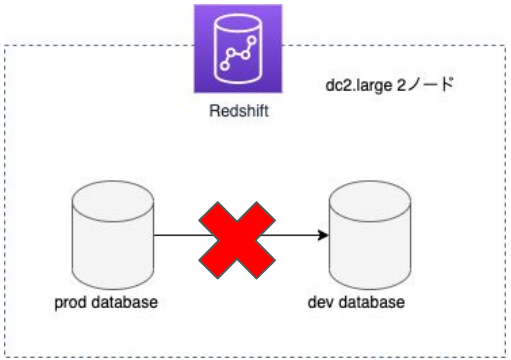
例えば、アプリケーションの開発環境のデータでは量が少ないので、本番環境のデータで新しいデータ変換を試したい時など

RA3なら出来るけどね！

Amazon Redshift クロスデータベースクエリの一般提供を発表

投稿日: Mar 10, 2021

Amazon Redshift クロスデータベースクエリでは、Redshift クラスター内のデータベース全体でクエリを実行できます。この機能は、Amazon Redshift RA3 ノードタイプが利用可能なすべてのリージョンで一般的にご利用いただけるようになりました。クロスデータベースクエリを使用することで、接続しているデータベースに関係なく、クラスター内の任意のデータベースに格納されているデータに、シ



https://aws.amazon.com/jp/about-aws/whats-new/2021/03/announcing-general-availability-of-amazon-redshift-cross-database-queries/

EP1:開発環境について

スモールスタート
dc2.Large を 2ノードで開始

開発環境用のClusterを
用意する余力はない

1つのClusterで
Databaseを分けて同居

現行世代

リージョン: アジアパシフィック (東京)

	vCPU	メモリ	対応ストレージ容量	I/O
高密度コンピューティング DC2				
dc2.large	2	15 GiB	0.16TB SSD	0.60 GB/s
dc2.8xlarge	32	244 GiB	2.56TB SSD	7.50 GB/s
高密度ストレージ DS2				
ds2.xlarge	4	31 GiB	2TB HDD	0.40 GB/s
ds2.8xlarge	36	244 GiB	16TB HDD	3.30 GB/s
Redshift マネージドストレージを備えた RA3*				
ra3.xlplus	4	32 GiB	32TB RMS	0.65 GB/s
ra3.4xlarge	12	96 GiB	128 TB RMS	2.00 GB/s
ra3.16xlarge	48	384 GiB	128 TB RMS	8.00 GB/s

*各 RA3 ノード付属マネージドストレージの対応ストレージ容量の合計

https://aws.amazon.com/jp/redshift/pricing/?nc=sn&loc=3

Tonamel

EP1:開発環境について

スモールスタート
dc2.Large を **2**ノードで開始

開発環境用のClusterを
用意する余力はない

1つのClusterで
Databaseを分けて同居

ra3.xlplusで開始してれば・・・

現行世代

リージョン: アジアパシフィック (東京)

	vCPU	メモリ	対応ストレージ容量	I/O
高密度コンピューティング DC2				
dc2.large	2	15 GiB	0.16TB SSD	0.60 GB/s
dc2.8xlarge	32	244 GiB	2.56TB SSD	7.50 GB/s
高密度ストレージ DS2				
ds2.xlarge	4	31 GiB	2TB HDD	0.40 GB/s
ds2.8xlarge	36	244 GiB	16TB HDD	3.30 GB/s
Redshift マネージドストレージを備えた RA3*				
ra3.xlplus	4	32 GiB	32TB RMS	0.65 GB/s
ra3.4xlarge	12	96 GiB	128 TB RMS	2.00 GB/s
ra3.16xlarge	48	384 GiB	128 TB RMS	8.00 GB/s

*各 RA3 ノード付属マネージドストレージの対応ストレージ容量の合計

https://aws.amazon.com/jp/redshift/pricing/?nc=sn&loc=3

Tonamel

EP1:開発環境について

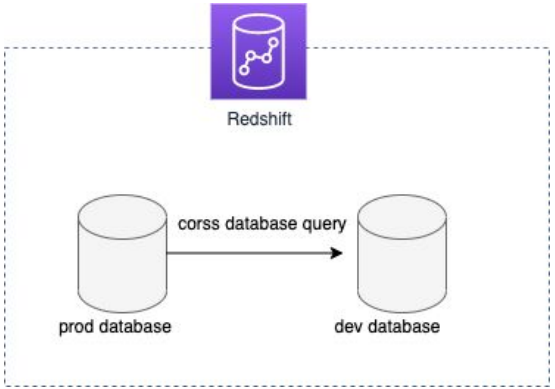
RA3のみ提供されている機能 がある

- Data Sharing
- クロスデータベースクエリ
- AQUA (Advanced Query Accelerator)
- etc…

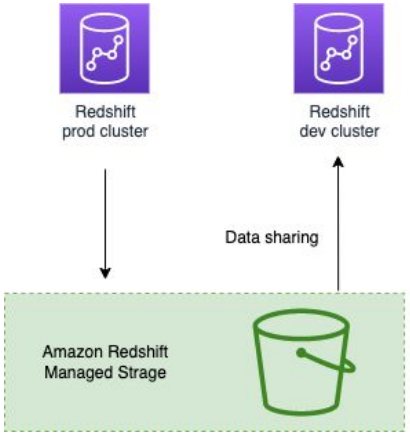
例えば、RA3なら開発環境について以下のようにできる
一部のETLのテストで本番データを用いたい時に役立つ

コスト要件が少し厳しくても、

教訓1:RA3で始めるべし



クラスタ同居・データベース分離型



クラスタ分離型

3つのエピソード と 5つの教訓

- Ep1: 開発環境に関して
  - **教訓1: ???**
- Ep2: Redshiftの新機能を試したい
  - 教訓2: ???
- Ep3: Clusterのお引越し
  - 教訓3: ???
  - 教訓4: ???
  - 教訓5: ???

3つのエピソード と 5つの教訓

- Ep1: 開発環境に関して
  - **教訓1: RA3で始めるべし**
- Ep2: Redshiftの新機能を試したい
  - 教訓2: ???
- Ep3: Clusterのお引越し
  - 教訓3: ???
  - 教訓4: ???
  - 教訓5: ???

Ep2: Redshiftの新機能を試したい

Redshiftは魅力的な機能のプレビューを多く提供しています

Amazon Redshift が、Kinesis データストリームのストリーミング取り込みのパブリックプレビューを発表

投稿日: Feb 10, 2022

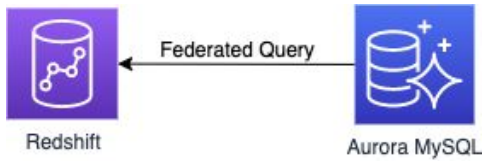
Amazon Redshift が、Kinesis Data Streams (KDS) のストリーミング取り込みのサポートを開始します。Amazon Redshift のストリーミング取り込みを使用すると、データを Amazon Redshift に取り込む前に Amazon S3 でステージングする必要がなくなり、1 秒あたり数百メガバイトのストリーミングデータをデータウェアハウスに取り込みながら、数秒で低レイテンシーを実現できます。

https://aws.amazon.com/jp/about-aws/whats-new/2022/02/amazon-redshift-public-preview-streaming-ingestion-kinesis-data-streams/

EP2: Redshiftの新機能を試したい

Federated Queryを活用すると、データ搬送が簡略化できる！
2021年2月時点ではMySQLへのFederatedQueryがプレビューでした。

Federated Queryなしの1例



Federated Queryを使った場合

Amazon Redshift now includes Amazon RDS for MySQL and Amazon Aurora MySQL databases as new data sources for federated querying (Preview)

Posted On: Dec 9, 2020

https://aws.amazon.com/jp/about-aws/whats-new/2020/12/amazon-redshift-now-includes-amazon-rds-for-mysql-and-amazon-aurora-mysql-databases-as-new-data-sources-for-federated-querying-preview/

EP2: Redshiftの新機能を試したい

メンテナンストラックを
Previewに・・・

EP2: Redshiftの新機能を試したい

メンテナンストラックを
Previewに・・・

しかし、**本番環境のClusterで**
Previewにすることはオススメされてません

EP2: Redshiftの新機能を試したい

なぜならば・・・

EP2: Redshiftの新機能を試したい

なぜならば・・・

1. メンテナンストラックは『最新』に**戻せない**

EP2: Redshiftの新機能を試したい

なぜならば・・・

1. メンテナンストラックは『最新』に**戻せない**

2. いつか**Previewは終了する**

Description

English follows Japanese | 英語のメッセージは日本語の後にございます

SQL Preview のプレビュープログラムにご参加いただき、ありがとうございます。 このプレビュープログラムは終了し、これらのプレビュートラックにおける Amazon Redshift クラスターのサポートは終了しました。 できるだけ早く、これらのクラスターの使用停止と削除を行ってください。 クラスターにデータを保持したい場合は、次のステップ [1] に従って Amazon S3 にデータをエクスポートできます。

2022 年 3 月 31 日までにご対応ください。

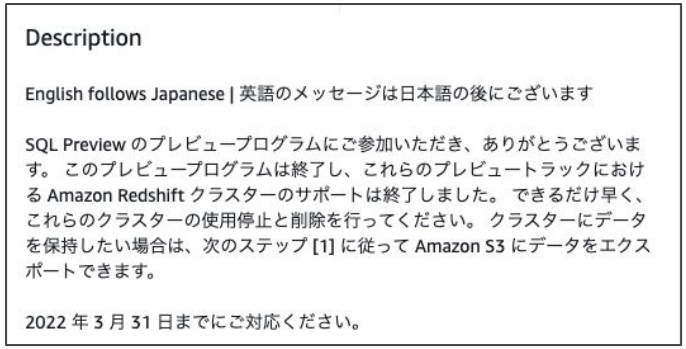
Personal Health Dashboard のRedshift operational notificationより抜粋

EP2: Redshiftの新機能を試したい

なぜならば・・・

1. メンテナンストラックは『最新』に**戻せない**

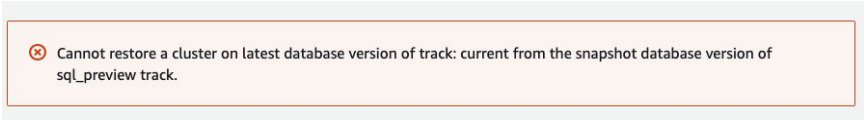
2. いつか**Previewは終了する**



Personal Health Dashboard のRedshift operational notificationより抜粋

3. 本番環境のCluster作り直しは大変

previewのメンテナンストラックにあるClusterのスナップショットは、復元でメンテナンストラック『最新』のCluserとして復元することはできない。



そのためUNLOADとCOPYを駆使して
S3を経由したデータのお引越しが必要

EP2: Redshiftの新機能を試したい

というわけで、

EP2: Redshiftの新機能を試したい

というわけで、

教訓2: 本番環境でPreviewにするべからず

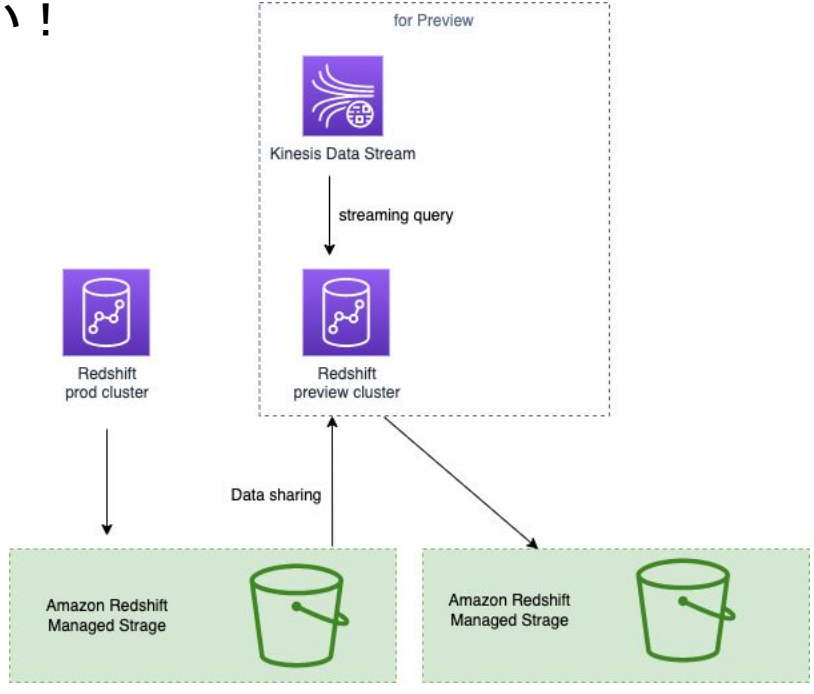
EP2: Redshiftの新機能を試したい

というわけで、

教訓2: 本番環境でPreviewにするべからず

しかし、本番環境中のデータで新機能を試したい！
そんなときは、**Data Sharingの出番**
(RA3のみで使える機能)

今Public Preview中の
ストリーミング取り込みなら



EP2: Redshiftの新機能を試したい

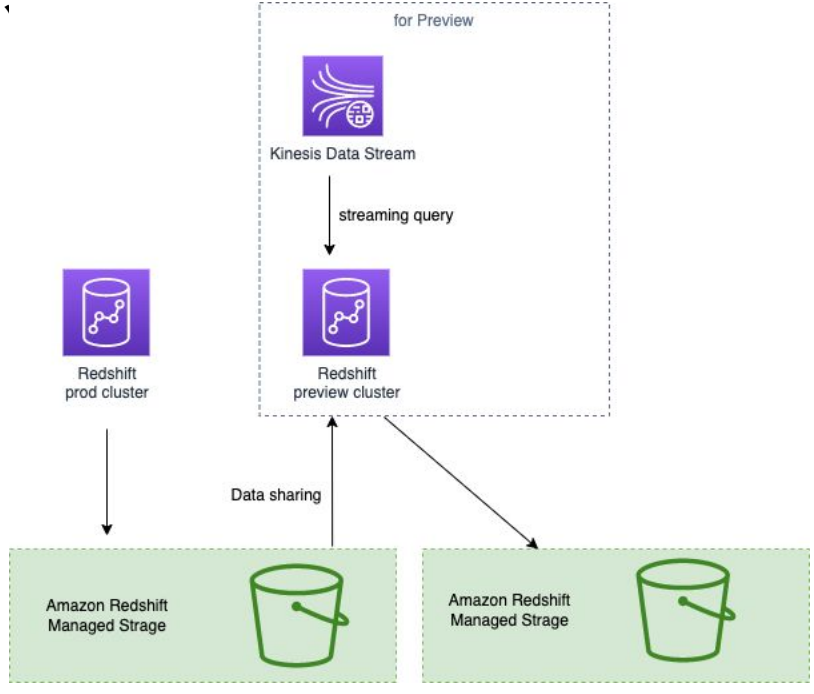
というわけで、

教訓1: RA3で始めるべし
教訓2: 本番環境でPreviewにするべからず

合わせて大事

しかし、本番環境中のデータで新機能を試したい
そんなときは、**Data Sharingの出番**
(RA3のみで使える機能)

今Public Preview中の
ストリーミング取り込みなら



3つのエピソード と 5つの教訓

- Ep1: 開発環境に関して
  - 教訓1: RA3で始めるべし
- Ep2: Redshiftの新機能を試したい
  - **教訓2: ???**
- Ep3: Clusterのお引越し
  - 教訓3: ???
  - 教訓4: ???
  - 教訓5: ???

3つのエピソード と 5つの教訓

- Ep1: 開発環境に関して
  - 教訓1: RA3で始めるべし
- Ep2: Redshiftの新機能を試したい
  - **教訓2: 本番環境でPreviewにするべからず**
- Ep3: Clusterのお引越し
  - 教訓3: ???
  - 教訓4: ???
  - 教訓5: ???

Ep3: Clusterのお引越し

みなさま、お気づきでしょうか？
運用1年目でClusterを新しく作成し、お引越し中です。

Description

English follows Japanese | 英語のメッセージは日本語の後にございます

SQL Preview のプレビュープログラムにご参加いただき、ありがとうございます。 このプレビュープログラムは終了し、これらのプレビュートラックにおける Amazon Redshift クラスターのサポートは終了しました。 できるだけ早く、これらのクラスターの使用停止と削除を行ってください。 クラスターにデータを保持したい場合は、次のステップ [1] に従って Amazon S3 にデータをエクスポートできます。

2022 年 3 月 31 日までにご対応ください。

再掲：Personal Health Dashboard のRedshift operational notificationより抜粋

2022年3月31日までに
できるだけ早く引っ越し！

EP3: Clusterのお引越し

Clusterのお引越しをする必要が出てきたのですが、
これを気に色々と見直すべきところはあることがわかりました。

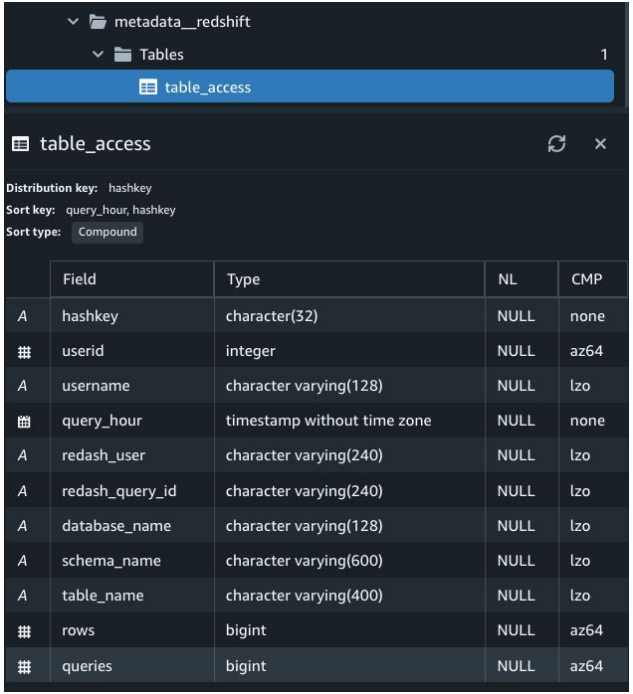
- 使用頻度の低い古いテーブルが有る
  (実は作り直して、別の新しいものがある)
- ワークフロー (airflow) が頑張りすぎてる
- 無加工データがないものがある

EP3: Clusterのお引越し … 利用状況調査

『テーブルにどれだけアクセスしているのか？』

算出方法の例として
- BIのクエリログを集計
- stl_explainを集計
- stl_queryを集計

誰が？いつ？どのテーブルを
利用していたのかを記録していた



	Field	Type	NL	CMP
A	hashkey	character(32)	NULL	none
#	userid	integer	NULL	az64
A	username	character varying(128)	NULL	lzo
🕒	query_hour	timestamp without time zone	NULL	none
A	redash_user	character varying(240)	NULL	lzo
A	redash_query_id	character varying(240)	NULL	lzo
A	database_name	character varying(128)	NULL	lzo
A	schema_name	character varying(600)	NULL	lzo
A	table_name	character varying(400)	NULL	lzo
#	rows	bigint	NULL	az64
#	queries	bigint	NULL	az64

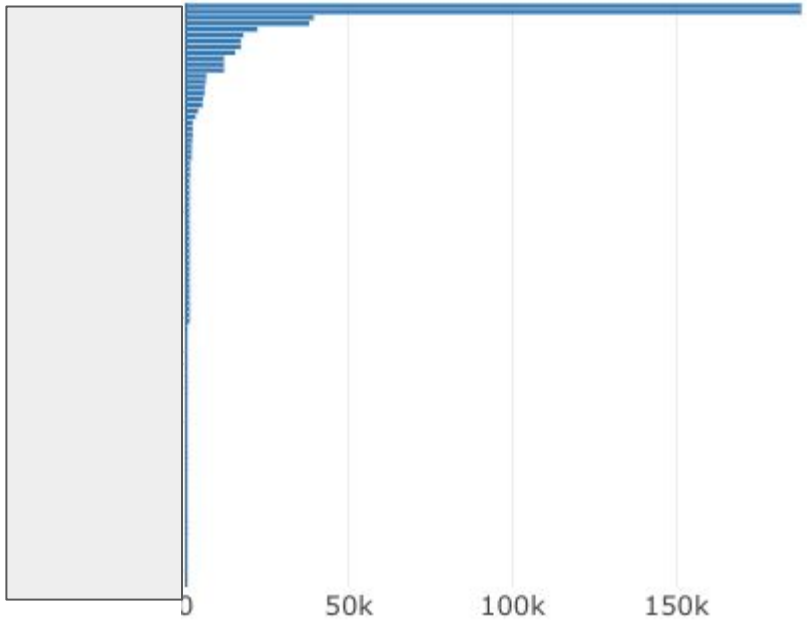
Tonamel

EP3: Clusterのお引越し … 利用状況調査

テーブルの利用状況を集計すると
よく参照される
価値のあるテーブルが一目瞭然

使用されてないテーブルはDeprecated
引っ越し対象から除外

教訓3: テーブルの利用状況把握は大事です



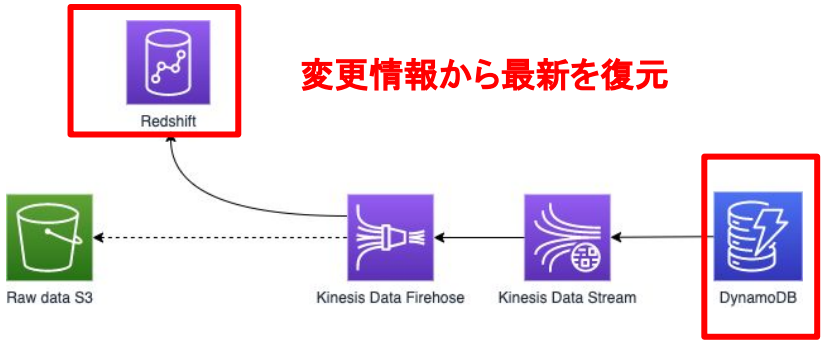
直近1ヶ月のテーブルごとのクエリ発行数ランキング

Tonamel

EP3: Clusterのお引越し … ワークフローで頑張りすぎ

DynamoDB の最新状態を復元するために**5分に1回起動する**ワークフローがある

```
CREATE TABLE "source__ddb"."stream_event" (
  "awsRegion"     varchar ENCODE ZSTD,
  "dynamodb"      super   ENCODE ZSTD,
  "eventID"       varchar ENCODE ZSTD,
  "eventName"     varchar ENCODE ZSTD,
  "userIdentity"  varchar ENCODE ZSTD,
  "recordFormat"  varchar ENCODE ZSTD,
  "tableName"     varchar ENCODE ZSTD,
  "eventSource"   varchar ENCODE ZSTD
)
DISTSTYLE AUTO SORTKEY AUTO ENCODE AUTO;
```



EP3: Clusterのお引越し … ワークフローで頑張りすぎ

Materialized Viewで
同じことが実現できる

```
CREATE TABLE "source__ddb"."stream_event" (
  "awsRegion"      varchar ENCODE ZSTD,
  "dynamodb"       super   ENCODE ZSTD,
  "eventID"        varchar ENCODE ZSTD,
  "eventName"      varchar ENCODE ZSTD,
  "userIdentity"   varchar ENCODE ZSTD,
  "recordFormat"   varchar ENCODE ZSTD,
  "tableName"      varchar ENCODE ZSTD,
  "eventSource"    varchar ENCODE ZSTD
)
DISTSTYLE AUTO SORTKEY AUTO ENCODE AUTO;
```

```
create materialized view source__ddb.stream_event_latest_mv
auto refresh yes
as
select
    "awsRegion"
    ,"tableName"
    ,"dynamodb"."Keys"
    ,max("dynamodb"."ApproximateCreationDateTime") as "latestApproximateCreationDateTime"
from source__ddb.stream_event
group by 1,2,3
```

最新行を特定するためのMV

```
create view source__ddb.snapshot_latest_v as
select e.*
from source__ddb.stream_event_latest_mv l
inner join source__ddb.stream_event e
 on l."awsRegion" = e."awsRegion"
 and l."tableName" = e."tableName"
 and l."Keys" = e."dynamodb"."Keys"
 and l."latestApproximateCreationDateTime" = e."dynamodb"."ApproximateCreationDateTime"
```

参照用のDB View

EP3: Clusterのお引越し … ワークフローで頑張りすぎ

Materialized Viewで
同じことが実現できる

Auto Refreshを有効にすると
ワークフローなしでOK

```
CREATE TABLE "source__ddb"."stream_event" (
  "awsRegion"      varchar ENCODE ZSTD,
  "dynamodb"       super   ENCODE ZSTD,
  "eventID"        varchar ENCODE ZSTD,
  "eventName"      varchar ENCODE ZSTD,
  "userIdentity"   varchar ENCODE ZSTD,
  "recordFormat"   varchar ENCODE ZSTD,
  "tableName"      varchar ENCODE ZSTD,
  "eventSource"    varchar ENCODE ZSTD
)
DISTSTYLE AUTO SORTKEY AUTO ENCODE AUTO;
```

```
create materialized view source__ddb.stream_event_latest_mv
auto refresh yes
as
select
    "awsRegion"
    ,"tableName"
    ,"dynamodb"."Keys"
    ,max("dynamodb"."ApproximateCreationDateTime") as "latestApproximateCreationDateTime"
from source__ddb.stream_event
group by 1,2,3
```

最新行を特定するためのMV

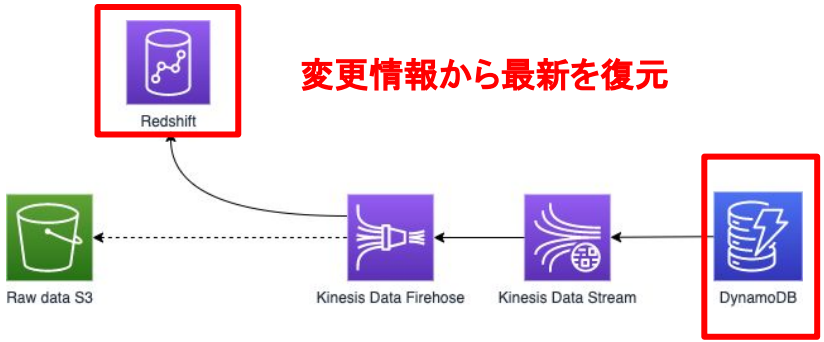
```
create view source__ddb.snapshot_latest_v as
select e.*
from source__ddb.stream_event_latest_mv l
inner join source__ddb.stream_event e
 on l."awsRegion" = e."awsRegion"
 and l."tableName" = e."tableName"
 and l."Keys" = e."dynamodb"."Keys"
 and l."latestApproximateCreationDateTime" = e."dynamodb"."ApproximateCreationDateTime"
```

参照用のDB View

EP3: Clusterのお引越し … ワークフローで頑張りすぎ

DynamoDB の最新状態を復元するために**5分に1回起動する**ワークフローがある

```
CREATE TABLE "source__ddb"."stream_event" (
  "awsRegion"     varchar ENCODE ZSTD,
  "dynamodb"      super   ENCODE ZSTD,
  "eventID"       varchar ENCODE ZSTD,
  "eventName"     varchar ENCODE ZSTD,
  "userIdentity"  varchar ENCODE ZSTD,
  "recordFormat"  varchar ENCODE ZSTD,
  "tableName"     varchar ENCODE ZSTD,
  "eventSource"   varchar ENCODE ZSTD
)
DISTSTYLE AUTO SORTKEY AUTO ENCODE AUTO;
```

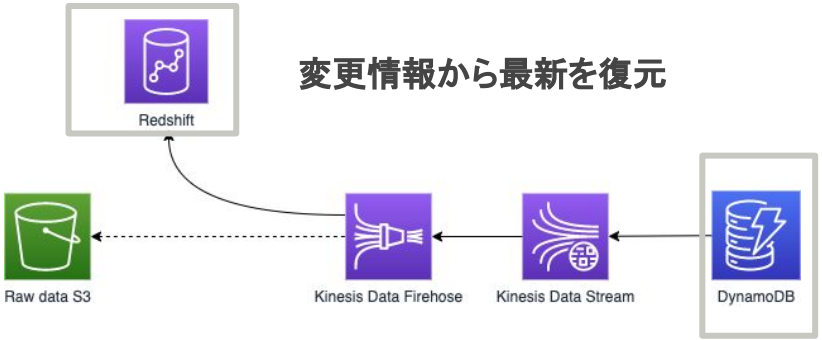


EP3: Clusterのお引越し … ワークフローで頑張りすぎ

DynamoDB の最新状態を復元するために**5分に1回起動するワークフローがある**

```
CREATE TABLE "source__ddb"."stream_event" (
  "awsRegion"     varchar ENCODE ZSTD,
  "dynamodb"      super   ENCODE ZSTD,
  "eventID"       varchar ENCODE ZSTD,
  "eventName"     varchar ENCODE ZSTD,
  "userIdentity"  varchar ENCODE ZSTD,
  "recordFormat"  varchar ENCODE ZSTD,
  "tableName"     varchar ENCODE ZSTD,
  "eventSource"   varchar ENCODE ZSTD
)
DISTSTYLE AUTO SORTKEY AUTO ENCODE AUTO;
```

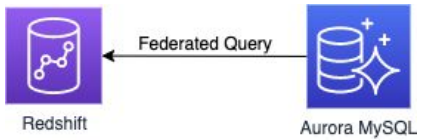
教訓4: Materialized Viewを積極的に使おう



EP3: Clusterのお引越し … 無加工データがないものがある

Federated Queryを使わずにAurora MySQLのデータを取得する場合

単純にFederated Queryを使った場合

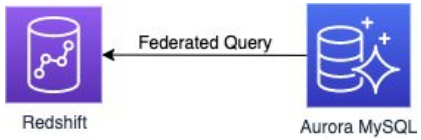


EP3: Clusterのお引越し … 無加工データがないものがある

Federated Queryを使わずにAurora MySQLのデータを取得する場合

S3に無加工データが自然に残る

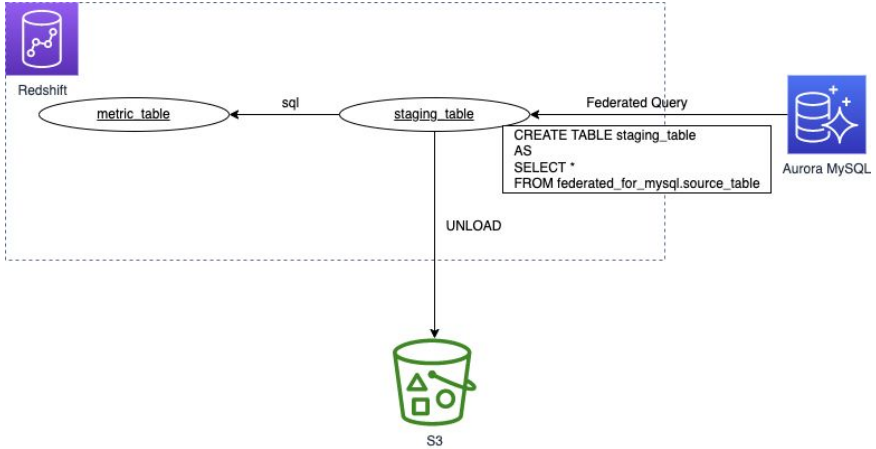
単純にFederated Queryを使った場合



S3に無加工データは残らない

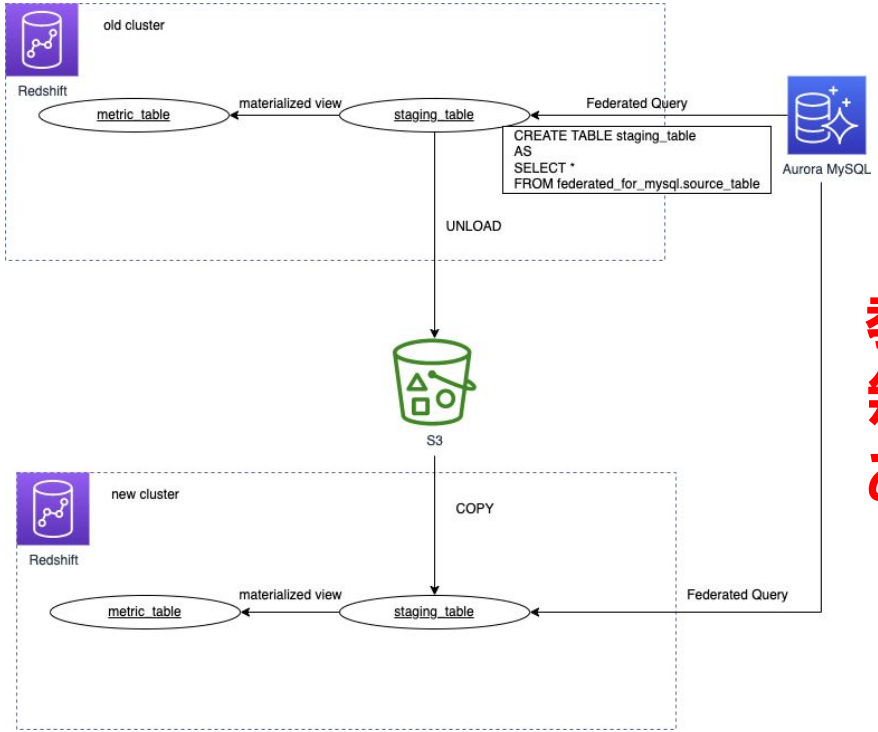
EP3: Clusterのお引越し … 無加工データがないものがある

Federated Queryを使う場合、CTASで無加工データをRedshiftに取り込み
UNLOADを用いてS3へ出力したほうが良い



EP3: Clusterのお引越し … 無加工データがないものがある

その上で、Materialized Viewと組み合わせておけば、
引っ越しも簡単だった・・・かもしれない



教訓5:
無加工データがS3にあるのは大事です

3つのエピソード と 5つの教訓

- Ep1: 開発環境に関して
  - 教訓1: RA3で始めるべし
- Ep2: Redshiftの新機能を試したい
  - 教訓2: 本番環境でPreviewにするべからず
- Ep3: Clusterのお引越し
  - 教訓3: ???
  - 教訓4: ???
  - 教訓5: ???

3つのエピソード と 5つの教訓

- Ep1: 開発環境に関して
  - 教訓1: RA3で始めるべし
- Ep2: Redshiftの新機能を試したい
  - 教訓2: 本番環境でPreviewにするべからず
- Ep3: Clusterのお引越し
  - **教訓3: テーブルの利用状況把握は大事です**
  - **教訓4: Materialized Viewを積極的に使おう**
  - **教訓5: 無加工データがS3にあるのは大事です**

3つのエピソード と 5つの教訓

- Ep1: 開発環境に関して
  - 教訓1: RA3で始めるべし
- Ep2: Redshiftの新機能を試したい
  - 教訓2: 本番環境でPreviewにするべからず
- Ep3: Clusterのお引越し
  - 教訓3: テーブルの利用状況把握は大事です
  - 教訓4: Materialized Viewを積極的に使おう
  - 教訓5: 無加工データがS3にあるのは大事です

- 教訓1: RA3で始めるべし

- 教訓2: 本番環境でPreviewにするべからず

- 教訓3: テーブルの利用状況把握は大事です
- 教訓4: Materialized Viewを積極的に使おう
- 教訓5: 無加工データがS3にあるのは大事です

- 教訓1: RA3で始めるべし
  新しいアーキテクチャーが出たら、再構築の可能性
- 教訓2: 本番環境でPreviewにするべからず
  本番環境のコピー環境を作ってPreviewしたい可能性
- 教訓3: テーブルの利用状況把握は大事です
- 教訓4: Materialized Viewを積極的に使おう
- 教訓5: 無加工データがS3にあるのは大事です

本番環境を再構築する可能性はまだある

⇒ 簡単に本番環境と同じものを作れることは大事

- 教訓1: RA3で始めるべし
  新しいアーキテクチャーが出たら、再構築の可能性
- 教訓2: 本番環境でPreviewにするべからず
  本番環境のコピー環境を作ってPreviewしたい可能性
- 教訓3: テーブルの利用状況把握は大事です
- 教訓4: Materialized Viewを積極的に使おう
- 教訓5: 無加工データがS3にあるのは大事です

Infrastructure as Code

HashiCorp Terraform

AWS CloudFormation

構成・IAM権限管理をコード管理して複製容易に

Analytics Engineering

https://www.getdbt.com/analytics-engineering/

dbt_

テスト、文書化、透明性 etc…

ソフトウェア・エンジニアリングの
ベストプラクティスを分析に

良いデータウェアハウスの運用とは、
再構築が容易であることだ！

まとめ

TonamelではRedshiftを1年運用した結果、5つの教訓が得られました。

- 教訓1: RA3で始めるべし
- 教訓2: 本番環境でPreviewにするべからず
- 教訓3: テーブルの利用状況把握は大事です
- 教訓4: Materialized Viewを積極的に使おう
- 教訓5: 無加工データがS3にあるのは大事です

この教訓から得た言葉として、

『良いデータウェアハウスの運用とは、
再構築が容易であることだ！』

この言葉を今後の対策方針として、締めくくりたいと思います

Tonamel