

AWS

S U M M I T

# AWS の NoSQL 入門

～Amazon ElastiCache, Amazon DynamoDB～

ソリューションアーキテクト 桑野章弘

2017年6月1日



# 自己紹介

- 桑野 章弘(くわの あきひろ)
- ソリューションアーキテクト
- メディア系のお客様を担当しております。
- 前職は渋谷のインフラエンジニア
- 趣味嗜好
  - 好きなAWSのサービス : ElastiCache, Route53
  - 好きなデータストア : **MongoDB**
  - インフラ全般 (昔は自作サーバ等やってました)



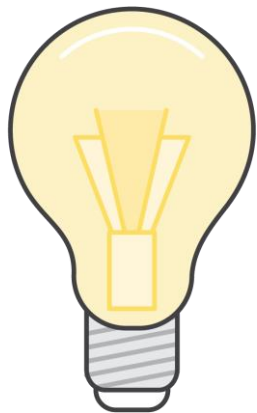
# アジェンダ

- 本セッションの目的
- NoSQLとは
- AWSのNoSQLサービス
  - Amazon ElastiCache
  - Amazon DynamoDB
- NoSQLの使い方

# 本セッションの目的

本日持って帰って頂くこと

**NoSQLを最大限に活用し  
可用性の高いシステムを作る！**



# 本セッションの目的

- NoSQLとRDBMSの得意／不得意な分野について説明し、NoSQLサービスの主要なユースケースについて理解する
- Amazon Web Service (AWS) の様々なデータベースサービスの中からNoSQLサービスである、Amazon ElastiCacheとAmazon DynamoDBについて理解する

# NoSQLとは

# NoSQLとは

- 従来のSQLで表されるようなリレーショナルデータベース（以下RDBMS）では解決できない課題を解決するために生まれたRDBMSではないデータベースの総称
  - NoSQLは非常に多くの種類がある
- 一意に決められるものではないが主な特徴として次のように分類することが出来る





# RDBMSとNoSQLの特徴

## RDBMS

- 正規化／リレーショナル
- SQLを使用可能
- 柔軟なクエリ
- トランザクション処理
- データの堅牢性／一貫性
- スケールアウトの煩雑さ
- テーブル管理の煩雑さ

## NoSQL

- 非正規化／階層化
- 高速なパフォーマンス
- 高いスケーラビリティ
- ユースケースに応じた様々なデータベース形式
- トランザクション処理は苦手
- クラスタの運用負荷



# NoSQLの種類

NoSQLは非常に多くの種類があるが、代表的な物にこの4種類がある

- キーバリューストア
- カラム指向データベース
- ドキュメント指向データベース
- グラフ指向データベース

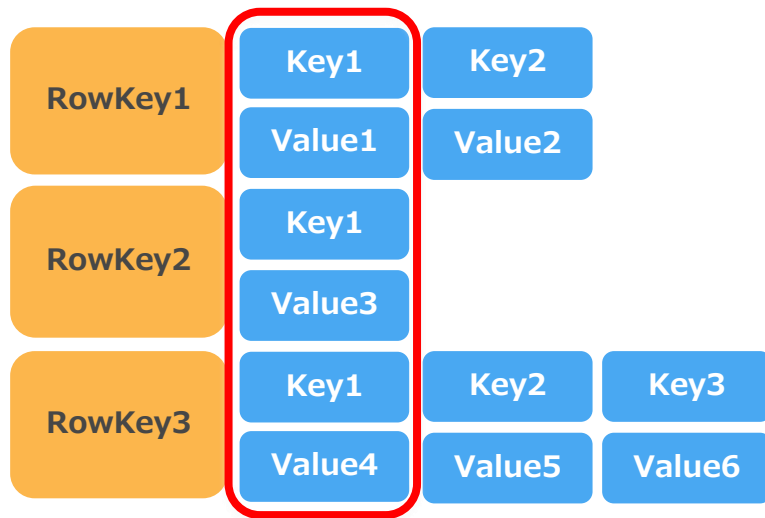
# キーバリューストア(KVS)

- キーとバリュー（値）という単純な構造
- 超高速なパフォーマンス
- RDBMSに比べ読み書きが高速

Key1	Value1
Key2	Value2
Key3	Value3

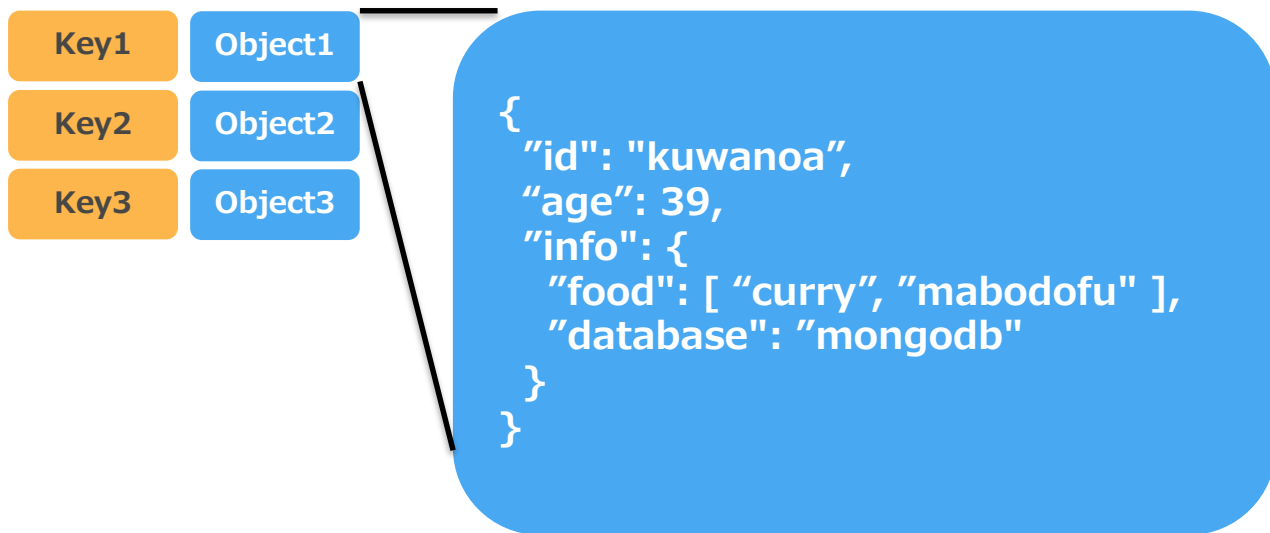
# カラム指向データベース

- カラム指向のデータ構造
- ログなど大量のデータ解析向き
- RDBMSに比べ書き込みが高速



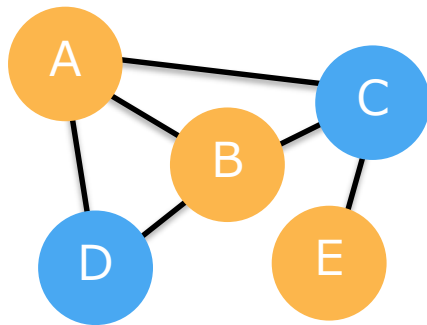
# ドキュメント指向データベース

- JSONやXML等の不定形なデータ構造に対応
- 複雑なデータモデリングを容易に実現可能



# グラフ指向データベース

- データ間を相互に結びつけてデータ同士の関係をグラフという形で表す
- 複雑な関係性を表すのが得意とする
  - SNSのフレンドの関連性等



# NoSQLの種類：主要なOSS

## キーバリューストア



## カラム指向データベース



## ドキュメント指向データベース



## グラフ指向データベース



# NoSQLのメリット

NoSQLを使うことによって何を実現するか

- 低レイテンシ・高スループット、シンプルなAPI
- Webセッション管理
- Publish・Subscribeモデル、イベント処理
- JSON形式データの格納
- ソーシャル・グラフ、BOM（Bill of material）ツリーの検索

これらの特徴が必要とされるところがNoSQLの使い所と言える  
そしてもう一つの一般的な特徴として、自前でクラスタ等の運用をするのにかなりの労力が必要とされる。



# オンプレミスのサーバの場合

App optimization

Scaling

High availability

Database backups

DB s/w patches

DB s/w installs

OS patches

OS installation

Server maintenance

Rack & stack

Power, HVAC, net



# Amazon EC2上に構築した場合

App optimization

Scaling

High availability

Database backups

DB s/w patches

DB s/w installs

OS patches



*you*

OS installation

Server maintenance

Rack & stack

Power, HVAC, net



# マネージドサービスを使った場合

App optimization

you

Scaling

High availability

Database backups

DB s/w patches

DB s/w installs

OS patches

OS installation

Server maintenance

Rack & stack

Power, HVAC, net



# AWSのNoSQLマネージドサービス

# AWS が提供するデータベースサービス



## Amazon DynamoDB

高速でシームレスな拡張性と信頼性を持つ  
NoSQL データベース



## Amazon ElastiCache

インメモリキャッシュ  
Memcached, Redis



## Amazon RDS

リレーショナルデータベース  
Aurora, PostgreSQL, MySQL,  
MariaDB, Oracle, SQL Server



## Amazon Redshift

ペタバイト規模のデータウェアハウス

# AWS が提供するデータベースサービス



Amazon DynamoDB

高速でシームレスな拡張性と信頼性を持つ  
NoSQL データベース



Amazon ElastiCache

インメモリキャッシュ  
Memcached, Redis



Amazon RDS

リレーショナルデータベース  
Aurora, PostgreSQL, MySQL,  
Amazon Redshift

**NoSQLサービス**

Amazon Redshift  
ペタバイト規模のデータウェアハウス

# NoSQLの種類：主要なOSS（再掲）

## キーバリューストア



## カラム指向データベース



## ドキュメント指向データベース



## グラフ指向データベース



# NoSQLの種類：AWSのサービスとのマッピング

キーバリューストア



Amazon ElastiCache

カラム指向データベース



Amazon DynamoDB

ドキュメント指向データベース

グラフ指向データベース



# NoSQLの種類：AWSのサービスとのマッピング

## キーバリューストア



Amazon ElastiCache

## カラム指向データベース

DynamoDBはキーバ  
リユー、ドキュメント  
指向、グラフ指向に対  
応可能



Amazon DynamoDB

## ドキュメント指向データ ベース

## グラフ指向データベース

# Amazon ElastiCache



# Amazon ElastiCache の特徴



インメモリキーバリューストア



フルマネージド



ハイパフォーマンス



高いスケーラビリティ



Redis と memcached が利用可能



Amazonによる拡張(Amazon Redis)

# Amazon ElastiCacheとは



キャッシュをマネージドで提供するNoSQLサービス  
代表的な2つのKVSエンジンに対応

構築／運用の自動化

- アプリケーションキャッシュサーバの構築／運用
- Redisの機能をマネージドに使う（ランキング、Pub/Sub等）
- バックアップ
- 監視
- 障害検知／復旧



# KVSエンジンの選択



- ElastiCacheの2つのエンジンはどちらもKVSという特徴を持っており、キャッシュに使われることが多い
- どちらを使えばいいのか？
- 例えば、「メモリに一時的なセッションのキャッシュを保持する」「ランキングシステム用の一時領域」など



VS



# KVSエンジンの選択 : Memcached or Redis



- マルチスレッド
- 非永続化
- 単純なデータタイプ  
(String、Objects)
- メンテナンスが楽
- 垂直分散が楽

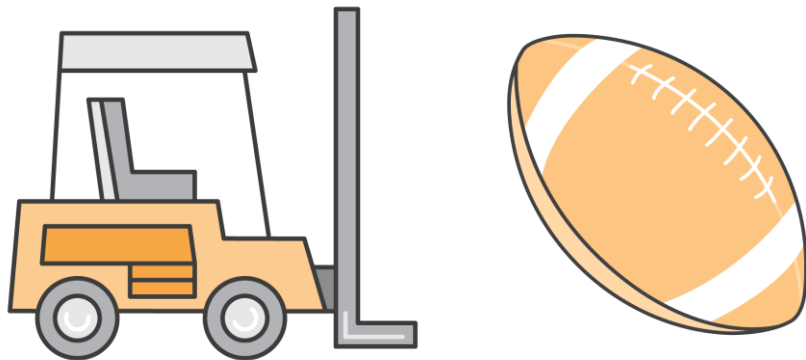
- シングルスレッド
- 永続化
- 多数のデータタイプ
- Atomic処理
- Pub/sub メッセージング
- リードレプリカ / フェイルオーバー

strings, lists,  
hashes, sets,  
sorted sets,  
bitmaps,  
HyperLogLogs



# 運用の改善

自分たちで運用すると大変なフェイルオーバーや、バックアップ、スケールアップ等をフルマネージドで行うことで運用負荷を最小限にすることが出来る

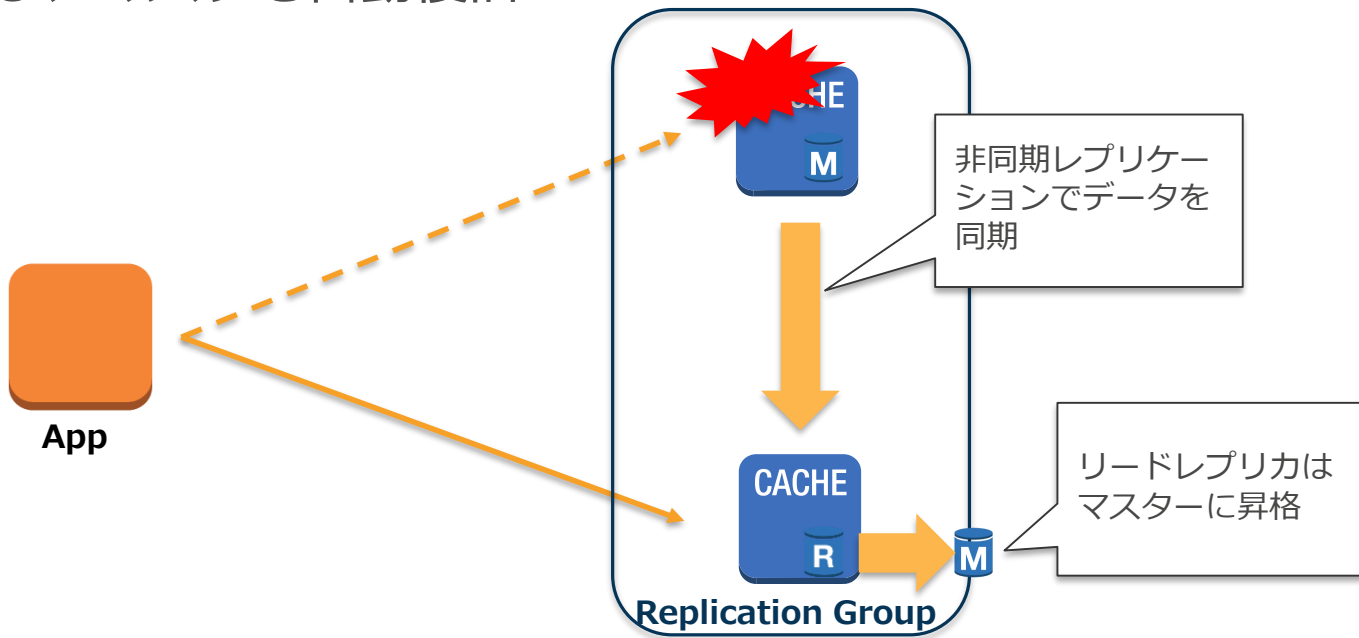


# フェイルオーバー



## レプリケーション機能を使ったフェイルオーバー

- 自動的に障害を検知し、同期先のリードレプリカへフェイルオーバーし、マスタも自動復旧

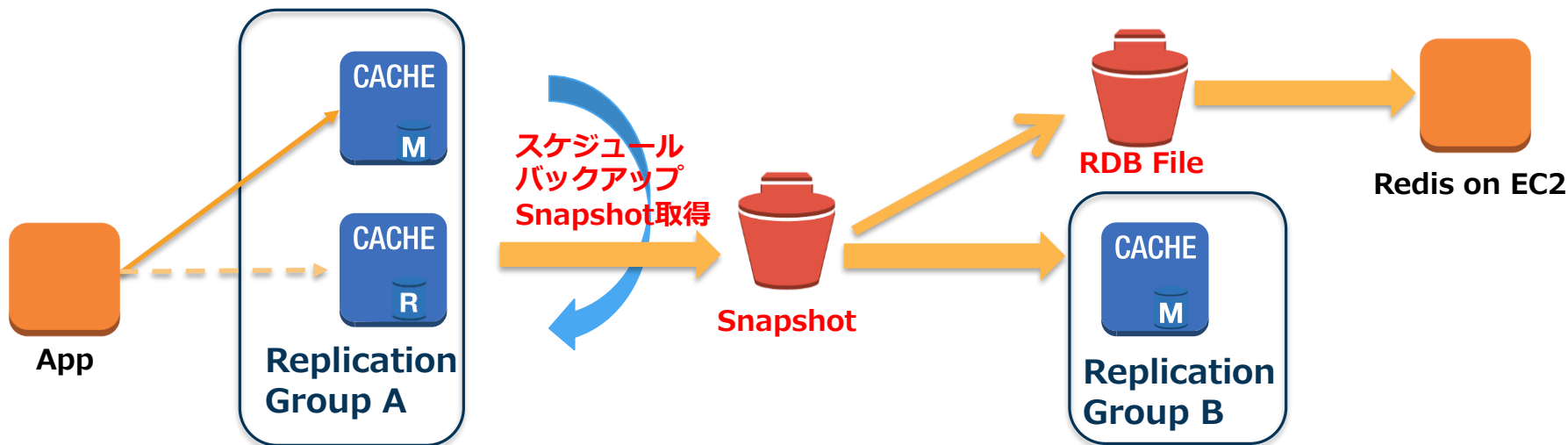




# バックアップ



- スナップショットバックアップが可能
  - S3へのスナップショットバックアップ取得
  - スケジュール、世代数など決めることが可能

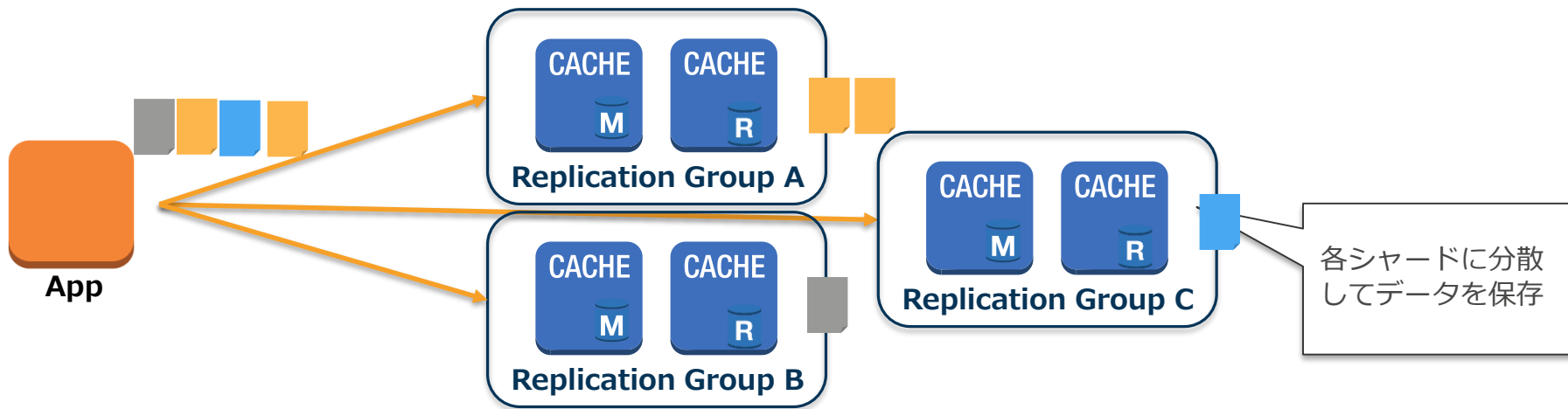


# Redis Cluster

New



- Redis 3.2 で Redis Cluster のサポート
- データを分散保存することで最大15シャード、3.5TiBのデータが保存可能
- 最大2000万/秒の読み込み、450万/秒の書き込み性能



# 価格



## オンデマンド キャッシュノード

- 初期費用無し、時間単位の従量課金モデル

## リザーブド キャッシュノード

- 予約金を支払うことで時間当たり価格を割引(最大70%節減)

## バックアップストレージ (Redis)

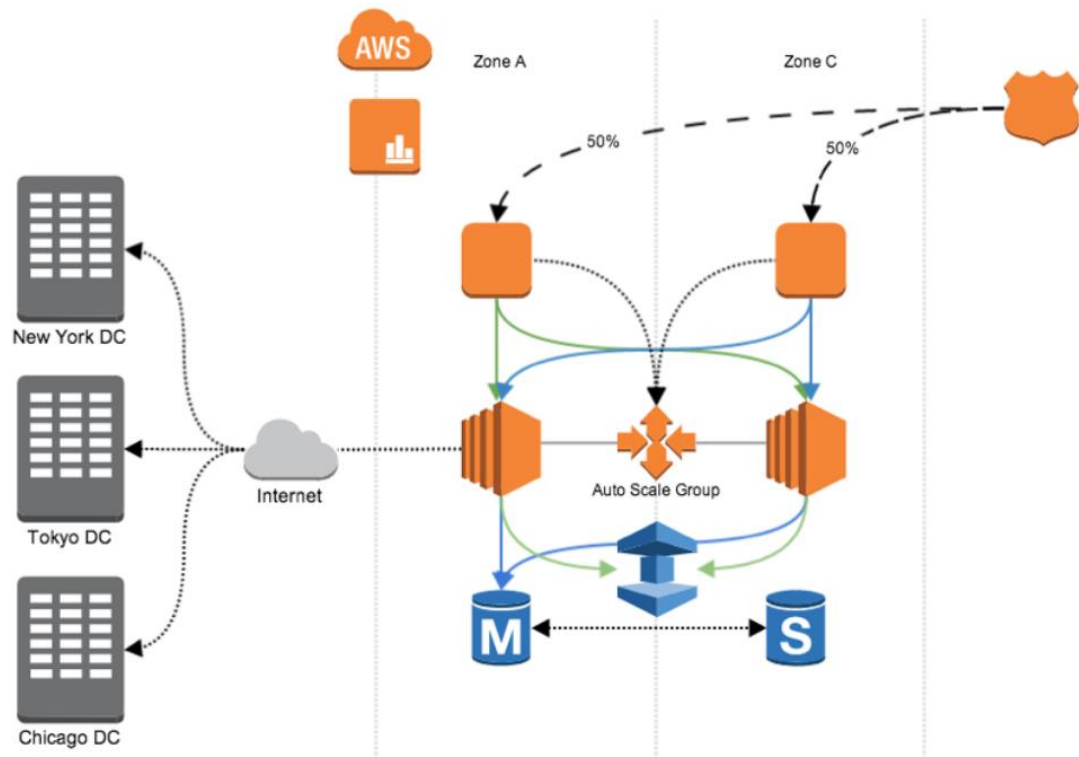
- 各クラスタに対して**1つのSnapshotは無料**
- 2つ以上のSnapshotから毎月 0.085 USD/GBが課金

## AZ間データ転送量

- ElastiCache間の通信は**課金対象外**
- EC2とElastiCache間でAZを超える場合 0.01 USD/GB が課金

# マネックス証券様 Monex Insight

Webベースリアルタイム  
市場データ配信システム  
株式市場の取引時間に合  
わせて、リソースの増減  
を制御し数分で自由にス  
ケール可能



\* AWS Summit Tokyo マネックス証券様のセッションより引用

# ElastiCacheの使いどころ

- 複雑な計算が必要となるデータをキャッシュ
- Redisの拡張された機能をマネージドサービスとして利用するための基盤
  - Pub/Sub
  - ランキングボード
  - 位置情報

詳しくはAWS Black Belt ElastiCache 資料を参照

<http://www.slideshare.net/AmazonWebServicesJapan/aws-black-belt-tech-amazon-elasticache>

# Amazon DynamoDB



# DynamoDB の特徴



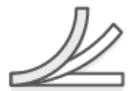
フルマネージド



高速、かつ一貫したパフォーマンス



高いスケーラビリティ



柔軟性



イベント指向のプログラミングも可能

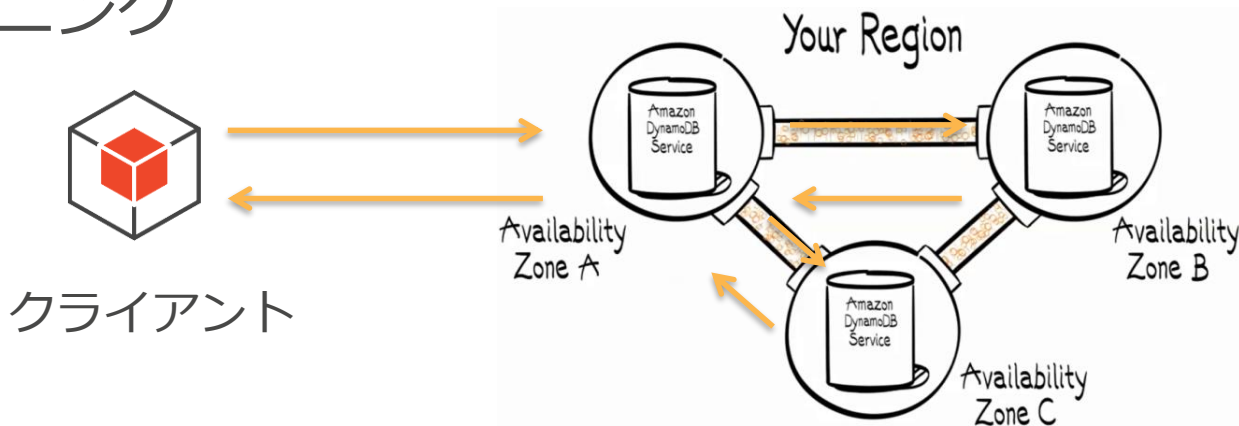


セキュリティ

# フルマネージドで高信頼性



- 単一障害点（SPOF）が存在しない
- データは3AZ（データセンタ）に保存される
- ストレージはデータ量、アクセス量に応じ自動でパーティショニング



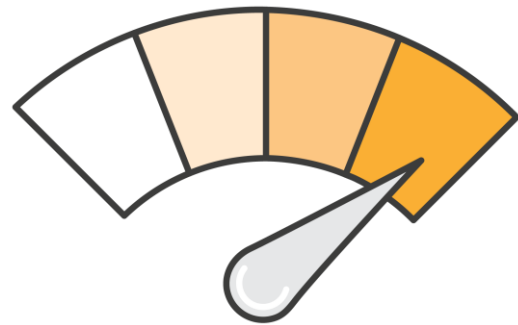


# プロビジョンドスループット



テーブルごとにReadとWriteそれぞれに対し、必要なスループットキャパシティを割り当てる（＝プロビジョンする）事が可能

さらにDB運用中にオンラインで変更可能

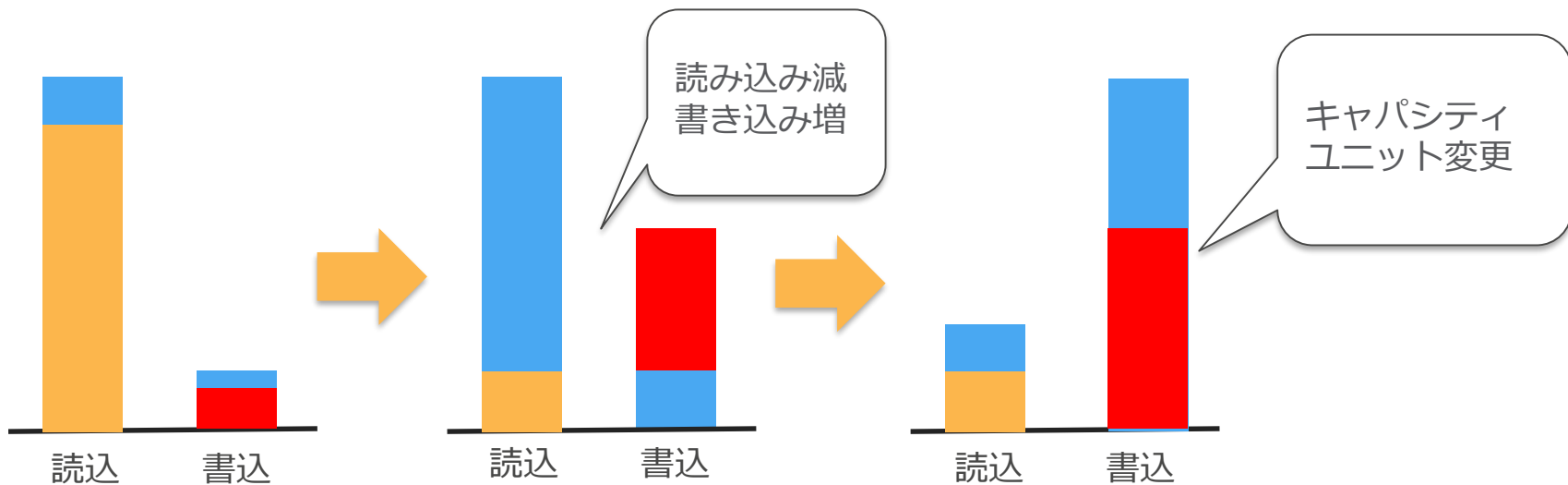


# プロビジョンドスループットの例

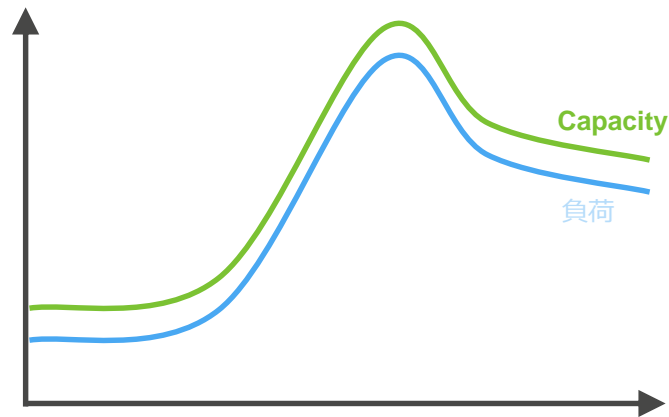
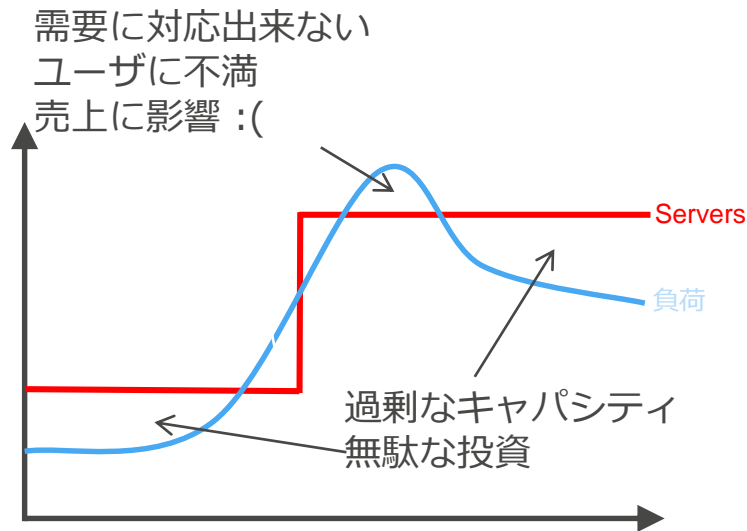


現在のアクセス量

プロビジョンした設定



# ユーザの需要に合わせてキャパシティをスケール



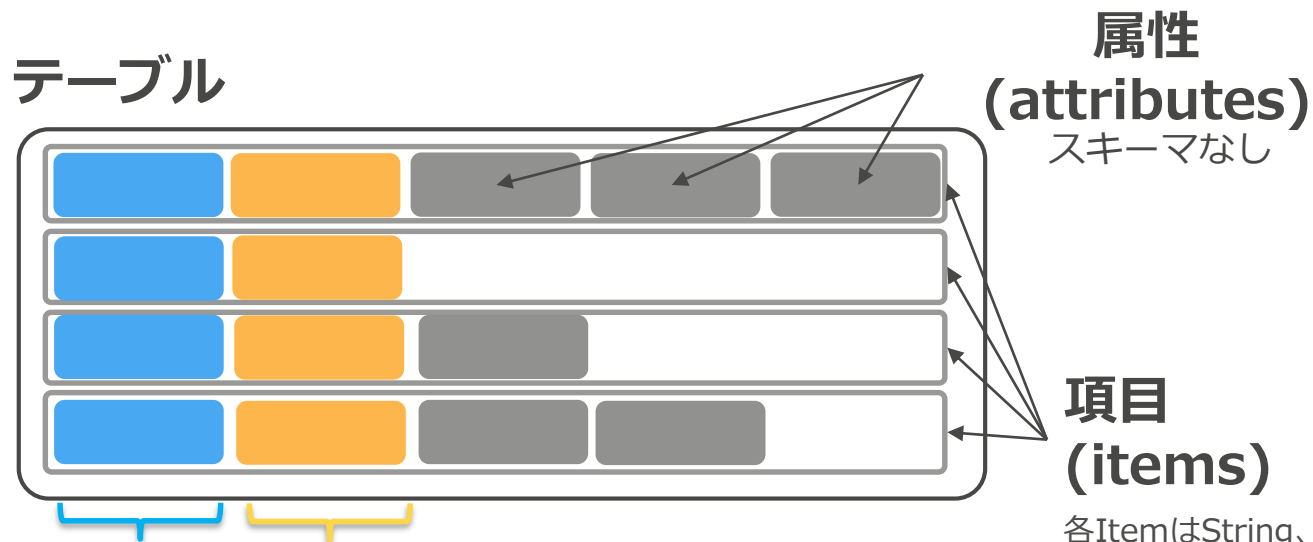
# ストレージの容量制限がない



使った分だけの従量課金制のストレージ  
データ容量の増加に応じたディスクやノードの増設作業は  
一切不要



# データの柔軟性：テーブル構造



パーティションキー  
(必須)

ソートキー  
(オプション)

パーティションキー + ソートキー(オプション) = プライマリキー

各ItemはString、Number、Binary等の多彩なデータ型を持つことが可能

# データの柔軟性：様々な用途



## キーバリューデータストア

テーブル内のそれぞれのアイテムは属性と値のリストで保持される

1つ以上のAttributeを持ち、アイテム間で不揃いでも問題ない

属性はあとから追加可能

## ドキュメントデータベース

Jsonフォーマットのサポート

## グラフデータベース

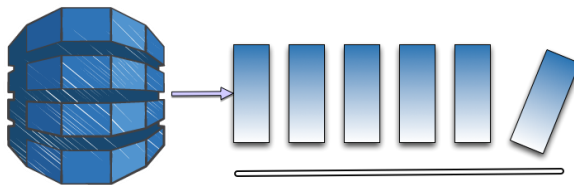
TITAN DBのバックエンドとして利用可能

# Stream 処理で非同期処理や、イベント駆動

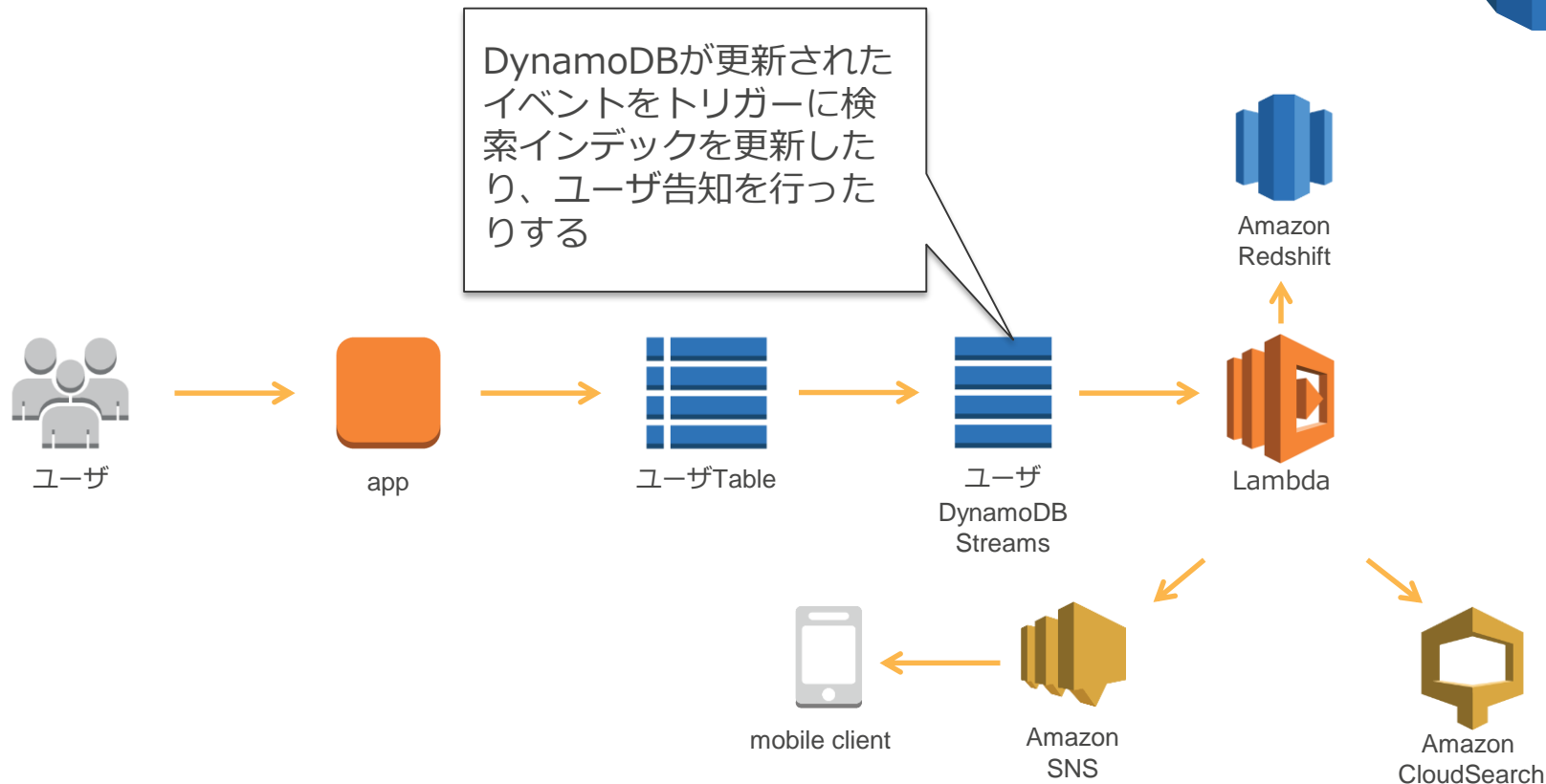


DynamoDB Streamsを使うことでテーブルの更新の情報を非同期（1秒未満）で更新することが可能

- 非同期アプリケーション
- グローバルアプリケーション

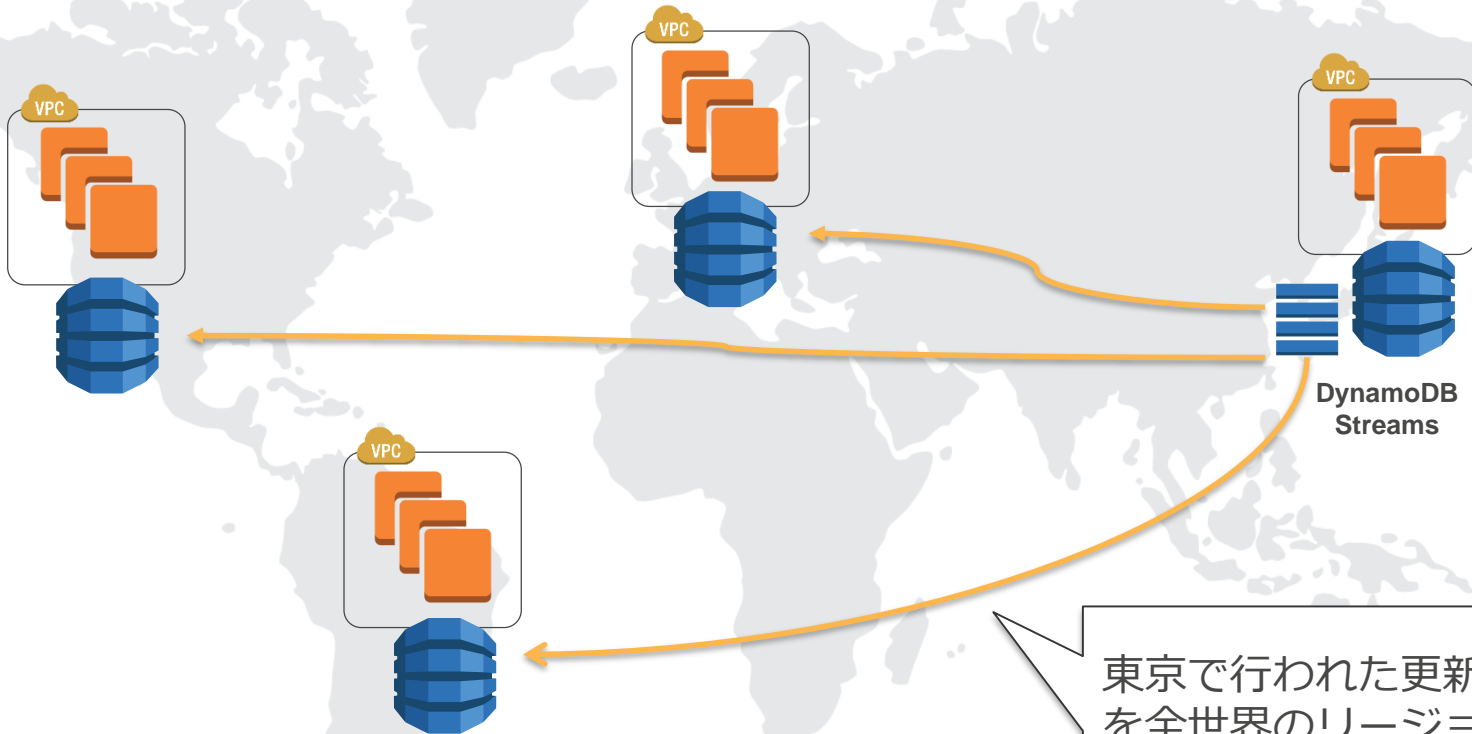


# 非同期アプリケーション





# グローバルアプリケーション



東京で行われた更新  
を全世界のリージョンに高速に反映する  
事が可能



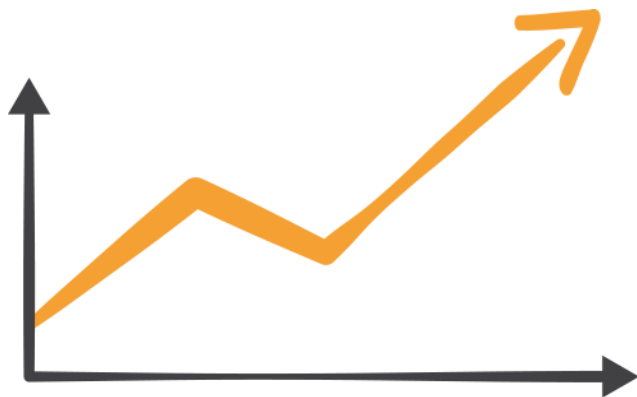
継続的な機能追加

# 継続的な機能追加



AWSの各サービスはAmazon DynamoDBにかぎらず継続的に新機能の追加が行われている

- Time-to-Live
- VPC Endpoint
- DAX



# DynamoDB Time-to-Live (TTL) *New*



一定期間を過ぎたアイテムを非同期に削除する事が可能

- 不要なアイテムを削除しコストを削減
- テーブルサイズを制御してパフォーマンスを最適化する

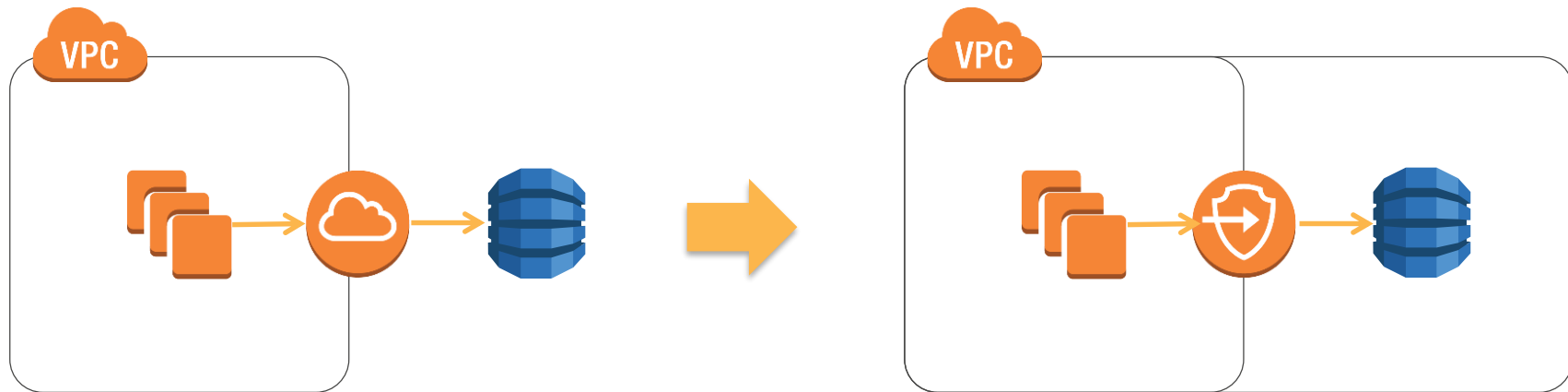
TTL Attribute			
ID	Name	Size	Expiry
1234	A	100	1456702305
2222	B	240	1456702400
3423	C	150	1459207905

**TTL Value**  
(Epoch format)

# VPC Endpoint (パブリックベータ) *New*



各VPC環境にEndpointを提供することができるようになり、  
今までのようにNATやEIPを付ける必要がなくなった



# DynamoDB Acceralator DAX *New*

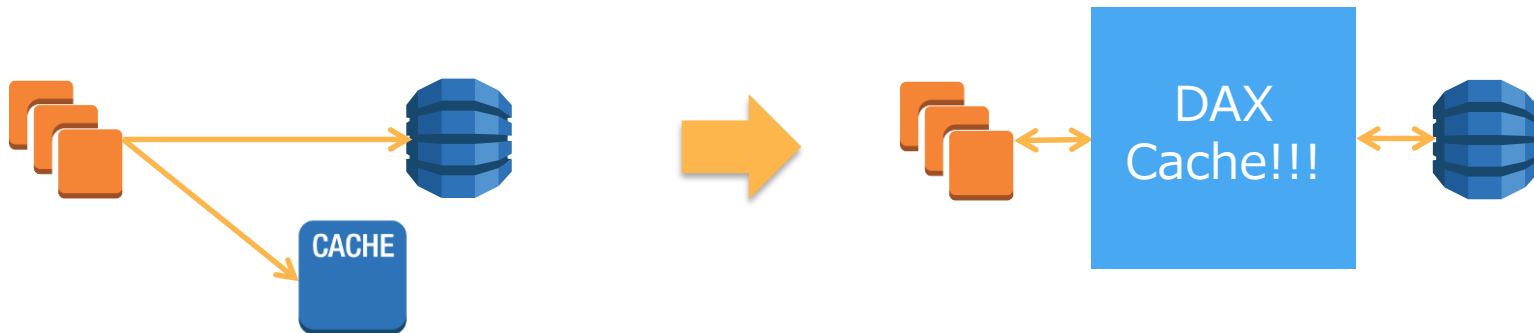
## (パブリックベータ)



DynamoDBの前面にマネージドなキャッシュクラスタを置くことが出来るサービス

読み込みの多いワークロードでは低レイテンシを実現

- パフォーマンス向上
- コスト最適化



# DynamoDBの料金体系



## プロビジョンドスループットで決まる時間料金

- Read/Writeのプロビジョンスループットによって時間あたりの料金決定
- リザーブドキャパシティによる割引

## ストレージ利用量

- 保存したデータ容量（GB当たりの単価）
- GBあたり\$0.285(2017/06/01現在@東京リージョン)

# DynamoDBの料金体系



## プロビジョンドスループット

- **書き込み** : \$0.00742 : 10 ユニットの書き込み容量あたり/1 時間
- **読み込み** : \$0.00742 : 50 ユニットの読み込み容量あたり/1 時間

## キャパシティユニット

- 上記で「ユニット」と呼ばれている単位のこと
- **書き込み** : 1ユニット : 最大1KBのデータを1回/秒書き込み可
- **読み込み** : 1ユニット : 最大4KBのデータを2回/秒読み込み可（強一貫性読み込みであれば1回/秒）



# DynamoDBの料金例



以下の条件の場合の料金試算

- 読み書き – 読み書き共に平均的に100万回ずつアクセス
- 1回あたりの読み書きデータ量 – 1KB以下
- 保存データ量 – 1TB

一ヶ月の料金…**約11.7ドル！！！！**

# 事例：東急ハンズ株式会社様

## Amazon DynamoDB導入の背景

- 毎年8月末にハンズ恒例の大セール「ハンズメッセ」開催。  
ネットストア(ハンズネット) も同時開催  
セール開始時(00:00)にお客様が殺到  
サーバーが落ちる
  - ということを2年くらい経験

コストを抑えつつ、セールの爆発的なトラフィックにも耐えられる仕組みの構築が急務

## 2. 東急ハンズでの活用方法について



### 参考：セール時のスパイクアクセス



参考 DynamoDBの紹介と東急ハンズでの活用についてより。詳細は以下をご参照ください。

<https://www.hands-lab.com/tech/entry/99.html>

# 事例：株式会社リプレーション様

短期間でのユーザー数の急上昇にも  
わずかなスケールパラメータの調整で対応



## AWSの利用:

人気のドット絵ファンタジー  
「騎士とドラゴン」のインフラ  
をAWSで運用

## ビジネス効果:

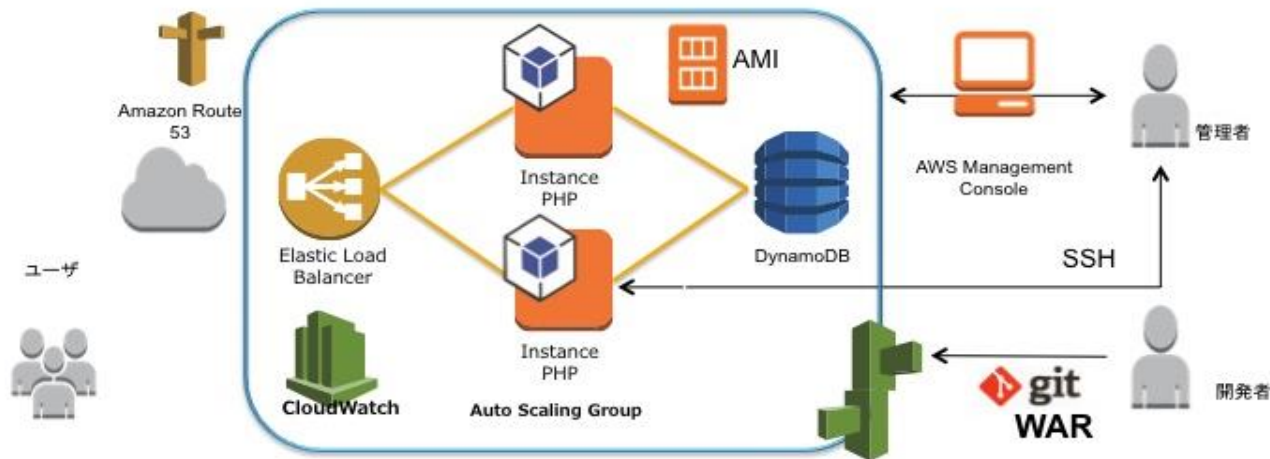
少数精鋭で開発を行う中で、運用  
面の負担を最小化、ゲームの開発  
リソースを最大化

# 事例：株式会社リプレーション様

デイリーアクティブユーザーが6日間で5,000/日から  
30,000/日に跳ね上がった時に行った作業

「AWS EC2のインスタンス数を増やす」

「Amazon DynamoDBのスループットを上げる」だけ



# DynamoDBの使いどころ

- 堅牢かつ高スケールするシステムの構築
- スループットが増減するようなピーク帯のあるシステム
- 大量のデータが貯めていくようなシステム

詳しくはAWS Black Belt DynamoDB資料を参照

<http://www.slideshare.net/AmazonWebServicesJapan/aws-black-belt-tech-amazon-dynamodb>

# NoSQLの使い方

# サービスの使い分け

- NoSQLサービスをどのように使い分けていくか
- ここではElastiCacheとDynamoDBでの用途の違いについてまとめる



# ElastiCache の主なユースケース

## アプリケーション高速化のためのキャッシュ

- アプリケーションキャッシュ
- セッションキャッシュ

## Redisの機能を実現したい場合

- Pub/Sub
- リーダーボード
- リアルタイム分析
- 位置情報処理





# DynamoDBの主なユースケース

Webアプリケーションの永続的なセッションデータベース

ユーザー情報を格納するデータベース

広告やゲームなどのユーザー行動履歴DB

- ユーザーIDごとに複数の行動履歴を管理データベース
- 保存できる容量も無制限

ソーシャルアプリのバックエンド

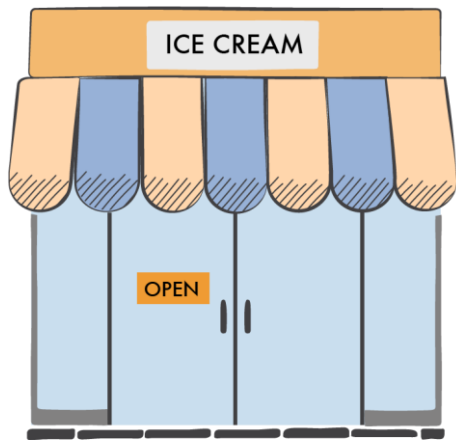
その他



# ECサイトを例に取ったデータモデリングの違い

ECサイトを構築する場合にRDBMSとDynamoDBの場合のデータモデリングの違いについて

同じことをする際に使うものが違うとデザインが変わる



# SQL (リレーショナル・データベース)の場合

## Products

Columns			
Primary Key		Index	
Product ID	Type	Price	Desc.
1	Book	\$11.50	One of 2 major ...
2	Album	\$14.95	The Partitas
3	Movie	\$8.99	Chaplin's first ...

# SQL (リレーショナル・データベース)

## Products

The diagram shows a table with four columns: Product ID, Type, Price, and Desc. The 'Product ID' column is highlighted in orange and labeled 'Primary Key' with a brown arrow. The 'Type' column is highlighted in blue and labeled 'Index' with a blue arrow. A bracket above the columns is labeled 'Columns'. A bracket to the left of the rows is labeled 'Rows'. A brown arrow points from the '1' in the 'Product ID' column of the first row to the 'Books' table below.

Product ID	Type	Price	Desc.
1	Book	\$11.50	One of 2 major ...
2	Album	\$14.95	The Partitas
3	Movie	\$8.99	Chaplin's first ...

## Books

Book ID	Author	Title	Date
1	Homer	Odyssey	1871

# SQL (リレーショナル・データベース)

## Products

Primary Key		Index	Columns	
Product ID		Type	Price	Desc.
Rows	1	Book	\$11.50	One of 2 major ...
	2	Album	\$14.95	The Partitas
	3	Movie	\$8.99	Chaplin's first ...

## Books

Book ID	Author	Title	Date
1	Homer	Odyssey	1871

## Movies

Movie ID	Title	Genre	Director
3	The Kid	Drama, Comedy	Chaplin

# SQL (リレーショナル・データベース)

## Products

Columns			
Product ID	Type	Price	Desc.
1	Book	\$11.50	One of 2 major ...
2	Album	\$14.95	The Partitas
3	Movie	\$8.99	Chaplin's first ...

## Books

Book ID	Author	Title	Date
1	Homer	Odyssey	1871

## Albums

Album ID	Artist	Title
2	Bach	6 Partitas

## Movies

Movie ID	Title	Genre	Director
3	The Kid	Drama, Comedy	Chaplin

# ではDynamoDBの場合は？

## Products

Primary Key

Index

Columns

Product ID	Type	Price	Desc.
1	Book	\$11.50	One of 2 major ...
2	Album	\$14.95	The Partitas
3	Movie	\$8.99	Chaplin's first ...

Rows

## Books

Book ID	Author	Title	Date
1	Homer	Odyssey	1871

## Albums

Album ID	Artist	Title
2	Bach	6 Partitas

## Movies

Movie ID	Title	Genre	Director
3	The Kid	Drama, Comedy	Chaplin

## Tracks

Album ID	Track ID	Track
2	1	Partita No. 1

Primary Key

Partition Key

Sort Key

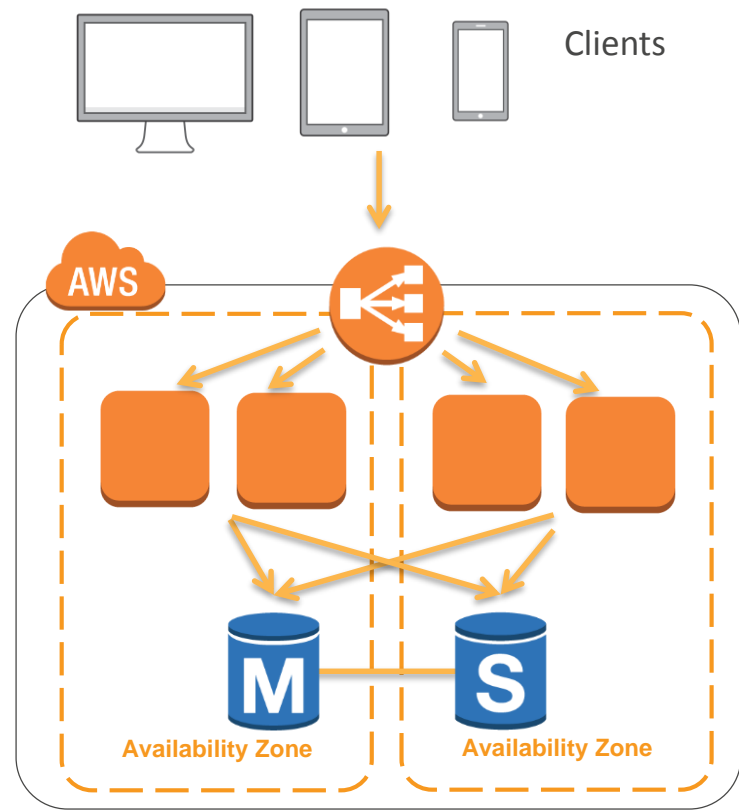
## Products

Attributes

Product ID	Type	Item 毎のスキーマデザイン		
1	Book ID	Odyssey	Homer	1871
2	Album ID	6 Partitas	Bach	
2	Album ID: Track ID	Partita No. 1		
3	Movie ID	The Kid	Drama, Comedy	Chaplin

# AWS上でのWeb + DBサーバの構成その1

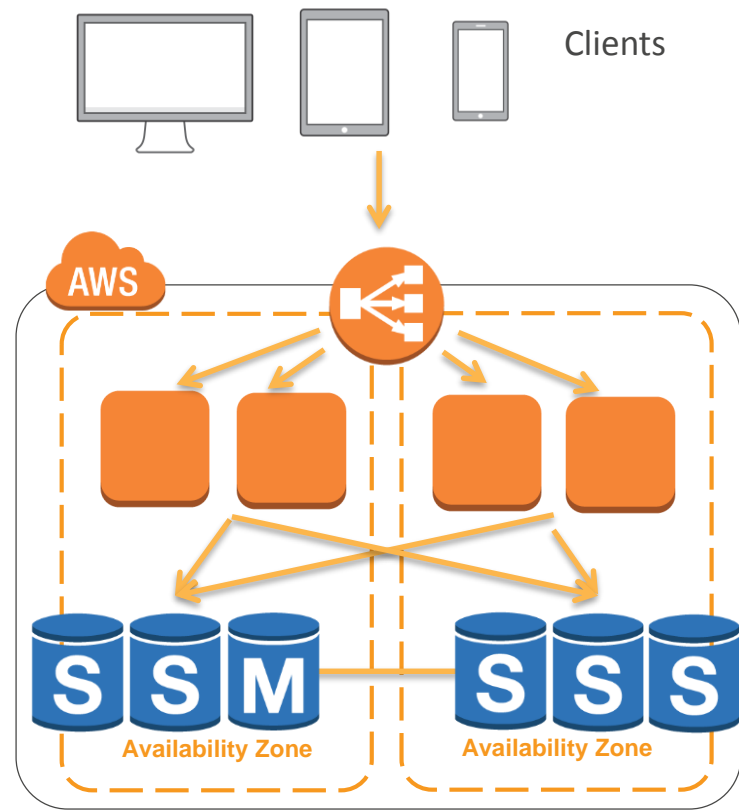
- どのようにデータベースを使い分けていくかの例
- 一般的なゲームやSNS等のアプリケーション
- RDBMS使用し構築
- スケールアウトに難があり運用が徐々に大変に





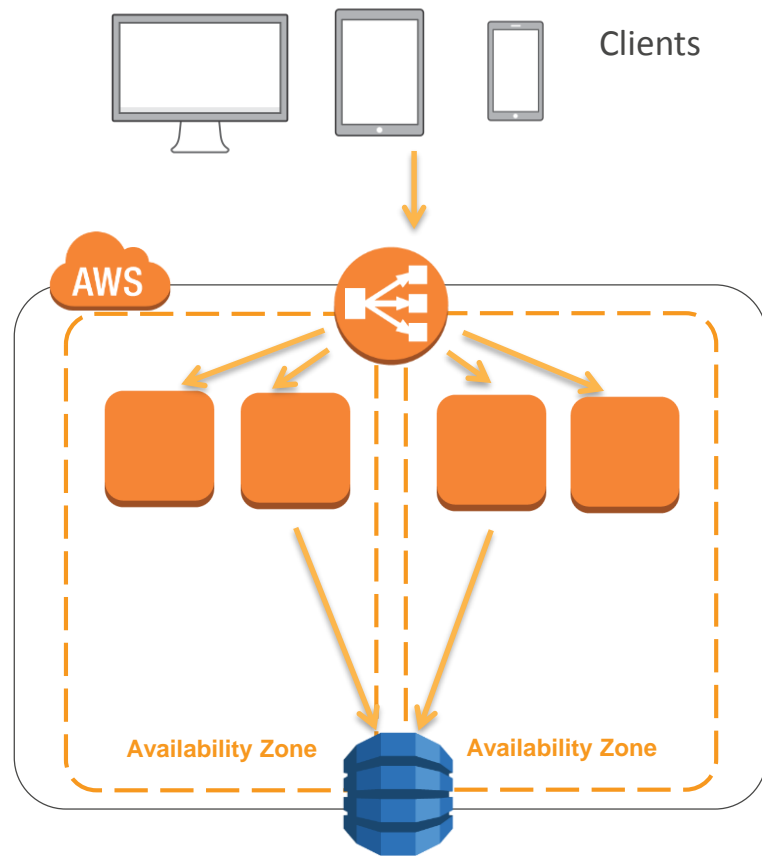
# AWS上でのWeb + DBサーバの構成その2

- 読み込みの負荷が非常に多くなった結果
- 性能面で課題がでてきた
- 読み込みを分散するためにDBスレーブを増設し、コスト増



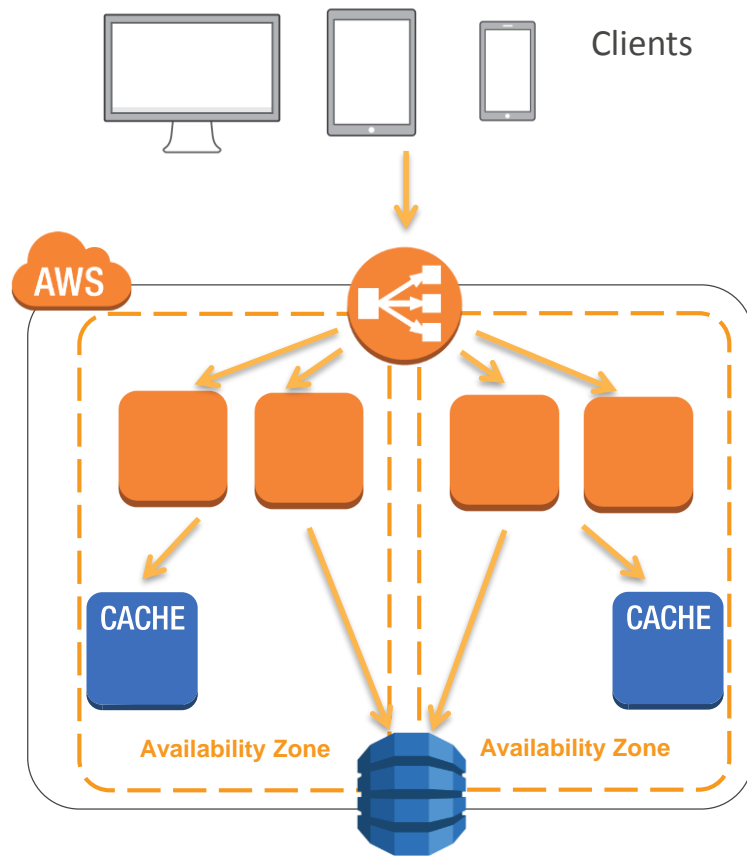
# AWS上でのWeb + DBサーバの構成その3

- RDBMSをDynamoDBへリプレイス、スケールアウトやプロビジョンキャパシティにより安定した性能を出すことができ、ストレージ容量も気にしなくて良くなった
- と、思ったら読み込みレイテンシの問題が発生、1ms以内のアクセスが必要な処理が必要に、、、！



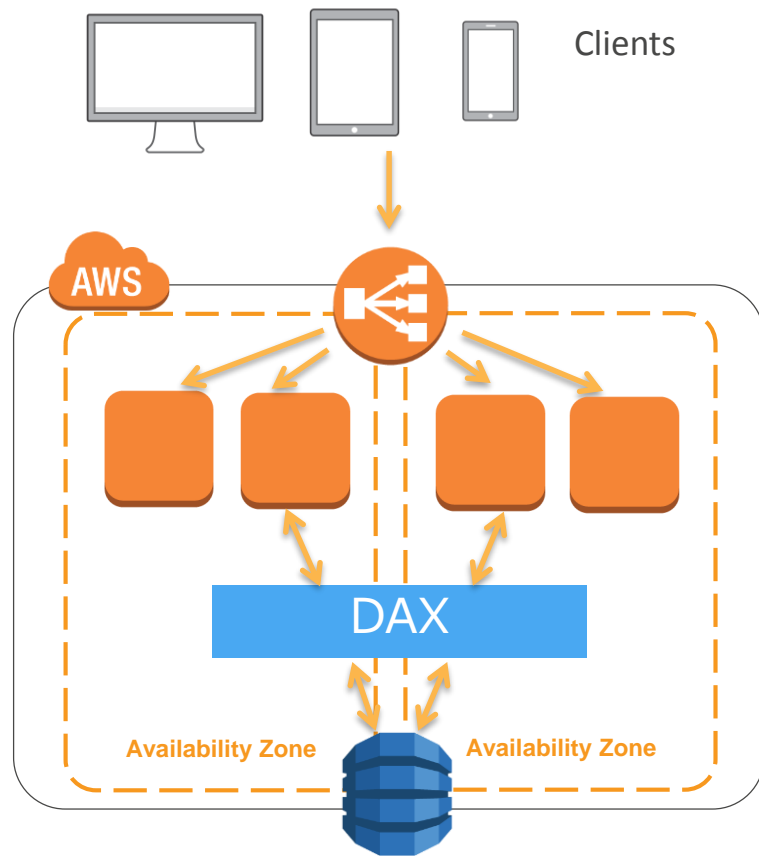
# AWS上でのWeb + DBサーバの構成その4

- ElastiCacheを使用し、DynamoDBへのクエリの一時的キャッシュを行う
- レイテンシの問題をクリア、1ms以内のアクセスが可能に



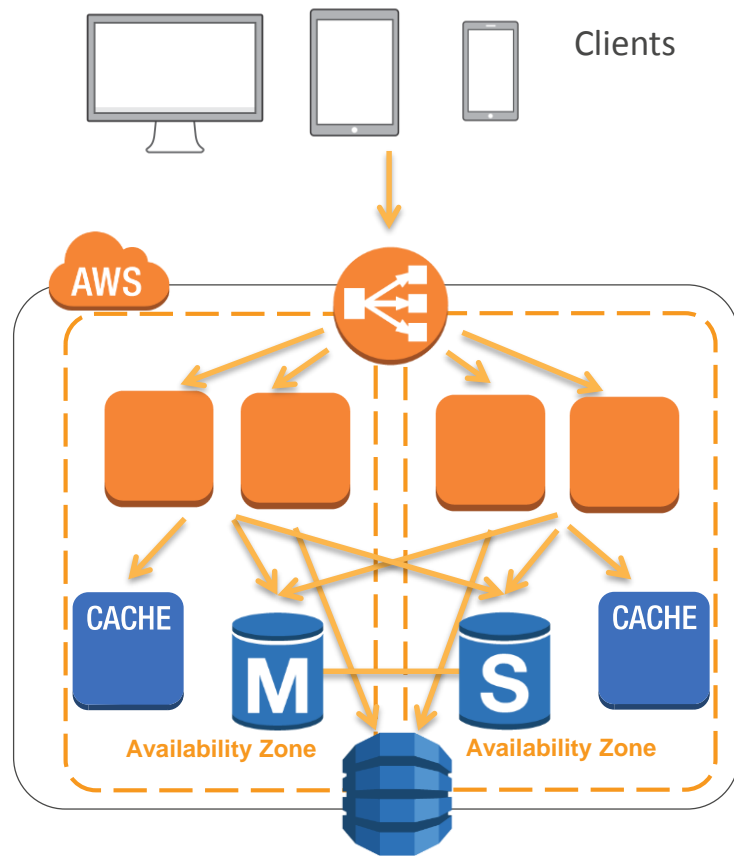
# AWS上でのWeb + DBサーバの構成その4の2

- DAX（プレビュー中）を活用することで、実装を変えずに読み込みの低レイテンシを実現することが可能
- 運用負荷の低減



# AWS上でのWeb + DBサーバの構成その5

- RDBMSとNoSQLの併用のパターン
- 課金部分などの強いトランザクション処理が欲しい処理にはRDBMSを使用
- スケールアウトをしたい所にはDynamoDBを使用



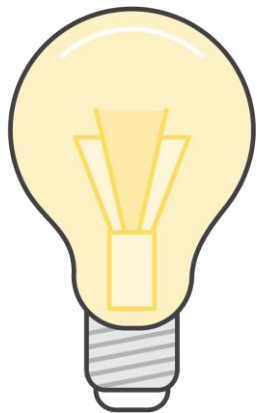
# まとめ

# まとめ

- RDBMSとNoSQLは適材適所！そのNoSQLが一番フィットする部分に導入検討する
- DynamoDB、ElastiCacheを上手く活用することで従来RDBMSだけでは実現が難しかった「柔軟なデータ構造」「低いレイテンシ」「スケーラビリティ」等を実現することが可能に
- 最小限の労力で、今まで実現が難しかった種類のアプリケーションを構築できる

本日持って帰って頂くこと

**NoSQLを最大限に活用し  
可用性の高いシステムを作る！**





# 本セッションのFeedbackをお願いします

受付でお配りしたアンケートに本セッションの満足度やご感想などをご記入ください  
アンケートをご提出いただきました方には、もれなく**素敵なAWSオリジナルグッズ**を  
プレゼントさせていただきます



アンケートは受付、パミール3FのEXPO展示会場内にて回収させていただきます

# AWS ソリューション Day 2017- Database Day

## ～すでに始まっている！「クラウドへのデータベース移行」と「データレイクを軸としたビッグデータ活用」～

### Database Dayとは？

ユーザー企業/パートナー/AWSによる導入事例や活用動向また技術情報をご紹介します  
するIT部門（エンジニア・管理者など）向けのカンファレンス

### 開催日時・会場

- 2017年7月5日(水) 10:00～17:30（9:30開場予定）
- 大崎ブライトコアホール（JR大崎駅より徒歩5分）

### セッション

①基調講演 ②ブレイクアウトセッション – 2トラック構成

トラック1：データベース移行（事例セッションあり）

トラック2：データレイク（JAWSUG-BigData支部 事例セッションあり）

### お申込み

<https://aws.amazon.com/jp/solutiondays20170705/>

AWS

S U M M I T

Thank you !

