

セクション8

2023年7月24日 15:36

アーキテクチャの課題：大量アクセスによりアクセスできない→オートスケール機能

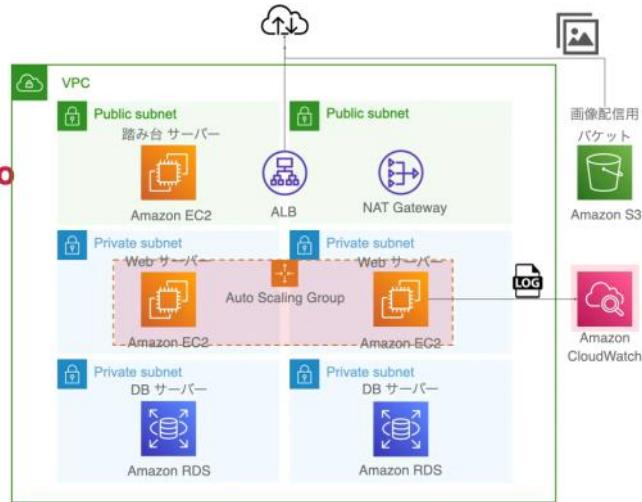
1. Amazon EC2 Auto Scaling の概要

2. Amazon CloudWatch の概要

3. [ハンズオン] Web サーバーの Auto Scaling 設定

4. [ハンズオン] Web サーバーの ログ出力設定

5. 振り返り



Amazon EC2 Auto Scaling



2023年7月25日 8:39

8-59.Amaz
onEC2Aut...

Amazon EC2 Auto Scaling：利用状況に応じて自動でEC2インスタンスの数を増減する機能

例) 使用率80%を超えたたら1つ増やす下がったら1つ減らす 最小数、最大数を決める

起動設定：Auto Scalingでインスタンスを作成するのに必要な情報の設定

Auto Scalingグループ：Auto Scalingの管理をする設定

メトリクス：データのようなもの

スケーリング方法

- ・手動スケーリング：利用者が設定変更することでEC2インスタンスの数を変更
- ・スケジュールスケーリング：定義したスケジュールに基づいて変更
- ・動的なスケーリング：3つのスケーリングポリシー

動的なスケーリング

シンプルなスケーリング：1つのメトリクスに対して、1つのしきい値を設定

ステップスケーリング：1つのメトリクスに対して、複数のしきい値を設定

ターゲット追跡スケーリング：1つのメトリクスに対して、ターゲット値を設定

クールダウン：指定した時間内はAuto Scaling を実行しない設定

ウォームアップ：複数閾値を超過時、差分のEC2インスタンスを作成

料金は無料

CloudWathch

2023年7月25日 8:51



8-60.Cloud
Watch

Amazon CloudWatch : モニタリングに関する機能を提供するサービス

標準メトリクス : AWSがあらかじめ定義している

カスタムメトリクス : 利用者が定義したメトリクス (メモリ使用率やディスク使用率など)

Amazon CloudWatch Alarms : メトリクスのしきい値を超えたらアクションする機能

Amazon CloudWatch Logs : ログファイルの保存、閲覧、監視ができる機能

Auto Scaling設定

2023年7月25日 8:57

手順

起動テンプレートの作成

Auto Scaling グループの作成

既存のEC2インスタンスの削除

テンプレート変更

- ・テンプレート変更(新しいバージョンを作成)
- ・ソーステンプレート選択(そこからの差分を変更していく)
- ・テンプレート変更後にAuto Scalingの「編集」からテンプレートのバージョンを変更する

ELBのヘルスチェックをする理由

<https://docs.aws.amazon.com/autoscaling/ec2/userguide/as-add-elb-healthcheck.html>

Auto Scaling グループのデフォルトのヘルスチェックは EC2 ヘルスチェックのみ

ELBにパスしないインスタンスがあると自動的に終了する

ログ

2023年7月25日 10:38

攻撃などでインスタンスが落ちてしまったときにログを残す

CloudWatch Logs リアルタイムでログを取れるが割高

```
sudo yum install -y amazon-cloudwatch-agent
sudo touch /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json
sudo vi /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json
sudo systemctl start amazon-cloudwatch-agent.service
systemctl status amazon-cloudwatch-agent.service
sudo systemctl status amazon-cloudwatch-agent.service
sudo systemctl enable amazon-cloudwatch-agent.service
sudo systemctl is-enabled amazon-cloudwatch-agent.service
/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json
```

```
{
  "logs": {
    "logs_collected": {
      "files": {
        "collect_list": [
          {
            "file_path": "/var/log/httpd/access_log",
            "log_stream_name": "{instance_id}",
            "log_group_name": "/var/log/httpd/access_log",
            "timestamp_format": "%d/%b/%Y:%H:%M:%S %z",
            "timezone": "UTC"
          },
          {
            "file_path": "/var/log/httpd/error_log",
            "log_stream_name": "{instance_id}",
            "log_group_name": "/var/log/httpd/error_log",
            "timestamp_format": "%d/%b/%Y:%H:%M:%S %z",
            "timezone": "UTC"
          }
        ]
      }
    }
  }
}
```

セクション11

2023年7月25日 13:35

サーバーレス

2023年7月25日 14:02

サーバーレス：サーバーの存在を意識しなくてもアプリケーション開発をすることができる。

サーバーレスの概要

▶ サーバーレス

- ▶ サーバー管理を必要としないアプリケーションを構築して実行するという概念

 物理サーバー	<ul style="list-style-type: none">▶ OSのパッチ適用▶ ミドルウェアのパッチ適用▶ 定期的な再起動▶ 耐障害性・高可用性のための冗長構成▶ バックアップ取得・保管
 仮想サーバー	 サーバーレス <ul style="list-style-type: none">▶ インフラのプロビジョニング、管理が不要▶ スケーラビリティの管理が不要▶ アイドル時のコストが不要▶ 耐障害性・高可用性が担保される

▶ 責任共有モデルの比較



▶ Functions-as-a-Service (FaaS)

- ▶ 関数を使用して、アプリケーションコードを実行および管理

▶ Backend-as-a-Service (BaaS)

- ▶ 認証やデータベースなど、アプリケーションが使用する機能を API 経由で利用できる



イベントドリブン：イベントによってトリガーされること

サーバーレスのメリット

- ・サーバーの事前プロビジョニングや管理なしでコードの実行環境を得られる→アプリケーションの開発に集中できる
- ・コスト削減（イベントドリブンでイベント発生時のみ課金）
- ・マイクロサービスに向いている

※マイクロサービス：アプリケーションがもつ機能を細かいサービスに分割し、それぞれのサービスを連携させて動かす開発手法

サーバーレスのデメリット

- ・使用上の制約がある（AWS Lambda：最大実行時間が15分、同時実行数が1000まで）
- ・コールドスタート：なにも起動していない状態からインフラストラクチャーの起動するので実行に時間がかかる
 常時起動しているサービスには向いていない
- ・大規模な基幹システム、長時間のバッチ処理には向かない
- ・監視が複雑（いくつものサービスを組み合わせて使うことが多いので）

セクション12

2023年7月25日 14:20



or

→
ドメインを取得

コーポレートサイト用ドメイン
example.com

freenom

ドメイン取得

2023年7月25日 14:24

「Route53]→「ドメインの登録」

Freenom

2023年7月25日 14:30

オランダ、アムステルダムのドメインプロバイダ

以下であれば無料で独自ドメインを取得可能

.TK(トケラウ)

.ML(マリ)

.GA(ガボン共和国)

.CF(中央アフリカ)

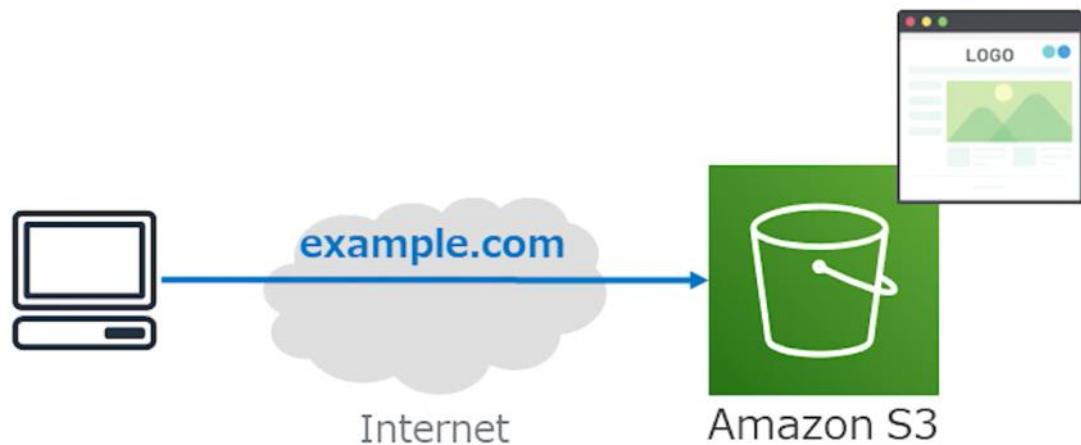
.GQ(赤道ギニア)

構築手順

- 1.Freenomでドメインを取得
- 2.Route53でホストゾーン作成
- 3.Route53へFreenomで取得したドメインの委任

セクション13

2023年7月25日 14:41



Amazon S3 ウェブサイトホスティング

2023年7月25日 14:50

Amazon S3 ウェブサイトホスティング：S3バケットに保管している静的コンテンツをインターネットにウェブページとして公開する機能

静的コンテンツ：HTML・CSS・JPGなど

Amazon S3 ウェブサイトホスティングの利用料は無料

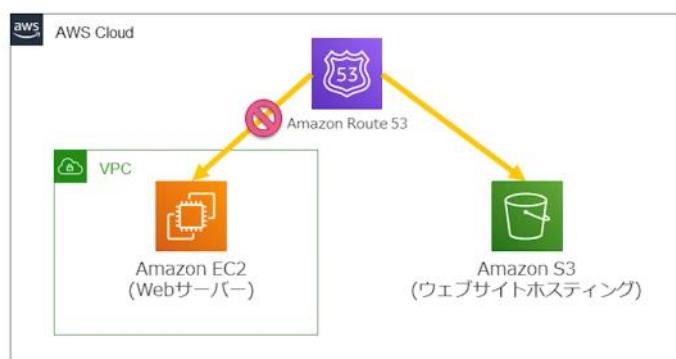
Amazon S3のストレージ料とリクエストに応じた料金が発生

▶ 特徴

- ▶ スケールを考慮する必要がなく、アクセス集中に強い
 - ▶ AWS 側で自動でスケール
 - ▶ 運用にかかるコストを減らすことができる
 - ▶ サーバーの管理が不要
- ▶ 公開できるのは静的コンテンツのみ

ユースケース

- ▶ メンテナンスや障害時に表示するSorryページ



独自ドメインの利用

- ▶ 独自ドメインが利用可能 (例: example.com)

- ▶ バケット名に独自ドメインのFQDNを設定

http://blog.example.com.s3-website-ap-northeast-1.amazonaws.com



http://**blog.example.com**



今回はCloudfrontのものを使用する

手順

2023年7月25日 14:56

1.S3バケットの作成

パブリックアクセスは有効にしない
名前が世界で一意でないと作成できない

2.静的ウェブサイトホスティング機能の有効化

バケットのプロパティから「静的ウェブサイトホスティングを編集」
インデックスドキュメントに「index.html」 ←htmlファイルのパス指定を省略できる
エラードキュメントにも「index.html」 ←事前に準備されたサイトがSPA(Single Page Application)で実装されている為
SPA：単一のページでウェブアプリケーションを構成する設計構造

セクション14

2023年7月25日 15:36

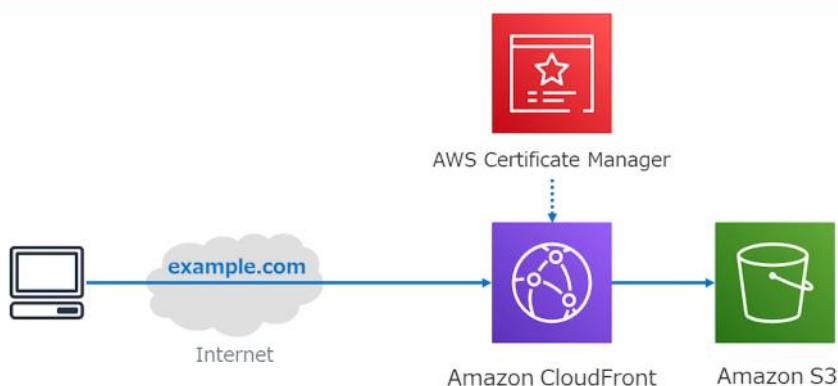
海外ユーザーに対して、高速配信をするためにCloudfrontを設定する

アーキテクチャの課題

ネットワーク遅延は（物理的、ネットワーク的な）距離に依存



作成する構成



CloudFront

2023年7月25日 15:37

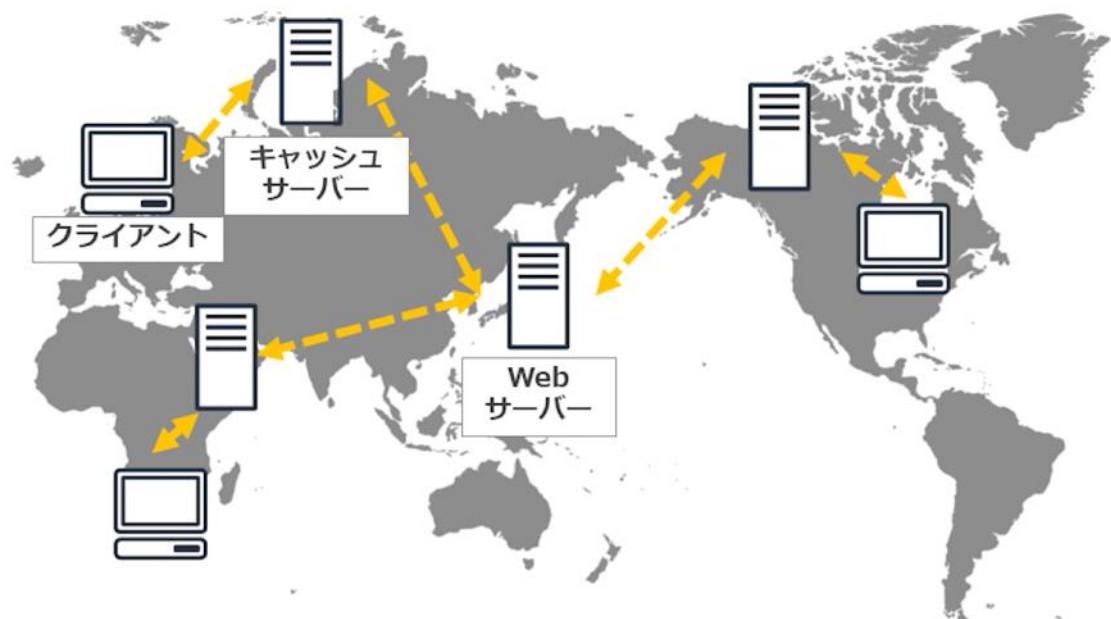
キャッシュを利用して通信を高速化する

S3へのアクセス数を減らすことができるのでコスト低減

エッジロケーションの利用でより近くからのコンテンツの配信

Contents Delivery Network (CDN)

- ▶ ウェブコンテンツを効率的に配信する仕組み



Amazon CloudFront

- ▶ エッジロケーションのサーバーでコンテンツをキャッシュして代理配信

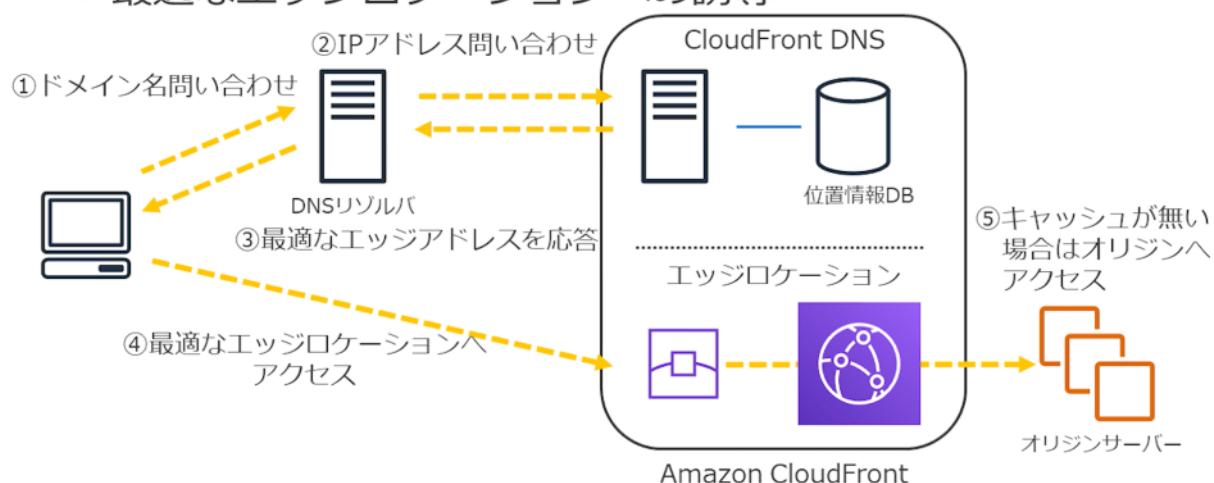


エッジロケーション

- ▶ 低レイテンシーでレスポンスを提供するためのデータセンターで、世界中に配置されている
- ▶ ユーザーから地理的に近い場所から応答することで、低レイテンシーを実現
- ▶ 利用サービス
 - ▶ **Amazon CloudFront**, Amazon Route 53, AWS WAF, AWS Shield, Lambda@Edge, Amazon API Gateway

Amazon CloudFront

▶ 最適なエッジロケーションへの誘導



ディストリビューション

CloudFrontの設定

ビヘイビア

キャッシュのルール設定

SSL/TLS

2023年7月25日 16:22

SSL：インターネット上の通信を暗号化するための仕組み

通信プロトコルの一種

TLS：SSLのセキュリティを強化した後続プロトコル

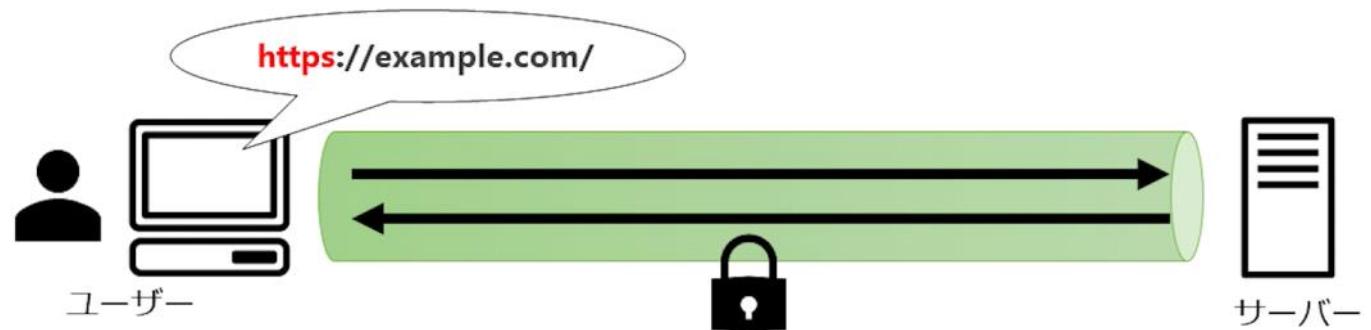
SSL/TLS：SSLもしくはTLS、あるいはその両方を指す

SSL/TLS暗号化通信

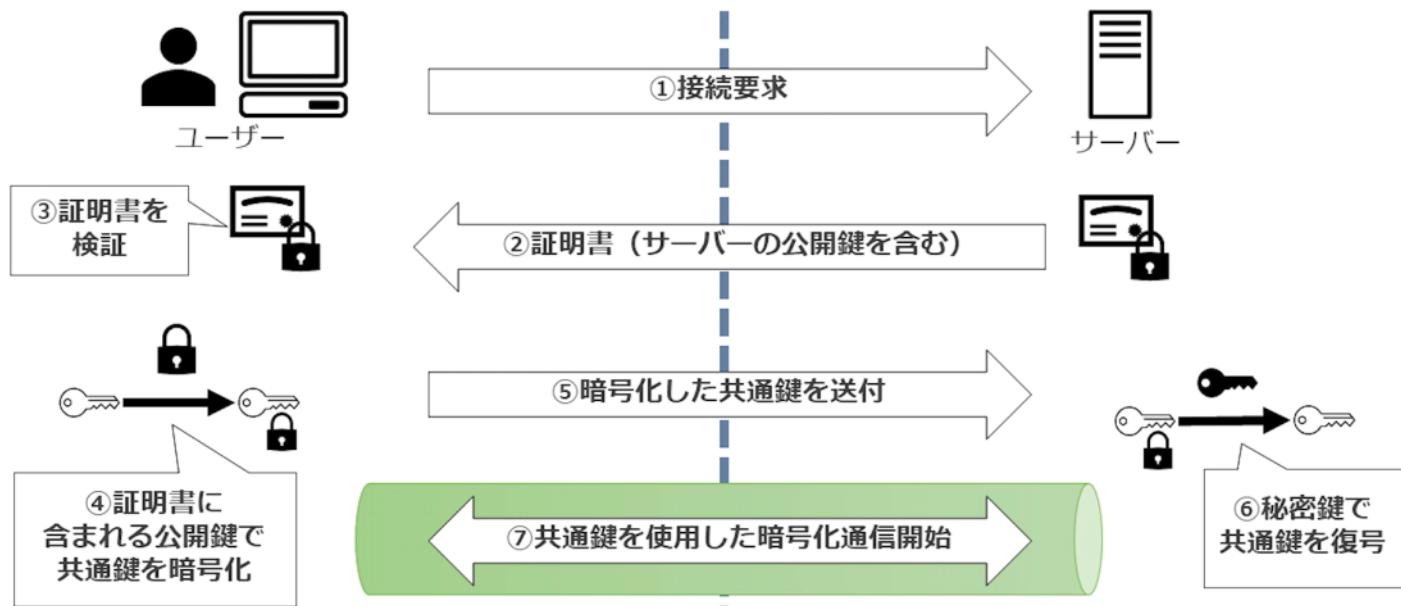
SSL/TLSプロトコルで通信を暗号化する

データは暗号化されるため、複号ができない第三者は内容を読みたくことができず、不正な読み取りを防ぐことができる

HTTPS：SSL/TLSプロトコルにより暗号化されたHTTP通信のこと



SSL / TLS 暗号化通信



SSL/TLS証明書

2023年7月25日 16:56

SSL/TLS証明書の役割

- 通信の暗号化
- 通信相手に偽りがないことを保証
- 認証局によって署名されている
- 市販のSSL/TLS証明書が必要

SSL/TLS証明書の運用上の課題

- 証明書にかかるコスト
- Webサーバーやドメインの数だけ証明書が必要
- 運用にかかる負荷
 - サーバー一台ごとに証明書をセットアップする手間
 - 更新忘れ

ACM

2023年7月25日 17:03

AWS Certificate Manager (ACM) の特徴

- ▶ AWS クラウド上で SSL/TLS 証明書を簡単にプロビジョニング、管理、展開、更新が可能
- ▶ 証明書の発行や更新の手間が削減できる
- ▶ AWS クラウド上で証明書を集中管理できる
- ▶ サードパーティ証明書のインポートと展開が可能
- ▶ ACM でプロビジョニングされた
パブリック SSL/TLS 証明書は無料



AWS Certificate Manager

AWS Certificate Manager (ACM) の機能

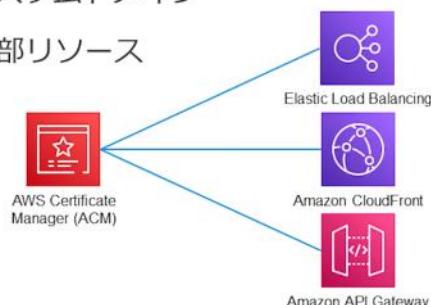
- ▶ 証明書の発行機能
 - ▶ パブリック DV 証明書の発行
(DNS と E メールによるドメイン認証)
 - ▶ プライベート証明書の発行
- ▶ 証明書の展開・更新機能
 - ▶ 発行した証明書の展開
 - ▶ インポートした証明書の展開



AWS Certificate Manager

AWS Certificate Manager (ACM) の機能

- ▶ 対象サービス
 - ▶ Elastic Load Balancing (ELB)
 - ▶ Amazon CloudFront ディストリビューション
 - ▶ Amazon API Gateway 上の API カスタムドメイン
 - ▶ EC2 やオンプレミスサーバー等の内部リソース
(プライベート証明書)



AWS Certificate Manager (ACM) の機能

- ▶ (例) CloudFront ディストリビューションに証明書を設定



料金

- ▶ ACM でプロビジョニングされた
パブリック SSL/TLS 証明書は無料
- ▶ 支払いはアプリケーションを実行するために作成した
AWS リソースの料金のみ
- ▶ AWS Certificate Manager プライベート認証機関を
使用する場合は料金が発生

構築手順

2023年7月25日 17:11

1. ACMでSSL証明書を発行

「ACM」→「発行リクエスト」→ドメイン名記入後DNS検証をする

Route53でCNAMEレコードを作成 (.acm-validations.aws.) のレコードが追加されている
発行済 & ステータス : 成功を確認する

2. CloudFrontディストリビューションの作成

・オリジンドメインの設定 : S3を選択する

・代替ドメイン名(CNAME)オプション

ドメイン名を記入

作成した証明書を選択する

・デフォルトルートオブジェクトにindex.html

3. Route53にAレコードを作成し、CloudFrontに設定

The screenshot shows the 'Record Type' section of the Route 53 Record Creation Wizard. It displays the following fields:

- Record Name:** subdomain (highlighted in blue)
- Type:** A - IPv4 (highlighted in blue)
- Alias Target:** CloudFront Distribution (highlighted in blue)
- Hosted Zone:** 米国東部 (バージニア北部) (highlighted in blue)
- Value:** d2ifslgkr8qu5f.cloudfront.net (highlighted in blue)
- Health Check:** Simple (highlighted in blue)
- Policy:** None (highlighted in blue)

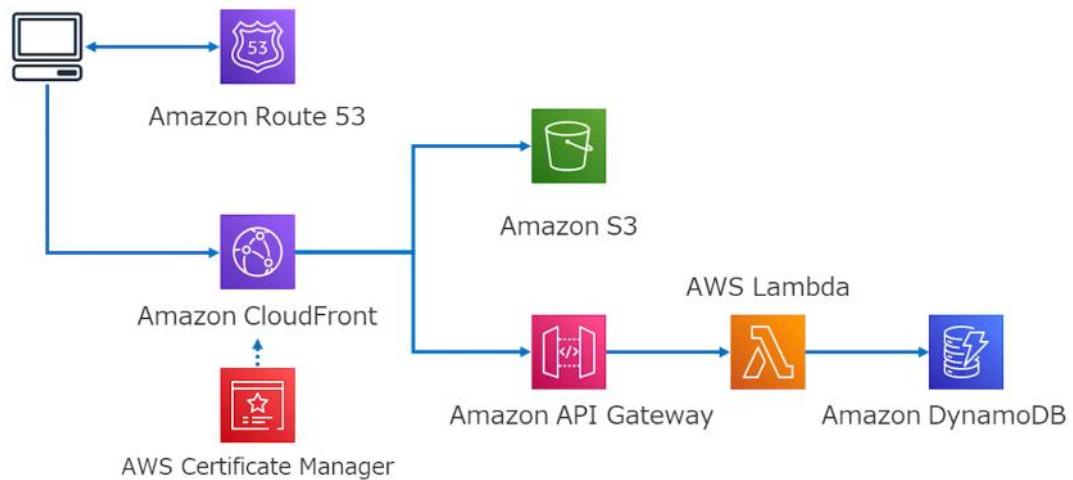
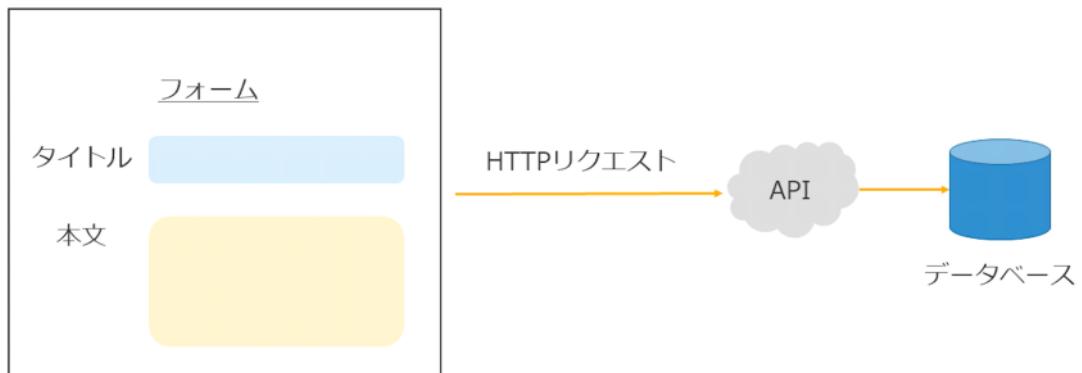
At the bottom right, there are 'Cancel' and 'Create Record' buttons.

4. ドメイン名でアクセスできることを確認

セクション15

2023年7月25日 18:05

作成する入力フォーム



NoSQL

2023年7月25日 18:13

NoSQL(Not Only SQL)とは、リレーションナルデータベース以外のデータモデルの総称

種類

▶ キーバリュー

- ▶ キーとバリューの単純な構造

```
{  
  "id0001": "佐藤",  
  "id0002": "田中",  
  "id0003": "加藤"  
}
```

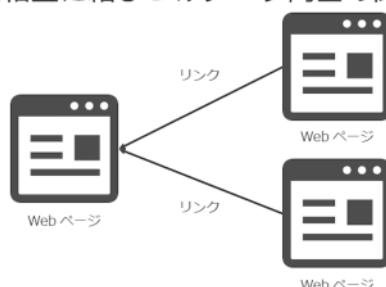
▶ ドキュメント指向データベース

- ▶ JSON や XML などのデータ構造に対応

```
[  
  {  
    "ユーザーID": "0001",  
    "基本情報": {"名前": "佐藤", "性別": "男"},  
    "趣味": ["映画", "音楽"],  
  },  
  {  
    "ユーザーID": "0002",  
    "基本情報": {"名前": "鈴木", "年齢": 23, "出身": "東京"},  
    "評価": {"英語": 79, "国語": 98}  
  }  
]
```

▶ グラフ指向データベース

- ▶ データ間を相互に結びつけデータ同士の関係を表現



SQL（リレーションナルデータベース）と NoSQL の比較

SQL (リレーションナルデータベース)	NoSQL
強固な一貫性をもつ	トランザクションは存在しない
トランザクション処理が可能	一度に高速な処理が可能
SQL 準拠の柔軟なクエリ	単純な構造のデータを高速処理
行、列、参照制約などの 厳密な定義	半構造データ対応など、 柔軟なデータモデリング

特徴を理解して選択することが重要

Amazon DynamoDB

2023年7月25日 18:17

フルマネージドなNoSQLデータベースサービス

キーバリュー型とドキュメント指向型の両方をサポート

キャパシティユニットという単位で読み込み、書き込みができるので高い拡張性

DynamoDB Acceleratorというフルマネージドのインメモリキャッシュを使用 マイクロスケールのレイテンシーを実現できる複数のアベイラビリティゾーンに自動的に保存

容量は必要に応じて自動的に増える

パーティションキーでしかデータを引けない

ソートキーが指定されている場合は パーティションキー+ソートキー が重複しなければ大丈夫

▶ フルマネージドな NoSQL データベース

- ▶ 高い拡張性、低レイテンシー(DynamoDB Accelerator)
- ▶ 高可用性 - 複数のアベイラビリティゾーンに保存
- ▶ ストレージの容量制限がない



Amazon DynamoDB

1. NoSQLとは

>>関係データベース管理システム(RDBMS)以外のデータベース管理システムを指す総称。Not only SQLの略。

膨大な量のデータを高速かつ動的に整理し分析することを可能にする、非リレーションナルな広域分散データベースシステム。

①NoSQLの利点と欠点

[利点]

- ・RDBMSよりもスケールアウト(サーバを増加させることで性能をアップ)が容易
- ・スキーマレスなので、カラムを自由に定義することができる
- ・トランザクションやテーブル間のジョインのような仕組みがないものが多い

[欠点]

- ・参照や追加処理が大半のシステムに適している。更新や削除処理が多い仕組みには向かない
- ・複数サーバにデータが分散している場合は、サーバ間の整合性が保証されないケースが多い

②NoSQLの種類

○キー・バリュー型

- ・Key-Valueを単位としてデータを格納する。
- ・シンプルで応答が早い。

○カラム志向型

- ・一般的の行志向型DBと同様に表の構造を持つつ、カラム単位でデータを保持する。
- ・行志向では苦手な列単位の大量集計、大量更新が得意。

○グラフ型

- ・ノード、リレーションシップ、プロパティによって定まるデータを単位とし、全体でグラフを形成する。
- ・グラフ型の名前の通り、Facebookの知り合い機能等で有効に利用できる。

○ドキュメント志向型

- ・JSONやXMLのような構造を持ったドキュメントを単位としてデータを格納する。
- ・スキーマレスで格納できる。

2. Redis

>>NoSQLデータベースの一つ。メモリ上で動作するキー・バリュー型の揮発性データベースである。

KVSは任意の保存したいデータ(値:value)に対し、対応する一意の標識(キー:key)を設定し、これらをペアで保存するデータベースの一種で、Redisはコンピュータ上のメインメモリ上にKVSを構築し、外部のプログラムからデータの保存と読み出しができる。

※揮発性であるが、Redisはデータをディスクに書き込み、永久化させる機能を持つため、Redisが停止してもデータは消えない。

○操作方法

・パッケージのインストール

```
#apt-get install redis-server
```

・データの書き込み・読み込み

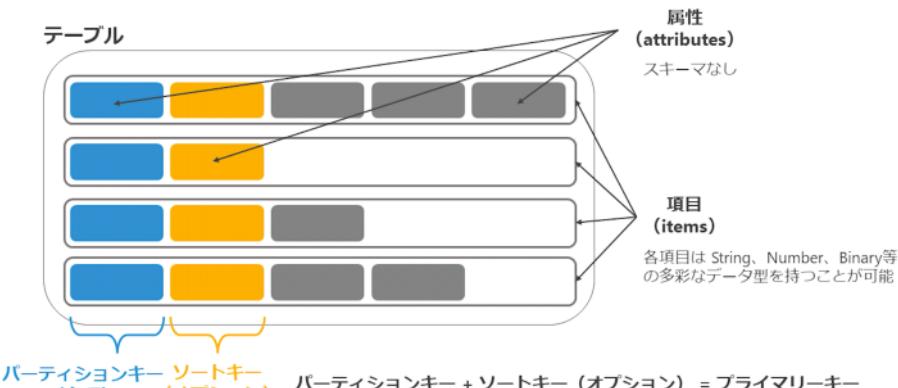
```
#redis-cli
```

>set [key] [value]	→ keyに対するvalueを設定
>get [key]	→ keyに設定されているvalueを取得
>del [key]	→ keyの削除

・リスト型のデータ操作

>push [key] [value]	→ key(リスト)を作成し、valueを設定 ※2回以上実行すると、値が追加されていく
>range [key] [start] [end]	→ key(リスト)を参照。startからendまでに登録された値を取得

テーブル構造



パーティションキー ソートキー (必須) パーティションキー + ソートキー (オプション) = プライマリーキー

一つのデータのことを項目といい、複数の属性をもつ

一つのテーブルに定義できる項目の数に制限なし

スキーマがないため項目ごとに異なる属性を持てる

テーブルの項目をパーティションに保持

項目がどの項目に保持されるかはパーティションキーで決定

ソートキーで範囲を指定した検索が可能

2つのデータの持ち方

- ・パーティションキーのみ
- ・パーティションキー+ソートキー

インデックスの特徴

- ▶ ローカルセカンダリインデックス (LSI)
 - ▶ ソートキー以外のキーとして検索するために利用
 - 例) ある 顧客ID の中から 値格 を指定して抽出
- ▶ グローバルセカンダリインデックス (GSI)
 - ▶ 異なるパーティションキーをまたいで検索するために利用
 - 例) 全データの中から 書籍名 を指定して抽出

プライマリキー

パーティションキー	ソートキー	属性	属性
顧客ID	注文番号	書籍名	値格

LSI

パーティションキー	ソートキー	属性	属性
顧客ID	注文番号	書籍名	値格

GSI

パーティションキー	ソートキー	属性	属性
顧客ID	注文番号	書籍名	値格

LSI : ソートキー以外での絞り込みができる

GSI : 異なるパーティションキーをまたいで検索できる

料金

- ▶ Amazon DynamoDB
 - ▶ キャパシティユニット
 - ▶ 1秒間に決められた回数の読み書きができる容量のこと
 - ▶ プロビジョニング済キャパシティ : 設定したユニット数に応じて課金
 - 書き込みキャパシティユニット (WCU) 1時間あたり 0.000742 USD/WCU
 - 読み込みキャパシティユニット (RCU) 1時間あたり 0.0001484 USD/RCU
 - 無料利用枠 : 25ユニットの書き込み/読み込み キャパシティ
例) WCU/RCU が 5 の場合、1秒間に 1KB以下の書き込み処理を 5回 実行できる
 - ▶ オンデマンドキャパシティ : 実際に消費されたユニット数に応じて課金
 - 書き込み要求単位 (WRU) 100万リクエストあたり 1.4269 USD
 - 読み出し要求単位 (RRU) 100万リクエストあたり 0.285 USD

※2022年3月時点の東京リージョンの価格例

AWS Lambda

2023年7月26日 13:43

サーバーのプロビジョニングや管理なしでプログラムを実行できるサービス



イベントドリブンなタスクコンピューティング



- ① S3 バケットに画像を保管する
- ② S3 バケットに画像が保管されたことをトリガーに Lambda が実行される
- ③ Lambda で画像のサイズを変更する
- ④ サイズを変更した画像を S3 バケットに保管する

Lambda関数の基本設定

メモリ

128MBから10240MBの間で1MB単位で調整可能

要領に応じてCPU能力なども比例

メモリ容量が一定を超えると使用するコア数も増える

タイムアウト

Lambda関数の実行時間に関するタイムアウト

最大900秒(15分)まで ←15分以上かかる処理の場合は分割・並列化 それでも難しい場合はEC2で実装する

実行ロール

必要なAWSリソースへのアクセスを許可するIAMロール

指定されたIAMロールに沿ってLambda関数からAWSリソースへのアクセスが許可

レイヤーを追加

AWSが用意している環境ではデフォルトでは使用できないモジュールがあるのでそれを使用できるようになる設定

サポートされている言語(2022/4/7)

Python 3.6、3.7、3.8、3.9

Node.js 12、14

.NET Core 3.1 (C#/PowerShell 7)、6 (C#/PowerShell 7.2)

Go 1.x

Java 8、11

Ruby 2.7

Lambda関数の構造

Lambdaで実行したい処理は、既定の書式に則った関数として記述する必要がある

関数の引数

関数の返り値の書式

関数のエラーの返し方

Lambda関数（ハンドラ）の書式

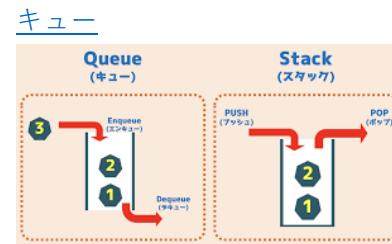
イベント：呼び出すときに渡された入力値

コンテキスト：実行環境の情報

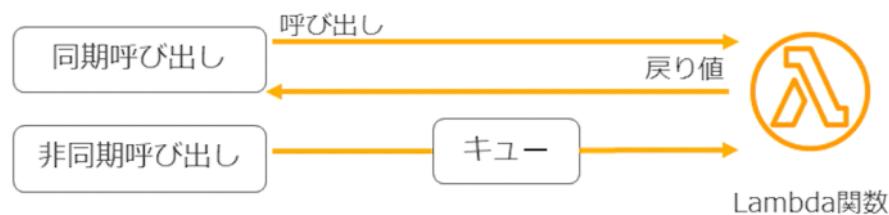
Lambda関数のイベントソース

Lambda関数のトリガーとなるイベント

※初回起動は環境を準備するので時間がかかる



プッシュモデル



同期

◆ 同期呼び出しの例



Lambda の実行時にレスポンスが返却

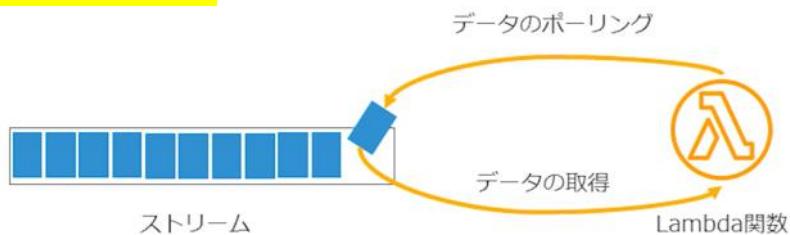
非同期

◆ 非同期呼び出しの例



Lambdaへのリクエストが正常に受け付けられたかどうかのみを返却

ストリームベース



Lambda 関数のイベントソース ※一部

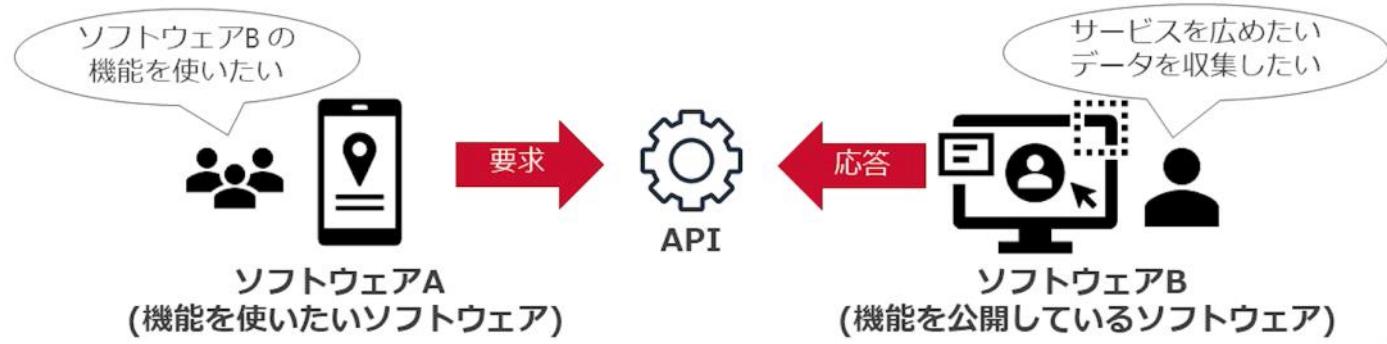
イベントソース	呼び出しタイプ	概要
Amazon S3	非同期	S3バケットへの書き込みや削除が発生したとき
Amazon DynamoDB	ストリームベース	DynamoDBのデータに対して操作したとき
Amazon Kinesis Data Streams	ストリームベース	Kinesisの新しいレコードが到着したとき
Amazon Simple Notification Service	非同期	SNSトピックに対してメッセージが到來したとき
Amazon Simple Email Service	非同期	メールを受信したとき
Amazon Simple Queue Service	同期	キューからメッセージを取得したとき
Amazon Cognito	同期	Cognitoイベントが発生したとき
Amazon API Gateway	同期	HTTPS経由でLambda関数を呼び出したいとき

API

2023年7月26日 14:25

ソフトウェアの機能を共有する仕組み

「機能を公開しているソフトウェア」と
「その機能を使いたいソフトウェア」をつなげる窓口のようなもの



0から機能を開発しなくとも既に公開されているものを使用することができる

Web API

▶ Web API の例

- ▶ 飲食店のホームページを作るので、店の位置情報を表示したい
- ▶ 位置情報を表示する機能が Web API として公開されていれば、Web API 経由で位置情報表示機能を利用できる



- ▶ Web API を公開するには
 - ▶ リクエストを受け付けレスポンスを返すためのインフラ設計が必要
 - ▶ 多くのリクエストを処理するためのスケーラビリティ
 - ▶ API を使用するための認証と認可の仕組み



API開発で考えること

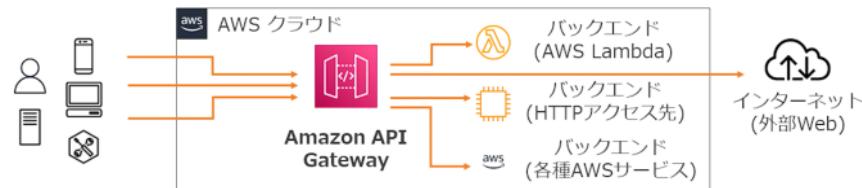
- ・ Web API機能の開発
- ・ 高可用性(いつでも利用できるようにする)/スケーラビリティ設計(急激に増えるアクセス負荷に耐える)
- ・ インフラの管理
- ・ APIキーの管理
- ・ デプロイ管理(APIの拡張や改善)

Amazon API Gateway

2023年7月26日 14:42

サーバーをプロビジョニング/管理することなく、APIを作成・管理できるマネージドサービス

リソースが来た時にどのような処理をするか設定する (Lambda関数など)



- API開発で必要なこと（可用性の担保、スケーラビリティ、API管理など）を任せられるので、開発者はビジネスの差別化につながる作業に集中できる

- REST APIとWebSocketに対応

▶ REST API と WebSocket に対応



REST

2023年7月26日 14:58

ウェブサービスを開始する時のソフトウェアアーキテクチャの設計方針

REST APIの4つの原則

- ・アドレス可用性

URIでリソースを表現

Web上の情報をURIで識別できるように

ユーザー情報を扱うAPIの例

<https://api.exammple.com/user>

userの部分がリソース名

- ・統一インターフェース（あらかじめ定義・共有されたやり方、方法でやりとりできる）

HTTPメソッド（webブラウザからwebサーバーに出されるリクエストの種類）

GET：リソースの取得

POST：リソースを追加

PUT：リソースを更新

DELETE：リソースを削除

REST API 例 ○

<https://api.example.com/user>

▶ GET : ユーザー情報を取得

▶ POST : ユーザー情報を追加

▶ PUT : ユーザー情報を更新

▶ DELETE : ユーザー情報を削除

▶ REST ではない Web API 例 ✗

▶ <https://api.example.com/getuser> : ユーザー情報を取得

▶ <https://api.example.com/createuser> : ユーザー情報を追加

▶ <https://api.example.com/updateuser> : ユーザー情報を更新

▶ <https://api.example.com/deleteuser> : ユーザー情報を削除

※URIで処理を分けているのでRESTではない

- ・接続性

情報の内部に別の情報やその情報へのリンクを含めることができる

次状態へ遷移するためのリンクをもつこと

- ・ステートレス性

サーバー側が利用者とのセッションを管理しない

ステートフルな会話の例

質問者 天気はどうですか。

回答者 いつの天気ですか。

質問者 明日です。

回答者 どこの天気ですか。

質問者 東京です。

回答者 晴れです。

▶ ステートレスな会話の例

質問者 天気はどうですか。

回答者 いつの天気ですか。

質問者 明日の天気です。

回答者 どこの天気ですか。

質問者 明日の東京の天気です。

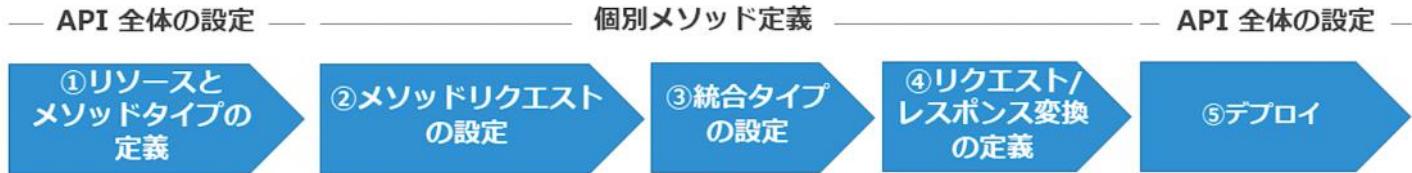
回答者 晴れです。

3種類のエンドポイントタイプから1つを選択



開発の流れ

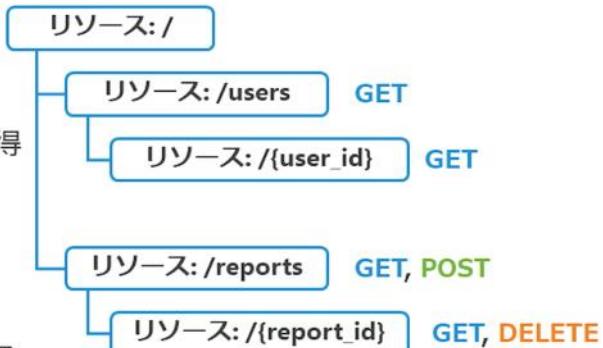
2023年7月26日 15:16



①

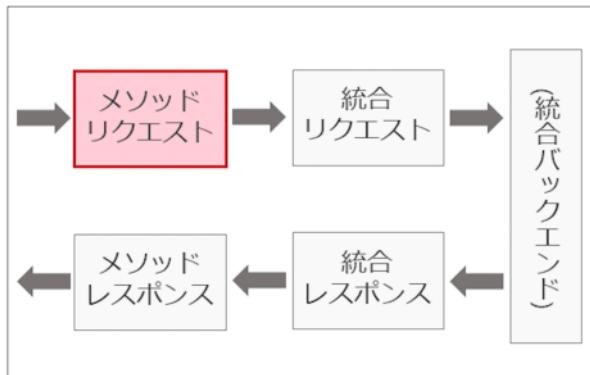
API の例

- ・ ユーザー一覧を取得
- ・ ユーザーIDを指定して、ユーザー情報を取得
- ・ レポートの一覧を取得
- ・ レポートを投稿する
- ・ レポートIDを指定して、レポートを削除する



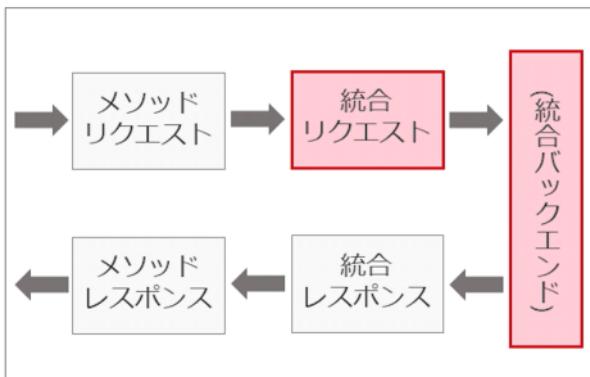
②

- ▶ リクエストの受付に関する設定を行なう
 - ▶ 認証の設定
 - ▶ 受け付けるクエリパラメータ
 - ▶ 必須とするHTTPヘッダの設定



③

- ▶ バックエンドの種別を設定する
 - ▶ Lambda関数
 - ▶ HTTP
 - ▶ Mock
 - ▶ AWS サービス
 - ▶ VPC リンク



④

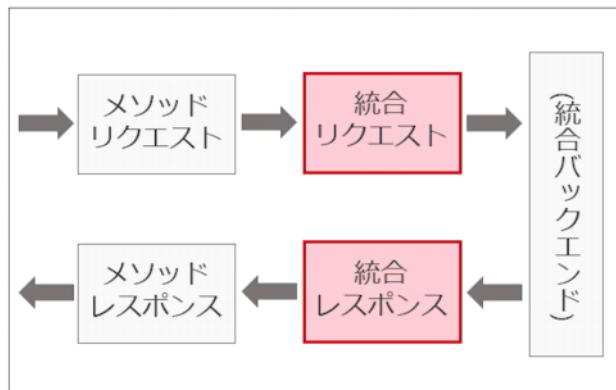
バックエンドへのインプット、
バックエンドからのアウトプット
変換をすることができる

プロキシ統合を利用することで、
変換せずにパススルーすることも
可能

⑤

APIごとにデプロイ

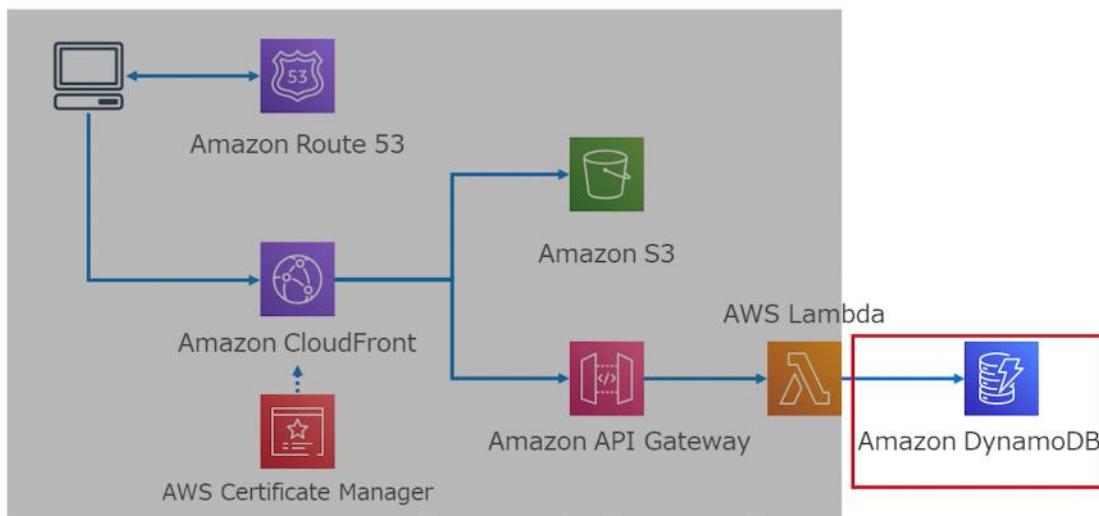
ステージを作成し、そのステージのみにデプロイ可能



テーブル作成

2023年7月26日 15:39

今回作成する構成の確認



シーケンス番号（連番で一意に定まる番号）でプライマリーキーを管理する
→採番用のテーブルを作ってそこから発番する方法

テーブル名：news

属性	型	プライマリーキー	説明
post_id	String	<input type="radio"/> (Hash Key)	記事ID
subject	String		件名
contents	String		本文

手順

「Dynamo DB」 → 「テーブル」

newsテーブル作成

テーブル名：news

※テーブル名：osuke-oyaizu-news

パーティションキーナメ：post_id

sequencesテーブル作成

テーブル名：sequences

※テーブル名：osuke-oyaizu-sequences

パーティションキーナメ：name

「sequences1テーブル」 → 「アクション」

項目を作成

項目の属性を追加、削除、または編集できます。属性は、最大 32 レベルの深さまで他の属性内にネストできます。詳細は[こちら](#)

フォーム

JSON ビュー

属性

新しい属性の追加 ▾

属性名

値

タイプ

name - パーティションキー

news

文字列

current_number

0

数値

削除

キャンセル

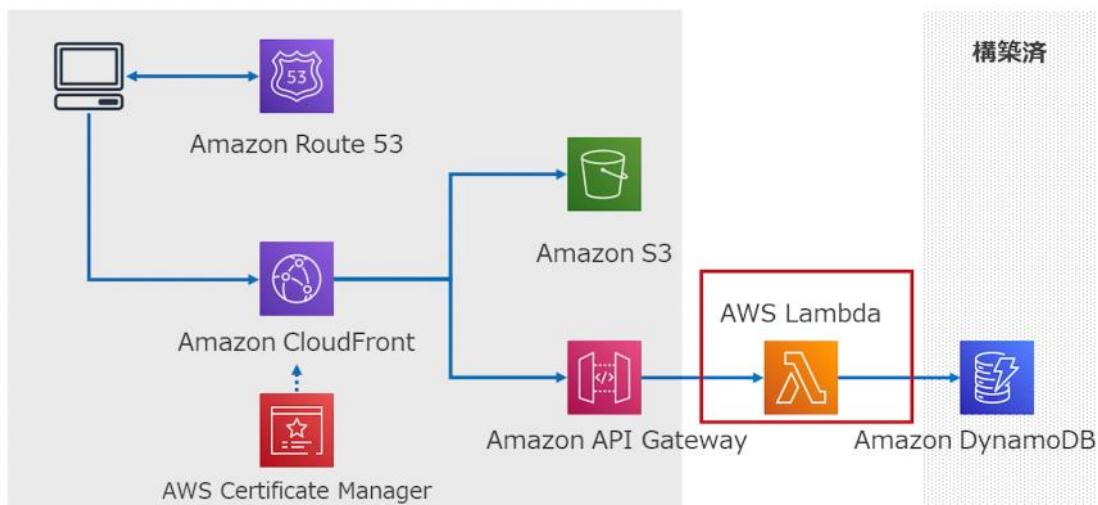
項目を作成

「項目を探索」で確認できる

Lambda関数作成

2023年7月26日 15:54

今回作成する構成の確認



①IAMロールの作成

ステップ2: 許可を追加する

許可ポリシーの概要	
ポリシー名	タイプ
AmazonDynamoDBFullAccess	AWS 管理
AWSLambdaBasicExecutionRole	AWS 管理

②Lambda関数作成

「Lambda」→「関数を作成」

ランタイム : Python

実行ロール : ①で作成したロール

③Lambda関数の設定

「関数」→コードタブ→内容を書き換える(python)→Deploy→設定タブ

→環境変数を設定する(キー : TableName 値 : news) ←ソースコードのTableNameに値が入る

※値 : osuke-oyaizu-news



section15-l
ambda_fu...

- ・レイヤー追加 ←デフォルトで準備されているものだと足りないモジュールがある

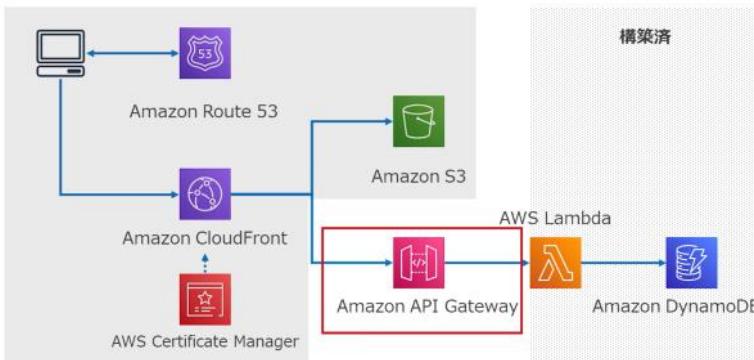
コードタブから→レイヤーを追加→ARNを指定→

arn:aws:lambda:ap-northeast-1:770693421928:layer:Klayers-p39-pillow:1

API Gateway作成

2023年7月26日 16:24

今回作成する構成の確認



①Amazon API Gatewayの作成

「API Gateway」 → 「REST API」

②リソースの作成

「アクション」 → 「リソースの作成」

※リソース名：osuke-oyaizu-news

リソース アクション ▾

新しい子リソース

このページを使用して、リソースの新しい子リソースを作成します。

新しいウィンドウにプロキシリソースが開きます

リソース名*

news

リソースパス*

/ news

角括弧を使用してパスパラメータを追加できます。たとえば、リソースパス {username} は、"username" という名前のパスパラメータを表します。プロキシリソースとして {proxy+} を設定すると、そのサブリソースへのすべてのリクエストがキャッチされます。これは例えば、/foo への GET リクエストとして機能します。/ へのリクエストを処理するには、/ リソースで新しい ANY メソッドを追加します。

API Gateway CORS を有効にする

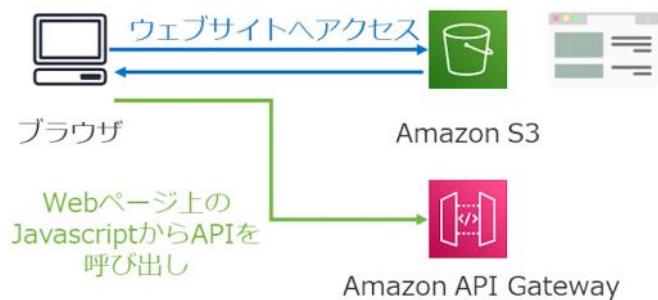
* 必須

キャンセル リソースの作成

※CORSとは

CORS : Cross-Origin Resource Sharing の略

ブラウザがオリジン以外のサーバーからデータを取得する仕組み。
この設定が有効でないAPIは、ドメインが異なるWebページ上の
Javascript から呼び出すことができない。



③リソースの設定

GETメソッド

統合タイプ : Lambda関数

Lambdaプロキシ統合の使用にチェック

Lambda関数 : 作成したもの

「テスト」で200番が返ってくることを確認する

※エラー原因

- ・関数の環境変数設定で正しいテーブル名を選択する
- ・sequencesテーブルの名前を変えた場合はソースコードを変更

POSTメソッド

同じ

④APIのデプロイ

「アクション」 → 「APIのデプロイ」

API のデプロイ

API がデプロイされるステージを選択します。たとえば、API のテスト版をベータという名前のステージにデプロイできます。

デプロイされるステージ	[新しいステージ]
ステージ名*	dev
ステージの説明	
デプロイメントの説明	

キャンセル デプロイ

dev ステージエディター

ステージの削除 タグの設定

URL の呼び出し: <https://7vq894m7rf.execute-api.ap-northeast-1.amazonaws.com/dev>

/osuke-oyaziu-news?post_id=0を追加

https://7vq894m7rf.execute-api.ap-northeast-1.amazonaws.com/dev/osuke-oyaziu-news?post_id=0

Amazon CloudFrontの設定変更

2023年7月27日 11:26

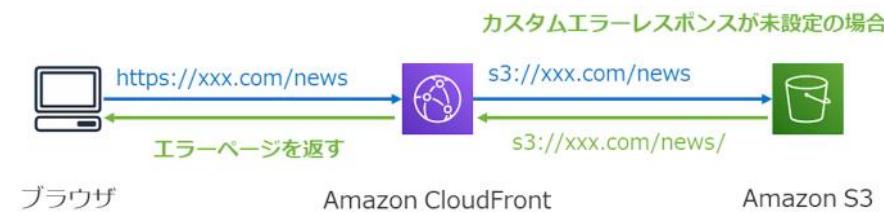
①オリジンの追加



②ビヘイビアの追加

パスパターン: dev/*
オリジンとオリジングループ: API Gatewayのオリジン
ビューワープロトコルポリシー: HTTPS only
キャッシング: クエリ文字列→すべて カスタムキャッシング→すべて0

③カスタムエラーレスポンス



HTTPエラーコード: 403
キャッシング: 0
エラーレスポンスをカスタマイズ: はい
レスポンスページのパス: /index.html
HTTPレスポンスコード: 200

アプリの再デプロイ

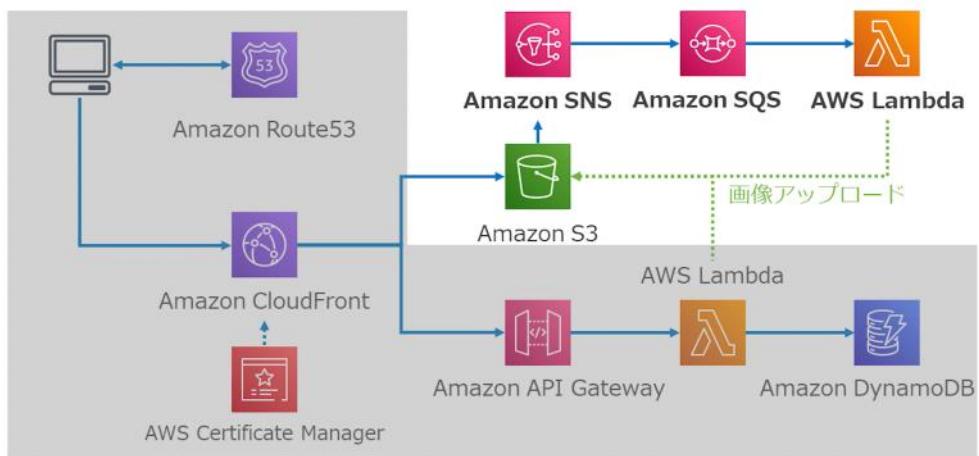
2023年7月27日 11:46



セクション16

2023年7月27日 15:30

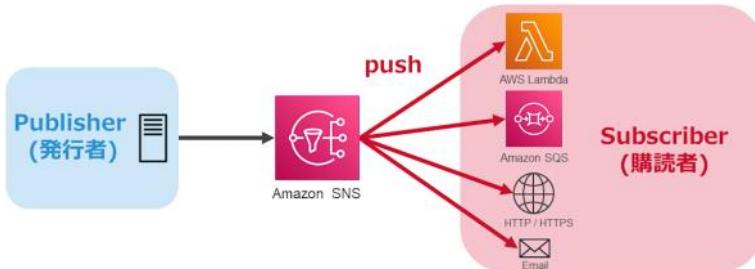
作成する構成



Amazon SNS

2023年7月27日 15:32

- ▶ フルマネージドの pub / sub メッセージ配信サービス



送信側は受信側の処理の完了を待たずに送信できる（非同期通信が可能）

受け取ったものを適切な場所に送る

Publisherが送ったものをその時に接続されているものにのみおくる

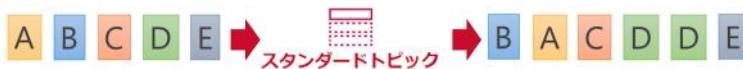
- ▶ 対応している Subscriber (例)

送信先	説明
Eメール	テキストが E メールとして、登録されたアドレスに送信される
SMS (モバイルテキストメッセージ)	SMS テキストメッセージとして、登録された電話番号に送信される
モバイルプッシュ通知	モバイルデバイスのアプリにプッシュ通知する
HTTP / HTTPS	購読登録時に URL を指定 HTTP POST を通じて通知が指定された URL に通知が届く
Amazon SQS	指定されたキューにメッセージを投入する
AWS Lambda	AWS Lambda 関数にメッセージを配信する
Amazon Kinesis Data Firehose	追加のストレージおよび分析エンドポイントに通知を送信する

スタンダードおよび FIFO トピック

- ▶ スタンダードトピック

- ▶ 配信順序はベストエフォート、少なくとも一回配信



- ▶ FIFO トピック

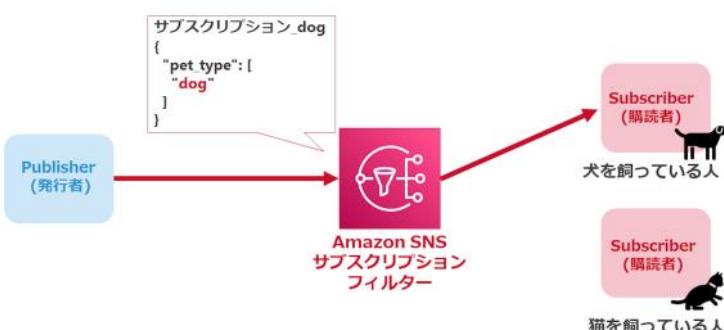
- ▶ 配信順序を厳守、1回のみ配信
- ▶ サブスクリプションとして Amazon SQS のみをサポート



できるだけ配信順序通り配信するが保証はできない
ほぼ無制限のスループットを提供→大量のリクエストを処理できる

高性能だがスループットに制限
高額

メッセージのフィルター処理



アクセスポリシー

▶ リソースベースのアクセス権限システム

▶ SNS のアクセスポリシー

アイデンティティベースの IAM ポリシー	SNS のアクセスポリシー
<ul style="list-style-type: none">IAM ユーザー、IAM グループ、IAM ロールが AWS リソースにアクセスできるかどうかを制限する自身の AWS アカウント内の IAM ユーザーのみに対してアクセスを制限する	<ul style="list-style-type: none">他の AWS アカウントに対して、特定のトピックアクション(発行など)を許可する通知配信プロトコルを HTTPS などに制限する他の AWS サービスが SNS にメッセージを発行することを許可する



Amazon SQS

2023年7月27日 16:49

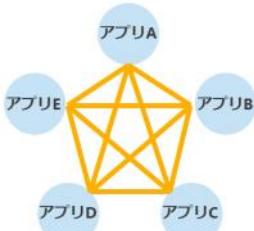
https://qiita.com/miyuki_samitani/items/e46ef9452fcfd73f9d240

ソフトウェア間を直接データを渡すのではなく、第三者経由でデータを渡すことで、送信側と受信側が好きなときに処理を行うことができます。

疎結合にするために設定 プログラム間の橋渡し

疎結合アーキテクチャ

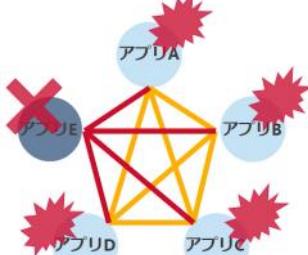
密結合アーキテクチャ



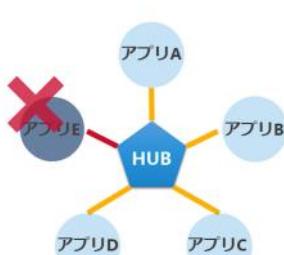
疎結合アーキテクチャ



密結合アーキテクチャ



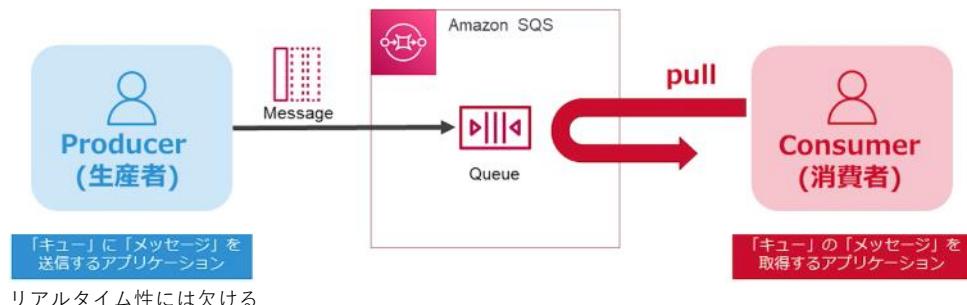
疎結合アーキテクチャ



密結合アーキテクチャ：コンポーネント間の結びつき、依存関係、関連性が強い 各々の独立性が低い
疎結合アーキテクチャ：密結合の逆

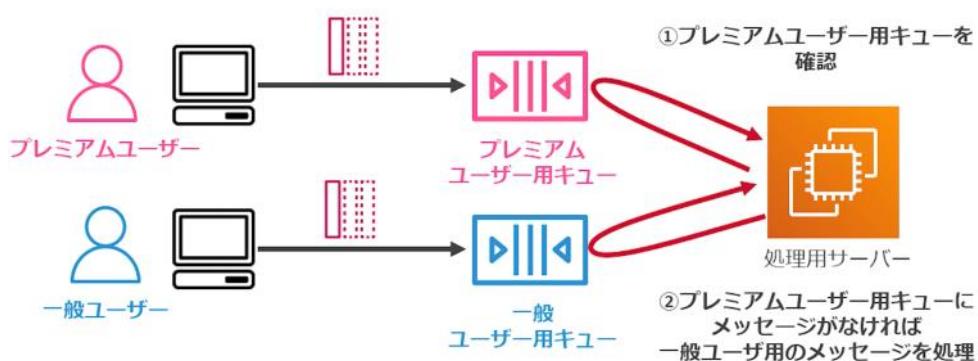
Amazon SQS

フルマネージドのメッセージキューイングサービス



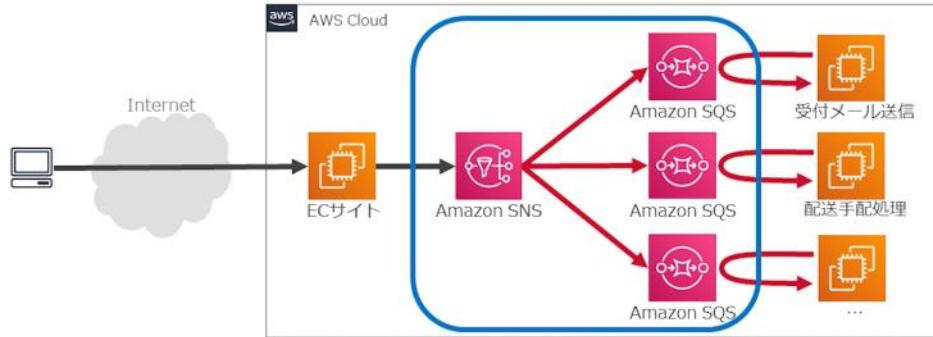
▶ ユースケース

- ▶ 処理の優先順位に応じて個別のキューを使用



▶ ファンアウト

- ▶ メッセージが複数の受信者にプッシュされる
- ▶ メッセージの並列/非同期処理が可能



ショートポーリングとロングポーリング

▶ ショートポーリング

- ▶ 即応答、キューが空なら「空」と応答
- ▶ 繰り返しポーリングするため
- API コール数が増え利用料金が増加

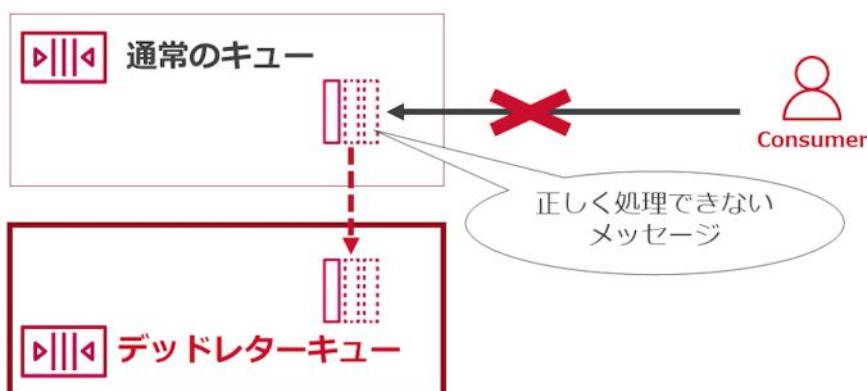
▶ ロングポーリング

- ▶ 最大 20 秒メッセージ受領を待つ
- ▶ API コール数を抑制できるため
- 利用料金を節約できる



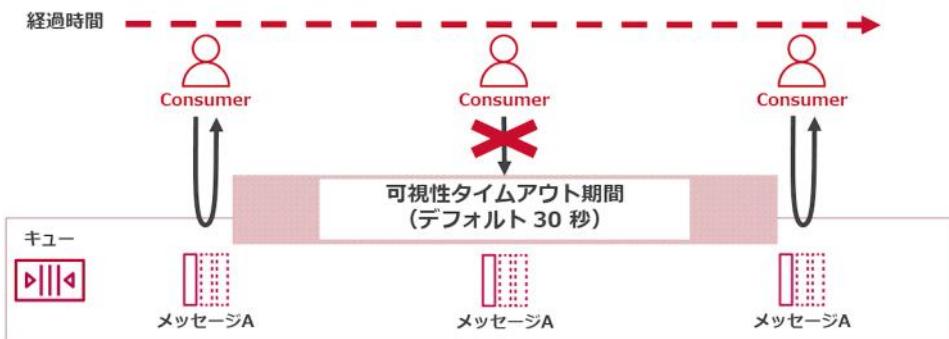
デッドレターキュー

- ▶ 配信失敗したメッセージをデッドレターキューに保存できる
- ▶ デッドレターキューのメッセージを解析し原因の分析に活用



可視性タイムアウト

- 同じメッセージが複数の Consumer に配信され重複処理が発生するのを防ぐ



アクセスポリシー

- リソースベースのアクセス権限システム
 - SQS のアクセスポリシー

アイデンティティベースの IAM ポリシー	SQS のアクセスポリシー
<ul style="list-style-type: none">特定の IAM ユーザーや IAM ロールに対してアクセス制御をおこなう場合に用いる	<ul style="list-style-type: none">特定のキューに対するアクセス制御をおこなう場合に用いる



SQS概要

2023年7月27日 17:06

Amazon SQS：フルマネージドのメッセージキューイングサービス

メリット：ProducerとConsumerを完全に切り離して疎結合アーキテクチャを実現

SQSを挟むことでお互い同期をとる必要がなくなる

最大14日間キューに残る

Producerが大量のメッセージを送信→SQSに貯めておく

ファンアウト：SNSに発行されたメッセージが複数のSQSにpushすることで様々な処理を同時に実行（並列処理）

キューからメッセージを受信するためにショートポーリングとロングポーリング

ショートポーリング：即応答、キューが空なら「空」と応答

ロングポーリング：最大20秒メッセージ受領を待本

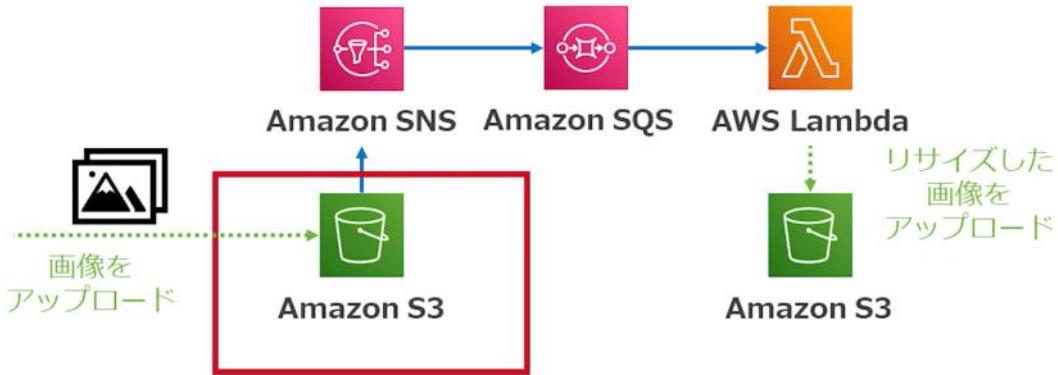
デッドレターキュー：配信失敗したメッセージを保存できる→メッセージの滞留を防ぐ、配信を再試行、解析して原因を解明

可視性タイムアウト：重複処理を防ぐ

手順

2023年7月27日 17:21

今回作成する構成の確認



①画像格納用バケット作成

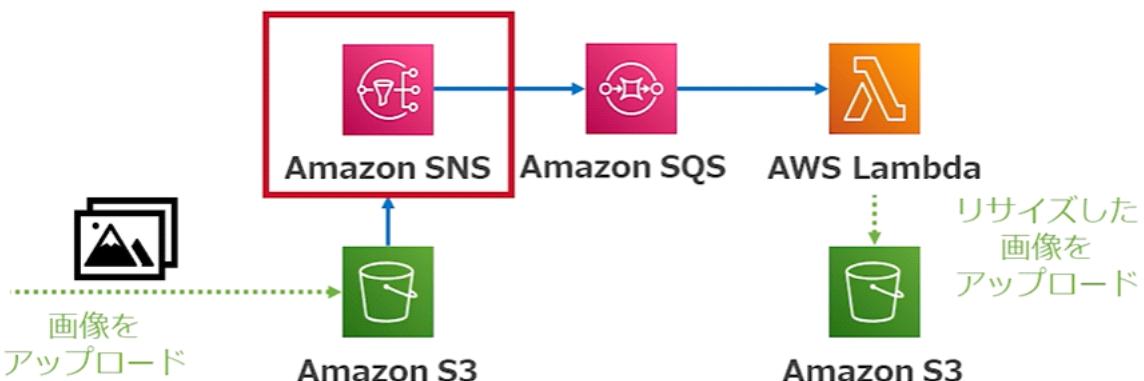
- ・S3からバケット作成
- ・パブリックアクセスのチェックは外さない

※バケットを分けないとループしてしまうので

SNS作成

2023年7月27日 17:41

今回作成する構成の確認



①ロール作成

AmazonDynamoDBFullAccess
AWSLambdaBasicExecutionRole
AWSLambdaSQSQueueExecutionRole
AmazonS3FullAccess

②Lambda関数作成

- ・リージョン：東京
- ・ランタイム：Python3.9
- ・既存のロール：①

③Lambda関数の設定

- ・ソースコード入力→Deploy

- ・環境変数

キー：TableName 値：osuke-oyaizu-news

キー：BucketName 値：サイトをホストしているバケット名

- ・一般設定

タイムアウト：1分 ※処理が1分以上かかる場合は実行されない

- ・レイヤー追加 ←デフォルトで準備されているものだと足りないモジュールがある

コードタブから→レイヤーを追加→ARNを指定→

arn:aws:lambda:ap-northeast-1:770693421928:layer:Klayers-p39-pillow:1



section16-l
ambda_fu...



section16-l
ambda_fu...

④既存のLambda関数の修正

- ・ソースコード編集→Deploy
- ・環境変数

キー：TableName 値：osuke-oyaizu-news

キー：BucketName 値：画像格納用バケット名



section15-l
ambda_fu...

①画像格納用バケットのARNをコピー

②「SNS」→「トピック作成」→アクセスポリシーを以下に変更

```
{  
    "Version": "2008-10-17",  
    "Id": "__default_policy_ID",  
    "Statement": [  
        {  
            "Sid": "__default_statement_ID",  
            "Effect": "Allow",  
            "Principal": {  
                "Service": "s3.amazonaws.com"  
            },  
            "Action": "SNS:Publish",  
            "Resource": "arn:aws:sns:ap-northeast-1:608728620263:osuke-oyaizu-sns-from-s3",  
            "Condition": {  
                "StringEquals": {  
                    "aws:SourceArn": "arn:aws:s3:::training-dev-image-storage-lab2-20230727",  
                    "AWS:SourceAccount": "608728620263"  
                }  
            }  
        }  
    ]  
}
```

②S3バケットのイベント通知設定

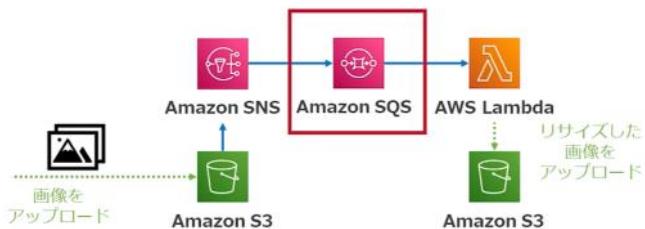
- ・画像格納用バケット→プロパティ→イベント通知を作成
 - ・イベントタイプ：PUT

- ・送信先：SNSトピック←②で作成したもの

SQS作成

2023年7月27日 17:58

今回作成する構成の確認



①SQSキュー作成

「SQS」 → 「キューを作成」

キューのタイプ：標準

可視性タイムアウト：60秒

②SQSキューの設定

SNSサブスクリプション

トピックを選択：①を選択

③確認

サブスクリプション選択→SNSで表示

→トピック名選択→メッセージの発行

→sqSキューのさらに表示→メッセージを表示

→メッセージをポーリング→見える

④SQSキューをトリガーにLambda関数を実行

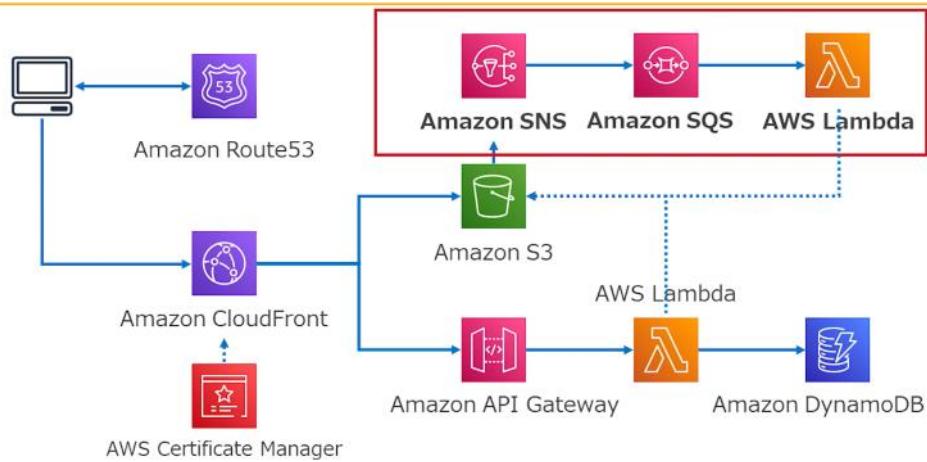
Lambdaトリガー→画像サイズ変更用関数を選択する

←①でコピーしたもの

振り返り

2023年7月28日 9:41

構成の確認



画像入力欄

画像をS3バケットへ

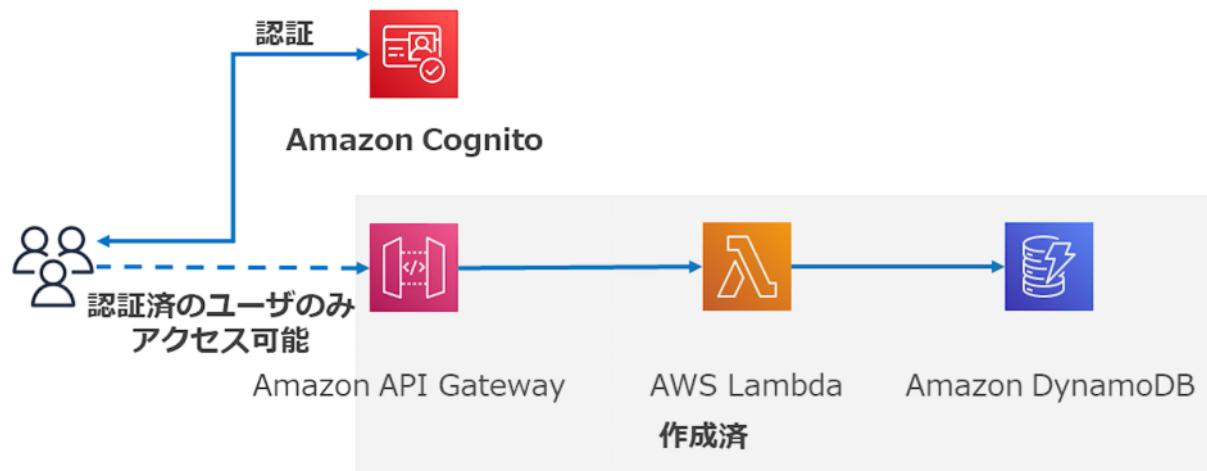
登録した画像のサイズ変更

サムネイルに使えるように

S3に画像がアップロードされたことをトリガーにLambda関数を実行

セクション17

2023年7月28日 9:44



Cognito

2023年7月28日 9:56

Amazon Cognito：不特定多数の一般ユーザーの認証・認可をするサービス

API直アクセス対策

認証

「誰」であるかを確認・特定すること

Amazon Cognitoの機能では、ユーザープールが相当

認可

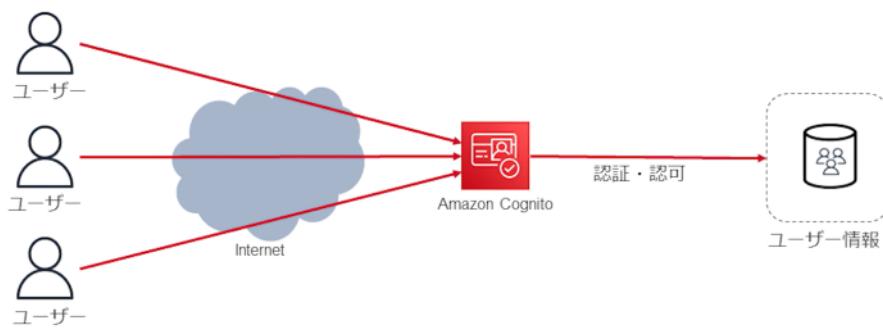
上記認証対象に、権限を与えること

Amazon Cognitoの機能では、IDプールが相当

もし失敗した場合は認証画面に遷移するのが一般的

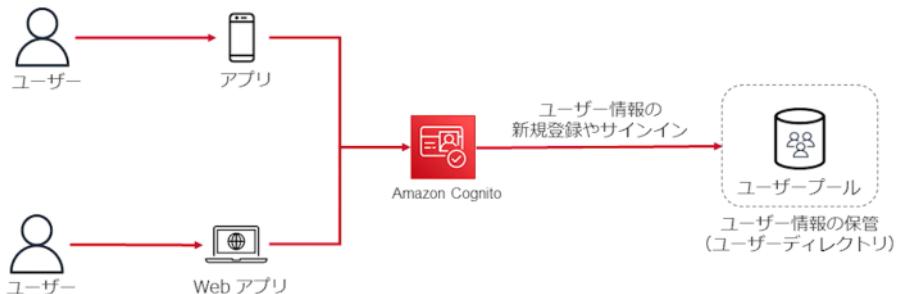
Amazon Cognito

- ▶ モバイルアプリや Web アプリなどで、不特定多数の一般ユーザーの認証・認可を行なうサービス
- ▶ 何百万人ものユーザー数の管理



ユーザープール

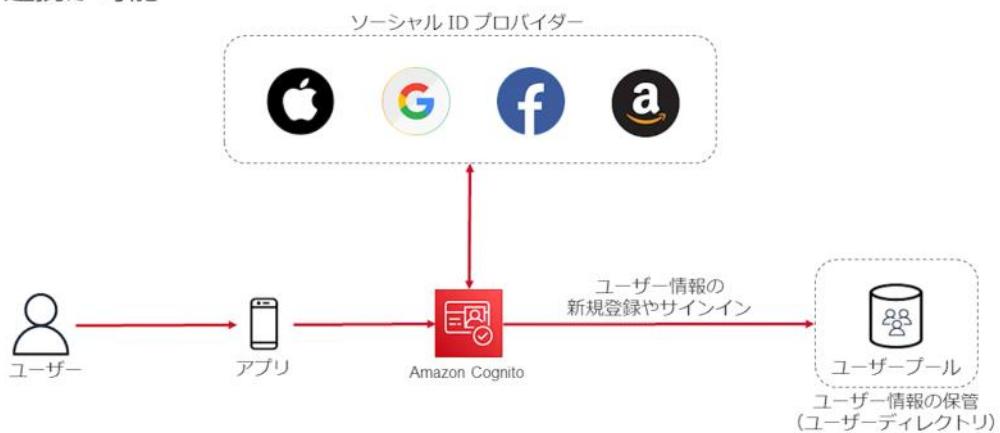
- ▶ 数百万人規模の管理が可能なユーザー ディレクトリで認証を行なう
- ▶ モバイルアプリや Web アプリへの、ユーザーの新規登録とサインインの管理を行なう事ができる



1 AWSアカウント毎に準備できるユーザー登録情報の上限数

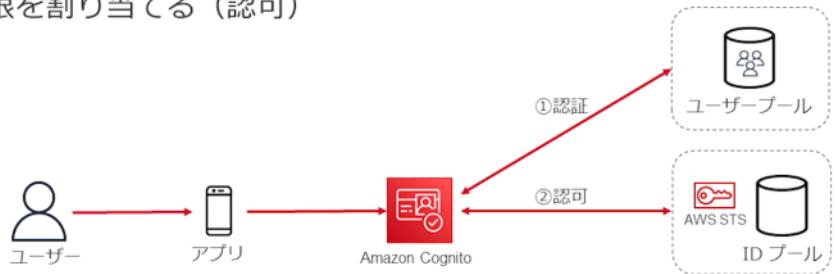
項目	数量
ユーザープール数	1,000ユーザープール (最大上限値は10,000ユーザープール)
ユーザープールに登録できるユーザー数	1ユーザープールあたり標準40,000,000人 (上限申請は別途問い合わせが必要)

- ▶ Apple、Google、Facebook、Amazonなどのソーシャル ID プロバイダーとの連携が可能



ID プール

- ▶ ユーザーに一意の ID を作成して、他の AWS サービスへのアクセス許可の権限を割り当てる（認可）



IAMとの違い

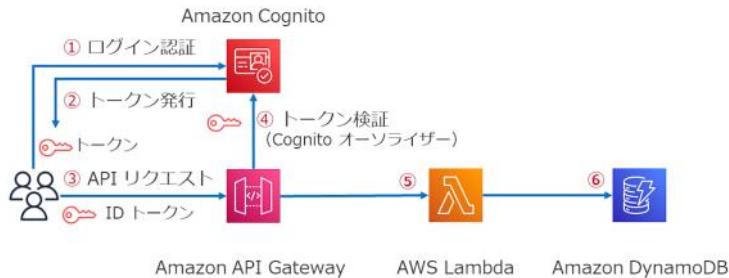
IAM : AWSのサービスへの認証・認可

Cognito : アプリケーションへの認証・認可

構築手順

2023年7月28日 10:07

今回作成する構成の確認



①ユーザー プールを作成

「cognito」 → 「ユーザー プール作成」

サインインオプション：ユーザー名

MAFなし（多要素認証なし）

サインアップエクスペリエンス設定：デフォルト

Eメールプロバイダ：CognitoでEメール

ホストされた認証ページ：CognitoのホストされたUIを使用

ドメイン：一意のドメイン

アプリケーションクライアント名：任意の名前

許可されているコールバックURL：CloudFrontのURL/news/new

高度なアプリケーションクライアントの設定（認証時にトークンを使用する設定）：ALLOW_ADMIN_USER_AUTH ←管理者権限で取得する
OAuth2.0ユーザー プール：暗黙的な付与

②ユーザー作成

①→ユーザー タブ

ユーザー名：各自の

メールアドレス：各自の

仮パスワード：各自の

アプリケーションの統合タブ→ホストされたUIを表示

サインインする

③オーソライザーの作成

API Gateway → API選択→オーソライザー

新しいオーソライザーの作成

タイプ：Cognito

Cognitoユーザー プール：①

トークンのソース：Authorization

テスト

②でアクセスした際のid_token=の後ろから&access_tokenの手前までコピーして貼り付け

→200ならOK

リソース→POST→メソッドリクエスト

認可：つくったオーソライザー

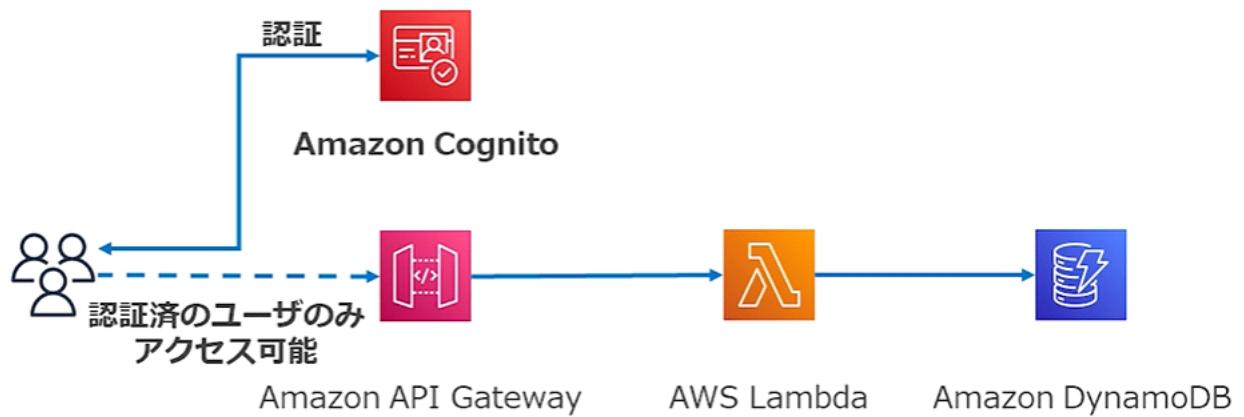
アクション→APIのデプロイ

デプロイされるステージ：dev

振り返り

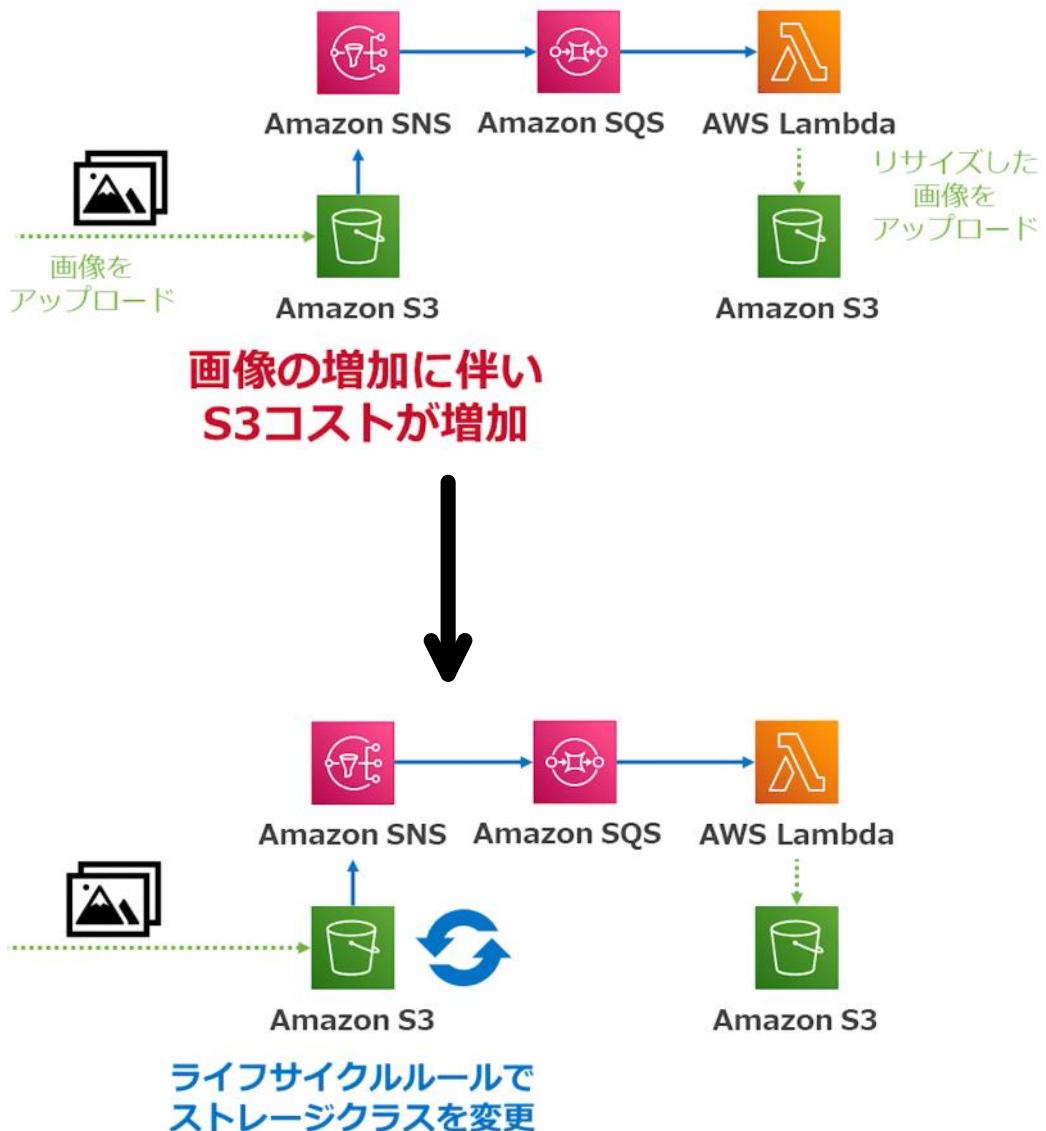
2023年7月28日 10:56

構成の確認



セクション18

2023年7月28日 13:02



Amazon S3 ストレージクラス

2023年7月28日 14:05

S3ストレージクラス

データのアクセス頻度、復元力、コスト要件に基づいて最適なクラスを選択できる

Amazon S3 ストレージクラスの概要

ストレージクラス	すぐに取り出せるか？	ストレージ料金	取り出し料金	特徴
Amazon S3 Standard (標準)	○	高	無料	デフォルトのストレージクラス。 アクセス頻度が高いデータに最適。
Amazon S3 Standard - IA (標準 - 低頻度アクセス)	○	中	低	S3 標準に比べて格納コストが安価。 データの取り出しに対してコストがかかる。
Amazon S3 One Zone - IA (1 ゾーン - 低頻度アクセス)	○	中	中	シングル AZ にデータを格納するため、AZ 障害が発生するとデータが失われる。
Amazon S3 Glacier Instant Retrieval	○	中	中	アクセスがほとんどないデータを保管。 すぐにデータへアクセス可能。
Amazon S3 Glacier Flexible Retrieval	△取り出し オプションを選択	中	中	データの取り出しにコストと時間がかかる。 データ取り出しオプションが選べる。
Amazon S3 Glacier Deep Archive	×	低	高	アクセスがほぼないデータを保管。 データの取り出しにコストと時間がかかる。
Amazon S3 Intelligent-Tiering	○アーカイブ だと時間がかかる場合がある	自動最適化	一部を 除き無料	アクセス頻度によって、データを最も費用 対効果の高いアクセス階層に自動的に移動 し、ストレージコストを最適化する。

Amazon S3 Standard (標準)

アクセス頻度	頻繁にアクセス
取り出し速度	すぐに取り出せる
ストレージ料金	通常料金
取り出し料金	不要
特徴	・デフォルトのストレージクラス ・幅広いケースに適している

Amazon S3 Standard - IA (標準-低頻度アクセス)

アクセス頻度	低頻度
取り出し速度	すぐに取り出せる
ストレージ料金	S3 Standardよりも低価格
取り出し料金	少しかかる
特徴	・アクセス頻度の低いデータに適している ・長期保存、バックアップ、災害対策ファイルのデータストアなど

S3 One Zone - IA (1ゾーン-低頻度アクセス)

アクセス頻度	低頻度
取り出し速度	すぐに取り出せる
ストレージ料金	S3 Standard-IAよりも低価格
取り出し料金	少しかかる
特徴	・ひとつのAZにデータが保存されるので、 AZが破壊されると保存データは失われる ・重要度が高くないデータや容易に再作成可能なデータなどに適している

S3 Glacier Instant Retrieval

アクセス頻度	ほとんどアクセスしない
取り出し速度	すぐに取り出せる
ストレージ料金	S3 One Zone - IAよりも低価格

取り出し料金	S3 One Zone - IAよりも高い データ取り出し料金
特徴	・すぐにアクセスする必要のある大容量の長期保存データに最適 ・医療画像、ニュースメディアデータなど

S3 Glacier Flexible Retrieval

アクセス頻度	ほとんどアクセスしない
取り出し速度	取り出しオプションを選択できる
ストレージ料金	S3 Glacier Instant Retrieval よりも低価格
取り出し料金	取り出しオプションを選択できる
特徴	・データの取り出し方に柔軟性が必要な長期間保存データに最適 ・バックアップ、災害対策、オフサイトのデータストレージなど

オプション	データ取り出し時間	取り出し料金
迅速 (Expedited)	1~5分	高い
標準 (Standard)	3~5時間	低い
大容量 (Bulk)	5~12時間	無料

S3 Glacier Deep Archive

アクセス頻度	ほぼアクセスしない
取り出し速度	12~48時間かかる場合もある
ストレージ料金	S3 ストレージクラスの中で最も低価格
取り出し料金	S3 ストレージクラスの中で最も高額
特徴	・7~10年以上データを保管しなければならないような、規制厳しい業界のコンプライアンス要件を満たす ・バックアップや災害対策 ・磁気テープの代替策

オプション	データ取り出し時間	取り出し料金
標準 (Standard)	12時間以内	高い
大容量 (Bulk)	48時間以内	低い

S3 Intelligent-Tiering

2023年7月28日 14:52

- ▶ データを最も費用対効果の高いアクセス階層に自動的に移動し、ストレージコストを最適化
- ▶ モニタリングと、自動階層化が可能
(月額料金がかかる)
- ▶ 基本的に、データ取り出し料金はかからない

S3 Intelligent – Tiering アクセス階層	特徴
高頻度アクセス階層	S3 Standard と同じ低レイテンシーと高スループットのパフォーマンス
低頻度アクセス階層	ストレージコストを最大 40% 節約 30 日間アクセスされないと低頻度アクセス階層に移行
アーカイブインスタント アクセス階層	ストレージコストを最大 68% 節約 90 日間アクセスされないとアーカイブインスタントアクセス階層に移行
アーカイブアクセス階層 (オプション)	アーカイブアクセス階層オプションを有効化する必要がある 90 日間アクセスされないと自動的にアーカイブ S3 Glacier Flexible Retrieval ストレージクラスと同じパフォーマンス
ディープアーカイブ アクセス階層 (オプション)	ストレージコストを最大 95% 節約 ディープアーカイブアクセス階層オプションを有効化する必要がある 180 日間アクセスされないと自動的にアーカイブ S3 Glacier Deep Archive ストレージクラスと同じパフォーマンス

s3ライフサイクル

2023年7月28日 14:57

ライフサイクルルール

- ▶ バケット内のオブジェクトに対して、ストレージクラスの変更や削除処理を自動で行う機能



構築手順

2023年7月28日 14:59



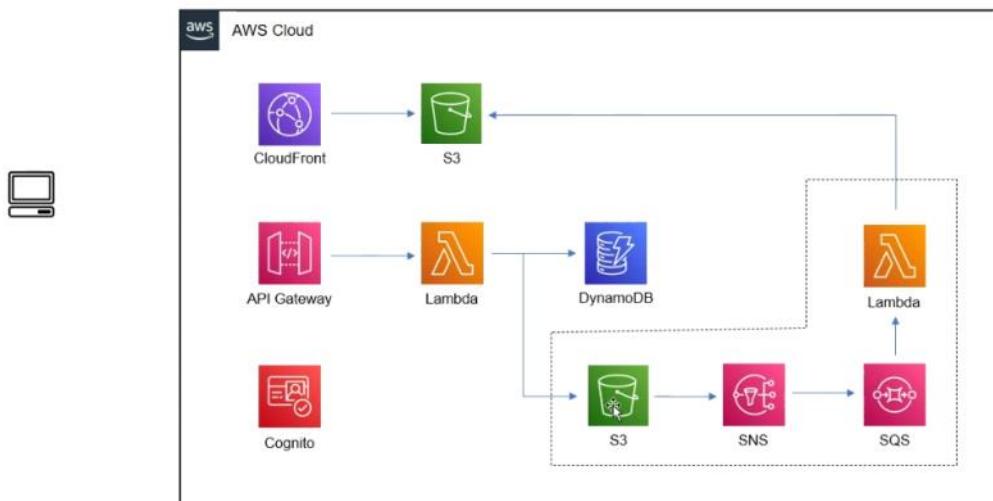
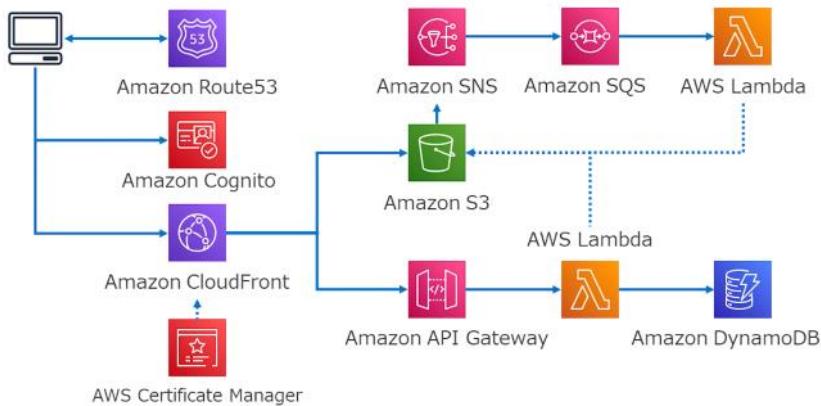
① ライフサイクルルート作成

画像格納用バケット選択→管理タブ→ライフサイクルルール作成

構成図

2023年7月28日 15:05

さいごに



- ・S3にホスティングしている ← 基本的にはしない
- ・CloudFrontでS3のコンテンツをHTTPPA経由でみれるようにする
- ・API Gateway作成 CORSの設定をしておく
- ・Lambda関数を実行するためにAPI GatewayをCloudFrontのオリジンに設定しておく
- ・Lambda関数実行(DB書き込み・読み取り用、S3画像格納用)
- ・S3に一時的に画像を保存する（直接S3に保存するとコストがかかる、PUTをトリガーにしているので分けないと無限ループしてしまう）
→対策としては、バケットを2つにわける・パスを分けるなどがある
- ・SNSで配信先をSQSへ指定する
- ・SQSをトリガーにLambda関数実行
- ・配信用のS3バケットに保存
- ・CognitoでAPI直アクセス対策＆ユーザー認証・認可設定をする

```
finally:  
    return {  
        'statusCode': statusCode,  
        'headers': {  
            # 'Access-Control-Allow-Headers': 'Content-Type',  
            # 'Access-Control-Allow-Origin': 'https://gakuen.ahaws.toyota-bibliotheca.com',  
            # 'Access-Control-Allow-Methods': 'GET'  
            'Access-Control-Allow-Headers': 'Content-Type',  
            'Access-Control-Allow-Origin': 'http://localhost:3000',  
            'Access-Control-Allow-Methods': 'GET'  
        },  
        'body': data  
    }
```

ドメインが違う場合はヘッダーに付与する設定を行って可能にする

SNSを挟まずに画像転送

2023年7月28日 17:02

<https://dev.classmethod.jp/articles/aws-s3-image-compress-lambda/>

①関数作成

SNS・SQS用の関数

```
records = event["Records"][0]
body = json.loads(records['body'])
message = json.loads(body['Message'])
s3 = message['Records'][0]['s3']
print(s3)
bucket = s3['bucket']['name']
key = s3['object']['key']
```

S3用の関数

```
import urllib.parse

bucket = event['Records'][0]['s3']['bucket']['name']
key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'],
encoding='utf-8')
```

※注意点

- ・レイヤーを追加する←AWSの既存の環境では使えないモジュールがある
- ・タイムアウトの設定←短すぎると処理が途中で終わってしまう

②トリガー

一時保存用のS3バケット→プロパティ→イベント通知で関数を設定する

CloudFormation

2023年7月31日 10:05

AWSリソースを自動的にデプロイするサービス

リソースの定義を.yamlファイルでできる

一つ作ってしまえば同じ環境を作ることができる

コードなので目に見える

エラーを減らすことができる

削除すればスタック上で作成されたものは全て削除できる

差異の確認もできる

検証環境と本番環境を分けるには違うユーザーで行う

検証環境をCloudFormationで作成し、完成したものを本番環境へ移せばミスがなくできる。

復習

2023年11月17日 16:03

①s3バケット作成し公開するファイルなどをアップロードする

②CloudFrontでディストリビューションを作成する

ディストリビューションを作成

オリジン

オリジンドメイン
AWS オリジンを選択するか、新しいオリジンのドメイン名を入力します。

オリジンバス - オプション [情報](#)
オリジンのエンドポイントのオリジンドメイン名に追加する URL パスを入力します。

名前
このオリジンの名前を入力します。

オリジンアクセス - 情報
 Public
Bucket must allow public access.
 Origin access control settings (recommended)
Bucket can restrict access to only CloudFront.
 Legacy access identities
Use a CloudFront origin access identity (OAI) to access the S3 bucket.

コントロール設定を作成する

- Origin access control
Select an existing origin access control (recommended) or create a new configuration.

- バケットポリシー
Policy must allow access to CloudFront IAM service principal role.
 ポリシーを手動で更新する

⚠️ S3 バケットポリシーを更新する必要があります
CloudFront は、ディストリビューションの作成後にポリシーステートメントを提供します。

カスタムヘッダーを追加 - オプション
CloudFront は、オリジンに送信するすべてのリクエストにこのヘッダーを含めます。

ヘッダーを追加

オリジンシールドを有効にする [情報](#)
Origin Shield は、オリジンの負荷を軽減し、可用性を保護するために役立つ追加のキャッシュレイヤーです。
 いいえ
 はい

▶ 追加設定

デフォルトのキャッシュビヘイビア

パスパターン [情報](#)
デフォルト (*)
オブジェクトを自動的に圧縮 [情報](#)

No
 Yes

ビューワー

ビューワープロトコルポリシー
 HTTP and HTTPS
 Redirect HTTP to HTTPS
 HTTPS only

許可された HTTP メソッド
 GET, HEAD
 GET, HEAD, OPTIONS
 GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE

HTTP メソッドをキャッシュ
GET メソッドと HEAD メソッドはデフォルトでキャッシュされます。
 オプション

ビューワーのアクセスを制限する
ビューワーのアクセスを無効する場合、ビューワーグルコンテンツにアクセスするには CloudFront 署名付き URL または署名付き cookie を使用する必要があります。

キャッシュキーとオリジンリクエスト

キャッシュキーとオリジンリクエストを削除するには、キャッシュポリシーとオリジンリクエストポリシーを使用することをお勧めします。

Cache policy and origin request policy (recommended)

Legacy cache settings

キャッシュポリシー
既存のキャッシュポリシーを選択するか、新しいキャッシュポリシーを作成します。

CachingDisabled Policy with caching disabled

Create cache policy ポリシーを表示

オリジンリクエストポリシー - オプション
既存のオリジンリクエストポリシーを選択するか、新しいオリジンリクエストポリシーを作成します。

オリジンポリシーを選択 Create origin request policy

レスポンスヘッダー - オプション
既存のレスポンスヘッダーを選択するか、新しいレスポンスヘッダーポリシーを作成します。

レスポンスヘッダーを選択 Create response headers policy

▶ 選択設定

その他デフォルト

③lambda関数作成

ロールに

- AmazonDynamoDBFullAccess
- AWSLambdaBasicExecutionRole

を追加する

④DynamoDBでテーブルの作成をする

型に注意する ※今回はpost_idも文字列

DynamoDB > テーブル > テーブルの作成

テーブルの作成

テーブルの詳細 情報

DynamoDBは、テーブルの作成時にテーブル名とプライマリキーのみを必要とするスキーマレスデータベースです。

テーブル名
テーブルを識別するために使用されます。
lab2-news

パーティションキー
パーティションキーは、テーブルのプライマリキーの一部です。これは、テーブルから項目を取得し、スケーラビリティと可用性のためにストリームデータを割り当てるために使用されるハッシュ値です。
post_id 文字列

ソートキー - オプション
ソートキーは、テーブルのプライマリキーの2番目の部分として使用できます。ソートキーにより、同じパーティションキーを共有するすべての項目をソートまたは検索できます。
ソートキー名を入力 文字列

DynamoDB > テーブル > テーブルの作成

テーブルの作成

テーブルの詳細 情報

DynamoDBは、テーブルの作成時にテーブル名とプライマリキーのみを必要とするスキーマレスデータベースです。

テーブル名
テーブルを識別するために使用されます。
lab2-sequences

パーティションキー
パーティションキーは、テーブルのプライマリキーの一部です。これは、テーブルから項目を取得し、スケーラビリティと可用性のためにストリームデータを割り当てるために使用されるハッシュ値です。
name 文字列

ソートキー - オプション
ソートキーは、テーブルのプライマリキーの2番目の部分として使用できます。ソートキーにより、同じパーティションキーを共有するすべての項目をソートまたは検索できます。
ソートキー名を入力 文字列

⑤テーブルに項目を追加する

DynamoDB > 新規作成 lab2-sequences > 項目を作成

項目を作成 フォーム JSON ビュー

項目を作成

既存の属性を追加、削除、または変更できます。属性は、最大 32 レイアの深さまで他の属性内にネストできます。詳細はこちら

属性

属性名	値	タイプ
name - パーティションキー	name	文字列
current_number	0	整数

新しい属性の追加

キャンセル 確認を作成

⑥APIゲートウェイの作成

REST API

API 管理機能とともに、リクエストとレスポンスを完全に制御できる REST API を開発します。

以下で動作します。

Lambda、HTTP、AWS サービス

REST API を作成

API の詳細

新しい API
新しい REST API を作成します。

現存の API のクローンを作成
この AWS アカウントに API のコピーを作成します。

API をインポート
OpenAPI 定義から API をインポートします。

サンプル API
サンプル API を使用して API Gateway の詳細を確認します。

API 名

説明 オプション

API エンドポイントタイプ
リージョンにレバリングの API は、現在の AWS リージョンでデプロイされます。エッジ最適化 API は、CloudFront の最も近い Point of Presence にリクエストをルーティングします。プライベート API には VPC からのみアクセスできます。

リージョン

⑦リソース作成

リソースを作成

リソースの詳細

プロキシのリソース 情報
プロキシリソースはすべてのサブリソースに対するリクエストを処理します。プロキシリソースを作成するには、末尾にプラス記号の / (スラッシュ) を使用します (例: proxy-*)。

リソース名

リソースパス

CORS (クロスオリジンリソース共有) 情報
すべてのオリジン、すべてのメソッド、およびいくつかの共通ヘッダーを許可する OPTIONS メソッドを作成します。

⑧GET,POST メソッド作成

メソッドの詳細

メソッドタイプ

統合タイプ

Lambda 関数
API を Lambda 関数と統合します。


HTTP
既存の HTTP エンドポイントと統合します。


Mock
API Gateway のマッピングと後方接続に基づいてレスポンスを生成します。


AWS のサービス
AWS のサービスと統合します。


VPC リンク
VPC リンク（アマゾンクラウドネットワーク）ではアクセスできないリソースと統合します。


Lambda プロキシ統合
リクエストを構造化されたイベントとして Lambda 関数に送信します。

Lambda 関数
Lambda 関数の名前またはエリアスを指定します。別のアカウントからの ARN を指定することもできます。

GET のテスト

リソースを作成

メソッドリクエスト | 総合リクエスト | 総合レスポンス | メソッドレスポンス | テスト

テストメソッド
リクエスト URL を実行します。API ゲートウェイが該当 URL を呼び出します。

クエリ文字列

ヘッダー
ヘッダーフィールドをコマンドで直接入力します。ヘッダーごとに改行してください。

クライアント証明書
No client certificates have been generated.

レスポンス
レスポンス
レスポンスヘッダー
レスポンス本文

```
[{"id":1,"title": "Hello World","body": "Welcome to AWS Lambda!","published": true,"category": "Tech"}, {"id":2,"title": "AWS Lambda Functions","body": "AWS Lambda is a serverless compute service that lets you run code without provisioning or managing servers.", "published": true,"category": "Tech"}]
```

メソッドの詳細

メソッドタイプ
POST

統合タイプ

- Lambda 関数
API を Lambda 関数と統合します。

- HTTP
既存の HTTP エンドポイントと統合します。

- Mock
API Gateway のマッピングと並んでに基づいてレスポンスを生成します。


- AWS のサービス
AWS のサービスと統合します。

- VPC リンク
ノードリンクインターネット経由ではアクセスできないリソースと統合します。


Lambda プロキシ統合
リクエストを構造化されたイベントとして Lambda 関数に送信します。

Lambda 関数
Lambda 関数の名前またはエイリアスを指定します。別のアカウントからの ARN を指定することもできます。

ap-northe... ▾ Q arm:aws:lambda:ap-northeast-1:157094121738:function:...

⑨APIをデプロイする

Deploy API

API がデプロイされるステージを選択します。例えば、API のテスト版を beta という名前のステージにデプロイできます。

ステージ
New stage

ステージ名
dev

❶ 新しいステージがデフォルト設定で作成されます。[ステージ] ページでステージ設定を編集します。

デプロイメントの説明

キャンセル テブロイ

⑨CloudFrontのオリジンを作成する(ディストリビューションを選択→オリジンタブ)

※オリジンパスを空白にしておく

オリジンを作成

設定

オリジンドメイン
AWS オリジンを選択するか、お使いのオリジンのドメイン名を入力します。
jaechkkbpZ.execute-api.ap-northeast-1.amazonaws.com

プロトコル 情報

HTTPのみ
 HTTPSのみ
 マッチビューワー

- HTTP ポート
オリジンの HTTP ポートを入力します。デフォルトはポート 80 です。
80

- HTTPS ポート
オリジンの HTTPS ポートを入力します。デフォルトはポート 443 です。
443

- 最小オリジン SSL プロトコル 情報
CloudFront がオリジンで使用する最小 SSL プロトコル。
 TLSv1.2
 TLSv1.1
 TLSv1

オリジンバス - オプション [情報](#)
オリジンエクストのオリジンドメイン名に追加する URL パスを入力します。

オリジンバスを入力

名前
このオリジンの名前を入力します。
jechkkbp2.execute-api.ap-northeast-1.amazonaws.com

カスタムヘッダーを追加 - オプション
CloudFront は、オリジンに送信するすべてのリクエストにこのヘッダーを含めます。

ヘッダーを追加

オリジンシールドを有効にする [情報](#)
Origin Shield は、オリジンの負荷を軽減し、可用性を保護するために役立つ追加のキャッシュレイヤーです。

いいえ
 はい

▶ 進級設定

キャンセル [オリジンを作成](#)

⑩ビヘイビアの作成

パスパターンをAPIのステージ名と一致させる

設定

パスパターン [情報](#)
dev/*

オリジンとオリジングループ
nnvvkqm83f.execute-api.ap-northeast-1.amazonaws.com

オブジェクトを自動的に圧縮 [情報](#)
 No
 Yes

ビューワー

ビューワーフロトコレボリシー
 HTTP and HTTPS
 Redirect HTTP to HTTPS
 HTTPS only

許可された HTTP メソッド
 GET, HEAD
 GET, HEAD, OPTIONS
 GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE

HTTP メソッドをキャッシュ
GET メソッドと HEAD メソッドはデフォルトでキャッシュされます。
 オプション

ビューワーのアクセスを制限する
ビューワーのアクセスを制限する場合、ビューワーがコンテンツにアクセスするには CloudFront 著名付き URL または著名付き cookie を使用する必要があります。

No
 Yes

キャッシュキーとオリジンリクエスト

キャッシュキーとオリジンリクエストを割離するには、キャッシュポリシーとオリジンリクエストポリシーを使用することをお勧めします。

Cache policy and origin request policy (recommended)
 Legacy cache settings

- ヘッダー
キャッシュキーに含めるヘッダーを選択します。
なし

- クエリ文字列
キャッシュキーに含めるクエリ文字列を選択します。
すべて

- cookie
キャッシュキーに含める cookie を選択します。
なし

- オブジェクトキャッシュ
 Use origin cache headers
 Customize

最小 TTL 最低存続時間 (秒)。 0	最大 TTL 最大存続時間 (秒)。 0	デフォルト TTL デフォルトの存続時間 (秒)。 10
----------------------------	----------------------------	------------------------------------

レスポンスヘッダーポリシー - オプション
既存のレスポンスヘッダーポリシーを選択するか、新しいレスポンスヘッダーポリシーを作成します。

レスポンスヘッダーを選択 [C](#)
Create response headers policy [D](#)

⑪カスタムエラーレスポンスを編集

カスタムエラーレスポンスを編集

エラーレスポンス 情報

HTTP エラーコード
オリジンがこのエラーコードを送信するときのカスタムエラーレスポンスをカスタマイズします。

403: Forbidden ▾

最小 TTL のキャッシュエラー
エラーキャッシュの最小存続時間 (TTL) を秒単位で入力します。
0

エラーレスポンスをカスタマイズ
オリジンから受け取ったエラーの代わりに、カスタムエラーレスポンスを送信します。
 いいえ
 はい

レスポンスページのパス
カスタムエラーレスポンスページのパスを入力します。
/index.html

HTTP レスポンスコード
HTTP ステータスコードを選択して、ビューワーに戻ります。CloudFront は、オリジンから受け取ったものとは異なるステータスコードをビューワーに返すことができます。

200: OK ▾

キャンセル **変更を保存**

カスタムエラーレスポンスを編集

エラーレスポンス 情報

HTTP エラーコード
オリジンがこのエラーコードを送信するときのカスタムエラーレスポンスをカスタマイズします。

404: Not Found ▾

最小 TTL のキャッシュエラー
エラーキャッシュの最小存続時間 (TTL) を秒単位で入力します。
0

エラーレスポンスをカスタマイズ
オリジンから受け取ったエラーの代わりに、カスタムエラーレスポンスを送信します。
 いいえ
 はい

レスポンスページのパス
カスタムエラーレspoんスページのパスを入力します。
/index.html

HTTP レスポンスコード
HTTP ステータスコードを選択して、ビューワーに戻ります。CloudFront は、オリジンから受け取ったものとは異なるステータスコードをビューワーに返すことができます。

200: OK ▾

キャンセル **変更を保存**

⑫画像格納用のs3バケットを作成する
lab2-s3-storage-image

⑬ロールを作成する

許可ポリシー [4] **確認**

最大 10 個の権限ポリシーを登録できます。

検索 タイプ

Q: lab2

結果

権限	タイプ	アタッチされたエンティティ
arn:aws:lambda:FullAccess	AWS 管理	1
arn:aws:lambda:InvokeRole	AWS 管理	3
arn:aws:lambda:SQSQueueExecutionRole	AWS 管理	1
lab2-dynamodb	カスタマーマイドイン	0

⑭画像サイズ変更用のLambda関数を作成する

関数の作成 **確認**

AWS Lambda Application Repository フォルダ「lambda」は「アーティキュレーション」に登録されました。

一括登録 Lambda関数を複数登録します。
 関数の構造を複数登録します。
 コンテナイメージを複数登録します。

基本的な情報

関数名: Lambda関数名として入力します。
 Lambda関数の構造

言語: Python 3.9

アーカイブ: Lambda関数の構造をzipアーカイブでアップロードします。
 Lambda関数の構造を直接アップロードします。
 Lambda関数の構造を直接アップロードします。

アクセス権限 **確認**

関数の実行権限を付与する IAM ロールを選択します。

関数の実行権限を付与する IAM ロールを選択します。
 lab2-dynamodb

詳細設定

関数の実行権限を付与する IAM ロールを選択します。
 lab2-dynamodb

確認 **確認**

⑮コード記入後に関数の設定をする
環境変数の設定

環境変数の編集

環境変数

関数のコードからアクセス可能なキーと値のペアを環境変数で定義できます。これらの環境変数は、関数のコードを変更することなく構成の設定を保存することができるので便利です。 [詳細はこちら](#)

キー	値	削除
BucketName	lab2-s3	削除
TableName	lab2-news	削除
環境変数の追加		

▶ 暗号化の設定

キャンセル 保存

タイムアウト時間設定

コード テスト モニタリング **設定** エイリアス パージons

一 般設定

詳細	標準	エフェクト
アラーム通知	XELU 128 MB	エフェクト(ル)ストレージ 512 MB
遅延	タイムアウト 1 分 0 秒	Asistant: 標準
機能	機能	
機能変数	機能変数	

レイヤーを追加する

arn:aws:lambda:ap-northeast-1:770693421928:layer:Klayers-p39-pillow:1

⑯既存のLambda関数を編集する

環境変数 (2)

以下の環境変数はデフォルトの Lambda サービスキーを使用して最初に暗号化されました。

キー	値	編集
BucketName	lab2-s3-image-storage	
TableName	lab2-news	

⑰SNSの作成

トピックの作成

詳細

タイプ: 標準

トピックの内容後にトピックタイプを変更することはできません

FIFO (先入れ先出し, 先出し)

- 最新に保存されたメッセージの順序付け
- 1回のみメッセージ配信
- 最大メッセージサイズ: 1000 件のバツリッシュ
- サブスクリプションプロトコル: SQS

スタンダード

- ベストエフォート型メッセージの順序付け
- 少なくとも 1 回のメッセージ配信
- 最大メッセージ数: 1000 件のバツリッシュで最高のスループット
- サブスクリプションプロトコル: SQS, Lambda, HTTP, SMS, メール, モバイルアプリケーションエンドポイント

名前: lab2-dev-sns-from-s3

最大文字数は 256 文字です。英数字、ハイフン(-)、およびアンダースコア(_)を含めることができます。

表示名: オプション 標準

SMS のサブスクリプションでこのトピックを使用するには、表示名を入力します。SMS メッセージには最初の 10 文字のみが表示されます。

My Topic

最大: 100 文字

⑲アクセスポリシーの変更

※ActionはPublicのみに変更する

```
{
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "__default_statement_ID",
      "Effect": "Allow",
      "Principal": {
        "Service": "s3.amazonaws.com"
      },
      "Action": "SNS:Publish",
      "Resource": "arn:aws:sns:ap-northeast-1:157094121738:lab2-sns-from-s3",
      "Condition": {
        "StringEquals": {
          "AWS:SourceAccount": "157094121738",
          "aws:SourceArn": "arn:aws:s3:::lab2-s3-image-storage"
        }
      }
    }
  ]
}
```

```
]  
}  
}
```

⑩画像格納用s3バケットのイベント通知を作成する

イベント通知を作成 情報

通知を有効にするには、まず、Amazon S3 が発行するイベントと、Amazon S3 が通知を送信する送信先を特定する通知設定を追加する必要があります。

一般的な設定

イベント名

イベント名には最大 255 文字まで使用できます。

プレフィックス - オプション
指定された文字で始まるキーがあるオブジェクトに通知を制限します。

サフィックス - オプション
指定された文字で終わるキーがあるオブジェクトに通知を制限します。

イベントタイプ

通知を受け取るイベントを少なくとも 1つ指定してください。グループごとに、すべてのイベントに対するイベントタイプを選択するか、1つ、または複数の個別のイベントを選択することができます。

オブジェクトの作成

- すべてのオブジェクト作成イベント
s3:ObjectCreated: PUT
s3:ObjectCreated:Put
- POST
s3:ObjectCreated:Post
- コピーする
s3:ObjectCreated:Copy
- 完了したマルチパートアップロード
s3:ObjectCreated:CompleteMultipartUpload

オブジェクトの削除

- すべてのオブジェクト削除イベント
s3:ObjectRemoved: 完全に削除されました
s3:ObjectRemoved:Delete
- 作成された削除マークー
s3:ObjectRemoved:DeleteMarkerCreated

オブジェクトの復元

- すべてのオブジェクト復元イベント
s3:ObjectRestore: 復元が開始されました
s3:ObjectRestore:Post
- 復元が完了しました
s3:ObjectRestore:Completed
- 復元されたオブジェクトの有効期限切れ
s3:ObjectRestore:Delete

送信先

① Amazon S3 が送信先にメッセージを発行する前に、関連する API を呼び出して SNS トピック、SQS キュー、または Lambda 関数にメッセージを発行するために必要なアクセス許可を Amazon S3 の原則に付与する必要があります。[詳細](#)

送信先
イベントを発行する送信先を選択します。[詳細](#)

- Lambda 関数
S3 イベントに基づいて Lambda 関数スクリプトを実行します。
- SNS トピック
メッセージをシステムにファンアウトして並列処理するか、直接人に送信します。
- SQS キュー
サーバーによる読み込みのため SQS キューに通知を送信します。

SNS トピックを特定

- SNS トピックから選択する
- SNS トピック ARN を入力

SNS トピック

キャンセル

変更の保存

②sns作成

キューを作成

詳細

タイプ
アプリケーションまたはクラウドインフラストラクチャのキュータイプを選択します。

標準 標準
少なくとも 1 回の配信。メッセージの順序は保持されません
• At-least once 配信
• ベストエフォート型の順序

FIFO 標準
先入れ先出し順道。メッセージの順序が保持されます
• FIFO 順道
• 1 回のみ処理

① キューの作成後にキュータイプを変更することはできません。

名前
lab2-sqs
キュー名は6文字と小文字で入力できます。英数字、ハイフント、アンダースコア(_)を使用できます。

設定 情報
最大メッセージサイズ、他のコンシューマーに対する可視性、およびメッセージの保持期間を設定します。

可視性タイムアウト 情報
60 秒
0 秒～12 時間の間である必要があります。

メッセージ保持期間 情報
4 日
1 分～14 日の間である必要があります。

配信遅延 情報
0 秒
0 秒～15 分の間である必要があります。

最大メッセージサイズ 情報
256 KB
1 KB～256 KB の間である必要があります。

メッセージ受信待機時間 情報
0 秒
0～20 秒の間である必要があります。

21.サブスクライブする

Amazon SNS トピックにサブスクライブ 情報

Amazon SNS トピック

キューが Amazon SNS トピックからのメッセージを受信できるようにするには、キューを Amazon SNS トピックにサブスクライブします。

このキューに使用できる Amazon SNS トピックを指定します。

arn:aws:sns:ap-northeast-1:157094121738:lab2-sns-from-s3

キャンセル 保存

※確認方法

lab2-sqs

詳細 情報

名前
lab2-sqs

タイプ
標準

ARN
arn:aws:sqs:ap-northeast-1:157094121738:lab2-sqs

URL
<https://sqs.ap-northeast-1.amazonaws.com/157094121738/lab2-sqs>

デットレーティキューム
-

さらに表示

SNS サブスクリプション | Lambda トリガー | EventBridge Pipes | デットレーティキューム | モニタリング | タグ付け | アクセスポリシー | 編号化 | デットレーティキュームの再処理タスク

サブスクリプションリージョン
ap-northeast-1

SNS サブスクリプション (1) 情報

サブスクリプションを検索

サブスクリプション ARN
arn:aws:sns:ap-northeast-1:157094121738:lab2-sns-from-s3

トピック ARN
arn:aws:sns:ap-northeast-1:157094121738:lab2-sns-from-s3

Amazon SNS > トピック > lab2-sns-from-s3

lab2-sns-from-s3

詳細

名前
lab2-sns-from-s3

表示名
-

ARN
arn:aws:sns:ap-northeast-1:157094121738:lab2-sns-from-s3

トピックの所有者
157094121738

タイプ
スタンダード

サブスクリプション | アクセスポリシー | データ保護ポリシー | 記信ポリシー (HTTP/S) | 記信ステータスのログ記録 | 編号化 | タグ | 総合

サブスクリプション (1)

サブスクリプションの作成

検索

ID	エンドポイント	ステータス	プロトコル
77428471-e4f0-483a-b0e9-d16617903960	arn:aws:sqs:ap-northeast-1:157094121738:lab2-sqs	確認済み	SQS

メッセージ本文

メッセージ構造

- すべての配信プロトコルに同一のペイロード。
配信プロトコルに勝手なく、トピックにサブスクライブしているエンドポイントに同じペイロードが送信されます。

- 配信プロトコルごとにカスタムペイロード。
配信プロトコルに基づいて、トピックにサブスクライブしているエンドポイントに異なるペイロードが送信されます。

エンドポイントに送信するメッセージ本文

1 メッセージ本文

メッセージ属性 情報

メッセージ属性を使用すると、メッセージの構造化メタデータ項目（タイムスタンプ、地理空間データ、署名、識別子など）を指定できます。

タイプ

名前

値

属性タイプの選択

属性名の入力

値または ["value1", "value2"]

削除

別の属性の追加

キャンセル

メッセージの発行

Amazon SQS > キュー > lab2-sqs

lab2-sqs

編集 削除 クリア メッセージを削除

DLQ 再投擲の開始

詳細

名前
lab2-sqs

タイプ
標準

ARN

arn:aws:sqs:ap-northeast-1:157094121738:lab2-sqs

暗号化
Amazon SQS キー (SSE-SQS)

URL
https://sqs.ap-northeast-1.amazonaws.com/157094121738/lab2-sqs

デッドレターキュー

さらに表示

メッセージを受信

ポーリング設定を編集 ポーリングを停止 メッセージをポーリング

利用可能なメッセージ

0

ポーリング間隔

30

最大メッセージ数

10

ポーリングの進行状況

1件登録

0%

メッセージ (0)

メッセージを検索

詳細を表示 削除

ID

送信済み

サイズ

送信数

メッセージがありません。キュー内のメッセージを表示するには、メッセージをポーリングします。

メッセージをポーリング

22.Lambda トリガーを設定する

Amazon SQS > キュー > lab2-sqs > AWS Lambda 関数をトリガー

AWS Lambda 関数をトリガー 情報

Lambda 関数

受信メッセージが Lambda 関数をトリガーするように設定します。

リージョン

ap-northeast-1

このキューに使用できる AWS Lambda 関数を指定します。

arn:aws:lambda:ap-northeast-1:157094121738:function:lab2-resize-image

キャンセル

保存

23.Cognitoの設定

サインインエクスペリエンスを設定 情報

アプリケーションユーザーは、ユーザー名とパスワードを使用してユーザープールにサインインするか、サードパーティのアイデンティティプロバイダーを使用してサインインできます。

認証プロバイダー

ユーザーがサインインするときに利用できるプロバイダーを設定します。

プロバイダーのタイプ

ユーザーが Cognito ユーザープール、フェデレーテッドアイデンティティプロバイダー、またはその両方にサインインするかどうかを選択します。Amazon Cognito では、フェデレーテッドユーザーとユーザープールユーザーの料金が異なります。料金の詳細は [こちら](#) []

Cognito ユーザープール

ユーザーは E メールアドレス、電話番号、またはユーザー名を使用してサインインできます。ユーザー属性、グループ、メタデータ、およびセキュリティ設定はユーザープールに保存され、ユーザープールに設定されます。

フェデレーテッドアイデンティティプロバイダー

ユーザーは、Facebook、Google、Amazon、Apple などのソーシャルアイデンティティプロバイダーの認証情報を使用して、または SAML、もしくは Open ID Connect を介して外部ディレクトリの認証情報を使用して、サインインできます。ユーザー名のフェデレーテッドユーザーのユーザー属性マッピングとセキュリティを管理できます。

Cognito ユーザープールのサインインオプション 情報

サインインに使用するユーザープールの属性を選択します。属性を 1 つだけ選択した場合、またはユーザー名と 1 つ以上の他の属性を選択した場合、ユーザーは選択したすべてのオプションを使用してサインインできます。電話番号と E メールのみを選択した場合、サインアップ時に 2 つのサインインオプションのいずれかを選択するよう求められます。

- ユーザー名
 E メール
 電話番号

ユーザー名の要件

- ユーザーが任意のユーザー名でサインインすることを許可
 ユーザー名の大文字と小文字を区別する

⚠️ ユーザープールの作成後に Cognito ユーザープールのサインインオプションを変更することはできません。

キャンセル

次へ

セキュリティ要件を設定 情報

多要素認証に加えて強力なパスワード要件を設定し、アプリケーションユーザーが誤って認証情報を漏えいするのを防ぎます。

パスワードポリシー 情報

パスワードポリシーを作成して、ユーザーが認定できるパスワードの長さと複雑さを定義します。

パスワードポリシーモード 情報

- Cognito のデフォルト
デフォルトのパスワード要件を使用します。

- カスタム
ユーザーが定義するパスワード要件を使用します。

パスワードの最小文字数

8 文字

パスワード要件

- 少なくとも 1 つの数字を含む
少なくとも 1 つの特殊文字を含む
少なくとも 1 つの大文字を含む
少なくとも 1 つの小文字を含む

管理者によって設定された仮パスワードの有効期限:

7 日

多要素認証

ユーザーのサインインプロセス中に多要素認証(MFA)を強制することで、アプリへの安全なアクセスを設定します。MFA の設定は、すべてのアプリクライアントに適用されます。

MFA の強制 情報

- MFA を必須にする - 推奨
ユーザーは、サインイン時に追加の認証要素を指定する必要があります。

- オプションの MFA
ユーザーは、1 つの認証要素でサインインでき、追加の認証要素を追加することもできます。

- MFA なし
ユーザーは 1 つの認証要素のみでサインインできます。これは、最も安全性が低いオプションです。

メッセージ配信を設定 情報

Amazon Cognito は、Amazon SES と Amazon SNS を使用して、E メールや SMS メッセージをアプリケーションユーザーに送信します。メッセージについては、追加の SES および SNS コストが発生する場合があります。

E メール

ユーザープールがユーザーに E メールメッセージを送信する方法を設定します。

E メールプロバイダー 情報

Amazon SES で E メールを送信 - 推奨

アカウントの Amazon SES 基本済みアイデンティティを使用して E メールを送信します。E メールの量が多く、本番ワークフローの場合は、このオプションをお勧めします。

Cognito で E メールを送信

Cognito のデフォルトの E メールアドレスは、既定で選択するに際しての一時的なものとして使用します。1 日に最大 50 通の E メールを送信するために使用できます。

SES の機能を使用するには、Amazon SES [で検証済み送信者が設定されている必要があります](#)。詳細は[こちら](#)

SES リージョン 情報

アジアパシフィック (東京)

送信元の E メールアドレス 情報

デフォルトでは、「no-reply@verificationemail.com」が使用されます。Amazon SES で検証済みの場合は、別のメールアドレスを選択することもできます。

no-reply@verificationemail.com



返信先 E メールアドレス - オプション 情報

無効な送信アドレスを設定すると、アカウントに送信制限が適用される場合があります。

メールアドレスを入力

キャンセル

戻る

次へ

アプリケーションを統合 情報

Cognito の組み込みの認証および承認フローを使用して、ユーザープールのためにアプリケーション統合を設定します。

ユーザープール名

ユーザープールのわかりやすい名前を作成します。

ユーザープール名

lab2-userpool

ユーザープール名は 128 文字以下である必要があります。名前には、英数字、スペース、+ * . @ - といった特殊文字のみを使用できます。

⚠ このユーザープールを作成すると、ユーザープール名を変更できなくなります。

ホストされた認証ページ

ユーザーのサインアップおよびサインインフローのために Cognito のホストされた UI と OAuth 2.0 サーバーを使用するかどうかを選択します。

Cognito のホストされた UI を使用

ホストされたサインアップ、サインイン、および OAuth 2.0 サービスエンドポイントを Amazon Cognito で構築します。この機能が有効になっていない場合は、Cognito API オペレーションを使用してサインアップとサインインを実行します。

Cognito ドメイン・・・一意ならOK

ドメイン 情報

ホストされた UI および OAuth 2.0 エンドポイントのドメインを設定します。ホストされた UI を使用するには、認証エンドポイントが作成されるドメインを選択する必要があります。

ドメインタイプ

Cognito ドメインを使用する

Amazon が所有するドメインで使用する識別プレフィックスを入力します。本番稼働用アプリケーションの場合は、代わりにカスタムドメインを使用することをお勧めします。

カスタムドメインを使用

Cognito がホストするサインアップページとサインインページで所有するドメインを入力します。カスタムドメインを使用するには、DNS レコードと AWS Certificate Manager (ACM) 証明書を指定する必要があります。本番稼働用ワークフローにはカスタムドメインを使用することをお勧めします。

Cognito ドメイン

ドメインプレフィックスを入力します。

.auth.ap-northeast-1.amazoncognito.com

ドメインプレフィックスには、小文字、英数字、ハイフンのみを使用できます。ドメインプレフィックスには aws、amazon、または cognito というテキストを使用することはできません。ドメインプレフィックスは、現在のリージョン内で一意である必要があります。

使用可能

コールバック URL は認証が通った際にアクセスされる URL

最初のアプリケーションクライアント

アプリケーションクライアントを設定します。アプリケーションクライアントは、認証されていない API オペレーション呼び出すための許可を持つユーザーブール内の唯一のアプリケーションプラットフォームです。ユーザーブールは複数のアプリケーションクライアントを持つことができます。

アプリケーションタイプ 情報

アプリケーションタイプを選択すると、一般的なデフォルト設定が自動的に入力されます。ユーザーブールの作成後にアプリケーションクライアントをさらに追加できます。

パブリッククライアント

ネイティブアプリケーション、ブラウザアプリケーション、またはモバイルデバイスアプリケーション。Cognito API リクエストは、クライアントのシークレットで信頼されていないユーザーステムから実行されます。

秘密クライアント

クライアントのシークレットを安全に保存できるサーバー側のクライアント。Cognito API リクエストは中央サーバーから実行されます。

その他

カスタムアプリケーション。独自の許可、認証フロー、およびクライアントのシークレットの設定を選択します。

アプリケーションクライアント名 情報

アプリケーションクライアントのフレンドリーナーを入力します。

lab2-dev-app

アプリケーションクライアントの名前は 128 文字以下にする必要があります。名前には、英数字、スペース、+ = . @ - といった特殊文字のみを使用できます。

クライアントシークレット 情報

アプリケーションクライアントがクライアントのシークレットを持つかどうかを選択します。クライアントのシークレットは、API リクエストを認証するためにアプリケーションのサーバー側のコンポーネントによって使用されます。クライアントのシークレットを使用すると、第三者がクライアントになりますことを防ぐことができます。

クライアントのシークレットを生成する

クライアントのシークレットを生成しない

⚠️ Amazon Cognito がアプリケーションクライアント用にクライアントのシークレットを生成することを許可した後で、当該シークレットを変更または削除することはできません。

許可されているコールバック URL 情報

認証後にユーザーをリダイレクトするコールバック URL を少なくとも 1 つ入力します。通常、これは Cognito によって発行された認証コードを受け取るアプリケーションの URL です。HTTPS URL とカスタム URL スキームを使用できます。

URL

https://d2khfmka5s3rgs.cloudfront.net/news/news

削除

コールバック URL の長さは 1~1024 文字にする必要があります。使用可能な文字は、文字、マーク、數字、記号、および記点です。Amazon Cognito では、テスト目的でのみ http://localhost を除く HTTP 諸由の HTTPS が必要です。myapp://example などのアプリケーションのコールバック URL もサポートされています。フラグメントを含めることはできません。

高度なアプリケーションクライアントの設定

以前の選択内容に基づいて、推奨される認証フロー、OAuth 2.0 許可タイプ、および OIDC スコープが入力されています。

認証フロー 情報

アプリケーションがサポートする認証フローを選択します。更新トークン認証は常に有効になっています。アプリケーションのタイプに基づいてオプションが設定されています。

認証フローを選択

ALLOW_REFRESH_TOKEN_AUTH X
刷新トークンベースの認証

ALLOW_ADMIN_USER_PASSWORD_AUTH X
認証のための管理 API のユーザー名/パスワード認証

認証フローセッションの持続期間 情報

3 分

3~15 分の間である必要があります。

更新トークンの有効期限 情報

30 日 0 分

60 分~10 年の間である必要があります。

アクセストークンの有効期限 情報

0 日 60 分

5 分~1 日の間で設定してください。値は更新トークンの有効期限を超えることはできません。

ID トークンの有効期限 情報

0 日 60 分

5 分~1 日の間で設定してください。値は更新トークンの有効期限を超えることはできません。

高度なセキュリティ設定 - オプション

トークンの取り消しを有効化 情報

Amazon Cognito では、失効を可能にするために、アクセスおよび ID トークンに新しいクレームが追加される予定です。これにより、トークンのサイズが大きくなります。

ユーザー存在エラーの防止 情報

Amazon Cognito 認証 API は、ユーザーが見つからなかったことを示す代わりに、ユーザー名またはパスワードが正しくないことを示す一般的な認証失敗レスポンスを返します。

追加のユーザー属性データを承認 情報

アプリケーションクライアントは、認証されていないリクエストにアプリケーションが追加する IP アドレスを受け入れます。この IP アドレスとデバイスフィンガープリントは、Amazon Cognito の高度なセキュリティ機能によるリスク評価に寄与します。追加のユーザー属性データを受け入れない場合、アプリケーションクライアントはデバイスフィンガープリントのみを受け入れます。

ID プロバイダー **情報**
このアプリケーションクライアントに対して使用可能なアイデンティティプロバイダーを選択します。

アイデンティティプロバイダーを選択

Cognito ユーザーブール
ユーザーは、メールアドレス、電話番号、またはユーザーネームを使用して Cognito にサインインできます。

OAuth 2.0 許可タイプ **情報**
少なくとも 1 つの OAuth 許可タイプを選択して、Cognito がこのアプリケーションにトークンを配信する方法を設定します。選択したアプリケーションタイプに基づいて、推奨オプションが自動で指定されています。

OAuth 2.0 許可タイプを選択

暗黙的な付与
クライアントがアクセストークン（およびオプションでスコープに基づく ID トークン）を直接取得するように指定します。

⚠️ 暗黙的な許可フローは、URL で OAuth トークンを公開します。パブリッククライアントには、PKCE を使用する認証コードフローのみを使用することをお勧めします。

① Cognito のホストされた UI を使用している場合、クライアント認証情報の許可タイプは、このクライアントの作成中に無効になりますが、アプリケーションクライアントのホストされた UI 設定の構築ページで追加できます。

OpenID Connect のスコープ **情報**
少なくとも 1 つの OpenID Connect (OIDC) スコープを選択して、このアプリケーションクライアントがアクセストークン用に取得できる属性を指定します。選択したアプリケーションタイプと必須属性に基づいて、推奨オプションが自動で指定されています。

OIDC スコープを選択

OpenID 電話番号 E メール
OpenID を選択する必要があります

許可されているサインアウト URL - オプション **情報**
少なくとも 1 つのサインアウト URL を入力します。サインアウト URL は、アプリケーションがユーザーをサインアウトする際に Cognito によって送信されるリダイレクトページです。これは、Cognito がサインアウトしたユーザーをコールバック URL 以外のページに転送するようにする場合にのみ必要です。

サインアウト URL を追加

URL をさらに 100 個追加できる

23.ユーザーを作成する

ユーザーを作成 **情報**

▶ ユーザーブールのサインインとセキュリティ要件
このユーザーを作成するときに適用されるユーザーブールのセキュリティ設定を確認します。

ユーザー情報
このユーザーの権限とサインインのオプションを設定します。

招待メッセージ **情報**
[Messaging] タブ で招待メッセージテンプレートを設定する

招待を送信しない
 E メールで招待を送信

ユーザー名
ユーザー名は、ユーザーブールおよび上記の設定に基づく必須の属性です。
user01

E メールアドレス・オプション
このユーザーの E メールアドレスを入力します。ユーザーの E メールアドレスは、サインイン、アカウントの復旧、およびアカウントの確認に使用できます。
mail@gmail.com

E メールアドレスを検証済みとしてマークする

仮パスワード
Amazon Cognito は、生成したパスワードを E メールメッセージでユーザーに送信します。

パスワードの設定
 パスワードの生成

パスワード
このユーザーの仮パスワードを入力します。仮パスワードは、招待メッセージでユーザーに送信されます。

パスワードを表示

キャンセル

24.仮パスワードなので初期パスワードの変更をする

アプリケーションの統合タブから

アプリケーションクライアントのリスト

アプリケーションとユーザーブールを結合するアプリケーションクライアント。ユーザーブールのデフォルト設定に対するクライアントの上書きを設定し、Amazon Pinpoint 分析を設定します。

アプリクライアントと分析 (1/1) 情報		<input type="button" value="C"/>	<input type="button" value="削除"/>	アプリケーションクライアントを作成
アプリケーションクライアントを設定します。アプリケーションクライアントは、アプリケーションにアタッチされたユーザーブールの認証リソースです。アプリケーションクライアントを選択して、アプリケーション用に許可される認証アクションを設定します。				
<input type="text" value="Q アプリケーションクライアントを名前または ID で検索"/>				
アプリケーションクライアント名	クライアント ID			
<input checked="" type="radio"/> lab2-dev-app	7o4u9s1m65cm459norvb33s42q			

ホストされた UI [情報](#)
このアプリケーション用のホストされた UI を設定します。

ホストされた UI ステータス
 使用可能

許可されているコールバック URL
https://d2khfmka5s3rjs.cloudfront.net/news/new

許可されているサインアウト URL
-

ID プロバイダー
Cognito ユーザープールディレクトリ

OAuth 付与タイプ
確実的な性質

OpenID Connect のスコープ
email
openid
phone

カスタムスコープ
-

[編集](#) [ホストされた UI を表示](#)

→初期パスワードを設定するとユーザーが使用可能になる

25.APIゲートウェイでCognitoを有効化する

[API Gateway](#) > [API](#) > [lab2-api\(jaechkbp2\)](#) > [オーソライザー](#) > オーソライザーを作成

オーソライザーを作成 [情報](#)

オーソライザーの詳細

オーソライザ名
lab2-dev-cognito

オーソライザーのタイプ [情報](#)
選択すると、いずれかの Lambda 関数または Cognito ユーザープールを使用して API 呼び出しを認可します。

Lambda
 Cognito

Cognito ユーザープール
API に対するリクエストを認証する Cognito ユーザープールを選択します。

ap-northeast-1 [▼](#) [×](#)

トークンのソース
認可トークンを含むヘッダーを入力します。
[Authorization](#)

トークンの検証 - オプション
トークンを検証するには、正規表現を入力します。

[キャンセル](#) [オーソライザーを作成](#)

POSTメソッドを編集し、認可欄にCognitoを選択する

リソースを作成

/news - POST - メソッドの実行

ARN
arn:aws:execute-api:ap-northeast-1:157094121738:jaechkkbp2/POST/news

リソース ID
t5qfhn

ドキュメントを更新 [削除](#)

クライアント → メソッドリクエスト → 結合リクエスト → Lambda
← メソッドレスポンス ← 結合レスポンスプロキシ構造

[メソッドリクエスト](#) [結合リクエスト](#) [結合レスポンス](#) [メソッドレスpons](#) [テスト](#)

[メソッドリクエストの設定](#)

認可
lab2-cognito
リクエスト/リデーター
なし

API キーは必須です
False

SDK オペレーション名
メソッドとパスに基づいて生成されました

リクエストパス (0)
名前 [名前](#) [必讀](#) [キャッシュ](#)

リクエストパスなし
リクエスト/パスが定義されていません

URL クエリ文字列/パラメータ (0)
名前 [名前](#) [必讀](#) [キャッシュ](#)

Cognitoのアプリケーションの統合タブからURLにアクセスして確認する

26.s3のライフサイクルルール設定

ライフサイクルルールを作成する 情報

ライフサイクルルールの設定

ライフサイクルルール名

lab2-rule

最大 255 文字

ルールスコープを選択

- 1つ以上のフィルターを使用してこのルールのスコープを制限する
 パケット内のすべてのオブジェクトに適用



パケット内のすべてのオブジェクトに適用

ルールを特定のオブジェクトに適用する場合は、フィルターを使用してそれらのオブジェクトを識別する必要があります。[Limit the scope of this rule using one or more filters] を選択します。[詳細](#)

- このルールがパケット内のすべてのオブジェクトに適用されることを了承します。

ライフサイクルルールのアクション

このルールで実行するアクションを選択します。リクエストごとの料金が適用されます。[詳細](#) または [Amazon S3 の料金](#) を参照してください

- オブジェクトの最新バージョンをストレージクラス間で移動
 オブジェクトの非現行バージョンをストレージクラス間で移動
 オブジェクトの現行バージョンを有効期限切れににする
 オブジェクトの非現行バージョンを完全に削除
 有効期限切れのオブジェクト削除マークまたは不完全なマルチパートアップロードを削除
オブジェクトタグまたはオブジェクトサイズでフィルタリングする場合、これらのアクションはサポートされません。

オブジェクトの現行バージョンをストレージクラス間で移行する

移行を選択して、ユースケースシナリオとパフォーマンスアクセス要件に基づいて、ストレージクラス間でオブジェクトの現行バージョンを移動します。これらの移行は、オブジェクトが作成された場合に開始され、連続して適用されます。[詳細はこちる](#)

ストレージクラスの移行を選択

オブジェクト作成後の日数

Glacier Deep Archive

日数

削除

有效的な整数値が必要です。

移行を追加する



小さなオブジェクトを Glacier Flexible Retrieval (旧 Glacier) または Glacier Deep Archive に移行すると、

オブジェクトあたりのコストが発生します

S3 Glacier Flexible Retrieval (旧 Glacier)、または S3 Glacier Deep Archive に移行するオブジェクトごとに料金が発生します。また、オブジェクトの管理メタデータを収容するために、各オブジェクトには一定量のストレージが追加されるため、ストレージコストが増加します。移行するオブジェクトの数を（フレイティクス、タグ、バージョンにより）制限するか、移行前にオブジェクトを集約することで、これらのコストを削減できます。[Glacier Flexible Retrieval \(旧 Glacier\) のコストに関する考慮事項](#) の詳細と一覧表についての説明は、[Amazon S3 の料金表](#) ページで、リクエストとデータ取り出しのタブを参照してください。

- 私は、このライフサイクルルールで小さなオブジェクトを移行する際、オブジェクトごとに1回限りのライフサイクルリクエストコストが発生することを認識しています。

移行と有効期限切れのアクションを確認

最新バージョンのアクション

0 日

- アップロードされたオブジェクト

↓

-- 日

- オブジェクトは Glacier Deep Archive に移動します

非現行バージョンのアクション

0 日

アクションが定義されていません。

キャンセル

ルールの作成

必要ポリシー

□ ポリシー名	▲ タイプ	▼ アタッチされたエンティティ
□ AWSLambdaBasicExecutionRole	AWS 管理	3
□ AWSLambdaSQSQueueExecutionRole	AWS 管理	1
□ lab2-dynamodb	カスタマーインライン	0

lab2-dynamodb	JSON をコピー	編集
---------------	---------------------------	--------------------

```

1- [{}]
2-   "Version": "2012-10-17",
3-   "Statement": [
4-     {
5-       "Sid": "VisualEditor0",
6-       "Effect": "Allow",
7-       "Action": [
8-         "dynamodb:PutItem",
9-         "dynamodb:GetItem",
10-        "dynamodb:UpdateItem"
11-      ],
12-      "Resource": [
13-        "arn:aws:dynamodb:ap-northeast-1:157094121738:table/lab2-news",
14-        "arn:aws:dynamodb:ap-northeast-1:157094121738:table/lab2-sequences"
15-      ]
16-    }
17-  ]
18- ]

```


□ lab2-s3	カスタマーインライン	0
---------------------------	------------	---

lab2-s3	JSON をコピー	編集
---------	---------------------------	--------------------

```

1- [{}]
2-   "Version": "2012-10-17",
3-   "Statement": [
4-     {
5-       "Sid": "VisualEditor0",
6-       "Effect": "Allow",
7-       "Action": [
8-         "s3:PutObject",
9-         "s3:GetObject"
10-      ],
11-      "Resource": "*"
12-    }
13-  ]
14- ]

```