

## RDS Proxy / Lambda ワークショップ

2021/09/29

シニアエバンジェリスト

亀田治伸

### [Aurora の起動]

まずは、Lambda から RDS Proxy を経由して呼び出される RDS インスタンスの Aurora を起動します。2021/09 現在 RDS Proxy は Aurora に加えて、MySQL, Postgres RDS をサポートしています。

#### 1. マネージメントコンソールの RDS 画面に移動します



#### 2. [データベースの作成]をおします

デフォルトで Aurora の MySQL 互換が選択されていますのでそのままにします。

#### 3. [テンプレート]で[開発/テスト]を選択し 1 台構成を選びます



#### 4. [設定]で任意のパスワードを指定します。このサンプルでは[password]を使います

## 設定

### DB クラスター識別子 [情報](#)

DB クラスターの名前を入力します。この名前は、AWS アカウントが現在の AWS リージョンで所有しているすべての DB クラスターにおいて一意である必要があります。

database-1

DB クラスター識別子は大文字と小文字を区別しませんが、すべて小文字で保存されます (例: "mydbcluster")。制約として、使用できるのは 1~60 文字以内で英数字またはハイフンのみです。1 字目は英文字でなければなりません。また、ハイフンを連続で 2 つ使ったり、最後の文字をハイフンにしたりすることはできません。

### ▼ 認証情報の設定

#### マスターユーザー名 [情報](#)

DB インスタンスのマスターユーザーのログイン ID を入力します。

admin

1~16 文字の英数字。1 字目は文字である必要があります

#### ☐ パスワードの自動生成

Amazon RDS がパスワードを生成するか、お客様がご自身でパスワードを指定することができます

#### マスターパスワード [情報](#)

\*\*\*\*\*

制約事項: 表示可能な ASCII 文字で 8 文字以上で入力してください次の文字を含めることはできません: / (スラッシュ)、' (単一引用符)、" (二重引用符)、および @ (アットマーク)。

#### パスワードを確認 [情報](#)

\*\*\*\*\*

5. [DB インスタンスクラス]で[バースト可能クラス]を選び一番小さい t3.small を選びます

## DB インスタンスクラス

### DB インスタンスクラス [情報](#)

☐ メモリ最適化クラス (r クラスを含む)

☒ バースト可能クラス (t クラスを含む)

db.t3.small

2 vCPUs 2 GiB RAM ネットワーク: 2,085 Mbps

☐ 以前の世代のクラスを含める

6. [可用性と耐久性]はデフォルトのままシングル構成にします
7. [接続]はデフォルト VPC を選びます

**接続**

Virtual Private Cloud (VPC) [情報](#)  
この DB クラスターの仮想ネットワーク環境を定義する VPC。

Default VPC (vpc-7653bb1f) ▼

対応する DB サブネットグループがある VPC のみが表示されます。

**データベースの作成後に、VPC を変更することはできません。**

サブネットグループ [情報](#)  
選択した VPC で DB インスタンスが使用できるサブネットと IP 範囲を定義する DB サブネットグループ。

デフォルト ▼

8. [VPC セキュリティグループ]は[default]を選びます

VPC セキュリティグループ  
データベースへのアクセスを許可する VPC セキュリティグループを選択します。セキュリティグループのルールで適切な着信トラフィックが許可されていることを確認します。

☒ 既存の選択  
既存の VPC セキュリティグループの選択

☐ 新規作成  
新しい VPC セキュリティグループの作成

既存の VPC セキュリティグループ

VPC セキュリティグループを選択します ▼

default ✕

9. [データベースの作成]をおします
10. 10 分から 15 分程度まちます

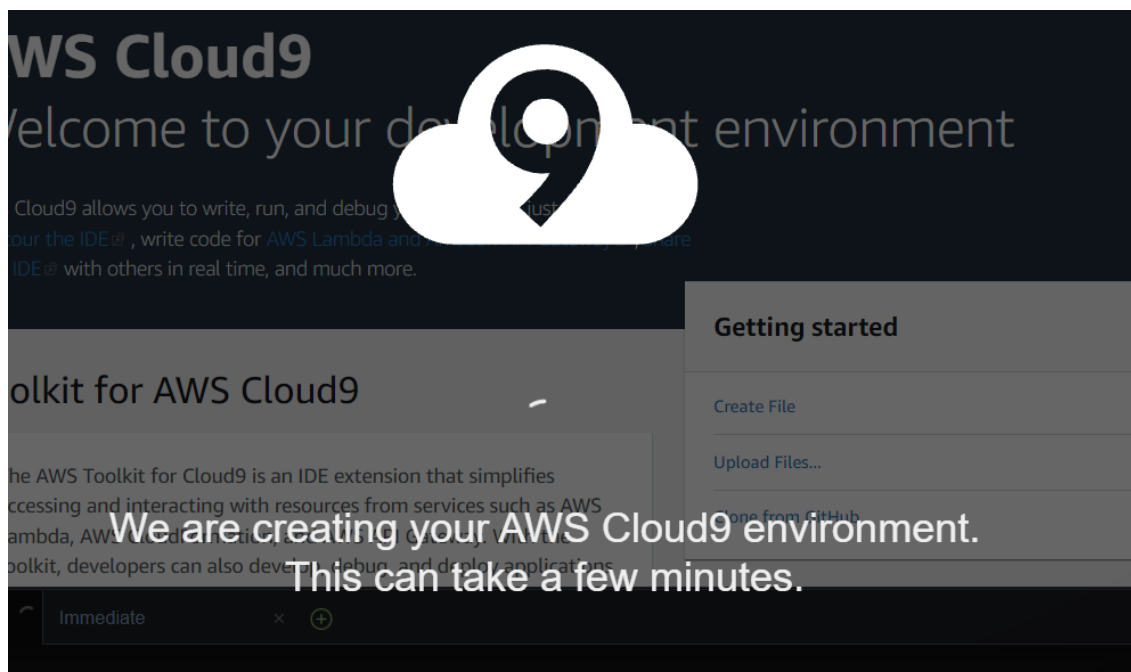
データベース						
		<input checked="" type="radio"/> グループリソース		変更	アクション ▼	S3 から復元
Q データベースのフィルタリング						
< 1 > ⚙						
DB 識別子	ロール	エンジン	リージョンと AZ	サイズ	ステータス	
database-1	リージョン別クラスター	Aurora MySQL	us-east-2	1 インスタンス	🕒 作成中	
database-1-instance-1	リーダーインスタンス	Aurora MySQL	-	db.t3.small	🕒 作成中	

## [Cloud9 の構築]

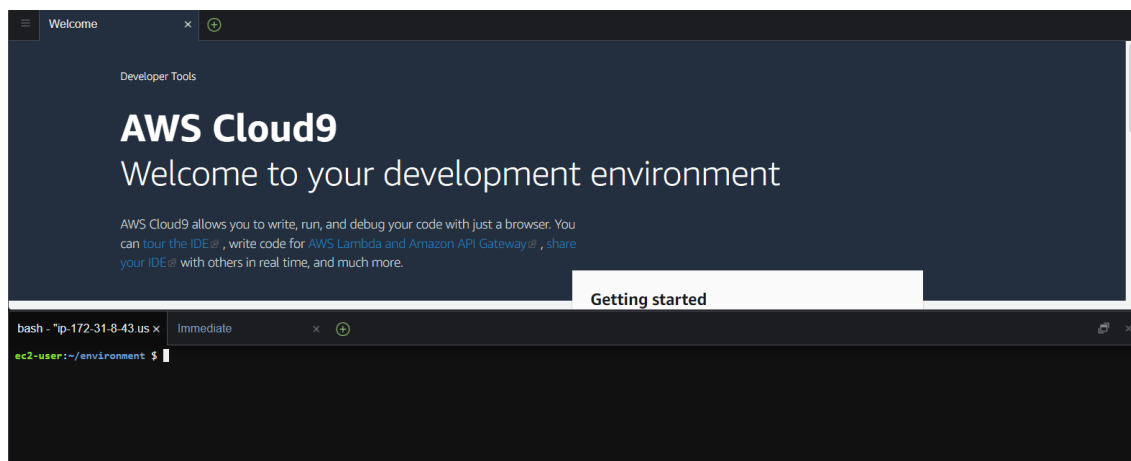
構築された MySQL に接続を行いテスト用データを書き込む Cloud9 を構築します

11. マネーコメントコンソールで Cloud9 の画面にアクセスします（ブラウザの別タブを開いた方が便利です）
12. [Create environment] ボタンをおします

13. [Name]に適当な名前をつけ、[Next Step]をおします
14. 全てデフォルトのまま[Next Step]をおします。この際、[Network settings(advanced)]で、RDS と同じデフォルト VPC が選ばれていることを確認してください。
15. [Create environment]をおします
16. 作成中が表示されますのでしばらく待ちます



17. 以下のようにコンソールで操作可能となれば構築成功です



[RDS Aurora のネットワーク解放]

構築した Cloud9 や後ほど構築する Lambda 関数からの接続を可能とするため、セキュリティグループを書き換えます

18. RDS のマネージメントコンソールに戻り Aurora が起動済であることを確認します

データベース グループリソース 変更 アクション S3 から復元 データベースの作成

データベースのフィルタリング

DB 識別子	ロール	エンジン	リージョンと AZ	サイズ	ステータス
database-1	リージョン別クラスター	Aurora MySQL	us-east-2	1 インスタンス	利用可能
database-1-instance-1	ライターインスタンス	Aurora MySQL	us-east-2b	db.t3.small	利用可能

19. ライターをクリックしてセキュリティグループを特定します。(注意、クラスターはライターに対する CNAME の存在ですので、設定変更は実体であるライターに対して行います)

関連

データベースのフィルタリング

DB 識別子	ロール	エンジン	リージョンと AZ	サイズ	ステータス
database-1	リージョン別クラスター	Aurora MySQL	us-east-2	1 インスタンス	利用可能
database-1-instance-1	ライターインスタンス	Aurora MySQL	us-east-2b	db.t3.small	利用可能

接続とセキュリティ | モニタリング | ログとイベント | 設定 | メンテナンス | タグ

接続とセキュリティ

<p>エンドポイントとポート</p> <p>エンドポイント</p> <p>database-1-instance-1.cagbxmtcmq2.us-east-2.amazonaws.com</p>	<p>ネットワーク</p> <p>アベイラビリティゾーン</p> <p>us-east-2b</p>	<p>セキュリティ</p> <p>VPC セキュリティグループ</p> <p>default (sg-2156a648) (アクティブ)</p>
--	--	--

20. セキュリティグループを右クリックして新しいタブでセキュリティグループの設定画面を開き、[インバウンドルール]のタブをクリックします

詳細 | **インバウンドルール** | アウトバウンドルール | タグ

You can now check network connectivity with Reachability Analyzer Run Reachability Analyzer

Inbound rules (1/1) Manage tags Edit inbound rules

Filter security group rules

21. [Edit inbound rules]をおして[削除]ボタンで今あるルールを全て削除します

インバウンドルール 情報

このセキュリティグループにはインバウンドルールがありません。

ルールを追加

22. [ルールの追加]をおして、以下のように VPC 内のすべての通信を通すようなルールを設定し、[ルールを保存]をおします

[Cloud9 からの接続テスト]

Cloud9 からまずは Aurora に対して接続テストを行ってみます。

23. Aurora のマネージメントコンソールで、リージョン別クラスターをクリックして[ライターインスタンス]のエンドポイントをコピーします

24. Cloud9 コンソールから以下のコマンドを実行します。<cluster endpoint>は上記のものに置換してください

```
mysql -u admin -h <cluster endpoint> -p
```

25. パスワードを入力して以下が表示されたらログインが成功しています

```
ec2-user:~/environment $ mysql -u admin -h database-1.cluster-cagbxmtcmq2.us-east-2.rds.amazonaws.com -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 17
Server version: 5.7.12 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> show databases;
```

26. 試しに[show databases;]を実行してみてください

[Secrets Manager]の設定

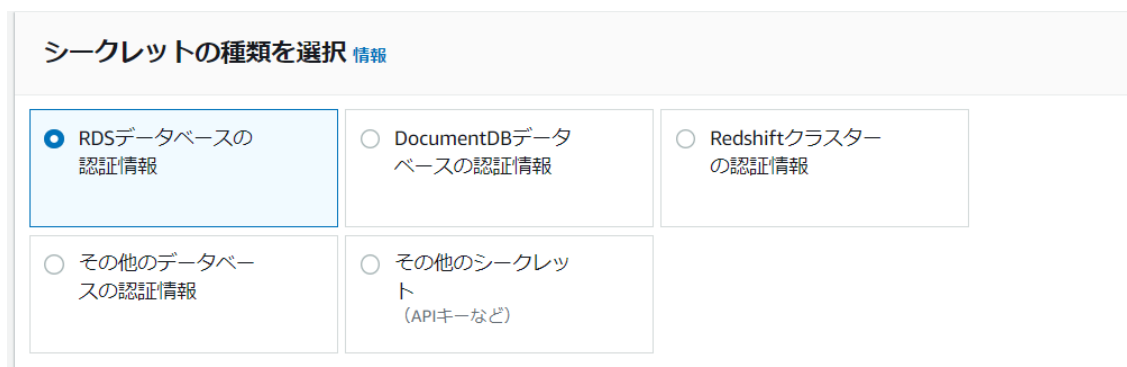
RDS Proxy は RDS への接続に用いるパスワードは独自ではなく Secrets Manager と連携して管理します。暗号化やローテーションも Secrets Manager で管理されます。

27. マネージメントコンソールで Secrets Manager にアクセスします（ブラウザの別タブで開くことをお勧めします）

28. [新しいシークレットを保存する]をおします



29. [シークレットの種類を選択]から[RDS データベースの認証情報]を選びます



30. ユーザー名、パスワードに Aurora へのログイン情報を入力します

31. 先程作成した Aurora が表示されていますので、そちらを選択し、[次へ]をおします



32. [シークレットの名前]に適当な値を入力し、[次へ]をおします

33. 次のシークレットのローテーション設定画面ではそのままデフォルトの状態です[次へ]をおします

34. 最後の画面で[保存]をおします。以下のようにシークレットが保存されます

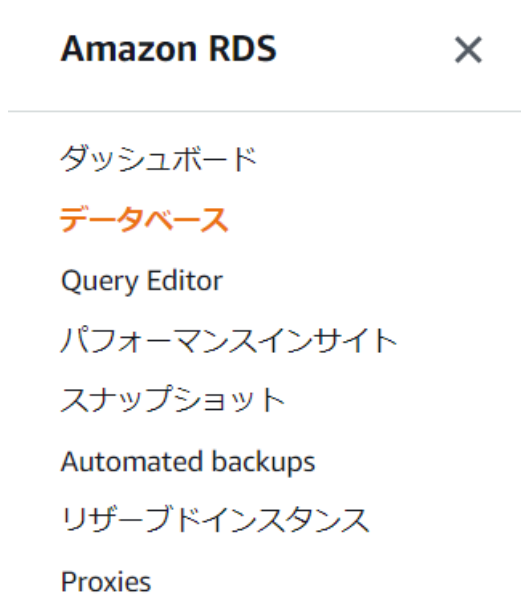


[RDS Proxy の起動]

上記で作成した Secrets Manager に保存されている ID/パスワードを用いて、作成済

Aurora へ接続するための RDS Proxy を起動します

35. RDS のマネージメントコンソール、左ペインから [Proxies] をおします



36. [プロキシを作成] をおします

37. [プロキシ識別子] に任意の名前をいれます

## プロキシを作成

### プロキシ設定

プロキシにより、アプリケーションのスケラビリティが向上し、データベースの障害に対して透過性が高くなり、安全性が向上します。

### プロキシ識別子

プロキシの名前を入力します。この名前は、AWS アカウントが現在の AWS リージョンで所有する、すべてのプロキシ間で一意である必要があります。

制約として、使用できるのは 1~60 文字以内で英数字またはハイフンのみです。1 文字目は英文字でなければなりません。また、ハイフンを連続で 2 つ使ったり、最後の文字をハイフンにしたりすることはできません。

### エンジンの互換性 [情報](#)

MySQL ▼

☐ Transport Layer Security が必要  
Transport Layer Security (TLS) は、ネットワークを介した通信を保護する暗号化プロトコルです。

38. [エンジンの互換性] は [MySQL] を選びます

### エンジンの互換性 [情報](#)

MySQL ▼

39. [データベース] に Aurora クラスターをドロップダウンから選びます



### ターゲットグループの設定

ターゲットグループは、プロキシが接続できるデータベースのコレクションです。現在、各ターゲットグループを 1 つの RDS DB インスタンスまたは Aurora DB クラスターに関連付けることができます。

**データベース**  
 プロキシに関連付ける RDS DB インスタンスまたは Aurora DB クラスターを選択します。

database-1 ▼

**接続プールの最大接続数** [情報](#)  
 データベースの最大接続制限に対する割合として、許可される最大接続数を指定します。

100 パーセント

データベースの最大接続制限に対する割合として、許可される最大接続数を指定します。たとえば、最大接続数を 5,000 接続に設定している場合、50% を指定すると、プロキシはデータベースに対して最大 2,500 接続を作成できます。

**リーダーエンドポイントを含める** [情報](#)

☒ リーダーエンドポイントを追加  
 このプロキシの作成時にリーダーエンドポイントを含めるかどうかを選択します。

▶ 追加のターゲットグループの設定

40. [接続]の項目で先程作成した Secrets Manager の値をドロップダウンから選びます

### 接続

**Secrets Manager シークレット** [情報](#)  
 プロキシが使用できるデータベースユーザーアカウントの認証情報を表す Secrets Manager シークレットを作成または選択します。

1 つ以上のシークレットを選択 ▼

rdsproxy0929 ✕

[新しいシークレットを作成する](#)

**IAM ロール**  
 プロキシが AWS Secrets Manager シークレットにアクセスする際に使用する IAM ロールを、作成または選択します。

IAM ロールを作成 ▼

**IAM 認証**  
 IAM 認証を使用して、データベース認証情報の指定に加えて、プロキシに接続できます。プロキシで IAM 認証を使用する方法を選択します。この選択は、上記で選択したすべてのシークレットに適用されます。

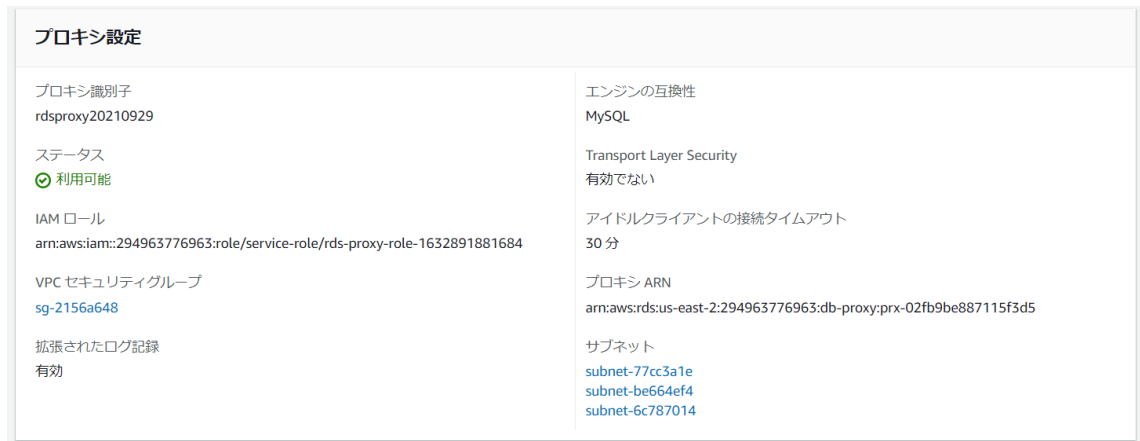
無効 ▼

41. [拡張されたログ記録を有効にする]にチェックをつけ、[プロキシを作成]をおします

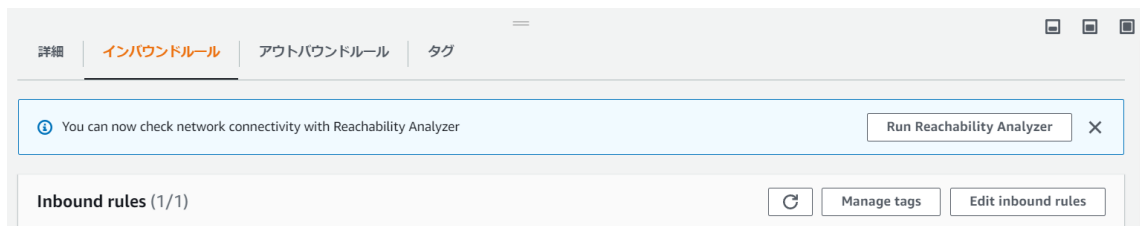
42. 作成中となりますので、10 分程度待ちます

プロキシ (1)			アクション ▼	プロキシを作成
Q プロキシのフィルタリング			< 1 >	⚙️
プロキシ識別子 ▲	ステータス ▼	エンジンの互換性 ▼		
<input type="radio"/> rdsproxy20210929	⊖ 作成中	MySQL		

43. 作成されたら、プロキシ識別子で作成済プロキシのリンクをクリックして、[VPC セキュリティグループ]を右クリックで開きます



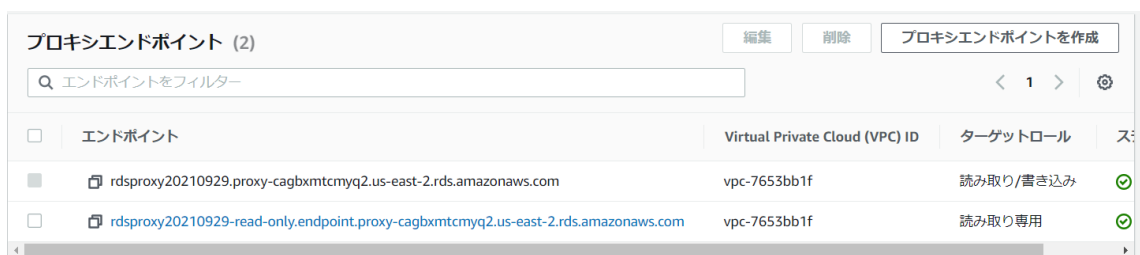
44. [インバウンドルール]のタブから[Edit inbound rules]をおします



45. [削除]をおして今あるルールを全て削除します
46. [ルールの追加]をおして以下のように VPC 内部の通信を全てとおして、[ルールを保存]をおします



47. 再度 RDS Proxy のマネージメントコンソールから、[エンドポイント]をコピーします (読み取り/書き込みの方です)



48. Cloud9 から以下のコマンドを実行します

```
mysql -u admin -h <proxyendpoint> -p
```

49. 以下が表示されれば完了です

```
ec2-user:~/environment $ mysql -u admin -h rdsproxy20210929.proxy-cagbxmtcmq2.us-east-2.rds.amazonaws.com -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 3642024877
Server version: 5.7.12 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> █
```

50. 以下のコマンドを 1 行ずつ実行します

```
show databases;
create database test;
create table test.sample (id int, value varchar(10));
INSERT INTO test.sample VALUES (1,"test");
select * from test.sample;
```

#### [Lambda 関数の作成]

今までの手順で、VPC 内部から RDS Proxy 経由で RDS へアクセスできるようになりました。これから VPC 内部へアクセス可能な Lambda 関数を作成していきます

51. Lambda のマネージメントコンソールを開きます

52. [関数の作成]をおします

53. [一から作成]を選んで、[関数名]に適当な名前を入れます。ランタイムは[Node.js14]を選びます

基本的な情報
<div>関数名</div> <div>関数の目的を名前として入力します。</div> <div><input type="text" value="20210929Lambda"/></div> <div>半角英数字、ハイフン、アンダースコアのみを使用でき、スペースは使用できません。</div>
<div>ランタイム <a href="#">情報</a></div> <div>関数の記述に使用する言語を選択します。コンソールコードエディタは Node.js、Python、および Ruby のみをサポートすることに注意してください。</div> <div><div>Node.js 14.x</div><div>▼</div></div>

54. [詳細設定]からデフォルト VPC を選んで、すべてのサブネットを登録します

▼ 詳細設定

コード署名

コード署名設定 - オプション [情報](#)  
 コード署名を有効にするには、署名検証ポリシーとコードの署名を許可されている署名プロファイルを定義する設定を選択します。

コード署名設定 ARN を選択

ネットワーク

Lambda 関数にネットワークアクセスを提供するには、Virtual Private Cloud (VPC)、VPC サブネット、および VPC セキュリティグループを指定します。ユーザーアクセス許可で VPC を設定する必要がある場合を除き、VPC 設定はオプションです。

VPC - オプション [情報](#)  
 関数がアクセスする VPC を選択します。

vpc-7653bb1f (172.31.0.0/16)

サブネット

Lambda が VPC 設定をセットアップするために使用する VPC サブネットを選択します。

サブネットを選択

subnet-77cc3a1e (172.31.0.0/20) us-east-2a × subnet-be664ef4 (172.31.32.0/20) us-east-2c × subnet-6c787014 (172.31.16.0/20) us-east-2b ×

55. セキュリティグループはデフォルトを選んで、以下の通り VPC 内部のすべての通信が通るようになっていることを確認してください

セキュリティグループ

Lambda が VPC 設定をセットアップするために使用する VPC セキュリティグループを選択します。選択したセキュリティグループのインバウンドとアウトバウンドのルールについて、下表に示します。

セキュリティグループを選択

sg-2156a648 (default) ×  
 default VPC security group

インバウンドルール    アウトバウンドルール

セキュリティグループ ID	プロトコル	ポート	ソース
sg-2156a648	All	All	0.0.0.0/0

56. [関数の作成]をおします
57. VPC 内部へのアクセスインターフェースを作成するため、いつもより Lambda 関数の作成に時間がかかりますが待ちます。以下のように緑のバーが表示されれば作成完了です

関数 20210929Lambda を作成中です。

関数 20210929Lambda を正常に作成しました。関数コードおよび設定を変更できるようになりました。テストイベントで関数を呼び出すには、[テスト] をクリックします。

58. 画面中ほどにあるタブで[設定]をクリックし、左ペインから[VPC]をクリックして下さい

コード    テスト    モニタリング    **設定**    エイリアス    バージョン

59. 以下の通り VPC 内部の通信が通るようになっていることを確認します

VPC 情報

編集

VPC

vpc-7653bb1f (172.31.0.0/16) | デフォルト

サブネット

- subnet-77cc3a1e (172.31.0.0/20) | us-east-2a
- subnet-be664ef4 (172.31.32.0/20) | us-east-2c
- subnet-6c787014 (172.31.16.0/20) | us-east-2b

セキュリティグループ

- sg-2156a648 (default)

インバウンドルール

アウトバウンドルール

セキュリティグループ ID	プロトコル	ポート	ソース
sg-2156a648	All	All	0.0.0.0/0

60. 左ペインから[データベースプロキシ]をクリックします。
61. [データベースプロキシの追加]をおします
62. [既存のデータベースプロキシの選択]を選び、作成済の RDS Proxy を選び、[追加]をおします

データベースプロキシ 情報

☐ 新しいデータベースプロキシの作成
 ☒ 既存のデータベースプロキシの選択

既存のプロキシ

rdsproxy20210929

▼

🔄

キャンセル

追加

63. 以下のように表示されれば設定が完了です

コード

テスト

モニタリング

設定

エイリアス

バージョン

一般設定

トリガー

アクセス権限

送信先

データベースプロキシ 情報

データベースプロキシの追加

プロキシ識別子	ステータス	エンジンの互換性	IAM 認証
rdsproxy20210929	Available	MySQL	DISABLED

64. タブで[コード]を選びます

コード

テスト

モニタリング

設定

エイリアス

バージョン

コードソース 情報

アップロード元 ▼

File Edit Find View Go Tools Window

Test Deploy Changes not deployed

Go to Anything (Ctrl-P)

index.js

Environment

20210929Lambda

index.js

```

1 exports.handler = async (event) => {
2   // TODO implement
3   const response = {
4     statusCode: 200,
5     body: JSON.stringify('Hello from Lambda!'),
6   };
7   return response;
8 };
9

```

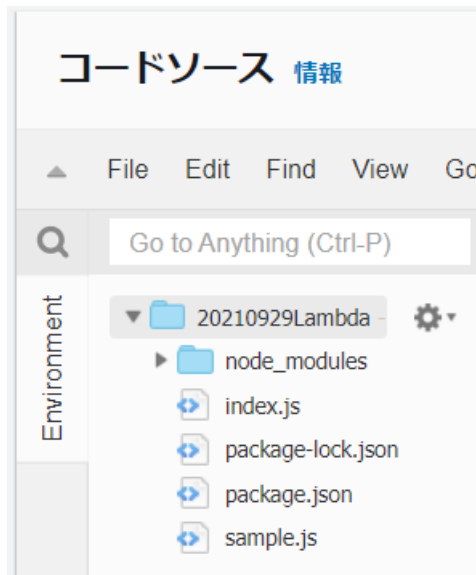
65. [アップロード元]ボタンから[.zip ファイル]を選びます



66. Github にある、[lambdamysql.zip]を適当な個所に保存しアップロードします



67. 以下のようにパッケージがインポートされます



68. Index.js を開いて、[host]部分をまずは RDS のエンドポイントに設定します

69. [Deploy]をおして、[Test]をおします

70. イベント名に適当な名前を付けて[作成]をおします

71. 再度[Test]をおします

72. 以下の通り SQL の実行結果が出力されたら成功です

<b>Test Event Name</b>	
test	
<b>Response</b>	
null	
<b>Function Logs</b>	
START RequestId: 611bc801-cff3-45c6-871e-7ef7883f7f80 Version: \$LATEST	
2021-09-29T06:21:02.342Z	611bc801-cff3-45c6-871e-7ef7883f7f80 INFO [ RowDataPacket { id: 1, value: 'test' } ]
2021-09-29T06:21:02.399Z	611bc801-cff3-45c6-871e-7ef7883f7f80 INFO [
FieldPacket {	
catalog: 'def',	
db: 'test',	
table: 'sample',	
orgTable: 'sample',	
name: 'id',	
orgName: 'id',	
charsetNr: 63,	
length: 11,	
type: 3,	
flags: 0,	
decimals: 0,	
default: undefined,	
zeroFill: false,	
protocol41: true	
},	
FieldPacket {	

73. 再度、index.js の [host] を RDS Proxy のエンドポイントに置き換えて、  
[Deploy],[Test]の順番におします
74. 以下の通り Select クエリの実行結果が表示されれば成功です

<b>Test Event Name</b>	
test	
<b>Response</b>	
null	
<b>Function Logs</b>	
START RequestId: 79c17f56-507d-4f6a-9fe5-a53462649011 Version: \$LATEST	
2021-09-29T06:22:33.884Z	79c17f56-507d-4f6a-9fe5-a53462649011 INFO [ RowDataPacket { id: 1, value: 'test' } ]
2021-09-29T06:22:33.922Z	79c17f56-507d-4f6a-9fe5-a53462649011 INFO [
FieldPacket {	
catalog: 'def',	
db: 'test',	
table: 'sample',	
orgTable: 'sample',	
name: 'id',	
orgName: 'id',	
charsetNr: 63,	
length: 11,	
type: 3,	
flags: 0,	
decimals: 0,	
default: undefined,	
zeroFill: false,	
protocol41: true	
},	
FieldPacket {	

おつかれさまでした！  
以下を削除してください

- RDS Proxy
- RDS
- Lambda 関数
- IAM ロール (Lambda 用、RDS Proxy 用)