

はじめに：Aurora Serverless はアプリケーションのニーズに基づいてデータベースの容量を拡大または縮小することのできる Amazon Aurora 用のオンデマンドの自動スケーリング設定です。Aurora PostgreSQL と MySQL それぞれ以下の環境に対応しています。

Aurora MySQL 3.02.0 or higher. (MySQL 8.0.)

Aurora PostgreSQL 13.6 or higher.

1 つの Aurora クラスターの中にインスタンス型とサーバレス型を混在させることができるのが大きな特徴です。

まず 1 台構成の MySQL クラスターを Aurora Serverless を用いずに起動させます。

1. RDS のマネージメントコンソールに移動します
2. [データベースの作成]をおします
3. [フィルターの表示]をクリックし、[Serverless v2 をサポートするバージョンを表示]をオンにします

▼ フィルターの非表示

- ☐ グローバルデータベース機能をサポートするバージョンを表示
1 つの Amazon Aurora データベースが複数の AWS リージョンにまたがることができます。
- ☐ 並列クエリ機能をサポートするバージョンを表示
Aurora ストレージレイヤーまで処理をプッシュして、分析クエリのパフォーマンスを向上させます。
- ☒ Serverless v2 をサポートするバージョンを表示
最も負荷の高いワークロードにもインスタンススケーリングを提供します。

利用可能なバージョン (1/60) [情報](#)

Aurora MySQL 3.02.0 (compatible with MySQL 8.0.23) ▼

4. 適当なバージョンを選びます(2022/06/01 現在、バージョンは 1 つのみになります)
5. [テンプレート]で[開発/テスト]を選びます

テンプレート

お客様のユースケースに合わせてサンプルテンプレートを選択します。

☐ 本番稼働用

高い可用性と、高速で安定したパフォーマンスのために、デフォルト値を使用します。

☒ 開発/テスト

このインスタンスは本番稼働環境ではない開発で使用します。

6. 適当なパスワードを入力します

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](#)

.....

7. [インスタンスの設定]で[バースト可能クラス]を選びます

インスタンスの設定

以下の DB インスタンスの設定オプションは、上記で選択したエンジンでサポートされているものに制限されています。

DB インスタンスクラス [情報](#)

☐ サーバーレス

☐ 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.medium

2 vCPUs 4 GiB RAM ネットワーク: 2,085 Mbps

☐ 以前の世代のクラスを含める

(インスタンスはデフォルトで問題ありません)

- VPC はデフォルト VPC をこのハンズオンでは用いますが、別の VPC を選択しても動作します
- セキュリティグループに適当な名前を付けます。あとで使いますので名前はメモを取っておいてください。

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

新しい VPC セキュリティグループ名

10. [データベースの作成]をおします
11. 起動中の画面に切り替わりますので、数分間待ちます。起動されました！と表示してもクラスター全体が起動していないケースがありますので以下のステータスを確認します

エンジン ▼	リージョンと AZ ▼	サイズ ▼	ステータス ▼
Aurora MySQL	ap-northeast-1	1 インスタンス	✔ 利用可能
Aurora MySQL	-	db.t3.medium	🕒 作成中

12. 2 つとも利用可能になったら[database-1]をクリックし、2 つのエンドポイントをメモしておきます

```

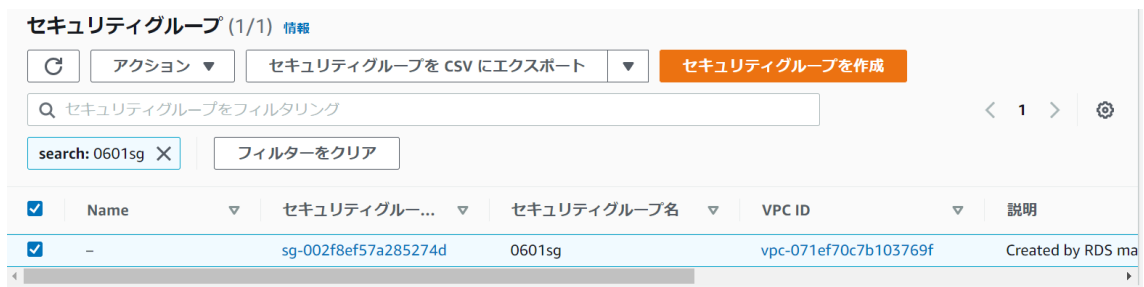
writer
database-1.cluster-cezfqx1ilrp8.ap-northeast-1.rds.amazonaws.com

reader
database-1.cluster-ro-cezfqx1ilrp8.ap-northeast-1.rds.amazonaws.com

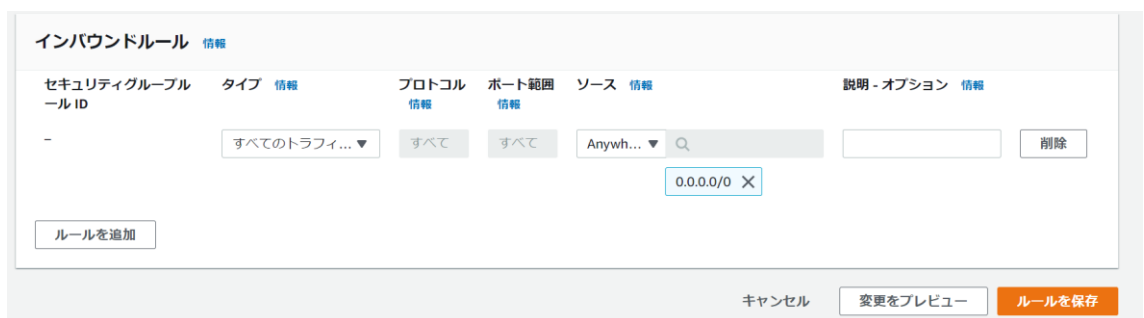
```

Cloud9 を起動し、MySQL へログインし新しいデータベースを作ります

13. Cloud9 のマネージメントコンソールを新しいタブで開きます
14. [create environment]をおします
15. 適当な名前を入力し[Next step]をおします
16. 全てデフォルトのまま[Next step]をおします。(上記手順で RDS をデフォルト VPC 以外に起動した方は、同じ VPC を選んでください)
17. [Create environment]をおします
18. しばらく待つとコンソールへのアクセスが可能となります。
19. さらにブラウザの別タブを開き、VPC のマネージメントコンソールへ移動します
20. 左ペインから[セキュリティグループ]をクリックします
21. 先程メモしておいたセキュリティグループ名で検索をかけ、特定しクリックします

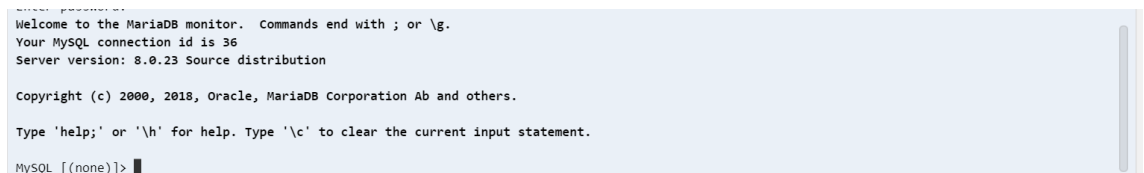


22. 画面下部、インバウンドルールタブをクリックし、[インバウンドのルールを編集]をおします
23. 今あるルールを削除し、[ルールを追加]を押し以下のように設定します

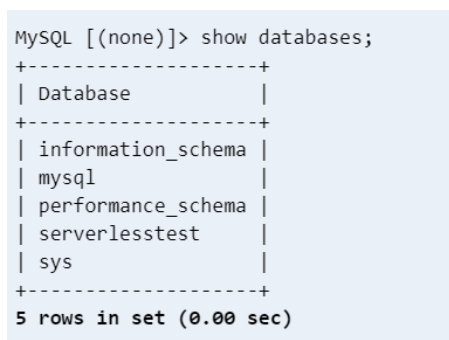


24. [ルールの保存]をおします。これにより VPC 内部のすべての通信が RDS へ届くようになったため、Cloud9 からの通信が届くようになります。
25. Cloud9 のコンソールに戻り以下のコマンドを実行します。<writer endpoint>は先程コピーしたエンドポイントに置換します


```
mysql -h <writer endpoint> -u admin -p
```
26. パスワードを入力します。以下の様になれば成功です



27. [create database serverlesstest;], [show databases;]を続けて実行します。以下のように新しいデータベースが作成されます



28. 先程と異なるリーダーエンドポイントに同様のコマンドでログインして[show databases;]を実行すると新しいデータベースが見えるようになっています

Aurora Serverless v2 のリードレプリカ追加

29. RDS のマネージメントコンソールに戻ります
30. クラスターを選び、[アクション]から[リーダーの追加]を選びます



31. [DB インスタンス識別子]に[serverless]と入力します

設定

Aurora レプリカのソース

database-1-instance-1 (DB クラスター: database-1)

DB インスタンス識別子

DB インスタンス識別子。これは DB インスタンスを識別する一意のキーです。このパラメータは小文字の文字列 (mydbinstance など) として保存されます。

serverless

32. [インスタンスの設定]で[Serverless v2]を選びます。ACU (Aurora Capacity Unit)はそのままで大丈夫です

インスタンスの設定

以下の DB インスタンスの設定オプションは、上記で選択したエンジンでサポートされているものに制限されています。

DB インスタンスクラス [情報](#)

- ☒ Serverless v2
- ☐ 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)

容量の範囲 [情報](#)

データベース容量は Aurora 容量ユニット (ACU) で測定されます。1 ACU は 2 GiB のメモリと、対応するコンピューティングとネットワークを提供します。

最小 ACU

(1 GiB)

0.5~128 (0.5 の増分)

最大 ACU

(256 GiB)

1~128 (0.5 の増分)

33. 拡張モニタリングを有効化し[詳細度]を 10 秒に設定します。(Aurora Serverless v2 は v1 と異なりインスタンス単位でメトリックスを収集可能です。また拡張モニタリングは微小ながらモニタリングで ACU 自体を消費することに注意して下さい)

モニタリング

☒ 拡張モニタリングの有効化

拡張モニタリングメトリックスを有効にすると、さまざまなプロセスやスレッドで CPU がどのように使用されているのかを確認したいときに便利です。

詳細度

34. [Add reader]を押し、しばらく待ちます。以下のように作成中になります


リージョンと AZ ▼	サイズ ▼	ステータス ▼	CPU	現在のアクティビティ
ap-northeast-1	2 インスタンス	✔ 利用可能	-	
ap-northeast-1d	db.t3.medium	✔ 利用可能	<div><div></div></div> 9.48%	<div><div></div></div> 2
-	Serverless v2 (0.5~128 ACU)	🕒 作成中	<div><div></div></div> 0.46%	<div><div></div></div> 0

35. 利用可能となった後、再度リーダーに Cloud9 コンソールからログインし、[show databases;]を実行すると、先程作成したデータベースが表示され、正しくクラスターに参加できていることがわかります。
36. Aurora マネージメントコンソール上で serverless をクリックします

関連

Q

フィルター条件 データベース



<div><div><div></div></div></div>	DB 識別子	▲	ロール	▼	エンジン	▼	リージョンと AZ
<div><div><div></div></div></div>	<div><div><div></div></div>database-1</div>		リージョン別クラスター		Aurora MySQL		ap-northeast-1
<div><div><div></div></div></div>	<div><div><div></div></div>database-1-instance-1</div>		ライターインスタンス		Aurora MySQL		ap-northeast-1d
<div><div><div></div></div></div>	<div><div><div></div></div>serverless</div>		リーダーインスタンス		Aurora MySQL		ap-northeast-1d

37. もう一つ別のエンドポイントが存在していることがわかります。これは先程リーダーとして追加した **serverless v2** インスタンスの実エンドポイントです。上の手順でログインしたリーダーエンドポイントは、この実エンドポイントに対する仮想エンドポイントです。[CNAME] この仕組みにより複数のリーダーが存在していたとしても、アプリケーションは常に一つのエンドポイント名でアクセスできるようになっています。先程と同様に実エンドポイントでもログインして[show databases;]を実行してみてください。ログインしている先は同じなので、同じ結果がでます。

フェイルオーバーによるライターの切り替え

38. RDS のマネージメントコンソールからライターインスタンスを選び、[アクション]から[フェイルオーバー]を選びます

database-1-instance-1

変更

アクション ▲

再起動

削除

フェイルオーバー

スナップショットの取得

関連

Q

フィルター条件 データベース

<div><div></div></div>	DB 識別子 ▲	ロール ▼	エンジン ▼	リージョンと AZ
<div><div></div></div>	database-1	リージョン別クラスター	Aurora MySQL	ap-northeast-1
<div><div></div></div>	database-1-instance-1	ライターインスタンス	Aurora MySQL	ap-northeast-1d
<div><div></div></div>	serverless	リーダーインスタンス	Aurora MySQL	ap-northeast-1d

39. 次の画面で[フェールオーバー]のボタンをおします
40. しばらく画面をリロードしながら待つと、以下の通りライターとリーダーが入れ替わっていることがわかります。このように Aurora Serverless v2 は既存インスタンスとの混在で利用することが可能です。

	DB 識別子	▲	ロール	▼	エンジン	▼	リージョンと AZ
<input type="radio"/>	<input type="checkbox"/> database-1		リージョン別クラスター		Aurora MySQL		ap-northeast-1
<input type="radio"/>	serverless		ライターインスタンス		Aurora MySQL		ap-northeast-1d
<input type="radio"/>	database-1-instance-1		リーダーインスタンス		Aurora MySQL		ap-northeast-1d

ライターとリーダーで区別するだけではなく以下のようにリーダーのなかで混在させることも可能です。

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

Q フィルター条件 データベース

< 1 >

	DB 識別子	▲	ロール	▼	エンジン	▼	リージョンと AZ	▼	サイズ
<div><div><div></div><div></div></div></div>	database-1		リージョン別クラスター		Aurora MySQL		ap-northeast-1		3 インスタンス
<div><div><div></div><div></div></div></div>	database-1-instance-1		ライターインスタンス		Aurora MySQL		ap-northeast-1d		db.t3.medium
<div><div><div></div><div></div></div></div>	serverless		リーダーインスタンス		Aurora MySQL		ap-northeast-1d		Serverless v2 (0.5~128 ACU)
<div><div><div></div><div></div></div></div>	t-instance-for-reader		リーダーインスタンス		Aurora MySQL		-		db.t3.medium

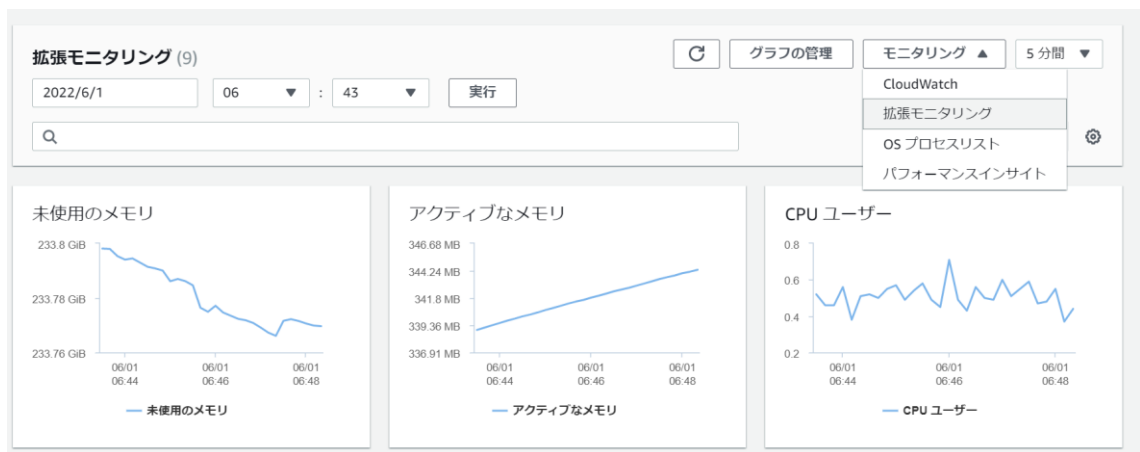
モニタリング

mysqlslap などに慣れている方は、再度 serverless をライターに昇格させ、コマンドを実行をして、少しでも DB に負荷をかけてみてください。。

41. Serverless をクリックして[モニタリング]のタブをクリックすると以下のように様々なグラフが取得可能です



42. 画面右上[モニタリング]をクリックし[拡張モニタリング]を選ぶとさらに様々なメトリックスが取得可能になります。



おつかれさまでした！削除は以下を行ってください

- ・リーダーの削除

RDS > データベース

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

Q フィルター条件 データベース

DB 識別子	ロール	エンジン	AZ	サイズ
database-1	リージョン別クラスター	Aurora MySQL	ap-northeast-1	3 インスタンス
database-1-instance-1	ライターインスタンス	Aurora MySQL	ap-northeast-1d	db.t3.medium
serverless	リーダーインスタンス	Aurora MySQL	ap-northeast-1d	Serverless v2 (0.5~128 ACU)
t-instance-for-reader	リーダーインスタンス	Aurora MySQL	ap-northeast-1c	db.t3.medium

- ・ライターの削除
- ・Cloud9 の削除
- ・サブネットグループ (RDS)
- ・CloudWatch ログ