

クラウドにおけるPgpool-IIの活用方法

2022/12/15

SRA OSS LLC
彭博 (ペンボ)

- クラウドにおけるPgpool-IIの稼働環境
- Pgpool-IIの設定
 - 冗長化構成
 - 仮想IPの設定方法

EC2

- オンプレミス環境の場合とほぼ同じ
- 柔軟に設定できる

マネージドリレーショナルデータベースサービス

- Amazon RDS for PostgreSQL/Amazon Aurora PostgreSQL
- セットアップや運用管理が容易
- スケーリングや自動フェイルオーバー、バックアップなど、便利な機能が提供されている

Kubernetes

- クラウドネイティブなプラットフォーム
- マイクロサービスで使われているデータベースの運用の課題を解決
- システム全体のプラットフォームの統一

SRA OSS 稼働環境によって利用するPgpool-IIの機能の差異

PostgreSQL 稼働環境	Pgpool-II 稼働環境	利用するPgpool-IIの機能				
		フェイルオーバ	ヘルスチェック	Watchdog	負荷分散	コネクションプー リング
EC2	EC2	✓	✓	✓	✓	✓
Amazon RDS for PostgreSQL	EC2	-	-	✓ / - ※ Watchdogまたは AWSの機能を利用	✓	✓
Amazon Aurora PostgreSQL	EC2	-	-	✓ / - ※ Watchdogまたは AWSの機能を利用	✓	✓
Kubernetes	Kubernetes	-	-	-	✓	✓

- Watchdogとは
 - Pgpool-II自身の可用性を高める機能
 - 仮想IPがリーダーPgpool-IIに付与される
 - リーダーPgpool-IIの切り替え時に仮想IPの付け替えを行う
- **しかし、クラウドでは仮想IPが使えない**

クラウド上で使用可能な仮想IPの代替案

- Elastic IPの付け替え
- セカンダリプライベートIPの付け替え
- ルートテーブルの書き換え
- DNSサービス(Route53など)のAレコードの書き換え

クラウド上で使用可能な仮想IPの代替案

	Multi-AZ対応	クライアント (接続元)	Pgpool-II EC2インスタンスのサブネット	備考
Elastic IPの付け替え	○	インターネット経由で接続 (パブリックサブネット)	パブリックサブネットに配置する必要ある	
セカンダリプライベートIPの付け替え	×	VPC内部	プライベートサブネットで利用可	
ルートテーブルの書き換え	○	VPC内部	プライベートサブネットで利用可	
DNSサービスのAレコードの書き換え	○	VPC内部 別のピア接続VPC オンプレミス	プライベートサブネットで利用可	設定変更の反映には時間がかかる

SRAOSS WatchdogによるPgpool-IIの冗長化(3/3)

- pgpool.confの設定例

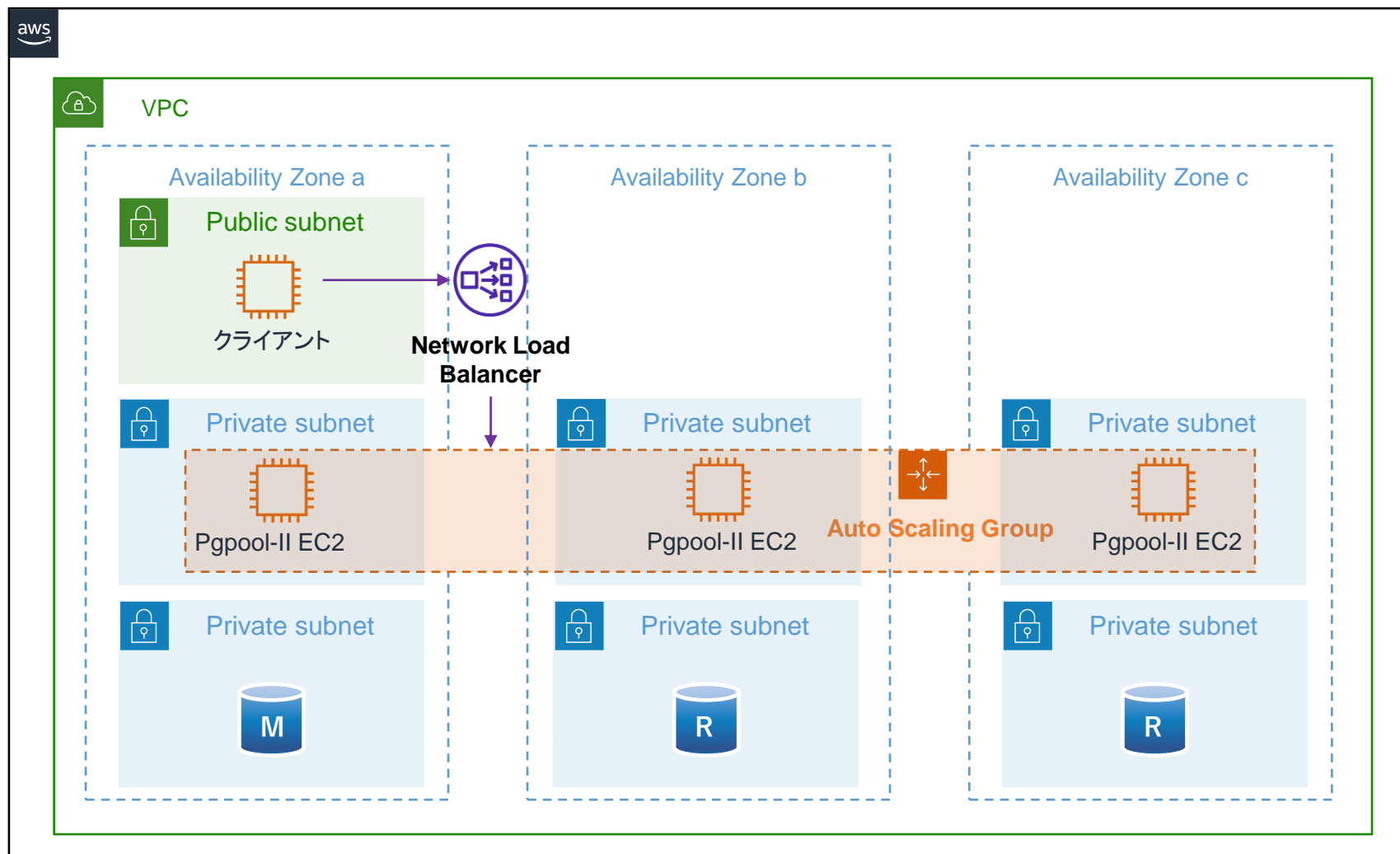
```
delegate_IP = '20.0.0.50'  
if_up_cmd = /etc/pgpool-II/aws_rtb_if_cmd.sh up $_IP_$ eth0 rtb-012345abcd,rtb-67890abcd /usr/local/bin/aws'  
if_down_cmd = '/etc/Pgpool-II/aws_rtb_if_cmd.sh up $_IP_$ eth0 rtb-012345abcd,rtb-67890abcd /usr/local/bin/aws'  
arping_cmd = 'true'
```

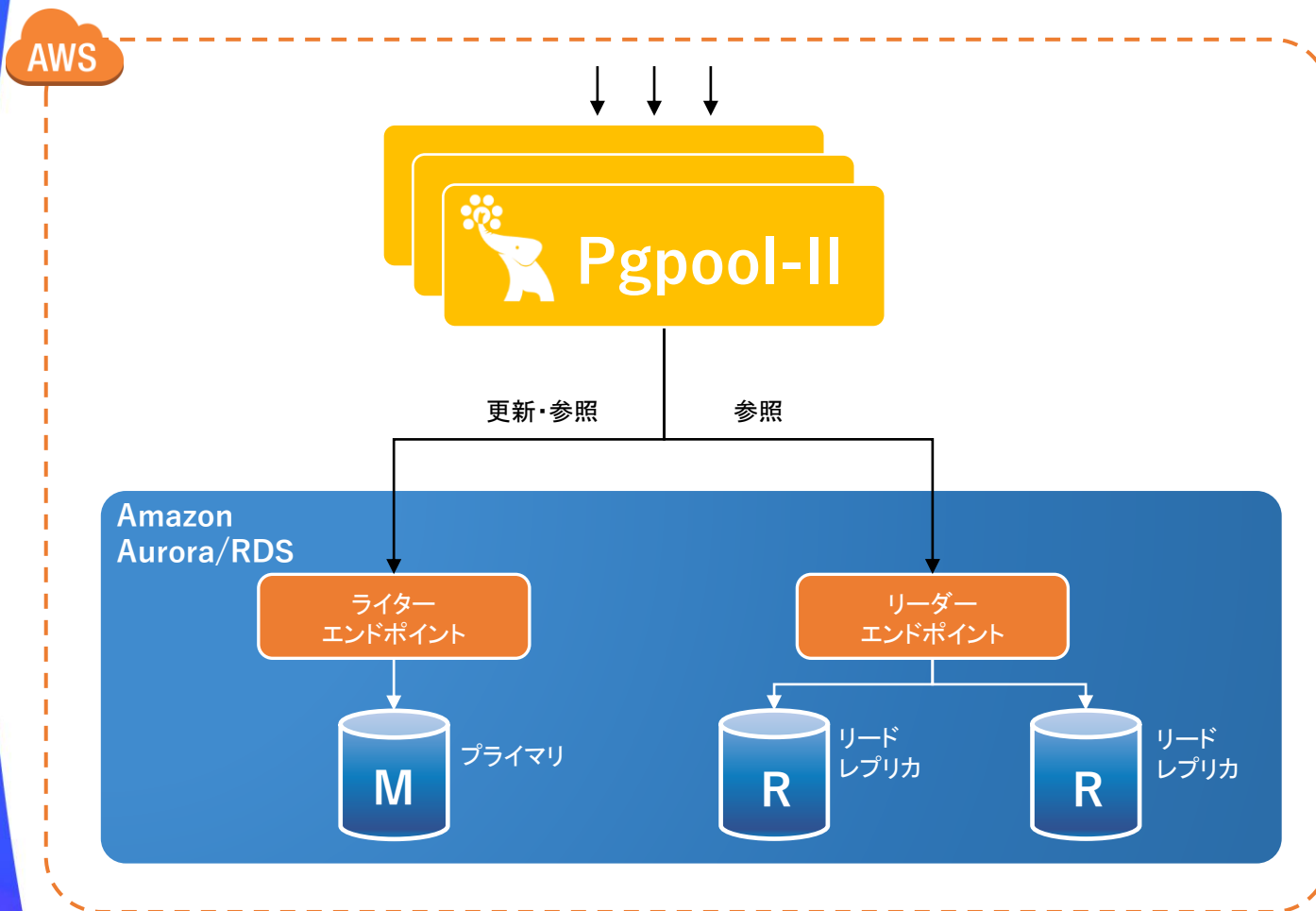
- aws_rtb_if_cmd.shスクリプトの設定例 (詳細については、ドキュメントを参照してください)

```
#!/bin/bash  
CMD=$1  
VIP=$2  
LOCAL_INTERFACE=$3  
ROUTE_TABLE_ID=$4  
AWSCLI=$5  
TOKEN=$(curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-seconds: 21600")  
MAC_ADDR=$(ip -br link show dev ${LOCAL_INTERFACE} | tr -s ' ' | cut -d ' ' -f3)  
NETWORK_INTERFACE_ID=$(curl -H "X-aws-ec2-metadata-token: $TOKEN" -s http://169.254.169.254/latest/meta-  
data/network/interfaces/macs/${MAC_ADDR}/interface-id)  
  
if [ "${CMD}" == "up" ]; then  
    # クライアント側のルートテーブルが指しているターゲットを自身のネットワークインターフェイスIDに書き換える  
    for rtb in $(echo ${ROUTE_TABLE_ID} | sed -e 's/,/ /g'); do  
        ${AWSCLI} ec2 create-route --route-table-id ${rtb} ¥  
            --destination-cidr-block ${VIP}/32 --network-interface-id ${NETWORK_INTERFACE_ID}  
  
        if [ $? -ne 0 ]; then  
            ${AWSCLI} ec2 replace-route --route-table-id ${rtb} ¥  
                --destination-cidr-block ${VIP}/32 --network-interface-id ${NETWORK_INTERFACE_ID}  
        fi  
    done  
    # IPアドレスを割り当てる  
    /usr/bin/sudo /sbin/ip addr add ${VIP}/32 dev ${LOCAL_INTERFACE} label ${LOCAL_INTERFACE}:1  
fi
```

ルートテーブル
の書き換え
の設定例

AWSの **Elastic Load Balancing (ELB)** + **EC2 Auto Scaling**を用いてPgpool-IIの高可用性を実現





- ストリーミングレプリケーションモード
(デフォルトで負荷分散、コネクションプールが有効)

```
backend_clustering_mode = 'streaming_replication'
load_balance_mode = on
connection_cache = on
```

- バックエンドノード情報にライターエンドポイントとリーダーエンドポイントを設定

```
backend_hostname0 = 'ライターエンドポイント'
backend_hostname1 = 'リーダーエンドポイント'
```

- プライマリノードを自動判別せず固定

```
backend_flag0 = 'ALWAYS_PRIMARY|DISALLOW_TO_FAILOVER'
backend_flag1 = 'DISALLOW_TO_FAILOVER'
```

- フェイルオーバはAWSに任せるので、ヘルスチェックを無効にする (デフォルトで無効)

```
health_check_period = 0
```

- バックエンドエラーが発生した時にフェイルオーバを起こさないように

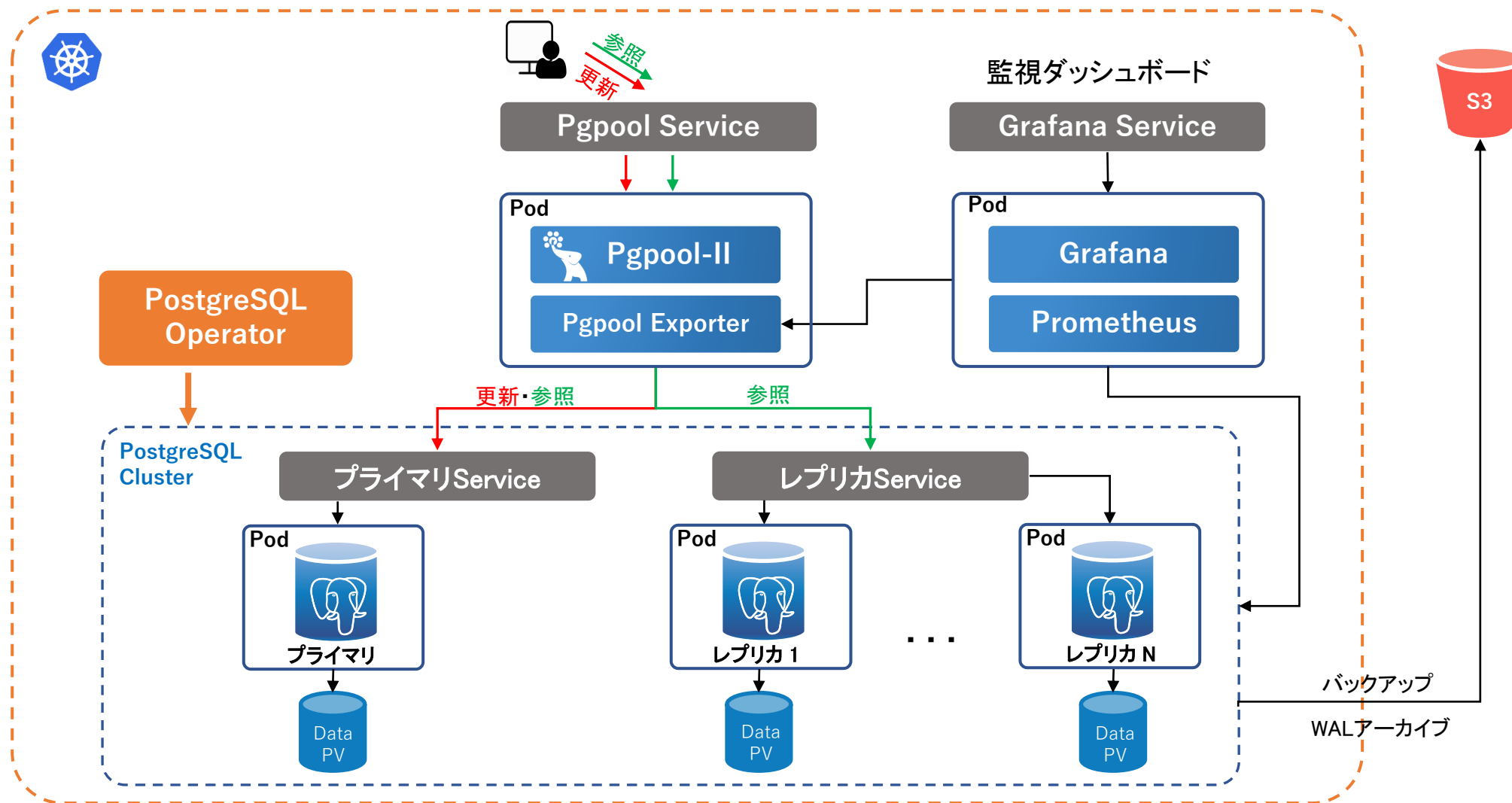
```
failover_on_backend_error = off
```

- ストリーミングレプリケーション遅延チェックを無効にする

```
sr_check_period = 0
```

- Kubernetes上でPostgreSQLを運用するには専用のOperatorが必要
- PostgreSQL Operator
 - PostgreSQLクラスタのPrimary/Standbyの役割管理
 - PostgreSQLの管理タスクの自動化
- 各社がPostgreSQL Operatorを開発中
 - Zalando Postgres Operator
 - PGO, the Postgres Operator from Crunchy Data
 - KubeDB
 - EDB Postgres for Kubernetes
 - など

Kubernetes全体構成図



SRA OSS KubernetesにおけるPgpool-IIの設定

機能

負荷分散

コネクションプール

これらの機能のみを有効にする

ヘルスチェック

自動フェイルオーバー

オンラインリカバリ

Watchdog (Pgpool-II の HA機能)



Kubernetesに任せる

最小設定

バックエンド情報

(2台のみ: Primary Service と Replica Service)

backend_hostname0='Primary Service'

backend_hostname1='Replica Service'

backend_port0='5432'

backend_port1='5432'

backend_flag0='ALWAYS_PRIMARY|DISALLOW_TO_FAILOVER'

backend_flag1='DISALLOW_TO_FAILOVER'

バックエンドエラーが発生した時にフェイルオーバーを起こさないように

failover_on_backend_error = off

ヘルスチェックを無効にする

health_check_period = 0

ストリーミングレプリケーション遅延チェック (任意)

sr_check_userのパスワードはpool_passwdファイルで設定

sr_check_period = 10

sr_check_user = 'PostgreSQLユーザ名'

その他

load_balance_mode = on

connection_cache = on

listen_addresses = '*'

- クラウドにおけるPgpool-IIの活用
 - 様々なPostgreSQLの稼働環境に対応
 - EC2
 - Amazon RDS/Aurora
 - Kubernetes
 - PostgreSQLの稼働環境によって、Pgpool-IIの稼働環境や設定が異なる
 - クラウド上でPgpool-IIの冗長化
 - Pgpool-IIの機能
 - AWSの機能
 - クラウド上で使用可能な仮想IPの代替案

- Pgpool-II
 - <https://pgpool.net/>
 - <https://www.pgpool.net/docs/latest/ja/html/>
- AWSドキュメント
 - https://docs.aws.amazon.com/ja_jp/
- Kubernetesの設定例
 - https://github.com/pgpool/pgpool2_on_k8s
- AWS/Amazon Auroraの設定例
 - <https://www.pgpool.net/docs/latest/ja/html/example-aws.html>
 - <https://www.pgpool.net/docs/latest/ja/html/example-aurora.html>

ご清聴ありがとうございました。

