クラウド時代を担う PostgreSQLクラスタ管理ツール Pgpool-IIの紹介

EDB Postgres Vision Tokyo 2022
2022/9/7

SRA OSS LLC
彭 博 (ペン ボ)

自己紹介

ペン ボ

- 名前：彭 博 (Bo Peng)
  pengbo@sraoss.co.jp
- 所属：SRA OSS LLC
  基盤技術グループ
- 職務：
  - OSS技術サポート、ミドルウェア構築
  - PostgreSQLクラスタ管理ツールPgpool-IIの開発者

会社概要

社名	SRA OSS合同会社
略称	SRA OSS LLC
設立	2022 年 6 月
社長	稲葉　香理
事業内容	• オープンソースソフトウェア関連のテクニカルサポート • オープンソースソフトウェア関連のコンサルティング・構築 • オープンソースソフトウェア関連プロダクトの開発・販売 • オープンソースソフトウェアの教育 • オープンソースソフトウェアの開発 • オープンソースソフトウェアコミュニティの運営支援 • ソフトウェアの研究開発

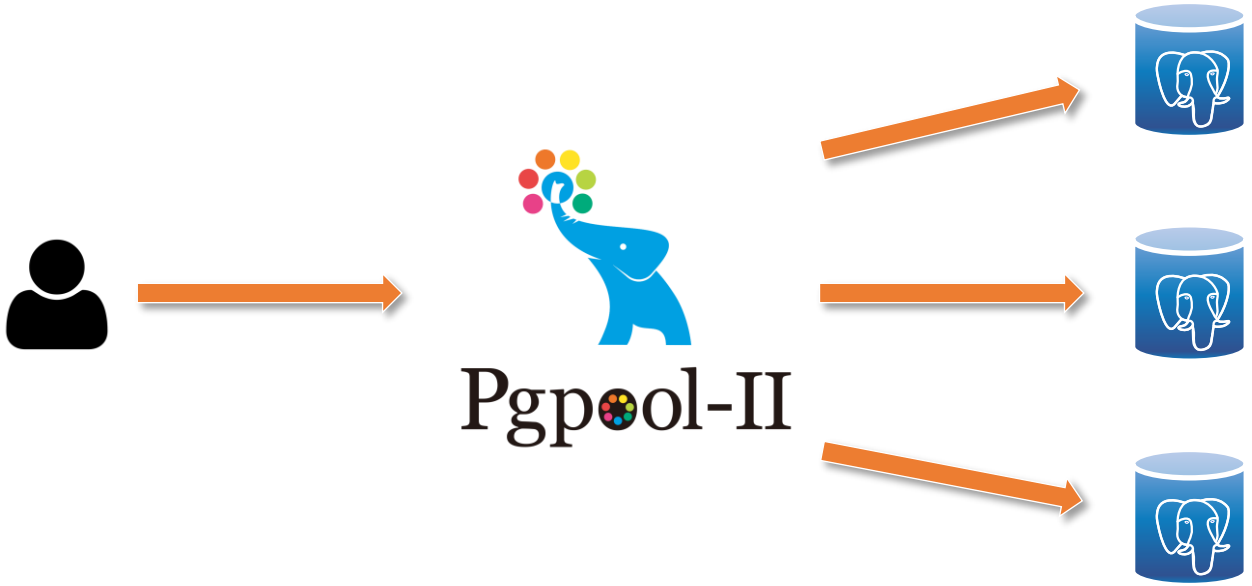
1999 年	日本で初めて PostgreSQL の商用サポートを開始
2003 年	PostgreSQL 完全互換で使いやすさと安心のサポートを提供する「PowerGres」の販売を開始
2005 年	SRA OSS, Inc. 日本支社 設立
2007 年	テキスト変換ライブラリ「libTextConv」の発売を開始
2009 年	メールの高速全文検索ソフトウェアSylpheed Pro の販売を開始
2011 年	OSS ワンストップサポート「OSS プロフェッショナルサポートサービス」を開始
2012 年	PostgreSQLの多機能ミドルウェア 「Pgpool-II」 のサポートサービスを開始
2022 年	**SRA OSS 合同会社設立** SRA OSS, Inc. 日本支社よりすべての事業を譲受

本日の流れ

- Pgpool-IIとは
- Pgpool-IIの機能
- クラウドにおけるPgpool-IIの活用

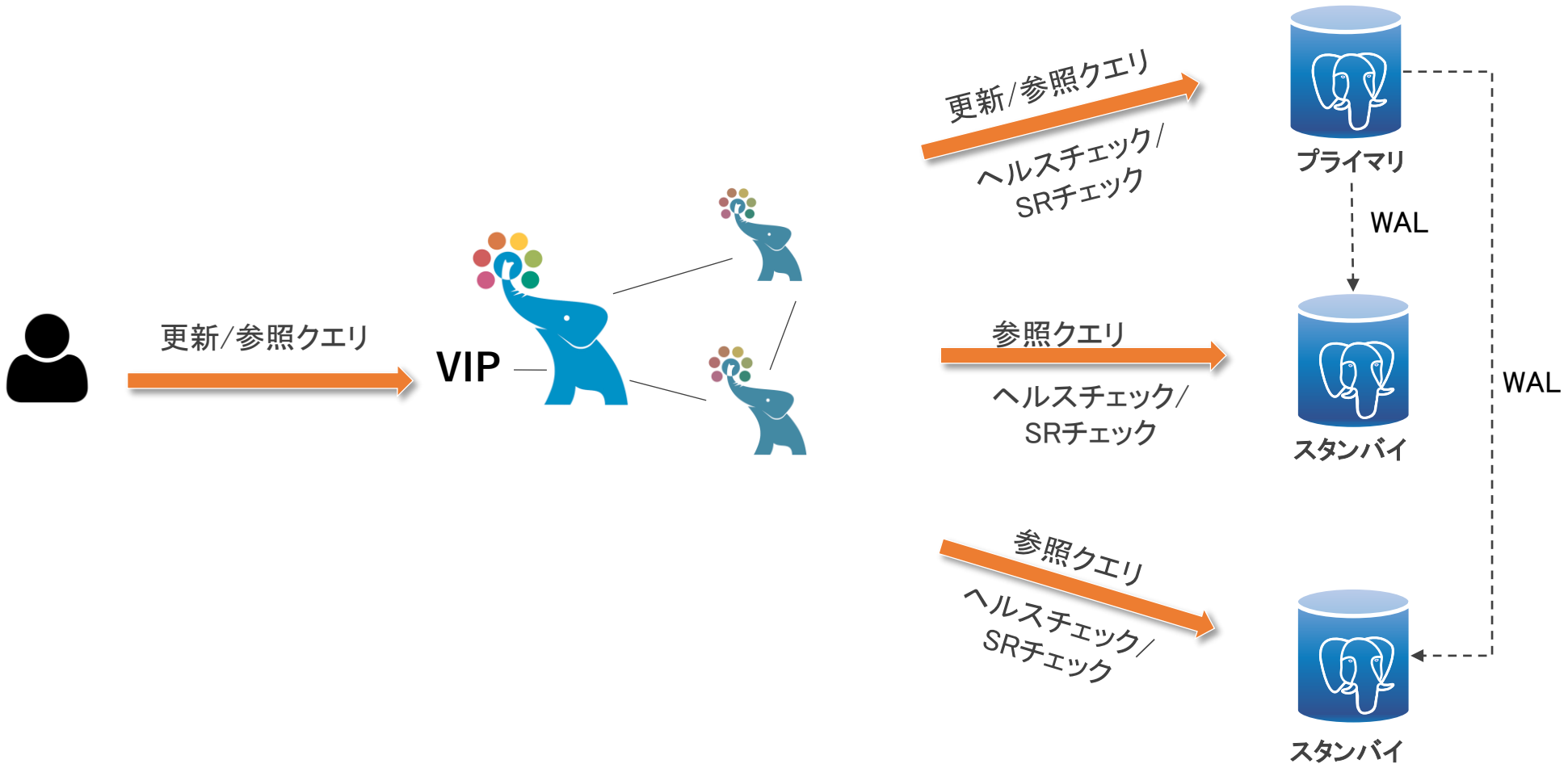
Pgpool-IIとは

- クライアントとPostgreSQLの間で動作するミドルウェア
- ユーザは複数PostgreSQLサーバを意識せず、1台のように見える



Pgpool-IIの機能

Pgpool-IIの基本構成



クラウドにおけるPgpool-IIの活用

AWSにおけるPostgreSQLの稼働環境

- 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

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によるPgpool-IIの冗長化(1/3)

- 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

WatchdogによるPgpool-IIの冗長化(2/3)

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

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/if_cmd.sh up $_IP_$ eth0 rtb-08ba4c1b34 /usr/local/bin/aws'
if_down_cmd = '/etc/pgpool-II/if_cmd.sh down $_IP_$ eth0 rtb-08ba4c1b34 /usr/local/bin/aws'
arping_cmd = 'true'
```

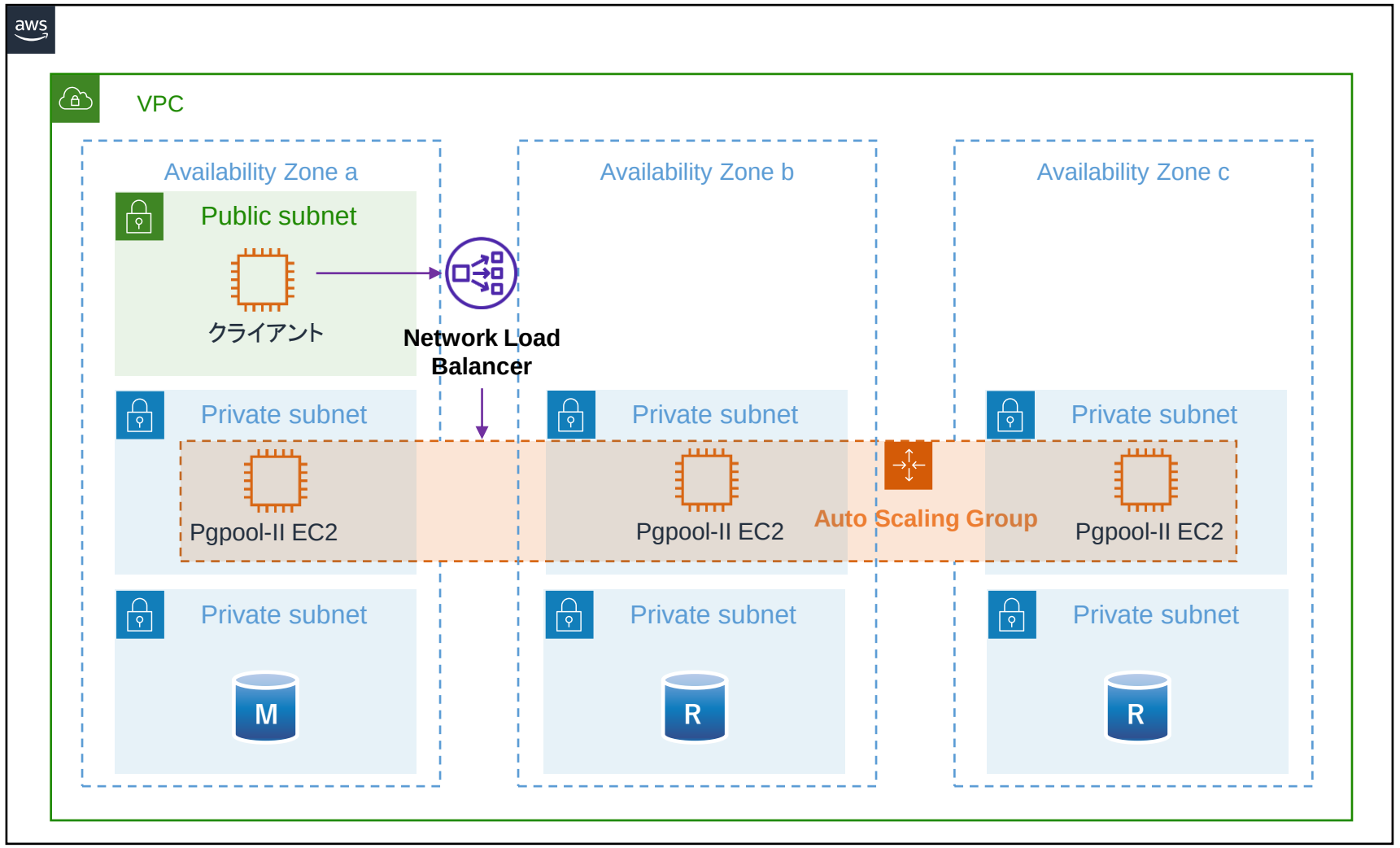
- if_cmd.shスクリプトの設定例

```
#!/bin/bash
CMD=$1
(省略)
TOKEN=$(curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-seconds: 21600")
INSTANCE_ID=$(curl -H "X-aws-ec2-metadata-token: $TOKEN" -s http://169.254.169.254/latest/meta-data/instance-id)
EC2_NETWORK_INTERFACE_ID=$(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EC2インスタンスIDに書き換え
    ${AWSCLI} ec2 create-route --route-table-id ${ROUTE_TABLE_ID} ¥
      --destination-cidr-block ${VIP}/32 --instance-id ${INSTANCE_ID}
    if [ $? -ne 0 ]; then
        ${AWSCLI} ec2 replace-route --route-table-id ${ROUTE_TABLE_ID} ¥
          --destination-cidr-block ${VIP}/32 --instance-id ${INSTANCE_ID}
    fi

    # IPアドレスを割り当て
    /usr/bin/sudo /sbin/ip addr add ${VIP}/32 dev ${LOCAL_INTERFACE} label ${LOCAL_INTERFACE}:1

    # 自身のEC2ネットワークインターフェイスの設定で「送信元/送信先チェック」を無効にする
    ${AWSCLI} ec2 modify-network-interface-attribute ¥
      --network-interface-id ${EC2_NETWORK_INTERFACE_ID} --no-source-dest-check
fi
```

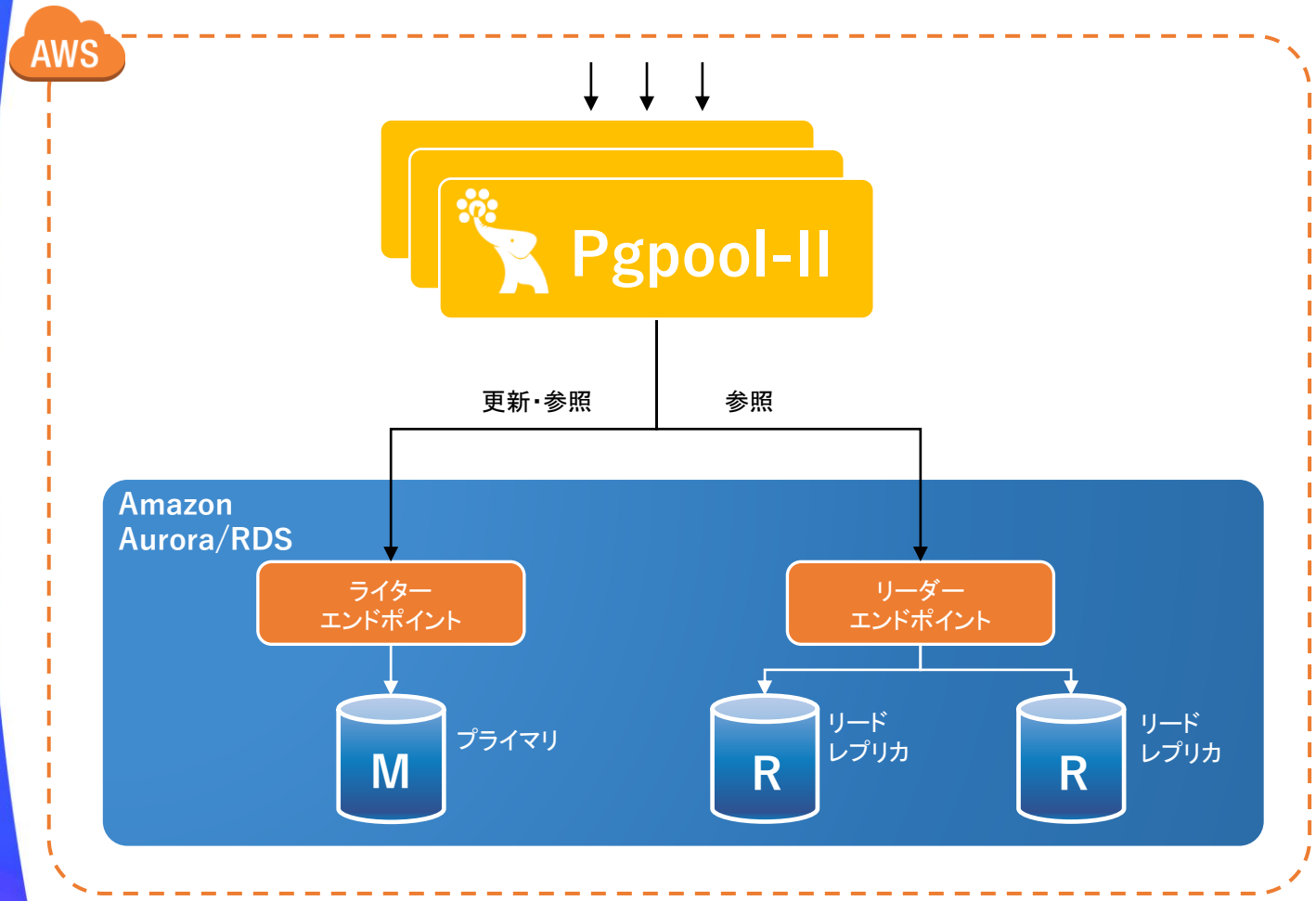
ルートテーブルの書き換えの設定例

AWSの機能によるPgpool-IIの冗長化

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



設定例: Pgpool-II + Amazon Aurora/RDS



- 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'
```

- WRITE/READエラー時にフェイルオーバを起こさないように

```
failover_on_backend_error = off
```

- 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
```

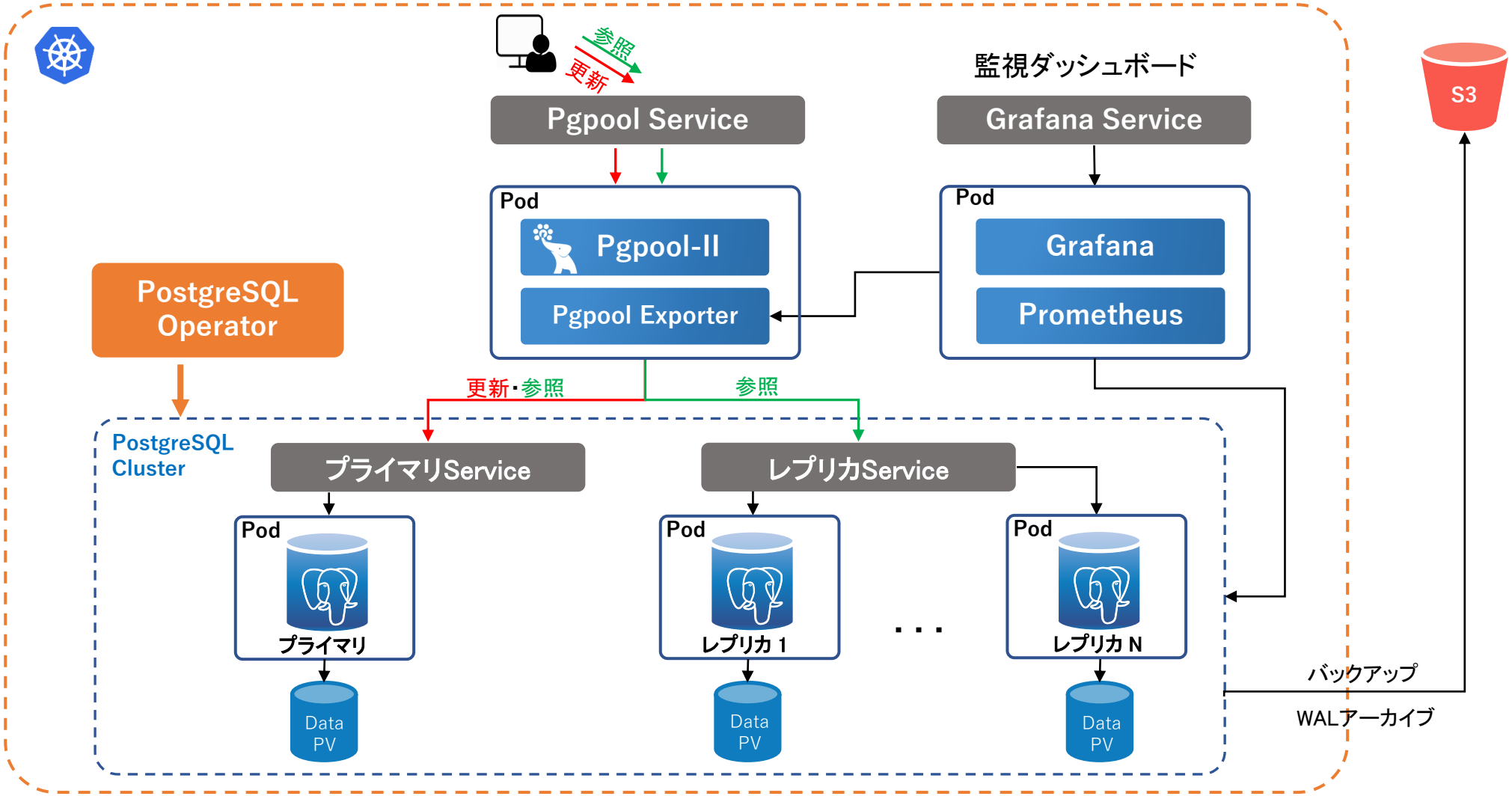
- ストリーミングレプリケーション遅延チェックを無効にする

```
sr_check_period = 0
```

KubernetesにおけるPgpool-IIの活用

- 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



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

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'

# WRITE/READエラー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 = '*'
```

まとめ

- 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

参考情報

- 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
- Amazon Auroraの設定例
  - 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。