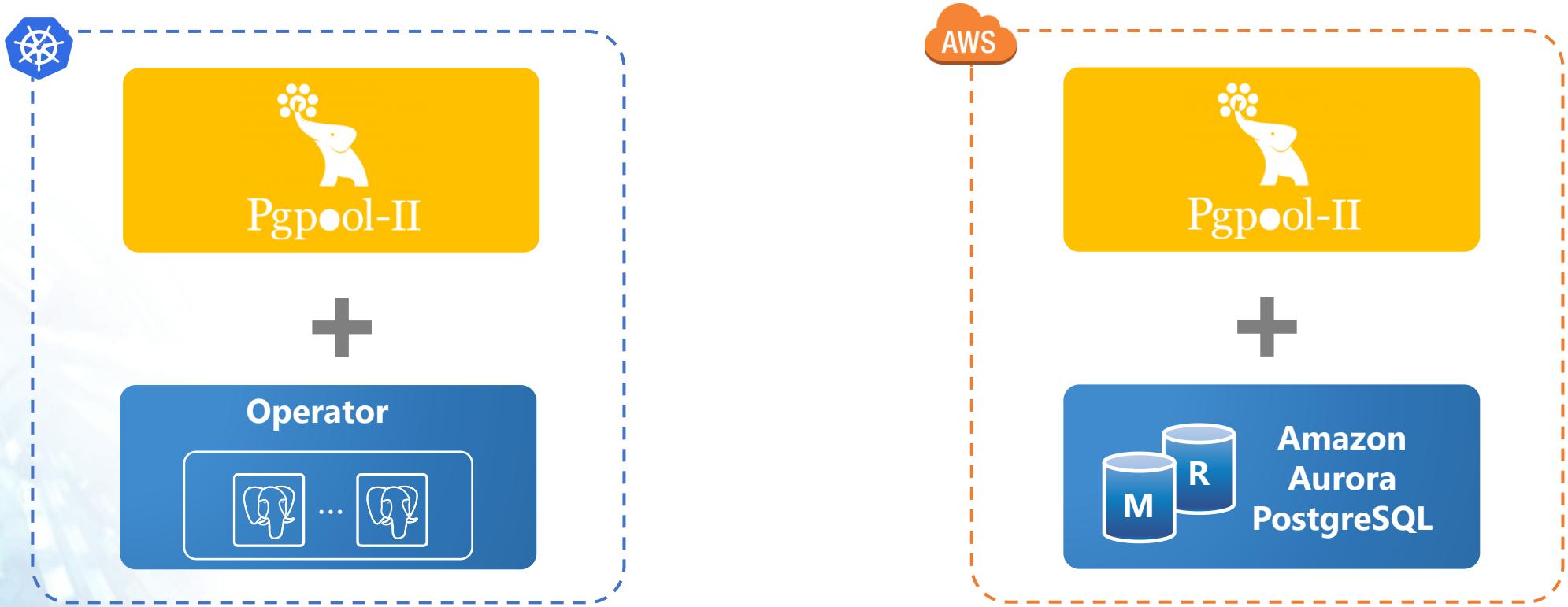
Pgpool-IIの構成紹介と活用事例

2021-01-19

SRA OSS, Inc. 日本支社
彭 博 (ペン ボ)

クラウドにおけるPgpool-IIの活用

- Amazon Aurora PostgreSQLやKubernetesでもPgpool-IIを利用可能
  - クエリの振り分け
  - コネクションプーリング



Kubernetes上でPgpool-IIを動かす

Kubernetesとは

コンテナの管理を自動化するためのプラットフォーム

- コンテナの死活監視
- リソース管理
- スケーリング
- サービス ディスカバリ
- セルフヒーリング
- スケジューリング
- ローリング アップデート
- ロードバランシング

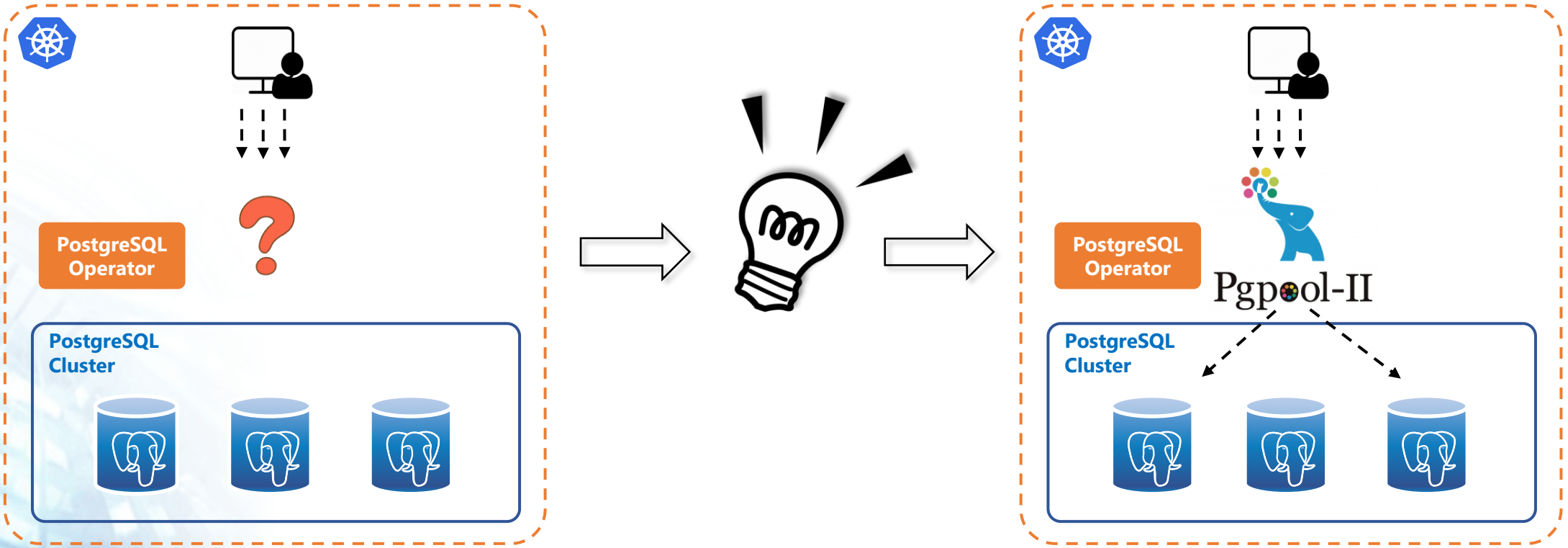
PostgreSQL Operator

- Operator
  - Kubernetesの本来の機能を拡張し、様々な管理をコードとして記述し、自動化する
- PostgreSQL Operator
  - PostgreSQLの管理タスクの自動化 (バックアップ、死活監視、フェイルオーバなど)
  - PostgreSQLクラスタのPrimary/Standbyの役割管理

	 Zalando PostgreSQL Operator	 Crunchy PostgreSQL Operator
開発元	Zalando SE	Crunchy Data
ライセンス	MIT License	Apache License 2.0
対応バージョン	PostgreSQL 9.6以降	PostgreSQL 9.5以降
動作環境	Amazon EKS, Google Kubernetes Engine (GKE), Red Hat OpenShift	Amazon EKS, Google Kubernetes Engine (GKE), Red Hat OpenShift, VMWare Enterprise PKS, IBM Cloud Pak Data
プロジェクトURL	https://github.com/zalando/postgres-operator	https://github.com/CrunchyData/postgres-operator

KubernetesにおけるPrimaryとReplicaへのクエリ振り分け

- 既存のPostgreSQL Operatorにはクエリ振り分け機能がない
- クライアントとPostgreSQLの間に位置し、PostgreSQLのクエリを解析し振り分けるツールが必要



KubernetesにおけるPgpool-IIの設定

機能

クエリの振り分け
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='hippo'
backend_hostname1='hippo-replica'
backend_port0='5432'
backend_port1='5432'
backend_flag0='ALWAYS_PRIMARY|DISALLOW_TO_FAILOVER'
backend_flag1='DISALLOW_TO_FAILOVER’
```

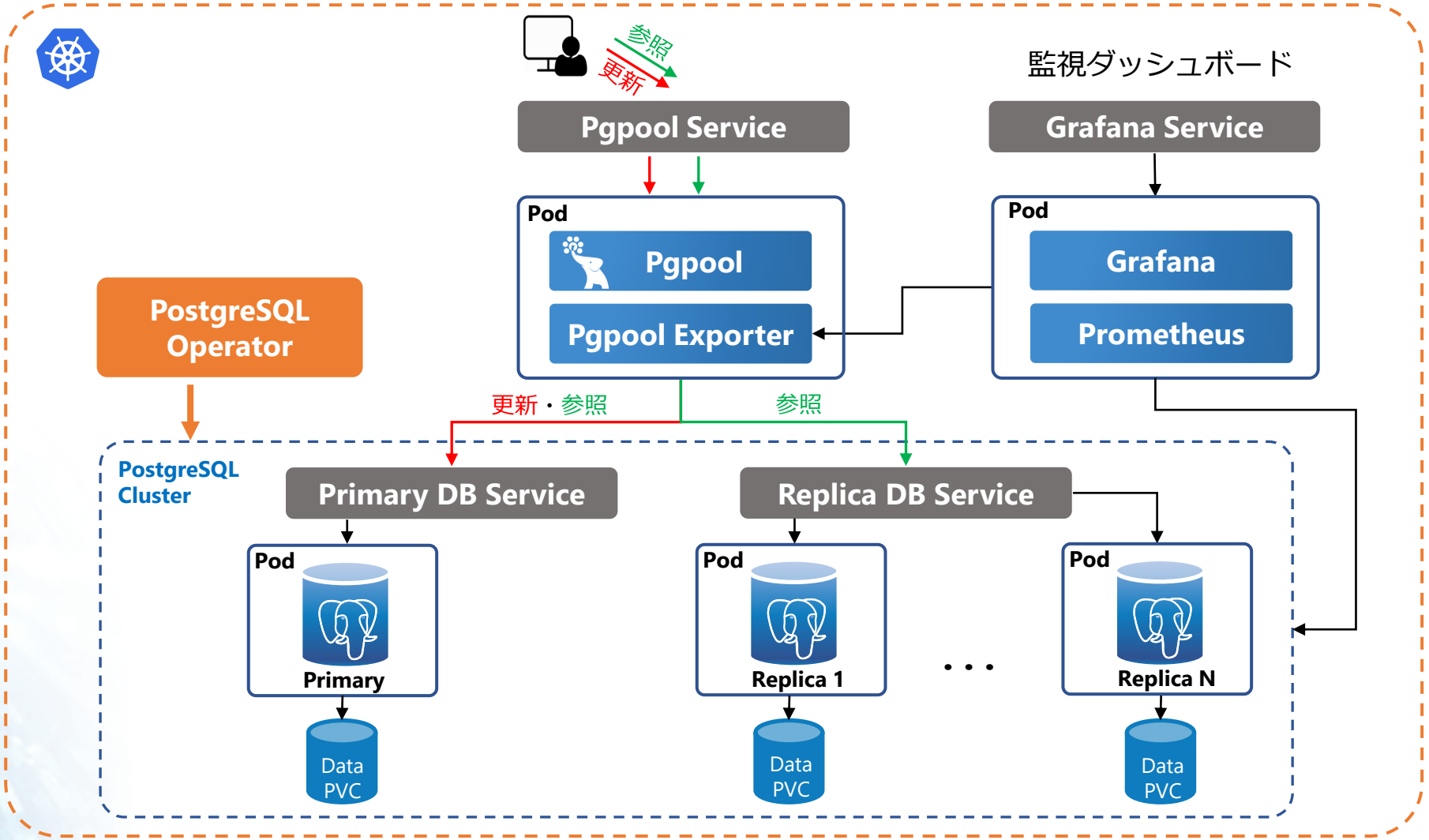
レプリケーションチェック

```
sr_check_period = 0     # Default
```

その他

```
load_balance_mode = on # Default
connection_cache = on  # Default
listen_addresses = '*’ # Default
```

全体構成図



Pgpool-IIのデプロイ- 環境変数を用いたPgpool-IIの設定

```
# pgpool_deploy.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: pgpool
spec:
  replicas: 1
...
spec:
      containers:
      - name: pgpool
        image: pgpool/pgpool:4.2
        env:
        - name: PGPOOL_PARAMS_BACKEND_HOSTNAME0
          value: "hippo"
        - name: PGPOOL_PARAMS_BACKEND_HOSTNAME1
          value: "hippo-replica"
        - name: PGPOOL_PARAMS_BACKEND_PORT0
          value: "5432"
        - name: PGPOOL_PARAMS_BACKEND_PORT1
          value: "5432"
          ...
      - name: pgpool-stats
        image: pgpool/pgpool2_exporter:1.0
        env:
         - name: PGPOOL_SERVICE
           value: "localhost"
```

- Kubernetes上でPgpool-IIをデプロイするための各種サンプルYAMLファイル

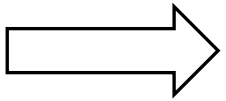
  https://github.com/pgpool/pgpool2_on_k8s

- ドキュメント

  https://www.pgpool.net/docs/latest/ja/html/example-kubernetes.html

```
# kubectl create namespace demo
# kubectl apply -f pgpool_deploy.yaml -namespace=demo
```

PGPOOL_PARAMS_<パラメータ名>
形式の環境変数が
設定パラメータに変換される



環境変数を用いて
任意の Pgpool-II の
パラメータを設定できる

```
backend_hostname0='hippo'
backend_hostname1='hippo-replica'
backend_port0='5432'
backend_port1='5432'
...
```

※テストや学習を目的とした必要最低限のパラメータ設定の場合

Pgpool-IIのデプロイ - ConfigMapを用いたPgpool-IIの設定

ConfigMap

- 環境変数や設定ファイルをPodに設定するための仕組み
- 各種設定ファイルのパラメータを直接指定してConfigMapを作成可能
- PodのVolumeとしてマウント可能
- 暗号化されない

```
# pgpool_configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: pgpool-config
  labels:
    app: pgpool-config
data:
  pgpool.conf: |-
    listen_addresses =
    port = 9999
    socket_dir = '/var/
    pcp_listen_addresse
    pcp_port = 9898
...
  pcp.conf: |-
    postgres:e8a4865385
...
  pool_passwd: |-
    postgres:md53175bce
    user1:md542b72f913c
    user2:md548b83a2a92
...
  pool_hba.conf: |-
    local   all     all
    host    all     all
    host    all     all
    host    all     all         0.0.0.0/0         md5
```

mount

```
# pgpool_deploy.yaml
apiVersion: apps/v1
kind: Deployment
...
    spec:
      containers:
      - image: pgpool/pgpool:4.1
        command: ["sh", "-c"]
        args: ["$PGPOOL_BINARY_DIR/pgpool -n -f
$PGPOOLCONF -F $PCP_CONF -a $POOL_HBA_CONF"]
        env:
        ...
        volumeMounts:
        - name: pgpool-config
          mountPath: /usr/local/pgpool-II/etc
      volumes:
      - name: pgpool-config
        configMap:
          name: pgpool-config
```

Pgpool-IIによる読み取りクエリの負荷分散

```
$ kubectl -n demo port-forward svc/pgpool 9999:9999 &
$ psql -h localhost -U postgres -c "show pool_nodes"
node_id|   hostname      | port |status| lb_weight |  role   | select_cnt | load_balance_node| replication_delay
-------+-----------------+------+------+----------+---------+-------------+------------------+-----------------+
0      | hippo           | 5432 | up   | 0.500000  | primary | 0           | false            | 0
1      | hippo-replica   | 5432 | up   | 0.500000  | standby | 0           | true             | 0
```

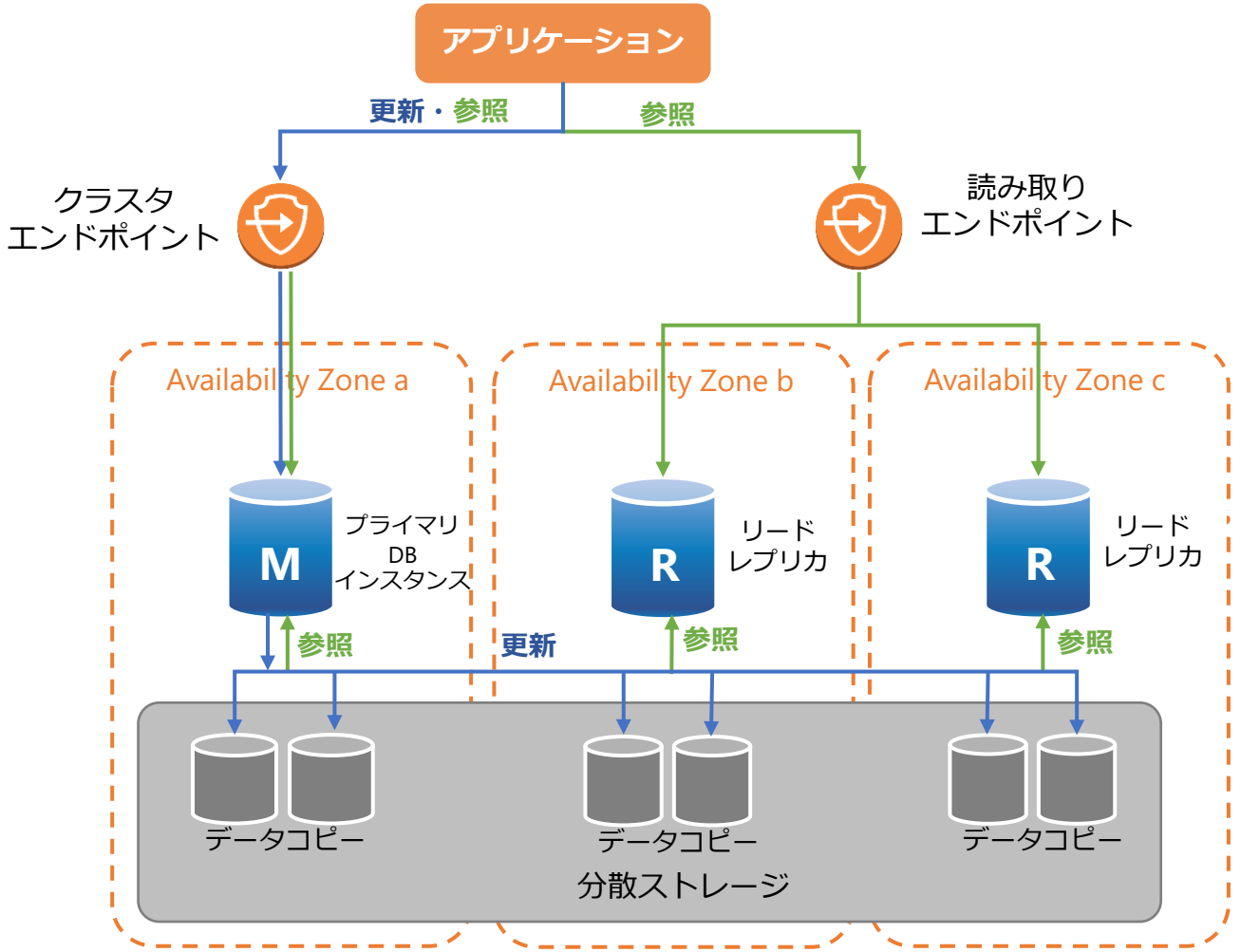
```
$ psql -h localhost -U postgres -c "SELECT 1"

$ psql -h localhost -U postgres -c "SELECT 1"
...
```

```
$ psql -h localhost -U postgres -c "show pool_nodes"
node_id |  hostname       | port |status| lb_weight|  role    | select_cnt | load_balance_node| replication_delay
--------+-----------------+------+------+----------+---------+-------------+------------------+-----------------+
0       | hippo           | 5432 | up    | 0.500000 | primary | 3           | false            | 0
1       | hippo-replica   | 5432 | up    | 0.500000 | standby | 3           | true             | 0
```

Amazon Aurora PostgreSQLにおける Pgpool-IIによるクエリ振り分け

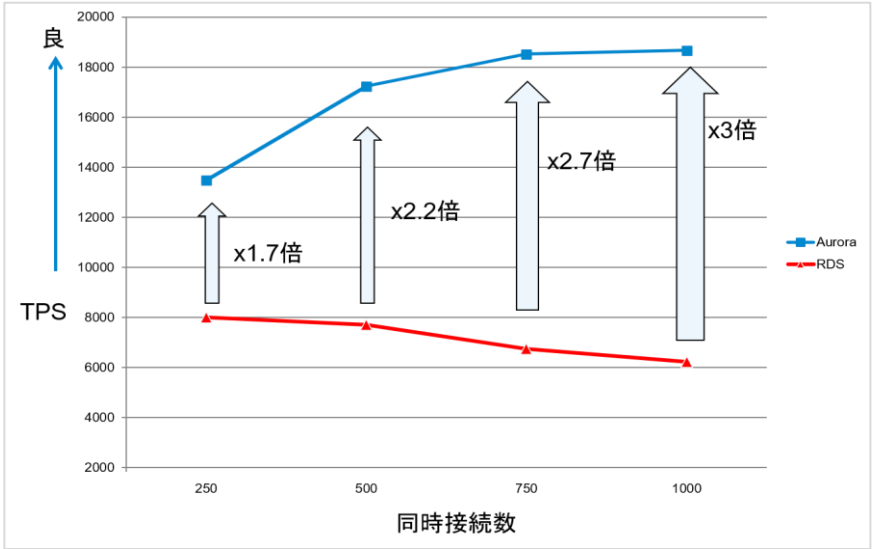
Amazon Aurora PostgreSQL

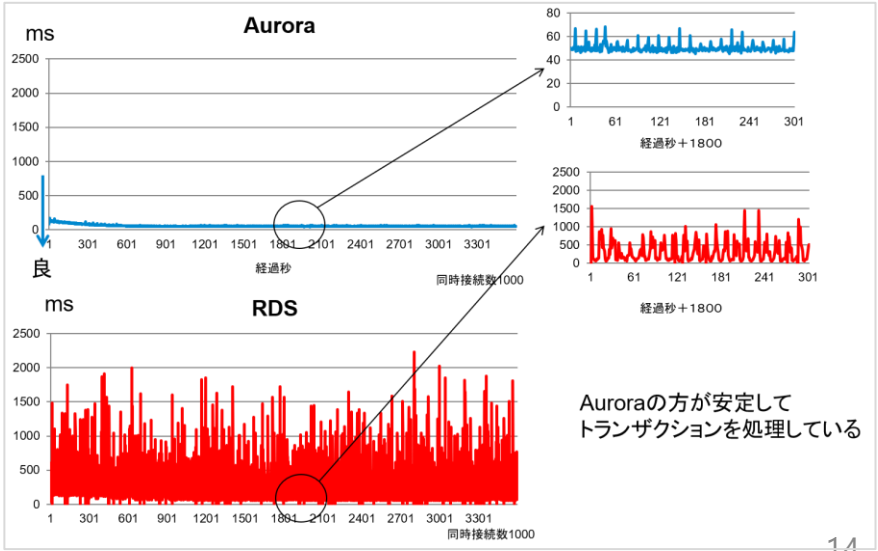
- PostgreSQLと互換性を持ったマネージド型のリレーショナルデータベースエンジン
- DBインスタンス
  - プライマリDBインスタンス (Writer)
    - 書き込み・読み取り
  - リードレプリカ (Reader)
    - 読み取りのみ (最大15個まで)
- エンドポイント
  - クラスタエンドポイント：プライマリDBインスタンスに接続
  - 読み取りエンドポイント：すべてのリードレプリカに接続 (読み取りの負荷分散)
- 自動フェイルオーバ
  - プライマリDBインスタンスに障害が発生した場合、リードレプリカが自動的にプライマリDBインスタンスに昇格



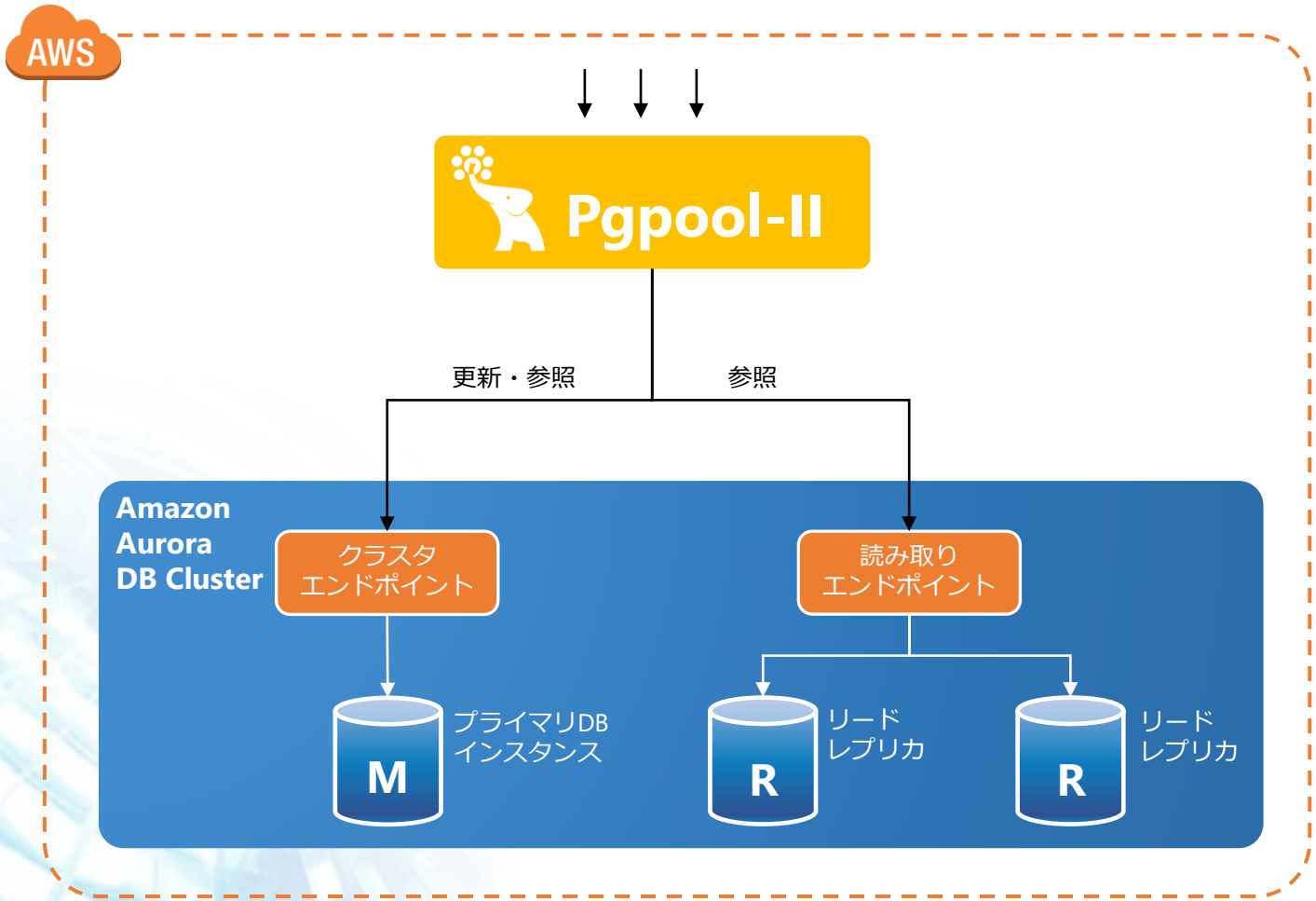
Amazon Aurora PostgreSQLの利点

- 高可用性・耐障害性
  - 自動フェイルオーバ
  - 複数のAZにまたがるストレージとDBインスタンス
- パフォーマンスの向上
  - Amazon RDS for PostgreSQLの最大3倍のスループット
  - 同時接続数を増やしてもスループットの劣化が少ない
  - レスポンス時間が短く、安定している
- スケーラビリティ
  - 最大15個までリードレプリカを作成可能





Amazon Aurora PostgreSQLにおけるPgpool-IIによるクエリ振り分け



- ストリーミングレプリケーションモード
  (デフォルトではクエリの振り分け、コネクションプールが有効)

```
cp /etc/pgpool-II/pgpool.conf{.sample-stream,}
```

- フェイルオーバはAuroraに任せるので、ヘルスチェックを無効にする

```
health_check_period = 0
```

- 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'
```

- バックエンドノード情報にクラスタエンドポイントと読み取りエンドポイントを設定

```
backend_hostname0 = 'クラスタエンドポイント'
backend_hostname1 = '読み取りエンドポイント'
```

- 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
```

- Pgpool-II のクライアント認証でmd5認証を有効にする

```
enable_pool_hba = on
```

参考情報

- Pgpool-II
  - https://pgpool.net/
  - https://www.pgpool.net/docs/latest/ja/html/
- Kubernetes の設定例
  - https://github.com/pgpool/pgpool2_on_k8s
  - https://www.pgpool.net/docs/latest/ja/html/example-kubernetes.html
- Aurora の設定例
  - https://www.pgpool.net/docs/latest/ja/html/example-aurora.html
- Pgpool-II Exporter
  - https://github.com/pgpool/pgpool2_exporter

ご清聴ありがとうございました。