Pgpool-II 徹底入門 【第2部】

~クラウド時代の Pgpool-II の活用および Pgpool-II 4.2 の新機能紹介~

PostgreSQL Conference Japan 2020
2020-11-13

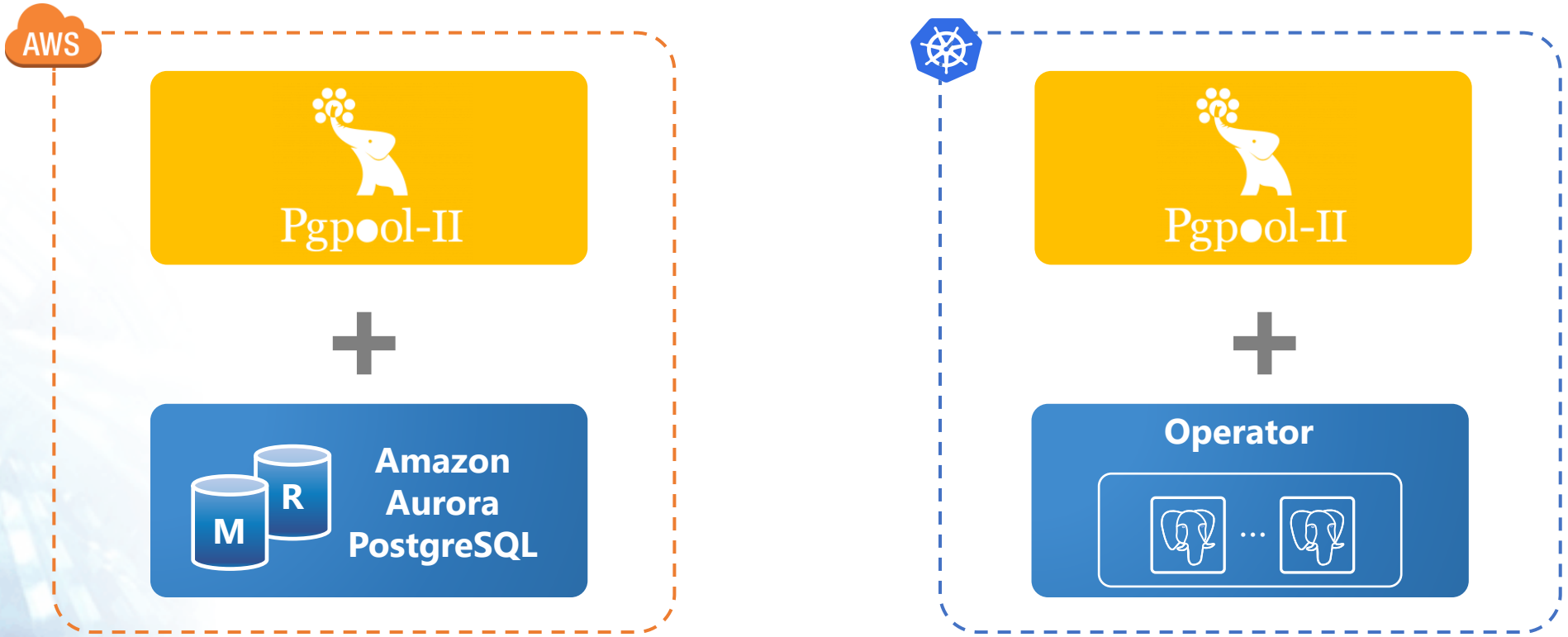
SRA OSS, Inc. 日本支社
彭 博 (ペン ボ)

自己紹介

ペン ボ
- 名前： 彭 博 (Bo Peng)
  pengbo@sraoss.co.jp
- 所属：SRA OSS, Inc. 日本支社
  基盤技術グループ
- 職務：
  - PostgreSQL 以外の OSS 全般の技術サポート
  - ミドルウェアの構築
  - PostgreSQL クラスタ管理ツールである Pgpool-II 開発者

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

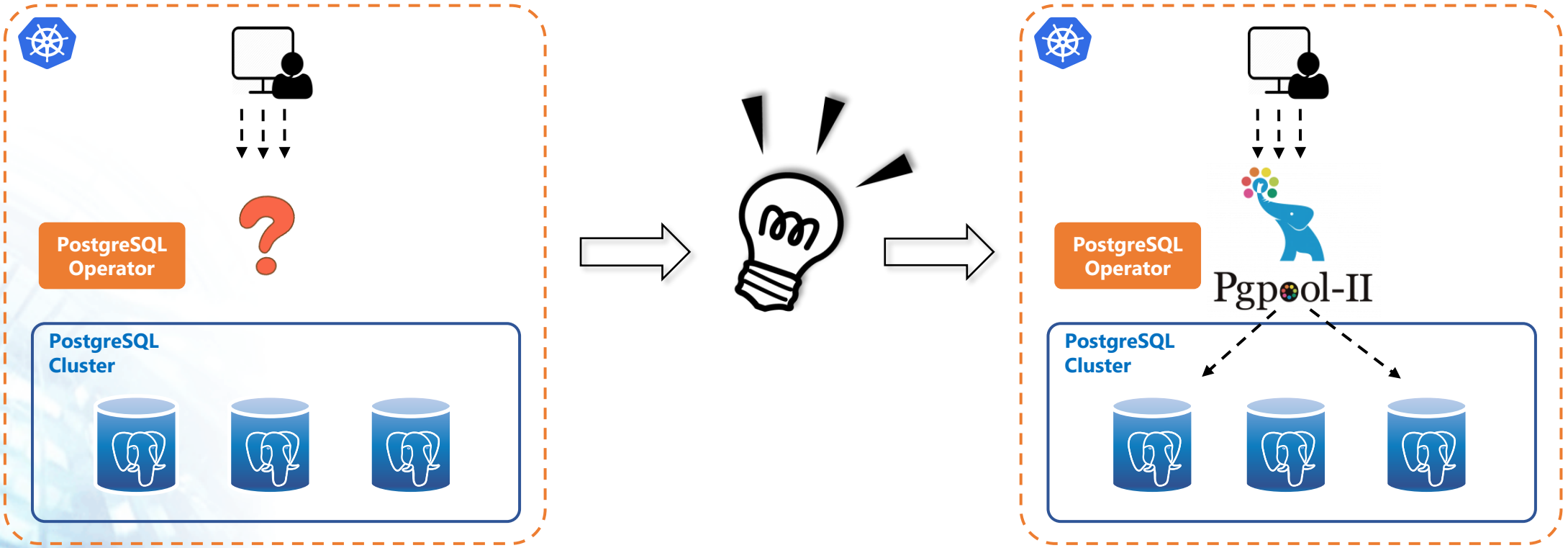
PostgreSQL Operator

- 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	 Crunchy
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'
```

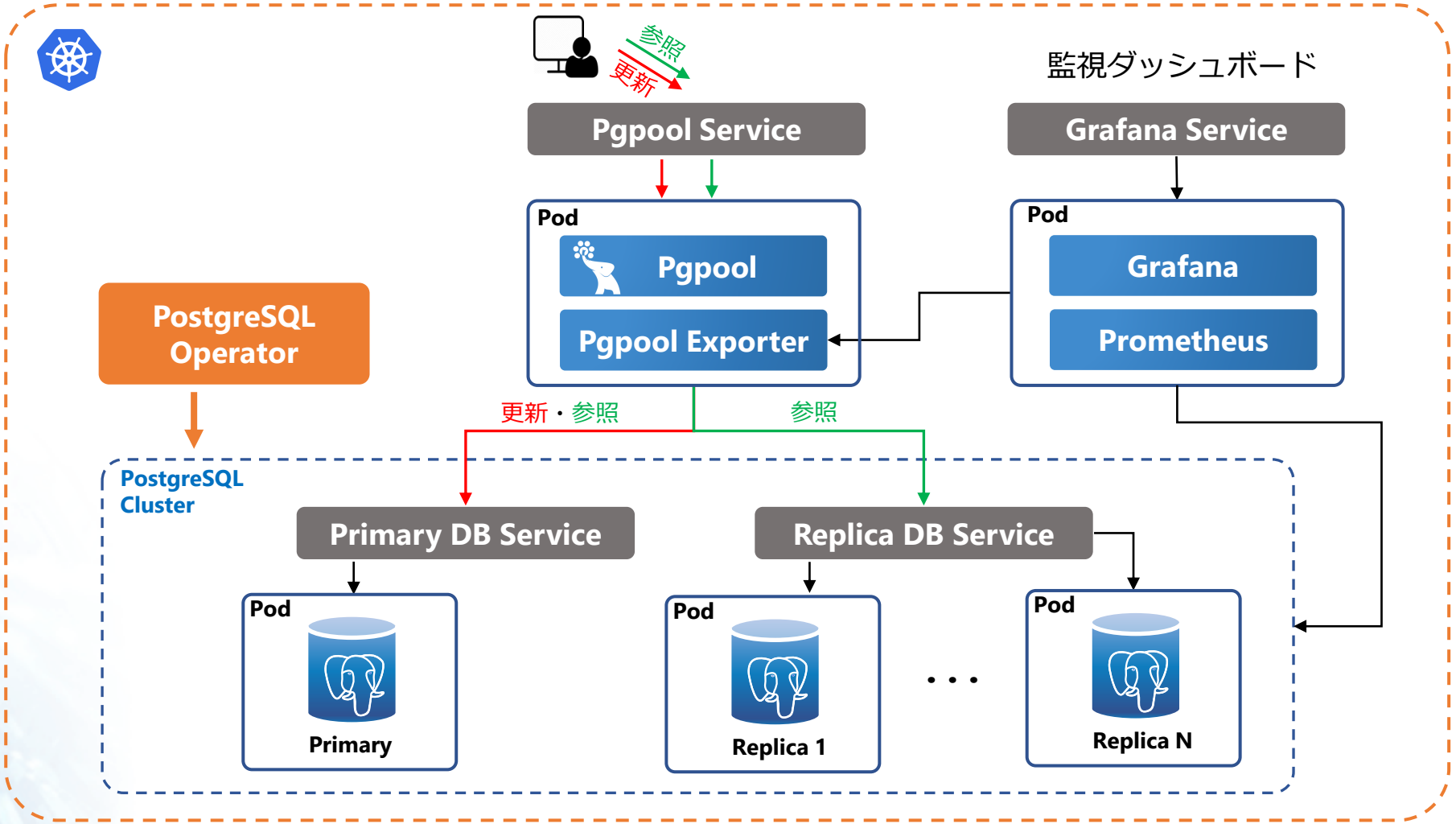
接続ユーザ

```
sr_check_user='postgres'
```

その他

```
load_balance_mode = on
connection_cache = on
listen_addresses = '*'
```

全体構成図

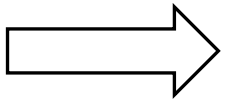


Pgpool-II のデプロイ

```
# 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_PORT
          value: "5432"
        - name: PGPOOL_PARAMS_BACKEND_PORT
          value: "5432"
          ...
      - name: pgpool-stats
        image: pgpool/pgpool2_exporter:1.0
        env:
         - name: PGPOOL_SERVICE
           value: "localhost"
```

- Kubernetes 上で Pgpool-II のデプロイ方法
  https://github.com/pgpool/pgpool2_on_k8s
- Docker イメージ
  https://hub.docker.com/repository/docker/pgpool/pgpool
  https://hub.docker.com/repository/docker/pgpool/pgpool2_exporter

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



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

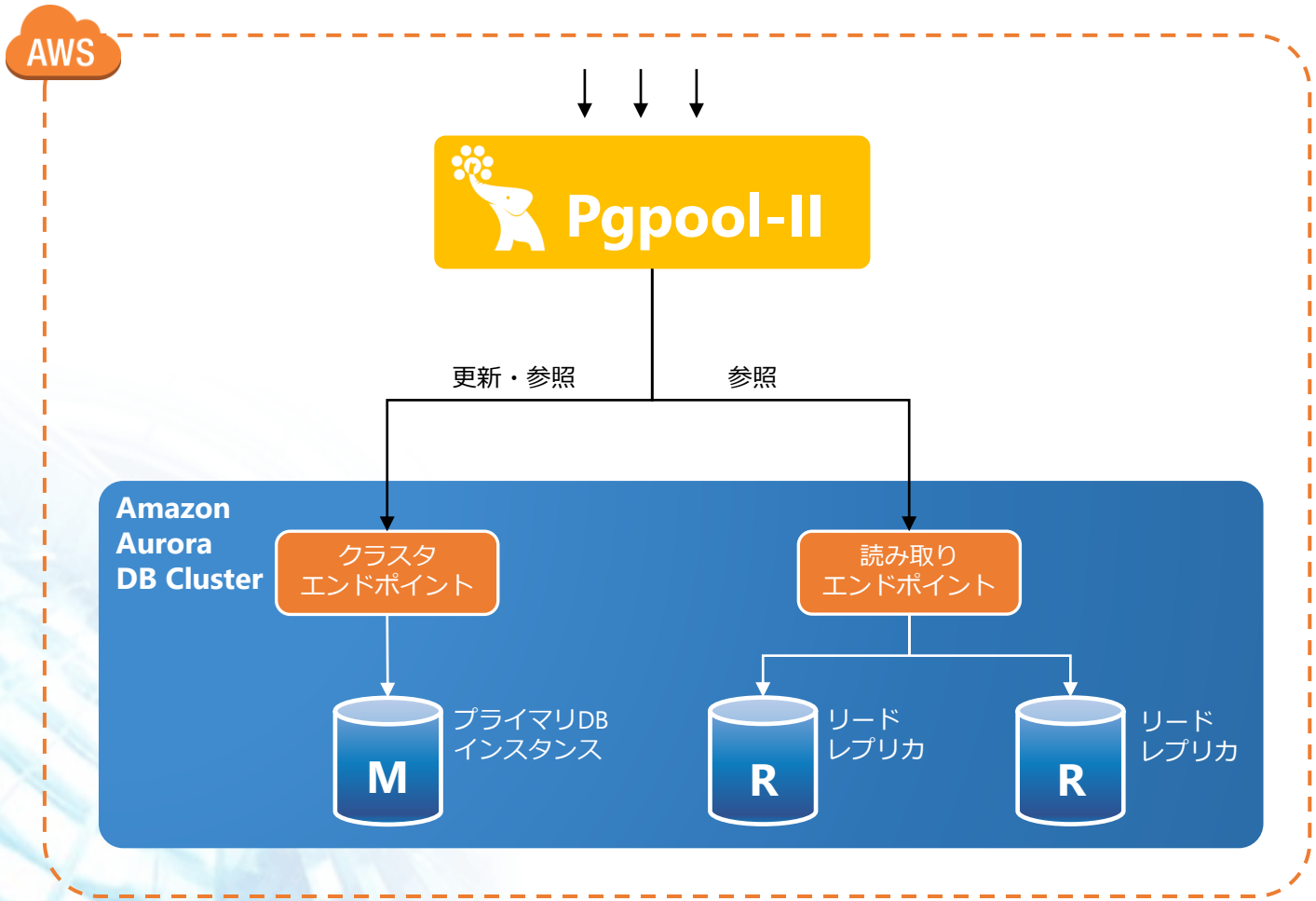
```
backend_hostname0='hippo'
backend_hostname1='hippo-replica'
backend_port0='5432'
backend_port1='5432'
...
```

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 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
```

Pgpool-II 4.2 の新機能

主な新機能・改良

- 設定と管理が容易になった
  - 設定パラメータの改善 (watchdog 関連、動作モード)
  - logger プロセスの実装
  - ユーザパスワードの一括登録
  - write_function_list と read_only_function_list の記述の省略
- 複数 PostgreSQL の間で読み取り一貫性の保証
  - snapshot isolation mode の導入
- LDAP 認証のサポート
- ヘルスチェックおよび発行 SQL の統計情報の取得
- ソースコードおよびドキュメントの用語の変更
- PostgreSQL 13 の SQL パーサの移植

write_function_list と read_only_function_list の記述の省略

Pgpool-II 4.1 以前

- 関数が書き込みを行うかを解析できない
- 書き込み関数を write_function_list に記述する必要がある

Pgpool-II 4.2 以降

- write_function_list と read_only_function_list が空文字の場合、システムカタログの情報を参照する
- VOLATILE 関数であれば、書き込みを行う関数と見なされ、負荷分散されない

各種統計情報の取得

SQL コマンドの統計情報

実行されたSQLコマンドの数と、バックエンドからエラーが返された回数を表示

```
test=# SHOW pool_backend_stats;
 node_id | hostname | port  | status |  role   | select_cnt | insert_cnt | update_cnt | delete_cnt | ddl_cnt | other_cnt | panic_cnt | fatal_cnt | error_cnt
---------+----------+-------+--------+---------+------------+------------+------------+------------+---------+-----------+-----------+-----------+----------
 0       | /tmp     | 11002 | up     | primary | 12         | 10         | 30         | 0          | 2       | 30        | 0         | 0         | 1
 1       | /tmp     | 11003 | up     | standby | 12         | 0          | 0          | 0          | 0       | 23        | 0         | 0         | 1
(2 rows)
```

ヘルスチェックの統計情報

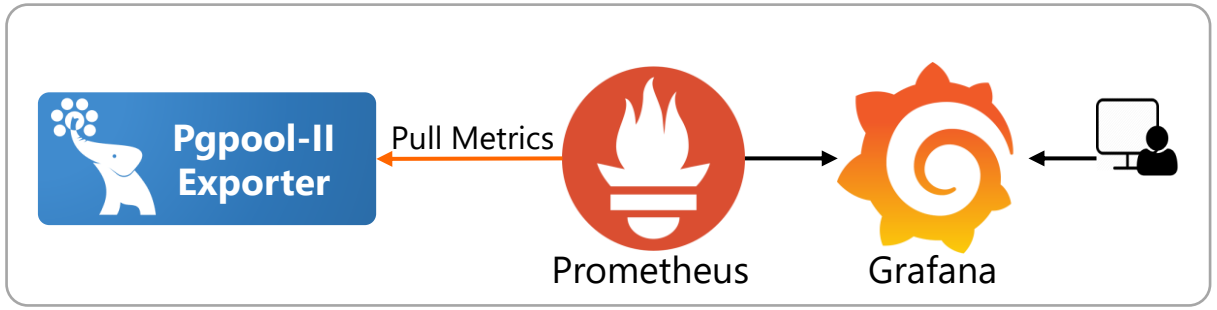
ヘルスチェックの統計情報を表示

```
test=# SHOW pool_health_check_stats;
 node_id | hostname | port  | status |  role   | last_status_change  | total_count | success_count | fail_count | skip_count | retry_count |
---------+----------+-------+--------+---------+---------------------+-------------+---------------+------------+------------+-------------+ ...
 0       | /tmp     | 11002 | up     | primary | 2020-11-09 01:53:19 | 1187        | 1187          | 0          | 0          | 0           |
 1       | /tmp     | 11003 | up     | standby | 2020-11-09 01:53:19 | 1187        | 1187          | 0          | 0          | 0           |
(2 rows)
```

各種統計情報を Prometheus フォーマットで出力

Pgpool-II Exporter

- Prometheus サーバからのリクエストに応じて、Pgpool-II の SHOW コマンドで取得した各種メトリクス情報を Prometheus フォーマットで出力
- https://github.com/pgpool/pgpool2_exporter



Prometheus

- サーバのリソース状況やサービスの各種メトリクスを収集して監視を行うモニタリングシステム
- Prometheus サーバが定期的に全ての Exporter に対してポーリングを行い様々な情報を収集 (Pull 型監視)
- 収集したデータを Prometheus 内の DB に保存
- アラート通知

Snapshot isolation mode とは

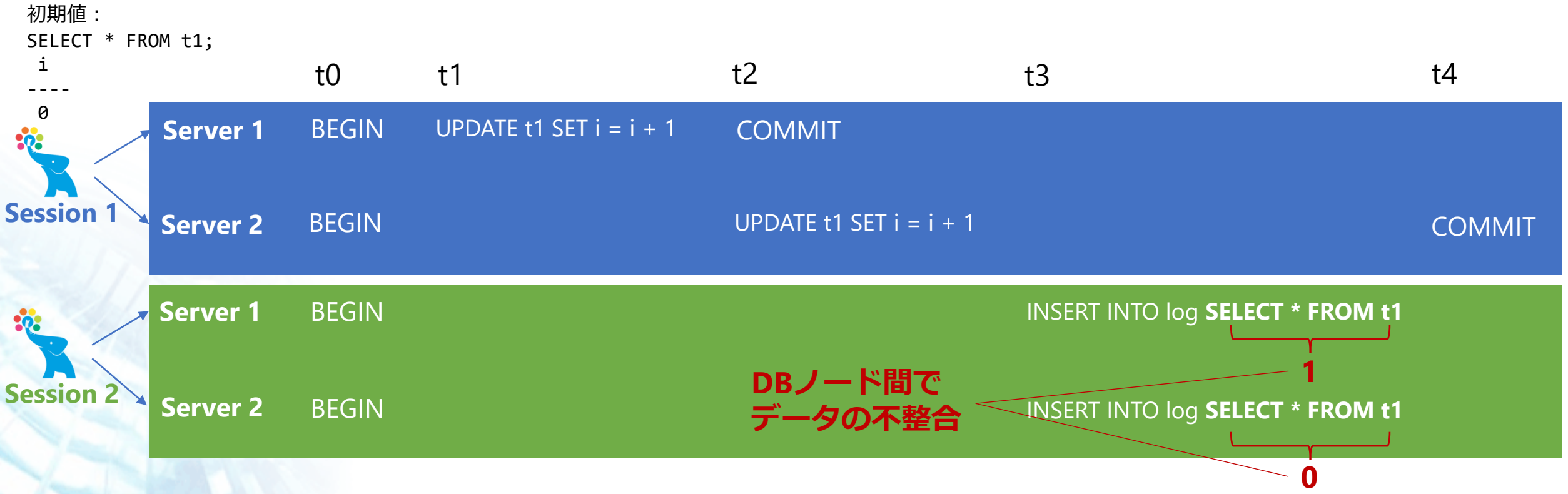
- Native replication mode の拡張
- スナップショットの管理機能を追加して、ノードをまたがる読み取りの一貫性を保証する

Pgpool-II の動作モード

- Streaming replication mode
  - PostgreSQL の streaming replication でレプリケーションを行うモード
- Native replication mode
  - Pgpool-II が各ノードで同じ SQL を実行することで、レプリケーションを行うモード

Native replication modeにおけるDBノード間でのデータ不整合問題

- タイミングによって、あるバックエンドではコミット済みのデータが読めるが、他のバックエンドではコミット前のデータが読み出されることがある



Snapshot isolation mode の仕組み

- Snapshot の管理機能を追加して、ノード間での読み取り一貫性を保証
- トランザクション内の最初の SQL コマンドでスナップショットを取得するときに COMMIT と排他制御する
- トランザクション分離レベルは **Repeatable read のみをサポート**
- 研究論文に基づいて実装されている
  - Pangea: An Eager Database Replication Middleware guaranteeing Snapshot Isolation without modification of Database Servers

Snapshot isolation mode の可能性

- PostgreSQL に一切手を入れることなく Global Snapshot Isolation が保証できる
  - 発表されている Global Snapshot Isolation を保証する方法は、すべて PostgreSQL の大幅な改造が必要
- サーバの PostgreSQL のメジャーバージョンが異なっていても構わない
- サーバは PostgreSQL 互換であれば利用可能
  - PowerGres Plus、Amazon RDS、EDB Postgres など

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
- 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。