SRA OSS, INC.

クラウド時代の Pgpool-II の活用

Kubernetes におけるクエリ負荷分散とモニタリング機能を備えた PostgreSQL クラスタの構築

OSC2020 Online/Fall
2020-10-23

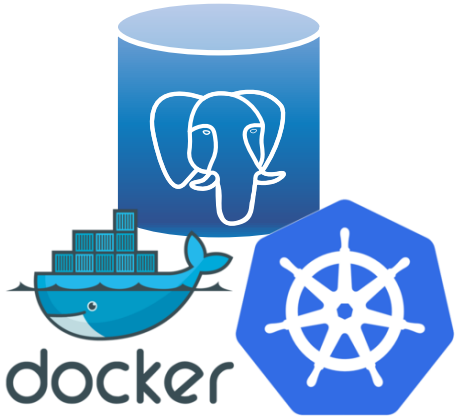
SRA OSS, Inc. 日本支社
彭 博 (ペン ボ)

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

概要

- Kubernetes の概要・特徴
- Kubernetes 上で PostgreSQL を運用するメリット
- Kubernetes 上で Pgpool-II によるクエリ振り分け
- PostgreSQL クラスタのモニタリング



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

- Kubernetes の歴史
  - Google 社内で10年以上利用されていた大規模コンテナクラスタ管理システムを元に作られた OSS
  - オープンソースとして公開 (2014)
  - Cloud Native Computing Foundation (CNCF) に移管 (Version 1.0、2015)
  - CNCF によって開発・メンテナンス (2015～)
- Kubernetes の利用環境
  - ローカルマシン: Minikube (学習・テスト目的)
  - オンプレミス/クラウド: 構築ツール (kubeadm など)
  - マネージド Kubernetes サービス
    - Amazon EKS
    - Google Kubernetes Engine (GKE)
    - Azure Kubernetes Service (AKS)

Kubernetes の特徴

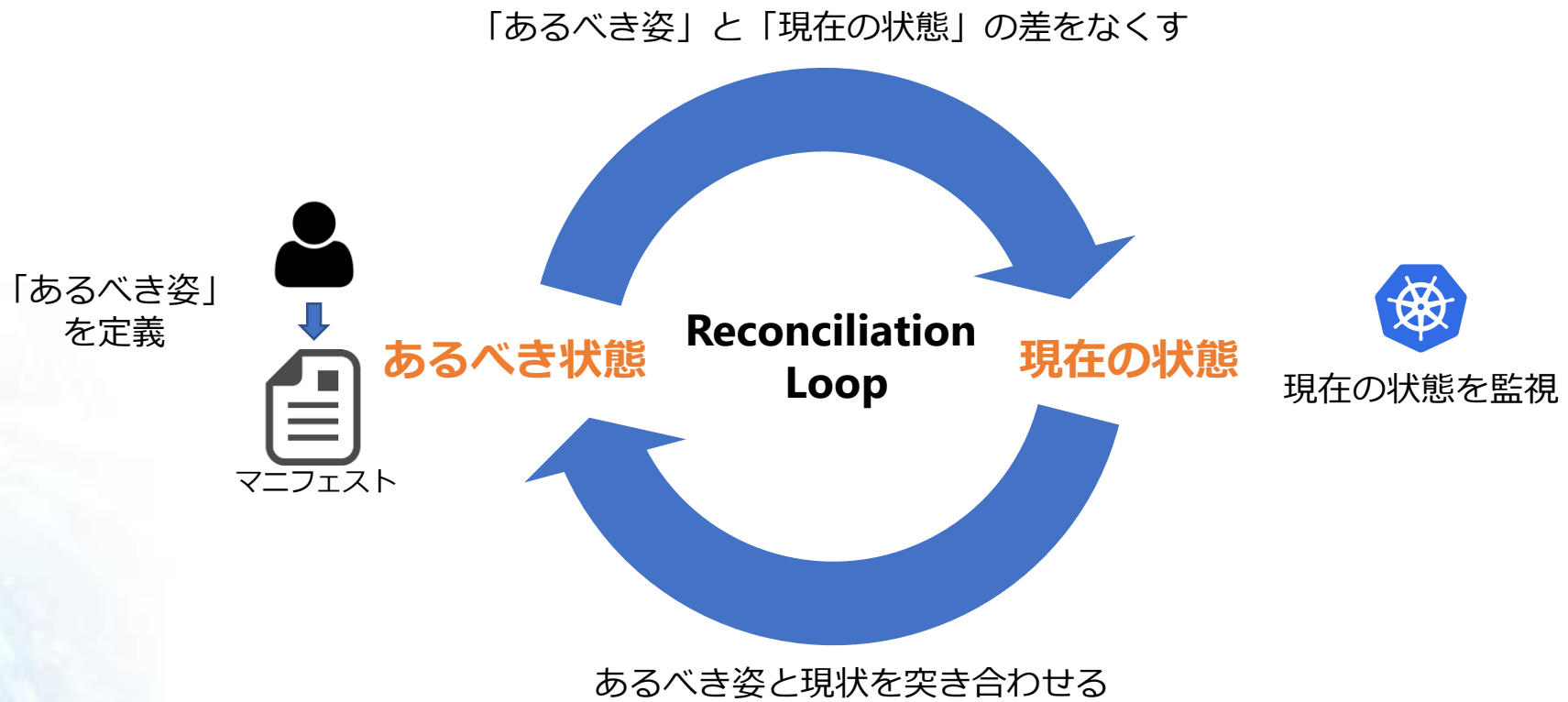
コンテナの死活監視	リソース管理	スケーリング	サービス ディスカバリ
セルフヒーリング	スケジューリング	ローリング アップデート	ロードバランシング

宣言的設定

Reconciliation Loop
(突合せループ)

Reconciliation Loop (突合せループ)

- 宣言的設定 (マニフェスト) により、あるべき姿を定義
- Kubernetes はシステムを定義された「あるべき姿」に収束させる



なぜ Kubernetes 上で PostgreSQL を動かすのか？

システム全体のプラットフォームの統一

- アプリケーションや Web サーバを Kubernetes 上で構築しているので、データベースも Kubernetes を使えば、システム全体の管理コストを軽減できる

Kubernetes や専用 Operator の機能を利用できる

- PostgreSQL HA クラスタ環境を自動的に構築・運用管理できる
- バックアップ・リカバリの自動化
- PostgreSQL の検証環境・テスト環境をすぐに構築・削除できる
- 負荷に応じて、レプリカの台数を増減可能
- 可用性・耐障害性の向上 (Multi-Cluster、Multi-Zone)
- Namespace により、PostgreSQL クラスタをサービスごとに隔離可能 (Multi-Tenant)
- 必要に応じて、ボリュームサイズ拡張可能
- PostgreSQL のバージョン管理が容易にできる

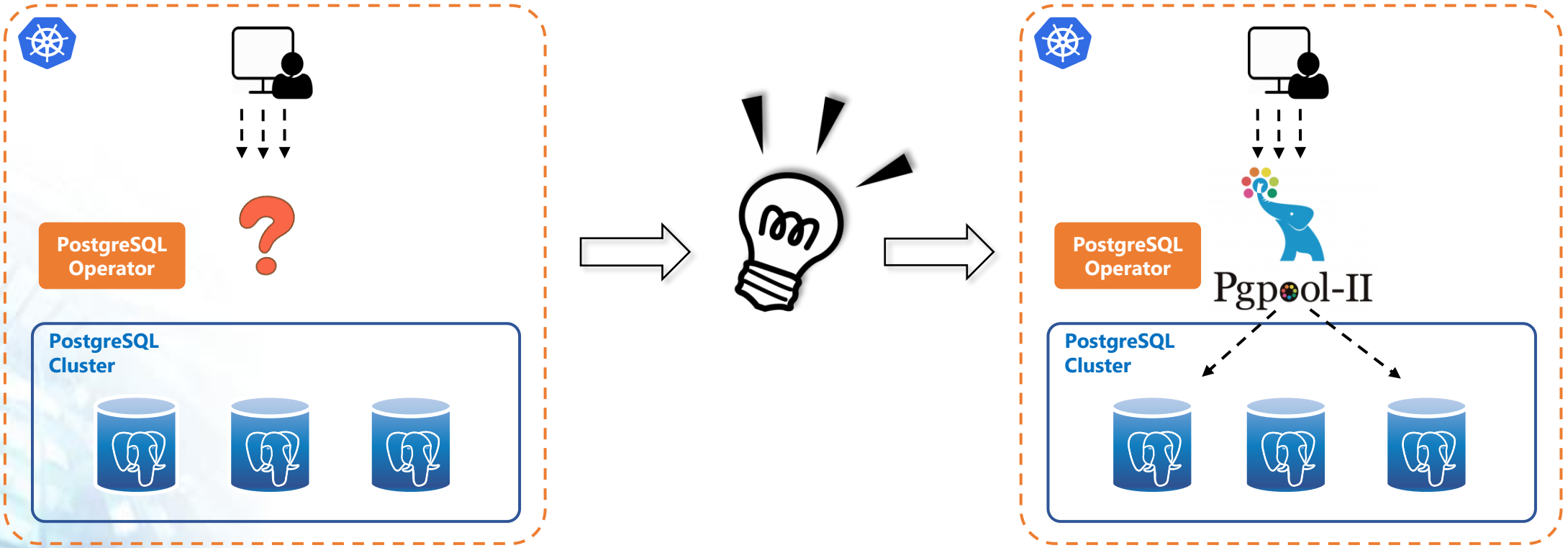
PostgreSQL Operator

- 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
  - 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 Cloud 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 上で 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

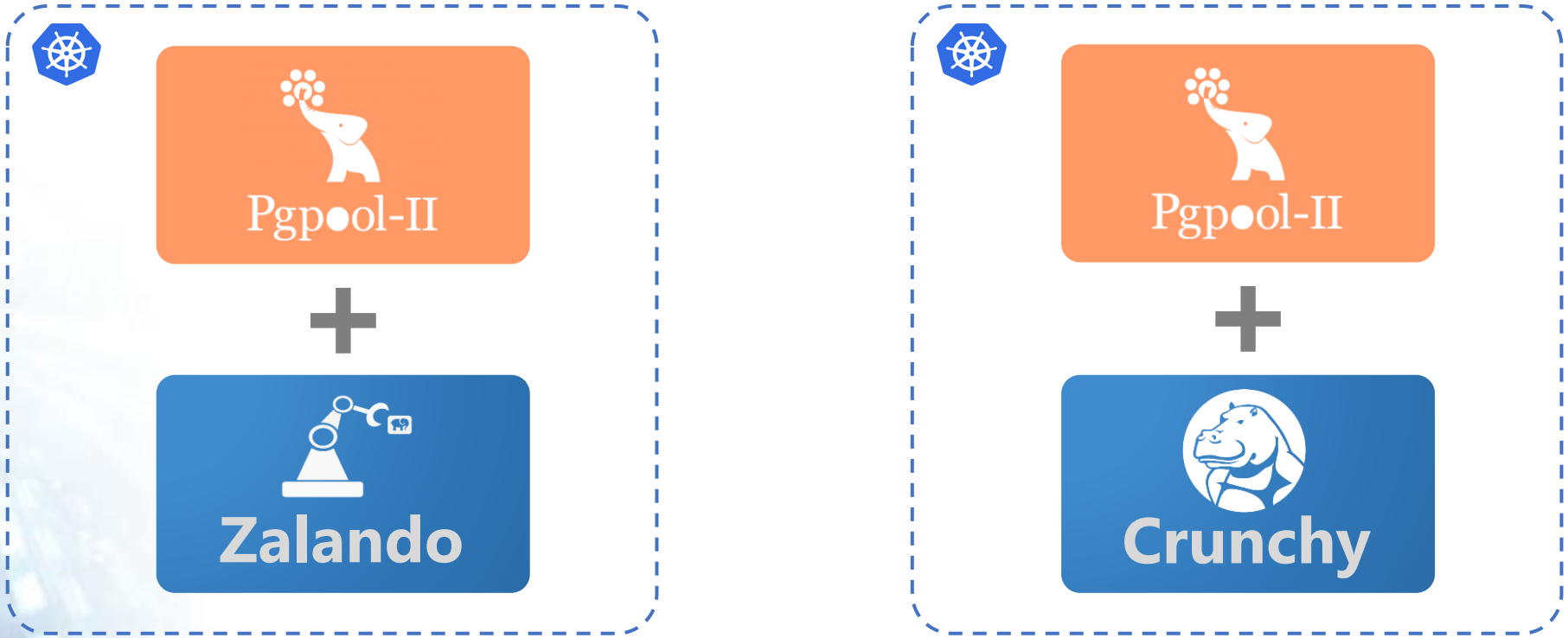


Pgpool-II とは

- PostgreSQL クラスタを管理するミドルウェア
  - PostgreSQL 7.4 以降
  - OSS, BSD license
  - https://pgpool.net/
- Pgpool-II の主な機能
  - クエリ振り分け
  - コネクションプーリング
  - 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)
  - インメモリキャッシュ

Kubernetes における Pgpool-II の動作環境

- マネージド Kubernetes サービス
  - Amazon EKS
  - Google Kubernetes Engine (GKE)



Kubernetes における Pgpool-II の設定

機能

クエリ振り分け
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'
```

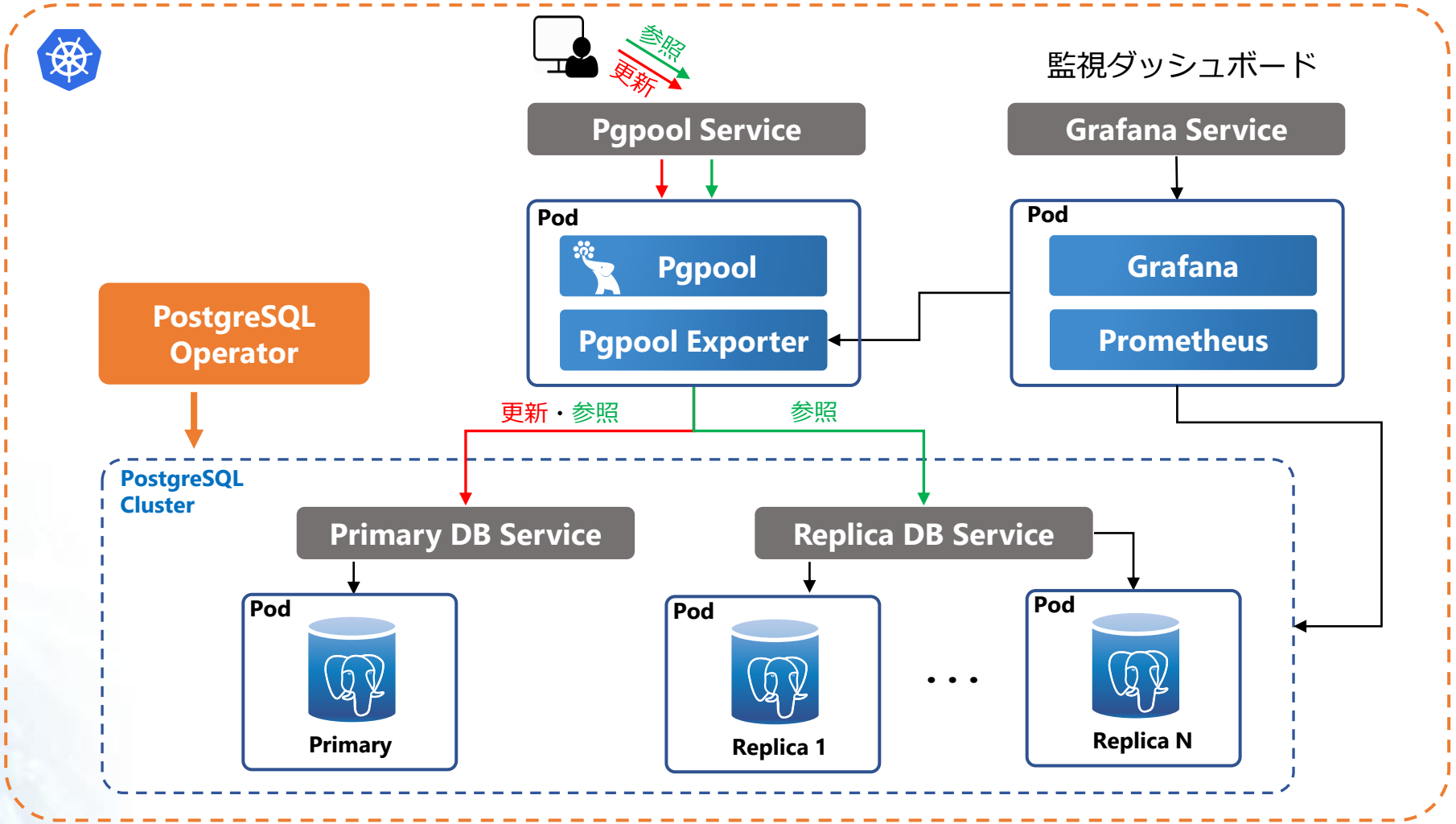
接続ユーザ

```
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

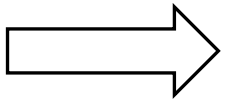
```
# pgpool_deploy.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: pgpool
spec:
  replicas: 1
...
spec:
      containers:
      - name: pgpool
        image: pgpool/pgpool:4.2
        env:
        - name: PGPOOL_PARAMS_BACKEND_HOSTNAME0
          value: "hippo"
        - name: PGPOOL_PARAMS_BACKEND_HOSTNAME1
          value: "hippo-replica"
        - name: PGPOOL_PARAMS_BACKEND_DATA_DIRECTORY0
          value: "/pgdata/hippo"
        - name: PGPOOL_PARAMS_BACKEND_DATA_DIRECTORY1
          value: "/pgdata/hippo"
...
      - 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

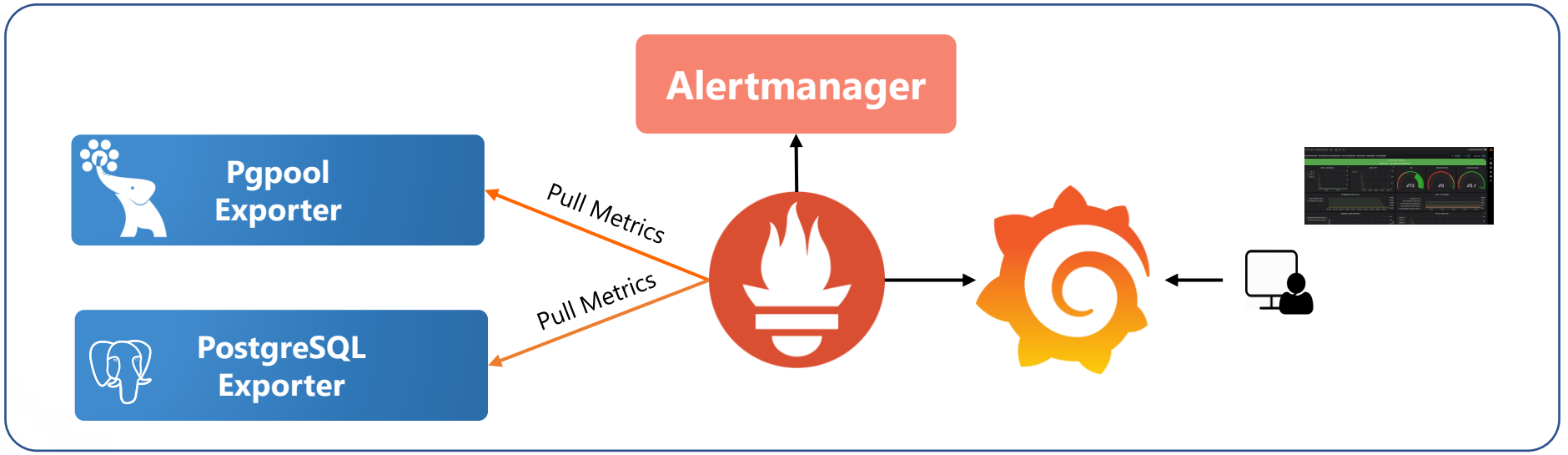
```
backend_hostname0='hippo'
backend_hostname1='hippo-replica'
backend_data_directory0='/pgdata/hippo'
backend_data_directory1='/pgdata/hippo'
...
```

クエリ負荷分散

```
$ 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"
  ?column?
 ----------
         1
 (1 row)
$ psql -h localhost -U postgres -c "SELECT 1"
  ?column?
 ----------
         1
 (1 row)
...
$ 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
```

モニタリングの仕組み



Prometheus

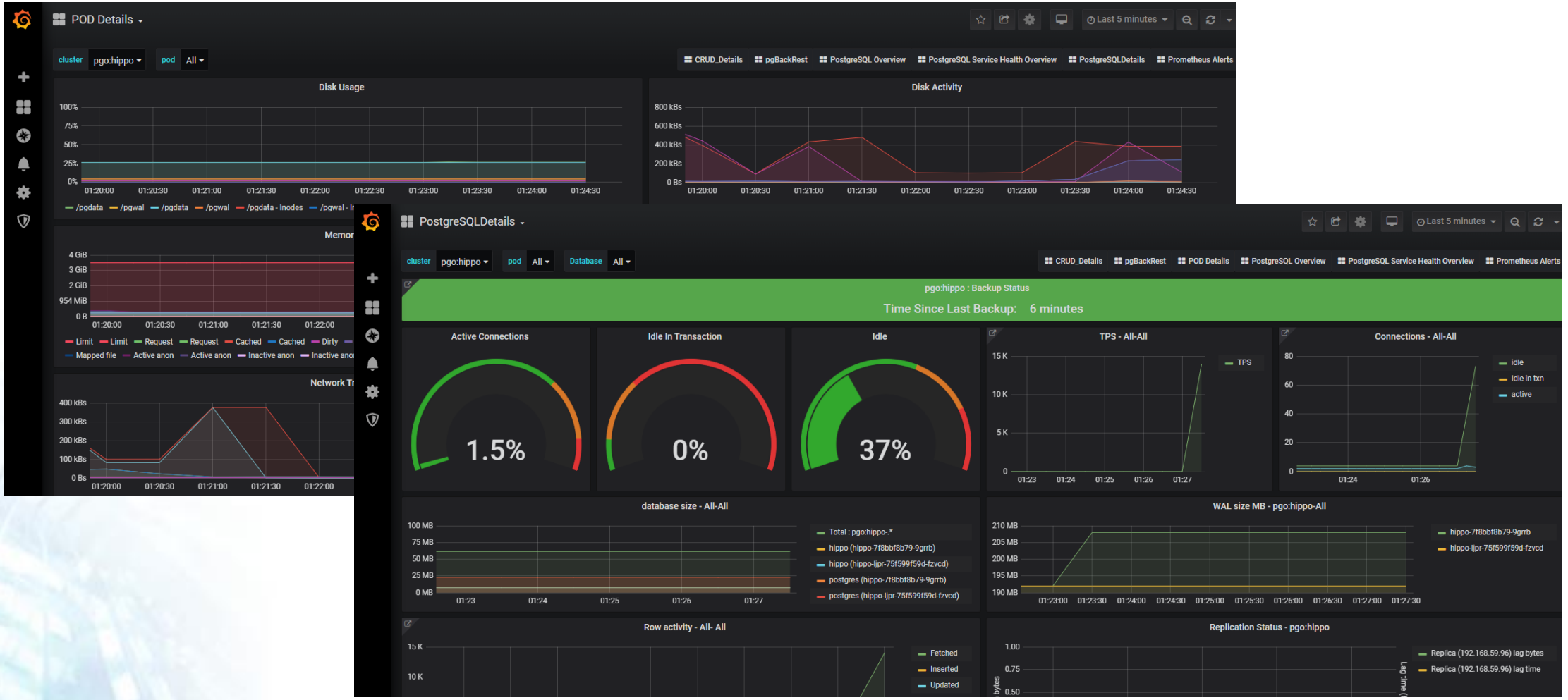
- サーバのリソース状況やサービスの各種メトリクスを収集して監視を行うモニタリングシステム
- Prometheus サーバが定期的に全ての Exporter に対してポーリングを行い様々な情報を収集 (Pull 型監視)
- 収集したデータを Prometheus 内の DB に保存
- アラート通知

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

Name	Description
pgpool2_frontend_total	起動中のプロセス数
pgpool2_frontend_used	使用中のプロセス数
pgpool2_pool_nodes_status	バックエンドノードの状態
pgpool2_pool_nodes_replication_delay	レプリケーション遅延
pgpool2_pool_backend_stats_select_cnt	実行された SELECT クエリの数
pgpool2_pool_backend_stats_insert_cnt	実行された INSERT クエリの数
pgpool2_pool_backend_stats_update_cnt	実行された UPDATE クエリの数
pgpool2_pool_backend_stats_delete_cnt	実行された DELETE クエリの数
pgpool2_pool_backend_stats_error_cnt	Error レベルメッセージの数
pgpool2_pool_backend_stats_fatal_cnt	Fatal レベルメッセージの数
pgpool2_pool_backend_stats_panic_cnt	Panic レベルメッセージの数
…	…

各種メトリクスの可視化



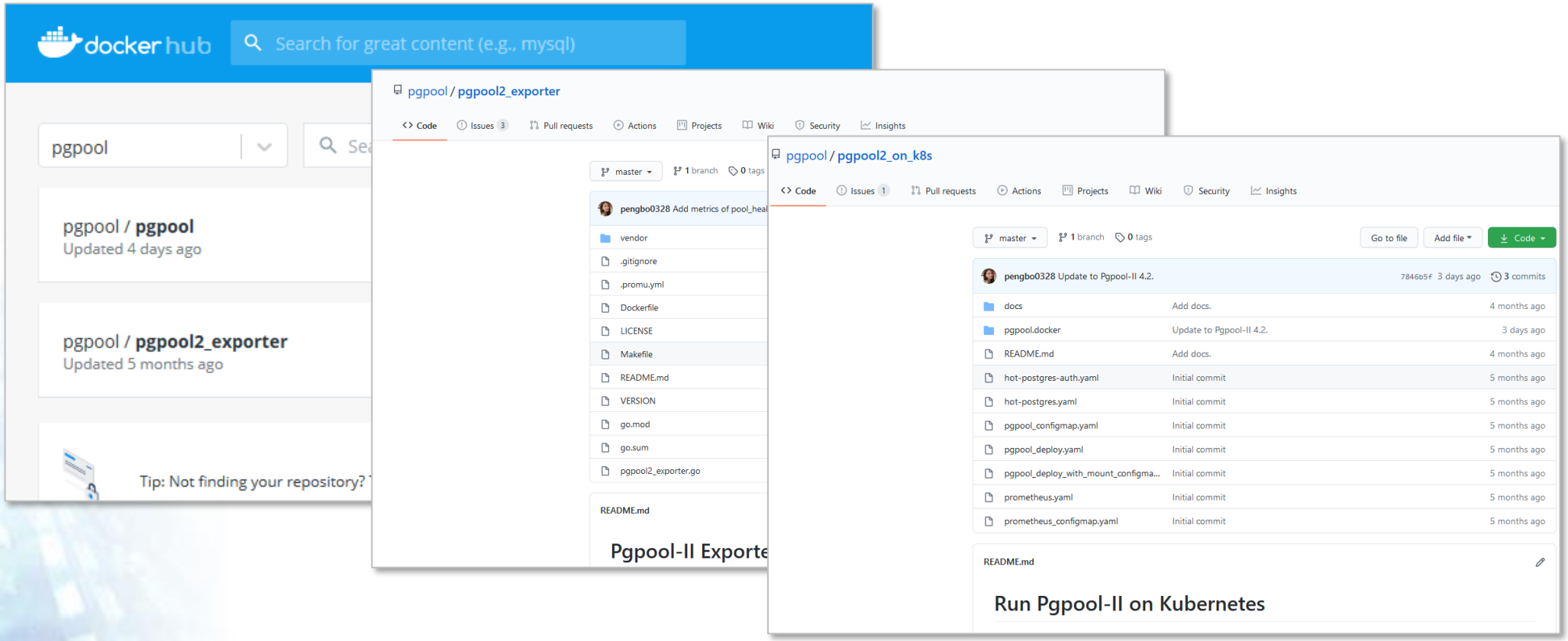
まとめ

- Kubernetes 上で PostgreSQL を運用することで多くのメリットを得られる
- 既存の PostgreSQL Operator にはクエリ振り分け機能がない
- Kubernetes 上で Pgpool-II を利用するメリット
  - クエリの振り分けができる
  - Pgpool-II を通じて PostgreSQL の統計情報を収集・可視化できる

Appendix

Kubernetes における Pgpool-II のデプロイ方法

https://github.com/pgpool/pgpool2_on_k8s
https://github.com/pgpool/pgpool2_exporter
https://hub.docker.com/u/pgpool



Pgpool-II Exporter の起動方法

```
[pengbo@localhost]$ git clone https://github.com/pgpool/pgpool2_exporter.git
[pengbo@localhost]$ cd pgpool2_exporter
[pengbo@localhost]$ make
[pengbo@localhost pgpool2_exporter]$ export
DATA_SOURCE_NAME="postgresql://pengbo:postgres@127.0.0.1:11000/postgres?sslmode=disable"

[pengbo@localhost pgpool2_exporter]$ ./pgpool2_exporter
INFO[0000] Starting pgpool2_exporter (version=0.0.0-dev, branch=master, revision=6419beee395419c5cb15fa12910681b229ff515d)
for postgresql://pengbo:postgres@127.0.0.1:11000/postgres?sslmode=disable  source="pgpool2_exporter.go:610"
INFO[0000] Listening on :9719                                source="pgpool2_exporter.go:611"

[pengbo@localhost ~]$ curl -s localhost:9719/metrics | grep pgpool2_pool_backend_stats
# HELP pgpool2_pool_backend_stats_ddl_cnt DDL statement counts issued to each backend
# TYPE pgpool2_pool_backend_stats_ddl_cnt gauge
pgpool2_pool_backend_stats_ddl_cnt{hostname="/tmp",port="11002",role="main"} 0
pgpool2_pool_backend_stats_ddl_cnt{hostname="/tmp",port="11003",role="replica"} 0
# HELP pgpool2_pool_backend_stats_delete_cnt DELETE statement counts issued to each backend
# TYPE pgpool2_pool_backend_stats_delete_cnt gauge
pgpool2_pool_backend_stats_delete_cnt{hostname="/tmp",port="11002",role="main"} 0
pgpool2_pool_backend_stats_delete_cnt{hostname="/tmp",port="11003",role="replica"} 0
# HELP pgpool2_pool_backend_stats_error_cnt Error message counts returned from backend
# TYPE pgpool2_pool_backend_stats_error_cnt gauge
pgpool2_pool_backend_stats_error_cnt{hostname="/tmp",port="11002",role="main"} 0
pgpool2_pool_backend_stats_error_cnt{hostname="/tmp",port="11003",role="replica"} 0
...
```

参考情報

- Pgpool-II
  - https://pgpool.net/
  - https://www.pgpool.net/docs/latest/ja/html/
- Kubernetes
  - https://kubernetes.io/ja/docs/home/
- Zalando PostgreSQL Operator
  - https://postgres-operator.readthedocs.io/en/latest/
  - https://github.com/zalando/postgres-operator
- Crunchy PostgreSQL Operator
  - https://access.crunchydata.com/documentation/postgres-operator/latest/
  - https://github.com/CrunchyData/postgres-operator

ご清聴ありがとうございました。