Kubernetes を用いた PostgreSQL の運用

db tech showcase 2020
2020-11-11

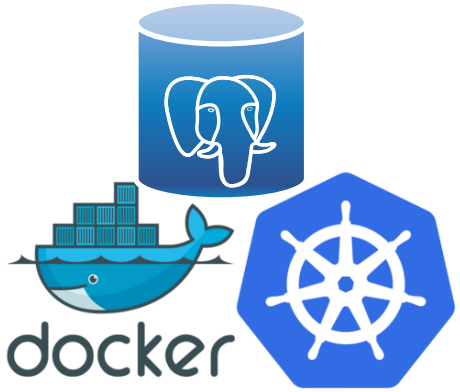
SRA OSS, Inc. 日本支社
彭 博 (ペン ボ)

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
- PostgreSQL Operator とは
- 各 PostgreSQL Operator の機能紹介及び活用例



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～)
- 大規模クラスタをサポート

https://kubernetes.io/docs/setup/best-practices/cluster-large/

項目	サポート対象
ノード数	5000 台以下
Pod 数	150,000 以下
コンテナ数	300,000 以下
1ノードあたりの pod 数	100 以下

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

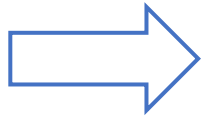
宣言的設定

Reconciliation Loop
(突合せループ)

宣言的設定

- 宣言的設定 (マニフェスト) により、あるべき姿を定義

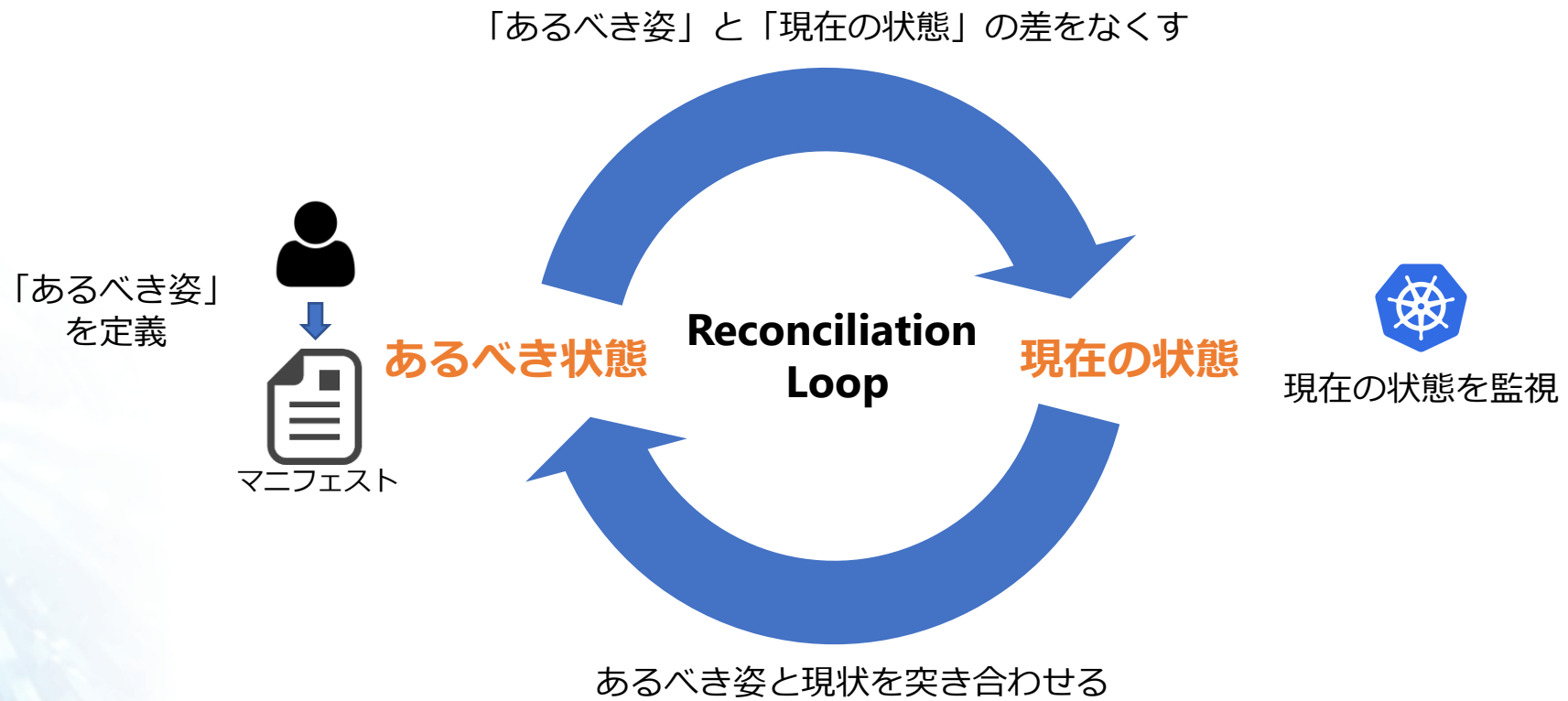
```
# マニフェスト
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```



- 構築や管理の自動化
- 手作業によるミスがなくなる
- 大規模インフラに対してスケーラビリティを持てる
- インフラ構成をマニフェストで定義でき、管理しやすい
- 問題があったときに更新履歴を使って過去の状態に戻せる

Reconciliation Loop (突合せループ)

- ユーザがシステムの「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



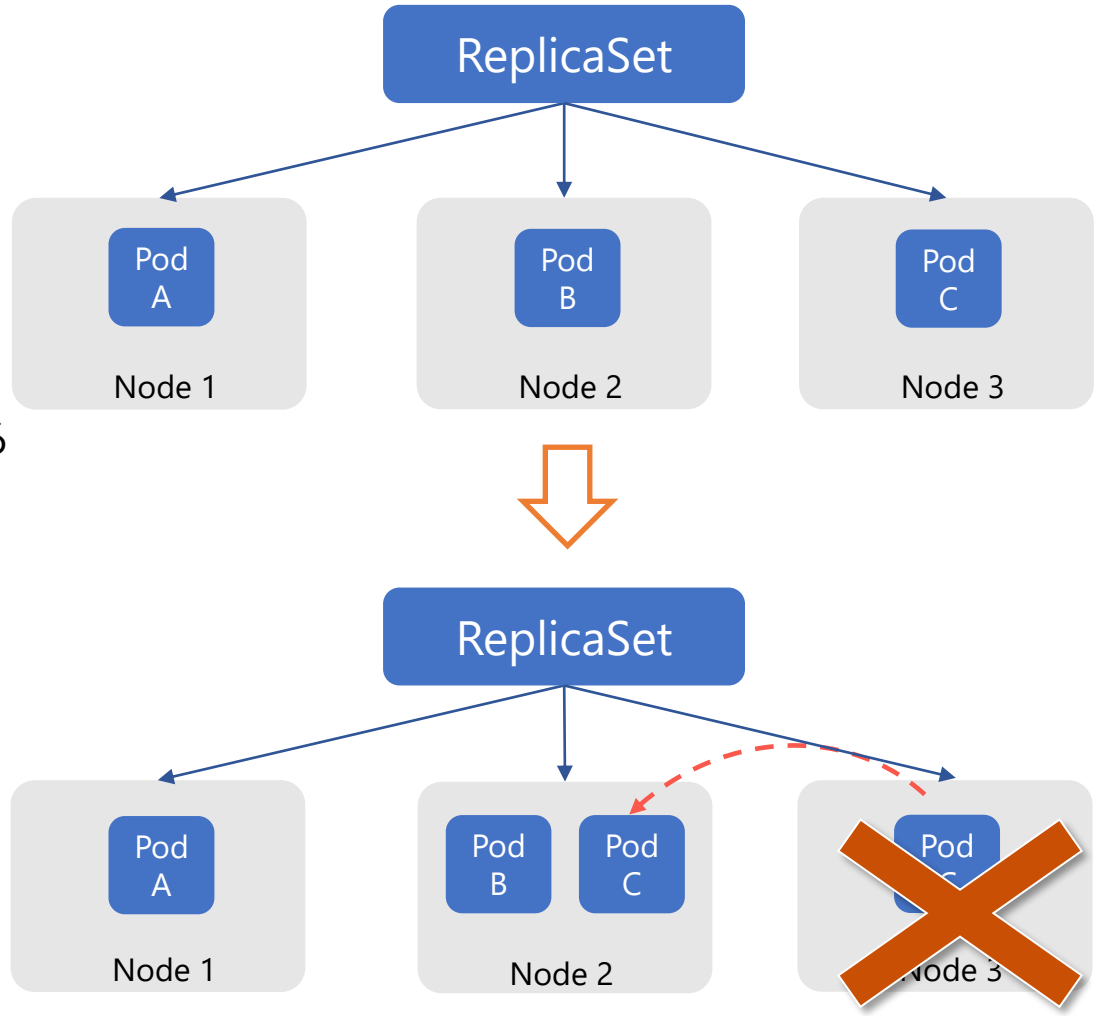
Reconciliation Loop (突合せループ)

```
# マニフェスト
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: sample-rs
spec:
  replicas: 3
  selector:
    matchLabels:
      app: sample-app
  template:
    metadata:
      labels:
        app: sample-app
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

定義された Pod の数を維持し続ける

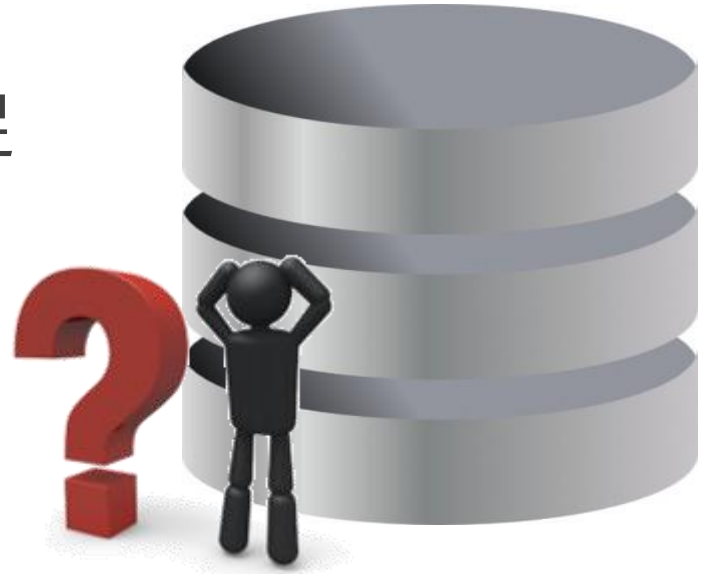
セルフヒーリング

- Pod 障害時に Pod の数が定義された数を満たすように Pod を再作成
- ノード障害時にも定義された数を維持するように別のノードで Pod を起動



これまでのデータベース管理者の悩み

- PostgreSQL クラスタ環境を構築するのは面倒
- テスト・開発環境の用意が大変
- 監視、バックアップ、障害対策など、運用の手間がかかる
- 負荷の急な増加に即座に対応できない
- データ量増加に伴うディスク容量の不足



Kubernetes 上で PostgreSQL を運用するメリット

- 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

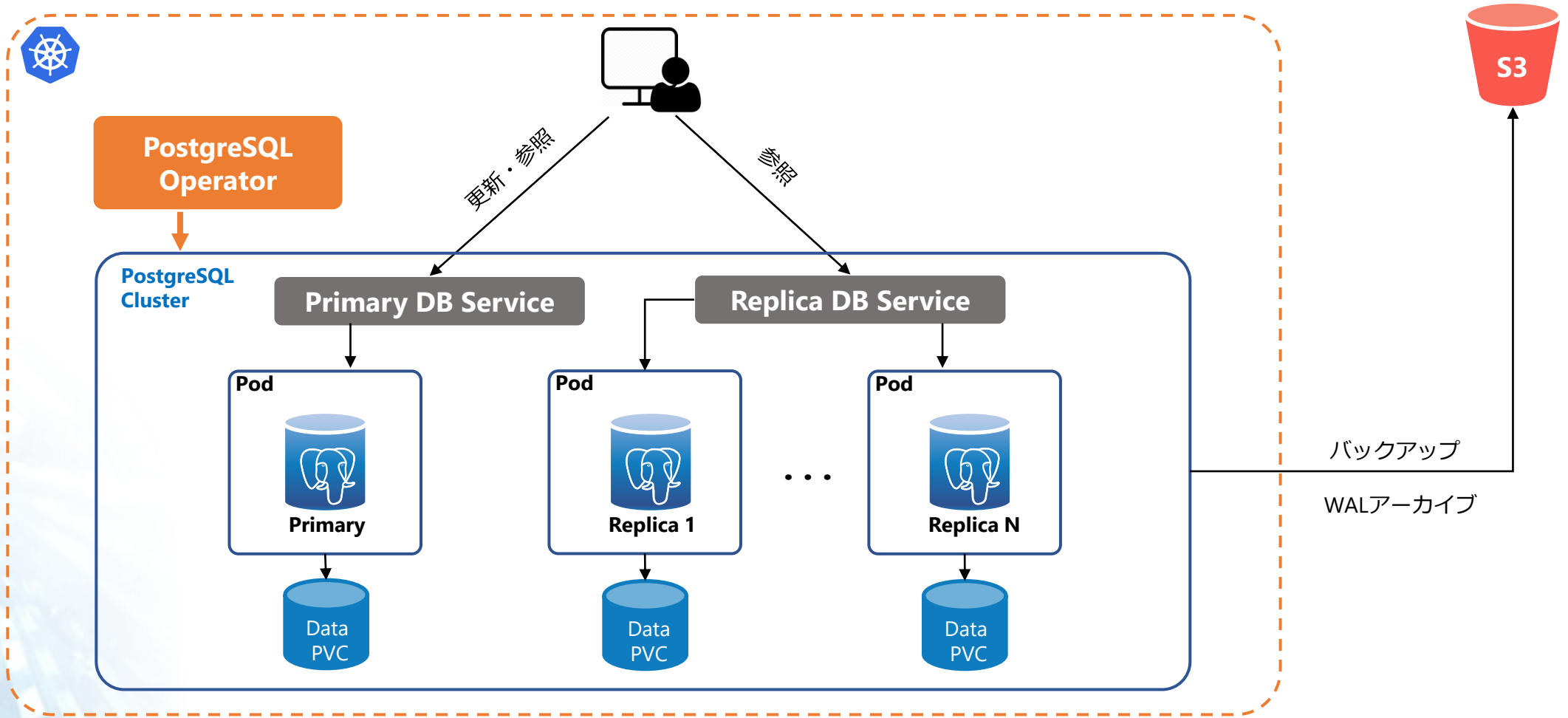
PostgreSQL Operator とは

- 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 の管理タスクを自動化
    - バックアップ
    - 障害時の対応
    - 監視など
  - PostgreSQL クラスタの Primary/Standby の役割管理

PostgreSQL Operator: Zalando & Crunchy

	 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+	PostgreSQL 9.5+
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

基本構成図



PostgreSQL Operator の機能紹介及び活用例

機能 (1) : PostgreSQL クラスタの自動構築

	Zalando	Crunchy
自動デプロイ	○	○
専用 CLI 対応	×	○
GUI 管理インターフェイスの有無	○	×

機能 (1) : PostgreSQL クラスタの自動構築

```
# マニフェスト
apiVersion: "acid.zalan.do/v1"
kind: postgresql
metadata:
  name: acid-minimal-cluster
  namespace: default
spec:
  teamId: "acid"
  volume:
    size: 1Gi
  numberOfInstances: 3
```

Deploy
replicas: 3

Deploy
replicas: 3

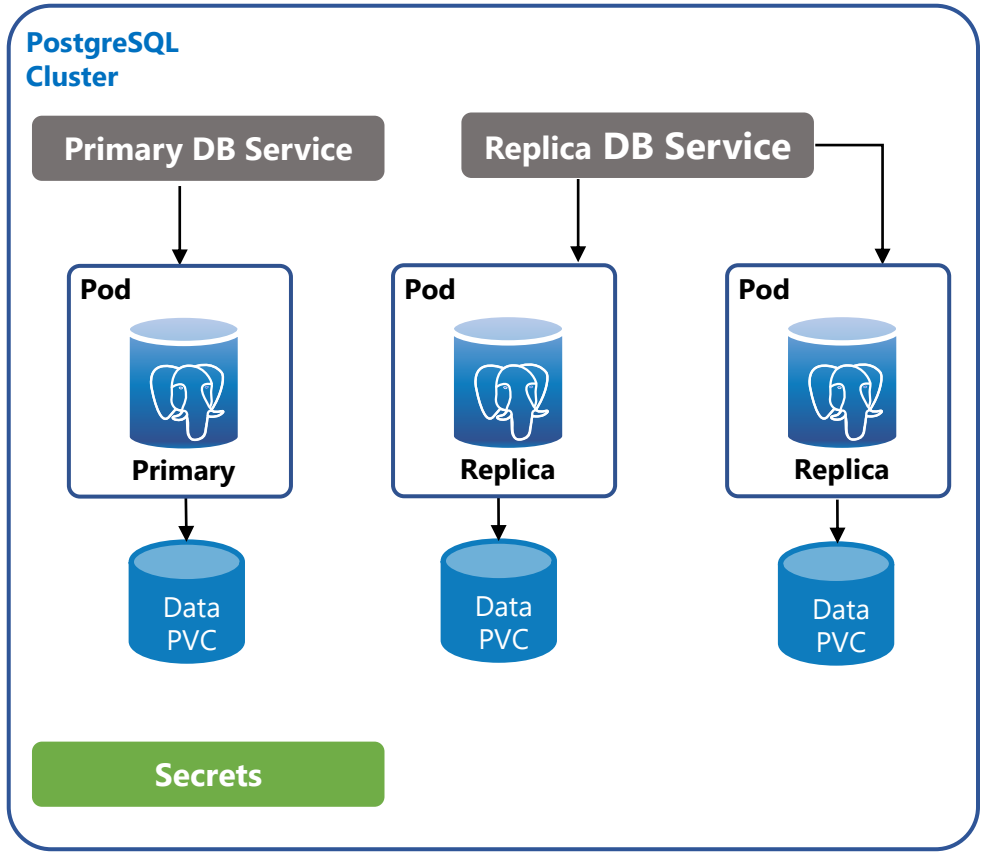
PostgreSQL Operator

Service を作成

PostgreSQL Pod を作成

PVC を作成

Secret を作成

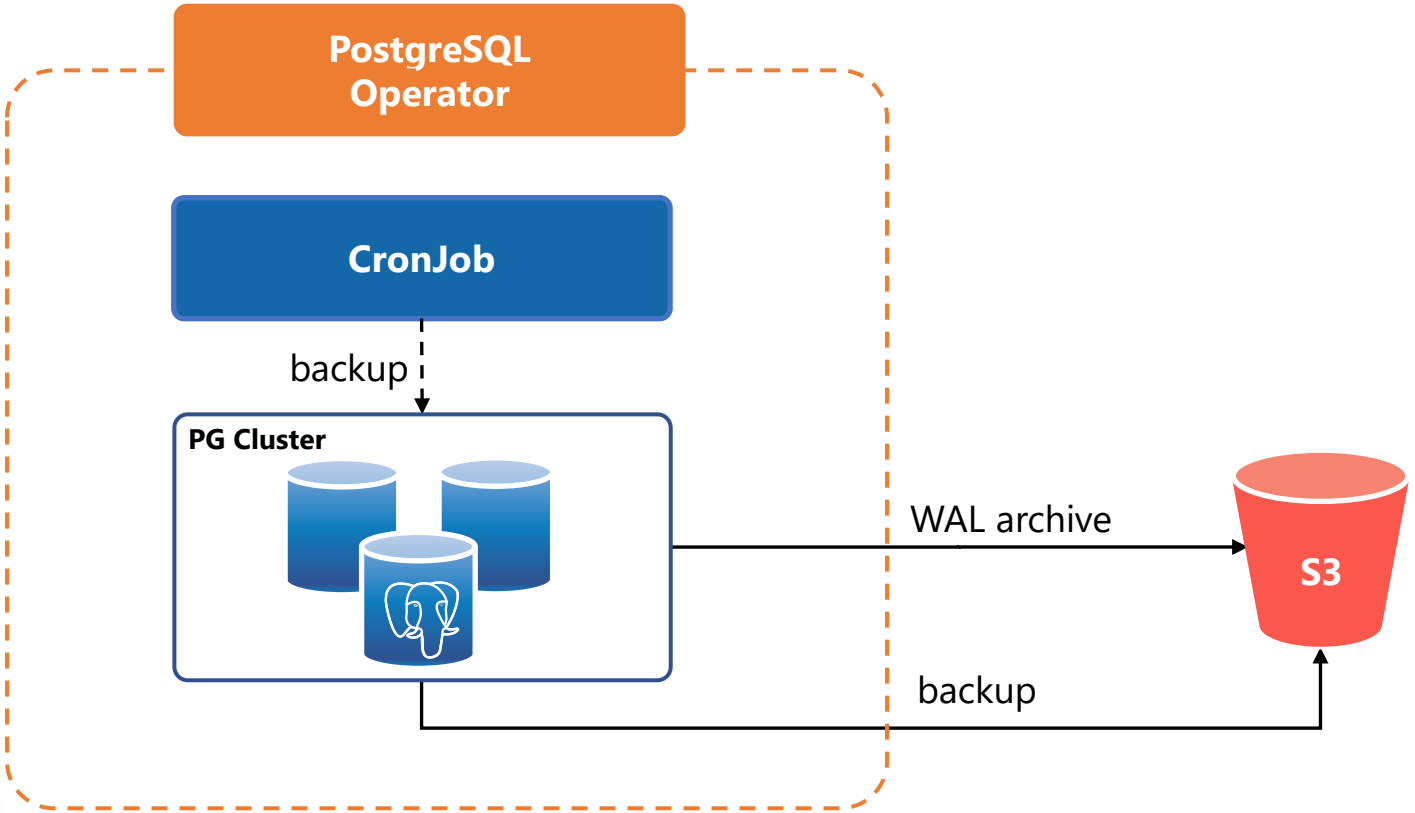


機能 (2) :バックアップ・リカバリの自動化

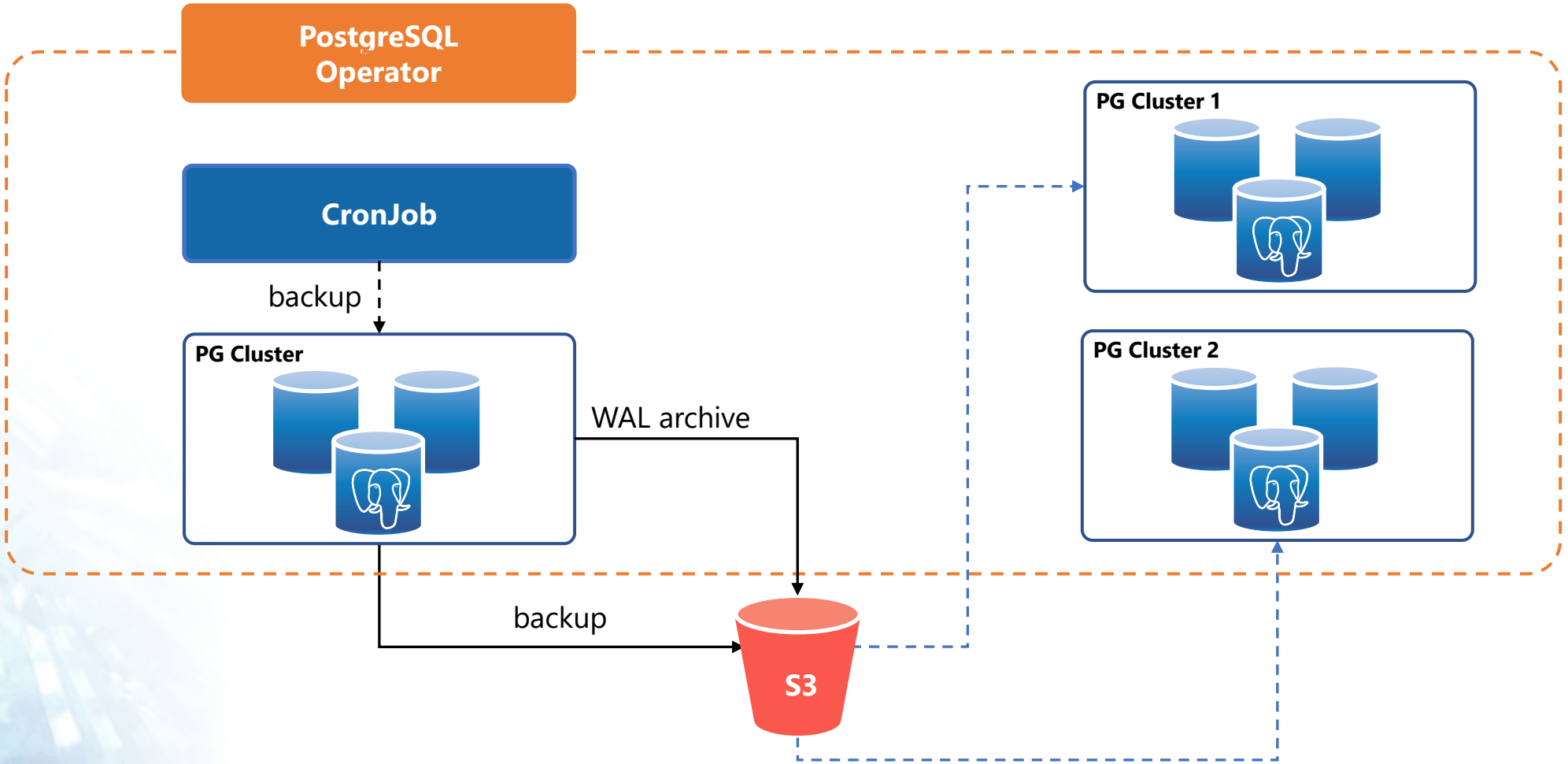
	Zalando	Crunchy
バックアップ	○ ※S3対応	○ ※S3対応
リストア	△ ※手動リストアが必要	○
PITR	○	○

活用例: 定期バックアップ

- 指定した時刻にバックアップを自動的に行う
- 取得したバックアップや WAL アーカイブは Amazon S3 に保存できる



活用例: バックアップから複数テスト環境を作成



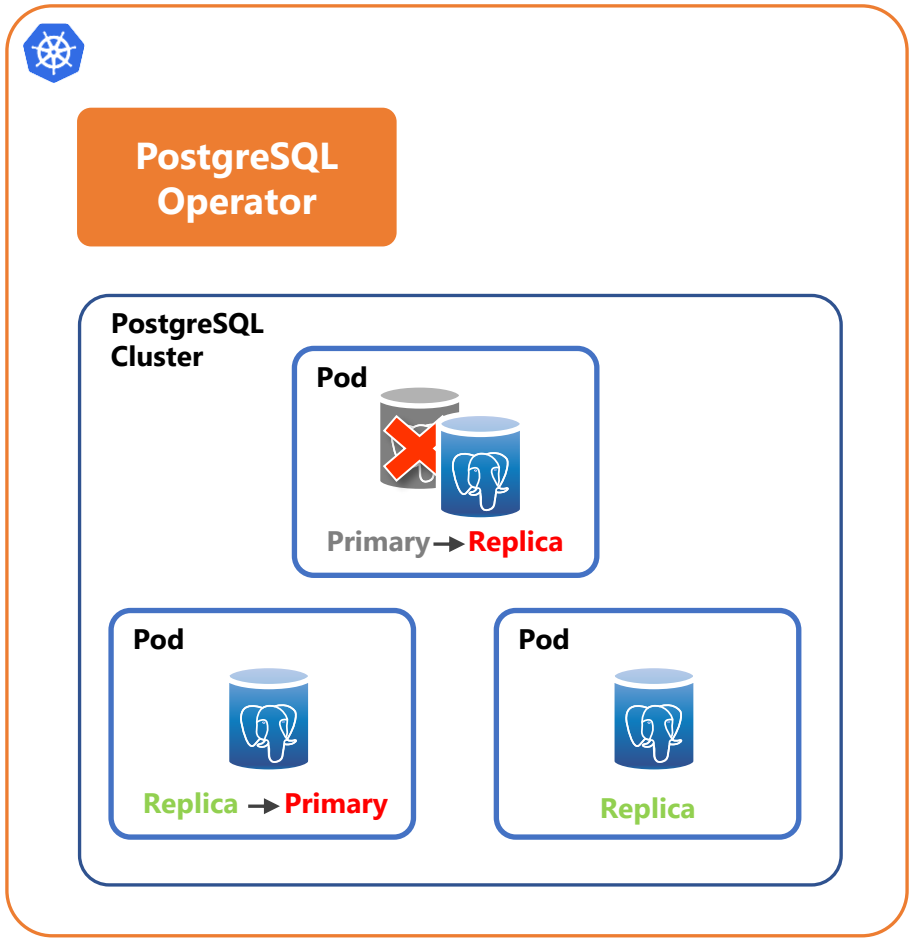
機能 (3): 可用性・耐障害性の向上

	Zalando	Crunchy
自動フェイルオーバ	○	○
自動フェイルバック	○	○
Multi-Zone	○	○
Multi-Kubernetes クラスタ	○	○

活用例: 高可用性・耐障害性 - クラスタ内フェイルオーバ

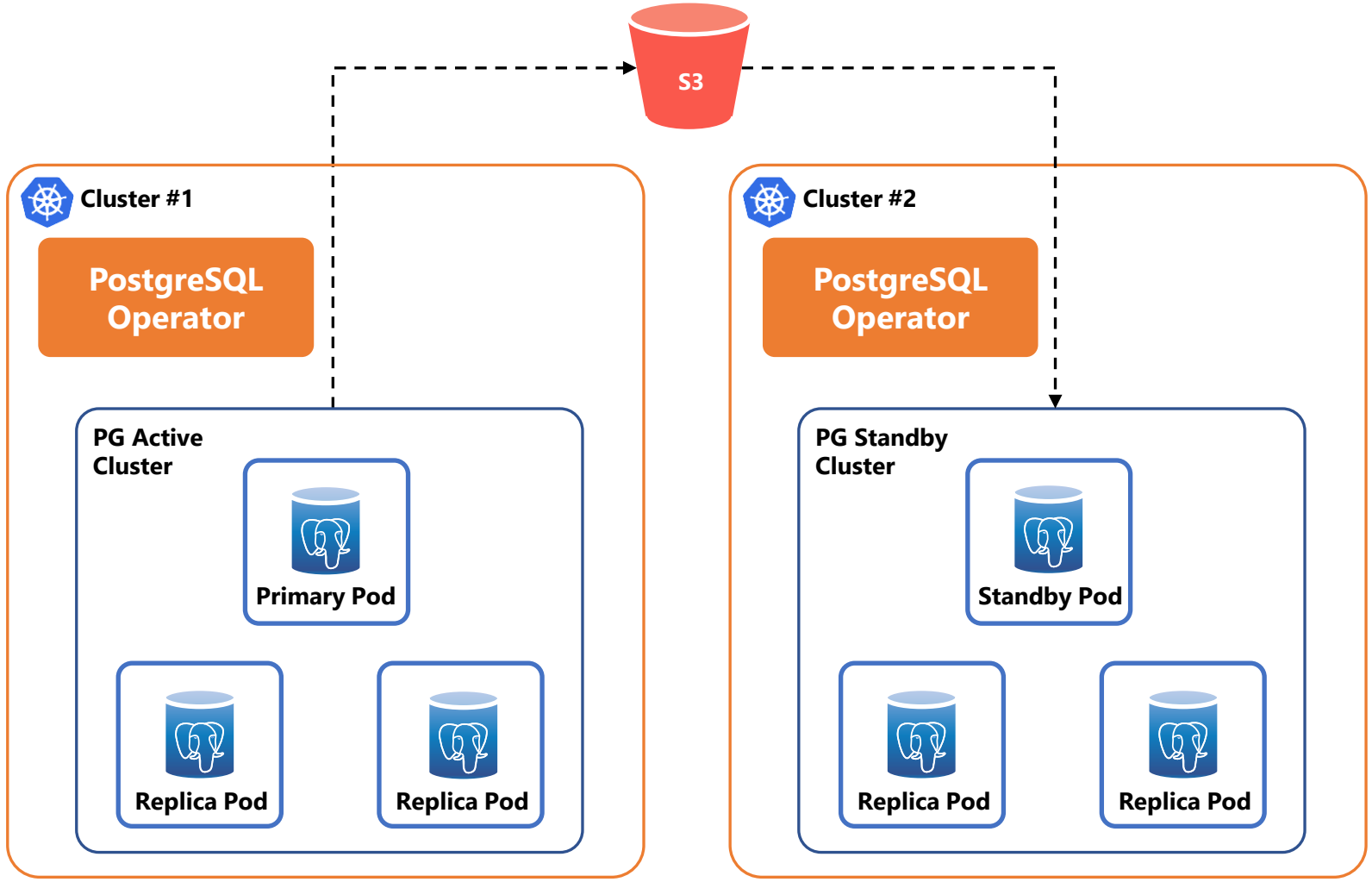
Primary Pod に障害が発生した場合

- Replica が自動的に Primary に昇格
- 新しい Replica Pod が作成される



活用例: 高可用性・耐障害性 - クラスタ間フェイルオーバ

- 2つの Kubernetes クラスタを用意
- Cluster #1
  - PostgreSQL Active Cluster
  - S3 にバックアップとWALアーカイブを保存
- Cluster#2
  - PostgreSQL Standby Cluster
  - S3 から変更を適用
- Cluster #1 に障害が発生した場合
  - Standby Cluster が Active Cluster に昇格可能 (手動昇格)
  - 旧 Active Cluster は Standby Cluster として復旧可能 (手動操作)

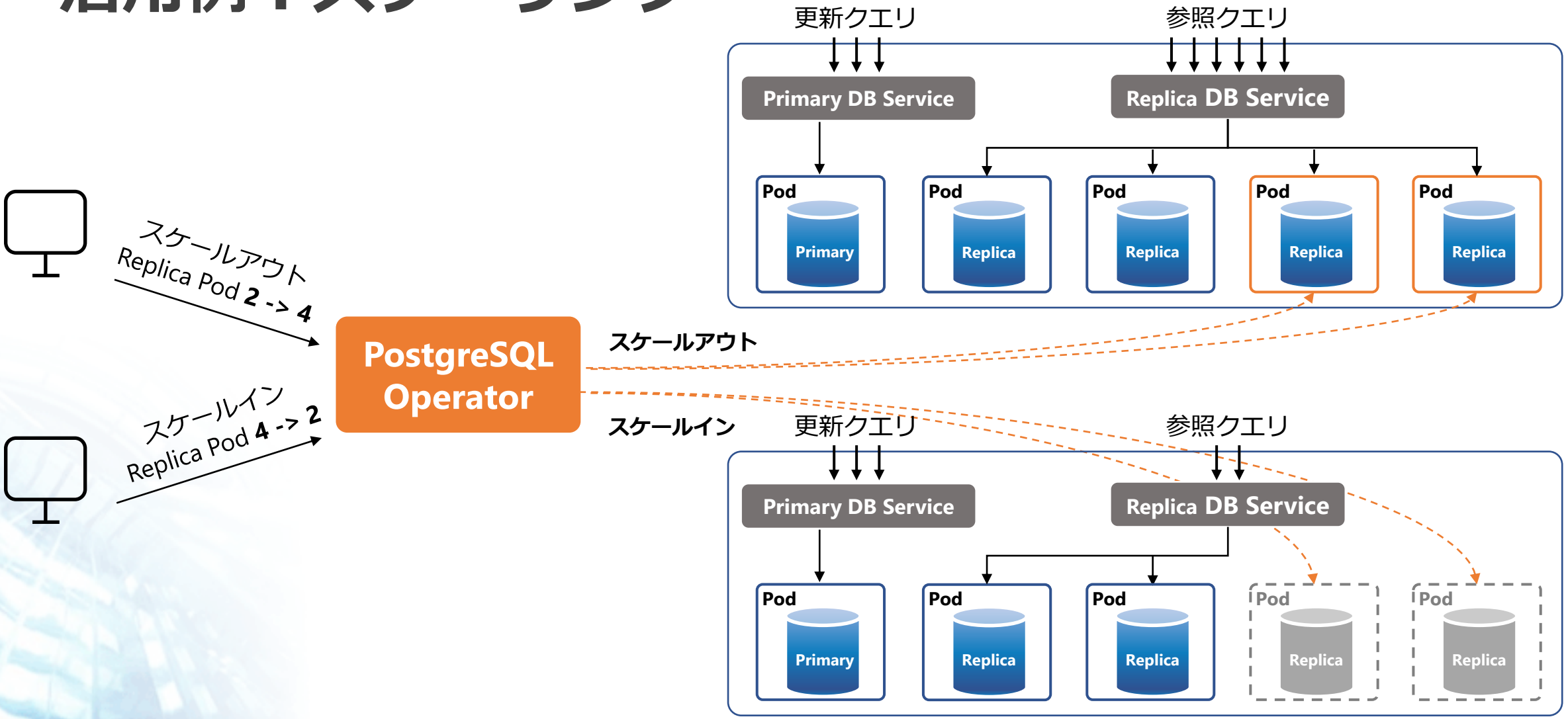


機能 (4)：スケーリング

- スケールアウト/スケールイン
  - データベースサーバの台数を増加/減少させる
- レプリカを増やすことで、複数のレプリカへ参照クエリを分散させる

	Zalando	Crunchy
スケールイン	○	○
スケールアウト	○	○

活用例：スケーリング



機能 (5)：ロールベースのアクセス制御

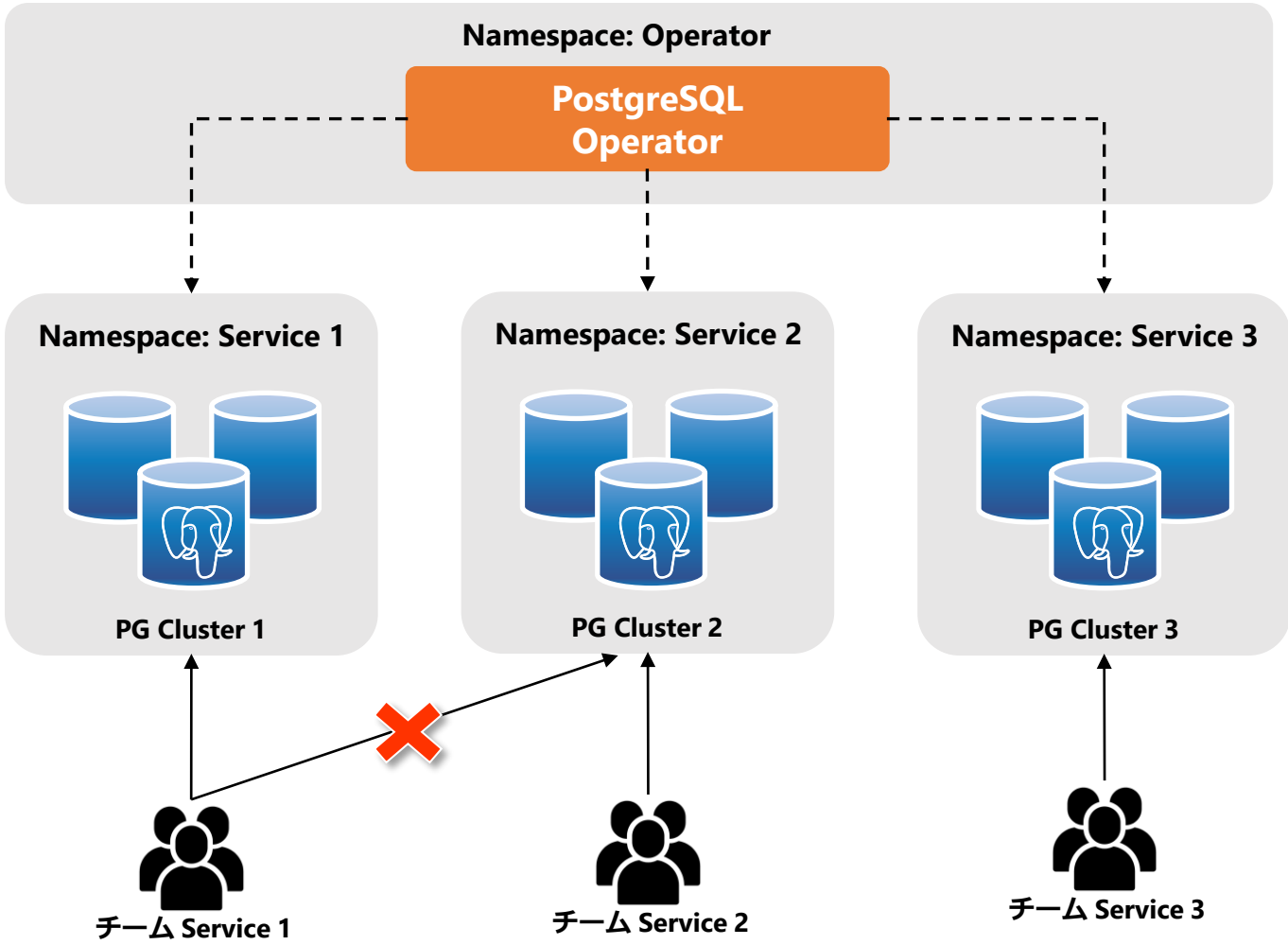
- 複数チーム・サービスが同一 Kubernetes クラスタを利用している場合は、サービスへのアクセス管理が必要になる
- RBAC を設定することで、Kubernetes のアカウントに対して特定の Namespace へのアクセス権限を制御する
  - RBAC：ロールベースアクセス制御
  - Namespace：サービスごとに分割する方法

	Zalando	Crunchy
Single-Namespace	○	○
Multi-Namespace	○	○

活用例： Namespace で PostgreSQL をサービスごとに分離

Multi-Tenant

- 複数 Namespace を管理可能
- Namespace ごとにアクセス制限を設定可能
- 信頼性、安全性を向上させる

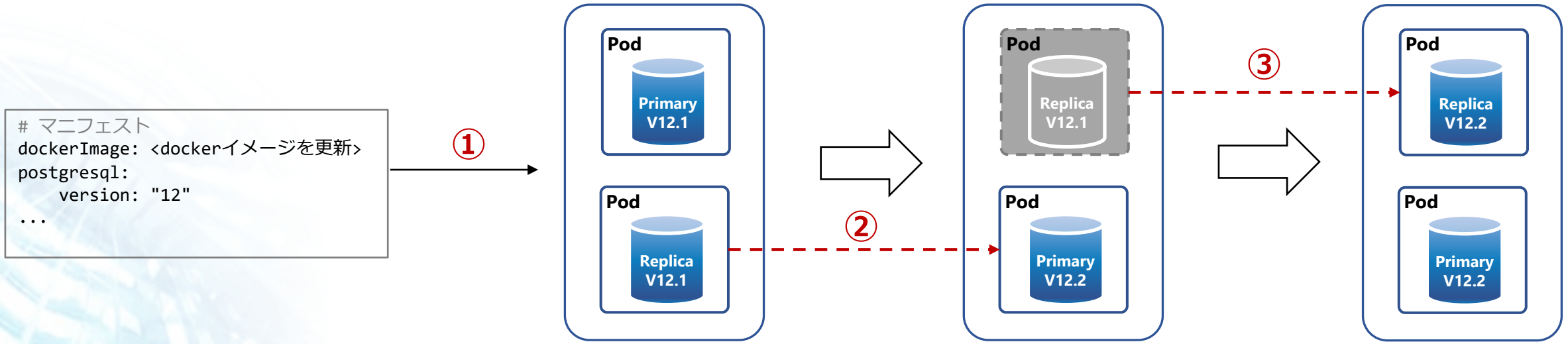


機能 (6)：自動バージョンアップ

	Zalando	Crunchy
自動バージョンアップ	○	○
ローリングアップデート	○ ※バージョンアップに伴う停止時間なし	× ※バージョンアップに伴う停止時間あり

活用例： ローリングアップデートによるマイナーバージョンアップ

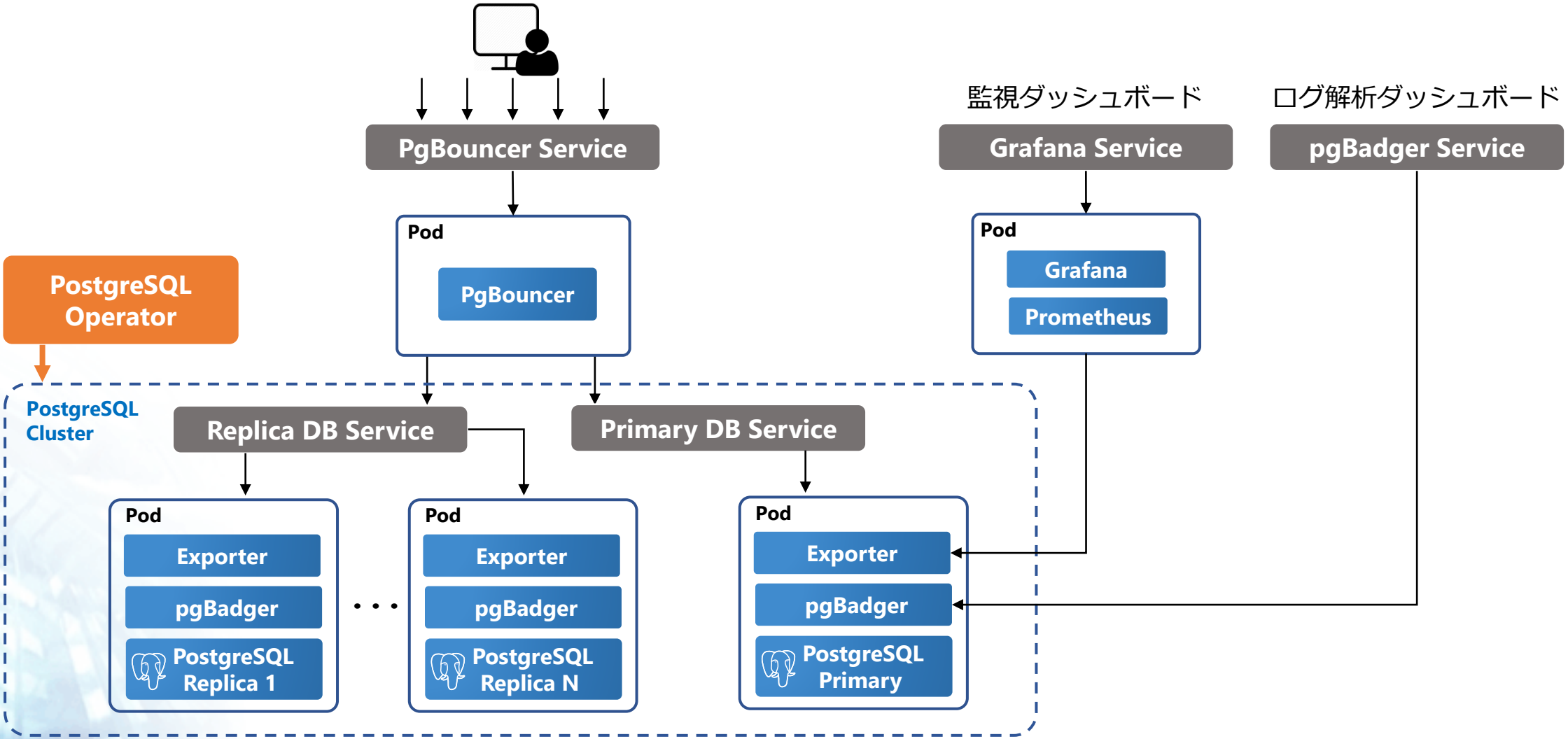
① コンテナの Docker イメージのアップデート
② Replica が新バージョンの Primary に昇格
③ 元 Primary Pod が再作成され、Replica として復帰



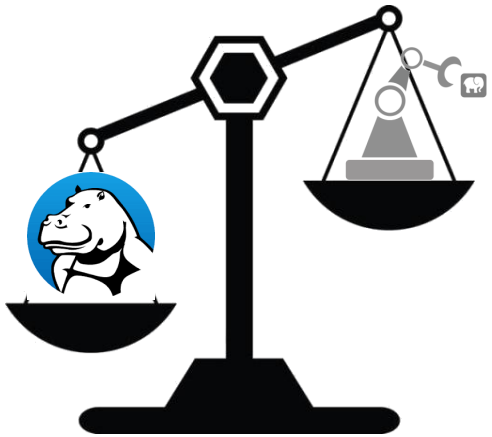
その他の機能

	Zalando	Crunchy
コネクションプーリング	○	○
監視	×	○
PostgreSQL のログ解析	×	○
ボリュームサイズ拡張	○	△ ※既存のクラスタのボリュームサイズを拡張できず、クラスタを複製してサイズ変更が必要

活用例：便利な機能の活用



Zalando と Crunchy のどちらを選択するか?

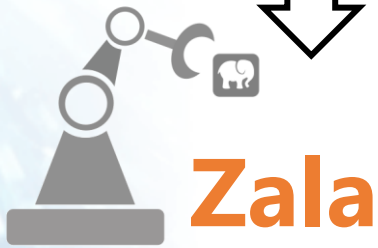


Zalando と Crunchy のどちらを選択するか?

基本的な機能は同じ

Zalando 社で本番環境での導入実績あり

- Kubernetes の操作に慣れている人向け
- 設定追加・変更の内容を履歴として残したい
- GUI (Webインタフェース) でクラスタを管理したい
- バージョンアップ時のダウンタイムは極力減らしたい



Zalando

PostgreSQL の専門家が開発している

- 専用コマンドで簡単に操作できるのがよい
- 多くの便利な機能を標準で使いたい
  - pgAdmin
  - ログ解析
  - 監視

Crunchy

参考情報

- 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。