



**EDB**

Postgres® for the AI Generation

# **EDB Postgres Distributed for Kubernetes**

リリース *1.0.1*

**EnterpriseDB Corporation.**

2024 年 10 月 16 日



# 目次

|              |                                                                       |           |
|--------------|-----------------------------------------------------------------------|-----------|
| <b>第 1 章</b> | <b>Prerequisite and Release Note</b>                                  | <b>3</b>  |
| 1.1          | Introduction . . . . .                                                | 3         |
| 1.1.1        | 始める前に . . . . .                                                       | 3         |
| 1.1.2        | サポートされている Kubernetes ディストリビューション . . . . .                            | 3         |
| 1.1.3        | 要件 . . . . .                                                          | 3         |
| 1.1.4        | API リファレンス . . . . .                                                  | 4         |
| 1.1.5        | 商標 . . . . .                                                          | 4         |
| 1.2          | EDB Postgres for Kubernetes Release notes . . . . .                   | 4         |
| 1.2.1        | EDB Postgres Distributed for Kubernetes 1.0.1 release notes . . . . . | 4         |
| 1.2.2        | EDB Postgres Distributed for Kubernetes 1.0.0 release notes . . . . . | 5         |
| <b>第 2 章</b> | <b>Getting Started</b>                                                | <b>7</b>  |
| 2.1          | Before you start . . . . .                                            | 7         |
| 2.1.1        | Kubernetes の用語 . . . . .                                              | 7         |
| 2.1.2        | PGD 用語 . . . . .                                                      | 8         |
| 2.1.3        | クラウド用語 . . . . .                                                      | 9         |
| 2.1.4        | 次に何をするか . . . . .                                                     | 9         |
| 2.2          | Use cases . . . . .                                                   | 9         |
| 2.2.1        | ケース 1Kubernetes 内のアプリケーション . . . . .                                  | 9         |
| 2.2.2        | ケース 2Kubernetes 外部のアプリケーション . . . . .                                 | 10        |
| 2.3          | Architecture . . . . .                                                | 10        |
| 2.3.1        | EDB Postgres 分散との関係 . . . . .                                         | 11        |
| 2.3.2        | Kubernetes の EDB Postgres との関係 . . . . .                              | 11        |
| 2.3.3        | Kubernetes 用に分散された EDB Postgres を使用した PGD の管理 . . . . .               | 12        |
| 2.3.4        | Kubernetes での PGD の展開 . . . . .                                       | 13        |
| 2.4          | Installation . . . . .                                                | 15        |
| 2.4.1        | 前提条件 . . . . .                                                        | 15        |
| 2.4.2        | リポジトリを追加する . . . . .                                                  | 15        |
| 2.4.3        | 環境変数を設定する . . . . .                                                   | 16        |
| 2.4.4        | イメージを展開する . . . . .                                                   | 16        |
| 2.4.5        | 証明書発行者を作成する . . . . .                                                 | 16        |
| 2.4.6        | Red Hat OpenShift . . . . .                                           | 17        |
| 2.5          | Quick start . . . . .                                                 | 17        |
| 2.5.1        | パート 1-ローカル Kubernetes プレイグラウンドをセットアップする . . . . .                     | 17        |
| 2.5.2        | パート 2-Kubernetes 用に分散された EDB Postgres のインストール . . . . .               | 18        |
| 2.5.3        | パート 3-PGD クラスターを展開する . . . . .                                        | 18        |
| <b>第 3 章</b> | <b>Using</b>                                                          | <b>23</b> |
| 3.1          | Managing EDB Postgres Distributed (PGD) databases . . . . .           | 23        |
| 3.1.1        | PGD CLI . . . . .                                                     | 23        |

|        |                                      |    |
|--------|--------------------------------------|----|
| 3.1.2  | データベースへのアクセス                         | 24 |
| 3.2    | Backup on object stores              | 26 |
| 3.2.1  | 共通オブジェクトストア                          | 26 |
| 3.2.2  | WAL アーカイブ                            | 26 |
| 3.2.3  | スケジュールされたバックアップ                      | 27 |
| 3.2.4  | リテンション ポリシー                          | 29 |
| 3.2.5  | 圧縮アルゴリズム                             | 30 |
| 3.2.6  | バックアップオブジェクトのタグ付け                    | 30 |
| 3.2.7  | PGD ノードのオンデマンドバックアップ                 | 30 |
| 3.3    | Recovery                             | 30 |
| 3.3.1  | 前提条件                                 | 31 |
| 3.3.2  | オブジェクトストアからのリカバリー                    | 31 |
| 3.3.3  | オブジェクトストアからの PITR                    | 32 |
| 3.3.4  | backupID を指定したオブジェクトストアからのリカバリー      | 33 |
| 3.3.5  | volumeSnapshot からのリカバリー              | 34 |
| 3.3.6  | volumeSnapshot からの PITR              | 35 |
| 3.3.7  | リカバリーターゲット                           | 36 |
| 3.4    | Security                             | 36 |
| 3.4.1  | コード                                  | 37 |
| 3.4.2  | コンテナ                                 | 37 |
| 3.4.3  | クラスター                                | 38 |
| 3.5    | Connectivity                         | 41 |
| 3.5.1  | サービス                                 | 41 |
| 3.5.2  | ドメイン名解決                              | 43 |
| 3.5.3  | TLS 構成                               | 43 |
| 3.5.4  | アプリケーションから PGD クラスターに接続する            | 45 |
| 3.6    | Certificates                         | 46 |
| 3.7    | Client TLS/SSL connections           | 47 |
| 3.8    | Declarative pausing and resuming     | 47 |
| 3.9    | Installation images and repositories | 48 |
| 3.9.1  | EDB' s private image registries      | 48 |
| 3.9.2  | Identifying image names              | 51 |
| 3.9.3  | Assembling a deployment command      | 52 |
| 3.10   | Predefined labels                    | 55 |
| 3.10.1 | 事前定義されたアノテーション                       | 55 |
| 3.11   | Managed Configuration                | 56 |
| 3.12   | PGDGroup parting                     | 56 |
| 3.12.1 | 削除とファイナライザー                          | 56 |
| 3.12.2 | 制限時間                                 | 57 |
| 3.12.3 | ファイナライザーをスキップ                        | 57 |
| 3.12.4 | PGD グループのクリーンアップ                     | 57 |
| 3.13   | SQLMutations                         | 58 |
| 3.13.1 | SQL ミューテーションタイプ                      | 59 |
| 3.14   | Red Hat OpenShift                    | 59 |
| 3.14.1 | EDB プライベートレジストリへのアクセスの保証             | 60 |
| 3.14.2 | オペレーターのインストール                        | 60 |
| 3.14.3 | PGD クラスターの作成                         | 62 |

|         |                                   |    |
|---------|-----------------------------------|----|
| 3.15    | Transparent Data Encryption (TDE) | 71 |
| 3.16    | LDAP Authentication               | 73 |
| 3.17    | Examples of configuration         | 74 |
| 3.18    | API Reference                     | 77 |
| 3.18.1  | リソースタイプ                           | 77 |
| 3.18.2  | 証明書キーストア                          | 78 |
| 3.18.3  | CertificatePrivateKey             | 78 |
| 3.18.4  | 証明書仕様                             | 79 |
| 3.18.5  | コンディションステータス                      | 81 |
| 3.18.6  | JKSKeystore                       | 81 |
| 3.18.7  | KeyUsage                          | 82 |
| 3.18.8  | LocalObjectReference              | 82 |
| 3.18.9  | ObjectReference                   | 82 |
| 3.18.10 | PKCS12Keystore                    | 83 |
| 3.18.11 | PrivateKeyAlgorithm               | 83 |
| 3.18.12 | PrivateKeyEncoding                | 83 |
| 3.18.13 | PrivateKeyRotationPolicy          | 84 |
| 3.18.14 | SecretKeySelector                 | 84 |
| 3.18.15 | X509 件名                           | 84 |
| 3.18.16 | PGDGroup                          | 85 |
| 3.18.17 | PGDGroupCleanup                   | 85 |
| 3.18.18 | バックアップ                            | 85 |
| 3.18.19 | BackupStatus                      | 86 |
| 3.18.20 | CNP ステータス                         | 86 |
| 3.18.21 | CertManagerTemplate               | 87 |
| 3.18.22 | ClientCertConfiguration           | 87 |
| 3.18.23 | Client 事前プロビジョニングされた証明書           | 88 |
| 3.18.24 | CnpBaseConfiguration              | 88 |
| 3.18.25 | CnpConfiguration                  | 90 |
| 3.18.26 | ConnectionString                  | 90 |
| 3.18.27 | ConnectivityConfiguration         | 91 |
| 3.18.28 | 接続ステータス                           | 91 |
| 3.18.29 | DNS 構成                            | 92 |
| 3.18.30 | DiscoveryJobConfig                | 92 |
| 3.18.31 | 継承されたメタデータ                        | 92 |
| 3.18.32 | メタデータ                             | 93 |
| 3.18.33 | NodeCertificateStatus             | 93 |
| 3.18.34 | NodeKindName                      | 94 |
| 3.18.35 | ノードの概要                            | 94 |
| 3.18.36 | NodesExtensions ステータス             | 94 |
| 3.18.37 | OTELConfiguration                 | 94 |
| 3.18.38 | OTELTLS 構成                        | 95 |
| 3.18.39 | PGDGroupCleanupSpec               | 95 |
| 3.18.40 | PGDGroupCleanupStatus             | 96 |
| 3.18.41 | PGDGroupSpec                      | 96 |
| 3.18.42 | PGDGroupStatus                    | 97 |
| 3.18.43 | PGDNodeGroupEntry                 | 98 |

|         |                                                         |     |
|---------|---------------------------------------------------------|-----|
| 3.18.44 | PGDNodeGroupSettings                                    | 98  |
| 3.18.45 | PGDProxyConfiguration                                   | 99  |
| 3.18.46 | PGDProxyEntry                                           | 99  |
| 3.18.47 | PGDP プロキシ設定                                             | 100 |
| 3.18.48 | PGDP プロキシステータス                                          | 101 |
| 3.18.49 | PGD ステータス                                               | 101 |
| 3.18.50 | ParentGroupConfiguration                                | 102 |
| 3.18.51 | PauseStatus                                             | 102 |
| 3.18.52 | PgdConfiguration                                        | 103 |
| 3.18.53 | 事前プロビジョニングされた証明書                                        | 103 |
| 3.18.54 | メソッド別リカバリーポイント                                          | 104 |
| 3.18.55 | ReplicationCertificateStatus                            | 104 |
| 3.18.56 | リストア                                                    | 104 |
| 3.18.57 | RestoreStatus                                           | 105 |
| 3.18.58 | RootDNSConfiguration                                    | 105 |
| 3.18.59 | SQL ミューテーション                                            | 106 |
| 3.18.60 | SQLMutationType                                         | 106 |
| 3.18.61 | SQL ミューテーション                                            | 106 |
| 3.18.62 | ScheduledBackupSpec                                     | 107 |
| 3.18.63 | ScheduledBackupStatus                                   | 108 |
| 3.18.64 | ServerCertConfiguration                                 | 108 |
| 3.18.65 | ServiceTemplate                                         | 108 |
| 3.18.66 | ServiceUpdate 戦略                                        | 109 |
| 3.18.67 | TLS 構成                                                  | 109 |
| 3.18.68 | TLS モード                                                 | 109 |
| 3.18.69 | VolumeSnapshotRestoreStatus                             | 110 |
| 3.18.70 | VolumeSnapshots 構成                                      | 110 |
| 3.19    | Supported versions                                      | 110 |
| 3.19.1  | Kubernetes リリースの EDB Postgres 分散のサポートステータス              | 110 |
| 3.19.2  | EDB Postgres Distributed for Kubernetes リリースバージョンフォーマット | 111 |
| 3.20    | Known issues and limitations                            | 111 |
| 3.20.1  | Postgres メジャーバージョンのアップグレード                              | 111 |
| 3.20.2  | データの移行                                                  | 111 |
| 3.20.3  | PgBouncer との接続                                          | 111 |
| 3.20.4  | バックアップ操作                                                | 112 |
| 3.20.5  | EDB Postgres 分散の既知の問題と制限                                | 112 |

Kubernetes の EDB Postgres 分散 PGD4K は、PGD プロキシによってルーティングされるトラフィックを使用して、Kubernetes で EDB Postgres 分散 PGD ワークロードを管理するように設計されたオペレーターです。

オペレーターが提供するメインのカスタムリソースは、PGDGroup と呼ばれます。

アーキテクチャは、異なる Kubernetes クラスターにまたがって展開することもできます。



# 第1章 Prerequisite and Release Note

## 1.1 Introduction

### 1.1.1 始める前に

EDB Postgres Distributed for Kubernetes は、Kubernetes 環境に EDB Postgres Distributed を展開する方法を提供します。したがって、[EDB Postgres Distributed documentation](#) を読むことをお勧めします。

Kubernetes 用 EDB Postgres Distributed での作業を開始するには、PGD ドキュメントの以下を読んでください。

- [Terminology](#)
- [PGD overview](#)
- [Choosing your architecture](#)
- [Choosing a Postgres distribution](#)

高度な使用法と最大限のカスタマイズのためには、[Architecture](#) で説明しているように、

[EDB Postgres for Kubernetes documentation](#) に慣れることも重要です。

### 1.1.2 サポートされている Kubernetes ディストリビューション

Kubernetes の EDB Postgres Distributed は、次の場合に利用できます。

- Helm チャートを介した Kubernetes バージョン 1.23 以降
- Red Hat OpenShift 認定オペレーターを介した Red Hat OpenShift バージョン 4.10 以降のみ

### 1.1.3 要件

Kubernetes 用 EDB Postgres 分散では、分散 PGD クラスターをホスティングする Kubernetes / OpenShift クラスターが、以下に対応するようにお客様が準備したものである必要があります。

- PGD グローバルグループが分散されているすべての Kubernetes クラスターを含む公開キーインフラストラクチャ PKI。mTLS は、メッシュトポロジ内のすべてのノードを認証および承認し、暗号化された通信を保証するために必要です。
- PGD グローバルグループに関係するすべての Kubernetes クラスターのネットワーキングインフラストラクチャで、各ノードが相互に通信できるようにする

EDB Postgres Distributed for Kubernetes には、Cert Manager 1.10 以降も必要です。

参考:

詳細については、'[Connectivity](#)' を参照してください。

1.1.4 API リファレンス

EDB Postgres Distributed for Kubernetes が提供するリソースのリストについては、[API リファレンス](#) を参照してください。

1.1.5 商標

Postgres, PostgreSQL, and the Slonik logo

は、カナダ PostgreSQL コミュニティ協会の商標または登録商標であり、その許可を得て使用されています。

1.2 EDB Postgres for Kubernetes Release notes

EDB Postgres Distributed for Kubernetes ドキュメントでは、マイナーリリースとパッチを含む EDB Postgres Distributed for Kubernetes のメジャーバージョンについて説明しています。リリースノートは、各リリースの新機能に関する情報を提供します。マイナーまたはパッチリリースで導入された新機能の場合、コンテンツには、機能を導入したリリースも示されます。

| Version          | Release date |
|------------------|--------------|
| <i>Bootstrap</i> | 14 Oct 2024  |
| <i>Bootstrap</i> | 24 Apr 2024  |

1.2.1 EDB Postgres Distributed for Kubernetes 1.0.1 release notes

リリース日 2024 年 10 月 14 日

機能

| Component | Description                     |
|-----------|---------------------------------|
| PGD4K     | オペレーターベースイメージが UBI9 にアップグレードされた |
| PGD4K     | PGDGroup での LDAP 認証構成のサポート      |
| PGD4K     | PGDGroup でのテーブルスペース構成のサポート      |

[次のページに続く](#)

表 2 – 前のページからの続き

| Component | Description                                   |
|-----------|-----------------------------------------------|
| PGD4K     | PGDGroup での予測ボリュームテンプレート構成のサポート               |
| PGD4K     | プロキシサービスの作成による pgd の読み取りノードルーティングのサポート        |
| PGD4K     | VolumeSnapshot を使用した PGDGroup のバックアップと復元をサポート |
| PGD4K     | さまざまなバックアップ方法に対応する PGDGroup の複数のスケジューラーをサポート  |
| PGD4K     | 書き込みリードノードに対してのみ実行される新しいミューテーションタイプをサポートします。  |
| PGD4K     | PGDGroup 構成が変更されるときに不必要な書き込みリーダー転送を回避する       |
| PGD4K     | 一時停止された PGDGroup のノードセクターと許容変更の処理             |
| PGD4K     | 検出ジョブでの nodeSelector と tolerance を再利用し        |
| PGD4K     | 参加する前に適用されるサポートアプリケーションデータベース構成               |
| PGD4K     | トポロジ SpreadConstraints 構成のサポート                |

## 1.2.2 EDB Postgres Distributed for Kubernetes 1.0.0 release notes

リリース日 2024 年 4 月 24 日

これは、EDB Postgres 分散クラスターを展開および管理する Kubernetes オペレーターである Kubernetes の EDB Postgres Distributed の最初のメジャー安定リリースです。

### Kubernetes 1.0.0 用に分散された EDB Postgres のハイライト

オペレーターは、API グループ `pgd.k8s.enterprisedb.io` に PGDGroup カスタムリソースを実装します。このリソースを使用して、Kubernetes 内に EDB Postgres 分散クラスターを作成および管理できます。次の機能を使用します。

- バージョン 5 以降での EDB Postgres 分散クラスターの展開。
- 障害が発生した PGD ノードの復旧や再起動など、Postgres Distributed の自己修復機能に加えて、追加の自己修復機能。

\*アプリケーションを各 PGD グループの書き込みリーダーに接続できるサービス定義。

**注釈:** EDB Postgres Distributed for Kubernetes オペレーターは、[EDB Postgres for Kubernetes PG4K](#) を活用し、そのプロジェクトの機能の多くを継承します。Kubernetes バージョン 1.0.0 の EDB Postgres Distributed は、特に PG4K のリリース 1.22 に基づいています。詳細については、[PG4K release notes](#) を参照してください。## 機能

| Component | Description                                          |
|-----------|------------------------------------------------------|
| PGD4K     | Kubernetes 内のバージョン 5 以降を使用した EDB Postgres 分散クラスターの展開 |
| PGD4K     | 障害が発生した PGD ノードのリカバリや再起動などの自己修復機能                    |
| PGD4K     | アプリケーションを各 PGD グループの書き込みリーダーに接続できる定義されたサービス          |
| PGD4K     | Raft サブグループの実装                                       |
| PGD4K     | TLS 接続とクライアント証明書認証                                   |
| PGD4K     | S3 互換オブジェクトストアへの継続的なバックアップ                           |

## 第2章 Getting Started

### 2.1 Before you start

開始する前に、Kubernetes と PGD に固有のいくつかの用語に精通することが重要です。

#### 2.1.1 Kubernetes の用語

##### Node

ノードは、Kubernetes の仮想または物理的なワーカーマシンであり、ポッドを実行するために必要なすべてのサービスがコントロールプレーンノードによって管理されます。

##### Pod

ポッドは、Kubernetes クラスターに展開できる最小のコンピューティングユニットであり、ネットワークとストレージを共有する 1 つ以上のコンテナで構成されます。

##### Service

*service* は、ポッドのグループで実行され、アプリケーション全体のサービス検出、ロードバランシング、フェイルオーバーなどの重要な機能を標準化するアプリケーションをネットワークサービスとして公開する抽象化です。

##### Secret

シークレットは、ポッド内で使用するパスワード、アクセスキー、またはトークンなどの少量の機密データを保存するように設計されたオブジェクトです。

##### Storage class

**ストレージクラス** を使用すると、管理者は、プロビジョナー AWS EBS など、再利用ポリシー、マウントオプション、ボリューム拡張などを含むクラスター内のストレージのクラスを定義できます。

##### Persistent volume

永続ボリューム PV は、管理者によって手動でプロビジョニングされた、または**ストレージクラス**コントローラによって動的にプロビジョニングされたストレージを表す Kubernetes クラスター内のリソースです。PV は、**永続的なボリューム要求**を使用してポッドに関連付けられ、そのライフサイクルはそれを使用するポッドから独立しています。通常、PV はネットワークボリューム、特にパブリッククラウドでは、**\*local persistent volume\* (LPV)** は、それを使用するポッドが実行されている特定のノードにのみ存在する永続ボリュームです。

##### Persistent volume claim

**永続的なボリューム要求** PVC は、ストレージの要求を表します。これには、サイズ、アクセスモード、または特定のストレージクラスが含まれる場合があります。ポッドがノードリソースを消費する方法と同様に、PVC は PV のリソースを消費します。

### Namespace

**名前空間**は、Kubernetes クラスターの論理的で分離されたサブセットであり、より広範な物理クラスター内の**仮想クラスター**として見ることができます。名前空間を使用すると、管理者はプロジェクト、部門、チームなどに基づいて分離された環境を作成できます。

### RBAC

**ロールベースセキュリティ** としても知られる**ロールベースのアクセス制御** RBAC は、コンピューターシステムのセキュリティで使用され、システムのネットワークとリソースへのアクセスを許可されたユーザーのみに制限する方法です。Kubernetes には、名前空間とクラスターレベルでロールを制御し、特定のリソースと個人に関連付けるネイティブ API があります。

### CRD

**カスタムリソース定義** CRD は、Kubernetes API の拡張機能であり、開発者は**カスタムリソース**と呼ばれる新しいデータ型とオブジェクトを作成できます。

### Operator

**オペレーター**は、1 つ以上のアプリケーションまたは特定のサービスを管理するときに通常人間のオペレーターが実行する手順を自動化する Kubernetes ソフトウェア拡張機能です。オペレーターは、Kubernetes を支援して、リソースの定義された状態が観察された状態と常に一致するようにします。

### kubectl

kubectl は、Kubernetes クラスターの管理に使用されるコマンドラインツールです。

Kubernetes の EDB Postgres Distributed には、コミュニティでサポートされている Kubernetes バージョンが必要です。詳細については、[Supported releases](#) を参照してください。

## 2.1.2 PGD 用語

詳細については、PGD ドキュメントの [Kubernetes の用語](#) を参照してください。

### Data node

PGD データベースインスタンス。

### Failover

高可用性データベースクラスターの障害を認識し、アプリケーションを別のアクティブなデータベースに接続するアクションを実行する自動プロセス。

### Switchover

アプリケーションとクラスター内のアクティブなデータベースノード間の接続の計画的な変更。通常はメンテナンスのために行われます。

### Write leader

常時オンアーキテクチャでは、ノードはアプリケーションの正しい接続エンドポイントとして選択されます。このノードは書き込みリーダーと呼ばれます。書き込みリーダーは、データノードのクォーラムのコンセンサスによって選択されます。

### 2.1.3 クラウド用語

リージョンクラウド内のリージョンは、アベイラビリティゾーンで編成された隔離された独立した地理的エリアです。リージョン内のゾーンの往復ネットワーク遅延はほとんどありません。

ゾーンクラウドの\*アベイラビリティゾーン\*\*ゾーン\*としても知られるは、リソースを展開できるリージョンのエリアです。通常、アベイラビリティゾーンは、データセンターまたは同じデータセンターの分離された建物に対応します。

### 2.1.4 次に何をするか

用語に慣れたので、選択したクラウド環境にオペレーターを展開する前に *'Quick start'* できます。

## 2.2 Use cases

Kubernetes の EDB Postgres Distributed は、完全なクラウドネイティブエクスペリエンスを実現するために、同じ Kubernetes クラスターにあるアプリケーションと連携するように設計されました。

ただし、データベースは Kubernetes クラスター内でホストできますが、アプリケーションを同時にコンテナ化できず、VM などの従来の環境で実行する必要がある場合があります。

以下は、基本的な考慮事項の概要です。 [EDB Postgres for Kubernetes documentation](#) を参照してください  
詳細については。

### 2.2.1 ケース 1Kubernetes 内のアプリケーション

通常、アプリケーションとデータベースは、Kubernetes クラスター内の同じ名前空間で実行されます。

アプリケーションは通常ステートレスであり、標準の展開として管理され、複数のレプリカがさまざまな Kubernetes ノードに分散され、ClusterIP サービスを介して内部公開されます。

サービスは、Ingress および HTTPS を介してプロバイダーのロードバランサー機能を介してエンドユーザーに外部公開されます。

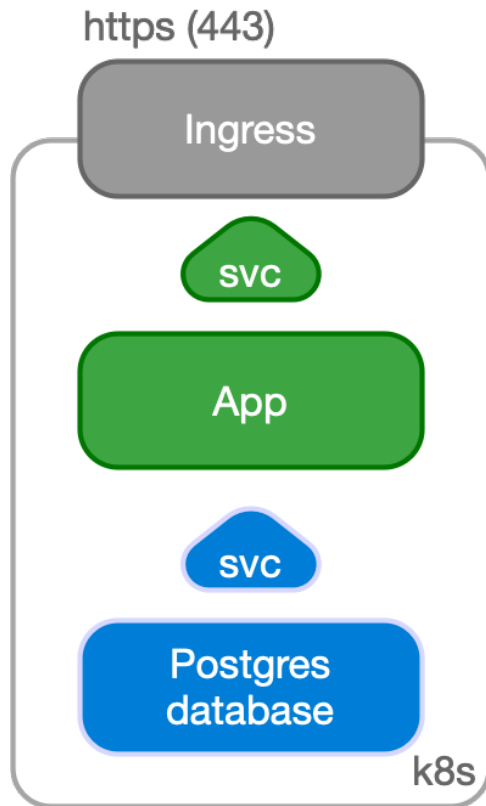


図 1: Application and Database inside Kubernetes

### 2.2.2 ケース 2Kubernetes 外部のアプリケーション

もう 1 つの考えられるユースケースは、Kubernetes 内で PGD データベースを管理し、アプリケーションをその外部、たとえば仮想化環境に持たせることです。この場合、PGD は、Kubernetes で定義された Ingress リソースに対応する IP アドレスまたはホスト名と TCP ポートで表されます。

アプリケーションは、PGD への TLS 接続のメリットを得ることができます。

## 2.3 Architecture

Kubernetes で EDB Postgres 分散を展開するときは、これらの主なアーキテクチャの側面を考慮してください。

Kubernetes 用に分散された EDB Postgres は [Kubernetes operator](#)

プライベート、パブリック、ハイブリッド、またはマルチクラウド環境で実行される EDB Postgres 分散クラスターを展開および管理するように設計されています。

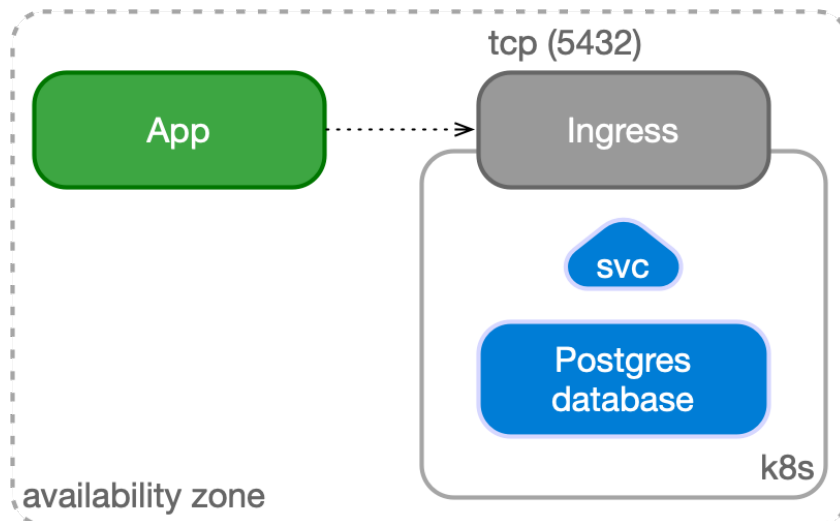


図 2: Application outside Kubernetes

### 2.3.1 EDB Postgres 分散との関係

は、高いパフォーマンスと可用性を実現するように設計された Postgres のマルチマスター実装です。PGD は通常、[Trusted Postgres Architect \(TPA\)](#) を使用した展開が必要です。このツールは、[Ansible](#) を使用して PGD クラスターをプロビジョニングおよび展開するツールです。

EDB Postgres Distributed for Kubernetes は、コンテナと Kubernetes を活用して、PGD クラスターを展開する別の方法を提供します。利点は、結果のアーキテクチャが次のことです。

- 自己修復性と堅牢性があります。
- 宣言的構成を介して管理されます。
- 広大で成長を続ける Kubernetes エコシステムを利用します。

### 2.3.2 Kubernetes の EDB Postgres との関係

PGD クラスターは、1つ以上の *PGD グループ* で構成され、それぞれが1つ以上の *PGD ノード* があります。PGD ノードは Postgres データベースです。EDB Postgres Distributed for Kubernetes は、EDB Postgres for Kubernetes で定義された *Cluster* リソース、特に単一のインスタンスつまりレプリカなしのクラスターを使用して各 PGD ノードを内部管理します。

[.spec.cnp](#) の各 *Cluster* によって作成された単一の PostgreSQL インスタンスを構成できます。

PGD グループ仕様の

EDB Postgres Distributed for Kubernetes では、EDB Postgres for Kubernetes と同様に、基になるデータベース実装がデータレプリケーションを担当します。ただし、フェイルオーバーとスイッチオーバーの動作が異なり、Raft 選挙と新しい書き込みリーダーの指名が必要になることに注意することが重要です。Kubernetes の EDB Postgres は、データノードの展開と修復のみを処理します。

### 2.3.3 Kubernetes 用に分散された EDB Postgres を使用した PGD の管理

EDB Postgres Distributed for Kubernetes オペレーターは、PGD クラスターの完全なライフサイクルを管理できます。そのため、PGD ノード単一インスタンス **Clusters** として表されるに加えて、PGD に関連付けられている他のオブジェクトを管理する必要があります。

PGD は、分散コンセンサスのための Raft アルゴリズムに依存して、ノードのメタデータ、特に**書き込みリーダー**に関する契約を管理します。データノード間のコンセンサスは、新しいグローバルシーケンスの生成や分散 DDL の実行などの操作にも必要です。

これらの考慮事項により、データベースノードの上の PGD に追加のアクターが強制されます。

EDB Postgres Distributed for Kubernetes は、以下を管理します。

- データノード。ノードはデータベースであり、EDB Postgres for Kubernetes によって管理され、単一のインスタンスで **Cluster** を作成します。
- **Witness nodes**

は、データレプリケーションに参加しない基本的なデータベースインスタンスです。彼らの機能は、偶数のデータノードを持つグループ、またはネットワーク分割後のコンセンサスが可能であることを保証することです。Witness ノードは、単一インスタンスの **Cluster** リソースを使用して管理されます。

- **PGD proxies**

書き込みリーダーの情報を使用して Postgres プロキシとして機能します。PGD プロキシは、書き込みを現在の書き込みリーダーにルーティングするために Raft からの情報を必要とします。

#### プロキシとルーティング

PGD グループは、PGD ノードのフルメッシュ接続を想定しています。各ノードは、適切な接続文字列 `libpq` スタイルの DSN を使用して他のすべてのノードに接続する必要があります。書き込み操作はすべてのノードに送信される必要はありません。PGD は、1 つのノードにコミットされた後、データの複製を担当します。

パフォーマンスを向上させるために、書き込み操作のほとんどを単一のノード**書き込みリーダー**に送信することをお勧めします。Raft は、どのノードが書き込みリーダーであるかを識別し、PGD ノードに関するメタデータを保持するために使用されます。PGD プロキシは、書き込みを書き込みリーダーに透過的にルーティングし、スイッチオーバーまたはフェイルオーバーが発生した場合に新しい書き込みリーダーにすばやくピボットするために使用されます。

それぞれが別個の書き込みリーダーを維持できる **Raft サブグループ**を構成することが可能です。EDB Postgres Distributed for Kubernetes では、PGD プロキシを含む PGD グループは Raft サブグループを構成します。

PGD プロキシでは 2 種類のルーティングが使用できます。

- グローバルルーティングは、トップレベルの Raft グループを使用し、1 つのグローバル書き込みリーダーを維持します。
- ローカルルーティングは、サブグループを使用して個別の書き込みリーダーを維持します。ローカルルーティングは、書き込みの地理的な分離を実現するために使用されることがよくあります。

EDB Postgres Distributed for Kubernetes では、デフォルトでローカルルーティングが使用され、グローバルルーティングを選択する構成オプションを使用できます。

詳細については、[PGD documentation of routing with Raft](#) を参照してください。

## PGD アーキテクチャと高可用性

EDB は、PGD の分散マルチマスター機能を有効に活用し、高可用性を提供するために、いくつかの推奨アーキテクチャを提案しています。

Always On アーキテクチャは、1 つの場所にある 1 つのグループ、または 2 つの別の場所にある 2 つのグループから構築されます。詳細については、PGD ドキュメントの [Choosing your architecture](#) を参照してください。

### 2.3.4 Kubernetes での PGD の展開

EDB Postgres Distributed for Kubernetes は、Kubernetes を活用して PGD クラスターを展開および管理します。そのため、PGD を Kubernetes エコシステムに変換するには、いくつかの適応が必要です。

#### イメージとオペランド

3 つの Postgres ディストリビューションのいずれかを実行するように PGD を構成できます。

[PGD documentation](#) を参照してください。

各ディストリビューションの特徴を理解するため。

Kubernetes で機能するには、Postgres ディストリビューションごとにコンテナが提供されます。これらはオペランドです。さらに、オペレーターイメージはこれらと同じリポジトリに保持されます。

EDB's private image registries を参照

画像へのアクセスの詳細については、

#### Kubernetes アーキテクチャ

[PG4K document on Kubernetes architecture](#) のポイントの一部

はここに再現されています。詳細については、PG4K ドキュメントを参照してください。

Kubernetes は、別の物理的な場所にまたがる可能性をネイティブに提供します。これらの物理的な場所は、データセンター、障害ゾーン、またはより多くの場合アベイラビリティゾーンとしても知られます。これらは、冗長、低遅延、プライベートネットワーク接続を介して相互に接続されます。

分散システムであるため、*Kubernetes クラスター* の推奨最小数は 3 です。この最小値により、コントロールプレーンは単一ゾーンの障害に対する回復力があります。これは、各データセンターがいつでもアクティブで、ワークロードを同時に実行できることを意味します。

Kubernetes 用 EDB Postgres 分散を **単一の Kubernetes クラスター** にインストールできます

または **複数の Kubernetes クラスター** 全体にわたって。

### 単一の Kubernetes クラスター

マルチアベイラビリティゾーンの Kubernetes アーキテクチャは、クラウドプロバイダーが管理する Kubernetes サービスの一般的なものです。このようなアーキテクチャにより、EDB Postgres Distributed for Kubernetes と EDB Postgres for Kubernetes オペレーターは、すべてのゾーンがアクティブであることを考慮して、アベイラビリティゾーンにわたってワークロードとノードをスケジュールできます。

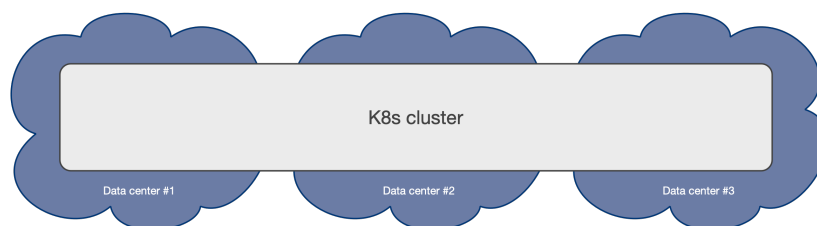


図 3: Kubernetes cluster spanning over 3 independent data centers

PGD クラスターは、単一の Kubernetes クラスターに展開でき、Kubernetes 可用性ゾーンを利用して、Always On 推奨アーキテクチャを含む高可用性アーキテクチャを有効にすることができます。

PGD ドキュメントの [Choosing your architecture](#) に示されている *Always On Single Location* アーキテクチャは、3つのアベイラビリティゾーンを持つ単一の Kubernetes クラスターで実現できます。



図 4: Always On Single Region

EDB Postgres Distributed for Kubernetes オペレーターは、EDB Postgres for Kubernetes の場合と同様に、アフィニティ、トレランス、ノードセクターを使用して、ポッドのスケジューリングつまりどのポッドがどのデータセンターに行くかを制御できます。個々のスケジューリング制御は、ノードだけでなくプロキシでも利用できます。

詳細については、EDB Postgres for Kubernetes ドキュメントの [Kubernetes documentation on scheduling](#) および [Scheduling](#) を参照してください。

## 複数の Kubernetes クラスター

PGD クラスターは、相互に確実に通信できる複数の Kubernetes クラスターに展開することもできます。

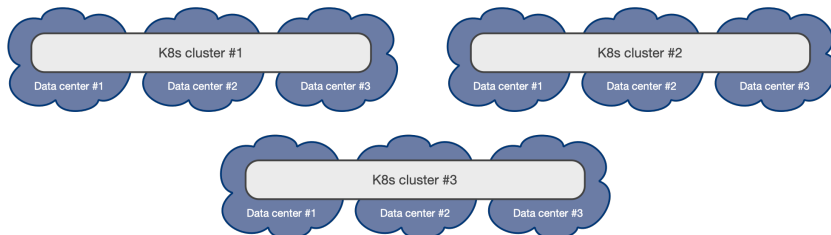


図 5: Multiple Kubernetes clusters

### *Always On multi-location PGD architectures*

接続要件を満たす複数の Kubernetes クラスターで実現できます。

詳細については、'[Connectivity](#)' を参照してください。

## 2.4 Installation

提供されている [Helm chart](#) を使用して、Kubernetes 用 EDB Postgres 分散を展開できます。

---

**ヒント:** Helm チャートの詳細については、[Helm chart repo documentation](#) を参照してください。

---

これらのインストール手順では、`helm` を使用して、使用可能な最新バージョンでデフォルトイメージのセットを展開する方法をカバーしています。別のオペランドまたはプロキシイメージを指定する場合は、インストールを続行する前に [Identify your installation images and repositories](#) を参照してください。

### 2.4.1 前提条件

- [Helm](#) をインストールします。これらの [instructions](#) に従って、システムにインストールします。
- `kubectl` をインストールします。手順については、[Install Tools](#) を参照してください。
- 少なくとも 1 つの Kubernetes クラスターを作成します。

### 2.4.2 リポジトリを追加する

`helm` を使用して、すべてのイメージを含むリポジトリを追加します。

```
helm repo add edb \  
https://enterprisedb.github.io/edb-postgres-for-kubernetes-charts/
```

### 2.4.3 環境変数を設定する

REPOSITORY\_NAME および EDB\_SUBSCRIPTION\_TOKEN の環境変数を設定します。

1. REPOSITORY\_NAME をリポジトリの名前に設定します。この例では、画像は `k8s_enterprise_pgd` リポジトリから取得しています。

```
export REPOSITORY_NAME=k8s_enterprise_pgd
```

1. 個人トークンを使用するように EDB\_SUBSCRIPTION\_TOKEN を設定します。

```
export EDB_SUBSCRIPTION_TOKEN=<my_registry_token>
```

### 2.4.4 イメージを展開する

**重要:** オペレーターイメージとオペランドイメージの両方が保存されているプライベート EDB リポジトリへのアクセスが必要です。アクセスには、有効な [EDB subscription plan](#) が必要です。アクセス資格情報を特定するには、EDB's private image registries を参照してください。

演算子と選択したオペランドの両方のコンテナイメージが EDB のプライベートレジストリにあることを考えると、`helm` がそれらを取得できるようにするには資格情報が必要です。

次のコマンドのリポジトリとトークンを必ず置き換えます。

```
helm upgrade --dependency-update \
  --install edb-pg4k-pgd \
  --wait \
  --namespace pgd-operator-system \
  --create-namespace \
  edb/edb-postgres-distributed-for-kubernetes \
  --set image.imageCredentials.username=${REPOSITORY_NAME} \
  --set image.imageCredentials.password=${EDB_SUBSCRIPTION_TOKEN}
```

### 2.4.5 証明書発行者を作成する

PGD クラスターの展開を開始する前に、証明書発行者を作成してください。Helm チャートはこれを行うように求めますが、見逃した場合に備えて、たとえば次のことを実行できます。

```
kubectl apply -f \
  https://raw.githubusercontent.com/EnterpriseDB/edb-postgres-for-kubernetes-charts/
  ↪main/hack/samples/issuer-selfsigned.yaml
```

オペレーターと自己署名証明書発行者が展開されたら、PGD クラスターの作成を開始できます。例については、[Quick start](#) を参照してください。

## 2.4.6 Red Hat OpenShift

Red Hat OpenShift に Kubernetes 用 EDB Postgres 分散をインストールしようとしている場合は、[Red Hat OpenShift](#) を参照してください。これには、EDB によって維持されている認定オペレーターに関する情報が含まれています。

## 2.5 Quick start

**Kind** で構築された単一のローカル Kubernetes クラスターで Kubernetes 用 EDB

Postgres 分散を使用して、ラップトップまたはコンピューターで EDB Postgres 分散 PGD クラスターをテストできます。

**警告:** これらの指示は、デモンストレーション、テスト、および練習のみを目的としているため、実稼働環境では使用しないでください。

このクイックスタートでは、ローカルの Kubernetes インストールで EDB Postgres 分散クラスターを起動して、実験できる方法を示します。

---

**重要:** Kubernetes クラスターに接続するには、マシンに `kubectl` がインストールされていることを確認します。 *installing `kubectl`* <<https://kubernetes.io/docs/tasks/tools/install-kubectl/>>\_ に関する Kubernetes ドキュメントを参照してください。

---

### 2.5.1 パート 1-ローカル Kubernetes プレイグラウンドをセットアップする

Docker コンテナノードを使用してローカル Kubernetes クラスターを実行するためのツールである Kind をインストールします。Kind は Kubernetes IN Docker の略です。既に Kubernetes クラスターにアクセスしている場合、パート 2 に進んでください。

**Kind Quick Start** の指示に従って、Kind を環境にインストールします。次に、Kubernetes クラスターを作成します。

```
kind create cluster --name pgd
```

## 2.5.2 パート 2-Kubernetes 用に分散された EDB Postgres のインストール

ラップトップで Kubernetes インストールをアップして実行したら、Kubernetes の EDB Postgres Distributed をインストールできます。

詳細は、'[Installation](#)' を参照してください。

## 2.5.3 パート 3-PGD クラスターを展開する

Kubernetes の他のデプロイメントと同様に、PGD クラスターを展開するには、PGD クラスターを構成する目的の PGDGroup リソースを定義する構成ファイルを適用する必要があります。

一部のサンプルファイルは、EDB Postgres Distributed for Kubernetes リポジトリに含まれています。マニフェストには、2つのデータグループと3つのリージョンにまたがるグローバル監視ノードを含む PGD クラスターの定義が含まれています。各データグループは、2つのデータノードと1つのローカル監視ノードで構成されています。

```
apiVersion: pgd.k8s.enterprisedb.io/v1beta1
kind: PGDGroup
metadata:
  name: region-a
spec:
  instances: 2
  proxyInstances: 2
  witnessInstances: 1
  pgd:
    parentGroup:
      name: world
      create: true
    discovery:
      - host: region-a-group
      - host: region-b-group
      - host: region-c-group
  cnp:
    storage:
      size: 1Gi
  connectivity:
    dns:
      additional:
        - domain: alternate.domain
        - domain: my.domain
      hostSuffix: -dc1
    tls:
      mode: verify-ca
      clientCert:
        caCertSecret: client-ca-key-pair
```

(次のページに続く)

(前のページからの続き)

```

    certManager:
      spec:
        issuerRef:
          name: client-ca-issuer
          kind: Issuer
          group: cert-manager.io
    serverCert:
      caCertSecret: server-ca-key-pair
    certManager:
      spec:
        issuerRef:
          name: server-ca-issuer
          kind: Issuer
          group: cert-manager.io
  -- --
apiVersion: pgd.k8s.enterprisedb.io/v1beta1
kind: PGDGroup
metadata:
  name: region-b
spec:
  instances: 2
  proxyInstances: 2
  witnessInstances: 1
  pgd:
    parentGroup:
      name: world
    discovery:
      - host: region-a-group
      - host: region-b-group
      - host: region-c-group
  cnp:
    storage:
      size: 1Gi
  connectivity:
    tls:
      mode: verify-ca
      clientCert:
        caCertSecret: client-ca-key-pair
        certManager:
          spec:
            issuerRef:
              name: client-ca-issuer
              kind: Issuer
              group: cert-manager.io
      serverCert:

```

(次のページに続く)

```

    caCertSecret: server-ca-key-pair
    certManager:
      spec:
        issuerRef:
          name: server-ca-issuer
          kind: Issuer
          group: cert-manager.io
  --
apiVersion: pgd.k8s.enterprisedb.io/v1beta1
kind: PGDGroup
metadata:
  name: region-c
spec:
  instances: 0
  proxyInstances: 0
  witnessInstances: 1
  pgd:
    parentGroup:
      name: world
    discovery:
      - host: region-a-group
      - host: region-b-group
      - host: region-c-group
  cnp:
    storage:
      size: 1Gi
  connectivity:
    tls:
      mode: verify-ca
      clientCert:
        caCertSecret: client-ca-key-pair
        certManager:
          spec:
            issuerRef:
              name: client-ca-issuer
              kind: Issuer
              group: cert-manager.io
      serverCert:
        caCertSecret: server-ca-key-pair
        certManager:
          spec:
            issuerRef:
              name: server-ca-issuer
              kind: Issuer
              group: cert-manager.io

```

最初に保存して実行することにより、`flexible-3-regions` 例を展開できます。

```
kubectl apply -f flexible_3regions.yaml
```

`get pods` コマンドを使用して、ポッドが作成されていることを確認できます。

```
kubectl get pods
```

ポッドは PGD ノードの一部として作成されています。*'Architecture'* で説明しているように、これらは Kubernetes クラスターの EDB Postgres の上に実装されています。

次に、クラスターをリストすることができます。これは、PGD ノードを表示します。

```
$ kubectl get clusters
```

| NAME       | AGE   | INSTANCES | READY | STATUS                          | PRIMARY      |
|------------|-------|-----------|-------|---------------------------------|--------------|
| region-a-1 | 2m50s | 1         | 1     | Cluster <b>in</b> healthy state | region-a-1-1 |
| region-a-2 | 118s  | 1         | 1     | Cluster <b>in</b> healthy state | region-a-2-1 |
| region-a-3 | 91s   | 1         | 1     | Cluster <b>in</b> healthy state | region-a-3-1 |
| ...        |       |           |       |                                 |              |
| ...        |       |           |       |                                 |              |

最終的に、PGD ノードは、PGD クラスターを構成する PGD グループの一部として作成されます。

```
$ kubectl get pgdgroups
```

| NAME     | DATA INSTANCES | WITNESS INSTANCES | PHASE              | AGE   |
|----------|----------------|-------------------|--------------------|-------|
| region-a | 2              | 1                 | PGDGroup - Healthy | 4m50s |
| region-b | 2              | 1                 | PGDGroup - Healthy | 4m50s |
| region-c | 0              | 1                 | PGDGroup - Healthy | 4m50s |

region-c グループが監視ノードにすぎないことに注意してください。



## 第3章 Using

### 3.1 Managing EDB Postgres Distributed (PGD) databases

'*Architecture*' で説明しているように、EDB Postgres Distributed for Kubernetes は、PGD データベースを展開するために作成されたオペレーターです。Kubernetes エコシステムを活用することで、自己修復と宣言的制御を提供できます。オペレーターは、バックアップおよび復元操作も担当します。'*Backup on object stores*' を参照してください。

ただし、PGD クラスターの操作と制御の多くは、オペレーターによって管理されていません。EDB Postgres Distributed for Kubernetes によって作成されたポッドには、**PGD CLI** がインストールされた状態で提供されます。このツールを使用して、スイッチオーバーを実行できます。

#### 3.1.1 PGD CLI

**警告:** PGD CLI を使用してリソースを作成および削除しないでください。たとえば、`create-proxy` および `delete-proxy` コマンドは避けてください。リソースのプロビジョニングはオペレーターの制御下にあり、手動での作成と削除はサポートされていません。

例として、`switchover` コマンドを実行します。

プロキシポッドから PGD CLI を使用することをお勧めします。それらを見つけるには、クラスターのポッドリストを取得します。

```
kubectl get pods -n my-namespace
```

| NAME               | READY | STATUS  | RESTARTS | AGE |
|--------------------|-------|---------|----------|-----|
| location-a-1-1     | 1/1   | Running | 0        | 2h  |
| location-a-2-1     | 1/1   | Running | 0        | 2h  |
| location-a-3-1     | 1/1   | Running | 0        | 2h  |
| location-a-proxy-0 | 1/1   | Running | 0        | 2h  |
| location-a-proxy-1 | 1/1   | Running | 0        | 2h  |

プロキシノードの名前には `proxy` が含まれます。1 つを選択して、コマンドプロンプトを取得します。

```
kubectl exec -n my-namespace -ti location-a-proxy-0 -- bash
```

これで、プロキシポッドとの `bash` セッションが開きました。 `pgd` コマンドは使用できます。

```
pgd
```

Available Commands:

```
check-health      Checks the health of the EDB Postgres Distributed cluster.
<- snipped ->
switchover        Switches over to new write leader.
<- snipped ->
```

スイッチオーバーに必要な情報を取得して、簡単に移動できます。

```
pgd switchover --help
```

```
$ pgd switchover --group-name group_a --node-name bdr-a1
switchover is complete
```

```
pgd show-groups
```

| Group      | Group ID   | Type   | Parent Group | Location | Raft | Routing | Write Leader |
|------------|------------|--------|--------------|----------|------|---------|--------------|
| world      | 3239291720 | global |              |          | true | true    | location-a-2 |
| location-a | 2135079751 | data   | world        |          | true | true    | location-a-1 |

```
pgd show-nodes
```

| Node         | Node ID    | Group      | Type | Current State | Target State | Status | Seq ID |
|--------------|------------|------------|------|---------------|--------------|--------|--------|
| location-a-1 | 3165289849 | location-a | data | ACTIVE        | ACTIVE       | Up     | 1      |
| location-a-2 | 3266498453 | location-a | data | ACTIVE        | ACTIVE       | Up     | 2      |
| location-a-3 | 1403922770 | location-a | data | ACTIVE        | ACTIVE       | Up     | 3      |

### 3.1.2 データベースへのアクセス

'Use cases' は、Kubernetes クラスター内でのデータベースの使用と外部からのデータベースの使用に関するディスカッションです。'Connectivity' では、アプリケーションからデータベースへのアクセスに関連するサービスに関するディスカッションを見つけることができます。

システムを実装する方法にかかわらず、アプリケーションはプロキシサービスを使用して接続し、PGD と EDB Postgres Distributed for Kubernetes オペレーターによって追加された自己修復機能の強化のメリットを享受する必要があります。

**重要:** EDB Postgres Distributed ドキュメントで説明されている `bdrdb` データベースとは対照的に、Kubernetes のデフォルトの EDB Postgres に従って、データノードは `app` と呼ばれるデータベースで作成され、`app` という名前のユーザーが所有します。これらの値は、マニフェストの `cnp` セクションで構成できます。参考については、EDB Postgres for Kubernetes ドキュメントの [Bootstrap](#) を参照してください。

ただし、管理タスクのために psql CLI を使用して PGD データノードにアクセスすることが必要になる場合があります。

PGD クラスターと `kubectl exec` のポッドリストを取得して、データノードに入れることができます。

```
kubectl exec -n my-namespace -ti location-a-1-1 -- psql
```

psql のよく知られた領域では、デフォルトで作成されるデータベースの名前が `app` であることに注意してください。前述の警告を参照してください。

```
postgres=# \c app
You are now connected to database "app" as user "postgres".
app=# \x
Expanded display is on.
app=# select * from bdr.node_summary;
- [ RECORD 1 ]-----
node_name          | location-a-1
node_group_name    | location-a
interface_connstr  | host=location-a-1-node user=streaming_replica sslmode=verify-
↪ca port=5432 sslkey=/controller/certificates/streaming_replica.key sslcert=/
↪controller/certificates/streaming_replica.crt sslrootcert=/controller/certificates/
↪server-ca.crt application_name=location-a-1 dbname=app
peer_state_name    | ACTIVE
peer_target_state_name | ACTIVE

<- snipped ->
```

アプリケーションには、非特権ロールデフォルトで `app` を使用します。

ユーザーの資格情報が必要です。これは Kubernetes シークレットに保存されています。

```
kubectl get secrets
```

| NAME           | TYPE                     | DATA | AGE |
|----------------|--------------------------|------|-----|
| <- snipped ->  |                          |      |     |
| location-a-app | kubernetes.io/basic-auth | 2    | 2h  |

このシークレットには、Postgres DSN に必要な base64 でエンコードされたユーザー名とパスワードが含まれています。

```
kubectl get secrets location-a-app -o yaml
```

```
apiVersion: v1
data:
  password: <base64-encoded-password>
  username: <base64-encoded-username>
kind: Secret
metadata:
```

(次のページに続く)

(前のページからの続き)

```
creationTimestamp: <timestamp>
labels:

<- snippet ->
```

## 3.2 Backup on object stores

Kubernetes の EDB Postgres Distributed は、オブジェクトストアでの物理バックアップと WAL アーカイブを介して PGD クラスターのオンライン/ホットバックアップをサポートします。これは、データベースが常に稼働しており、ダウンタイムが必要なく、ポイントインタイムリカバリ PITR が利用できることを意味します。

### 3.2.1 共通オブジェクトストア

AWS S3、Microsoft Azure Blob Storage、Google Cloud Storage、MinIO Gateway、または S3 互換プロバイダーなど、複数のオブジェクトストアがサポートされています。EDB Postgres Distributed for Kubernetes が Kubernetes の EDB Postgres に依存してオブジェクトストアとの接続を構成することを考えると、

[EDB Postgres for Kubernetes common object stores for backups](#) を参照してください。

詳細については、ドキュメントを参照してください。

---

**重要:** EDB Postgres for Kubernetes ドキュメントのクラウドプロバイダー構成セクションは、`spec.backup.barmanObjectStore` で入手できます。EDB Postgres Distributed for Kubernetes の例では、オブジェクトストアセクションは別のパス `spec.backup.configuration.barmanObjectStore` にあります。

---

### 3.2.2 WAL アーカイブ

WAL アーカイブは、WAL ファイルをオブジェクトストレージに送信するプロセスであり、オンライン/ホットバックアップまたは PITR を実行することが重要です。Kubernetes の EDB Postgres 分散では、各 PGD ノードはオブジェクトストアに WAL ファイルを個別にアーカイブするように設定されます。

WAL アーカイブは PGD Group `spec.backup.configuration.barmanObjectStore` スタンザで定義され、宛先パスとクラウド資格情報が設定されるとすぐに有効になります。WAL ファイルはアップロードされる前に圧縮し、暗号化できます。パラレル WAL アーカイブを有効にすることもできます。

```
apiVersion: pgd.k8s.enterprisedb.io/v1beta1
kind: PGDGroup
[...]
spec:
  backup:
```

(次のページに続く)

(前のページからの続き)

```
configuration:
  barmanObjectStore:
    [...]
  wal:
    compression: gzip
    encryption: AES256
    maxParallel: 8
```

詳細については、[EDB Postgres for Kubernetes WAL archiving](#) ドキュメントを参照してください。

### 3.2.3 スケジュールされたバックアップ

スケジュールされたバックアップは、Kubernetes の EDB Postgres Distributed でバックアップ戦略を構成するための推奨される方法です。PGD グループ `spec.backup.configuration.barmanObjectStore` および `spec.backup.schedulers[].schedule` スタンザが構成されている場合、オペレーターは `Scheduled Backup` リソースを作成する選択したバックアップノードとして PGD データノードの 1 つを選択します。

`.spec.backup.schedulers[].method` フィールドでは、スケジュールされたバックアップ方法を定義できます。2 つのバックアップ方法がサポートされています。

- `volumeSnapshot`
- `barmanObjectStore` デフォルト

複数のスケジューラーを定義できますが、各メソッドは 1 つのスケジューラーでのみ使用できます。つまり、2 つのスケジューラーが同じメソッドを使用することはできません。

オブジェクトストアのバックアップの場合、デフォルトの `barmanObjectStore` 方法では、スタンザ `spec.backup.configuration.barmanObjectStore` を使用して、バックアップと `wal` アーカイブの両方のオブジェクトストア情報を定義します。詳細については、[EDB Postgres for Kubernetes Backup on Object Stores](#) を参照してください。

`volumeSnapshot` バックアップを実行するには、`volumeSnapshot` 方法を選択できます。スタンザ `spec.backup.configuration.barmanObjectStore.volumeSnapshot` は、`volumeSnapshot` 構成を定義するために使用されます。詳細については、[EDB Postgres for Kubernetes Backup on Volume Snapshots](#) を参照してください。

次の例は、バックアップに `volumeSnapshot` メソッドを使用する方法を示しています。WAL アーカイブは引き続き `barman` オブジェクトストアで行われます。

```
apiVersion: pgd.k8s.enterprisedb.io/v1beta1
kind: PGDGroup
[...]
spec:
  backup:
    configuration:
      volumeSnapshot:
```

(次のページに続く)

(前のページからの続き)

```

    className: csi-hostpath-snapclass
  barmanObjectStore:
    destinationPath: "<destination path here>"
    s3Credentials:
      accessKeyId:
        name: backup-storage-creds
        key: ID
      secretAccessKey:
        name: backup-storage-creds
        key: KEY
    wal:
      compression: gzip
      encryption: AES256
      maxParallel: 8
  schedulers:
  - method: volumeSnapshot
    schedule: "@ 0 0 0 ** *"
    backupOwnerReference: self
    suspend: false
    immediate: true

```

2つのバックアップ方法の比較の詳細については、*EDB Postgres for Kubernetes for Object stores or volume snapshots* を参照してください。

**.spec.backup.schedulers[].schedule** フィールドでは、

Go `cron` package format <[https://pkg.go.dev/github.com/robfig/cron#hdr-CRON\\_Expression\\_Format](https://pkg.go.dev/github.com/robfig/cron#hdr-CRON_Expression_Format)>`\_`  
で表現される cron スケジュールを定義できます。

```

apiVersion: pgd.k8s.enterprisedb.io/v1beta1
kind: PGDGroup
[...]
spec:
  backup:
    schedulers:
    - method: barmanObjectStore
      schedule: "@ 0 0 0 ** *"
      backupOwnerReference: self
      suspend: false
      immediate: true

```

必要に応じて、**.spec.backup.schedulers[].suspend** を **true** に設定することにより、スケジュールされたバックアップを一時停止できます。これにより、新しいバックアップがスケジュールされなくなります。

**ScheduledBackup** リソースが作成されたらすぐにバックアップを実行したい場合は、**.spec.backup.schedulers[].immediate** を **true** に設定します。

**.spec.backupOwnerReference** は、作成したバックアップリソースで使用する **ownerReference** を示しま

す。選択肢は次のとおりです。

- **none** — 作成されたバックアップオブジェクトの所有者参照はありません。
- **self** — `ScheduledBackup` オブジェクトをバックアップの所有者として設定します。
- **cluster** — クラスターをバックアップの所有者として設定します。

**警告:** `.spec.backup.cron` フィールドは非推奨になりました。代わりに `.spec.backup.schedulers` を使用してください。 `.spec.backup.cron` は引き続き使用できますが、`.spec.backup.schedulers` と同時に使用できないことに注意してください。

**注釈:** EDB Postgres for Kubernetes `ScheduledBackup` オブジェクトには、バックアップするクラスターを指定する `cluster` オプションが含まれています。このオプションは現在 EDB Postgres Distributed for Kubernetes ではサポートされておらず、指定されても無視されます。

選択されたバックアップノードが削除される場合、オペレーターは新しいバックアップノードを透過的に選択し、それに応じて `ScheduledBackup` リソースを調整します。

### 3.2.4 リテンション ポリシー

Kubernetes の EDB Postgres Distributed は、リカバリウィンドウに基づいて保持ポリシーを使用して、バックアップオブジェクトストアからのバックアップファイルの自動削除を管理できます。このプロセスは、未使用の WAL ファイルと、削除がスケジュールされているバックアップに関連付けられている WAL の削除も行います。

30 日の保持ポリシーを使用してバックアップを定義できます。

```
apiVersion: pgd.k8s.enterprisedb.io/v1beta1
kind: PGDGroup
[...]
spec:
  backup:
    configuration:
      retentionPolicy: "30d"
```

詳細については、EDB Postgres for Kubernetes ドキュメントの [EDB Postgres for Kubernetes retention policies](#) を参照してください。

**重要:** 現在、保持ポリシーは、選択された `Backup Node` バックアップと WAL ファイルにのみ適用されます。他の各 PGD ノードも自分の WAL を独立してアーカイブすることを考えると、オブジェクトストレージデータ保持ポリシーを活用するなど、これらの WAL ファイルのライフサイクルを管理するのはあなたの責任です。また、すべての PGD ノードディレクトリにオブジェクトストレージデータ保持ポリシーを設定している場合、オペレーターが管理する保持ポリシーと重複または干渉していないことを確認します。

### 3.2.5 圧縮アルゴリズム

バックアップと WAL ファイルはデフォルトで非圧縮です。ただし、複数の圧縮アルゴリズムがサポートされています。詳細については、[EDB Postgres for Kubernetes compression algorithms](#) ドキュメントを参照してください。

### 3.2.6 バックアップオブジェクトのタグ付け

バックアップオブジェクト、つまりベースバックアップ、WAL ファイル、履歴ファイルのキーと値のペアとしてタグを指定できます。詳細については、[tagging of backup objects](#) に関する EDB Postgres for

Kubernetes ドキュメントを参照してください。

### 3.2.7 PGD ノードのオンデマンドバックアップ

PGD ノードは、Kubernetes Cluster オブジェクトの単一インスタンス EDB Postgres として表されます。そのため、必要に応じて、Kubernetes Backup リソースの EDB Postgres を作成することにより、特定の PGD ノードのオンデマンドバックアップを要求できます。これを行うには、EDB Postgres for Kubernetes ドキュメントの [EDB Postgres for Kubernetes on-demand backups](#) を参照してください。

---

**ヒント:** `kubectrl get cluster -l k8s.pgd.enterprisedb.io/group=my-pgd-group -n my-namespace` を実行して、PGD グループを構成する Kubernetes クラスターの EDB Postgres のリストを取得できます。

---

## 3.3 Recovery

Kubernetes の EDB Postgres 分散では、PGD ノードの使用可能な物理バックアップから開始して、新しい PGD グループをブートストラップする方法として、リカバリを利用できます。リカバリは、既存の PGD グループでインプレースで実行できません。

Kubernetes の EDB Postgres 分散は、ポイントインタイムリカバリー PITR もサポートしており、カタログで最初に使用可能なバックアップから最後にアーカイブされた WAL まで、任意のポイント・インタイムに PGD グループを復元できます。PITR には WAL アーカイブがあることが必須です。

### 3.3.1 前提条件

バックアップから復元する前に

- 復旧したクラスターの PostgreSQL 構成 `.spec.cnp.postgresql.parameters` が、物理レプリケーションの観点から元の構成と互換性があることを確認します。
- 新しく作成した名前空間で復旧する場合、復旧した PGD グループを展開する前に、最初に cert-manager CA 発行者を設定します。

詳細については、EDB Postgres for Kubernetes ドキュメントの *EDB Postgres for Kubernetes recovery - Additional considerations* を参照してください。

### 3.3.2 オブジェクトストアからのリカバリー

Barman Cloud によって作成され、サポートされているオブジェクトストレージに保存されている PGD ノードバックアップから復旧できます。

たとえば、バックアップが使用可能な 3 つのインスタンスを持つ `pgdgroup-example` という名前の PGD グループがあると、オブジェクトストレージには各ノードのディレクトリが含まれます。

`pgdgroup-example-1`、`pgdgroup-example-2`、`pgdgroup-example-3`

この例では、オブジェクトストアからの完全なリカバリーを定義します。オペレーターは、定義された `serverNames` から最新のバックアップを透過的に選択し、最後の使用可能な WAL まで再生します。

```
apiVersion: pgd.k8s.enterprisedb.io/v1beta1
kind: PGDGroup
metadata:
  name: pgdgroup-restore
spec:
  [...]
  restore:
    serverNames:
      - pgdgroup-backup-1
      - pgdgroup-backup-2
      - pgdgroup-backup-3
    barmanObjectStore:
      destinationPath: "<destination path here>"
      s3Credentials:
        accessKeyId:
          name: backup-storage-creds
          key: ID
        secretAccessKey:
          name: backup-storage-creds
          key: KEY
      wal:
        compression: gzip
```

(次のページに続く)

(前のページからの続き)

```

encryption: AES256
maxParallel: 8

```

**重要:** ソースクラスターに従って、WAL セクションを正しく構成してください。この例では、*pgdgroup-example* PGD グループは `compression` および `encryption` を使用しているため、*restore* によって作成されている PGD グループでも適切なパラメーターを設定してください。

**注釈:** この例では、並列 WAL リストア機能を利用し、最大 8 つのジョブを専用にアーカイブから必要な WAL ファイルを同時に取得します。この機能により、復旧時間を大幅に短縮できます。このシナリオを事前に計画し、ご使用の環境に合わせてこのパラメーターの値を調整してください。必要なときに違いを生みます。

### 3.3.3 オブジェクトストアからの PITR

ベースバックアップを抽出した後、最新の WAL までのすべての WAL を再生するのではなく、任意の時点で WAL の再生を停止するように PostgreSQL に依頼できます。PostgreSQL は、この技術を使用して PITR を達成します。WAL アーカイブの存在は必須です。

この例では、リカバリーの時間ベースターゲットを定義します。

```

apiVersion: pgd.k8s.enterprisedb.io/v1beta1
kind: PGDGroup
metadata:
  name: pgdgroup-restore
spec:
  [...]
  restore:
    recoveryTarget:
      targetTime: "2023-08-11 11:14:21.000000+02"
    serverNames:
      - pgdgroup-backup-1
      - pgdgroup-backup-2
      - pgdgroup-backup-3
    barmanObjectStore:
      destinationPath: "<destination path here>"
      s3Credentials:
        accessKeyId:
          name: backup-storage-creds
          key: ID
        secretAccessKey:
          name: backup-storage-creds

```

(次のページに続く)

(前のページからの続き)

```

    key: KEY
  wal:
    compression: gzip
    encryption: AES256
    maxParallel: 8

```

**重要:** PITR では、**リカバリーターゲット** で説明するオプションを使用して、*targetTime* リカバリーターゲットを指定する必要があります。 *targetTime* または `targetLSN`` を使用する場合、オペレーターは、そのターゲットより前に完了した最も近いバックアップを選択します。それ以外の場合、指定された `serverNames`` 間で時系列で最後の使用可能なバックアップを選択します。

### 3.3.4 backupID を指定したオブジェクトストアからのリカバリー

`.spec.restore.recoveryTarget.backupID` オプションを使用すると、リカバリプロセスを開始するベースバックアップを指定できます。デフォルトでは、この値は空です。値を割り当てると、オペレーターはそのバックアップをリカバリーのベースとして使用します。値は Barman バックアップ ID の形式である必要があります。

この例では、pgdgroup-backup-1 PGD ノードの特定のバックアップ ID から新しい PGD グループを復旧します。

```

apiVersion: pgd.k8s.enterprisedb.io/v1beta1
kind: PGDGroup
metadata:
  name: pgdgroup-restore
spec:
  [...]
  restore:
    recoveryTarget:
      backupID: 20230824T133000
      serverNames:
        - pgdgroup-backup-1
    barmanObjectStore:
      destinationPath: "<destination path here>"
      s3Credentials:
        accessKeyId:
          name: backup-storage-creds
          key: ID
        secretAccessKey:
          name: backup-storage-creds
          key: KEY
      wal:

```

(次のページに続く)

(前のページからの続き)

```
compression: gzip
encryption: AES256
maxParallel: 8
```

**重要:** *backupID* が指定される場合、*serverNames* オプションで関連する PGD ノードのみをリストし、他のものはリストしないようにしてください。

**注釈:** 特定の ``backupID`` の定義は、次のリカバリターゲット *targetName*、*targetXID*、および ``targetImmediate`` のいずれかを使用する場合に特に必要です。このような場合、カタログ内の最後の使用可能なバックアップに問題がない場合を除き、*backupID* を指定することが重要です。

### 3.3.5 volumeSnapshot からのリカバリー

volumeSnapshot バックアップから pgdgroup を復旧することもできます。スタンザ `spec.restore.volumeSnapshots` は、volumeSnapshots リストア候補の基準を定義するために使用されます。オペレーターは、候補の中から最新の volumeSnapshot を透過的に選択します。

オペレーターには、volumeSnapshot に次のアノテーション/ラベルが必要です。これらの注釈/ラベルは、オペレーターが volumeSnapshot を撮影した場合、自動的に追加されます。

注釈

- `k8s.enterprisedb.io/backupEndTime` は、最新のスナップショットを比較および選択するために使用されます。
- `k8s.enterprisedb.io/pvcRole` は、volumeSnapshot の pvcRole を表します。サポートされているロールには、PG\_WAL および PG\_DATA が含まれます。

ラベル

- `k8s.enterprisedb.io/cluster` は、volumeSnapshot が取得されるノードを示します。WAL 再生のオブジェクトストアで *serverName* を取得するために重要です。
- `k8s.enterprisedb.io/backupName` は、volumeSnapshot のバックアップ名であり、バックアップでより多くのボリュームが定義されている場合に、volumeSnapshot をグループ化するために使用されます。
- `k8s.enterprisedb.io/tablespaceName` は、volumeSnapshot ロールが PG\_TABLESPACE の場合、volumeSnapshot のテーブルスペース名を表します。

次の例は、volumeSnapshots からの完全なリカバリーを示しています。volumeSnapshot リカバリーの後、完全なリカバリーの WAL 再生はターゲットサーバー `pgdgroup-backup-vs-1` になります。

```
apiVersion: pgd.k8s.enterprisedb.io/v1beta1
kind: PGDGroup
```

(次のページに続く)

(前のページからの続き)

```

metadata:
  name: pgdgroup-restore
spec:
  [...]
  restore:
    volumeSnapshots:
      selector:
        matchLabels:
          "k8s.enterprisedb.io/cluster": pgdgroup-backup-vs-1
          "k8s.pgd.enterprisedb.io/group": pgdgroup-backup-vs
    barmanObjectStore:
      destinationPath: "<destination path here>"
      s3Credentials:
        accessKeyId:
          name: backup-storage-creds
          key: ID
        secretAccessKey:
          name: backup-storage-creds
          key: KEY
      wal:
        compression: gzip
        encryption: AES256
        maxParallel: 8

```

詳細については、*EDB Postgres for Kubernetes recovery from volumeSnapshot objects* を参照してください。

### 3.3.6 volumeSnapshot からの PITR

オブジェクトストアからリカバリを行う場合と同じように、volumeSnapshot リカバリ中の特定の瞬間に、先行書き込みログ WAL の再生を停止するように PostgreSQL に指示できます。

この例では、ボリュームスナップショットを使用したリカバリーの時間ベースのターゲットを設定する方法を示しています。

```

apiVersion: pgd.k8s.enterprisedb.io/v1beta1
kind: PGDGroup
metadata:
  name: pgdgroup-restore
spec:
  [...]
  restore:
    recoveryTarget:
      targetTime: "2023-08-11 11:14:21.000000+02"
    volumeSnapshots:

```

(次のページに続く)

```
selector:
  matchLabels:
    "k8s.enterprisedb.io/cluster": pgdgroup-backup-vs-1
    "k8s.pgd.enterprisedb.io/group": pgdgroup-backup-vs
barmanObjectStore:
  destinationPath: "<destination path here>"
  s3Credentials:
    accessKeyId:
      name: backup-storage-creds
      key: ID
    secretAccessKey:
      name: backup-storage-creds
      key: KEY
wal:
  compression: gzip
  encryption: AES256
  maxParallel: 8
```

### 3.3.7 リカバリーターゲット

PITR を超えて、使用できる他の回復ターゲット基準があります。使用可能なすべてのリカバリターゲットの詳細については、EDB Postgres for Kubernetes ドキュメントの「」を参照してください。

## 3.4 Security

EDB Postgres Distributed for Kubernetes のセキュリティは、コード、コンテナ、クラスターの 3 つのレイヤーで分析されます。

**警告:** ここで説明するセキュリティ対策に加えて、Kubernetes クラスターで通常の InfoSec 業務を実行する必要があります。 *Overview of Cloud Native Security* を理解する

Kubernetes ドキュメント

参考:

*The 4C's Security Model in Kubernetes* を参照

Kubernetes の EDB Postgres Distributed のセキュリティに対して EDB が採用するアプローチのより良い理解とコンテキストに関するブログ記事。

### 3.4.1 コード

Kubernetes 用 EDB Postgres 分散のソースコードは、セキュリティ問題を含む静的分析のために体系的にスキャンされています。EDB は、[GolangCI-Lint](#) と呼ばれる Go の人気のオープンソースリンターを CI/CD パイプラインで直接使用します。GolangCI-Lint は、同じソースコードで複数のリンターを実行できます。

これらの 1 つが [Golang Security Checker](#) または `gosec` です。gosec は、コードに隠された既知の脆弱性、脅威、および弱点を検出することを目的としたルールセットと比較して、ソースの抽象構文ツリーをスキャンするリンターです。これらのスレッドには、ハードコーディングされた資格情報、整数オーバーフロー、SQL インジェクションなどが含まれます。

---

**重要:** CI/CD パイプラインの静的コード分析フェーズの障害は、Kubernetes 用に配布された EDB Postgres の配信全体のブロッカーです。つまり、各コミットは GolangCI-Lint で定義されたすべてのリンターに対して検証されます。

---

### 3.4.2 コンテナ

EDB Postgres Distributed for Kubernetes の一部であるすべてのコンテナイメージは、すべてのコミットに続いて CI/CD パイプラインを介してビルドされます。このようなイメージには、演算子だけでなくオペランド、特にサポートされているすべての PostgreSQL バージョンのイメージが含まれます。パイプラインでは、画像は次の方法でスキャンされます。

- [Docker](#) コンテナビルドプロセスに関するベストプラクティス
- [Clair](#) 基になるオペレーティングシステムと、それらが実行するライブラリとアプリケーションの両方で見つかった脆弱性

---

**重要:** すべてのオペランドイメージは、ベースイメージとパッケージレベルでセキュリティ更新が必要な場合、パイプラインによって 1 日 1 回再ビルドされ、EDB が配布するコンテナイメージのパッチレベルの更新を提供します。

---

コンテナレベルのセキュリティでは、次のガイドラインとフレームワークが考慮されました。

- [Container Image Creation and Deployment Guide](#)、米国国防総省 DoD の防衛情報システム庁 DISA によって開発されました
- [CIS Benchmark for Docker](#)、インターネットセキュリティセンター CIS によって開発されました

参考:

[Security and Containers in EDB Postgres Distributed for Kubernetes](#) を参照してください

EDB Postgres Distributed for Kubernetes のコンテナレベルでセキュリティに対して EDB が採用するアプローチの詳細については、ブログ記事を参照してください。

### 3.4.3 クラスター

クラスターレベルのセキュリティでは、コントロールプレーンとノードの両方を形成するすべての Kubernetes コンポーネント、および PostgreSQL を含むクラスターで実行されるアプリケーションを考慮します。

#### ロールベースのアクセス制御 RBAC

オペレーターは、pgd-operator-controller-manager と呼ばれる専用のサービスアカウントを使用して Kubernetes API サーバーと対話します。Kubernetes では、このアカウントはデフォルトで `pgd-operator-system` 名前空間にインストールされます。クラスターロールは、このサービスアカウントと、オペレーターに付与されるルール、リソース、および動詞のセットを定義する `pgd-operator-controller-manager` クラスターロールの間をバインドします。

RedHat OpenShift は、*OperatorLifecycleManager (OLM)* を介してオペレーター RBAC エンティティを直接管理します。OLM では、最小特権の原則を実装して、必要な場合にのみ権限を付与できます。

---

**重要:** これらの権限は、Kubernetes API サーバーと対話するためにオペレーターのサービスアカウント専用に予約されています。 `PGDGroup` および `PGDGroupCleanup` リソースとのみ対話するオペレーターのユーザーは、これらに直接アクセスできません。

---

以下は、いくつかの例と、最も重要な理由です。Kubernetes の EDB Postgres Distributed は、標準の Kubernetes 名前空間リソースの完全または部分的な管理を必要とする理由です。

**jobs** オペレーターは、ジョブを処理して、さまざまな `PGDGroup` フェーズを管理する必要があります。

**poddisruptionbudgets** オペレーターはポッド中断バジェットを使用して、メンテナンス操作中に十分な PGD ノードがアクティブに保たれるようにします。

**Pods** オペレーターは、PGD ノードを `Cluster` リソースとして管理する必要があります。

**secrets** 証明書とパスワードをデータノードに提供しない限り、オペレーターは、ランダムに生成されたパスワードと TLS 証明書を自己プロビジョニングし、シークレットに保存することにより、「構成より規約」パラダイムを採用します。

**serviceaccounts** オペレーターはサービスアカウントを作成して、`PGDGroup` リカバリジョブが存在するオブジェクトストアからバックアップオブジェクトを取得できるようにする必要があります。

**services** オペレーターは、アプリケーションから PGD クラスターへのネットワークアクセスを制御し、自動化された方法でフェイルオーバー/スイッチオーバー操作を適切に管理する必要があります。

**statefulsets** オペレーターは PGD プロキシを管理する必要があります。

**validatingwebhookconfigurations** および **mutatingwebhookconfigurations** オペレーターは、管理するすべてのリソースを検証および変更するために必要な両方の Webhook 構成に、自己署名 Webhook CA を注入します。詳細については、[Kubernetes documentation](#) を参照してください。

オペレーターに必要なすべての権限を表示するには、`kubectl describe clusterrole pgd-operator-manager-role` を実行できます。

EDB Postgres Distributed for Kubernetes は、EDB Postgres for Kubernetes で定義された Cluster リソースを使用して PGD ノードを内部管理します。['Security'](#) を参照してください

Kubernetes オペレーターサービスアカウントの EDB Postgres が使用する権限のリストは。

### インスタンスマネージャーによる API サーバーへの呼び出し

オペランドコンテナのエントリーポイントであるインスタンスマネージャーは、Kubernetes API サーバーへのいくつかの呼び出しを行って、一部のリソースのステータスが正しく更新されることを確認し、その Postgres クラスターに関連付けられている構成マップとシークレットにアクセスする必要があります。このような呼び出しは、同じ PostgreSQL Cluster リソース名を共有するオペレーターによって作成された専用の ServiceAccount を介して実行されます。

---

**重要:** このオペランドは、API サーバーを介してリソースの特定の限定されたサブセットにのみアクセスできます。サービスアカウントは、ポッド内から API サーバーにアクセスするための推奨される方法です。詳細は、[Kubernetes documentation](#) を参照してください。

---

['Security'](#) を参照してください

インスタンスマネージャーの詳細については、

### ポッドセキュリティポリシー

#### *pod security policy*

これは、クラスターで実行するためにポッドが満たす必要があるセキュリティルールと仕様を定義する Kubernetes 方法です。InfoSec の理由から、すべての Kubernetes プラットフォームはそれらを実装する必要があります。

EDB Postgres Distributed for Kubernetes は、コンテナの実行に特権モードを必要としません。PostgreSQL コンテナは、postgres システムユーザーとして実行されます。root として実行する必要があるコンポーネントはありません。

同様に、ボリュームへのアクセスには特権モードまたは root 権限は必要ありません。Kubernetes プラットフォームまたは管理者が適切な権限を割り当てる必要があります。PostgreSQL コンテナは、読み取り専用のルートファイルシステム、つまり書き込み可能なレイヤーなしで実行されます。

オペレーターは、必要なセキュリティコンテキストを明示的に設定します。

Red Hat OpenShift では、クラウドネイティブ PostgreSQL は、最も制限の厳しい **restricted** セキュリティコンテキスト制約で実行されます。目標は、ポッドの実行を名前空間に割り当てられた UID と SELinux コンテキストに制限することです。

#### 参考:

OpenShift のセキュリティコンテキスト制約 SCC の詳細については、[Managing SCC in OpenShift](#) を参照してください。

記事。

## 公開ポート

EDB Postgres Distributed for Kubernetes は、表に示すように、オペレーター、インスタスマネージャー、およびオペランドレベルでポートを公開します。

| System           | Port number | Exposing            | Name                  | Certificates | Authentication |
|------------------|-------------|---------------------|-----------------------|--------------|----------------|
| operator         | 9443        | webhook server      | <i>webhook-server</i> | TLS          | Yes            |
| operator         | 8080        | metrics             | <i>metrics</i>        | no TLS       | No             |
| instance manager | 9187        | metrics             | <i>metrics</i>        | no TLS       | No             |
| instance manager | 8000        | status              | <i>status</i>         | no TLS       | No             |
| operand          | 5432        | PostgreSQL instance | <i>postgresql</i>     | optional TLS | Yes            |

## PGD

Kubernetes の EDB Postgres Distributed の現在の実装は、postgres スーパーユーザーとデータベース所有者のパスワードを作成します。

パスワードの暗号化に関する限り、Kubernetes の EDB Postgres Distributed は PostgreSQL のデフォルト動作に従います。PostgreSQL 14 以降、`password_encryption` はデフォルトで `scram-sha-256` に設定されます。以前のバージョンでは、`md5` に設定されています。

---

**重要:** Connection DSNs and SSL を参照

---

詳細については、PGD ドキュメントを参照してください。

EDB Postgres Distributed for Kubernetes は、`postgres` ロールを使用して PGD ノードを維持します。このため、仕様の `cnf` セクションで `enableSuperuserAccess` から `false` を設定することにより、`postgres` ユーザーアクセスを無効にすることはできません。

## ストレージ

EDB Postgres Distributed for Kubernetes は、保存時の暗号化を基になるストレージクラスに委任します。運用環境でのデータ保護のために、保存時の暗号化をサポートするストレージクラスを選択することを強くお勧めします。

## 3.5 Connectivity

PGD クラスター内の安全なネットワーク通信に関する情報には次のものが含まれます。

- [サービス](#)
- 完全修飾ドメイン名 FQDN を使用した [ドメイン名解決](#)
- [TLS 構成](#)

### 3.5.1 サービス

PGD クラスター内のリソースは、Kubernetes サービスを介してアクセスできます。すべての PGD グループは、それらのいくつかを管理します。つまり

- ノードごとに1つのサービス、内部通信に使用されます [ノードサービス](#)
- グループ内の任意のノードに到達する [グループサービス](#)。主に EDB Postgres Distributed for Kubernetes がクラスター内の新しいグループを検出するために使用します
- PGD プロキシを使用して、アプリケーションが透過的にグループの書き込みリーダーに到達できるようにする [プロキシサービス](#)
- アプリケーションが PGD プロキシを透過的に使用して、グループの読み取りノードに到達できるようにする [プロキシ読み取りサービス](#)。このサービスはデフォルトで無効になっており、`.spec.proxySettings.enableReadNodeRouting` 設定によって制御されます

これらのサービスを使用する例については、[アプリケーションから PGD クラスターに接続する](#) を参照してください。

各サービスは、マニフェストの `.spec.connectivity` セクションのカスタマイズ可能なテンプレートから生成されます。

すべてのサービスは、すべての Kubernetes クラスターのすべての PGD ノードから FQDN を使用して到達可能である必要があります。[ドメイン名解決](#) を参照してください。

EDB Postgres Distributed for Kubernetes は、次の3つのレベルでサービスを簡単にカスタマイズできる可用性を提供するサービステンプレートフレームワークを提供します。

ノードサービステンプレート 各 PGD ノードは、`.spec.connectivity.nodeServiceTemplate` セクションで構成できるサービスを使用して到達可能です。

グループサービステンプレート 各 PGD グループには、グループ全体の単一のエン트리ポイントであり、`.spec.connectivity.groupServiceTemplate` セクションで構成できるグループサービスがあります。

プロキシサービステンプレート 各 PGD グループには、PGD プロキシを介してグループ書き込みリーダーに到達するプロキシサービスがあり、`.spec.connectivity.proxyServiceTemplate` セクションで構成できます。これは、アプリケーションのエン트리ポイントサービスです。

プロキシ読み取りサービステンプレート 各 PGD グループには、PGD プロキシを介してグループ読み取りノードに到達するプロキシサービスがあり、`.spec.proxySettings.enableReadNodeRouting` で有効にでき、`.spec.connectivity.proxyReadServiceTemplate` セクションで構成できます。これは、アプリケーションのエン트리ポイントサービスです。

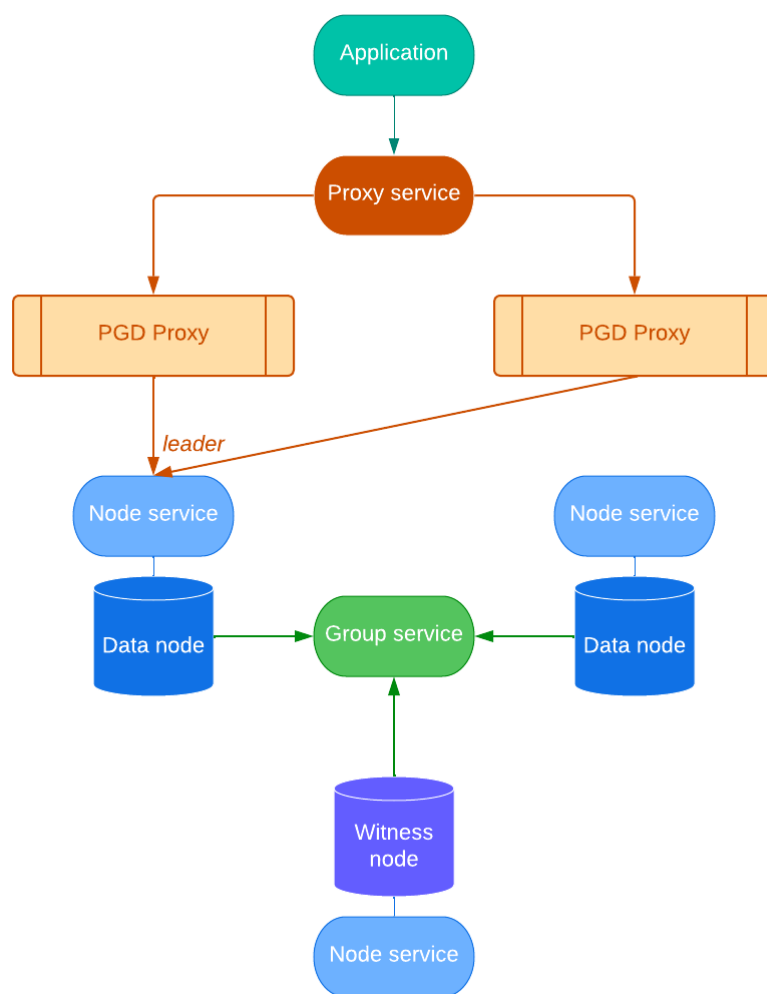


図 1: Basic architecture of an EDB Postgres Distributed for Kubernetes PGD group

テンプレートを使用して、LoadBalancer サービスを作成したり、任意のアノテーションとラベルをサービスに追加して、Kubernetes システムで使用可能な他のコンポーネントと統合したり、外部 DNS 名を作成したり、生成されたロードバランサーを調整したりできます。

### 3.5.2 ドメイン名解決

Kubernetes の EDB Postgres Distributed は、PGD グループ名をすべてのプレフィックスとして使用する規則を採用することにより、PGD グループ内のすべてのリソースが FQDN を持つことを保証します。

その結果、PGD グループのドメイン名を定義することが想定されます。これは、`.spec.connectivity.dns` セクションを介して行うことができます。このセクションは、2つのフィールドでリソースの FQDN を生成する方法を制御します。

- `domain` —使用する PGD グループ内のすべてのオブジェクトのドメイン名必須。
- `hostSuffix` —PGD グループの各サービスに追加するサフィックスオプション。

### 3.5.3 TLS 構成

EDB Postgres Distributed for Kubernetes では、PGD クラスター内のリソースが安全な接続を介して通信する必要があります。PostgreSQL の [SSL connections](#) のネイティブサポートに依存しています

セキュリティを向上させるために TLS プロトコルを使用してクライアント/サーバー通信を暗号化します。

現在、EDB Postgres Distributed for Kubernetes では、

`cert-manager` をインストールする必要があります。

Cert-manager は、Kubernetes 環境の標準として広く認識されているため、動的証明書をプロビジョニングするツールとして選択されました。

`spec.connectivity.tls` セクションでは、ノード間の通信がどのように発生するかを説明します。

- `mode` は、PGD グループノードの通信中にサーバー証明書が検証される方法を説明する列挙体です。[SSL Support](#) に記載されているように、次の値を受け入れます。

PostgreSQL ドキュメント内

- `verify-full`
- `verify-ca`
- `required`
- `serverCert` は、PGD グループノードが要求を受け入れるために使用するサーバー証明書を定義します。クライアントは、渡された TLS モードに応じてこの証明書を検証します。`mode` と同じ値を受け入れます。
- `clientCert` は、ノードが相互に認証するために使用する `streaming_replica` ユーザー証明書を定義します。

## サーバー TLS 構成

サーバー証明書構成は、PGDGroup カスタムリソースの `.spec.connectivity.tls.serverCert.certManager` セクションで指定されます。

このセクションが機能するためには、次の前提条件が作成されました。

- 発行者 `.spec.connectivity.tls.serverCert.certManager.issuerRef` は、ドメイン `.spec.connectivity.dns.domain` および `.spec.connectivity.tls.serverCert.certManager.altDnsNames` が使用するその他のドメインで利用できます。
- 発行者 `.spec.connectivity.tls.serverCert.caCertSecret` が使用する CA のパブリック証明書を包含シークレットがあります。

`.spec.connectivity.tls.serverCert.certManager` は、ノードごとの `cert-manager` 証明書要求を作成するために使用されます。結果の証明書は、基になる Postgres インスタンスが TLS 接続を終了するために使用されます。

オペレーターは、次の `altDnsNames` を証明書に追加します。

- `$node$hostSuffix.$domain`
- `$groupName$hostSuffix.$domain`

---

**重要:** 基になるネットワークアーキテクチャ、たとえば、ユーザーがノードに到達するために使用するロードバランサーから必要な名前を `.spec.connectivity.tls.serverCert.certManager.altDnsNames` に追加するのは自分の責任です。

---

## クライアント TLS 構成

オペレーターは、`cert-manager` 推奨アプローチを使用してクライアント証明書を動的にプロビジョニングするか、シークレットを使用して事前プロビジョニングを必要とします。

### cert-manager を介した動的プロビジョニング

クライアント証明書構成は、PGDGroup カスタムリソースの `.spec.connectivity.tls.clientCert.certManager` セクションで管理されます。このセクションが機能するためには、次の前提条件が作成されました。

- 発行者 `.spec.connectivity.tls.clientCert.certManager.issuerRef` が利用でき、一般名 `streaming_replica` で証明書に署名します。
- 発行者 `.spec.connectivity.tls.clientCert.caCertSecret` が使用する CA のパブリック証明書を包含シークレットがあります。

オペレーターは、`.spec.connectivity.tls.clientCert.certManager` の下の構成を使用して、`streaming_replica` Postgres ユーザーごとに証明書要求を作成します。結果の証明書は、ノード間の通信を保護するために使用されます。

## シークレットを介した事前プロビジョニングされた証明書

または、`.spec.connectivity.tls.clientCert.preProvisioned.streamingReplica.secretRef` オプションを使用して、ストリーミングレプリケーションユーザーの事前プロビジョニングされたクライアント証明書をを含むシークレットを指定できます。この場合の証明書ライフサイクルは、シークレットの内容を更新することにより、サードパーティが手動または自動で完全に管理します。

### 3.5.4 アプリケーションから PGD クラスターに接続する

同じ Kubernetes クラスター内またはクラスターの外部から実行されているアプリケーションから PGD グループに接続するのは、簡単な手順です。どちらの場合も、`app` ユーザーとして PGD グループのプロキシサービスに接続します。プロキシサービスは、どのプロキシサービスが接続しているかに応じて、PGD グループの書き込みリーダーまたは読み取りノードに接続をルーティングする LoadBalancer サービスです。

#### クラスター内からの接続

クラスター内から接続する場合、プロキシサービス名を使用して PGD グループに接続できます。プロキシサービス名は、PGD グループ名と、PGDGroup カスタムリソースの `.spec.connectivity.dns` セクションで定義されたオプションのホストサフィックスで構成されています。

たとえば、PGD グループ名が `my-group` で、ホストサフィックスが `.my-domain.com` の場合、プロキシサービス名は `my-group.my-domain.com` になります。

接続する前に、アプリユーザーのシークレットからアプリユーザーのパスワードを取得する必要があります。シークレットの命名形式は、`my-group` という名前の PGD グループの場合 `my-group-app` です。

次のコマンドを使用して、シークレットからユーザー名とパスワードを取得できます。

```
kubectl get secret my-group-app -o jsonpath={.data.username} | base64 --decode
kubectl get secret my-group-app -o jsonpath={.data.password} | base64 --decode
```

これにより、PGD グループへの接続文字列のすべての部分が揃います。

```
postgresql://<app-user>:<app-password>@<proxy-service-name>:5432/<database>
```

または、`psql` 呼び出しの場合

```
psql -U <app-user> -h <proxy-service-name> <database>
```

`app-user` および `app-password` はシークレットから取得した値であり、`database` は接続するデータベースの名前です。アプリユーザーのデフォルトは `app` です。

## Kubernetes クラスターの外部からの接続

Kubernetes クラスターの外部から接続する場合、一般的には、

Ingress リソースまたは load balancer が必要です。環境での動作の詳細については、クラウドプロバイダーまたはローカルインストールを確認してください。

イングレスとロードバランサーでは、接続を PGD プロキシに転送するためにポッドセクターが必要です。それらを構成するときは、次のラベルを使用することをお勧めします。

- `k8s.pgd.enterprisedb.io/group` — PGD グループ名を設定します。
- `k8s.pgd.enterprisedb.io/workloadType` — `pgd-proxy` に設定します。

ローカル開発に Kind または他のソリューションを使用している場合、外部から PGD グループにアクセスする最も簡単な方法は、プロキシサービスへのポート転送を使用することです。次のコマンドを使用して、ローカルマシンのポート 5432 をプロキシサービスに転送できます。

```
kubect1 port-forward svc/my-group.my-domain.com 5432:5432
```

`my-group.my-domain.com` は、前の例のプロキシサービス名です。

## 3.6 Certificates

Kubernetes の EDB Postgres Distributed は、TLS 証明書をネイティブにサポートするように設計されました。PGD クラスターをセットアップするには、各 PGD ノードに以下が必要です。

- サーバー認証局 CA 証明書
- サーバー CA によって署名されたサーバー TLS 証明書
- クライアント CA 証明書
- クライアント CA によって生成されたストリーミングレプリケーションクライアント証明書

---

**注釈:** クラスター PGD ノードのステータスで、各 PGD ノードが使用するすべてのシークレットと有効期限を見つけることができます。

---

Kubernetes の EDB Postgres Distributed は、TLS 証明書に関して非常に柔軟です。主に 2 つのモードで動作します。

- **オペレーター管理**—証明書は、オペレーターによって完全に自動化された方法で内部管理され、EDB Postgres Distributed for Kubernetes によって作成された CA を使用して署名されます。
- **ユーザー指定**—証明書はオペレーターの外部で生成され、シークレットとしてクラスター定義にインポートされます。EDB Postgres Distributed for Kubernetes は、`cert-manager` と `self` を統合します。

詳細については、[EDB Postgres for Kubernetes documentation](#) を参照してください。

### 3.7 Client TLS/SSL connections

参考:

['Certificates'](#) を参照

EDB Postgres Distributed for Kubernetes が TLS 証明書をサポートする方法の詳細については、こちらをご覧ください。

EDB Postgres Distributed for Kubernetes オペレーターは、転送中の暗号化とサーバーサイドとクライアントサイドでの認証の両方で TLS / SSL と連携するように設計されました。PGD ノードは、EDB Postgres for Kubernetes オペレーターを使用してクラスターリソースとして作成されます。これには、TLS クライアント証明書を作成および署名するための認証局 CA を展開することが含まれます。

[EDB Postgres for Kubernetes documentation](#) を参照してください

発行者と証明書の詳細については、[こちら](#)をご覧ください。

### 3.8 Declarative pausing and resuming

宣言的な一時停止と再開機能は、データベース PVC を維持しながらデータベースポッドを削除することにより、CPU パワーを節約できます。

宣言的な一時停止と再開は、Kubernetes の EDB Postgres で利用可能なハイバーネーション機能を活用します。ハイバーネーションの詳細と説明については、[Postgres for Kubernetes documentation on declarative hibernation](#) を参照してください。

目的の PGD グループに `k8s.pgd.enterprisedb.io/pause` アノテーションを追加して、一時停止を要求します。

例

```
kubectl annotate pgdgroup region-a k8s.pgd.enterprisedb.io/pause=on
```

数秒後、要求された PGD グループは一時停止状態になり、すべてのデータベースポッドが削除されます。

```
kubectl get pgdgroups
```

| NAME     | DATA INSTANCES | WITNESS INSTANCES | PHASE              | AGE |
|----------|----------------|-------------------|--------------------|-----|
| region-a | 2              | 1                 | PGDGroup - Paused  | 25m |
| region-b | 2              | 1                 | PGDGroup - Healthy | 25m |
| region-c | 0              | 1                 | PGDGroup - Healthy | 25m |

一時停止した PGD グループを再開するには、アノテーションを `off` に設定します。 `--overwrite` フラグを追加することを忘れないでください。

```
kubectl annotate pgdgroup region-a k8s.pgd.enterprisedb.io/pause=off --overwrite
```

数秒で、ノードが再開を開始し、ポッドが再作成されるのが表示されます。

```
kubectl get pgdggroups
```

| NAME     | DATA INSTANCES | WITNESS INSTANCES | PHASE                | AGE |
|----------|----------------|-------------------|----------------------|-----|
| region-a | 2              | 1                 | Pause - resume nodes | 1m  |
| region-b | 2              | 1                 | PGDGroup - Healthy   | 25m |
| region-c | 0              | 1                 | PGDGroup - Healthy   | 25m |

一時停止アノテーションが PGD グループを一時停止する前に、いくつかの要件があります。理想的には、PGD グループは正常正常状態である必要があります。または、PGD グループ内のすべてのデータノードが個々のレベルで正常な場合、一時停止を開始することもできます。

## 3.9 Installation images and repositories

Postgres Distributed またはバージョンを展開する予定がある場合は、適切なイメージとイメージバージョンを選択する必要があります。'Installation' の前

1. EDB's private image registries の説明に従って、リポジトリ名を特定し、ユーザートークンを取得します。
2. 'Identifying image names'。
3. 'Assembling a deployment command'。

### 3.9.1 EDB' s private image registries

EDB Postgres Distributed for Kubernetes と EDB Postgres for Kubernetes オペレーター、およびさまざまなオペランドのイメージは、`docker.enterprisedb.com` の下のプライベートコンテナイメージレジストリに保持されます。

---

**重要:** プライベートレジストリへのアクセスには、EDB のアカウントが必要で、有効な [subscription plan](#) を持つ EDB 顧客のために予約されています。資格情報は EDB アカウントを介して実行されます。トライアルについては、[トライアル](#) を参照してください。

---

どのリポジトリを選択しますか？

Kubernetes の EDB Postgres Distributed は、EDB Enterprise プランまたは EDB Standard プランに加えて、Extreme High Availability アドオンの一部として利用できます。

サブスクリプションプランによっては、Kubernetes の EDB Postgres Distributed は次のリポジトリのいずれかにあります。

| Plan                  | Repository                |
|-----------------------|---------------------------|
| EDB Standard Plan     | <i>k8s_standard_pgd</i>   |
| EDB EnterpriseDB Plan | <i>k8s_enterprise_pgd</i> |

`docker login` または [kubernetes.io/dockerconfigjson](https://kubernetes.io/dockerconfigjson) を介して EDB コンテナレジストリにログインするときに、リポジトリの名前をユーザー名として使用します。

**重要:** 各リポジトリには、プランでアクセスできるすべてのイメージが含まれています。演算子やオペランドイメージなどのさまざまなイメージにアクセスするために別のリポジトリに接続する必要はありません。

## トークンの取得方法

[repos page in EDB](#) には、リポトークンが隠れて表示される EDB Repos 2.0 セクションがあります。

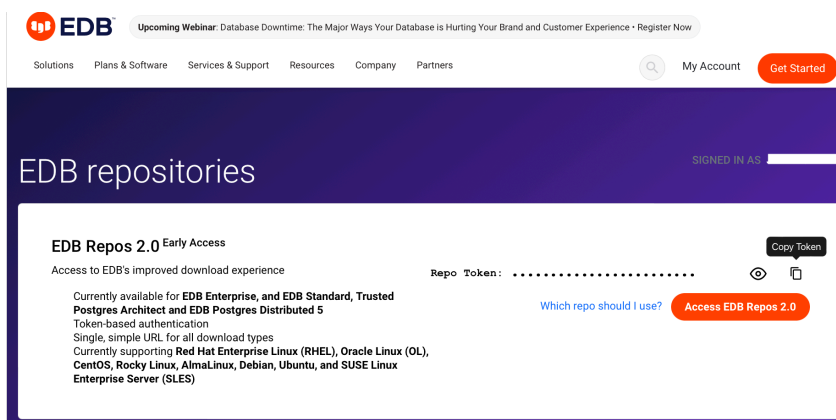


図 2: EDB Repo Portal

リポトークンの横には、トークンをコピーするための **Copy Token** ボタンと、トークンのコンテンツを確認するための目のアイコンがあります。

EDB コンテナレジストリにログインするときは、パスワードとしてリポトークンを使用します。

## `docker login` を使用した例

ターミナルから Docker を使用してログインできます。 **Copy Token** を使用してリポトークンをコピーすることをお勧めします。 `docker` コマンドは、ユーザー名とパスワードの入力を求めます。

ユーザー名はアクセスしようとしているリポジトリであり、パスワードは先ほどコピーしたトークンです。

```
$ docker login docker.enterprisedb.com
Username: k8s_enterprise_pgd
```

(次のページに続く)

(前のページからの続き)

```
Password:
Login Succeeded
```

## トライアル

トライアルリストまたはプレビューユーザーの場合、リポジトリの名前として `k8s_enterprise_pgd` を使用し、トークンの [トークンの取得方法](#) の指示に従ってください。

## オペランドイメージ

EDB Postgres Distributed for Kubernetes は、3 つの PostgreSQL ディストリビューションでの EDB Postgres DistributedPGD バージョン 5 の実行をサポートするオペレーターです。

- PostgreSQL
- EDB Postgres Advanced Server
- EDB Postgres 拡張

---

**重要:** [Choosing a Postgres distribution](#) を参照

---

詳細と、サポートされているさまざまな PostgreSQL ディストリビューションでの PGD の比較については、PGD ドキュメントを参照してください。

EDB オペレータにおける不変のアプリケーションコンテナの採用のため、オペレータはコンテナイメージに、Postgres の必要なディストリビューションとバージョンに加えて PGD の要求されたバージョンを実行するために必要なすべてのバイナリを含めることを想定しています。

これらのイメージは、[Container image requirements](#) で説明されている要件と規則に従います。

EDB Postgres for Kubernetes ドキュメントに、`bdr5` 拡張機能を追加します。

この表には、各 Postgres ディストリビューションのイメージ名前の接頭辞が示されています。

| Postgres distribution | Versions | Image name                             | Repositories                                                    |
|-----------------------|----------|----------------------------------------|-----------------------------------------------------------------|
| EDB Postgres Extended | 15, 14   | <code>edb-postgres-extended-pgd</code> | <code>k8s_standard_pgd</code> , <code>k8s_enterprise_pgd</code> |
| EDB Postgres Advanced | 15, 14   | <code>edb-postgres-advanced-pgd</code> | <code>k8s_enterprise_pgd</code>                                 |

### 3.9.2 Identifying image names

製品をテストすることを目的とした簡単なインストールについては、'[Quick start](#)' を参照してください。よりターゲットを絞ったテストまたは実稼働を目的として、Postgres ディストリビューションに適切な特定のオペラントとプロキシイメージバージョンを選択できます。

#### オペラントイメージ名

イメージに関する一般的な情報については、[オペラントイメージ](#) を参照してください。

#### 名前付け要素

オペラントイメージ名は、Postgres タイプ、EDB Postgres Distributed、それらのバージョン番号、およびビルドバージョンで構成されます。

**Postgres タイプ** イメージ名の最初の部分は、使用している Postgres タイプです。この表は、Postgres タイプがイメージ名のこの部分にどのように対応するかを示しています。

| Postgres type                | Image name            |
|------------------------------|-----------------------|
| EDB Postgres Advanced Server | edb-postgres-advanced |
| EDB Postgres Extended Server | edb-postgres-extended |
| PostgreSQL                   | postgresql            |

**EDB Postgres Distributed:** すべての環境構成で、pgd はイメージ名の次の部分です。

**バージョン番号** Postgres タイプと PGD のバージョンは、ダッシュで区切られ、イメージ名の次にあります。

**ビルドバージョン** 値は、現在のビルドを反映します。例、1. EDB は、セキュリティパッチを含む新しいビルドを作成します。

#### オペラントイメージ名のアセンブル

これらの識別子は、そのバージョンとともに、オペラントイメージ名を形成します。表にいくつかの例を示します。

| Postgres version           | EDB Postgres Distributed version | Build | Operand image name                       |
|----------------------------|----------------------------------|-------|------------------------------------------|
| EDB Postgres Advanced 16.2 | PGD 5.4.1                        | 1     | edb-postgres-advanced-pgd:16.2.0-5.4.1-1 |

次のページに続く

表 5 – 前のページからの続き

| Postgres version           | EDB Postgres Distributed version | Build | Operand image name                     |
|----------------------------|----------------------------------|-------|----------------------------------------|
| EDB Postgres Extended 15.6 | PGD 5.4.0                        | 1     | edb-postgres-extended-pgd:15.6-5.4.0-1 |
| PostgreSQL 16.2            | PGD 5.4.1                        | 1     | postgresql-pgd:16.2-5.4.1-1            |

Postgres の互換性の概要については、[Platform compatibility](#) を参照してください。

## プロキシイメージ名

プロキシイメージの名前とバージョンは、EDB Postgres Distributed から取得されます。PGD バージョンおよび対応する PGD プロキシバージョンの概要については、[EDB Postgres Distributed release notes](#) を参照してください。表に一例を示します。

| Proxy version   | Proxy image name    |
|-----------------|---------------------|
| PGD Proxy 5.4.0 | edb-pgd-proxy:5.4.0 |

### 3.9.3 Assembling a deployment command

製品をテストすることを目的とした簡単なインストールについては、['Quick start'](#) を参照してください。

より対象を絞ったテストまたは運用を目的として、選択したオペラントとプロキシイメージバージョンを使用して Kubernetes 用 EDB Postgres 分散を展開するコマンドをアセンブルできます。

## 前提条件

次の情報を収集します。

- Repository name
- トークンの取得方法
- オペラントイメージ名
- プロキシイメージ名

## コマンドのアセンブル

<...> のプレースホルダーを、収集した情報に置き換えます。

```
helm upgrade --dependency-update \
  --install edb-pg4k-pgd \
  --namespace pgd-operator-system \
  --create-namespace \
  edb/edb-postgres-distributed-for-kubernetes \
  --set edb-postgres-for-kubernetes.enabled=true \
  --set image.repository=docker.enterprisedb.com/<REPOSITORY_NAME>/pg4k-pgd \
  --set edb-postgres-for-kubernetes.image.repository=docker.enterprisedb.com/
  ↪<REPOSITORY_NAME>/edb-postgres-for-kubernetes \
  --set config.data.PGD_IMAGE_NAME=docker.enterprisedb.com/<REPOSITORY_NAME>/<OPERAND_
  ↪NAME> \
  --set config.data.PGD_PROXY_IMAGE_NAME=docker.enterprisedb.com/<REPOSITORY_NAME>/
  ↪<PGD-PROXY>
  --set image.imageCredentials.username=<REPOSITORY_NAME> \
  --set image.imageCredentials.password=<EDB_SUBSCRIPTION_TOKEN>
```

必要なイメージでコマンドをアセンブルした後、リポジトリの追加、イメージの展開、および証明書発行者の作成方法については、'[Installation](#)' を参照してください。

## 例

これらのコマンド例

- `k8s_enterprise_pgd` リポジトリからイメージをプルします。
- Postgres オプションとして EDB Postgres Advanced Server 15.6.2 を使用します。
- Postgres Distributed バージョンとして PGD 5.4.1 を使用します。
- PGD プロキシバージョンとして 5.4.0 を使用します。

### 例 1 展開コマンドを実行する前に環境変数を設定する

展開コマンドの環境変数を設定します。

1. `k8s_enterprise_pgd` リポジトリからイメージをプルするように環境変数を設定します。

```
export REPOSITORY_NAME=k8s_enterprise_pgd
```

1. [トークンの取得方法](#) を使用するように環境変数を設定します。

```
export EDB_SUBSCRIPTION_TOKEN=<my_registry_token>
```

1. Postgres オプションとして EDB Postgres Advanced Server 15.6.2 を使用し、Postgres Distributed バージョンとして PGD 5.4.1 を使用するように環境変数を設定します。

```
export OPERAND_NAME=edb-postgres-advanced-pgd:15.6.2-5.4.1-1
```

1. PGD プロキシバージョンとして 5.4.0 を使用するように環境変数を設定します。

```
export PGD-PROXY=edb-pgd-proxy:5.4.0
```

1. 展開コマンドを実行します。

```
helm upgrade --dependency-update \
  --install edb-pg4k-pgd \
  --namespace pgd-operator-system \
  --create-namespace \
  edb/edb-postgres-distributed-for-kubernetes \
  --set edb-postgres-for-kubernetes.enabled=true \
  --set image.repository=docker.enterprisedb.com/${REPOSITORY_NAME}/pg4k-pgd \
  --set edb-postgres-for-kubernetes.image.repository=docker.enterprisedb.com/$
  ↪{REPOSITORY_NAME}/edb-postgres-for-kubernetes \
  --set config.data.PGD_IMAGE_NAME=docker.enterprisedb.com/${REPOSITORY_NAME}/$
  ↪{OPERAND_NAME} \
  --set config.data.PGD_PROXY_IMAGE_NAME=docker.enterprisedb.com/${REPOSITORY_NAME}/$
  ↪{PGD-PROXY}
  --set image.imageCredentials.username=${REPOSITORY_NAME} \
  --set image.imageCredentials.password=${EDB_SUBSCRIPTION_TOKEN}
```

## 例 2 プレースホルダーを手動で置き換える

1. トークンの取得方法 を使用するように環境変数を設定します。

```
export EDB_SUBSCRIPTION_TOKEN=<my_registry_token>
```

1. リポジトリ、イメージ、およびプロキシ名をコマンドに挿入します。

```
helm upgrade --dependency-update \
  --install edb-pg4k-pgd \
  --namespace pgd-operator-system \
  --create-namespace \
  edb/edb-postgres-distributed-for-kubernetes \
  --set edb-postgres-for-kubernetes.enabled=true \
  --set image.repository=docker.enterprisedb.com/k8s_enterprise_pgd/pg4k-pgd \
  --set edb-postgres-for-kubernetes.image.repository=docker.enterprisedb.com/k8s_
  ↪enterprise_pgd/edb-postgres-for-kubernetes \
  --set config.data.PGD_IMAGE_NAME=docker.enterprisedb.com/k8s_enterprise_pgd/edb-
  ↪postgres-advanced-pgd:15.6.2-5.4.1-1 \
  --set config.data.PGD_PROXY_IMAGE_NAME=docker.enterprisedb.com/k8s_enterprise_pgd/
  ↪edb-pgd-proxy:5.4.0
  --set image.imageCredentials.username=k8s_enterprise_pgd \
  --set image.imageCredentials.password=${EDB_SUBSCRIPTION_TOKEN}
```

## 3.10 Predefined labels

これらの事前定義されたラベルは、EDB Postgres Distributed for Kubernetes オペレーターによって管理されます。

`k8s.pgd.enterprisedb.io/certificateType` 証明書の種類を示します。`replication` は、レプリケーションクライアントの認証に使用される証明書を示します。`server` は、サーバー認証に使用される証明書を示します。

`k8s.pgd.enterprisedb.io/group` リソースが属する PGD グループの名前。クラスターまたはインスタンスリソースに追加されます。

`k8s.pgd.enterprisedb.io/isWitnessService` サービスが監視ノード用であることを示します。

`k8s.pgd.enterprisedb.io/type` クラスターまたはインスタンスリソースに追加されるリソースのタイプ、通常は `node`。

`k8s.pgd.enterprisedb.io/workloadType` クラスターまたはインスタンスリソースに追加されたリソースのワークロードタイプを示します。`pgd-node-data` はデータノードを示します。`pgd-node-witness` 監視ノード。PGD プロキシノードの場合は `pgd-proxy`。`proxy-svc` PGD プロキシサービスの場合。`group-svc` PGD グループサービスが PGDGroup 内のノードと通信する場合。`node-svc` は、CNP サービステンプレートから作成されたサービスです。`scheduled-backup` は `scheduledBackup` リソースに追加されます。`bootstrap-cross-location-pgd-group` は、クロスロケーション PGD グループを作成するポッドに追加されます。`pgd-node-restore` は、ノードの復元プロセスを開始するポッドに追加されます。

### 3.10.1 事前定義されたアノテーション

`k8s.pgd.enterprisedb.io/dirtyMetadata` バックアップから生成され、PGD ノードを作成する前にメタデータをクリーンアップする必要がある CNP クラスターに設定します。これは、復元ジョブによって書き込まれます。

`k8s.pgd.enterprisedb.io/hashCluster`、`ScheduledBackup`、`StatefulSet`、`Service` (`node`、`group` and `proxy service`) などのさまざまなエンティティで利用される PGDGroup 仕様の特定の部分のハッシュを保持して、対応するリソースに更新が必要かどうかを判断します。

`k8s.pgd.enterprisedb.io/latestCleanupExecuted` PGDGroup で設定され、クリーンアップが実行されたことを示します。

`k8s.pgd.enterprisedb.io/node` 特定の証明書が生成されたノードの名前が含まれています。証明書リソースに追加されました。

`k8s.pgd.enterprisedb.io/nodeRestartHash` CNP 構成のハッシュを PGDGroup に保存します。構成を変更する場合は再起動が必要です。

`k8s.pgd.enterprisedb.io/noFinalizers` ファイナライザーの実行をスキップするように値 `true` を使用して PGDGroup で設定します。内部使用のみ。

`k8s.pgd.enterprisedb.io/pause` PGDGroup を一時停止するように PGDGroup に設定します。

`k8s.pgd.enterprisedb.io/recoverabilityPointsByMethod` メソッドごとに CNP クラスターの最初のリカバリアビリティポイントを改ざん防止の場所に保存するように PGDGroup で設定します。

k8s.pgd.enterprisedb.io/seedingServer PGDGroup で設定して、どのサーバーを復元するかをオペレーターに示します。これは、復元ジョブによって書き込まれます。

k8s.pgd.enterprisedb.io/seedingSnapshots PGDGroup で設定して、どのスナップショットを復元するかをオペレーターに示します。これは、復元ジョブによって書き込まれます。

## 3.11 Managed Configuration

PGD オペレーターを使用すると、PG4K クラスターの managed セクションを構成できます。spec.cnp.managed スタンザは、クラスター内でサポートされている管理ロールを構成するために使用されます。

この例では、postgres で設定された指定されたプロパティを使用して foo という名前の管理ロールを持つように pgdgroup を構成します。

```
apiVersion: pgd.k8s.enterprisedb.io/v1beta1
kind: PGDGroup
metadata:
  name: group-example-with-managed
spec:
  [...]
  cnp:
    [...]
    managed:
      roles:
        - name: foo
          comment: Foo
          ensure: present
          connectionLimit: 100
          login: true
          superuser: true
          createdb: true
          createrole: true
          replication: true
```

管理対象ロールの詳細については、[EDB Postgres for Kubernetes recovery - Database Role Management](#) を参照してください。

## 3.12 PGDGroup parting

### 3.12.1 削除とファイナライザー

PGD グループを削除する場合、オペレーターは最初にグループ内のすべてのノードの分割を開始します。アクティブなインスタンスに接続し、ターゲットグループ内のすべてのノードを分割します。ノードが分離されると、レプリケーションとコンセンサス操作に参加しなくなります。ノードが削除される前に正し

く分割されることを確認するために、オペレーターは `k8s.pgdb.enterprisedb.io/partNodes` ファイナライザーを使用します。状況については、[kubernetes document on finalizers](#) を参照してください。

**注釈:** PGD グループを保持する名前空間が直接削除される場合、削除と分割シーケンスが正しく実行されることを保証できません。名前空間を削除する前に、含まれるすべての PGD グループを削除することをお勧めします。

### 3.12.2 制限時間

ノードを分離するとき、オペレーターはアクティブなインスタンスに接続して `bdr.part_node` ファンクションを実行する必要があります。この操作のハングを回避するために、ファイナライザーの時間制限が使用されます。デフォルトでは 300 秒です。時間制限が経過すると、ファイナライザーが削除され、いずれにせよノードが削除されます。グローバル PGD カタログに古いメタデータが残る可能性があります。この時間制限は、秒単位で指定される `spec.failingFinalizerTimeLimitSeconds` を介して構成できます。

### 3.12.3 ファイナライザーをスキップ

テスト目的のみで、オペレーターはファイナライザー `k8s.pgdb.enterprisedb.io/noFinalizers` をスキップするためのアノテーションも提供します。このアノテーションが PGD グループに追加されると、PGD グループが削除されるときにファイナライザーがスキップされ、ノードは PGD クラスターから分離されません。

### 3.12.4 PGD グループのクリーンアップ

#### 分割されたノードのクリーンアップ

PGDGroup が削除されると、そのメタデータは、`bdr.node_summary` テーブルの `PARTED` 状態でカタログに残ります。PGD4K オペレーターは、`PGDGroupCleanup` という名前の CRD を定義して、`PARTED PGDGroup` のクリーンアップに役立ちます。

以下の例では、`PGDGroupCleanup` は `region-a` からローカルに実行され、すべてのノードが `PARTED` 状態である必要があるという前提条件で、すべてのリージョン `b` をクリーンアップします。

```
apiVersion: pgd.k8s.enterprisedb.io/v1beta1
kind: PGDGroupCleanup
metadata:
  name: region-b-cleanup
spec:
  executor: region-a
  target: region-b
```

ターゲットグループこの例では region-b に PARTED 状態にないノードが含まれている場合、グループクリーンアップは PGDGroupCleanup - Target PGDGroup is not parted, waiting for it to be parted before executing PGDGroupCleanup フェーズで停止することに注意してください。非常に必要な場合は、force オプションを追加できます。

**警告:** *force* を使用すると、PGD クラスターが一貫性のない状態のままになる場合があります。他の方法でグループノードを分離できない障害から回復するためにのみ使用します。

```
apiVersion: pgd.k8s.enterprisedb.io/v1beta1
kind: PGDGroupCleanup
metadata:
  name: region-b-cleanup
spec:
  force: true
  executor: region-a
  target: region-b
```

### 3.13 SQLMutations

SQLMutation は、pgd ノードが pgd グループに参加した後、スーパーユーザーロールを介してアプリケーションデータベースで実行される SQL クエリのリストで構成されています。各 SQLMutation には、isApplied クエリーの isApplied リストと exec クエリーの exec リストが含まれます。isApplied SQL クエリーは、変更が既に適用されているかどうかを確認するために使用されます。isApplied クエリーのいずれかが false を返した場合、SQL クエリーの exec リストが実行キューに追加されます。

これは SQL ミューテーションのサンプルです

```
apiVersion: pgd.k8s.enterprisedb.io/v1beta1
kind: PGDGroup
[...]
spec:
  pgd:
    mutations:
      - isApplied:
          - |
            WITH node_route_config AS (
              SELECT node_name, route_dsn
              FROM bdr.node_routing_config_summary
            )
            SELECT NOT EXISTS(SELECT * FROM node_route_config WHERE regexp_
            ↪count(route_dsn, application_name=) >= 1);
      exec:
        - |
```

(次のページに続く)

(前のページからの続き)

```

WITH node_route_config AS (
    SELECT node_name,
           regexp_replace(route_dsn, concat(application_name=, node_name),
    ) as route_dsn
    FROM bdr.node_routing_config_summary
)
SELECT bdr.alter_node_option(node_name, route_dsn, route_dsn)
FROM node_route_config;

```

### 3.13.1 SQL ミューテーションタイプ

演算子は、`spec.pgd.mutations[].type` で指定できる 3 種類の `SQLMutation` を提供し、`always` がデフォルトオプションです。

- `beforeSubgroupRaft`
- `いつも`
- `writeLeader`

`beforeSubgroupRaft` および `always` 変異は、すべての `reconcile` ループで評価されます。2つのミューテーションの違いは、実行フェーズにあります。

- `always` 変異の場合、`pgdgroup` に制限なしで各 `reconcile` ループで実行されます。
- 一方、`beforeSubgroupRaft` 変異は、`pgd` グループにデータノードと `pgd` プロキシが定義されている場合、特にサブグループ `raft` の準備が整う前にものみ実行されます。

`beforeSubgroupRaft` および `always` 変異は両方とも、監視ノードを含む `pgd` グループ内の任意の `pgd` ノードで実行できます。したがって、監視ノードにはアプリケーションデータベースデータが含まれないため、アプリケーションデータベースのデータを変更するために使用しないでください。

`writeLeader` ミューテーションは、書き込みリーダーが選択された後にトリガーおよび実行されます。`exec` 操作は、書き込みリーダーノードで排他的に実行されます。

## 3.14 Red Hat OpenShift

EDB Postgres Distributed for Kubernetes は、Web インターフェイスを使用して OpenShift にインストールできる認定オペレーターです。

### 3.14.1 EDB プライベートレジストリへのアクセスの保証

**重要:** オペレーターイメージとオペランドイメージの両方が保存されているプライベート EDB リポジトリへのアクセスが必要です。アクセスには、有効な [EDB subscription plan](#) が必要です。詳細は、EDB's private image registries を参照してください。

OpenShift インストールは、プルシークレットを使用して、プライベートリポジトリに保持されているオペランドおよびオペレーターイメージにアクセスします。

プライベートリポジトリへの資格情報を取得したら、`openshift-operators` 名前空間に 2 つのプルシークレットを作成する必要があります。

- EDB Postgres Distributed for Kubernetes オペレーターイメージの `pgd-operator-pull-secret`
- `postgresql-operator-pull-secret` Kubernetes オペレーターイメージの EDB Postgres の場合

`oc create` コマンドを使用して、各シークレットを作成できます。

```
oc create secret docker-registry pgd-operator-pull-secret \
  -n openshift-operators --docker-server=docker.enterprisedb.com \
  --docker-username="@@REPOSITORY@" \
  --docker-password="@@TOKEN@"

oc create secret docker-registry postgresql-operator-pull-secret \
  -n openshift-operators --docker-server=docker.enterprisedb.com \
  --docker-username="@@REPOSITORY@" \
  --docker-password="@@TOKEN@"
```

そこで

- `@@REPOSITORY@` は、Which repository to choose? で説明したように、リポジトリの名前です。
- `@@TOKEN@` は、[トークンの取得方法](#) で説明したように、EDB アカウントのリポジトリトークンです。

### 3.14.2 オペレーターのインストール

EDB Postgres Distributed for Kubernetes オペレーターは、OpenShift ダッシュボードから Red Hat OperatorHub で直接見つけることができます。

1. ハンバーガーメニューから、**Operators > OperatorHub** を選択します。
2. Web コンソールで、検索ボックスを使用してリストをフィルタリングします。たとえば、EDB または `pgd` と入力します。
3. オペレーターに関する情報を読み、**Install** を選択します。
4. Operator Installation ページで、次を選択します。
  - インストールモード。 [クラスター全体のインストール](#) は、現在唯一のモードです。

## OperatorHub

Discover Operators from the Kubernetes community and Red Hat partners, curated by installation, the Operator capabilities will appear in the [Developer Catalog](#) providing a

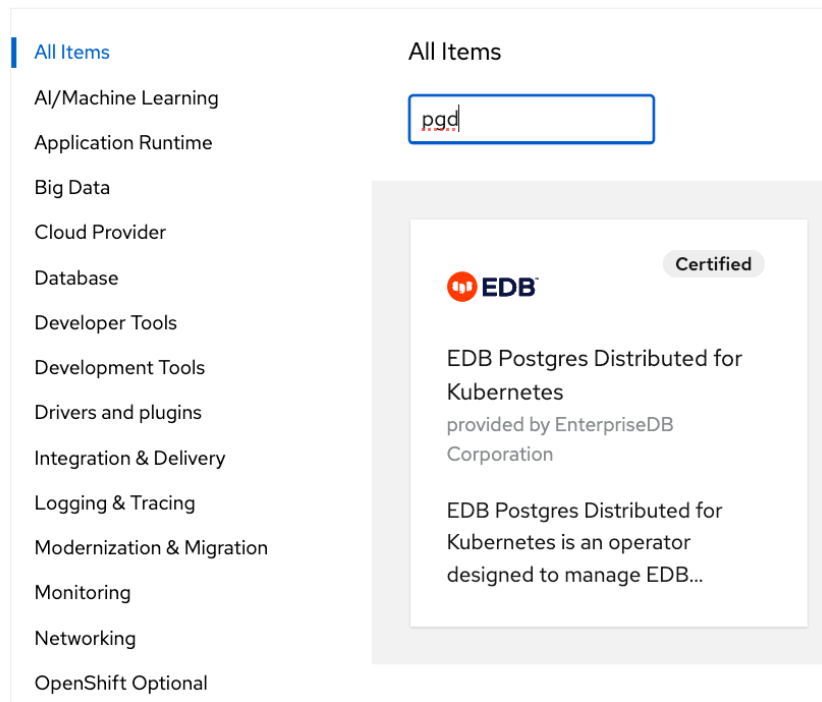


図 3: Install OperatorHub

- 更新チャンネル現在 **プレビュー**
- Red Hat によって認定された、オペレーターの新しいリリースのマーケットプレイスでの入手可能性後の承認戦略
- **自動**: OLM は、実行中のオペレーターを新しいバージョンにアップグレードします。
- **マニュアル** OpenShift は、**Installed Operators** セクションでの承認を必要とすることにより、人間の介入を待機します。

## クラスター全体のインストール

クラスター全体のインストールでは、オペレーターをデフォルトの `openshift-operators` 名前空間にインストールし、クラスター内のすべてのプロジェクトで使用できるように OpenShift に指示します。これは、Kubernetes 用の EDB Postgres Distributed をインストールするためのデフォルトであり、通常推奨されるアプローチです。

Web コンソールから、**インストールモード** で、**クラスター上のすべての名前空間(デフォルト)** を選択します。

インストールすると、オペレーターはすべての名前空間で可視されます。インストール中に問題が発生した場合は、他の OpenShift オペレーターの場合と同様に、**Workloads > Pods** ページの `openshift-operators` プロジェクトのポッドのログを確認します。

### 3.14.3 PGD クラスターの作成

OpenShift によるインストール後、オペレーターのデプロイメントは `openshift-operators` 名前空間にあります。EDB Postgres for Kubernetes オペレーター `postgresql-operator-controller-manager` と同様に、`cert-manager` オペレーターもインストールされたことに注意してください。

```
$ oc get deployments -n openshift-operators
```

| NAME                                                       | READY | UP-TO-DATE | AVAILABLE | AGE |
|------------------------------------------------------------|-------|------------|-----------|-----|
| <code>cert-manager-operator</code>                         | 1/1   | 1          | 1         | 11m |
| <code>pgd-operator-controller-manager</code>               | 1/1   | 1          | 1         | 11m |
| <code>postgresql-operator-controller-manager-1-20-0</code> | 1/1   | 1          | 1         | 23h |
| ...                                                        |       |            |           |     |

`pgd-operator-controller-manager` 展開の準備ができていることを確認した後、PGD クラスターの作成を開始できます。EDB Postgres Distributed for Kubernetes リポジトリには、いくつかの役立つサンプルファイルが含まれています。

PGD クラスターは、専用の名前空間/プロジェクトに展開する必要があります。デフォルトの名前空間は予約されています。

最初に、新しい名前空間を作成し、`self-signed certificate `Issuer <Examples of configuration>`` をデプロイします

その中で

```
oc create ns my-namespace
oc apply -n my-namespace -f \
  https://raw.githubusercontent.com/EnterpriseDB/edb-postgres-for-kubernetes-charts/
  ↪main/hack/samples/issuer-selfsigned.yaml
```

### 単一リージョンの単一の OpenShift クラスタで PGD を使用する

これで、2つのデータグループと1つの監視グループを含むフレキシブル3リージョンなどのPGDクラスターを展開できるようになりました。ファイルにYAMLマニフェストがあります。

```
apiVersion: pgd.k8s.enterprisedb.io/v1beta1
kind: PGDGroup
metadata:
  name: region-a
spec:
  instances: 2
  proxyInstances: 2
  witnessInstances: 1
  pgd:
    parentGroup:
      name: world
      create: true
    discovery:
      - host: region-a-group
      - host: region-b-group
      - host: region-c-group
  cnp:
    storage:
      size: 1Gi
  connectivity:
    dns:
      additional:
        - domain: alternate.domain
        - domain: my.domain
      hostSuffix: -dc1
  tls:
    mode: verify-ca
    clientCert:
      caCertSecret: client-ca-key-pair
      certManager:
        spec:
          issuerRef:
            name: client-ca-issuer
            kind: Issuer
```

(次のページに続く)

(前のページからの続き)

```

        group: cert-manager.io
serverCert:
  caCertSecret: server-ca-key-pair
  certManager:
    spec:
      issuerRef:
        name: server-ca-issuer
        kind: Issuer
        group: cert-manager.io
-- --
apiVersion: pgd.k8s.enterprisedb.io/v1beta1
kind: PGDGroup
metadata:
  name: region-b
spec:
  instances: 2
  proxyInstances: 2
  witnessInstances: 1
  pgd:
    parentGroup:
      name: world
    discovery:
      - host: region-a-group
      - host: region-b-group
      - host: region-c-group
  cnp:
    storage:
      size: 1Gi
  connectivity:
    tls:
      mode: verify-ca
      clientCert:
        caCertSecret: client-ca-key-pair
        certManager:
          spec:
            issuerRef:
              name: client-ca-issuer
              kind: Issuer
              group: cert-manager.io
      serverCert:
        caCertSecret: server-ca-key-pair
        certManager:
          spec:
            issuerRef:
              name: server-ca-issuer

```

(次のページに続く)

(前のページからの続き)

```

        kind: Issuer
        group: cert-manager.io
-- --
apiVersion: pgd.k8s.enterprisedb.io/v1beta1
kind: PGDGroup
metadata:
  name: region-c
spec:
  instances: 0
  proxyInstances: 0
  witnessInstances: 1
  pgd:
    parentGroup:
      name: world
    discovery:
      - host: region-a-group
      - host: region-b-group
      - host: region-c-group
  cnp:
    storage:
      size: 1Gi
  connectivity:
    tls:
      mode: verify-ca
      clientCert:
        caCertSecret: client-ca-key-pair
        certManager:
          spec:
            issuerRef:
              name: client-ca-issuer
              kind: Issuer
              group: cert-manager.io
      serverCert:
        caCertSecret: server-ca-key-pair
        certManager:
          spec:
            issuerRef:
              name: server-ca-issuer
              kind: Issuer
              group: cert-manager.io

```

```
oc apply -f flexible_3regions.yaml -n my-namespace
```

PGD グループが表示され始めます。

```
$ oc get pgdgroups -n my-namespace
```

| NAME     | DATA INSTANCES | WITNESS INSTANCES | PHASE              | PHASE DETAILS |
|----------|----------------|-------------------|--------------------|---------------|
| →AGE     |                |                   |                    |               |
| region-a | 2              | 1                 | PGDGroup - Healthy |               |
| →23m     |                |                   |                    |               |
| region-b | 2              | 1                 | PGDGroup - Healthy |               |
| →23m     |                |                   |                    |               |
| region-c | 0              | 1                 | PGDGroup - Healthy |               |
| →23m     |                |                   |                    |               |

### 複数のリージョンの複数の OpenShift クラスターで PGD を使用する

複数のリージョンの複数の OpenShift クラスターに PGD をデプロイするには、最初に PGD グループが相互に通信する方法を確立する必要があります。複数の OpenShift クラスターでこれを実現する推奨方法は、[Submariner](#) を使用することです。接続の構成は、このドキュメントの範囲外です。ただし、OpenShift クラスター間の接続を確立すると、互いに同期した PGD グループを展開できます。

**警告:** この例では、各 OpenShift クラスターに 1 つずつ、3 つの PGD グループを展開し、Submariner を使用して OpenShift クラスター間の接続を確立していることを前提としています。

**単一リージョンの単一の OpenShift クラスターで PGD を使用する**と同様に、この例では、2 つのデータ PGD グループと 1 つの監視グループを作成します。その例とは対照的に、各グループは別の OpenShift クラスターに存在します。

OpenShift クラスター間の基本的な接続に加えて、各 OpenShift クラスターに、他の OpenShift クラスターによって信頼される認証局が含まれていることを確認する必要があります。この条件は、PGD グループが相互に通信するために必要です。

OpenShift クラスターはすべて同じ認証局を使用するか、各クラスターが独自の認証局を持つことができます。いずれの場合も、各 OpenShift クラスターの証明書が他の OpenShift クラスターの認証局を信頼していることを確認する必要があります。

この例では、すべての OpenShift クラスターのすべての証明書に使用される単一の認証局を持つ自己署名証明書を使用します。

この例では、OpenShift クラスターを OpenShift Cluster A、OpenShift Cluster B、および OpenShift Cluster C として参照します。OpenShift では、OperatorHub からの EDB Postgres Distributed for Kubernetes オペレーターのインストールに、cert-manager オペレーターのインストールが含まれます。cert-manager で証明書を作成および管理することをお勧めします。

1. OpenShift Cluster A を保持する名前空間を作成し、その中に自己署名証明書に必要なオブジェクトも作成します。PGD オペレーターと cert-manager がインストールされていると仮定して、self-signed certificate `Issuer <Examples of configuration>`

その名前空間で。

```
oc create ns pgd-group
oc apply -n pgd-group -f \
  https://raw.githubusercontent.com/EnterpriseDB/edb-postgres-for-kubernetes-charts/
  ↪main/hack/samples/issuer-selfsigned.yaml
```

1. しばらくして、cert-manager は発行者と証明書を作成します。pgd-group 名前空間には、server-ca-key-pair および client-ca-key-pair の2つのシークレットがあります。これらのシークレットには、サーバーおよびクライアント認証局の証明書とプライベートキーが含まれています。issuer-selfsigned.yaml マニフェストを適用する前に、これらのシークレットを他の OpenShift クラスターにコピーする必要があります。oc get secret コマンドを使用して、シークレットの内容を取得できます。

```
oc get secret server-ca-key-pair -n pgd-group -o yaml > server-ca-key-pair.yaml
oc get secret client-ca-key-pair -n pgd-group -o yaml > client-ca-key-pair.yaml
```

1. これらのシークレット uid、リソースバージョン、タイムスタンプなどから OpenShift Cluster A に固有のコンテンツを削除した後、コンテキストを OpenShift Cluster B に切り替えることができます。次に、名前空間を作成し、その中にシークレットを作成して、のみに issuer-selfsigned.yaml ファイルを適用します。

```
oc create ns pgd-group
oc apply -n pgd-group -f server-ca-key-pair.yaml
oc apply -n pgd-group -f client-ca-key-pair.yaml
oc apply -n pgd-group -f \
  https://raw.githubusercontent.com/EnterpriseDB/edb-postgres-for-kubernetes-charts/
  ↪main/hack/samples/issuer-selfsigned.yaml
```

1. コンテキストを OpenShift Cluster C に切り替え、クラスター B で従った同じプロセスを繰り返すことができます。

```
oc create ns pgd-group
oc apply -n pgd-group -f server-ca-key-pair.yaml
oc apply -n pgd-group -f client-ca-key-pair.yaml
oc apply -n pgd-group -f \
  https://raw.githubusercontent.com/EnterpriseDB/edb-postgres-for-kubernetes-charts/
  ↪main/hack/samples/issuer-selfsigned.yaml
```

1. OpenShift Cluster A で、region-a と呼ばれる最初の PGD グループを作成できます。PGD グループの YAML マニフェストは次のとおりです。

```
apiVersion: pgd.k8s.enterprisedb.io/v1beta1
kind: PGDGroup
metadata:
  name: region-a
spec:
  instances: 2
```

(次のページに続く)

```

proxyInstances: 2
witnessInstances: 1
pgd:
  parentGroup:
    name: world
    create: true
  discovery:
    - host: region-a-group.pgd-group.svc.clusterset.local
    - host: region-b-group.pgd-group.svc.clusterset.local
    - host: region-c-group.pgd-group.svc.clusterset.local
  cnp:
    storage:
      size: 1Gi
  connectivity:
    dns:
      domain: "pgd-group.svc.clusterset.local"
      additional:
        - domain: alternate.domain
        - domain: my.domain
      hostSuffix: -dc1
    tls:
      mode: verify-ca
      clientCert:
        caCertSecret: client-ca-key-pair
        certManager:
          spec:
            issuerRef:
              name: client-ca-issuer
              kind: Issuer
              group: cert-manager.io
      serverCert:
        caCertSecret: server-ca-key-pair
        certManager:
          spec:
            issuerRef:
              name: server-ca-issuer
              kind: Issuer
              group: cert-manager.io

```

**重要:** *discovery* セクションのホスト名の形式は、単一クラスターの例とは異なります。これは、Submariner が OpenShift クラスターの接続に使用されており、Submariner が `region-a-group は、*region-a* という名前の PGD グループ用に作成されるサービスの名前です。

1. region-a PGD グループ YAML を適用します。

```
oc apply -f region-a.yaml -n pgd-group
```

1. コンテキストを OpenShift Cluster B に切り替え、2 番目の PGD グループを作成できます。クラスター B の PGD グループの YAML は次のとおりです。唯一の違いは `metadata.name` です。

```
apiVersion: pgd.k8s.enterprisedb.io/v1beta1
kind: PGDGroup
metadata:
  name: region-b
spec:
  instances: 2
  proxyInstances: 2
  witnessInstances: 1
  pgd:
    parentGroup:
      name: world
    discovery:
      - host: region-a-group.pgd-group.svc.clusterset.local
      - host: region-b-group.pgd-group.svc.clusterset.local
      - host: region-c-group.pgd-group.svc.clusterset.local
  cnp:
    storage:
      size: 1Gi
  connectivity:
    dns:
      domain: "pgd-group.svc.clusterset.local"
    tls:
      mode: verify-ca
      clientCert:
        caCertSecret: client-ca-key-pair
        certManager:
          spec:
            issuerRef:
              name: client-ca-issuer
              kind: Issuer
              group: cert-manager.io
      serverCert:
        caCertSecret: server-ca-key-pair
        certManager:
          spec:
            issuerRef:
              name: server-ca-issuer
              kind: Issuer
              group: cert-manager.io
```

1. region-b PGD グループ YAML を適用します。

```
oc apply -f region-b.yaml -n pgd-group
```

1. コンテキストを OpenShift Cluster C に切り替え、3 番目の PGD グループを作成できます。PGD グループの YAML は次のとおりです。

```
apiVersion: pgd.k8s.enterprisedb.io/v1beta1
kind: PGDGroup
metadata:
  name: region-c
spec:
  instances: 0
  proxyInstances: 0
  witnessInstances: 1
  pgd:
    parentGroup:
      name: world
    discovery:
      - host: region-a-group.pgd-group.svc.clusterset.local
      - host: region-b-group.pgd-group.svc.clusterset.local
      - host: region-c-group.pgd-group.svc.clusterset.local
  cnp:
    storage:
      size: 1Gi
  connectivity:
    dns:
      domain: "pgd-group.svc.clusterset.local"
    tls:
      mode: verify-ca
      clientCert:
        caCertSecret: client-ca-key-pair
        certManager:
          spec:
            issuerRef:
              name: client-ca-issuer
              kind: Issuer
              group: cert-manager.io
      serverCert:
        caCertSecret: server-ca-key-pair
        certManager:
          spec:
            issuerRef:
              name: server-ca-issuer
              kind: Issuer
              group: cert-manager.io
```

1. region-c PGD グループ YAML を適用します。

```
oc apply -f region-c.yaml -n pgd-group
```

これで、コンテキストを OpenShift Cluster A に戻し、そこで PGD グループのステータスを確認できます。

```
oc get pgdgroup region-a -n pgd-group
```

PGD グループはフェーズ PGD - Waiting for node discovery にあります。

各 OpenShift クラスターで PGD グループを作成し、各ノードのサービスを作成した後、サービスを他の OpenShift クラスターに公開する必要があります。これはさまざまな方法で行うことができます。

サブマリーナーを使用している場合、

`subctl` を使用して行うことができます

コマンド。 `-group` または `-node` サフィックスを持つ `pgd-group` 名前空間のサービスごとに `subctl export service` コマンドを実行します。これは、各クラスターで次の `bash for` ループを実行することにより実行できます。

```
for service in $(oc get svc -n pgd-group --no-headers -o custom-columns="NAME:.
→metadata.name" | grep -E (-group|-node)$); do
  subctl export service $service -n pgd-group
done
```

数分後、ステータスは PGD グループが正常であることを示します。各 PGD グループが正常な状態になると、2つのデータノード `region-a` または `region-b` のいずれかで `app` データベースに書き込むことができます。データは他のデータノードに複製されます。

## 3.15 Transparent Data Encryption (TDE)

**重要:** TDE は、それをサポートするオペランドでのみ使用できます。EPAS バージョン 15 以降、Postgres Extended バージョン 15 以降。

Transparent Data Encryption (TDE) は、**保存データ**、つまりディスク上のデータベースファイルを暗号化するために複数のデータベースベンダーが使用しているテクノロジーです。ただし、TDE は使用中のデータを暗号化しません。

TDE は、バージョン 15 以降の EDB Postgres Advanced Server ServerEPAS または EDB Postgres Extended に含まれており、EDB Postgres Distributed for Kubernetes でサポートされています。

**重要:** 続行する前に、時間をかけてに慣れてください。

TDE をアクティブにすると、WAL ファイルとテーブルのファイルの両方が暗号化されます。データの暗号化/復号化は、アプリケーションの変更やクライアントドライバの更新を必要とせずにデータベースによって管理されるため、ユーザーに対して完全に透過的です。

EDB Postgres Distributed for Kubernetes での TDE のサポートは、EDB Postgres for Kubernetes PG4K からの実装に依存しています。参照してください

完全なコンテキストについては。

次に、Kubernetes Secret に保存されたパスフレーズで TDE を使用方法を示します。これは、EPAS バイナリキーの暗号化に使用されます。

#### 参考:

EPAS 暗号キーの詳細については、[こちら](#) を参照してください。

Kubernetes の EDB Postgres Distributed の TDE は、PG4K 実装に依存しています。クラスターで TDE をアクティブにするには、マニフェストの `epas` セクションを使用します。これは、ストレージなどの PG4K レベルの指示子に使用される `cnp` セクション内にあります。 `tde` スタンザを使用して TDE を有効にし、TDE 暗号化キーを保持する Kubernetes シークレットの名前を設定します。

次の YAML 部分には、パスフレーズ base-64 エンコードを保持するシークレットと、パスフレーズで TDE をアクティブにする `epas` セクションの両方が含まれています。

```
--
apiVersion: v1
kind: Secret
metadata:
  name: tde-key
data:
  key: ↵
  ↵bG9zcG9sbGl0b3NkaWN1bnBpb3Bpb3Bpb2N1YW5kb3RpZW51bmhhbWJyZW51YW5kb3RpZW51bmZyaW8=

--
apiVersion: pgd.k8s.enterprisedb.io/v1beta1
kind: PGDGroup
[...]
```

```
spec:
  instances: 3
[...]
```

```
  cnp:
    postgresql:
      epas:
        tde:
          enabled: true
          secretKeyRef:
            name: tde-key
            key: key
      storage:
        size: 1Gi
```

繰り返しになりますが、暗号化シークレットの作成方法や TDE の追加の使用方法など、詳細については、[を参照してください](#)。

に示すように、情報は保存時に暗号化されます。

たとえば、データノードの 1 つで psql ターミナルを開きます。

```
kubectl exec -ti <DATA-NODE> -- psql app
```

テキスト列を含む新しいテーブルを作成します。

```
create table foo(bar int, baz varchar);
insert into foo(bar, baz) values (1, hello), (2, goodbye);
```

次に、新しく定義されたテーブルがディスク上の保存される場所を確認します。

```
select pg_relation_filepath(foo);
pg_relation_filepath
-----
base/16385/16387
```

同じデータノードでターミナルを開くことができます。

```
kubectl exec -ti <DATA-NODE> -- bash
```

ファイルが暗号化されたことを確認します。

```
cd $PGDATA/base/16385
hexdump -C 16387 | grep hello
hexdump -C 16387 | grep goodbye
```

## 3.16 LDAP Authentication

EDB Postgres Distributed for Kubernetes は、LDAP 認証をサポートし、EDB EDB Postgres Distributed for Kubernetes の LDAP 構成は、EDB Postgres for Kubernetes PG4K からの実装に依存しています。[the PG4K documentation](#) を参照してください。

完全なコンテキストについては、

---

**重要:** 続行する前に、時間をかけて [LDAP authentication feature in the postgres documentation](#) を理解してください。

---

LDAP サポートでは、ユーザー認証のみが LDAP に送信されるため、ユーザーは postgres データベースに既に存在する必要があります。

次に、PGDGroup で simple bind モードを使用した LDAP 構成の例を示します。postgres は単に prefix + username + suffix とパスワードを使用して LDAP サーバーをバインドし、認証を達成します。

```
spec:
  [...]
  cnp:
    postgresql:
      ldap:
        server: ldap-service.ldap.svc.cluster.local
        bindAsAuth:
          prefix: "uid="
          suffix: ",dc=example,dc=org"
```

次に、PGDGroup で search+bind モードを使用した LDAP 構成の例を示します。このモードでは、postgres は最初に bindDN を使用して LDAP にバインドされ、パスワードはシークレット bindPassword に保存されます。次に、postgres は baseDN の下で検索を実行して、searchAttribute で指定された項目と一致するユーザー名を見つけようとします。一致するものが見つかった場合、postgres は、最終的に LDAP サーバーに対してエントリーとパスワードを検証します。

```
spec:
  [...]
  cnp:
    postgresql:
      ldap:
        server: ldap-service.ldap.svc.cluster.local
        bindSearchAuth:
          baseDN: "dc=example,dc=org"
          bindDN: "cn=admin,dc=example,dc=org"
          searchAttribute: "uid"
          bindPassword:
            name: ldap-bind-password
            key: password
```

## 3.17 Examples of configuration

**重要:** 使用可能な例は、デモンストレーションと実験のみを目的としています。

これらの例は、Kubernetes 環境で EDB Postgres 分散クラスターをセットアップするための構成ファイルです。

柔軟な 3 つのリージョン

```
apiVersion: pgd.k8s.enterprisedb.io/v1beta1
kind: PGDGroup
metadata:
  name: region-a
```

(次のページに続く)

(前のページからの続き)

```

spec:
  instances: 2
  proxyInstances: 2
  witnessInstances: 1
  pgd:
    parentGroup:
      name: world
      create: true
    discovery:
      - host: region-a-group
      - host: region-b-group
      - host: region-c-group
  cnp:
    storage:
      size: 1Gi
  connectivity:
    dns:
      additional:
        - domain: alternate.domain
        - domain: my.domain
      hostSuffix: -dc1
    tls:
      mode: verify-ca
      clientCert:
        caCertSecret: client-ca-key-pair
        certManager:
          spec:
            issuerRef:
              name: client-ca-issuer
              kind: Issuer
              group: cert-manager.io
      serverCert:
        caCertSecret: server-ca-key-pair
        certManager:
          spec:
            issuerRef:
              name: server-ca-issuer
              kind: Issuer
              group: cert-manager.io
  ---
apiVersion: pgd.k8s.enterprisedb.io/v1beta1
kind: PGDGroup
metadata:
  name: region-b
spec:

```

(次のページに続く)

(前のページからの続き)

```

instances: 2
proxyInstances: 2
witnessInstances: 1
pgd:
  parentGroup:
    name: world
  discovery:
    - host: region-a-group
    - host: region-b-group
    - host: region-c-group
cnp:
  storage:
    size: 1Gi
connectivity:
  tls:
    mode: verify-ca
    clientCert:
      caCertSecret: client-ca-key-pair
      certManager:
        spec:
          issuerRef:
            name: client-ca-issuer
            kind: Issuer
            group: cert-manager.io
    serverCert:
      caCertSecret: server-ca-key-pair
      certManager:
        spec:
          issuerRef:
            name: server-ca-issuer
            kind: Issuer
            group: cert-manager.io
- - -
apiVersion: pgd.k8s.enterprisedb.io/v1beta1
kind: PGDGroup
metadata:
  name: region-c
spec:
  instances: 0
  proxyInstances: 0
  witnessInstances: 1
  pgd:
    parentGroup:
      name: world
    discovery:

```

(次のページに続く)

(前のページからの続き)

```

- host: region-a-group
- host: region-b-group
- host: region-c-group
cnf:
  storage:
    size: 1Gi
connectivity:
  tls:
    mode: verify-ca
    clientCert:
      caCertSecret: client-ca-key-pair
      certManager:
        spec:
          issuerRef:
            name: client-ca-issuer
            kind: Issuer
            group: cert-manager.io
    serverCert:
      caCertSecret: server-ca-key-pair
      certManager:
        spec:
          issuerRef:
            name: server-ca-issuer
            kind: Issuer
            group: cert-manager.io

```

2つのデータグループと3つのリージョンにまたがるグローバル監視ノードを含む PGD クラスター。各データグループは2つのデータノードとローカル監視ノードで構成されます。

使用可能なオプションのリストについては、[API リファレンス](#)を参照してください。

## 3.18 API Reference

パッケージ v1beta1 には、pgd v1beta1 API グループの API スキーマ定義が含まれています

### 3.18.1 リソースタイプ

- *PGDGroup parting*
- *PGDGroupCleanup*

### 3.18.2 証明書キーストア

使用される箇所

- 証明書仕様

CertificateKeystores は、証明書の出力シークレットで作成される追加のキーストア出力形式を構成します。

| Field                          | Description                                                              |
|--------------------------------|--------------------------------------------------------------------------|
| <i>jks</i> _JKSKeystore_       | JKS は、JKS キーストアを `spec.secretName` Secret リソースに保存するためのオプションを構成します。       |
| <i>pkcs12</i> _PKCS12Keystore_ | PKCS12 は、PKCS12 キーストアを `spec.secretName` Secret リソースに保存するためのオプションを構成します。 |

### 3.18.3 CertificatePrivateKey

使用される箇所

- 証明書仕様

CertificatePrivateKey には、証明書コントローラが使用するプライベートキーの構成オプションが含まれています。これにより、プライベートキーをローテーションする方法を制御できます。

| Field                                            | Description                                                                                                                                                                                                                                                          |
|--------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>rotationPolicy</i> _PrivateKeyRotationPolicy_ | RotationPolicy は、再発行が処理されているときにプライベートキーを再生成する方法を制御します。Never に設定されている場合、プライベートキーはターゲット `spec.secretName` にまだ存在しない場合にのみ生成されます。存在するが、正しいアルゴリズムまたはサイズがない場合、警告が発生してユーザーの介入を待ちます。Always に設定されている場合、指定された要件と一致するプライベートキーは、再発行が発生するたびに生成されます。下位互換性を維持するために、デフォルトは'Never' です。 |
| <i>encoding</i> _PrivateKeyEncoding_             | この証明書のプライベートキーのプライベートキー暗号化標準 PKCS エンコードでエンコードされます。提供される場合、許可される値は `PKCS1` および `PKCS8` で、それぞれ PKCS#1 および PKCS#8 を表します。指定しない場合、デフォルトの `PKCS1`。                                                                                                                         |

次のページに続く

表 8 – 前のページからの続き

| Field                                        | Description                                                                                                                                                                                                                                                                                 |
|----------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>algorithm _PrivateKeyAlgorithm_</code> | Algorithm は、この証明書に対応するプライベートキーのプライベートキーアルゴリズムです。提供される場合、許可される値は `RSA`、`Ed25519` または `ECDSA` です。`algorithm` が指定され、`size` が提供されない場合、256 のキーサイズが `ECDSA` キーアルゴリズムに使用され、2048 のキーサイズが `RSA` キーアルゴリズムに使用されます。`Ed25519` キーアルゴリズムを使用する場合、キーサイズは無視されます。                                              |
| <code>size _int_</code>                      | Size は、この証明書に対応するプライベートキーのキービットサイズです。`algorithm` が `RSA` に設定されている場合、有効な値は `2048`、`4096` または `8192` であり、指定しない場合はデフォルトの `2048` になります。`algorithm` が `ECDSA` に設定されている場合、有効な値は `256`、`384` または `521` であり、指定しない場合はデフォルトの `256` になります。`algorithm` が `Ed25519` に設定されている場合、Size は無視されます。他の値は許可されません。 |

### 3.18.4 証明書仕様

#### 使用される箇所

- *CertManagerTemplate*

CertificateSpec は、証明書の目的の状態を定義します。有効な証明書には、CommonName、DNSName、または URISAN の少なくとも 1 つが有効である必要があります。

| Field                              | Description                                                                                                                                                                                                                                                         |
|------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>subject _X509Subject_</code> | 完全な X509 名前仕様 <a href="https://golang.org/pkg/crypto/x509/pkix/#Name">https://golang.org/pkg/crypto/x509/pkix/#Name</a> 。                                                                                                                                           |
| <code>commonName _string_</code>   | CommonName は、証明書で使用される一般的な名前です。無効な CSR の生成を回避するには、CommonName の長さが 64 文字以下である必要があります。サブジェクトの代替名が設定されている場合、この値は TLS クライアントによって無視されます。これは x509 動作です。 <a href="https://tools.ietf.org/html/rfc6125#section-6.4.4">https://tools.ietf.org/html/rfc6125#section-6.4.4</a> |

次のページに続く

表 9 – 前のページからの続き

| Field                                                           | Description                                                                                                                                                                                                                                                                                                                          |
|-----------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>duration</i> <i>_Duration_</i>                               | 証明書の要求された「期間」つまり有効期間。このオプションは、一部の発行者タイプでは無視/オーバーライドされる場合があります。設定しない場合、これはデフォルトの 90 日です。証明書は、その期間の 2/3、または有効期限が切れる前の `renewBefore` 期間のいずれか遅い方で更新されます。受付される最小期間は 1 時間です。値は Go で受け入れられる単位である必要があります <code>time.ParseDuration</code> <a href="https://golang.org/pkg/time/#ParseDuration">https://golang.org/pkg/time/#ParseDuration</a> |
| <i>renewBefore</i> <i>_Duration_</i>                            | 現在発行されている証明書の有効期限がどれくらい前に、 <code>cert-manager</code> が証明書を更新するか。デフォルトは、発行された証明書の期間の 2/3 です。最小許容値は 5 分です。値は Go で受け入れられる単位である必要があります <code>time.ParseDuration</code> <a href="https://golang.org/pkg/time/#ParseDuration">https://golang.org/pkg/time/#ParseDuration</a>                                                               |
| <i>dnsNames</i> <i>_[]string_</i>                               | DNSNames は、証明書に設定する DNS <code>subjectAltName</code> のリストです。                                                                                                                                                                                                                                                                          |
| <i>ipAddresses</i> <i>_[]string_</i>                            | IPAddresses は、証明書に設定する IP アドレスの <code>subjectAltNames</code> のリストです。                                                                                                                                                                                                                                                                 |
| <i>uris</i> <i>_[]string_</i>                                   | URIs は、証明書に設定する URI <code>subjectAltName</code> のリストです。                                                                                                                                                                                                                                                                              |
| <i>emailAddresses</i> <i>_[]string_</i>                         | EmailAddresses は、証明書に設定する電子メールの <code>subjectAltNames</code> のリストです。                                                                                                                                                                                                                                                                 |
| <i>secretName</i> <b>[Required]</b> <i>_string_</i>             | SecretName は、この証明書リソースによって自動的に作成および管理されるシークレットリソースの名前です。これには、指定された発行者によって署名されたプライベートキーと証明書が入力されます。重要 このフィールドは、元の <code>cert-manager</code> API 宣言では必須でした                                                                                                                                                                             |
| <i>keystores</i> <i>_CertificateKeystores_</i>                  | Keystores は、 <i>secretName</i> Secret リソースに保存される追加のキーストア出力形式を構成します。                                                                                                                                                                                                                                                                  |
| <i>issuerRef</i> <b>** Required]**</b> <i>_ObjectReference_</i> | IssuerRef は、この証明書の発行者への参照です。`kind` フィールドが設定されていない場合、または `Issuer` に設定されている場合、証明書と同じ名前空間にある指定された名前の発行者リソースが使用されます。`kind` フィールドが `ClusterIssuer` に設定されている場合、指定された名前の ClusterIssuer が使用されます。このスタンザの `name` フィールドは常に必要です。                                                                                                                 |
| <i>isCA</i> <i>_bool_</i>                                       | IsCA は、この証明書を証明書署名に対して有効なものとしてマークします。これにより、`cert sign` の使用法が `usages` のリストに自動的に追加されます。                                                                                                                                                                                                                                               |
| <i>usages</i> <i>_[]KeyUsage_</i>                               | 使用法は、証明書のために要求された x509 使用法のセットです。指定しない場合、デフォルトの `digital signature` および `key encipherment` になります。                                                                                                                                                                                                                                    |

次のページに続く

表 9 – 前のページからの続き

| Field                                           | Description                                                                                                                                                                                                                                                                                                                         |
|-------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>privateKey _CertificatePrivateKey_</code> | 証明書に使用されるプライベートキーを制御するオプション。                                                                                                                                                                                                                                                                                                        |
| <code>encodeUsagesInRequest _bool_</code>       | <code>EncodeUsagesInRequest</code> は、キーの使用法を証明書要求に存在するかどうかを制御します                                                                                                                                                                                                                                                                    |
| <code>revisionHistoryLimit _int32_</code>       | <code>versionHistoryLimit</code> は、証明書の履歴に維持される <code>CertificateRequest</code> リビジョンの最大数です。各リビジョンは、この証明書が作成、更新、または仕様が変更されたときに作成された単一の <code>CertificateRequest</code> を表します。リビジョンの数がこの数を超える場合、リビジョンは古い順に削除されます。設定する場合、 <code>revisionHistoryLimit</code> は `1` 以上の値である必要があります。`nil` を解除すると、リビジョンはガベージコレクションされません。デフォルト値は `nil` です。 |

### 3.18.5 コンディションステータス

(文字列のエイリアス)

`ConditionStatus` は、条件のステータスを表します。

### 3.18.6 JKSKeystore

使用される箇所

- 証明書キーストア

`JKSKeystore` は、JKS キーストアを `spec.secretName` Secret リソースに保存するためのオプションを構成します。

| Field                                                               | Description                                                                                                                                                                                                                                                                                                                                        |
|---------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>create [Required] _bool_</code>                               | <code>Create</code> は、証明書の JKS キーストアの作成を有効にします。 <code>true</code> の場合、 <code>keystore.jks</code> という名前のファイルがターゲットシークレットリソースに作成され、 <code>passwordSecretRef</code> に保存されているパスワードを使用して暗号化されます。キーストアファイルは、再発行の場合にのみ更新されます。 <code>truststore.jks</code> という名前のファイルもターゲットシークレットリソースに作成され、発行認証局を含む <code>passwordSecretRef</code> に保存されたパスワードを使用して暗号化されます |
| <code>passwordSecretRef ** Required] ** [_SecretKeySelector_</code> | <code>PasswordSecretRef</code> は、JKS キーストアの暗号化に使用されるパスワードを含む Secret リソースのキーへの参照です。                                                                                                                                                                                                                                                                 |

### 3.18.7 KeyUsage

(文字列のエイリアス)

使用される箇所

- 証明書仕様

KeyUsage は、キーの有効な使用コンテキストを指定します。以下を参照してください。 <https://tools.ietf.org/html/rfc5280#section-4.2.1.3>

<https://tools.ietf.org/html/rfc5280#section-4.2.1.12>

有効な KeyUsage 値は次のとおりです。“signing”、“digital 署名”、“content commitment”、“key encipherment”、“key Agreement”、“data encipherment”、“certsign”、“crlsign”、“encipher Only”、“復号のみ”、“any”、“サーバー認証”、“クライアント認証”、“コード署名”、“電子メール保護”、“s/mime”、“ipsec エンドシステム”、“ipsec トンネル”、“ipsec user”、“timestamping”、“ocsp signed”、“microsoft sgc”、“netscape sgc”

### 3.18.8 LocalObjectReference

使用される箇所

- *SecretKeySelector*

LocalObjectReference は、参照対象と同じ名前空間内のオブジェクトへの参照です。参照対象がクラスタースコープのリソース ClusterIssuer である場合、参照は代わりに、構成済みの’クラスターリソース名前空間’で指定された名前を持つリソースを参照します。これは、コントローラコンポーネントのフラグとして設定され、デフォルトで cert-manager が実行される名前空間)。

| Field                           | Description                                                                                                                                                                                 |
|---------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>name</i> [Required] _string_ | 参照されるリソースの名前。詳細 <a href="https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names">https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names</a> |

### 3.18.9 ObjectReference

使用される箇所

- 証明書仕様

ObjectReference は、指定された名前、種類、およびグループを持つオブジェクトへの参照です。

| Field                           | Description     |
|---------------------------------|-----------------|
| <i>name</i> [Required] _string_ | 参照されるリソースの名前。   |
| <i>group</i> _string_           | 参照されるリソースのグループ。 |

### 3.18.10 PKCS12Keystore

使用される箇所

- [証明書キーストア](#)

PKCS12Keystore は、PKCS12 キーストアを `spec.secretName` Secret リソースに保存するためのオプションを構成します。

| Field                                                               | Description                                                                                                                                                                                                                                                                                                                        |
|---------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>create [Required] _bool_</code>                               | Create は、証明書の PKCS12 キーストアの作成を有効にします。true の場合、 <code>`keystore.p12`</code> という名前のファイルがターゲットシークレットリソースに作成され、 <code>`passwordSecretRef`</code> に保存されているパスワードを使用して暗号化されます。キーストアファイルは、再発行の場合にのみ更新されます。 <code>`truststore.p12`</code> という名前のファイルもターゲットシークレットリソースに作成され、発行認証局を含む <code>`passwordSecretRef`</code> に保存されたパスワードを使用して暗号化されます |
| <code>passwordSecretRef ** [Required]** [_SecretKeySelector_</code> | PasswordSecretRef は、PKCS12 キーストアの暗号化に使用されるパスワードを含む Secret リソースのキーへの参照です。                                                                                                                                                                                                                                                           |

### 3.18.11 PrivateKeyAlgorithm

(文字列のエイリアス)

使用される箇所

- [CertificatePrivateKey](#)

PrivateKeyAlgorithm はプライベートキーアルゴリズムを表します

### 3.18.12 PrivateKeyEncoding

(文字列のエイリアス)

使用される箇所

- [CertificatePrivateKey](#)

PrivateKeyEncoding はプライベートキーエンコーディングを表します

### 3.18.13 PrivateKeyRotationPolicy

(文字列のエイリアス)

使用される箇所

- *CertificatePrivateKey*

PrivateKeyRotationPolicy は、証明書が発行されるときにプライベートキーを生成またはソースする方法を示します。

### 3.18.14 SecretKeySelector

使用される箇所

- *JKSKeystore*
- *PKCS12Keystore*

SecretKeySelector は、Secret リソース内の特定の「キー」への参照です。場合によっては、**key** は必須フィールドです。

| Field                                              | Description                                                                                 |
|----------------------------------------------------|---------------------------------------------------------------------------------------------|
| <i>LocalObjectReference</i> _LocalObjectReference_ | `LocalObjectReference`のメンバーはこのタイプに埋め込まれます。参照されるシークレットリソースの名前。                               |
| <i>key</i> _string_                                | 使用する Secret リソースの `data` フィールドのエントリのキー。このフィールドの一部のインスタンスはデフォルトである場合も、他のインスタンスは必須である場合があります。 |

### 3.18.15 X509 件名

使用される箇所

- 証明書仕様

X509 件名 完全な X509 名前仕様

| Field                                 | Description   |
|---------------------------------------|---------------|
| <i>organizations</i> _[]string_       | 証明書で使用する組織。   |
| <i>countries</i> _[]string_           | 証明書で使用する国。    |
| <i>organizationalUnits</i> _[]string_ | 証明書で使用する組織単位。 |
| <i>localities</i> _[]string_          | 証明書で使用する都市。   |
| <i>provinces</i> _[]string_           | 証明書で使用する州/地方。 |
| <i>streetAddresses</i> _[]string_     | 証明書で使用する住所。   |

次のページに続く

表 15 – 前のページからの続き

| Field                         | Description     |
|-------------------------------|-----------------|
| <i>postalCodes</i> _[]string_ | 証明書で使用する郵便番号。   |
| <i>serialNumber</i> _string_  | 証明書で使用するシリアル番号。 |

### 3.18.16 PGDGroup

PGDGroup は、pgdgroups API のスキーマです

| Field                                                 | Description                            |
|-------------------------------------------------------|----------------------------------------|
| <i>apiVersion</i> <b>[Required]</b> string            | <i>pgd.k8s.enterprisedb.io/v1beta1</i> |
| <i>kind</i> <b>[Required]</b> string                  | <i>PGDGroup</i>                        |
| <i>spec</i> <b>** Required</b> ** [_PGDGroupSpec_     | 説明はありません。                              |
| <i>status</i> <b>** Required</b> ** [_PGDGroupStatus_ | 説明はありません。                              |

### 3.18.17 PGDGroupCleanup

PGDGroupCleanup は、pgdgroupcleanups API のスキーマです

| Field                                                             | Description                            |
|-------------------------------------------------------------------|----------------------------------------|
| <i>apiVersion</i> <b>[Required]</b> string                        | <i>pgd.k8s.enterprisedb.io/v1beta1</i> |
| <i>kind</i> <b>[Required]</b> string                              | <i>PGDGroupCleanup</i>                 |
| <i>spec</i> <b>** Required</b> ** [_PGDGroupCleanup-<br>Spec_     | 説明はありません。                              |
| <i>status</i> <b>** Required</b> ** [_PGDGroupCleanup-<br>Status_ | 説明はありません。                              |

### 3.18.18 バックアップ

使用される箇所

- *PGDGroupSpec*

Backup は cnp-pgd ノードのバックアップを構成します

| Field                                    | Description                                           |
|------------------------------------------|-------------------------------------------------------|
| <i>configuration</i> <b>** Bootstrap</b> | バックアップに使用される CNP 構成。ServerName 値は、オペレーターによって予約されています。 |

次のページに続く

表 18 – 前のページからの続き

| Field                                                      | Description                                                                                            |
|------------------------------------------------------------|--------------------------------------------------------------------------------------------------------|
| <i>cron</i> _ScheduledBackupSpec_                          | データのスケジュールされたバックアップ。このフィールドは非推奨であり、将来のバージョンでは削除される予定です。                                                |
| <i>schedulers</i> ** Required]*** [_[]ScheduledBackupSpec_ | バックアップのスケジューラーを定義します。スケジューラーごとに異なるバックアップ方法があります。<br>`backup.cron`または`backup.schedulers`のいずれかのみを定義できます。 |

### 3.18.19 BackupStatus

使用される箇所

- *PGDGroupStatus*

BackupStatus には、pgd バックアップの現在のステータスが含まれます

| Field                                                              | Description                                                                                 |
|--------------------------------------------------------------------|---------------------------------------------------------------------------------------------|
| <i>clusterName</i> [Required] _string_                             | Cluster バックアップを取るように選択されたクラスターの名前を付けます。バックアップはさまざまな方法でスケジュールできますが、1つのクラスターのみがバックアップを取得します   |
| <i>scheduledBackupName</i> [Required] _string_                     | ScheduledBackupName は、スケジュールされたバックアップの名前です。このフィールドは非推奨であり、ScheduledBackupStatus に移行されます     |
| <i>scheduledBackupHash</i> [Required] _string_                     | ScheduledBackupHash は、スケジュールされたバックアップ構成のハッシュです。このフィールドは非推奨であり、ScheduledBackupStatus に移行されます |
| <i>scheduledBackups</i> ** Required]*** [_[]ScheduledBackupStatus_ | ScheduledBackupStatus には、スケジュールされたすべてのバックアップのステータスが含まれます                                    |

### 3.18.20 CNP ステータス

使用される箇所

- *PGDGroupStatus*

CNPStatus には、CNP に関するオペレーターの関連ステータスが含まれています

| Field                                   | Description |
|-----------------------------------------|-------------|
| <i>dataInstances</i> [Required] _int32_ | 説明はありません。   |

次のページに続く

表 20 – 前のページからの続き

| Field                                                                                              | Description                                     |
|----------------------------------------------------------------------------------------------------|-------------------------------------------------|
| <i>witnessInstances</i> <b>[Required]</b> _int32_                                                  | 説明はありません。                                       |
| <i>firstRecoverabilityPoints-ByMethod</i> ** Required]***_map[string]RecoverabilityPointsByMethod_ | 方法別の復旧可能ポイント、CNP ごとにキー付け クラスター Name nolint: lll |
| <i>firstRecoverabilityPoints</i> <b>[Required]</b> _map[string]string_                             | RFC3339 形式の日付としての CNP クラスター名前ごとにキー付けられた復旧ポイント   |
| <i>superUserSecretIsPresent</i> <b>[Required]</b> _bool_                                           | 説明はありません。                                       |
| <i>applicationUserSecretIsPresent</i> <b>[Required]</b> _bool_                                     | 説明はありません。                                       |
| <i>podDisruptionBudgetIsPresent</i> <b>[Required]</b> _bool_                                       | 説明はありません。                                       |

### 3.18.21 CertManagerTemplate

使用される箇所

- [ClientCertConfiguration](#)
- [ServerCertConfiguration](#)

CertManagerTemplate には、証明書要求を生成するためのデータが含まれています

| Field                                        | Description  |
|----------------------------------------------|--------------|
| <i>spec</i> ** Required]***_CertificateSpec_ | 証明書オブジェクト仕様  |
| <i>metadata</i> ** Required]***_Metadata_    | ラベルと注釈のメタデータ |

### 3.18.22 ClientCertConfiguration

使用される箇所

- [OTELTLS 構成](#)

ClientCertConfiguration には、streaming\_replica ユーザーの証明書を生成するための情報が含まれています

| Field                                                   | Description                                        |
|---------------------------------------------------------|----------------------------------------------------|
| <i>caCertSecret</i> <b>[Required]</b> _string_          | CACertSecret は、CloudNativePG 構成に注入される CA のシークレットです |
| <i>certManager</i> ** Required]***_CertManagerTemplate_ | 証明書の生成に使用される cert-manager テンプレート                   |

次のページに続く

表 22 – 前のページからの続き

| Field                                                                        | Description                                           |
|------------------------------------------------------------------------------|-------------------------------------------------------|
| <i>preProvisioned</i> ** Required]***_ClientPre-<br>ProvisionedCertificates_ | PreProvisioned には、事前生成されたクライアント証明書<br>を取得する方法が含まれています |

### 3.18.23 Client 事前プロビジョニングされた証明書

使用される箇所

- *ClientCertConfiguration*

ClientPreProvisionedCertificates は、事前生成されたクライアント証明書を取得する方法を指示します

| Field                                                                   | Description                                              |
|-------------------------------------------------------------------------|----------------------------------------------------------|
| <i>streamingReplica</i> ** Required]***_PreProvi-<br>sionedCertificate_ | StreamingReplica 'streaming_replica' ユーザーの事前生成さ<br>れた証明書 |

### 3.18.24 CnpBaseConfiguration

使用される箇所

- *CnpConfiguration*
- *PGDGroupSpec*

CnpBaseConfiguration には、CNP Witness ノードとデータノードの両方に適用できる構成パラメーターが含まれています

| Field                                                        | Description                                                                                                                                                    |
|--------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>startDelay</i> [Required] _int32_                         | PostgreSQL インスタンスが正常に起動するまでの秒単位<br>の時間デフォルト 3600                                                                                                               |
| <i>stopDelay</i> [Required] _int32_                          | PostgreSQL インスタンスノードが正常にシャットダウン<br>できる秒単位の時間デフォルト 180                                                                                                          |
| <i>smartShutdownTimeout</i> _int32_                          | Postgres のスマートシャットダウンが完了するために予約<br>された時間枠を制御する秒単位の時間。オペレーターが<br>Postgres の高速シャットダウンを要求するための十分な時<br>間を確保してください <i>stopDelay</i> - <i>smartShutdownTimeout</i> 。 |
| <i>storage</i> ** Required]***_StorageConfigura-<br>tion_    | インスタンスのストレージの構成                                                                                                                                                |
| <i>walStorage</i> ** Required]***_StorageConfig-<br>uration_ | インスタンスの WAL ストレージの構成                                                                                                                                           |

次のページに続く

表 24 – 前のページからの続き

| Field                                                                                                            | Description                                                                                                                                                                                                                                    |
|------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>clusterMaxStartDelay</i> <b>[Required]</b> _int32_                                                            | PostgreSQL インスタンスが正常に起動するまでの秒単位の時間デフォルト 300                                                                                                                                                                                                    |
| <i>affinity</i> _AffinityConfiguration_<br><i>resources</i> _ResourceRequirements_                               | ポッドのアフィニティ/アンチアフィニティルール<br>生成されたすべてのポッドのリソース要件。詳細については、 <a href="https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/">https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/</a> を参照してください。 |
| <i>postgresql</i> _PostgresConfiguration_<br><i>monitoring</i> <b>** Required]**</b> [_Monitoring-Configuration_ | PostgreSQL サーバーの構成<br>このクラスターの監視インフラストラクチャの構成                                                                                                                                                                                                  |
| <i>logLevel</i> <b>[Required]</b> _string_                                                                       | インスタンスのログレベル、次の値のいずれか Error、Warning、info デフォルト、debug、trace                                                                                                                                                                                     |
| <i>serviceAccountTemplate</i> <b>** Required]**</b> [_ServiceAccountTemplate_                                    | CNP に渡されるサービスアカウントテンプレート                                                                                                                                                                                                                       |
| <i>otel</i> <b>** Required]**</b> [_OTELConfiguration_<br><i>postInitSQL</i> _[]string_                          | OpenTelemetry 構成<br>ノードが作成された直後にスーパーユーザーとして実行される SQL クエリーのリスト-細心の注意を払って使用されますデフォルトでは空                                                                                                                                                          |
| <i>postInitTemplateSQL</i> _[]string_                                                                            | ノードが作成された後、`template1` のスーパーユーザーとして実行される SQL クエリーのリスト。細心の注意を払って使用しますデフォルトでは空                                                                                                                                                                   |
| <i>postInitApplicationSQL</i> _[]string_                                                                         | クラスターが作成された直後にアプリケーションデータベースでスーパーユーザーとして実行される SQL クエリーのリスト。論理的結合の前にアプリケーションデータベースで作成されたデータが結合に失敗するため、細心の注意を払って使用されますデフォルトでは空                                                                                                                   |
| <i>postInitApplicationSQLRefs</i> _PostInitApplicationSQLRefs_                                                   | クラスターが作成された直後にアプリケーションデータベースでスーパーユーザーとして実行される SQL ファイルを含む ConfigMap または Secret への参照のリスト。参照は特定の順序で処理されます。最初にすべてのシークレットが処理され、次にすべての ConfigMap が処理されます。各グループ内で、処理順序は、それぞれの配列で指定されたシーケンスに従います。細心の注意を払って使用しますデフォルトでは空                               |
| <i>seccompProfile</i> <b>** Required]**</b> [_Seccomp-Profile_                                                   | すべてのポッドとコンテナに適用される Seccomp プロファイル。デフォルト `RuntimeDefault`                                                                                                                                                                                       |
| <i>metadata</i> <b>** Required]**</b> [_InheritedMetadata_                                                       | 生成されたクラスターリソースにのみ適用されるメタデータ。AppArmor プロファイルを適用する場合に役立ちます。                                                                                                                                                                                      |
| <i>managed</i> <b>** Bootstrap</b>                                                                               | CNP インスタンスマネージャーによって管理される PostgreSQL の部分で使用される構成                                                                                                                                                                                               |
| <i>projectedVolumeTemplate</i> _ProjectedVolumeSource_                                                           | 投影ボリュームを定義するために使用されるテンプレート。投影ボリュームは `/projected` ベースフォルダーの下にマウントされます                                                                                                                                                                           |

次のページに続く

表 24 – 前のページからの続き

| Field                                                                         | Description                                                                                                                                                                                                                                            |
|-------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>tablespaces</i> <a href="#">_[]TablespaceConfiguration_</a>                | テーブルスペース構成                                                                                                                                                                                                                                             |
| <i>topologySpreadConstraints</i> <a href="#">_[]TopologySpreadConstraint_</a> | TopologySpreadConstraints は、指定されたトポロジ間で一致するポッドを分散する方法を指定します。詳細 <a href="https://kubernetes.io/docs/concepts/scheduling-eviction/topology-spread-constraints/">https://kubernetes.io/docs/concepts/scheduling-eviction/topology-spread-constraints/</a> |

### 3.18.25 CnpConfiguration

使用される箇所

- *PGDGroupSpec*

CnpConfiguration には、PGD グループを構成する結果のクラスターに注入されるデータノードの構成が含まれています

| Field                                                              | Description                                                                                                                                                                                                                                   |
|--------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>CnpBaseConfiguration</i> <a href="#">_CnpBaseConfiguration_</a> | <code>`CnpBaseConfiguration`</code> のメンバーはこのタイプに埋め込まれます。説明はありません。                                                                                                                                                                             |
| <i>enableSuperuserAccess</i> <a href="#">_bool_</a>                | このオプションが有効になっている場合、CNP オペレーターは SuperuserSecret で定義されたシークレットを作成または使用して、スーパーユーザー postgres のデータベースへのアクセスを許可します。実行中のグループでこのオプションが無効になっている場合、オペレーターはシークレットの内容を無視し、 <code>`postgres`</code> ユーザーのパスワードを <code>`NULL`</code> に設定します。デフォルトで有効になっています。 |
| <i>superuserSecret</i> <a href="#">_LocalObjectReference_</a>      | スーパーユーザーのパスワードを含むシークレット。定義されていない場合、ランダムに生成されたパスワードを使用して新しいシークレットが作成されます。このフィールドは、CNP インスタンス構成でのみ許可されます。Witness ノードは常に他のインスタンスと同じスーパーユーザーシークレットを使用します。                                                                                         |

### 3.18.26 ConnectionString

(`map[string]string` のエイリアス)

使用される箇所

- *PgdConfiguration*

ConnectionString は、PostgreSQL クラスターに接続するためのパラメーターを表します

### 3.18.27 ConnectivityConfiguration

使用される箇所

- *PGDGroupSpec*

ConnectivityConfiguration では、PGDGroup のサービスと証明書を生成する方法について説明します

| Field                                                                                   | Description                                         |
|-----------------------------------------------------------------------------------------|-----------------------------------------------------|
| <i>dns</i> <b>** Required]**</b> <i>[_RootDNSConfigura-</i><br><i>tion_</i>             | リソースの FQDN を生成する方法について説明します                         |
| <i>tls</i> <b>** Required]**</b> <i>[_TLSConfiguration_</i>                             | TLS インフラストラクチャの構成                                   |
| <i>nodeServiceTemplate</i> <b>** Required]**</b> <i>[_Ser-</i><br><i>viceTemplate_</i>  | 各ノードのサービスを生成する方法を指示します                              |
| <i>groupServiceTemplate</i> <b>** Required]**</b> <i>[_Ser-</i><br><i>viceTemplate_</i> | PGDGroup のサービスを生成する方法を指示します                         |
| <i>proxyServiceTemplate</i> <b>** Required]**</b> <i>[_Ser-</i><br><i>viceTemplate_</i> | 書き込みリーダーノードルーティングの PGD プロキシを<br>指すサービスを生成する方法を指示します |
| <i>proxyReadServiceTemplate</i> <b>** Required]**</b><br><i>[_ServiceTemplate_</i>      | 読み取りノードルーティングの PGD プロキシを指すサー<br>ビスを生成する方法を指示します     |

### 3.18.28 接続ステータス

使用される箇所

- *PGDGroupStatus*

ConnectivityStatus には、接続に関するオペレーターの関連ステータスが含まれています

| Field                                                                                            | Description                                                             |
|--------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------|
| <i>replicationTLSCertificate</i> <b>** Required]**</b><br><i>[_ReplicationCertificateStatus_</i> | ReplicationTLSCertificate は、レプリケーション TLS 証明<br>書の名前です。それがあある場合          |
| <i>nodeTLSCertificates</i> <b>** Required]**</b><br><i>[_[]NodeCertificateStatus_</i>            | NodeTLSCertificates は、PGD ノード用に作成された証明<br>書の名前です。                       |
| <i>unusedCertificates</i> <b>[Required]</b> <i>[_[]string_</i>                                   | 未使用の証明書は、PGD ノードで使用されなくなった証<br>明書の名前です。                                 |
| <i>nodesWithoutCertificates</i> <b>[Required]</b><br><i>[_[]string_</i>                          | NodesWithoutCertificates は、サーバー証明書を持っていな<br>いノードの名前です。                  |
| <i>nodesNeedingServiceReconciliation</i> <b>[Re-</b><br><b>quired]</b> <i>[_[]string_</i>        | NodesNeedingServiceReconciliation は、サーバー証明書を持<br>っていないノードの名前です。         |
| <i>configurationHash</i> <b>[Required]</b> <i>_string_</i>                                       | ConfigurationHash は、接続構成のハッシュコードであり、<br>構成に変更があったかどうかを確認するために使用されま<br>す |

### 3.18.29 DNS 構成

使用される箇所

- *RootDNSConfiguration*

DNSConfiguration は、リソースの FQDN を生成する方法を説明します

| Field                                               | Description                                                                                                                                                  |
|-----------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>domain</i> <b>[Required]</b> <i>_string_</i>     | PGDGroup 内のすべてのサービスのドメイン名が含まれます。ここで指定された値がレンダリングされた <i>nodeServiceTemplate</i> および <i>groupServiceTemplate</i> と一致することを確認するのはユーザーの責任です。                      |
| <i>hostSuffix</i> <b>[Required]</b> <i>_string_</i> | PGDGroup 内のすべてのサービス名に追加するオプションのサフィックスが含まれます。この設定の意味は、ユーザーがルーティングの目的で場所で作成されたすべてのサービスを簡単にマークできるようにすることです。つまり、汎用ルールを CoreDNS に追加して、一部のサービスサフィックスをローカルとして書き換えます |

### 3.18.30 DiscoveryJobConfig

使用される箇所

- *PgdConfiguration*

DiscoveryJobConfig には、ディスカバリージョブを構成する一連のフィールドが含まれています

| Field                                         | Description                    |
|-----------------------------------------------|--------------------------------|
| <i>delay</i> <b>[Required]</b> <i>_int_</i>   | 再試行間のスリープまでの遅延時間秒単位で測定         |
| <i>retries</i> <b>[Required]</b> <i>_int_</i> | Retries 操作を再試行する回数             |
| <i>timeout</i> <b>[Required]</b> <i>_int_</i> | タイムアウト 操作が成功するまでに与えられた時間秒単位で測定 |

### 3.18.31 継承されたメタデータ

使用される箇所

- *CnpBaseConfiguration*
- *PGDGroupSpec*

InheritedMetadata には、クラスターに関連するすべてのリソースによって継承されるメタデータが含まれます

| Field                                                    | Description |
|----------------------------------------------------------|-------------|
| <i>labels</i> <b>[Required]</b> _map[string]string_      | 説明はありません。   |
| <i>annotations</i> <b>[Required]</b> _map[string]string_ | 説明はありません。   |

### 3.18.32 メタデータ

使用される箇所

- *CertManagerTemplate*
- *ServiceTemplate*

メタデータは `metav1.ObjectMeta` に似た構造ですが、`controller-gen` によって解析され、ユーザーに適切な CRD を作成できます。

| Field                                  | Description                                                                                                                                                                                                                           |
|----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>labels</i> _map[string]string_      | オブジェクトを整理および分類および選択するために使用できる文字列キーと値のマッピング。レプリケーションコントローラとサービスのセレクトターと一致する場合があります。詳細 <a href="http://kubernetes.io/docs/user-guide/labels">http://kubernetes.io/docs/user-guide/labels</a>                                            |
| <i>annotations</i> _map[string]string_ | アノテーションは、リソースと共に保存される非構造化キー値マッピングであり、任意のメタデータを保存および取得する外部ツールによって設定される場合があります。これらは照会不可であり、オブジェクトを変更するときに保存する必要があります。詳細 <a href="http://kubernetes.io/docs/user-guide/annotations">http://kubernetes.io/docs/user-guide/annotations</a> |

### 3.18.33 NodeCertificateStatus

使用される箇所

- **接続ステータス**

`NodeCertificateStatus` は、CNP ノードのサーバー証明書のステータスをカプセル化します

| Field                                                               | Description                                                  |
|---------------------------------------------------------------------|--------------------------------------------------------------|
| <i>ReplicationCertificateStatus</i> _Replication-CertificateStatus_ | `ReplicationCertificateStatus` のメンバーはこのタイプに埋め込まれます。説明はありません。 |
| <i>nodeName</i> <b>[Required]</b> _string_                          | <code>nodeName</code> は、この証明書を使用する CNP クラスターの名前です。           |

### 3.18.34 NodeKindName

(文字列のエイリアス)

使用される箇所

- ノードの概要

NodeKindName は、bdr.node\_summary からの node\_kind\_name の潜在的な値を含むタイプです

### 3.18.35 ノードの概要

使用される箇所

- PGDGroupStatus

Node 概要は bdr.node\_summary からの関連情報を表示します

| Field                                                    | Description                    |
|----------------------------------------------------------|--------------------------------|
| <i>node_name</i> <b>[Required]</b> _string_              | ノードの名前                         |
| <i>node_group_name</i> <b>[Required]</b> _string_        | NodeGroupName は、参加したグループの名前です。 |
| <i>peer_state_name</i> <b>[Required]</b> _string_        | 人間が判読可能な形式でのノードの一貫した状態         |
| <i>peer_target_state_name</i> <b>[Required]</b> _string_ | ノードが到達しようとしている状態参加または昇格中に      |
| <i>node_kind_name</i> ** Required]**                     | ノードの種類 witness または data        |
| [_NodeKindName_                                          |                                |

### 3.18.36 NodesExtensions ステータス

[[github.com/EnterpriseDB/pg4k-pgd/api/v1beta1.NodeExtensionStatus](https://github.com/EnterpriseDB/pg4k-pgd/api/v1beta1.NodeExtensionStatus)] のエイリアス

NodesExtensionsStatus には、NodeExtensionStatus エントリのリストが含まれています

### 3.18.37 OTELConfiguration

使用される箇所

- CnpBaseConfiguration

OTELConfiguration は、外部 openTelemetry の構成です

| Field                                        | Description                                   |
|----------------------------------------------|-----------------------------------------------|
| <i>metricsURL</i> <b>[Required]</b> _string_ | メトリックデータを受け入れる OpenTelemetry HTTP エンドポイント URL |

[次のページに続く](#)

表 34 – 前のページからの続き

| Field                                                           | Description                                                                   |
|-----------------------------------------------------------------|-------------------------------------------------------------------------------|
| <i>traceURL</i> <b>[Required]</b> <i>_string_</i>               | トレースデータを受け入れる OpenTelemetry HTTP エンドポイント URL                                  |
| <i>traceEnable</i> <b>[Required]</b> <i>_bool_</i>              | トレースデータを OpenTelemetry の <i>traceUrl</i> にプッシュするかどうか                          |
| <i>tls</i> <b>** Required]**</b> <i>[_OTELTLSConfiguration_</i> | TLSConfiguration は、MetricsURL および TraceURL が HTTPS を使用している場合に TLS 証明書構成を提供します |

### 3.18.38 OTELTLS 構成

使用される箇所

- *OTELConfiguration*

OTELTLSConfiguration には、openTelemetry への TLS 接続の証明書構成が含まれています

| Field                                                                        | Description                                                                                                                           |
|------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| <i>caBundleSecretRef</i> <b>** Required]**</b> <i>[_SecretKeySelector_</i>   | CABundleSecretRef は、openTelemetry サーバー証明書を検証するための CA バンドルを含むシークレットフィールドへの参照です                                                         |
| <i>clientCertSecret</i> <b>** Required]**</b> <i>[_LocalObjectReference_</i> | ClientCertSecret は、openTelemetry への接続に使用されるクライアント証明書を含むシークレットの名前です。これには、PEM 形式でエンコードされた標準の「tls.crt」ファイルと「tls.key」ファイルの両方が含まれる必要があります。 |

### 3.18.39 PGDGroupCleanupSpec

使用される箇所

- *PGDGroupCleanup*

PGDGroupCleanupSpec は、PGDGroupCleanup の目的の状態を定義します

| Field                                             | Description                                                |
|---------------------------------------------------|------------------------------------------------------------|
| <i>executor</i> <b>[Required]</b> <i>_string_</i> | 説明はありません。                                                  |
| <i>target</i> <b>[Required]</b> <i>_string_</i>   | 説明はありません。                                                  |
| <i>force</i> <b>[Required]</b> <i>_bool_</i>      | Force は、ターゲット PGDGroup ノードが分離されていない場合でも、PGDGroup の削除を強制します |

### 3.18.40 PGDGroupCleanupStatus

使用される箇所

- *PGDGroupCleanup*

PGDGroupCleanupStatus は、PGDGroupCleanup の観察された状態を定義します

| Field                                                                                                       | Description |
|-------------------------------------------------------------------------------------------------------------|-------------|
| <i>phase</i> <b>[Required]</b> <i>_github.com/EnterpriseDB/pg4k-pgd/pkg/resources.OperatorPhaseCleanup_</i> | 説明はありません。   |

### 3.18.41 PGDGroupSpec

使用される箇所

- *PGDGroup parting*

PGDGroupSpec は、PGDGroup の目的の状態を定義します

| Field                                                                          | Description                                                                                                                                                                                                                                                                                                       |
|--------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>imageName</i> <b>[Required]</b> <i>_string_</i>                             | コンテナイメージの名前、タグ <code>&lt;image&gt;:&lt;tag&gt;</code> と決定的で反復可能な展開用のダイジェスト <code>&lt;image&gt;:&lt;tag&gt;@sha256:&lt;digestValue&gt;</code> の両方をサポート                                                                                                                                                             |
| <i>imagePullPolicy</i> <i>_PullPolicy_</i>                                     | イメージプルポリシー。 <code>Always</code> 、 <code>Never</code> または <code>IfNotPresent</code> のいずれか。定義されていない場合、デフォルトの <code>IfNotPresent</code> が使用されます。更新できません。詳細 <a href="https://kubernetes.io/docs/concepts/containers/images#updating-images">https://kubernetes.io/docs/concepts/containers/images#updating-images</a> |
| <i>imagePullSecrets</i> <b>** Required]**</b> <i>[_[]LocalObjectReference_</i> | 演算子またはオペランドイメージをプルするために使用されるプルシークレットのリスト                                                                                                                                                                                                                                                                          |
| <i>inheritedMetadata</i> <b>** Required]**</b> <i>[_InheritedMetadata_</i>     | pgdGroup に関連するすべてのオブジェクトが継承するメタデータ                                                                                                                                                                                                                                                                                |
| <i>instances</i> <b>[Required]</b> <i>_int32_</i>                              | クラスターに必要なインスタンスの数                                                                                                                                                                                                                                                                                                 |
| <i>proxyInstances</i> <b>[Required]</b> <i>_int32_</i>                         | クラスターに必要なプロキシインスタンスの数                                                                                                                                                                                                                                                                                             |
| <i>witnessInstances</i> <b>[Required]</b> <i>_int32_</i>                       | クラスターに必要な監視インスタンスの数                                                                                                                                                                                                                                                                                               |
| <i>backup</i> <b>** Required]**</b> <i>[_Backup_</i>                           | CNP インスタンスのバックアップに使用される構成。                                                                                                                                                                                                                                                                                        |
| <i>restore</i> <b>** Required]**</b> <i>[_Restore_</i>                         | オブジェクトストアサービスからこの PGD グループを復元する構成                                                                                                                                                                                                                                                                                 |
| <i>cnp</i> <b>** Required]**</b> <i>[_CnpConfiguration_</i>                    | PGD グループを構成する CNP クラスターに注入されるインスタンス構成                                                                                                                                                                                                                                                                             |

次のページに続く

表 38 – 前のページからの続き

| Field                                                               | Description                                                                               |
|---------------------------------------------------------------------|-------------------------------------------------------------------------------------------|
| <i>witness</i> ** Required]***_CnpBaseConfigura-<br>tion_           | WitnessInstances CNP クラスターに注入される WitnessIn-<br>stances 構成定義されていない場合、デフォルトのインスタ<br>ンス構成    |
| <i>pgd</i> ** Required]***_PgdConfiguration_                        | Pgd には、このクラスターをブートストラップするための<br>指示が含まれています                                                |
| <i>pgdProxy</i> ** Required]***_PGDProxyCon-<br>figuration_         | PGDProxy には、PGD プロキシの構成手順が含まれていま<br>す                                                     |
| <i>connectivity</i> ** Required]***_Connectivity-<br>Configuration_ | 使用されるサービスや証明書など、PGDGroup の接続を構<br>成します。                                                   |
| <i>failingFinalizerTimeLimitSeconds</i> [Re-<br>quired] _int32_     | ファイナライザーが失敗した場合にオペレーターが待機す<br>る秒数。オペレーターが PGD グループのノードに到達で<br>きない場合、ファイナライザーは失敗したと考えられます。 |

### 3.18.42 PGDGroupStatus

使用される箇所

- *PGDGroup parting*

PGDGroupStatus は、PGDGroup の観察された状態を定義します

| Field                                                                                         | Description                                           |
|-----------------------------------------------------------------------------------------------|-------------------------------------------------------|
| <i>latestGeneratedNode</i> [Required] _int32_                                                 | 最後に生成されたノードの ID ノード名の衝突を回避する<br>ために使用されます             |
| <i>phase</i> [Required]<br>_github.com/EnterpriseDB/pg4k-<br>pgd/pkg/resources.OperatorPhase_ | このクラスターの初期化フェーズ                                       |
| <i>phaseDetails</i> [Required] _string_                                                       | 現在のフェーズの詳細                                            |
| <i>phaseTroubleshootHints</i> [Required] _string_                                             | PhaseTroubleshootHints 特定のフェーズの一般的なトラブ<br>ルシューティングの兆候 |
| <i>phaseType</i> [Required]<br>_github.com/EnterpriseDB/pg4k-<br>pgd/pkg/resources.PhaseType_ | PhaseType は、フェーズカテゴリを説明します。                           |
| <i>conditions</i> ** Required]***_[]Condition_                                                | PGDGroup オブジェクトの条件                                    |
| <i>nodes</i> ** Required]***_[]NodeSummary_                                                   | グループ内のノードのサマリーのリスト                                    |
| <i>backup</i> ** Required]***_BackupStatus_                                                   | この PGD グループのバックアップを取得しているノード                          |
| <i>restore</i> ** Required]***_RestoreStatus_                                                 | 復元プロセスのステータス                                          |
| <i>PGD</i> ** Required]***_PGDStatus_                                                         | PGD の最後の既知のステータス                                      |
| <i>CNP</i> ** Required]***_CNPStatus_                                                         | CNP の最後に知られているステータス                                   |
| <i>PGDProxy</i> ** Required]***_PGDProxySta-<br>tus_                                          | PGDP プロキシの最後の既知のステータス                                 |

次のページに続く

表 39 – 前のページからの続き

| Field                                                      | Description      |
|------------------------------------------------------------|------------------|
| <i>connectivity</i> ** Required]**** [_ConnectivityStatus_ | 接続の最後の既知のステータス   |
| <i>pause</i> ** Required]**** [_PauseStatus_               | 一時停止の最後の既知のステータス |

### 3.18.43 PGDNodeGroupEntry

使用される箇所

- *PGD ステータス*

PGDNodeGroupEntry は、PGD 構成で使用可能なノードグループに関する情報を示します

| Field                                                | Description                                                                                                           |
|------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------|
| <i>name</i> <b>[Required]</b> _string_               | Name はノードグループの名前です                                                                                                    |
| <i>enableProxyRouting</i> <b>[Required]</b> _bool_   | EnableProxyRouting は true です。ノードグループで PGD プロキシの実行が許可されています                                                            |
| <i>enableRaft</i> <b>[Required]</b> _bool_           | ノードグループにサブグループ raft インスタンスがある場合、EnableRaft は true です                                                                  |
| <i>routeWriterMaxLag</i> <b>[Required]</b> _int64_   | RouteWriterMaxLag 書き込みリーダーとして選択される新しい書き込み候補のバイト単位の最大遅延。これを通過する候補がない場合、自動的に選択されるライタはありません                              |
| <i>routeReaderMaxLag</i> <b>[Required]</b> _int64_   | RouteReaderMaxLag 実行可能な読み取り専用ノードと考えられるノードのバイト単位の最大遅延                                                                  |
| <i>routeWriterWaitFlush</i> <b>[Required]</b> _bool_ | RouteWriterWaitFlush <code>bdr.routing_leadership_transfer()</code> を使用する場合、新しいリーダーに切り替える前にレプリケーションキューのフラッシュを待機するかどうか |

### 3.18.44 PGDNodeGroupSettings

使用される箇所

- *PgdConfiguration*

PGDNodeGroupSettings には、PGD グループの設定が含まれています

| Field                                                                   | Description                                                                                                                         |
|-------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| <code>routeWriterMaxLag</code> <b>[Required]</b> <code>_int64_</code>   | RouteWriterMaxLag 書き込みリーダーとして選択される新しい書き込み候補のバイト単位の最大遅延。候補がこれを通過しない場合、ライタは自動的に選択されません デフォルト-1                                        |
| <code>routeReaderMaxLag</code> <b>[Required]</b> <code>_int64_</code>   | RouteReaderMaxLag 実行可能な読み取り専用ノードと考えられるノードのバイト単位の最大遅延デフォルト-1                                                                         |
| <code>routeWriterWaitFlush</code> <b>[Required]</b> <code>_bool_</code> | RouteWriterWaitFlush <code>`bdr.routing_leadership_transfer()`</code> を使用する場合、新しいリーダーに切り替える前にレプリケーションキューのフラッシュを待機するかどうか デフォルト false |

### 3.18.45 PGDProxyConfiguration

使用される箇所

- [PGDGroupSpec](#)

PGDProxyConfiguration は、PGD プロキシの構成を定義します

| Field                                                                             | Description                                             |
|-----------------------------------------------------------------------------------|---------------------------------------------------------|
| <code>imageName</code> <b>[Required]</b> <code>_string_</code>                    | PGDProxy コンテナイメージの名前                                    |
| <code>logLevel</code> <b>[Required]</b> <code>_string_</code>                     | PGD プロキシログレベル、次の値のいずれかエラー、警告、情報デフォルト、デバッグ、トレース          |
| <code>logEncoder</code> <b>[Required]</b> <code>_string_</code>                   | ログ出力のフォーマット                                             |
| <code>proxyAffinity</code> <b>** Required</b> <code>** [_Affinity_</code>         | ポッドのプロキシアフィニティ/アンチアフィニティルール                             |
| <code>proxyNodeSelector</code> <b>[Required]</b> <code>_map[string]string_</code> | ポッドの ProxyNodeSelector ルール                              |
| <code>proxyTolerations</code> <b>** Required</b> <code>** [_[]Toleration_</code>  | ポッドのプロキシトレランスルール                                        |
| <code>proxyResources</code> <code>_ResourceRequirements_</code>                   | プロキシに割り当てられるリソースを定義します。定義されていない場合、デフォルトの要求を使用し、値を制限します。 |

### 3.18.46 PGDProxyEntry

使用される箇所

- [PGD ステータス](#)

PGDProxyEntry は、PGD 構成で使用可能なプロキシに関する情報を表示します

| Field                                                  | Description                                                      |
|--------------------------------------------------------|------------------------------------------------------------------|
| <i>name</i> <b>[Required]</b> _string_                 | Name はプロキシの名前です                                                  |
| <i>fallbackGroupNames</i> <b>[Required]</b> _[]string_ | FallbackGroupNames は、このプロキシ用に構成されたフォールバックグループの名前です。              |
| <i>parentGroupName</i> <b>[Required]</b> _string_      | ParentGroupName は、このプロキシの親 PGD グループです。                           |
| <i>maxClientConn</i> <b>[Required]</b> _int_           | MaxClientConn プロキシが受け入れる接続の最大数                                   |
| <i>maxServerConn</i> <b>[Required]</b> _int_           | MaxServerConn プロキシが Postgres ノードへの接続の最大数                         |
| <i>serverConnTimeout</i> <b>[Required]</b> _int64_     | ServerConnTimeout サーバー接続の秒単位の接続タイムアウト                            |
| <i>serverConnKeepalive</i> <b>[Required]</b> _int64_   | ServerConnKeepalive サーバー接続の秒単位のキープアライブ間隔                         |
| <i>fallbackGroupTimeout</i> <b>[Required]</b> _int64_  | FallbackGroupTimeout ルーティングが fallback_group のいずれかにフォールバックするまでの間隔 |
| <i>consensusGracePeriod</i> <b>[Required]</b> _int64_  | コンセンサスグレースピリオド Raft リーダーを失ってもプロキシがルーティングを継続する秒単位の期間。             |
| <i>readListenPort</i> <b>[Required]</b> _int_          | ReadListenPort は、プロキシがクエリを監視し、読み取りノードにルーティングするポートです              |

### 3.18.47 PGDP プロキシ設定

使用される箇所

- [PgdConfiguration](#)

PGDProxySettings には、プロキシの設定が含まれています

| Field                                                 | Description                                                               |
|-------------------------------------------------------|---------------------------------------------------------------------------|
| <i>fallbackGroups</i> <b>[Required]</b> _[]string_    | FallbackGroups は、この PGD グループのすべてのデータノードが利用できないときにプロキシが接続を転送するグループのリストです   |
| <i>maxClientConn</i> <b>[Required]</b> _int_          | MaxClientConn プロキシが受け入れる接続の最大数。デフォルト 32767                                |
| <i>maxServerConn</i> <b>[Required]</b> _int_          | MaxServerConn プロキシが Postgres ノードへの接続の最大数。デフォルト 32767                      |
| <i>serverConnTimeout</i> <b>[Required]</b> _int64_    | ServerConnTimeout サーバー接続の秒単位の接続タイムアウト。デフォルト 2                             |
| <i>serverConnKeepalive</i> <b>[Required]</b> _int64_  | ServerConnKeepalive サーバー接続の秒単位のキープアライブ間隔。デフォルト 10                         |
| <i>fallbackGroupTimeout</i> <b>[Required]</b> _int64_ | FallbackGroupTimeout ルーティングが fallback_group のいずれかにフォールバックするまでの間隔。デフォルト 60 |

次のページに続く

表 44 – 前のページからの続き

| Field                                                 | Description                                                                                              |
|-------------------------------------------------------|----------------------------------------------------------------------------------------------------------|
| <i>consensusGracePeriod</i> <b>[Required]</b> _int64_ | コンセンサスグレースピリオド Raft リーダーを失ってもプロキシがルーティングを継続する秒単位の期間。0 に設定すると、プロキシはルーティングをすぐに停止します。デフォルト 6                |
| <i>enableReadNodeRouting</i> <b>[Required]</b> _bool_ | EnableReadNodeRouting は、プロキシが read_listen_port を介して読み取りノードにクエリーをルーティングするかどうかを制御するスイッチです。デフォルトでは false です |

### 3.18.48 PGDP プロキシステータス

使用される箇所

- *PGDGroupStatus*

PGDProxyStatus PGDP プロキシに関するオペレーターの関連ステータス

| Field                                           | Description                                                        |
|-------------------------------------------------|--------------------------------------------------------------------|
| <i>proxyInstances</i> <b>[Required]</b> _int32_ | 説明はありません。                                                          |
| <i>writeLead</i> <b>[Required]</b> _string_     | WriteLead はオペレーターの予約フィールドであり、外部使用を目的としたものではありません。将来のバージョンでは削除される予定 |
| <i>proxyHash</i> <b>[Required]</b> _string_     | ProxyHash には、プロキシを調整する必要があるかどうかを検出するために使用するハッシュが含まれています            |

### 3.18.49 PGD ステータス

使用される箇所

- *PGDGroupStatus*

PGDStatus PGD に関するオペレーターの関連ステータス

| Field                                                                                                                  | Description                                                                    |
|------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------|
| <i>raftConsensusLastChangedStatus</i> <b>[Required]</b> _github.com/EnterpriseDB/pg4k-pgd/pkg/resources.PGDRaftStatus_ | RaftConsensusLastChangedStatus は、bdr.monitor_group_raft から最新に報告されたステータスを示します   |
| <i>raftConsensusLastChangedMessage</i> <b>[Required]</b> _string_                                                      | RaftConsensusLastChangedMessage は、bdr.monitor_group_raft からの最新の報告されたメッセージを示します |

次のページに続く

表 46 – 前のページからの続き

| Field                                                               | Description                                                     |
|---------------------------------------------------------------------|-----------------------------------------------------------------|
| <i>raftConsensusLastChangedTimestamp</i> <b>[Required]</b> _string_ | RaftConsensusLastChangedTimestamp は、ステータスとメッセージが最初に報告されたときを示します |
| <i>registeredProxies</i> ** Required]***_PGDProxyEntry_             | RegisteredProxies は、登録されたプロキシのステータスです                           |
| <i>nodeGroup</i> ** Required]***_PGDNodeGroupEntry_                 | NodeGroup は、PGDGroup に関連付けられたノードグループのステータスです                    |

### 3.18.50 ParentGroupConfiguration

使用される箇所

- *PgdConfiguration*

ParentGroupConfiguration には、PGD のトポロジ構成が含まれます

| Field                                  | Description                                        |
|----------------------------------------|----------------------------------------------------|
| <i>name</i> <b>[Required]</b> _string_ | 親グループの名前                                           |
| <i>create</i> <b>[Required]</b> _bool_ | オペレーターが親グループが存在しない場合に作成する必要がある場合に、Create は true です |

### 3.18.51 PauseStatus

使用される箇所

- *PGDGroupStatus*

PauseStatus には、グループ冬眠の情報が含まれています

| Field                                               | Description                                              |
|-----------------------------------------------------|----------------------------------------------------------|
| <i>active</i> <b>[Required]</b> _bool_              | Active は、PGD グループが次のいずれかであることを示します。 _一時停止中 _既に一時停止中 *再開中 |
| <i>instances</i> <b>[Required]</b> _int32_          | Instances は、一時停止された PGD インスタンスの数です                       |
| <i>lastStartedTime</i> ** Required]**_Time_         | LastStartedTime は、PGDGroup が最後に一時停止を開始した時間です             |
| <i>lastCompletedTime</i> ** Required]**_Time_       | LastCompletedTime は、PGDGroup が最後に一時停止を完了した時間です           |
| <i>lastResumeStartedTime</i> ** Required]**_Time_   | LastResumeStartedTime は、PGDGroup が最後に再開を開始した時間です         |
| <i>lastResumeCompletedTime</i> ** Required]**_Time_ | LastCompletedTime は、PGDGroup が最後に再開を完了した時間です             |

### 3.18.52 PgDConfiguration

使用される箇所

- *PgDGroupSpec*

PgDConfiguration は、PGD グループ構造の構成です

| Field                                                                       | Description                                                                                                                                                                                                                                |
|-----------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>parentGroup</i> <b>** Required]**</b> [_ParentGroup-Configuration_       | ParentGroup は PGD グループのトポロジを構成します                                                                                                                                                                                                          |
| <i>discovery</i> <b>** Required]**</b> [_[]Connection-String_               | 親 PGD グループに属するノードに接続するために使用するパラメーター。指定した場合でも、次のパラメーターはデフォルト値でオーバーライドされます `application_name`、 <i>sslmode</i> 、 <i>dbname</i> 、および `user`。次のパラメーターは、デフォルトでオーバーライドされないため、提供または使用しないでください。 <i>sslkey</i> 、 <i>sslcert</i> 、 <i>sslrootcert</i> |
| <i>discoveryJob</i> <b>** Required]**</b> [_DiscoveryJobConfig_             | DiscoveryJob PGD Discovery ジョブの構成                                                                                                                                                                                                          |
| <i>databaseName</i> <b>[Required]</b> _string_                              | アプリケーションが使用するデータベースの名前。デフォルト `app`                                                                                                                                                                                                         |
| <i>ownerName</i> <b>[Required]</b> _string_                                 | アプリケーションが使用するインスタンス内のデータベースの所有者の名前。 `database` キーの値のデフォルト。                                                                                                                                                                                 |
| <i>ownerCredentialsSecret</i> <b>** Required]**</b> [_LocalObjectReference_ | ユーザーデータベースの所有者の初期資格情報を含むシークレットの名前。空の場合、新しいシークレットは最初から作成されます                                                                                                                                                                                |
| <i>proxySettings</i> <b>** Required]**</b> [_PGDProxy-Settings_             | プロキシの構成                                                                                                                                                                                                                                    |
| <i>nodeGroupSettings</i> <b>** Required]**</b> [_PGDNodeGroupSettings_      | PGD グループの構成                                                                                                                                                                                                                                |
| <i>globalRouting</i> <b>[Required]</b> _bool_                               | GlobalRouting は、グローバルルーティングが有効になっている場合に true であり、この場合、プロキシは親グループに作成されます                                                                                                                                                                    |
| <i>mutations</i> <b>** Required]**</b> [_SQLMutations_                      | ノードグループに適用する SQL ミューテーションのリスト                                                                                                                                                                                                              |

### 3.18.53 事前プロビジョニングされた証明書

使用される箇所

- *Client* 事前プロビジョニングされた証明書

PreProvisionedCertificate には、事前に生成された証明書を提供するために必要なデータが含まれています

| Field                                              | Description                                  |
|----------------------------------------------------|----------------------------------------------|
| <i>secretRef</i> <b>[Required]</b> <i>_string_</i> | SecretRef tls.crt および tls.key を含むシークレットを指す名前 |

### 3.18.54 メソッド別リカバリーポイント

(map のエイリアス [github.com/EnterpriseDB/cloud-native-postgres/api/v1.BackupMethod]k8s.io/apimachinery/pkg/apis/meta/v1.TypeMeta)

使用される箇所

- *CNP* ステータス

RecoverabilityPointsByMethod には、特定のバックアップ方法の最初の復旧ポイントが含まれます

### 3.18.55 ReplicationCertificateStatus

使用される箇所

- 接続ステータス
- *NodeCertificateStatus*

ReplicationCertificateStatus は証明書ステータスをカプセル化します

| Field                                                 | Description                                    |
|-------------------------------------------------------|------------------------------------------------|
| <i>name</i> <b>[Required]</b> <i>_string_</i>         | Name は証明書の名前です                                 |
| <i>hash</i> <b>[Required]</b> <i>_string_</i>         | ハッシュは、生成された構成のハッシュです                           |
| <i>isReady</i> <b>[Required]</b> <i>_bool_</i>        | 証明書の準備ができている場合、Ready は true です                 |
| <i>preProvisioned</i> <b>[Required]</b> <i>_bool_</i> | 証明書が事前プロビジョニングされている場合、PreProvisioned は true です |

### 3.18.56 リストア

使用される箇所

- *PGDGroupSpec*

Restore は、オブジェクトストアからの PGD グループのリストアを構成します

| Field                                                        | Description            |
|--------------------------------------------------------------|------------------------|
| <i>volumeSnapshots</i> <i>_VolumeSnapshotsConfiguration_</i> | volumeSnapshot リストアの構成 |

次のページに続く

表 52 – 前のページからの続き

| Field                                                                            | Description                                                                                                                                                                                                                                                                                                                                                                                                                      |
|----------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>barmanObjectStore</i> <b>** Required</b> ** [_BarmanObjectStoreConfiguration_ | barman-cloud ツールスイートの構成                                                                                                                                                                                                                                                                                                                                                                                                          |
| <i>recoveryTarget</i> <b>** Bootstrap</b>                                        | デフォルトでは、復旧プロセスは、アーカイブ内の使用可能なすべての WAL ファイルを適用します完全復旧。ただし、一貫した状態に到達するとすぐにリカバリーを終了するか、PostgreSQL で予想されるように `RecoveryTarget` オブジェクトつまりタイムスタンプ、トランザクション ID、LSN、.. を指定することにより、ポイントインタイム PITR にリカバリーすることもできます。.)。詳細 <a href="https://www.postgresql.org/docs/current/runtime-config-wal.html#RUNTIME-CONFIG-WAL-RECOVERY-TARGET">https://www.postgresql.org/docs/current/runtime-config-wal.html#RUNTIME-CONFIG-WAL-RECOVERY-TARGET</a> |
| <i>serverNames</i> <b>[Required]</b> _[]string_                                  | 復旧元として使用されるサーバー名のリスト。リカバリターゲットを評価するときに、これらのサーバーのいずれかがシードサーバーとして選択されます。このオプションは、barmanObjectStore から復元する場合にのみ使用されます。                                                                                                                                                                                                                                                                                                              |

### 3.18.57 RestoreStatus

使用される箇所

- *PGDGroupStatus*

RestoreStatus には、復元プロセスの現在のステータスが含まれています

| Field                                                                         | Description           |
|-------------------------------------------------------------------------------|-----------------------|
| <i>serverName</i> <b>[Required]</b> _string_                                  | 復元するサーバーの名前           |
| <i>VolumeSnapshots</i> <b>** Required</b> ** [_[]VolumeSnapshotRestoreStatus_ | 選択したボリューム復元するスナップショット |

### 3.18.58 RootDNSConfiguration

使用される箇所

- *ConnectivityConfiguration*

RootDNSConfiguration は、リソースの FQDN を生成する方法を説明します

| Field                                                                    | Description                                                     |
|--------------------------------------------------------------------------|-----------------------------------------------------------------|
| <i>DNSConfiguration</i> _DNSConfiguration_                               | <code>DNSConfiguration`</code> のメンバーはこのタイプに埋め込まれます。説明はありません。    |
| <i>additional</i> <b>** Required</b> ** <code>[]DNSConfiguration_</code> | <code>AdditionalDNSConfigurs</code> は、リソースに使用可能な FQDN をさらに追加します |

### 3.18.59 SQL ミューテーション

SQLMutation は、アトミックに適用する一連の SQL ステートメントです

| Field                                                     | Description                                                                                                                                        |
|-----------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>isApplied</i> <b>[Required]</b> <code>[]string_</code> | ブール値を返す SQL クエリーのリスト。それらのいずれかが <code>false</code> を返す場合、ミューテーションが適用されます                                                                            |
| <i>exec</i> <b>[Required]</b> <code>[]string_</code>      | この変更を適用するために実行される SQL クエリのリスト                                                                                                                      |
| <i>type</i> <code>_SQLMutationType_</code>                | Type は、SQL ミューテーションが発生する時期を決定します。'always' 各調整サイクルでミューテーションを調整します 'beforeSubgroupRaft' サブグループラフトが有効になる前にのみ実行されます 指定しない場合、タイプはデフォルトの 'always' になります。 |

### 3.18.60 SQLMutationType

(文字列のエイリアス)

使用される箇所

- *SQLMutations*

SQLMutationType は、サポートされている SQL ミューテーションのタイプです

### 3.18.61 SQL ミューテーション

`[]github.com/EnterpriseDB/pg4k-pgd/api/v1beta1.SQLMutation` のエイリアス

使用される箇所

- *PgdConfiguration*

SQLMutation SQLMutation のリスト

### 3.18.62 ScheduledBackupSpec

使用される箇所

- *Backup on object stores*

ScheduledBackupSpec は、 ScheduledBackup の目的の状態を定義します

| Field                                                               | Description                                                                                                                                                                                                                                                                                   |
|---------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>suspend</i> <b>[Required]</b> <code>_bool_</code>                | このバックアップが一時停止されているかどうか                                                                                                                                                                                                                                                                        |
| <i>immediate</i> <b>[Required]</b> <code>_bool_</code>              | 最初のバックアップを作成後にすぐに開始する必要があるかどうか                                                                                                                                                                                                                                                                |
| <i>schedule</i> <b>[Required]</b> <code>_string_</code>             | スケジュールは、追加の 2 番目の指定子が含まれるため、Kubernetes Cron ジョブで使用される形式に従いません。参照してください。 <a href="https://pkg.go.dev/github.com/robfig/cron#hdr-CRON_Expression_Format">https://pkg.go.dev/github.com/robfig/cron#hdr-CRON_Expression_Format</a>                                                               |
| <i>backupOwnerReference</i> <b>[Required]</b> <code>_string_</code> | 作成したバックアップリソース内にどの ownerReference を配置するかを示します。 <code>_none</code> 作成されたバックアップオブジェクトの所有者参照なしフィールドが導入される前と同じ動作 <code>_self</code> スケジュールされたバックアップオブジェクトをバックアップの所有者として設定します <code>*cluster</code> クラスタをバックアップの所有者として設定します                                                                        |
| <i>target</i> <b>** Bootstrap</b>                                   | このバックアップを実行するインスタンスを決定するポリシー。空の場合、デフォルトの <code>`cluster.spec.backup.target`</code> になります。使用可能なオプションは、空の文字列、 <code>`primary`</code> および <code>`prefer-standby`</code> です。 <code>`primary`</code> は常にプライマリインスタンスでバックアップを実行します。 <code>`prefer-standby`</code> は、可能な場合は最も更新されたスタンバイでバックアップを実行します。 |
| <i>method</i> <b>Bootstrap</b>                                      | 使用するバックアップ方法、可能なオプションは <code>`barmanObjectStore`</code> および <code>`volumeSnapshot`</code> です。デフォルト <code>`barmanObjectStore`</code>                                                                                                                                                           |
| <i>online</i> <code>_bool_</code>                                   | ボリュームスナップショットを使用したバックアップのデフォルトのタイプがオンライン/ホット <code>true</code> 、default またはオフライン/コールド <code>false</code> かどうか クラスタフィールド <code>'.spec.backup.volumeSnapshot.online'</code> で指定されたデフォルト設定をオーバーライドします                                                                                            |
| <i>onlineConfiguration</i> <code>_OnlineConfiguration_</code>       | ボリュームスナップショットを使用してオンライン/ホットバックアップを制御する構成パラメーター クラスタで指定されたデフォルト設定をオーバーライドします <code>'.backup.volumeSnapshot.onlineConfiguration'</code> スタンザ                                                                                                                                                    |

### 3.18.63 ScheduledBackupStatus

使用される箇所

- *BackupStatus*

ScheduledBackupStatus には、スケジュールされたバックアップの状態が含まれます

| Field                             | Description                            |
|-----------------------------------|----------------------------------------|
| <i>method</i> ** <i>Bootstrap</i> | Method は、バックアップを取得するために使用されるバックアップ方法です |
| <i>name</i> [Required] _string_   | Name はスケジュールされたバックアップの名前です             |
| <i>hash</i> [Required] _string_   | Hash は、スケジュールされたバックアップ構成のハッシュです        |

### 3.18.64 ServerCertConfiguration

使用される箇所

- *OTELTLS* 構成

ServerCertConfiguration には、ノードの証明書を生成するための情報が含まれています

| Field                                                     | Description                                        |
|-----------------------------------------------------------|----------------------------------------------------|
| <i>caCertSecret</i> [Required] _string_                   | CACertSecret は、CloudNativePG 構成に注入される CA のシークレットです |
| <i>certManager</i> ** [Required]** [_CertManagerTemplate_ | 証明書の生成に使用される cert-manager テンプレート                   |

### 3.18.65 ServiceTemplate

使用される箇所

- *ConnectivityConfiguration*

ServiceTemplate は、ユーザーが Service 生成のテンプレートを設定できる構造です。

| Field                      | Description                                                                                                                                                                                                                 |
|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>metadata</i> _Metadata_ | 標準オブジェクトのメタデータ。詳細情報 <a href="https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata">https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata</a> |

次のページに続く

表 59 – 前のページからの続き

| Field                                         | Description                                                                                                                                                                                                                              |
|-----------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>spec</i> _ServiceSpec_                     | サービスの目的な動作の仕様。詳細情報 <a href="https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status">https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status</a> |
| <i>updateStrategy</i> _ServiceUpdateStrategy_ | UpdateStrategy は、このテンプレートによって生成されたサービスを更新する方法を示します。                                                                                                                                                                                      |

### 3.18.66 ServiceUpdate 戦略

(文字列のエイリアス)

使用される箇所

- *ServiceTemplate*

ServiceUpdateStrategy は、ロードバランサーを更新するためのタイプを定義します。使用できる値は「patch」および「replace」です。

### 3.18.67 TLS 構成

使用される箇所

- *ConnectivityConfiguration*

TLSConfiguration は、PGD がノードに接続するために使用する TLS インフラストラクチャの構成です

| Field                                                             | Description  |
|-------------------------------------------------------------------|--------------|
| <i>mode</i> ** Required]*** [_TLSMode_                            | 説明はありません。    |
| <i>serverCert</i> ** Required]*** [_ServerCertCon-<br>figuration_ | サーバー証明書の構成   |
| <i>clientCert</i> ** Required]*** [_ClientCertCon-<br>figuration_ | クライアント証明書の構成 |

### 3.18.68 TLS モード

(文字列のエイリアス)

使用される箇所

- *OTELTLS* 構成

TLSMode は、ノード間の通信にどのモードを使用するかを説明します

3.18.69 VolumeSnapshotRestoreStatus

使用される箇所

- RestoreStatus

VolumeSnapshotRestoreStatus 復元する volumeSnapshot

| Field                                                              | Description                                                          |
|--------------------------------------------------------------------|----------------------------------------------------------------------|
| snapshotName [Required] _string_                                   | SnapshotName は、復元するスナップショットの名前です。                                    |
| pvcRole ** [Required]**                                            | PVCRole は、復元する pvcRole スナップショットです                                    |
| [_github.com/EnterpriseDB/cloud-native-postgres/pkg/utils.PVCRole_ |                                                                      |
| tablespaceName [Required] _string_                                 | TablespaceName は、pvcRole が`PG_TABLESPACE`の場合、スナップショットが属するテーブルスペース名です |

3.18.70 VolumeSnapshots 構成

使用される箇所

- リストア

VolumeSnapshotsConfiguration には、volumeSnapshots リストアの構成が含まれます

| Field                                     | Description                        |
|-------------------------------------------|------------------------------------|
| selector ** [Required]** [_LabelSelector_ | 復元するボリュームスナップショットの選択に使用されるラベルセレクター |

3.19 Supported versions

このページには、Kubernetes の EDB Postgres Distributed の現在サポートされているリリースのステータスをリストしています。

3.19.1 Kubernetes リリースの EDB Postgres 分散のサポートステータス

| Version | Currently Supported | Release Date   | End of Life     | Supported Kubernetes Versions | Supported PG4K V  |
|---------|---------------------|----------------|-----------------|-------------------------------|-------------------|
| 1.0.x   | Yes                 | April 24, 2024 | ~ July 24, 2025 | v1.28 - v1.31                 | v1.22.2 - v1.22.6 |

Postgres オペランドバージョンは、でサポートされているものに制限されています。

**重要:** このページは情報提供のみを目的としています。 EDB ウェブサイトの [Platform Compatibility](#) ページには、サポートされているソフトウェアと Kubernetes ディストリビューションの公式リストが含まれています。

---

### 3.19.2 EDB Postgres Distributed for Kubernetes リリースバージョンフォーマット

EDB Postgres Distributed for Kubernetes PGD4K は、EDB Postgres for Kubernetes PG4K オペレーターを活用して、各 PGD ノードを PG4K クラスターとして管理します。PGD4K の minor バージョンは、PG4K LTS リリースを追跡しています。PGD4K は、追跡している PG4K LTS リリースと同じバージョンのオペランドをサポートしています。

## 3.20 Known issues and limitations

これらの既知の問題と制限は、Kubernetes の EDB Postgres Distributed の現在のリリースに含まれています。

### 3.20.1 Postgres メジャーバージョンのアップグレード

Kubernetes の EDB Postgres Distributed のこのバージョンは、Postgres のメジャーバージョンのアップグレードをサポートしていません\*\*。

### 3.20.2 データの移行

EDB Postgres Distributed for Kubernetes のこのバージョンは、既存の Postgres データベースからの移行をサポートしていません\*\*。

### 3.20.3 PgBouncer との接続

Kubernetes 用 EDB Postgres 分散は、

[PgBouncer](#) を使用してクライアント接続要求をプールすることを

サポートしていません。この制限は、PgBouncer のオープンソースと EDB バージョンの両方に適用されます。

### 3.20.4 バックアップ操作

Kubernetes 環境の EDB Postgres 分散を構成するには、PGDGroup YAML オブジェクトを各 Kubernetes クラスターに適用する必要があります。このオブジェクトを適用すると、分散アーキテクチャを実装するために必要なすべてのサービスが作成されます。

バックアップ構成をセットアップする目的で `spec.backup` セクションをこの PGDGroup オブジェクトに追加した場合、`spec.backup.cron.schedule` 値も設定しない限り、バックアップは失敗します。

エラー出力例

```
The PGDGroup "region-a" is invalid: spec.backup.cron.schedule: Invalid value: "":  
→ Empty spec string
```

#### 回避策

この問題をワークアラウンドするには、要件を満たすスケジュールを使用して `spec.backup.cron.schedule` セクションを追加します。例

```
spec:
  instances: 3
  proxyInstances: 2
  pgd:
    parentGroup:
      create: true
      name: world
  backup:
    configuration:
      barmanObjectStore:
        ...
  cron:
    suspend: false
    immediate: true
    schedule: "@  */5*  **  *"
```

### 3.20.5 EDB Postgres 分散の既知の問題と制限

展開に含める EDB Postgres 分散バージョンの既知の問題と制限はすべて、EDB Postgres Distributed for Kubernetes インスタンスに影響を与えます。

たとえば、使用している EDB Postgres 分散バージョンが最新の場合、EDB Postgres Distributed for Kubernetes インスタンスは、これらの *latest known issues* および *limitations* の影響を受けます。