



**EDB**

Postgres® for the AI Generation

## **Barman Manual**

リリース **3.11.1**

**EnterpriseDB Corporation.**

2024 年 10 月 08 日



# 目次

|       |                                                                            |    |
|-------|----------------------------------------------------------------------------|----|
| 第 1 章 | はじめに                                                                       | 3  |
| 第 2 章 | 始める前に                                                                      | 5  |
| 第 3 章 | デザインとアーキテクチャ                                                               | 7  |
| 3.1   | Barman をインストールする場所                                                         | 7  |
| 3.2   | 1 つの Barman、多数の PostgreSQL サーバー                                            | 8  |
| 3.3   | ストリーミングバックアップと rsync / SSH                                                 | 9  |
| 3.4   | Barman WAL アーカイブ                                                           | 9  |
| 3.5   | バックアップの 2 つの一般的なシナリオ                                                       | 10 |
| 3.5.1 | シナリオ 1 ストリーミングプロトコルを介したバックアップ                                              | 10 |
| 3.5.2 | シナリオ 2 rsync / SSH を介したバックアップ                                              | 11 |
| 第 4 章 | システム要件                                                                     | 13 |
| 4.1   | バックアップの要件                                                                  | 13 |
| 4.2   | リカバリの要件                                                                    | 14 |
| 第 5 章 | インストール                                                                     | 15 |
| 5.1   | RPM パッケージを使用した Red Hat Enterprise Linux RHEL および RHEL ベースのシステム<br>へのインストール | 15 |
| 5.2   | パッケージを使用した Debian/Ubuntu へのインストール                                          | 16 |
| 5.3   | パッケージを使用した SLES へのインストール                                                   | 16 |
| 5.4   | ソースからのインストール                                                               | 17 |
| 5.5   | PostgreSQL クライアント/サーバーバイナリ                                                 | 17 |
| 5.5.1 | サードパーティの PostgreSQL バリエーション                                                | 18 |
| 第 6 章 | Barman のアップグレード                                                            | 19 |
| 6.1   | Barman 3.0.0 へのアップグレード                                                     | 19 |
| 6.1.1 | Rsync バックアップのデフォルトのバックアップアプローチがコンカレントになり<br>ました                            | 19 |
| 6.1.2 | メタデータの変更                                                                   | 19 |
| 6.2   | Barman 2.10 からのアップグレード                                                     | 20 |
| 6.3   | Barman 2.X 2.8 より前からのアップグレード                                               | 20 |
| 6.4   | Barman 1.X からのアップグレード                                                      | 20 |
| 第 7 章 | 構成                                                                         | 21 |
| 7.1   | オプションスコープ                                                                  | 22 |
| 7.2   | 構成の例                                                                       | 23 |
| 第 8 章 | Barman での新しいサーバーのセットアップ                                                    | 25 |
| 第 9 章 | 準備的な手順                                                                     | 27 |

|                                              |                                                 |           |
|----------------------------------------------|-------------------------------------------------|-----------|
| 9.1                                          | PostgreSQL 接続                                   | 27        |
| 9.2                                          | PostgreSQL WAL のアーカイブとレプリケーション                  | 28        |
| 9.3                                          | PostgreSQL ストリーミング接続                            | 29        |
| 9.4                                          | SSH 接続                                          | 30        |
| 9.5                                          | postgres ユーザーの SSH 構成                           | 30        |
| 9.6                                          | barman ユーザーの SSH 構成                             | 30        |
| 9.7                                          | PostgreSQL から Barman へ                          | 31        |
| 9.8                                          | Barman から PostgreSQL へ                          | 31        |
| <b>第 10 章 サーバー構成ファイル</b>                     |                                                 | <b>33</b> |
| <b>第 11 章 WAL ストリーミング</b>                    |                                                 | <b>35</b> |
| 11.1                                         | レプリケーションスロット                                    | 36        |
| 11.2                                         | WAL ストリーミングを構成する方法                              | 36        |
| 11.3                                         | さまざまなホストからのストリーミング WAL とバックアップ Barman 3.10.0 以降 | 37        |
| 11.4                                         | リカバリーを含む部分的な WAL ファイルの制限                        | 38        |
| <b>第 12 章 archive_command を介した WAL アーカイブ</b> |                                                 | <b>39</b> |
| 12.1                                         | barman-wal-archive を介した WAL アーカイブ               | 39        |
| 12.2                                         | rsync / SSH を介した WAL アーカイブ                      | 40        |
| <b>第 13 章 WAL アーカイブ構成の検証</b>                 |                                                 | <b>43</b> |
| <b>第 14 章 ストリーミングバックアップ</b>                  |                                                 | <b>45</b> |
| <b>第 15 章 rsync / SSH によるバックアップ</b>          |                                                 | <b>47</b> |
| <b>第 16 章 クラウドスナップショットを使用したバックアップ</b>        |                                                 | <b>49</b> |
| 16.1                                         | クラウドスナップショットの前提条件                               | 49        |
| 16.2                                         | Google Cloud Platform スナップショットの前提条件             | 49        |
| 16.3                                         | Azure スナップショットの前提条件                             | 50        |
| 16.4                                         | AWS スナップショットの前提条件                               | 51        |
| 16.5                                         | スナップショットバックアップの構成                               | 51        |
| 16.6                                         | Google Cloud Platform スナップショットの構成               | 52        |
| 16.7                                         | Azure スナップショットの構成                               | 52        |
| 16.8                                         | AWS スナップショットの構成                                 | 53        |
| 16.9                                         | スナップショットバックアップの取得                               | 53        |
| <b>第 17 章 Windows ベースのサーバーをセットアップする方法</b>    |                                                 | <b>55</b> |
| <b>第 18 章 一般的なコマンド</b>                       |                                                 | <b>57</b> |
| 18.1                                         | cron                                            | 57        |
| 18.2                                         | 診断する                                            | 58        |
| 18.3                                         | list-servers                                    | 58        |
| <b>第 19 章 Server コマンド</b>                    |                                                 | <b>59</b> |
| 19.1                                         | archive-wal                                     | 59        |
| 19.2                                         | バックアップ                                          | 59        |
| 19.3                                         | check                                           | 60        |
| 19.4                                         | config-update                                   | 60        |

|               |                               |           |
|---------------|-------------------------------|-----------|
| 19.5          | config-switch                 | 61        |
| 19.6          | generate-manifest             | 61        |
| 19.7          | get-wal                       | 62        |
| 19.8          | list-backups                  | 63        |
| 19.9          | resume-xlogdb                 | 64        |
| 19.10         | resume-wal                    | 64        |
| 19.10.1       | レシーブウォールプロセス管理                | 64        |
| 19.10.2       | レプリケーションスロット管理                | 65        |
| 19.11         | レプリケーションステータス                 | 65        |
| 19.12         | show-servers                  | 65        |
| 19.13         | ステータス                         | 65        |
| 19.14         | switch-wal                    | 66        |
| 19.15         | verify-backup                 | 66        |
| <b>第 20 章</b> | <b>バックアップコマンド</b>             | <b>67</b> |
| 20.1          | バックアップ ID のショートカット            | 67        |
| 20.2          | check-backup                  | 67        |
| 20.3          | 削除                            | 68        |
| 20.4          | keep                          | 68        |
| 20.5          | list-files                    | 69        |
| 20.6          | recovery                      | 70        |
| 20.6.1        | リモートリカバリー                     | 71        |
| 20.6.2        | テーブルスペースのリマッピング               | 71        |
| 20.6.3        | ポイント インタイム リカバリー              | 72        |
| 20.6.4        | Barman サーバーから WAL の取得         | 73        |
| 20.6.5        | 圧縮バックアップのリカバリー                | 74        |
| 20.6.6        | ブロックレベルのインクリメンタル バックアップのリカバリー | 74        |
| 20.7          | show-backup                   | 75        |
| <b>第 21 章</b> | <b>機能の詳細</b>                  | <b>77</b> |
| 21.1          | バックアップ機能                      | 77        |
| 21.1.1        | インクリメンタル バックアップ               | 77        |
| 21.1.2        | ファイルレベルのインクリメンタル バックアップ       | 77        |
| 21.1.3        | ブロックレベルのインクリメンタル バックアップ       | 79        |
| 21.1.4        | 帯域幅の使用を制限する                   | 79        |
| 21.1.5        | ネットワーク圧縮                      | 80        |
| 21.1.6        | バックアップ圧縮                      | 80        |
| 21.1.7        | 圧縮アルゴリズム                      | 80        |
| 21.1.8        | 圧縮ワーカー                        | 81        |
| 21.1.9        | 圧縮レベル                         | 81        |
| 21.1.10       | 圧縮場所                          | 82        |
| 21.1.11       | 圧縮形式                          | 82        |
| 21.1.12       | 同時バックアップ                      | 83        |
| 21.1.13       | スタンバイの同時バックアップ                | 84        |
| 21.1.14       | 即時チェックポイント                    | 85        |
| 21.1.15       | ローカルバックアップ                    | 85        |
| 21.2          | アーカイブ機能                       | 86        |

|                                                            |                                   |            |
|------------------------------------------------------------|-----------------------------------|------------|
| 21.2.1                                                     | WAL 圧縮                            | 86         |
| 21.2.2                                                     | 同期 WAL ストリーミング                    | 86         |
| 21.3                                                       | カタログ管理機能                          | 87         |
| 21.3.1                                                     | 最小限の冗長性の安全性                       | 87         |
| 21.3.2                                                     | リテンションポリシー                        | 88         |
| 21.3.3                                                     | スコープ                              | 89         |
| 21.3.4                                                     | 仕組み                               | 89         |
| 21.3.5                                                     | 構成と構文                             | 90         |
| 21.4                                                       | フックスクリプト                          | 91         |
| 21.4.1                                                     | バックアップスクリプト                       | 92         |
| 21.4.2                                                     | バックアップ削除スクリプト                     | 92         |
| 21.4.3                                                     | WAL アーカイブスクリプト                    | 93         |
| 21.4.4                                                     | WAL 削除スクリプト                       | 94         |
| 21.4.5                                                     | リカバリスクリプト                         | 94         |
| 21.5                                                       | カスタマイズ                            | 95         |
| 21.5.1                                                     | ロックファイルディレクトリ                     | 95         |
| 21.5.2                                                     | バイナリパス                            | 95         |
| 21.6                                                       | クラスター管理システムとの統合                   | 95         |
| 21.6.1                                                     | 構成モデル                             | 96         |
| 21.7                                                       | パラレルジョブ                           | 97         |
| 21.7.1                                                     | パラレルジョブと sshd MaxStartups         | 97         |
| 21.8                                                       | 地理的な冗長性                           | 98         |
| 21.8.1                                                     | 情報の同期                             | 98         |
| 21.8.2                                                     | 構成                                | 99         |
| 21.8.3                                                     | ノードの同期                            | 99         |
| 21.8.4                                                     | 手動同期                              | 99         |
| 21.9                                                       | クラウドスナップショットバックアップ                | 100        |
| 21.9.1                                                     | スナップショットバックアップの詳細                 | 100        |
| 21.9.2                                                     | スナップショットバックアップからのリカバリ             | 101        |
| 21.9.3                                                     | スナップショットバックアップのバックアップメタデータ        | 102        |
| 21.9.4                                                     | GCP プロバイダー固有のメタデータ                | 104        |
| 21.9.5                                                     | Azure プロバイダー固有のメタデータ              | 104        |
| 21.9.6                                                     | AWS プロバイダー固有のメタデータ                | 105        |
| <b>第 22 章 Barman クライアントユーティリティ barman-cli</b>              |                                   | <b>107</b> |
| 22.1                                                       | インストール                            | 107        |
| <b>第 23 章 クラウド用の Barman クライアントユーティリティ barman-cli-cloud</b> |                                   | <b>109</b> |
| 23.1                                                       | インストール                            | 110        |
| 23.2                                                       | barman-cloud フックスクリプト             | 110        |
| 23.3                                                       | クラウドプロバイダーの選択                     | 111        |
| 23.4                                                       | プロバイダーによる特異性                      | 111        |
| 23.4.1                                                     | Google Cloud Storage              | 111        |
| 23.4.2                                                     | セットアップ                            | 111        |
| 23.4.3                                                     | 使用法                               | 112        |
| 23.5                                                       | barman-cloud とスナップショットのバックアップ     | 112        |
| 23.5.1                                                     | barman-cloud-restore for スナップショット | 113        |

|                                   |            |
|-----------------------------------|------------|
| <b>第 24 章 障害対応</b>                | <b>115</b> |
| 24.1 Barman インストールを診断する . . . . . | 115        |
| 24.2 ヘルプの要求 . . . . .             | 115        |
| 24.2.1 バグの送信 . . . . .            | 115        |
| <b>第 25 章 Barman プロジェクト</b>       | <b>117</b> |
| 25.1 サポートとスポンサーの機会 . . . . .      | 117        |
| 25.2 Barman への貢献 . . . . .        | 117        |
| 25.3 著者 . . . . .                 | 118        |
| 25.4 リンク . . . . .                | 119        |
| 25.5 ライセンスと貢献 . . . . .           | 119        |
| <b>第 26 章 参考リンク</b>               | <b>121</b> |



% Barman Manual % EnterpriseDB UK Limited % 2024 年 8 月 22 日 3.11.1

**Barman** Backup and Recovery Manager は、Python で書かれた PostgreSQL サーバーのディザスターリカバリーのためのオープンソース管理ツールです。これにより、組織はビジネスクリティカルな環境で複数のサーバーのリモートバックアップを実行して、リスクを削減し、復旧フェーズ中に DBA を支援できます。

[Barman][11] は GNU GPL 3 の下で配布され、[PostgreSQL project][31] のプラチナスポンサーである [EnterpriseDB][13] によって維持されています。

---

**重要:** このマニュアルは、理論的なディザスターリカバリーの概念に精通しており、物理的バックアップとディザスターリカバリーの観点から PostgreSQL の基礎を理解していることを前提としています。詳細については、以下のセクション\_「始める前に」\_を参照してください。

---



# 第1章 はじめに

完璧な世界では、バックアップは必要ありません。ただし、特にビジネス環境では、“予期せぬ”が発生した場合に備えることが重要です。データベースシナリオでは、予期せぬものは次のいずれかの形式を取る可能性があります。

- データの破損
- システム障害ハードウェア障害を含む
- 人的エラー
- 自然災害

このような場合、ICT マネージャーまたは DBA は、可能な限り最短の時間でインシデントを修正し、データベースを復旧する必要があります。私たちは通常、この分野をディザスタリカバリー、より広範には **ビジネス継続性**と呼んでいます。

ビジネス継続性の範囲では、Wikipedia で定義されている 2 つの基本的なメトリックをよく理解することが重要です。

- [ **Recovery Point Objective(RPO)** ][rpo] “重大なインシデントにより IT サービスからデータが失われる可能性のある最大目標期間”
- [ **Recovery Time Objective (RTO)** ][rto] 「災害または中断後に、関連する受け入れがたい結果を回避するためにビジネスプロセスを復元する必要がある目標期間とサービスレベル」事業継続の中断とともに”

一言で言えば、RPO は損失してもよいデータの最大量を表し、RTO はサービスに許容できる最大ダウンタイムを表します。

当然のことながら、私たち全員が **RPO=0** 「ゼロデータ損失」と **RTO=0** ゼロダウンタイム\*、理想郷を望んでいます。たとえそれが祖母のレシピウェブサイトであったとしても。実際には、注意深いコスト分析フェーズにより、ビジネス継続性の要件を決定できます。

幸いなことに、**Barman** と **PostgreSQL** で構成されるオープンソーススタックを使用すると、同期ストリーミングレプリケーションのおかげで RPO=0 を達成できます。RTO は、[ **repmgr** ][repmgr] のような **高可用性**ソリューションに焦点を当てています。したがって、Barman と repmgr を統合することにより、RTO をほぼゼロに劇的に削減できます。

EnterpriseDB での経験に基づいて、Barman および repmgr を使用した PostgreSQL オープンソースクラスターは、適切に構成および監視されている場合、1 年間で 99.99%を超えるアップタイムを簡単に達成できることが確認できます。

いずれにせよ、実際のツールではなく、ディザスタリカバリーに関連する文化的な側面をより強調することが重要です。人間のいない道具は役に立ちません。

Barman との私たちの使命は、次のようなディザスターリカバリーの文化を促進することです。

- バックアップ手順に焦点を当てます
- 復旧手順にさらに焦点を当てます
- チームメンバーのための PostgreSQL のクラッシュリカバリー、バックアップ、ポイントインタイムリカバリー、およびレプリケーションの強力な理論的および実践的な概念に関する教育とトレーニングに依存しています。
- 手動または自動で、バックアップのテストを促進しますテストされたバックアップのみが有効であると考えられます。Barman のフックスクリプトを創造してください。
- Devops チームのすべてのメンバーによる、リカバリ手順の定期的な練習を促進しますはい、システム管理者と DBA だけでなく開発者もです
- 3〜6 か月ごとにチームとの定期的にスケジュールされた訓練とディザスターリカバリーシミュレーションを呼びかけます。
- PostgreSQL と Barman の継続的なモニタリングに依存し、異常を迅速に特定できます

さらに、災害が発生した場合に自分自身とあなたのチームを準備するためにできる限りのことを行ってくださいはい、**とき**、なぜなら、それが起こったとき

- それは金曜日の夕方になり、おそらくあなたがオフィスを出ようとしているときです。
- それは、あなたが休暇中で世界一周クルーズの真っ最中で、他の誰かがそれに対処しなければならないときです。
- それは確かにストレスになるでしょう。
- 最後に利用可能なバックアップが有効であることを確認しないと後悔するでしょう。
- 回復するまでにおよそどれくらいの時間がかかるかを知らなければ、一秒一秒が永遠のように見えるでしょう。

準備をしてください、怖がらないでください。

2011 年、これらの目標を念頭に置いて、2ndQuadrant は現在 PostgreSQL で最も使用されているバックアップツールの 1 つである Barman の開発を開始しました。Barman は、「Backup and Recovery Manager」の頭字語です。

現在、Barman は Linux および Unix オペレーティングシステムでのみ動作します。

## 第2章 始める前に

Barman を使用し始める前に、PostgreSQL と、ベースバックアップ、WAL アーカイブなどの物理バックアップ、ポイントインタイムリカバリー、レプリケーション周辺の概念に慣れることが基本です。

以下に、読むことをお勧めするリソースの非完全なリストを示します。

- *PostgreSQL documentation*:
- [SQL Dump][sqldump][^pgdump]
- [File System Level Backup][physicalbackup]
- [Continuous Archiving and Point-in-Time Recovery PITR][pitr]
- [信頼性と書き込み-Ahead Log][wal]
- *Book* [PostgreSQL 10 Administration Cookbook][adminbook]

[^pgdump] 論理バックアップと物理バックアップの違い、したがって、`pg_dump` と Barman のようなツールの違いを知ることが重要です。

これらのトピックに関する専門的なトレーニングは、これらの概念を学習する別の効果的な方法です。一年中いつでも、EnterpriseDB などの PostgreSQL 企業が提供する世界中で利用可能な多くのコースを見つけることができます。



## 第3章 デザインとアーキテクチャ

### 3.1 Barman をインストールする場所

Barman の基礎の 1 つは、ネットワークを介してデータベースサーバーからリモートで操作できる機能です。

理論的には、Barman サーバーを PostgreSQL サーバーから数千マイル離れた世界の別の地域のデータセンターに配置することができます。現実的には、バックアップとリカバリー時間の両方を制御できるように、Barman サーバーを PostgreSQL サーバーから遠すぎないようにします。

Barman をセットアップするための\_“フリーサイズ”\_方法はありませんが、特に従うことをお勧めする推奨事項がいくつかあります。

- 専用サーバーに Barman をインストールする
- PostgreSQL サーバーと同じストレージを共有しないでください
- Barman をモニタリングインフラストラクチャと統合します [^nagios]
- 運用環境に展開する前にすべてをテスト

[^nagios] Nagios/Icinga との統合は、Barman の最も重要な機能の 1 つであり真の救世主である `barman check --nagios` コマンドのおかげで簡単です。

ディザスタリカバリーアーキテクチャのモデリングを開始する合理的な方法は、次のことです。

- PostgreSQL と Barman に関して、いくつかの可能なアーキテクチャをデザインします。
  1. 同じデータセンター re
  2. 同じ首都圏の異なるデータセンター
  3. 異なるデータセンター
- 各仮説の長所と短所を詳しく説明します
- 費用対効果分析を使用して、システムの単一障害点 SPOF を評価します
- 決定を下し、最初のソリューションを実装します

そうは言っても、Barman の非常に一般的なセットアップは、PostgreSQL サーバーが存在するのと同じデータセンターにインストールされることです。この場合、単一障害点はデータセンターです。幸いなことに、このような SPOF の影響は、バックアップ層の数を増やすために Barman が提供する 2 つの機能のおかげで、緩和できます。

1. 地理的冗長性 Barman 2.6 で導入
2. フックスクリプト

地理的冗長性\_を使用すると、別のデータセンター/アベイラビリティゾーンにある *Barman* インスタンスを依存して、ソース *Barman* サーバーのコンテンツ全体を同期できます。さらにあります *Barman* の\_ハイブリッドインストール\_を作成できます。さまざまな *Barman* インストールの\_クロスサイトバックアップ\_。以下の図:raw-latex: `ref{georedundancy-design}` は、2つのアベイラビリティゾーン 1つはヨーロッパに、もう 1つは米国にあり、それぞれにプライマリ PostgreSQL サーバーがあり、ローカルの *Barman* インストールでバックアップされ、他の *Barman* サーバーに中継されます\_passive\_)、rsync / SSH を介した多層バックアップ用。地理的冗長性の詳細については、特定のセクションで入手できます。

## Backup architecture for PostgreSQL

### Example of geographical redundancy

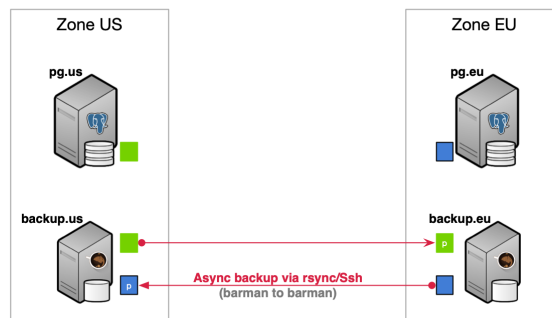


図 1: An example of architecture with geo-redundancy

代わりに\_hook scripts\_のおかげで、*Barman* のバックアップを *tar* を介した\_tape\_などのさまざまなメディア、または Amazon クラウドの\_S3 バケット\_のような場所にエクスポートできます。

どのような決定も永遠ではないことに注意してください。この方法で開始し、時間の経過とともに自分に最適なソリューションに適応できます。ただし、最初はシンプルにするようにしてください。

## 3.2 1つの *Barman*、多数の PostgreSQL サーバー

*Barman* によって最初に導入されたもう 1つの関連機能は、マルチプルサーバーのサポートです。*Barman* は、バージョンが異なる場合でも、複数の PostgreSQL インスタンスからのバックアップデータを一元化した方法で保存できます。[^recver]

[^recver] \_「リカバリの要件」\_セクションで詳細に説明しているように、同じ [PostgreSQL の PITR の要件][requirements\_recovery] がリカバリに適用されます。

その結果、複雑なディザスターリカバリーアーキテクチャをモデル化し、PostgreSQL サーバーが中央の *Barman* サーバーを中心にローテーションする「スタースキーマ」を形成できます。

すべてのアーキテクチャは、独自の方法で意味があります。実際の実験とテストに基づいて、自分に共鳴するもの、そして最も重要なことに、信頼できるものを選択してください。

これ以降、簡単にするために、このガイドでは基本的なアーキテクチャを前提とします。

- 1つの PostgreSQL インスタンスホスト名 *pg*

- Barman ホスト名 `backup` を使用した 1 つのバックアップサーバー

### 3.3 ストリーミングバックアップと `rsync` / SSH

Barman は、トランスポートメカニズムとして SSH を使用する `Rsync`、または PostgreSQL のストリーミングレプリケーションプロトコルを使用する `pg_basebackup` を使用してバックアップを取得できます。

これら 2 つの方法のいずれかを選択することは、決定する必要がありますが、一般的な使用のために、現在サポートされている PostgreSQL のすべてのバージョンでストリーミングレプリケーションを使用することをお勧めします。

`pg_basebackup` の制限が問題を引き起こす場合、`rsync` / SSH を使用したバックアップをお勧めします。

ストリーミングバックアップをお勧めする理由は、私たちの経験から、従来のバックアップよりもセットアップが簡単であるためです。また、ストリーミングバックアップでは、Windows[^windows] で PostgreSQL サーバーをバックアップでき、Docker を使用する際に作業が簡単になります。

[^windows] Windows での PostgreSQL サーバーのバックアップは可能ですが、継続的インテグレーションシステムの一部ではないため、まだ実験的です。詳細については、セクション “Windows ベースのサーバーをセットアップする方法” を参照してください。

### 3.4 Barman WAL アーカイブ

PostgreSQL バックアップの復旧は、トランザクションログ `_xlog_` または WAL ファイルとしても知られるを再生することに依存しています。したがって、WAL ファイルはベースバックアップとともに Barman によって保存され、復旧時に使用できるようにすることが重要です。これは、WAL ストリーミングまたは標準の WAL アーカイブを使用して、WAL を Barman の WAL アーカイブにコピーすることで実現できます。

WAL ストリーミングには、レプリケーションスロットを使用して `pg_receivewal` を使用して PostgreSQL サーバーから WAL ファイルをストリーミングすることが含まれます。WAL ストリーミングは、データ損失のリスクを削減し、RPO を `_近い_ゼロ_値` に下げることができます。PostgreSQL クラスターの同期 WAL レシーバーとして Barman を追加し、**ゼロデータ損失 RPO=0** を達成することもできます。

Barman は、PostgreSQL の `archive_command` `rsync` / SSH または `barman-cli` パッケージの `barman-wal-archive` 経由のいずれかを使用して実現される標準の WAL ファイルアーカイブもサポートしています。この方法では、PostgreSQL が新しい WAL ファイルに `_switches_` した場合にのみ WAL ファイルがアーカイブされます。簡単にするために、これは通常 16MB 相当のデータが変更されるごとに発生します。

WAL ストリーミングまたは WAL アーカイブのいずれかが構成されていることが**必須**です。オプションで WAL ストリーミングと標準の WAL アーカイブの両方を構成することが可能です。そのような場合、Barman は着信 WAL を自動的に重複排除します。これはフォールバックメカニズムを提供し、WAL ストリーミングが失敗した場合にも WAL は Barman のアーカイブにコピーされます。

一般的な使用の場合、WAL ストリーミングのみを構成することをお勧めします。

---

**注釈:** Barman の以前のバージョンでは、WAL アーカイブ\*と WAL ストリーミングの両方を使用することを推奨していました。これは、9.4 より前の PostgreSQL バージョンがレプリケーションスロットをサポート

していないため、WAL ストリーミングだけではすべての WAL が Barman の WAL アーカイブに安全に保存されることを保証できないためです。PostgreSQL のサポートされているすべてのバージョンにレプリケーションスロットがあるため、WAL ストリーミングのみを構成するだけで十分です。

## 3.5 バックアップの2つの一般的なシナリオ

あなたの人生を楽にするために、以下に、Barman の特定の PostgreSQL サーバーの 2 つの最も一般的なシナリオを要約します。

これは、Barman でバックアップすることにしたすべてのサーバーに対して行う必要がある決定であることに注意してください。これは、同じインストール内で異種セットアップを使用できることを意味します。

前述のように、PostgreSQL サーバー `pg` と Barman サーバー `backup` のみを心配します。ただし、実際には、アーキテクチャには、`repmgr`、`pgBouncer`、`Nagios/Icinga` などの他のテクノロジーが含まれる可能性が高くなります。

### 3.5.1 シナリオ 1 ストリーミングプロトコルを介したバックアップ

ほとんどのユースケースでは、ストリーミングバックアップインストールをお勧めします。以下の図:raw-latex:~{scenario1-design} を参照してください。

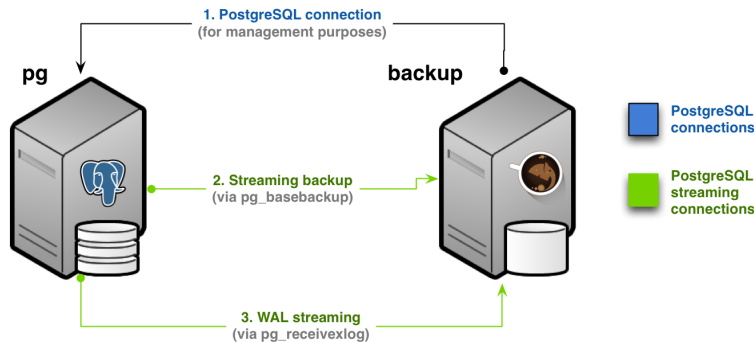


図 2: Streaming-only backup (Scenario 1)

このシナリオでは、以下を構成する必要があります。

1. 管理、調整、およびモニタリングを目的とした PostgreSQL への標準接続
2. `pg_basebackup` ベースバックアップ操作の場合と `pg_receivewal` WAL ストリーミングの場合の両方で使用されるストリーミングレプリケーション接続

Barman の用語では、このセットアップは、バックアップおよびアーカイブ操作に SSH 接続を使用しないため、**ストリーミング専用**セットアップとして知られています。これは、`Docker` 環境に特に適しており、非常に実用的です。

**Barman WAL アーカイブ** で説明したように、WAL ストリーミングに加えて、SSH を介して WAL アーカイブを構成できます。以下の図:raw-latex:~{scenario1b-design} を参照してください。

## Backup architecture for PostgreSQL

### Scenario 1b - Streaming backup with fallback WAL archiving

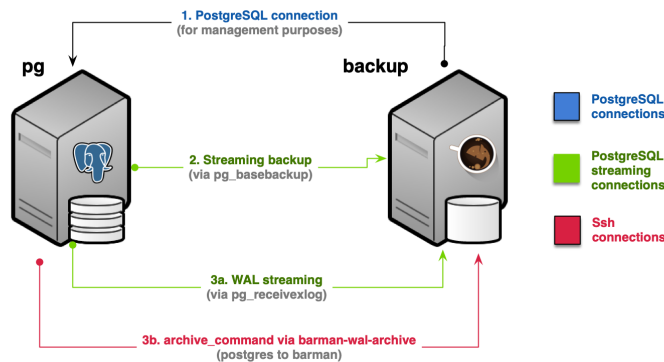


図 3: Streaming backup with WAL archiving (Scenario 1b)

SSH を介した WAL アーカイブには以下が必要です。

- PostgreSQL サーバーの `postgres` ユーザーが Barman サーバーの `barman` ユーザーとして接続できる追加の SSH 接続
- PostgreSQL の `archive_command` は、WAL ファイルを Barman に出荷するように構成します

### 3.5.2 シナリオ 2 rsync /SSH を介したバックアップ

rsync /SSH バックアップインストールは、次の機能が必要な場合に必要です。

- ファイルレベルのインクリメンタルバックアップ
- パラレルバックアップ
- テーブルスペースごとなど、帯域幅使用のより詳細な制御

このシナリオでは、以下を構成する必要があります。

1. 管理、調整、およびモニタリングを目的とした PostgreSQL への標準接続
2. `rsync` が使用するベースバックアップ操作のための SSH 接続。これにより、Barman サーバーの `barman` ユーザーは PostgreSQL サーバーの `postgres` ユーザーとして接続できます。
3. PostgreSQL の `archive_command` が使用する WAL アーカイブ用の SSH 接続、これにより、PostgreSQL サーバーの `postgres` ユーザーは、Barman サーバーの `barman` ユーザーとして接続できます

手順 3 で WAL アーカイブを構成する代わりに、[シナリオ 1 ストリーミングプロトコルを介したバックアップ](#)の説明に従って WAL ストリーミングを構成できます。これにより、`archive_command` の代わりにストリーミングレプリケーション接続が使用され、RPO が大幅に削減されます。[シナリオ 1 ストリーミングプロトコルを介したバックアップ](#)と同様に、以下の図:raw-latex: `\ref{scenario2b-design}` に示すように、WAL ストリーミングと WAL アーカイブの両方を構成することもできます。

## Backup architecture for PostgreSQL

### Scenario 2 - Traditional Barman setup

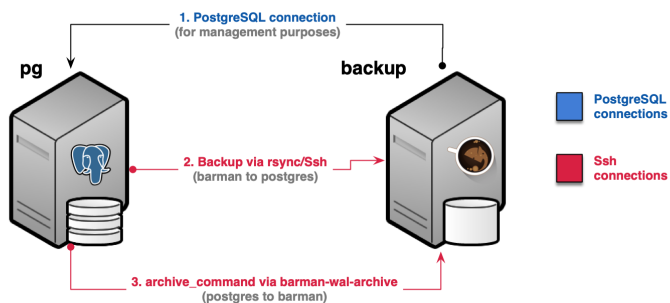


図 4: Scenario 2 - Backup via rsync/SSH

## Backup architecture for PostgreSQL

### Scenario 2b - Traditional Barman setup with WAL streaming

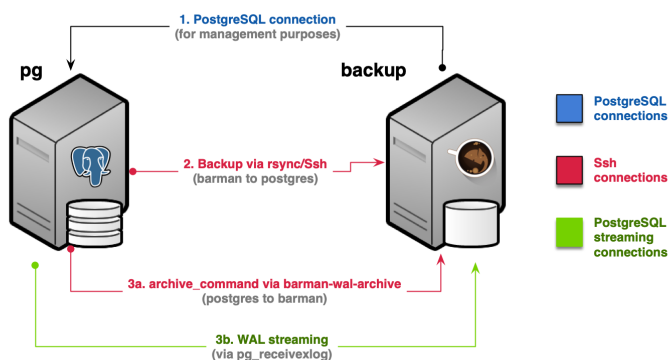


図 5: Backup via rsync/SSH with WAL streaming (Scenario 2b)

## 第4章 システム要件

- Linux/Unix
- Python  $\geq 3.6$
- Python モジュール
  - argcomplete オプション
  - psycopg2  $\geq 2.4.2$
  - python-dateutil
  - setuptools
- PostgreSQL  $\geq 10$  次のバージョンは PostgreSQL  $\geq 11$  が必要です。
- rsync  $\geq 3.1.0$  オプション

---

**重要:** RedHat Enterprise Linux、CentOS、および Scientific Linux のユーザーは、[Extra Packages Enterprise Linux (EPEL) repository][epel] をインストールする必要があります。

---

---

**注釈:** Python 2.6 および 3.5 のサポートが終了しました。Python 2.7 のサポートは Barman 3.4.X バージョンに制限されており、バグ修正のみを受け取ります。近い将来に販売終了となります。Python 3.6 のサポートは、将来のリリースで廃止されます。PostgreSQL  $< 10$  のサポートは、Barman 3.0.0 以降で廃止されました。PostgreSQL 10 のサポートは、Barman 3.5.0 後に終了します。

---

### 4.1 バックアップの要件

Barman サーバーの最も重要な要件は、利用可能なディスク領域の量です。クラスターのサイズ、1 日あたり生成される WAL ファイルの数、バックアップの頻度、および保持ポリシーに基づいて、必要なディスク領域を計画することをお勧めします。

**Barman 開発者は、XFS および ext4 を使用して Barman を定期的にテストしています。**

PostgreSQL と同様に、Barman は NFS に対して特別なことは何もしません。

NFS で Barman を安全に使用するには、次の点が必要です。

- `barman_lock_directory` は、非ネットワークファイルシステム上にある必要があります。
- NFS プロトコルのバージョン 4 を使用します。

- ファイルシステムは、ハードおよび同期オプション `hard, sync` を使用してマウントする必要があります。

## 4.2 リカバリの要件

Barman を使用すると、PostgreSQL インスタンスをローカルに Barman が存在する場所またはリモートで別のサーバーに復旧できます。

リモートリカバリは、Barman で PostgreSQL サーバーを復元する最も一般的な方法であることは間違いありません。

いずれの場合も、同じ [PostgreSQL のログ配布とポイントインタイムリカバリの要件が適用されます][requirements\_recovery]。

- 同一のハードウェアアーキテクチャ
- PostgreSQL の同一のメジャーバージョン

一般に、元のサーバーと同じではないにしても、できるだけ似たりカバリ環境を作成することを**強くお勧めします**。メンテナンスが簡単であるためです。たとえば、同じオペレーティングシステム、同じ PostgreSQL バージョン、同じディスクレイアウトなどを使用することをお勧めします。

さらに、各 PostgreSQL サーバーの専用のリカバリ環境は、オンデマンドでも、チームでディザスターリカバリ文化を育成できます。リカバリ操作を練習し、慣れておくことで、予期せぬことが発生した場合に備えることができます。

私たちの経験に基づいて、指定されたりカバリ環境は、実際の障害状況でのストレスの影響を軽減し、したがって、リカバリ操作の有効性を高めます。

最後に、NTP などを使用して、サーバー間で時間が同期していることが重要です。

## 第5章 インストール

Barman の公式パッケージは、[Barman downloads page][barman-downloads] にリストされているリポジトリを介して EnterpriseDB によって配布されています。

これらのパッケージは、ターゲットオペレーティングシステムが提供するデフォルトの python3 バージョンを使用します。代替の python3 バージョンが必要な場合は、ソースから Barman をインストールする必要があります。

---

**重要:** Barman をインストールすることをお勧めする方法は、GNU/Linux ディストリビューションで利用可能なパッケージを使用することです。

---

### 5.1 RPM パッケージを使用した Red Hat Enterprise Linux RHEL および RHEL ベースのシステムへのインストール

Barman は、RHEL8 および RHEL7 システムと、RHEL 派生の AlmaLinux、Oracle Linux、および Rocky Linux の同一バージョンに RPM パッケージを使用してインストールできます。事前に Extra Packages Enterprise Linux EPEL リポジトリと [PostgreSQL Global Development Group RPM repository][yumpgdg] をインストールする必要があります。

Barman の公式 RPM パッケージは、EnterpriseDB によって Yum を介して [public RPM repository][2ndqrpmrepo] を介して配布されます。その Web サイトで見つけられる指示に従ってください。

次に、`root` として次のように入力します。

```
yum install barman
```

EDB および PGDG リポジトリで入手可能な Barman パッケージに加えて、Fedora プロジェクトによって公開された Barman RPM は EPEL で見つけることができます。これらの RPM は Barman 開発者によってメンテナンスされておらず、PGDG および EDB リポジトリで使用可能なパッケージとは異なる構成レイアウトを使用します。

- EDB および PGDG パッケージは、メイン構成ファイルとして `/etc/barman.conf` を使用し、追加の構成ファイルとして `/etc/barman.d` を使用します。
- Fedora パッケージは、メイン構成ファイルとして `/etc/barman/barman.conf` を使用し、追加の構成ファイルとして `/etc/barman/conf.d` を使用します。

構成ファイルのレイアウトの違いは、EPEL と非 EPEL Barman パッケージ間のアップグレードが、構成ファイルが手動で更新されるまで既存の Barman インストールを中断する可能性があることを意味します。した

がって、Barman パッケージには単一のソースリポジトリを使用することをお勧めします。これは、Barman パッケージを取得したくないリポジトリの定義に次の行を追加することにより実現できます。

```
exclude=barman *python* -barman
```

具体的には

- EDB リポジトリから Barman パッケージのみを使用するには、上記の `exclude` 指示子を `/etc/yum.repos.d/epel.repo` および `/etc/yum.repos.d/pgdg-*.repo` のリポジトリ定義に追加します。
- PGDG リポジトリから Barman パッケージのみを使用するには、上記の `exclude` 指示子を `/etc/yum.repos.d/epel.repo` および `/etc/yum.repos.d/enterprisedb*.repo` のリポジトリ定義に追加します。
- EPEL リポジトリから Barman パッケージのみを使用するには、上記の `exclude` 指示子を `/etc/yum.repos.d/pgdg-*.repo` および `/etc/yum.repos.d/enterprisedb*.repo` のリポジトリ定義に追加します。

## 5.2 パッケージを使用した Debian/Ubuntu へのインストール

Barman は、パッケージを使用して Debian および Ubuntu Linux システムにインストールできます。

Debian および Ubuntu の公式リポジトリで直接入手できますが、これらのリポジトリには、利用可能な最新バージョンが含まれていない場合があります。Barman の最新バージョンが必要な場合、推奨される方法は、次の両方のリポジトリをインストールすることです。

- [パブリック APT リポジトリ][2ndqdebrepo]、Barman 開発者によって直接維持されています
- [PostgreSQL Community APT リポジトリ][aptpgdg]、[PostgreSQL Wiki の APT セクション][aptpgdg-wiki] の指示に従ってください。

---

**注釈:** PostgreSQL コミュニティ APT リポジトリプロジェクトへの Barman 開発者の直接参加のおかげで、常に最新のバージョンの Barman にアクセスできます。

---

Barman のインストールも同様に簡単です。root ユーザーとしては、次のように入力するだけです。

```
apt-get install barman
```

## 5.3 パッケージを使用した SLES へのインストール

Barman は、

[PGDG SLES repositories](#) で入手可能なパッケージを使用して SLES システムにインストールできます。PGDG site

で入手可能な手順に従って、必要なリポジトリをインストールします。

サポートされている SLES バージョン SLES 15 SP3。

必要なりポジトリがインストールされたら、Barman を root ユーザーとしてインストールできます。

```
zypper install barman
```

## 5.4 ソースからのインストール

**警告:** ソースからの Barman の手動インストールは、専門の GNU/Linux ユーザーのみが実行する必要があります。この方法で Barman をインストールするには、依存関係管理、*barman* ユーザーの作成、*barman.conf* ファイルの構成、*barman cron* コマンドの cron セットアップ、ログ管理などのシステム管理アクティビティが必要です。

backup サーバーに *barman* というシステムユーザーを作成します。*barman* ユーザーとして、ソースをダウンロードして解凍します。

システム全体をインストールする場合、次のように入力します。

```
barman@backup$ ./setup.py build
# run this command with root privileges or through sudo

barman@backup# ./setup.py install
```

ローカルインストールの場合、次のように入力します。

```
barman@backup$ ./setup.py install --user
```

*barman* アプリケーションは、ユーザーディレクトリ [make resume your PATH 環境変数が適切に設定されている][*setup\_user*]) にインストールされます。

[Barman は Python Package Index PyPI][*pypi*] でも入手でき、*pip* を介してインストールできます。

## 5.5 PostgreSQL クライアント/サーバーバイナリ

次の Barman 機能は PostgreSQL バイナリに依存しています。

- ストリーミングバックアップと *rsync* / *SSH* と *backup\_method = postgres pg\_basebackup* が必要
- *streaming\_archiver = on* を使用した Streaming WAL archiving *pg\_receivewal* または *pg\_receivevlog* が必要
- *barman verify-backup* を使用した Verifying backups *pg\_verifybackup* が必要

ターゲット OS に応じて、これらのバイナリは PostgreSQL クライアントまたはサーバーパッケージとともにインストールされます。

- RedHat / CentOS および SLES の場合 ``pg_basebackup`` および ``pg_receivewal``/``pg_receivexlog`` バイナリは、PostgreSQL クライアントパッケージとともにインストールされます。pg\_verifybackup バイナリは、PostgreSQL サーバーパッケージとともにインストールされます。※バイナリはすべて `/usr/pgsql-${PG_MAJOR_VERSION}/bin` にインストールされます。
- Debian / Ubuntu の場合\*すべてのバイナリは PostgreSQL クライアントパッケージとともにインストールされます。※バイナリは `/usr/lib/postgresql/${PG_MAJOR_VERSION}/bin` にインストールされます。

次のいずれかを確認する必要があります。

1. Barman ユーザーは、パス上に適切な PG\_MAJOR\_VERSION の bin ディレクトリがある、または
2. path\_prefix オプションは、各サーバーの Barman 構成で設定され、適切な PG\_MAJOR\_VERSION の bin ディレクトリをポイントします。

上記のバイナリに加えて、[psql][psql] プログラムをお勧めします。Barman はそれを直接使用しませんが、ドキュメントでは、PostgreSQL 接続が意図したとおりに動作していることを確認するために使用できる方法の例を提供しています。psql バイナリは、PostgreSQL クライアントパッケージにあります。

### 5.5.1 サードパーティの PostgreSQL バリエーション

サードパーティの PostgreSQL バリエーションのバックアップとリカバリーに Barman を使用している場合、上記で説明した PGDG クライアント/サーバーバイナリがバリエーションと互換性があるかどうかを確認する必要があります。互換性のない場合は、適切なパッケージから互換性のある代替品をインストールする必要があります。

## 第6章 Barmanのアップグレード

Barman はトランクベースの開発パラダイムに従っているため、安定バージョンは最新の 1 つだけです。コミットするたびに、Barman は、サポートされている各 PostgreSQL バージョン、およびサポートされている各 Linux ディストリビューションに対して数千の自動テストを実行します。

また、すべてのバージョンには、以前のバージョンとの下位互換性があります。したがって、Barman をアップグレードするには、通常、`yum update` または `apt update` を使用してパッケージを単純に更新する必要があります。

ただし、開発履歴には次の例外があり、構成にいくつかの小さな変更が必要でした。

### 6.1 Barman 3.0.0 へのアップグレード

#### 6.1.1 Rsync バックアップのデフォルトのバックアップアプローチがコンカレントになりました

`backup_options` で `concurrent_backup` も `exclusive_backup` も指定されていない場合、Barman はコンカレントバックアップを使用するようになりました。これは、デフォルトで排他的バックアップを使用する以前の Barman バージョンとは異なります。

排他的バックアップが必要な場合は、Barman 構成で `exclusive_backup` を `backup_options` に追加する必要があります。

PostgreSQL 15 に対して実行する場合、排他的バックアップは全くサポートされていないことに注意してください。

#### 6.1.2 メタデータの変更

`compression` という名前の新しいフィールドは、バージョン 3.0.0 で取得されたすべてのバックアップの `backup.info` ファイルに保存されているメタデータに追加されます。これは、`pg_basebackup` のビルトイン圧縮機能を使用して取得したバックアップからリカバリーするときに使用されます。

このフィールドの存在は、Barman Barman で作成されたバックアップを読み取ることができないことを意味します。これは、Barman 3.0.0 から以前のバージョンにダウングレードする場合、3.0.0 で取得したバックアップを手動で削除するか、各バックアップの `backup.info` ファイルを編集して `compression` フィールドを削除する必要があることを意味します。

同じメタデータの変更は `[pg-backup-api][pg-backup-api]` に影響を与えるため、`pg-backup-api` を使用している場合はバージョン 0.2.0 に更新する必要があります。

## 6.2 Barman 2.10 からのアップグレード

`barman-cloud-wal-archive` または `barman-cloud-backup` を使用している場合、バージョン 2.11 からすべてのクラウドユーティリティが新しい `barman-cli-cloud` パッケージに移動していることに注意する必要があります。したがって、最新バージョンへのアップグレードの一部として、`barman-cli-cloud` パッケージが適切にインストールされていることを確認する必要があります。上記のツールを使用していない場合は、通常のように最新バージョンにアップグレードできます。

## 6.3 Barman 2.X 2.8 より前からのアップグレード

Barman 2.7 以前のバージョンからアップグレードする前に、プライマリサーバーで `rsync` バックアップ方法を使用するユーザーは、`backup_options` を `concurrent_backup` PostgreSQL 9.6 以降で推奨または `exclusive_backup` 現在のデフォルトに明示的に設定する必要があります。それ以外の場合、Barman は実行されるたびに警告を発します。

## 6.4 Barman 1.X からのアップグレード

Barman インストールが 1.X の場合、アーカイブ戦略を明示的に構成する必要があります。以前は、`archiver` によって制御されるファイルベースのアーカイバーがデフォルトで有効になっていました。

Barman インストールを最新バージョンにアップグレードする前に、グローバルまたはそれを必要とするサーバーに対して次の行を追加してください。

```
archiver = on
```

さらに、いくつかのリリースでは、Barman はアーカイブ戦略を明示的に設定していないサーバーで `archiver = on` を透過的に設定し、警告を発します。

## 第7章 構成

Barman には 3 種類の構成ファイルがあります。

- グローバル/一般構成
- サーバー構成
- モデル構成

メイン構成ファイルデフォルトで `/etc/barman.conf` に設定されには、メインディレクトリ、システムユーザー、ログファイルなどの一般的なオプションが含まれています。

サーバー構成ファイルは、Barman によってバックアップされるサーバーごとに 1 つですが、`/etc/barman.d` ディレクトリにあり、`.conf` サフィックスが必要です。

同様に、モデル構成ファイルは `/etc/barman.d` ディレクトリにあり、`.conf` サフィックスが必要です。

**注意** モデルは、`barman config-switch` コマンドを介して、モデルと同じクラスターの一部である Barman サーバーの構成に適用できる構成オーバーライドを定義します。

---

**重要:** 歴史的な理由から、グローバルオプションとサーバーオプションとモデルオプションの両方を含む単一の構成ファイルを使用できます。ただし、メンテナンス上の理由から、このアプローチは非推奨です。

---

Barman の構成ファイルは `_INI_` 形式に従います。

構成ファイルは、異なるタイプのパラメーターを受け入れます。

- 文字列
- `enum`
- 整数
- ブール値、`on/true/1` も `off/false/0` と同様に受け入れられます。

どれも引用する必要はありません。

**注意** 一部の `enum` では、`off` は許可されますが、`false` は許可されません。

## 7.1 オプションスコープ

すべての構成オプションには `_scope_` があります。

- グローバル
- サーバー
- モデル
- `global/server` 通常グローバルレベルで設定できるサーバーオプション

グローバルオプションは、INI ファイルで `[barman]` label によって指定される `_general セクション_` で使用できます。

```
[barman]
; ... global and global/server options go here
```

サーバーオプションは、構成ファイルの行、角かっこ [ および ] で指定される `_server セクション_` でのみ指定できます。 `server` セクションは、Barman のそのサーバーの ID を表します。次の例は、`my-cluster` クラスタに属する `pg` という名前のサーバーのセクションを指定します。

```
[pg]
cluster=my-cluster
; Configuration options for the
; server named pg go here
```

モデルオプションは、`_model セクション_` でのみ指定できます。これは、`_server セクション_` と同じ方法で識別されます。`_server セクション_` と `_モデルセクション_` の識別子の間に競合があってはなりません。次の例では、`my-cluster` クラスタに属する `pg:switchover` という名前のモデルのセクションを指定します。

```
[pg:switchover]
cluster=my-cluster
model=true
; Configuration options for the model named pg:switchover, which belongs to
; the server which is configured with the option cluster=pg, go here
```

Barman にはサーバー名としてもモデル名としても使用できない予約語が 2 つあります。

- `barman` グローバルセクションの識別子
- `all` Barman が管理するすべてのサーバーでいくつかのコマンドを順番に実行できる便利なショートカット

Barman は、**構成よりコンベンションデザイン**パラダイムを実装しています。これは、柔軟性を失うことなく構成に必要なオプションの数を削減しようとしています。したがって、一部のサーバーオプションはグローバルレベルで定義し、サーバーレベルでオーバーライドできるため、ユーザーは汎用的な動作を指定し、1 つ以上のサーバーに合わせて調整できます。これらのオプションには、グローバル/サーバースコープがあります。

使用可能なすべての構成とそのスコープのリストについては、[section 5 of the ‘man’ page][man5] を参照してください。

```
man 5 barman
```

## 7.2 構成の例

以下は、メイン構成ファイルの基本的な例です。

```
[barman]
barman_user = barman
configuration_files_directory = /etc/barman.d
barman_home = /var/lib/barman
log_file = /var/log/barman/barman.log
log_level = INFO
compression = gzip
```

一方、次の例は、ストリーミングバックアップを使用するサーバー構成ファイルです。

```
[streaming-pg]
description = "Example of PostgreSQL Database (Streaming-Only)"
conninfo = host=pg user=barman dbname=postgres
streaming_conninfo = host=pg user=streaming_barman
backup_method = postgres
streaming_archiver = on
slot_name = barman
```

次の例は、クラスターが **streaming-pg** であるサーバーに適用できるオーバーライドのセットを含む構成モデルを定義します。

```
[streaming-pg:switchover]
cluster=streaming-pg
model=true
conninfo = host=pg-2 user=barman dbname=postgres
streaming_conninfo = host=pg-2 user=streaming_barman
```

次のコードは、**rsync** /**SSH** を使用した従来のバックアップの基本的な例を示しています。

```
[ssh-pg]
description = "Example of PostgreSQL Database (via Ssh)"
ssh_command = ssh postgres@pg
conninfo = host=pg user=barman dbname=postgres
backup_method = rsync
parallel_jobs = 1
reuse_backup = link
archiver = on
```

詳細については、配布された `barman.conf` ファイル、および `ssh-server.conf-template` および `streaming-server.conf-template` テンプレートファイルを参照してください。

## 第8章 Barmanでの新しいサーバーのセットアップ

\_ “デザインとアーキテクチャ” \_セクションで述べたように、次の規則を使用します。

- PostgreSQL がインストールされるサーバー ID とホスト名として `pg`
- Barman が存在するホスト名としての `backup`
- `backup` サーバーで Barman を実行しているユーザーとしての `barman backup` サーバー構成のパラメーター `barman_user` で識別されます
- `pg` サーバーで PostgreSQL を実行しているユーザーとしての `postgres`

---

**重要:** Barman のサーバーは、バックアップおよびリカバリー可能履歴全体として同じ PostgreSQL インスタンスつまり同じシステム識別子を参照する必要があります。\*\*\*これは、インスタンスのアップグレードを実行する場合たとえば `pg_upgrade`、Barman で同じサーバー定義を再利用せず、共通点は何もないため別のサーバー定義を使用する必要があることを意味します。

---



## 第9章 準備的な手順

このセクションには、Barman で PostgreSQL サーバーをセットアップする前に実行する必要がある予備手順が含まれています。

---

**重要:** 続行する前に、\_"デザインとアーキテクチャ"\_セクションで概要を示しているように、WAL アーカイブとバックアップ戦略の観点から決定を下すことが重要です。特に、使用する WAL アーカイブ方法とバックアップ方法を決定する必要があります。

---

### 9.1 PostgreSQL 接続

backup サーバーがスーパーユーザーとして pg の PostgreSQL サーバーに接続できること、またはデータベースに接続するユーザーに正しい特権セットが付与されていることを確認する必要があります。

次のとおりに、PostgreSQL で barman という名前の特定のスーパーユーザーを作成できます。

```
postgres@pg$ createuser -s -P barman
```

または、次のとおりに必要な特権セットを持つ通常のユーザーを作成します。

```
postgres@pg$ createuser -P barman
```

```
GRANT EXECUTE ON FUNCTION pg_backup_start(text, boolean) to barman;
GRANT EXECUTE ON FUNCTION pg_backup_stop(boolean) to barman;
GRANT EXECUTE ON FUNCTION pg_switch_wal() to barman;
GRANT EXECUTE ON FUNCTION pg_create_restore_point(text) to barman;

GRANT pg_read_all_settings TO barman;
GRANT pg_read_all_stats TO barman;
```

PostgreSQL バージョン 14 以前のバージョンを使用する場合、ファンクション `pg_backup_start` および `pg_backup_stop` は異なる名前と異なるシグネチャを持っていました。したがって、上記のブロックの最初の 2 行を次のように置き換える必要があります。

```
GRANT EXECUTE ON FUNCTION pg_start_backup(text, boolean, boolean) to barman;
GRANT EXECUTE ON FUNCTION pg_stop_backup() to barman;
GRANT EXECUTE ON FUNCTION pg_stop_backup(boolean, boolean) to barman;
```

実際のスーパーユーザーのない PostgreSQL バージョン 13 以下では、`barman switch-wal` コマンドの `--force` オプションは機能しないことに注意してください。PostgreSQL バージョン 15 以降を実行している場合は、`pg_checkpoint` ロールを付与できるため、スーパーユーザーなしでこの機能を使用できます。

```
GRANT pg_checkpoint TO barman;
```

**重要:** 上記の `createuser` コマンドは、パスワードの入力を求めます。パスワードは、*backup* サーバー上の `~barman/.pgpass` ファイルに追加することをお勧めします。詳細については、["The Password File"セクションの PostgreSQL Documentation][pgpass] を参照してください。

この接続は、Barman がサーバーとのアクティビティを調整するために、およびモニタリングのために必要です。

PostgreSQL が提供するものの中からお気に入りのクライアント認証方法を選択できます。詳細については、["Client Authentication"セクションの PostgreSQL Documentation][pg\_hba] を参照してください。

*backup* ホストで *barman* ユーザーとして次のコマンドを実行して、*backup* ホストが *pg* ホスト上の PostgreSQL に接続できることを確認します。

```
barman@backup$ psql -c SELECT version() -U barman -h pg postgres
```

上記の情報ユーザー名、ホスト名、データベース名を書き留め、後で使用するために保存しておきます。この例のように、サーバー構成の `conninfo` オプションでこれが必要になります。

```
[pg]
; ...
conninfo = host=pg user=barman dbname=postgres application_name=myapp
```

**注釈:** `application_name` はオプション。

## 9.2 PostgreSQL WAL のアーカイブとレプリケーション

続行する前に、*pg* で PostgreSQL を適切に構成して、Barman サーバーからのストリーミングレプリケーション接続を受け入れる必要があります。PostgreSQL ドキュメントの次のセクションを読んでください。

- [ロール属性][ロール]
- [pg\_hba.conf ファイル][authpg\_hba]
- [ストリーミングレプリケーションを使用したスタンバイサーバーのセットアップ][streamprot]

非常に重要な構成パラメーターの 1 つが `wal_level` パラメーターです。このパラメーターは、バックアップの一貫性を保つために必要なすべての有用な情報がトランザクションログファイルに含まれるように構成する必要があります。

```
wal_level = replica|logical
```

構成を更新するために PostgreSQL サーバーを再起動します。

## 9.3 PostgreSQL ストリーミング接続

WAL ストリーミングまたはストリーミングバックアップを使用する予定がある場合は、ストリーミング接続をセットアップする必要があります。次のとおり、PostgreSQL で `streaming_barman` という名前の特定のユーザーを作成することをお勧めします。

```
postgres@pg$ createuser -P --replication streaming_barman
```

**重要:** 上記のコマンドは、パスワードの入力を求めます。パスワードは、`backup` サーバー上の `~barman/.pgpass` ファイルに追加することをお勧めします。詳細については、["The Password File"セクションの PostgreSQL Documentation][pgpass] を参照してください。

次のコマンドを使用して、ストリーミング接続が機能していることを手動で確認できます。

```
barman@backup$ psql -U streaming_barman -h pg \
-c "IDENTIFY_SYSTEM" \
replication=1
```

接続が機能している場合は、システム識別子、現在のタイムライン ID、および現在の WAL フラッシュの場所を含む応答が表示されます。例

| systemid            | timeline | xlogpos    | dbname |
|---------------------|----------|------------|--------|
| 7139870358166741016 | 1        | 1/330000D8 |        |

(1 row)

**重要:** 続行する前に、ストリーミングレプリケーションを介して接続できることを確認してください。

PostgreSQL 構成ファイルで `max_wal_senders` パラメーターを構成する必要もあります。WAL 送信者の数は、実装した PostgreSQL アーキテクチャによって異なります。この例では、2 に設定しています。

```
max_wal_senders = 2
```

このオプションは、サーバーが管理できる同時ストリーミング接続の最大数を表します。

もう 1 つの重要なパラメーターは `max_replication_slots` で、これはサーバーが管理できるレプリケーションスロット [replslot94] の最大数を表します。このパラメーターは、ストリーミング接続を使用して WAL ファイルを受信する予定がある場合に必要です。

```
max_replication_slots = 2
```

[^replslot94] レプリケーションスロットは PostgreSQL 9.4 で導入されました。詳細については、セクション\_“WAL ストリーミング/レプリケーションスロット”\_を参照してください。

`max_replication_slots` および `max_wal_senders` として提案された値は例として考えられる必要があります、実際のセットアップで使用する値は、アーキテクチャを注意深く評価した後に選択する必要があります。ガイドラインと説明については、PostgreSQL ドキュメントを参照してください。

## 9.4 SSH 接続

SSH は、リモートサーバーへのリモートシェルを開き、サーバーとローカルシステム間でファイルをコピーできるプロトコルとツールセットです。Digital Ocean の記事 [“SSH Essentials”][ssh\_essentials] で、SSH の使用に関するドキュメントの詳細を見つけることができます。

SSH キー交換は、別のマシン上のユーザー間の安全なパスワードレス接続を実装するために使用される非常に一般的な手法であり、WAL アーカイブとバックアップに `rsync` を使用する必要があります。

---

**注釈:** このプロシージャは、トランザクションログのアーカイブと PostgreSQL サーバーのバックアップにのみストリーミング接続を使用する予定がある場合に必要です。

---

[ssh\_essentials] <https://www.digitalocean.com/community/tutorials/ssh-essentials-working-with-ssh-servers-clients-and-keys>

## 9.5 postgres ユーザーの SSH 構成

これまでに行ったことがない限り、PostgreSQL ユーザーの SSH キーを作成する必要があります。pg ホストに `postgres` としてログインし、次のように入力します。

```
postgres@pg$ ssh-keygen -t rsa
```

このキーは、パスワードを提供せずにホストから接続するために使用する必要があるため、キーペアの作成中にパズフレーズを入力する必要はありません。

## 9.6 barman ユーザーの SSH 構成

前の段落と同様に、Barman ユーザーの SSH キーを作成する必要があります。backup ホストに `barman` としてログインし、次のように入力します。

```
barman@backup$ ssh-keygen -t rsa
```

同じ理由で、パズフレーズは入力しないでください。

## 9.7 PostgreSQL から Barman へ

`archive_command` 設定を使用して WAL ファイルを正しくアーカイブするには、PostgreSQL サーバーからバックアップサーバーへの SSH 接続が必要です。

PostgreSQL サーバーからバックアップサーバーに正常に接続するには、PostgreSQL 公開キーを `barman` ユーザーのバックアップサーバーの承認されたキーに構成する必要があります。

承認される公開キーは、`postgres` ユーザーのホームディレクトリ内の `.ssh/id_rsa.pub` という名前のファイルに保存され、そのコンテンツは、バックアップサーバーの `barman` ユーザーのホームディレクトリ内の `.ssh/authorized_keys` という名前のファイルに含まれる必要があります。 `authorized_keys` ファイルが存在しない場合は、権限として `600` を使用して作成します。

SSH キーペアの交換が正常に完了した場合、次のコマンドは出力なしで成功するはずです。

```
postgres@pg$ ssh barman@backup -C true
```

`archive_command` 構成パラメーターの値については、\_“WAL archive via `archive_command` セクション”\_で説明します。

## 9.8 Barman から PostgreSQL へ

バックアップサーバーと PostgreSQL サーバー間の SSH 接続は、`rsync` を介した従来のバックアップに使用されます。PostgreSQL サーバーからバックアップサーバーへの接続と同様に、`postgres` ユーザーの PostgreSQL サーバーでバックアップサーバーの公開キーを承認する必要があります。

`barman` サーバーのファイル `.ssh/id_rsa.pub` の内容は、PostgreSQL サーバーの `.ssh/authorized_keys` という名前のファイルに配置する必要があります。そのファイルのアクセス許可は `600` である必要があります。

キーペアの交換が正常に完了した場合、次のコマンドは出力なしで成功するはずです。

```
barman@backup$ ssh postgres@pg -C true
```



## 第10章 サーバー構成ファイル

/etc/barman.d ディレクトリに、次の内容を使用して `pg.conf` という新しいファイルを作成します。

```
[pg]
description = "Our main PostgreSQL server"
conninfo = host=pg user=barman dbname=postgres
backup_method = postgres
# backup_method = rsync
```

`conninfo` オプションは、セクション\_“準備手順: PostgreSQL 接続”\_に従って設定されます。

`backup_method` オプションの意味については、このガイドのバックアップセクションで説明します。

WAL アーカイブ、またはサーバーのバックアップにストリーミング接続を使用する予定がある場合、サーバー構成ファイルに `streaming_conninfo` パラメーターも必要です。

```
streaming_conninfo = host=pg user=streaming_barman dbname=postgres
```

セクション\_“準備手順: PostgreSQL 接続”\_の説明に従って、この値を選択する必要があります。



## 第11章 WAL ストリーミング

Barman は、ユーザーが標準の `archive_command` 戦略に加えて、PostgreSQL サーバーから継続的な WAL ストリーミングを追加できることにより、目標復旧ポイント RPO を削減できます。

Barman は `[pg_receivewal][25]` に依存しており、ネイティブストリーミングレプリケーションプロトコルを利用し、PostgreSQL サーバースタンドバイからトランザクションログを継続的に受信します。PostgreSQL 10 より前では、`pg_receivewal` の名前は `pg_receivexlog` でした。

---

**重要:** Barman では、`pg_receivewal` が同じサーバーにインストールされている必要があります。下位互換性があるため、`pg_receivewal` の最新バージョンをインストールすることをお勧めします。それ以外の場合、ユーザーは Barman サーバーに `pg_receivewal` の複数のバージョンをインストールし、構成ファイルの `path_prefix` オプションを使用してサーバーの特定のバージョンを適切にポイントできます。

---

トランザクションログのストリーミングを有効にするには、次のことを行う必要があります。

1. 前述のようにストリーミング接続をセットアップします
2. `streaming_archiver` オプションを `on` に設定します

`cron` コマンドは、前述の要件が満たされている場合、`receive-wal` コマンドの実行を介してログストリーミングを透過的に管理します。これは推奨されるシナリオです。

ただし、ユーザーは `receive-wal` コマンドを手動で実行できます。

```
barman receive-wal <server_name>
```

---

**注釈:** `receive-wal` コマンドはフォアグラウンドプロセスです。

---

トランザクションログは、`streaming_wals_directory` 構成オプションで指定されたディレクトリに直接ストリーミングされ、`archive-wal` コマンドでアーカイブされます。

`streaming_archiver_name` パラメーターで指定しない限り、Barman は WAL ストリーマープロセスの `application_name` を `barman_receive_wal` に設定し、PostgreSQL サーバーの `pg_stat_replication` システムビューでそのステータスをモニタできます。

## 11.1 レプリケーションスロット

レプリケーションスロットは、PostgreSQL サーバーがすべてのアーカイバーが受信するまで WAL ファイルを削除しないようにする自動方法です。Barman は、このメカニズムを使用して PostgreSQL からトランザクションログを受信します。

レプリケーションスロットの詳細については、[PostgreSQL documentation][replication-slots] で見つけることができます。

ストリーミング接続のみに基づいてバックアップアーキテクチャをベースにすることもできます。このシナリオは、Docker ベースの PostgreSQL サーバーを構成し、Windows で実行されている PostgreSQL サーバーを操作するのに役立ちます。

---

**重要:** 現時点では、Windows サポートはまだ実験的であり、継続的インテグレーションシステムの一部ではありません。

---

## 11.2 WAL ストリーミングを構成する方法

最初に、トランザクションログファイルを Barman サーバーにストリーミングするように PostgreSQL サーバーを構成する必要があります。

Barman から PostgreSQL サーバーへのストリーミング接続を構成するには、サーバー構成ファイルに次の行を含めて、`streaming_archiver` を有効にする必要があります。

```
streaming_archiver = on
```

レプリケーションスロット推奨を使用する予定がある場合、ストリーミングベースのトランザクションログアーカイブのセットアップに重要なもう 1 つのオプションは `slot_name` オプションです。

```
slot_name = barman
```

このオプションは、Barman が使用するレプリケーションスロットの名前を定義します。レプリケーションスロットを使用する場合、これは必須です。

レプリケーションスロット名を構成するときに、次のコマンドを使用して Barman のレプリケーションスロットを手動で作成できます。

```
barman@backup$ barman receive-wal --create-slot pg
Creating physical replication slot barman on server pg
Replication slot barman created
```

Barman 2.10 以降、次の設定を使用してレプリケーションスロットを自動的に作成するように Barman を構成できます。

```
create_slot = auto
```

## 11.3 さまざまなホストからのストリーミング WAL とバックアップ Barman 3.10.0 以降

Barman は、WAL セグメントをストリーミングするための `pg_receivewal` プロセスを作成するときに `streaming_conninfo` で定義された接続情報を使用し、レプリケーションスロットのステータスを確認するときに `conninfo` を使用します。 `conninfo` および `streaming_conninfo` はバックアップを取るときにも使用されるため、このデフォルト構成では、Barman に WAL をストリーミングし、同じホストからバックアップを取得します。

プライマリからストリーミングされる WAL と、スタンバイからバックアップがソースされるなど、代替構成が必要な場合、次のオプションを使用して実現できます。

- `wal_streaming_conninfo` ストリーミングレプリケーションプロトコルを介して WAL セグメントを受信するとき、および WAL の受信に使用されるレプリケーションスロットのステータスを確認するときに、Barman が `streaming_conninfo` の代わりに使用する接続文字列。
- `wal_conninfo` WAL ストリーミングステータスを監視し、関連するチェックを実行するためのオプションの接続文字列。設定されている場合、Barman はレプリケーションスロットのステータスを確認するときに `wal_streaming_conninfo` の代わりにこれを使用します。

次の制限が適用され、チェック中に Barman によって強制されます。

- `wal_streaming_conninfo` および `wal_conninfo` によって定義された接続は、`streaming_conninfo` および `conninfo` 接続が到達するのと同じクラスターに属する PostgreSQL インスタンスに到達する必要があります。
- `wal_streaming_conninfo` 接続文字列は、ストリーミングレプリケーション接続を作成できる必要があります。
- `wal_streaming_conninfo` または `wal_conninfo` 設定されている場合、設定を読み取り、レプリケーションスロットのステータスを確認するための十分な権限が必要です。必要な権限は次のいずれかです。
  - `pg_monitor` ロール。
  - `pg_read_all_settings` および `pg_read_all_stats` ロールの両方。
  - `superuser` ロール。

---

**重要:** クラスター内の\*任意の\*PostgreSQL インスタンスから WAL をストリーミングすることは可能ですが、スタンバイから WAL をストリーミングするときに WAL セグメントが失われるリスクがあります、そのようなスタンバイが独自の上流ソースに追いつけない場合。このため、WAL は常にプライマリから直接ストリーミングすることを\*強くお勧めします。

---

## 11.4 リカバリーを含む部分的な WAL ファイルの制限

`pg_receivewal` の標準動作は、WAL セグメント名の後に `.partial` サフィックスが付いているファイルにトランザクション情報を書き込むことです。

Barman は、部分ファイルがサーバーの `streaming_wals_directory` にあることを想定しています。完了すると、`pg_receivewal` は `.partial` サフィックスを削除して次のサフィックスを開き、永続的なストレージと圧縮のために Barman の `archive-wal` コマンドにファイルを配信します。

マスター PostgreSQL サーバーに突然の回復不能な障害が発生した場合、Barman にストリーミングされた `.partial` ファイルには、標準のアーカイバー PostgreSQL の `archive_command` を介して Barman に配信できなかった非常に重要な情報が含まれています。

Barman 2.10 の時点で、`get-wal` コマンドは、`--partial/-P` オプションを介して現在の `.partial` WAL ファイルのコンテンツを返すことができます。これは、完全またはポイント・インタイムまでのリカバリーの場合に特に役立ちます。したがって、`get-wal` を有効にして `--standby-mode` を使用せずに `recover` コマンドを実行した場合、Barman は `restore_command` リカバリオプションの `barman-wal-restore` に `-P` オプションを自動的に追加します。これはリモートの `get-wal` コマンドに中継します。

`get-wal` は、WAL ファイルが既に Barman に出荷されているが、まだアーカイブされていない場合、`incoming` ディレクトリも検索します。

## 第12章 archive\_command を介した WAL アーカイブ

archive\_command は、WAL ファイルをアーカイブする従来の方法です。

この PostgreSQL 構成パラメーターの値は、PostgreSQL サーバーによって実行されて WAL ファイルを Barman 受信ディレクトリにコピーするシェルコマンドである必要があります。

これは2つの方法で行うことができ、どちらも SSH 接続が必要です。

- barman-wal-archive ユーティリティ経由 Barman 2.6 から
- rsync / SSH 経由 Barman 2.6 以前の一般的なアプローチ

詳細については、以下のセクションを参照してください。

---

**重要:** Barman がこの機能をサポートする方法の詳細については、「コンカレントバックアップとスタンバイからのバックアップ」セクションを読んでください。

---

### 12.1 barman-wal-archive を介した WAL アーカイブ

Barman 2.6 から、archive\_command を介して WAL ファイルを Barman に安全かつ確実にアーカイブする **推奨方法**は、barman-cli パッケージに含まれる barman-wal-archive コマンドを使用することです。EnterpriseDB パブリックリポジトリを介して配布され、GNU GPL3 ライセンスで利用できます。barman-cli は、Barman クラスターの一部である各 PostgreSQL サーバーにインストールする必要があります。

rsync / SSH の代わりに barman-wal-archive を使用すると、Barman サーバー上の出荷された WAL ファイルのデータ破損のリスクが削減されます。WAL ファイルの archive\_command として rsync / SSH を使用する場合、ファイルのコンテンツが宛先のディスクにフラッシュおよび fsync されることを保証するメカニズムはありません。

このため、私たちは Barman の put-wal コマンド 2.6 で導入とネイティブ通信する barman-wal-archive ユーティリティを開発しました。このコマンドは、ファイルを受信し、そのコンテンツを fsync し、そのサーバーの適切な incoming ディレクトリに配置します。したがって、archive\_command で使用される唯一のパラメーターはサーバーの ID であるため、barman-wal-archive は、Barman の間違った場所/ディレクトリに WAL ファイルをコピーするリスクを削減します。

barman-wal-archive コマンドの詳細については、PostgreSQL サーバーで man barman-wal-archive と入力してください。

次のコマンドで、`barman-wal-archive` が Barman サーバーに接続できること、および必要な PostgreSQL サーバーが着信 WAL ファイルを受け入れるように Barman で構成されていることを確認できます。

```
barman-wal-archive --test backup pg DUMMY
```

`backup` は Barman がインストールされているホストであり、`pg` は Barman で構成された PostgreSQL サーバーの名前であり、`DUMMY` はプレースホルダーです。`barman-wal-archive` には WAL ファイル名の引数が必要ですが、これは無視されます。

すべてが正しく構成されている場合、次の出力が表示されるはずです。

```
Ready to accept WAL files for the server pg
```

SSH を使用して Barman サーバーと通信するため、`postgres` ユーザーがバックアップサーバーに `barman` としてログインするには、SSH キー認証が必要です。SSH デフォルトの 22 以外のポートを使用する必要がある場合は、`--port` オプションを追加して、SSH 接続に使用するポートを指定できます。

`pg` データベース上の PostgreSQL インスタンスの `postgresql.conf` ファイルを編集し、アーカイブモードをアクティブにして、`barman-wal-archive` を使用するように `archive_command` を設定します。

```
archive_mode = on
wal_level = replica
archive_command = barman-wal-archive backup pg %p
```

次に、PostgreSQL サーバーを再起動します。

## 12.2 rsync / SSH を介した WAL アーカイブ

`show-servers` Barman コマンドを使用し、`incoming_wals_directory` 値を探して、着信 WAL ディレクトリを取得できます。

```
barman@backup$ barman show-servers pg |grep incoming_wals_directory
incoming_wals_directory: /var/lib/barman/pg/incoming
```

`pg` データベース上の PostgreSQL インスタンスの `postgresql.conf` ファイルを編集し、アーカイブモードをアクティブにします。

```
archive_mode = on
wal_level = replica
archive_command = rsync -a %p barman@backup:INCOMING_WALS_DIRECTORY/%f
```

`INCOMING_WALS_DIRECTORY` プレースホルダーを、上記の `barman show-servers pg` コマンドによって返された値に変更してください。

PostgreSQL サーバーを再起動します。

場合によっては、より厳密なチェックを `archive_command` プロセスに追加することができます。たとえば、一部のユーザーは次のことを提案しました。

```
archive_command = test $(/bin/hostname --fqdn) = HOSTNAME \  
&& rsync -a %p barman@backup:INCOMING_WALS_DIRECTORY/%f
```

HOSTNAME プレースホルダーは、`hostname --fqdn` によって返される値に置き換える必要があります。この `_trick_` は、サーバーがクローンされ、復旧した PostgreSQL インスタンスから WAL ファイルの受信を回避する場合の安全対策です。



## 第13章 WAL アーカイブ構成の検証

継続的なアーカイブがオンで適切に動作していることをテストするには、PostgreSQL サーバーとバックアップサーバーの両方をチェックする必要があります。特に、WAL ファイルが宛先ディレクトリに正しく収集されていることを確認する必要があります。

この目的に、WAL アーカイブプロセスの検証を容易にするために、`switch-wal` コマンドが開発されました。

```
barman@backup$ barman switch-wal --force --archive pg
```

上記のコマンドは、PostgreSQL に WAL ファイルを強制的に切り替え、Barman のアーカイブプロセスをトリガーします。Barman は、1 つのファイルが 30 秒以内に到着するまで待機します。`--archive-timeout` オプションを介してタイムアウトは変更できます。WAL ファイルを受信しない場合、エラーが返されます。

`barman check` コマンドを使用して、WAL アーカイブが正しく構成されているかどうかを確認できます。



## 第14章 ストリーミングバックアップ

Barman は、`pg_basebackup` に依存して、ストリーミング接続を使用して PostgreSQL サーバーをバックアップできます。バージョン 3.11 以降、Barman は、ストリーミング接続を使用したブロックレベルのインクリメンタルバックアップもサポートしています。詳細については、「機能の詳細」セクションを参照してください。

---

**重要:** Barman では、`pg_basebackup` が同じサーバーにインストールされている必要があります。下位互換性があるため、`pg_basebackup` の最後の利用可能なバージョンをインストールすることをお勧めします。構成ファイルの `path_prefix` オプションを使用して、Barman サーバーに `pg_basebackup` の複数のバージョンをインストールし、サーバーの特定のバージョンを適切にポイントすることもできます。

---

ストリーミング接続を使用してサーバーを正常にバックアップするには、バックアップ方法として `postgres` を使用する必要があります。

```
backup_method = postgres
```

---

**重要:** WAL が `archiver` または `streaming_archiver` を介して Barman に正しくアーカイブされていない場合、バックアップを開始することはできません。

---

サーバー構成が有効であるかどうかを確認するには、`barman check` コマンドを使用できます。

```
barman@backup$ barman check pg
```

バックアップを開始するには、`barman backup` コマンドを使用できます。

```
barman@backup$ barman backup pg
```



## 第15章 rsync /SSHによるバックアップ

rsync によるバックアップは、バージョン 2.0 より前の Barman のバックアップの唯一の方法であり、3.11 より前は、インクリメンタルバックアップをサポートする唯一の方法でした。現在の Barman は、ファイルレベルおよびブロックレベルのインクリメンタルバックアップをサポートしています。rsync を使用したバックアップは、ファイルレベルのバックアップ機能を実装します。詳細については、[“機能の詳細”](#) セクションを参照してください。

rsync を使用してバックアップするには、これらのパラメーターを Barman サーバー構成ファイル内に配置する必要があります。

```
backup_method = rsync
ssh_command = ssh postgres@pg
```

backup\_method オプションは rsync バックアップ方法をアクティブにします Barman サーバーから PostgreSQL サーバーへの SSH 接続を正しく作成するには、ssh\_command オプションが必要です。

---

**重要:** WAL が ``archiver`` または ``streaming_archiver`` を介して Barman に正しくアーカイブされていない場合、バックアップを開始することはできません。

---

サーバー構成が有効であるかどうかを確認するには、`barman check` コマンドを使用できます。

```
barman@backup$ barman check pg
```

バックアップするには、`barman backup` コマンドを使用します。

```
barman@backup$ barman backup pg
```

---

**注釈:** Barman 3.11.0 以降、Barman は rsync ベースのバックアップを取得するときにキープアライブメカニズムを使用します。Barman が ``pg_backup_start`` / ``pg_backup_stop`` 低レベル API ファンクションを実行する libpq 接続を介して単純な ``SELECT 1`` クエリーを送信し続け、長時間アイドル状態になることができるため、ファイアウォールまたはルーターがその接続をドロップする確率を減らすために配置されています。ベースバックアップはコピー中です。keepalive\_interval 構成オプションを使用して、ハートビートの間隔を制御したり、メカニズムを無効にしたりできます。

---



## 第16章 クラウドスナップショットを使用したバックアップ

Barman は、ストレージボリュームのスナップショットを取得することにより、特定のクラウド環境内に展開された PostgreSQL サーバーのバックアップを作成できます。この方法で構成された場合、PostgreSQL ファイルの物理バックアップはクラウドに保存されるボリュームスナップショットであり、Barman は WAL とバックアップカタログのストレージサーバーとして機能します。これらのバックアップは、バックアップデータ自分自身がクラウドに保存されている場合でも、rsync または postgres バックアップ方法で取得された従来のバックアップと同じように Barman で管理できます。

適切な PostgreSQL サーバーで `barman-cloud-backup` コマンドを直接使用して、Barman サーバーを使用せずにスナップショットバックアップを作成することもできます。

### 16.1 クラウドスナップショットの前提条件

Barman でスナップショットバックアップ方法を使用するには、展開が次の前提条件を満たしている必要があります。

- PostgreSQL は、サポートされているクラウドプロバイダー内のコンピューティングインスタンスに展開する必要があります。
- PGDATA やテーブルスペースデータなどのすべての重要なデータが、スナップショットをサポートするストレージボリュームに保存されるように、PostgreSQL を構成する必要があります。
- `findmnt` コマンドは、PostgreSQL ホストで使用できる必要があります。

---

**重要:** PGDATA の外部に保存されている構成ファイルは、スナップショットに含まれません。このようなファイルの管理は、構成管理システムなどの別のメカニズムを使用して実行する必要があります。

---

### 16.2 Google Cloud Platform スナップショットの前提条件

`google-cloud-compute` および `grpcio` ライブラリは、Barman が使用する Python ディストリビューションで利用できる必要があります。これらのライブラリはオプションの依存関係であり、Barman パッケージによって標準としてインストールされません。これらは `pip` を使用して次のようにインストールできます。

```
pip3 install grpcio google-cloud-compute
```

**注釈:** google-cloud-compute ライブラリが必要とする Python の最小バージョンは 3.7 です。GCP スナップショットは、以前のバージョンの Python では使用できません。

---

次の追加前提条件は、Google Cloud Platform でのスナップショットバックアップに適用されます。

- スナップショットバックアップに含まれるすべてのディスクは、ゾーン永続ディスクである必要があります。リージョン永続ディスクは現在サポートされていません。
- 必要な権限セットを持つサービスアカウントを Barman が利用できる必要があります。これは、このようなアカウントを Barman を実行しているコンピューティングインスタンスに接続するか、GOOGLE\_APPLICATION\_CREDENTIALS 環境変数を使用して資格情報ファイルをポイントすることにより実現できます。

必要な権限は次のとおりです。

- `compute.disks.createSnapshot`
- `compute.disks.get`
- `compute.globalOperations.get`
- `compute.instances.get`
- `compute.snapshots.create`
- `compute.snapshots.delete`
- `compute.snapshots.list`

## 16.3 Azure スナップショットの前提条件

azure-mgmt-compute および azure-identity ライブラリは、Barman が使用する Python ディストリビューションで利用できる必要があります。

これらのライブラリはオプションの依存関係であり、Barman パッケージによって標準としてインストールされません。これらは `pip` を使用して次のようにインストールできます。

```
pip3 install azure-mgmt-compute azure-identity
```

**注釈:** azure-mgmt-compute ライブラリが必要とする Python の最小バージョンは 3.7 です。Azure スナップショットは、以前のバージョンの Python では使用できません。

---

次の追加前提条件は、Azure でのスナップショットバックアップに適用されます。

- スナップショットバックアップに含まれるすべてのディスクは、データディスクとして VM インスタンスに接続されている管理ディスクである必要があります。
- Barman は、マネージド ID または CLI ログインを使用して取得した資格情報を使用でき、これにより、必要な権限セットで Azure へのアクセスを付与する必要があります。

次の権限が必要です。

- Microsoft.Compute/disks/read
- Microsoft.Compute/virtualMachines/read
- Microsoft.Compute/snapshots/read
- Microsoft.Compute/snapshots/write
- Microsoft.Compute/snapshots/delete

## 16.4 AWS スナップショットの前提条件

boto3 ライブラリは、Barman が使用する Python ディストリビューションで利用できる必要があります。

このライブラリはオプショナルの依存関係であり、Barman パッケージによって標準としてインストールされません。pip を使用して次のようにインストールできます。

```
pip3 install boto3
```

次の追加前提条件は、AWS でのスナップショットバックアップに適用されます。

- スナップショットバックアップに含まれるすべてのディスクは非ルート EBS ボリュームであり、同じ VM インスタンスに接続されている必要があります。
- NVMe ボリュームは現在サポートされていません。

次の権限が必要です。

- ec2:CreateSnapshot
- ec2:CreateTags
- ec2:DeleteSnapshot
- ec2:DescribeSnapshots
- ec2:DescribeInstances
- ec2:DescribeVolumes

## 16.5 スナップショットバックアップの構成

クラウドスナップショットを介したバックアップ用に Barman を構成するには、`backup_method` パラメーターを `snapshot` に設定し、`snapshot_provider` をサポートされているクラウドプロバイダーに設定します。

```
backup_method = snapshot
snapshot_provider = gcp
```

現在、Google Cloud Platform gcp、Microsoft Azure azure、および AWS aws がサポートされています。

クラウドプロバイダーに関係なく、次のパラメーターを設定する必要があります。

```
snapshot_instance = INSTANCE_NAME
snapshot_disks = DISK_NAME,DISK2_NAME,...
```

`snapshot_instance` は、ストレージボリュームが接続されている VM またはコンピューティングインスタンスの名前に設定され、`snapshot_disks` は、バックアップに含めるディスクのコンマ区切りリストです。

---

**重要:** `snapshot_disks` には、PostgreSQL に必要なデータを保存するすべてのディスクが含まれていることを確認する必要があります。`snapshot_disks` にリストされているストレージボリュームに保存されていないデータは、バックアップに含まれないため、リカバリ時に使用できません。

---

## 16.6 Google Cloud Platform スナップショットの構成

GCP を使用する場合、次の追加パラメーターを設定する必要があります。

```
gcp_project = GCP_PROJECT_ID
gcp_zone = ZONE
```

`gcp_project` は、`snapshot_instance` および `snapshot_disks` で定義されたインスタンスとストレージボリュームを所有する GCP プロジェクトの ID に設定する必要があります。`gcp_zone` は、インスタンスが存在するアベイラビリティゾーンに設定する必要があります。

## 16.7 Azure スナップショットの構成

Azure を使用する場合、次の追加パラメーターを設定する必要があります。

```
azure_subscription_id = AZURE_SUBSCRIPTION_ID
azure_resource_group = AZURE_RESOURCE_GROUP
```

`azure_subscription_id` は、`snapshot_instance` および `snapshot_disks` で定義されたインスタンスとストレージボリュームを所有する Azure サブスクリプション ID の ID に設定する必要があります。`azure_resource_group` は、インスタンスとディスクが属するリソースグループに設定する必要があります。

## 16.8 AWS スナップショットの構成

`snapshot_instance` または `snapshot_disks` を指定すると、Barman は AWS または名前によってリソースに割り当てられたインスタンス/ボリューム ID を受け入れます。名前が使用される場合、Barman は AWS を照会して、一致する `Name` タグを持つリソースを見つけます。0 または複数の一致するリソースが見つかった場合、Barman はエラーで終了します。

AWS を使用する場合、次のオプションパラメータを設定できます。

```
aws_region = AWS_REGION
aws_profile = AWS_PROFILE_NAME
aws_await_snapshots_timeout = TIMEOUT_IN_SECONDS
```

`aws_profile` を使用する場合、AWS 資格情報ファイルのセクションの名前に設定する必要があります。`aws_profile` を使用しない場合、デフォルトのプロファイルが使用されます。資格情報ファイルが存在しない場合、資格情報は環境から取得されます。

`aws_region` が指定されている場合、AWS プロファイルで定義されているリージョンをオーバーライドします。

`aws_await_snapshots_timeout` が設定されていない場合、デフォルトの 3600 秒が使用されます。

## 16.9 スナップショットバックアップの取得

構成オプションが設定され、適切な資格情報を Barman で利用できるようになると、`barman backup` コマンドを使用してバックアップを取得できます。

Barman は、`barman check` コマンドの使用およびバックアップの開始時に、スナップショットバックアップの構成パラメーターを検証します。

`backup_method = snapshot` を使用する場合、次の引数/構成変数は使用できないことに注意してください。

| Command argument            | Config variable                  |
|-----------------------------|----------------------------------|
| N/A                         | <code>backup_compression</code>  |
| <code>--bwlimit</code>      | <code>bandwidth_limit</code>     |
| <code>--jobs</code>         | <code>parallel_jobs</code>       |
| N/A                         | <code>network_compression</code> |
| <code>--reuse-backup</code> | <code>reuse_backup</code>        |

スナップショットバックアップの管理とリカバリーに関する考慮事項を含む、スナップショットバックアップのより詳細な議論については、`cloud snapshots section in feature details` を参照してください。



## 第17章 Windowsベースのサーバーをセットアップする方法

WAL アーカイブとバックアップの両方でストリーミング接続を使用して、Windows で実行されている PostgreSQL サーバーをバックアップできます。

---

**重要:** この機能は、継続的インテグレーションシステムの一部ではないため、まだ実験的です。

---

ストリーミング接続のセットアップについては、前述のすべての手順に従います。

**警告:** 現時点では、Windows から Linux への `pg\_basebackup` の相互運用性はまだ実験的です。Windows サーバーからバックアップの取得に問題が発生し、PostgreSQL ロケールが英語でない場合、問題の可能な回避策は、英語でメッセージを発行するように PostgreSQL に指示することです。これは、次のパラメーターを `postgresql.conf` ファイルに配置することにより行うことができます。

`ini lc_messages = English` >これは問題を修正するために報告されています。

通常のようにサーバーをバックアップできます。

リモート復旧は Windows サーバーでサポートされていないため、Barman サーバーでクラスターをローカルに復旧し、Windows サーバー上のすべてのファイルをコピーするか、PostgreSQL サーバーと Barman サーバーで共有されるフォルダーを使用する必要があります。

さらに、PostgreSQL を実行するために選択したシステムユーザーが、復元されたデータにアクセスするために必要な権限があることを確認してください。基本的に、PostgreSQL データディレクトリを完全に制御する必要があります。



## 第18章 一般的なコマンド

Barman には多くのコマンドがあり、説明のために、スコープごとにそれらを整理できます。

**汎用コマンド**のスコープは、多くの PostgreSQL サーバーをバックアップできる Barman サーバー全体です。代わりに、**サーバーコマンド** は、指定されたサーバーでのみ動作します。**バックアップコマンド**は、特定のサーバーから取得したバックアップに対して動作します。

次のリストには、一般的なコマンドが含まれています。

### 18.1 cron

barman には、長時間実行デーモンまたはサービスファイルは含まれません `systemctl start`、`service start` などには何もありません。代わりに、`barman cron` サブコマンドは、`barman` のバックグラウンドでの定常状態のバックアップ操作を実行するために提供されています。

`cron` コマンドを使用して、WAL ファイルとバックアップの両方でメンテナンス操作を実行できます。

```
barman cron
```

---

**注釈:** このコマンドは `_cron script_` で実行する必要があります。 `barman cron` を 1 分ごとに実行するようにスケジュールすることをお勧めします。 `rpm` または `debian` パッケージを使用して Barman をインストールした場合、1 分ごとに実行される `cron` エントリが作成されます。

---

`barman cron` は、サーバーベースで WAL アーカイブ操作を同時に実行し、これにより、次のサーバーに保持ポリシーも適用されます。

- `retention_policy` は空でなく有効ではありません。
- `retention_policy_mode` を `auto` に設定します。

`cron` コマンドは、`receive-wal` コマンドを透過的に実行することにより、要求したサーバーで WAL ストリーミングが開始されるようにします。

`cron` コマンドによって開始された操作を停止するには、`cron` エントリをコメントアウトして実行します。

```
barman receive-wal --stop SERVER_NAME
```

`barman list-servers` をチェックして、すべてのサーバーを取得していることを確認することができます。

**注釈:** *barman cron* は、バックグラウンドでのメンテナンスタスクのみを実行し、スケジュールされたバックアップの実行は行いません。必要な定期的にスケジュールされたバックアップジョブは、たとえば *barman backup all* を実行する別の cron エントリで、別途スケジュールする必要があります。

---

## 18.2 診断する

*diagnose* コマンドは、診断とサポートの目的で役立つ JSON レポートを作成します。このレポートには、構成されているすべてのサーバーの情報が含まれています。

**注釈:** Barman 3.10.0 以降、オプションでコマンドに `--show-config-source` 引数を指定できます。その場合、Barman および Barman サーバーの構成オプションごとに、出力には構成値だけでなく、有効な値を提供する構成ファイルも含まれます。

---

**重要:** 診断が JSON で書かれ、その形式が機械可読であると考えられる場合でも、その構造はインターフェイスの一部とみなされません。フォーマットは、異なる Barman バージョン間で変更される場合があります。

---

## 18.3 list-servers

次の使用を使用して、バックアップシステム用に構成されたアクティブサーバーのリストを表示できます。

```
barman list-servers
```

`--minimal` オプションを使用して、機械可読出力を取得できます。

```
barman list-servers --minimal
```

## 第19章 Server コマンド

前のセクションで述べたように、サーバーコマンドは PostgreSQL サーバーまたは Barman のその領域で直接動作し、そのステータスの確認、メンテナンス操作の実行、バックアップの作成、および WAL アーカイブの管理に役立ちます。

### 19.1 archive-wal

`archive-wal` コマンドは、特定のサーバーの WAL ファイルに対してメンテナンス操作を実行します。この操作には、ストリーミング接続または `archive_command` またはその両方から受信した WAL ファイルの処理が含まれます。

---

**重要:** `archive-wal` コマンドは、直接呼び出すことができますが、`cron` 汎用コマンドから開始されるように設計されています。

---

### 19.2 バックアップ

`backup` コマンドは、指定されたサーバーの完全なバックアップ `_basebackup_` を取得します。新しいバックアップに対応する構成パラメーターをオーバーライドできる複数のオプションがあります。詳細については、マニュアルページを参照してください。

次の使用で、特定のサーバーの完全なバックアップを実行できます。

```
barman backup <server_name>
```

---

**Tip:** `barman backup all` を使用して、構成済みのすべてのサーバーを順次バックアップできます。

---

---

**Tip:** `barman backup <server_1_name> <server_2_name>` を使用して、`<server_1_name>` および `<server_2_name>` サーバーの両方をシークエンシャルにバックアップできます。

---

Barman でインクリメンタルバックアップを取る方法については、`incrementalbackupsection` を確認してください。

Barman 2.10 では、`backup` コマンドの `-w /--wait` オプションを導入しました。設定されると、Barman はバックアップの状態を一時的に `WAITING_FOR_WALS` に保存し、必要なすべての WAL ファイルがアー

カイクされるのを待ってから、状態を `DONE` に設定し、バックアップ後フックスクリプトを続行します。`--wait-timeout` オプションが提供される場合、Barman は指定された秒数の後に WAL ファイルの待機を停止し、状態は `WAITING_FOR_WALS` のままになります。`cron` コマンドは、不足している WAL ファイルがアーカイブされていることを確認し続け、バックアップに `DONE` としてラベル付けします。

## 19.3 check

`check` コマンドを使用して、特定のサーバーへの接続と構成の一貫性を確認できます。

```
barman check <server_name>
```

---

**Tip:** `barman check all` を使用して、構成されているすべてのサーバーを確認できます。

---

---

**重要:** `check` コマンドは、おそらく Barman が実装する最も重要な機能です。アラートおよび監視インフラストラクチャと統合することをお勧めします。`--nagios` オプションを使用すると、Nagios / Icinga のプラグインを簡単に作成できます。

---

## 19.4 config-update

`config-update` コマンドは、Barman でサーバーとモデルの構成を作成または更新するために使用されます。`config-update` コマンドを実行する構文は次のとおりです。

```
barman config-update <json_changes>
```

`json_changes` は、ドキュメントの配列を含む JSON 文字列である必要があります。各ドキュメントには、次のキーが含まれている必要があります。

- `scope` Barman サーバーまたは Barman モデルを作成または更新するかに応じて、`server` または `model` のいずれか。

また、``scope`` の値に応じて、次のキーのいずれかが含まれる必要があります。

- `server_name` `scope` が `server` の場合、このキーに Barman サーバー名を入力する必要があります。
- `model_name` `scope` が `model` の場合、このキーに Barman モデル名を入力する必要があります。

これらに加えて、各ドキュメントに 1 つ以上の Barman 構成オプションとそれらの目的の値を入力する必要があります。

これは、Barman サーバー `my_server` を `archiver=on` および `streaming_archiver=off` で更新する例です。

```
barman config-update \
    '[{"scope": "server", "server_name": "my_server", "archiver": "on",
    "streaming_archiver": "off"}]'
```

**注意** `barman config-update` コマンドは、`barman_home` の下に作成される `.barman.auto.conf` という名前のファイルに構成オプションを書き込みます。その構成ファイルはより優先され、Barman グローバル構成ファイル通常は `/etc/barman.conf` からの値と、`configuration_files_directory` ごとにインクルードファイル通常は `/etc/barman.d` のファイルからの値をオーバーライドします。後で、何らかの理由で、これらのファイルの構成オプションを手動で変更することにした場合は、そのことに注意してください。

## 19.5 config-switch

`config-switch` コマンドは、モデルを介して定義された構成オーバーライドのセットを Barman サーバーに適用するために使用されます。Barman サーバーの最終構成は、サーバーの構成と選択したモデルによって適用されたオーバーライドで構成されます。モデルはクラスター環境で特に役立つため、たとえば、フェールオーバーイベントに応答して使用できるさまざまな構成モデルを作成できます。

`config-switch` コマンドを介してモデルを適用する構文は次のとおりです。

```
barman config-switch <server_name> <model_name>
```

**注意** コマンドは `<model_name>` が存在し、`<server_name>` と同じ `cluster` に属している場合にのみ成功します。

**注** 一度にアクティブにできるモデルは 1 つだけです。異なるモデルでコマンドを 2 回実行した場合、最後に定義されたオーバーライドのみが適用されます。

サーバーの既存のアクティブモデルを適用解除するための構文は次のとおりです。

```
barman config-switch <server_name> --reset
```

これは、アクティブなモデルによって以前に設定されていたオーバーライドの適用解除を処理します。

**注** このコマンドは、特定の状況からの復旧にも役立ちます。以前に構成されたが、構成に存在しないアクティブなモデルを使用するサーバーがある場合。

## 19.6 generate-manifest

このコマンドは、バックアップがリモートで作成され、`pg_basebackup` が関係せず、`backup_manifest` ファイルがバックアップに存在しない場合に役立ちます。barman サーバーのバックアップを使用して、`backup_id` から `backup_manifest` ファイルを生成します。ファイルが既に存在する場合、`generate` コマンドは中止されます。

コマンドの例

```
barman generate-manifest <server_name> <backup_id>
```

backup\_id backup id stables のいずれかが使用できます。

このコマンドは、次のように post\_backup フックスクリプトとして使用することもできます。

```
post_backup_script=barman generate-manifest ${BARMAN_SERVER} ${BARMAN_BACKUP_ID}
```

## 19.7 get-wal

Barman では、ユーザーが get-wal コマンドを介して WAL アーカイブから\_xlog\_ファイルを要求できます。

```
barman get-wal [-o OUTPUT_DIRECTORY] [-j|-x] <server_name> <wal_id>
```

要求された WAL ファイルがサーバーアーカイブで見つかった場合、特に指定しない限り、非圧縮コンテンツは STDOUT に返されます。

get-wal コマンドでは、次のオプションが使用できます。

- -o を使用すると、ユーザーは Barman が要求された WAL ファイルをデポジットする宛先ディレクトリを指定できます
- -j は、bzip2 アルゴリズムを使用して出力を圧縮します
- -x は、gzip アルゴリズムを使用して出力を圧縮します
- -p SIZE は、要求されたファイルから開始して、アーカイブから WAL ファイルまでをピークします

リカバリオペレーション中に get-wal を使用し、Barman サーバーをサーバーの\_WAL ハブ\_に変換することが可能です。これは、get-wal 値を recovery\_options グローバル/サーバー構成オプションに追加することにより、自動的に達成できます。

```
recovery_options = get-wal
```

recovery\_options は、コンマ区切り値のリストを受け入れるグローバル/サーバーオプションです。リカバリオペレーション中にキーワード get-wal が存在する場合、Barman は restore\_command を設定してリカバリ構成を準備し、barman get-wal が必要な WAL ファイルを取得するために使用されるようになります。同様に、実行時に recover コマンドの--get-wal オプションを使用できます。

recovery\_options で get-wal が設定されているが、リカバリオペレーション中に必要ない場合、--no-get-wal オプションを recover コマンドとともに使用して、get-wal リカバリオプションを無効にできます。

これは、ローカルリカバリの restore\_command の例です。

```
restore_command = sudo -u barman barman get-wal SERVER %f > %p
```

get-wal コマンドは、常に barman ユーザーとして呼び出す必要があること、およびカタログから WAL ファイルを読み取るには正しい権限が必要であることに注意してください。これが、この例で sudo -u barman を使用する理由です。

リモートリカバリーに `recovery_options` から `get-wal` を設定すると、代わりに `barman-wal-restore` スクリプトを使用して `restore_command` が生成されます。`barman-wal-restore` は、SSH 接続エラーを管理する、より復元力の高いシェルスクリプトです。

このスクリプトには、WAL ファイルの自動圧縮および解凍や、PostgreSQL がいずれかの WAL ファイルを適用しているときに次の WAL ファイルを取得できる `_peek` 機能など、多くの便利なオプションが含まれています。これは、PostgreSQL と Barman 間の帯域幅の使用を最適化する優れた方法です。

`barman-wal-restore` は、`barman-cli` パッケージで入手できます。

これは、リモートリカバリーの `restore_command` の例です。

```
restore_command = barman-wal-restore -U barman backup SERVER %f %p
```

SSH を使用して Barman サーバーと通信するため、`postgres` ユーザーがバックアップサーバーに `barman` としてログインするには、SSH キー認証が必要です。SSH デフォルトの 22 以外のポートを使用する必要がある場合は、`--port` オプションを追加して、SSH 接続に使用するポートを指定できます。

`barman-wal-restore` が Barman サーバーに接続できること、および必要な PostgreSQL サーバーが次のコマンドで WAL ファイルを送信するように Barman で構成されていることを確認できます。

```
barman-wal-restore --test backup pg DUMMY DUMMY
```

`backup` は Barman がインストールされているホストであり、`pg` は Barman で構成された PostgreSQL サーバーの名前であり、`DUMMY` はプレースホルダーです。`barman-wal-restore` には、WAL ファイル名と宛先ディレクトリの 2 つの引数が必要ですが、これらは無視されます。

すべてが正しく構成されている場合、次の出力が表示されるはずです。

```
Ready to retrieve WAL files from the server pg
```

`barman-wal-restore` コマンドの詳細については、PostgreSQL サーバーで `man barman-wal-restore` と入力してください。

## 19.8 list-backups

次の使用で、特定のサーバーで使用可能なバックアップのカタログをリストできます。

```
barman list-backups <server_name>
```

**Tip:** サーバー名として ``all`` を使用して、すべてのサーバーのバックアップの完全なリストを要求できます。

機械読み取り可能な出力を取得するには、`--minimal` オプションを使用できます。JSON 形式で出力を取得するには、`--format=json` オプションを使用できます。

## 19.9 resume-xlogdb

all ショートカットを使用して、特定のサーバーまたはすべてのサーバーの WAL アーカイブのコンテンツをいつでも再生成できます。WAL アーカイブは `xlog.db` ファイルに含まれており、Barman によって管理されるすべてのサーバーには独自のコピーがあります。

`xlog.db` ファイルは、`rebuild-xlogdb` コマンドでリビルドできます。これにより、アーカイブされたすべての WAL ファイルがスキャンされ、アーカイブのメタデータが再生成されます。

例

```
barman rebuild-xlogdb <server_name>
```

## 19.10 resume-wal

このコマンドは、ストリーミングプロトコルを使用して PostgreSQL ストリーミング接続から WAL ファイルを受信する `receive-wal` プロセスを管理します。

### 19.10.1 レシーブウォールプロセス管理

オプションを指定せずにコマンドを実行すると、`receive-wal` プロセスが開始されます。このコマンドは、`pg_receivewal` PostgreSQL コマンドに基づいています。

```
barman receive-wal <server_name>
```

---

**注釈:** `receive-wal` コマンドはフォアグラウンドプロセスです。

---

`--stop` オプションを使用してコマンドを実行すると、現在実行中の `receive-wal` プロセスが停止されます。

`receive-wal` プロセスは、ステータスファイルを使用して、トランザクションログの最後に書き込まれたレコードを追跡します。ステータスファイルをクリーニングする必要がある場合、`--reset` オプションを使用できます。

---

**重要:** レプリケーションスロットを使用していない場合は、`wal_keep_segments` または PostgreSQL バージョン 13.0 以降では `wal_keep_size` の値に依存します。データベースのワークロードの高いピークでは、`receive-wal` プロセスが遅延し、非同期になる場合があることに注意してください。予防措置として、Barman では現在、ユーザーが `--reset` オプションを使用してコマンドを手動で実行して、誤った仮定を避けることを必要としています。

---

## 19.10.2 レプリケーションスロット管理

`receive-wal` プロセスは、Barman が WAL アーカイブ手順に必要なレプリケーションスロットを作成またはドロップするのに役立ちます。

`--create-slot` オプションを使用すると、`slot_name` 構成オプションの後に名前付けられたレプリケーションスロットが PostgreSQL サーバーに作成されます。

`--drop-slot` では、以前のレプリケーションスロットが削除されます。

## 19.11 レプリケーションステータス

`replication-status` コマンドは、Barman サーバー構成されている場合の `receive-wal` プロセスを含む、PostgreSQL サーバーに現在接続されているストリーミングクライアントのステータスを報告します。

次のようにコマンドを実行できます。

```
barman replication-status <server_name>
```

**Tip:** サーバー名として `all` を使用して、すべてのサーバーのレプリカの完全なステータスレポートを要求できます。

機械可読な出力を得るには、`--minimal` オプションを使用できます。

## 19.12 show-servers

次の使用で、特定のサーバーの構成パラメーターを表示できます。

```
barman show-servers <server_name>
```

**Tip:** サーバー名として `all` を使用して、完全な構成レポートを要求できます。

## 19.13 ステータス

`status` コマンドは、PostgreSQL サーバーまたはサーバー名として `all` を使用する場合はすべてのサーバーのライブ情報とステータスを表示します。

```
barman status <server_name>
```

## 19.14 switch-wal

このコマンドは、PostgreSQL サーバーを別のトランザクションログファイル WAL に切り替え、現在のログファイルを閉じ、受信、アーカイブできるようにします。

```
barman switch-wal <server_name>
```

最後のトランザクションログファイルのスイッチ以降、トランザクションアクティビティが存在しない場合、`--force` オプションを使用してスイッチを強制する必要があります。

`--archive` オプションは、xlog スイッチの後に WAL アーカイブをトリガーするように Barman に要求します。デフォルトでは、30 秒のタイムアウトが適用されますこれは `--archive-timeout` で変更できます。WAL ファイルを受信しない場合、エラーが返されます。

---

**注釈:** Barman 2.1 および 2.2 では、このコマンドは ``switch-xlog`` と呼ばれていました。PostgreSQL 10 以降との名前付けの一貫性のために名前が変更されました。

---

## 19.15 verify-backup

`verify-backup` コマンドは、バックアップから `backup_manifest` ファイルを使用し、それに対して `pg_verifybackup` を実行します。

```
barman verify-backup <server_name> <backup_id>
```

このコマンドは、`pg_verifybackup <path_to_backup_manifest> -n PG>=13` で使用可能 `pg_verifybackup` バックアップサーバーにインストールする必要があります。rsync バックアップの場合、`generate-manifest` コマンドで使用できます。

`backup_id` `backup id` `stables` のいずれかが使用できます。

## 第20章 バックアップコマンド

バックアップコマンドは、Barman のバックアップカタログに既に存在するバックアップに直接動作するコマンドです。

---

**注釈:** バックアップ ID は ``barman list-backups <server_name>`` で取得できることを覚えてください

---

### 20.1 バックアップIDのショートカット

Barman では、特別なキーワードを使用して特定のバックアップを識別できます。

- `last/latest` カタログ内の最新のバックアップを識別します
- `first/oldest` カタログ内の最も古いバックアップを識別します
- `last-failed` カタログ内の最新の失敗したバックアップを識別します

Barman コマンドでこれらのキーワードを使用すると、サーバーのバックアップの正確な ID を知らなくてもアクションを実行できます。例、次のことを発行できます。

```
barman delete <server_name> oldest
```

カタログで使用可能な最も古いバックアップを削除し、ディスク領域を再利用します。

さらに、バックアップが `--name <friendly_name>` オプションで取得された場合、バックアップ ID の代わりにフレンドリ名を使用して、その特定のバックアップを参照できます。

### 20.2 check-backup

バージョン 2.5 以降、フルバックアップの一貫性のために必要なすべての WAL ファイルが、`check-backup` コマンドを使用して `barman` によって正しくアーカイブされたことを確認できます。

```
barman check-backup <server_name> <backup_id>
```

---

**重要:** このコマンドは、`cron` によって、および ``backup`` オペレーションの最後に自動的に呼び出されます。これは、通常の状況では、実行する必要がないことを意味します。

---

バックアップの開始から終了までの 1 つ以上の WAL ファイルがまだアーカイブされていない場合、barman はバックアップに `WAITING_FOR_WALS` のラベルを付けます。cron コマンドは、不足している WAL ファイルがアーカイブされていることを継続して確認し、バックアップに `DONE` としてラベル付けします。

バックアップの最後に最初に必要な WAL ファイルが欠落している場合、このようなバックアップは `FAILED` としてマークされます。したがって、バックアップ操作を実行する前に、WAL アーカイブストリーミングまたは `archive_command` を介して適切に動作していることを確認することが重要です。特にスタンバイサーバーからバックアップする場合。

## 20.3 削除

次の方法で特定のバックアップを削除できます。

```
barman delete <server_name> <backup_id>
```

`delete` コマンドは、[バックアップ ID のショートカット](#) を受け入れてバックアップを識別します。

---

**重要:** 指定されたバックアップに依存するブロックレベルのインクリメンタルバックアップがある場合、チェーン内に親が見つからない場合のリカバリーに事実上使用できなくなるため、これらのバックアップとそのすべての依存関係もこの操作中に削除されます。

---

## 20.4 keep

サーバーのリテンションポリシーを超えて保持したいバックアップがある場合は、次の方法でアーカイブバックアップを作成できます。

```
barman keep <server_name> <backup_id> [--target TARGET, --status, --release]
```

---

**注釈:** バックアップシステムの完全性を保証するため、ブロックレベルのインクリメンタルバックアップは Barman の `keep` アノテーションを使用できません。この制限は、ブロックレベルのインクリメンタルバックアップが互いに依存する方法によるものです。このようなバックアップで `keep` アノテーションを使用すると、孤立したバックアップが発生する可能性があります。これは、特定のバックアップが必要な親バックアップなしで存在する可能性があることを意味します。 > 簡単に言えば、ブロックレベルのインクリメンタルバックアップに `keep` アノテーションを適用できる場合、バックアップチェーンの一部が必要な先行者なしで保持されるリスクがあります。この状況では、適切に復元するために必要な重要な親バックアップが欠落しているため、役に立たない、または完全でないバックアップが作成される可能性があります。

---

TARGET の可能な値は次のとおりです。

- `full` バックアップは、常に最新のポイント・インタイムにリカバリーするために使用できます。これを行うために、Barman は、バックアップと後続のすべての WAL の一貫性を保証するために必要なすべての WAL を保持します。

- **standalone** バックアップは、サーバーをバックアップを取った時の状態に復旧するためにのみ使用できます。Barman は、バックアップの一貫性を保証するために必要な WAL のみを保持します。

`--status` オプションが提供されている場合、Barman はバックアップのアーカイブステータスを報告します。これは、アーカイブバックアップの場合は **full** または **standalone**、またはアーカイブとしてフラグが立てられていないバックアップの場合は **nokeep** のリカバリターゲットになります。

`--release` オプションが提供されている場合、Barman はこのバックアップから **keep** フラグを解放します。これにより、アーカイブステータスが削除され、直接または保持ポリシーによって削除できるようになります。

バックアップがアーカイブバックアップとしてフラグが設定されると、Barman の動作が次のように変更されます。

- **barman delete** を使用して ID でそのバックアップを削除しようとする、失敗します。
- リテンションポリシーは、そのバックアップを **OBSOLETE** として考慮しないため、**barman cron** はそのバックアップを削除することはありません。
- そのバックアップに必要な WAL は、永久に保持されます。指定されたリカバリターゲットが **full** の場合、後続の **すべて** の WAL も保持されます。

これは、**barman keep <server\_name> <backup\_id> --release** で **keep** フラグを削除することにより元に戻すことができます。

**警告:** サーバーのリテンションポリシーで `standalone` アーカイブバックアップが必要とされなくなると、**barman cron** は、そのバックアップと次の最新のバックアップの `begin_wal` 値間の WAL を削除します。これは、ターゲットを `full` から `standalone` に変更するのは安全ですが、最新のポイントへのリカバリに必要な WAL がまだ残っているという保証がないため、ターゲットを `standalone` から `full` に変更するのは安全ではないことを意味します利用できます。

## 20.5 list-files

次の使用で、特定のバックアップのファイルベースバックアップと必要な WAL ファイルをリストできます。

```
barman list-files [--target TARGET_TYPE] <server_name> <backup_id>
```

`--target TARGET_TYPE` オプションを使用すると、特定のバックアップのリストの内容を選択できます。

**TARGET\_TYPE** の可能な値は次のとおりです。

- **data** データファイルをリストします
- **standalone** 必要な WAL ファイルを含むベースバックアップファイルをリストします
- **wal** ベースバックアップの先頭から次のバックアップの先頭またはログの最後まですべての WAL ファイルをリストします
- **full data + wal** と同じ

TARGET\_TYPE のデフォルト値は `standalone` です。

---

**重要:** `list-files` コマンドは外部ツールとの相互作用を促進するため、Barman をアーカイブプロシージャに統合するのに非常に役立ちます。

---

## 20.6 recovery

`recover` コマンドは、`backup` コマンドを使用してバックアップが実行された後、サーバー全体をリカバリするために使用されます。

これは、次のようなコマンドを発行して実現されます。

```
barman@backup$ barman recover <server_name> <backup_id> /path/to/recover/dir
```

---

**重要:** PostgreSQL インスタンスが実行されているターゲットデータディレクトリを使用して `recover` コマンドを発行しないでください。その場合、リカバリを発行する前に忘れずに停止してください。これは、テーブルスペースディレクトリにも適用されます。

---

リカバリの実行の最後に、選択したバックアップはローカルにリカバリされ、宛先パスには PostgreSQL インスタンスを起動するために使用する準備ができたデータディレクトリが含まれています。

---

**重要:** ユーザー `barman` としてこのコマンドを実行すると、データベースのスーパーユーザーになります。

---

バックアップの特定の ID は、[\*list-backups\*](#) を使用して取得できます。

コマンド。

---

**重要:** Barman は、現在 PGDATA 内のシンボリックリンクを追跡していません。pg\_tblspc 内のテーブルスペースを除きます。システム管理者は、シンボリックリンクを追跡し、元の場所に復元する必要がある場合に、ディザスターリカバリー計画/手順に追加することをお勧めします。

---

`recovery` コマンドには、コマンドの動作を変更するいくつかのオプションがあります。

## 20.6.1 リモートリカバリー

`--remote-ssh-command <COMMAND>` オプションを `recovery` コマンドの呼び出しに追加します。これにより、Barman は提供されたコマンドを使用してリモートホストに接続し、リモートサーバーでコピーを実行できます。

---

**注釈:** `postgres` ユーザーを使用して、リモートホストでリカバリーを実行することをお勧めします。

---



---

**重要:** PostgreSQL インスタンスが実行されているターゲットデータディレクトリを使用して `recover` コマンドを発行しないでください。その場合、リカバリーを発行する前に忘れずに停止してください。これは、テーブルスペースディレクトリにも適用されます。

---

リモートリカバリーの既知の制限は次のとおりです。

- Barman は、Barman 構成の `recovery_option` パラメーターで `get-wal` コマンドが指定されていない限り、システム一時ディレクトリに少なくとも 4GB の空き領域を必要とします。
- Barman とリモートホスト間の SSH 接続は、公開鍵交換認証方法を使用する必要があります\*\*
- リモートユーザー\*\*は、宛先ディレクトリにバックアップのディレクトリ構造を作成できる必要があります。
- リモートサーバーには、ベースバックアップと復旧に必要な WAL ファイルを含めるための十分な空き領域が必要です。

## 20.6.2 テーブルスペースのリマッピング

Barman は、`-tablespace` オプションを指定して `recover` コマンドを使用して、1 つ以上のテーブルスペースを自動的に再マップできます。このオプションは、`NAME:DIRECTORY` フォーマットを使用して引数として値のペアを受け入れます。

- `NAME` は、テーブルスペースの識別子です
- `DIRECTORY` は、テーブルスペースの新しい宛先パスです

宛先ディレクトリが存在しない場合、Barman はそれを作成しようとします必要な権限を持っていることを前提としています。

### 20.6.3 ポイント インタイム リカバリー

Barman は、PostgreSQL のポイントインタイムリカバリー PITR をラップし、タイムスタンプ、復元ラベル、またはトランザクション ID として、リカバリターゲットを指定できます。

---

**重要:** 特定のバックアップの最も古い PITR は、ベースバックアップ自分自身の終了です。バックアップの開始から終了までの任意の時点で復旧する場合は、以前のバックアップを使用する必要があります。Barman 2.3 から、`--target-immediate` オプションを使用して、一貫性が達成されたときにリカバリーを終了できます。

---

リカバリターゲットは、次の相互排他オプションのいずれかを使用して指定できます。

- `--target-time TARGET_TIME` タイムスタンプを指定する
- `--target-xid TARGET_XID` トランザクション ID を指定する
- `--target-lsn TARGET_LSN` ログシーケンス番号 LSN を指定する - PostgreSQL 10 以降が必要
- `--target-name TARGET_NAME` `pg_create_restore_point(name)` ファンクションで以前に作成した名前付けリストアポイントを指定します
- `--target-immediate` 一貫した状態に到達すると、リカバリは終了しますつまり、ベースバックアッププロセスの終了

---

**重要:** `time`、`XID`、および `LSN` を介したリカバリターゲットは、バックアップの終了後である必要があります\*\*。バックアップの開始と終了の間のポイントにリカバリする場合は、カタログ内の以前のバックアップからリカバリする必要があります。

---

---

**重要:** `--target-time` の使用時にタイムゾーンが指定されない場合、Barman ホストのタイムゾーンが使用されます。

---

`--exclusive` オプションを使用して、リカバリターゲットの直前に停止するか、直後に停止するかを指定できます。

Barman では、`--target-tli` オプションを使用して、リカバリのターゲットタイムラインを指定できます。これは、数値のタイムライン ID または特別な値 `latest` WAL アーカイブの最新のタイムラインにリカバリするためおよび `current` バックアップが取られたときに現在のタイムラインにリカバリするのいずれかに設定できます。このオプションを省略すると、PostgreSQL バージョン 12 以降は `latest` タイムラインにリカバリし、PostgreSQL バージョン 12 より前のバージョンは `current` タイムラインにリカバリします。「始める前に」セクションで説明しているように、PostgreSQL ドキュメントでタイムラインの詳細を見つけることができます。

Barman 2.4 では、`--target-action` オプションのサポートを導入し、次の値を受け入れます。

- `shutdown` 復旧目標に到達すると、PostgreSQL がシャットダウンされます

- **pause** 復旧目標に到達すると、PostgreSQL が一時停止状態で起動され、ユーザーはインスタンスを検査できます
- **promote** 復旧目標に到達すると、PostgreSQL はリカバリを終了し、マスターとして昇格します

---

**重要:** デフォルトでは、ターゲットアクションは定義されていません下位互換性のため。 `--target-action` オプションでは、ポイントインタイムリカバリーターゲットを指定する必要があります。

---

上記の設定の詳細については、[PostgreSQL documentation on recovery target settings][target] を参照してください。

Barman 2.4 は、`recover` コマンドの `--standby-mode` オプションを追加します。これを指定すると、`standby.signal` ファイル 12 より前の PostgreSQL バージョンから、または生成されたリカバリ構成に `standby_mode = on` を追加することにより、リカバリされたインスタンスをスタンバイとして適切に構成します。

Postgresql スタンバイモードの詳細については、公式ドキュメントで入手できます。

- Postgres 11 以前のバージョンの場合 [in the standby section of PostgreSQL documentation](#) 。
- PostgreSQL 12 以降のバージョン [in the replication section of PostgreSQL documentation](#) 。

---

**重要:** リカバリ中に `--standby-mode` を使用する場合、ユーザーは復旧したインスタンスの構成を変更する必要があります、Barman からの WAL ファイルのレプリケーションが正常に完了すると、復旧したサーバーはプライマリに接続できます。リカバリされたインスタンスバージョンが 11 以下の場合、これは `'primary_conninfo'` パラメーターを `'recovery.conf'` ファイルに追加することにより実現されます。リカバリされたインスタンスバージョンが 12 以上の場合、`primary_conninfo` パラメーターを `'postgresql.conf'` ファイルに追加する必要があります。

---

## 20.6.4 Barman サーバーから WAL の取得

`barman recover` コマンドは、オプションで、リカバリ中に Barman から WAL を取得するように PostgreSQL を構成できます。これは、`get-wal` section で説明しているように、`recovery_options` グローバル/サーバー構成オプションを `'get-wal'` に設定することにより有効にします。`recovery_options` が設定されていないか空の場合、Barman は代わりに `barman recover` コマンドの実行中にリカバリに必要な WAL をコピーします。

`--get-wal` および `--no-get-wal` オプションは、`recovery_options` で定義された動作をオーバーライドするために使用できます。`--get-wal` を `barman recover` とともに使用して、Barman サーバーからの WAL の取得を有効にするか、`--no-get-wal` を使用して無効にします。

---

**重要:** これらのターゲット [`--target-xid`、`--target-name`、`--target-time`] のいずれかと組み合わせて `--no-get-wal` を使用してリカバリする場合、Barman は WAL アーカイブ全体を Barman ホストからリカバリホストにコピーします。これにより、構成されたターゲットに到達するために必要なすべての WAL が既に Barman にアーカイブされていると仮定すると、少なくともこれらの WAL が Postgres で利用できること

---

を保証します。これは、現在 Barman で、これらの種類の復旧ターゲットを達成するために Postgres が必要な WAL を決定する信頼性の高いおよび/またはパフォーマンスの方法がないために発生します。

---

### 20.6.5 圧縮バックアップのリカバリー

`backup_compression` オプションを使用してバックアップが圧縮されている場合、`barman recover` は、リカバリー時にバックアップを解凍できます。これはマルチステップのプロセスです。

1. 圧縮されたバックアップファイルは、`Rsync` を使用してローカルまたはリモートサーバーのステージングディレクトリにコピーされます。
2. 圧縮ファイルはターゲットディレクトリに解凍されます。
3. Barman による特別な処理を必要とする構成ファイルは、復旧先からコピーされ、必要に応じて分析または編集され、`Rsync` を使用して復旧先にコピーされます。
4. バックアップのステージングディレクトリが削除されます。

`barman` は、展開される環境について何も知らないため、`recovery_staging_path` オプションに依存して、ステージングディレクトリの適切な場所を選択します。

`backup_compression` オプションを使用している場合、グローバル/サーバー構成で `recovery_staging_path` を設定するか、`barman recover` コマンドで `--recovery-staging-path` オプションを使用する必要があります。これらのいずれも行わず、圧縮バックアップを復元しようとすると、Barman は適切な場所を推測するのではなく、失敗します。

### 20.6.6 ブロックレベルのインクリメンタル バックアップのリカバリー

バックアップがブロックレベルのインクリメンタルの場合、`barman recover` は、`pg_combinebackup` を介したリカバリーにバックアップのチェーンを結合できます。バックアップのチェーンは、完全なバックアップからリカバリーのために要求されたものに進むツリーブランチです。これはマルチステップのプロセスです。

1. バックアップのチェーンは、新しい合成バックアップに結合されます。リカバリするインクリメンタルバックアップの ID を使用した名前付けのフォルダーが、`pg_combinebackup` を使用してローカルサーバーの特定のステージングディレクトリ内に作成されます。あらゆる種類のリカバリーローカルまたはリモートの場合、合成バックアップは `barman` サーバーにローカルに作成されます。
2. リモートリカバリーの場合、コンテンツは `Rsync` を使用して最終的な宛先にコピーされます。それ以外の場合、ローカルリカバリーの場合、コンテンツは最終的な場所に移動されるだけです。
3. 提供されたステージングディレクトリ内に作成され、復旧するインクリメンタルバックアップの ID で名前付けられたフォルダーは、復旧プロセスの最後に削除されます。

ブロックレベルのインクリメンタルバックアップからリカバリーする場合、したがって、グローバル/サーバー構成で `local_staging_path` を設定するか、`barman recover` コマンドで `--local-staging-path` オプションを使用する必要があります。これらのことのいずれも行わず、このようなバックアップを回復しようとすると、Barman は適切な場所を推測するのではなく、失敗します。

---

**重要:** チェーン内のバックアップのいずれかがチェックサムを無効にして取得されたが、最終的なバックアップがチェックサムを有効にして取得された場合、結果のディレクトリには無効なチェックサムを含むページが含まれる場合があります。 *Follow up the `limitations` section in `pg_basebackup` documentation* <<https://www.postgresql.org/docs/17/app-pgcombinebackup.html>>`\_。

---

## 20.7 show-backup

次の使用を使用して、特定のサーバーの特定のバックアップで使用可能な情報をすべて取得できます。

```
barman show-backup <server_name> <backup_id>
```

show-backup コマンドは、**バックアップ ID のショートカット**を受け入れてバックアップを識別します。



## 第21章 機能の詳細

このセクションでは、いくつかの Barman 機能を紹介し、それらの適用性とそれらを使用するために必要な構成について説明します。

Barman 構成で作業する多くのシナリオを作成できるため、このリストは包括的なものではありません。それにもかかわらず、一般的なパターンについて議論することは役立ちます。

### 21.1 バックアップ機能

#### 21.1.1 インクリメンタル バックアップ

インクリメンタルバックアップは、既存のバックアップを参照として使用して、PostgreSQL サーバーから必要なデータの変更のみをコピーするバックアップの種類です。\_WAL 継続的アーカイブ\_によって実装される差分バックアップと混同しないでください。

Barman でのインクリメンタルバックアップの主な目的は次のとおりです。

- フルバックアップ処理にかかる時間の削減
- いくつかの定期的なバックアップが占有するディスク領域を削減します**データ重複排除**

Barman は現在、**ファイルレベルのインクリメンタルバックアップ** `rsync` を使用および **ブロックレベルのインクリメンタルバックアップ** `pg_basebackup` を使用をサポートしています。

---

**注釈:** さまざまなバックアップタイプのインクリメンタルバックアップは、現在互換性がありません。つまり、ブロックレベルのインクリメンタルバックアップは `rsync` バックアップで取得できず、ファイルレベルのインクリメンタルバックアップはストリーミングバックアップ `pg_basebackup` で取得できません。

---

#### 21.1.2 ファイルレベルのインクリメンタル バックアップ

この機能は `rsync` および `[hardlink][8]` に大きく依存しているため、基になるオペレーティングシステムとバックアップデータが存在するファイルシステムの両方でサポートされている必要があります。

主な概念は、後続のベースバックアップが、前回のバックアップから変更されていないファイルを共有し、ディスク使用量の関連する節約につながるということです。これは、VLDB コンテキストと、高い割合の\_読み取り専用の履歴テーブル\_を含むデータベースに特に当てはまります。

`Rsync` インクリメンタルバックアップは、`barman backup` コマンドを透過的に管理する `reuse_backup` と呼ばれるグローバル/サーバーオプションを介して有効にできます。3つの値を受け入れます。

- `off` 標準のフルバックアップデフォルト
- `link` サーバーの最後のバックアップを再利用し、変更されていないファイルのハードリンクを作成することによるファイルレベルのインクリメンタルバックアップ バックアップ領域と時間の削減のため
- `copy` サーバーの最後のバックアップを再利用し、変更されていないファイルのコピーを作成することによる、ファイルレベルのインクリメンタルバックアップバックアップ時間の短縮のためのみ

最も一般的なシナリオは、次のように `reuse_backup` を `link` に設定することです。

```
reuse_backup = link
```

これをグローバルレベルで設定すると、すべてのサーバーのインクリメンタルバックアップが自動的に有効になります。

最後の注意として、ユーザーは `barman backup` コマンドの `--reuse-backup` ランタイムオプションを介して `reuse_backup` オプションの設定をオーバーライドできます。同様に、実行時オプションは3つの値 `off`、`link`、および `copy` を受け入れます。たとえば、次のとおりに1回限りのインクリメンタルバックアップを実行できます。

```
barman backup --reuse-backup=link <server_name>
```

---

**注釈:** Postgres のブロックレベルのインクリメンタルバックアップとは異なり、Rsync ファイルレベルのインクリメンタルバックアップは自分自身で独立しています。つまり、重複排除に以前のバックアップを再利用したバックアップは、親バックアップが削除された場合にまったく侵害されません。前述のように、Rsync バックアップの重複排除はハードリンクを使用して実装されるため、以前に再利用されたバックアップが削除されても、他のバックアップと共有されたファイルはディスクに残り、これらのファイルを使用した最後のバックアップも削除された場合にのみ削除されます。また、以前の完全バックアップが削除されたときに完全バックアップを取る必要がないことも意味します。各バックアップにはすべてのファイルが含まれ、心配せずに再利用できます。同様に、`reuse_backup = off` を使用して、rsync で完全なバックアップを取得する必要はありません。`reuse_backup = link` で取得される最初のバックアップである場合、ハードリンクを作成する既存のファイルがないため、本質的に `off` のように動作します。

---

---

**重要:** `reuse_backup` オプションは、バックアップ方法として `rsync` または `local-rsync` とともに使用する必要があります。

---

### 21.1.3 ブロックレベルのインクリメンタル バックアップ

バージョン 3.11 以降、Barman は、ネイティブ [incremental backup support introduced in PostgreSQL 17](#) を活用したブロックレベルのインクリメンタルバックアップのサポートを導入しました。

ブロックレベルのインクリメンタルバックアップでは、重複排除はデータブロックレベル PostgreSQL のページで発生します。これは、最後のバックアップ以降に変更されたページのみを保存する必要があることを意味し、特に分散書き込みパターンを使用する大規模なデータベースの場合、より効率的なオプションになります。PostgreSQL では、この機能は [WAL Summarization](#) を使用して実装されるため、使用するにはデータベースサーバーで `summarize_wal` を有効にする必要があります。

バックアップコマンドの実行時に `--incremental` オプションを使用して、Barman でブロックレベルのインクリメンタルバックアップを実行できます。引数としてバックアップ ID またはバックアップ ID ショートカットを受け入れます。これは、重複排除の親として使用されるカタログ内の以前のバックアップ完全またはインクリメンタルを参照します。さらに、`last-full` または `latest-full` を使用して、カタログ内の最新の適格なフルバックアップを参照することもできます。

```
barman backup --incremental <backup_id> <server_name>
```

Barman でブロックレベルのインクリメンタルバックアップを実行できるようにするには、次のことを行う必要があります。

- PostgreSQL 17 以降。
- `summarize_wal` を有効にします。
- バックアップ方法として `postgres` を使用します。

**重要:** ブロックレベルのインクリメンタルバックアップ間で ``data_checksums`` を有効にする場合、分岐したチェックサム構成はリカバリ中に問題が発生する可能性があるため、新しいフルバックアップを取ることをお勧めします。

### 21.1.4 帯域幅の使用を制限する

`bandwidth_limit` オプショングローバル/サーバーごとを使用して、1 秒あたりの最大キロバイト数を指定することにより、I/O 帯域幅の使用を制限できます。デフォルトでは、制限なしを意味する 0 に設定されています。

**重要:** `bandwidth_limit` オプションは ``postgres`` バックアップ方法でサポートされていますが、`tablespace_bandwidth_limit` オプションは ``rsync`` を使用する場合にのみ使用できます。

複数の表領域があり、1 つ以上の表領域でバックアッププロシージャの I/O ワークロードを制限したい場合、`tablespace_bandwidth_limit` オプショングローバル/サーバーごとを使用できます。

```
tablespace_bandwidth_limit = tname:bwlimit[, tname:bwlimit, ...]
```

このオプションは、テーブルスペース名と帯域幅制限キロバイト/秒で構成されるペアのコンマ区切りリストを受け入れます。

サーバーをバックアップするときに、Barman は上記のオプションで既存のテーブルスペースを見つけようとします。見つかった場合、指定された帯域幅制限が適用されます。それ以外の場合、そのサーバーのデフォルトの帯域幅制限が適用されます。

### 21.1.5 ネットワーク圧縮

圧縮を使用して、転送されるデータのサイズを削減することが可能です。`network_compression` オプションでグローバル/サーバーごとを使用して有効にできます。

---

**重要:** `network_compression` オプションは、`postgres` バックアップ方法では使用できません。

---

```
network_compression = true|false
```

このオプションを `true` に設定すると、ネットワーク転送中のデータ圧縮がバックアップとリカバリーの両方で有効になります。デフォルトでは、`false` に設定されます。

### 21.1.6 バックアップ圧縮

Barman は、`pg_basebackup` の圧縮機能を使用して、バックアッププロセス中にバックアップデータを圧縮できます。これは、`backup_compression` 構成オプションでグローバル/サーバーごとを使用して有効にできます。

---

**重要:** `backup_compression` およびこのセクションで説明するその他のオプションは、`rsync` または ``local-rsync`` バックアップ方法では使用できません。`postgres` バックアップ方法の場合のみ。

---

### 21.1.7 圧縮アルゴリズム

このオプションを設定すると、`pg_basebackup` は指定された圧縮アルゴリズムを使用してバックアップを圧縮します。現在、Barman でサポートされているアルゴリズムは `gzip`、`lz4`、`zstd`、および `none` です。`none` 圧縮アルゴリズムは、非圧縮のアーカイブを作成します。

```
backup_compression = gzip|lz4|zstd|none
```

Barman では、選択した圧縮アルゴリズムの CLI ユーティリティを Barman サーバーと PostgreSQL サーバーの両方で利用できる必要があります。CLI ユーティリティは、圧縮バックアップからバックアップラベルを抽出し、リカバリ中に PostgreSQL サーバーでバックアップを解凍するために使用されます。これらは、Debian、Ubuntu、RedHat、CentOS、および SLES システムに `gzip`、`lz4`、および `zstd` という名前のシステムパッケージを介してインストールできます。

---

**注釈:** Ubuntu 18.04bionic では、`liblz4-tool` パッケージで ``lz4`` ユーティリティを利用できます。

---



---

**注釈:** `zstd` バージョンは 1.4.4 以降である必要があります。Debian 10 Buster、Ubuntu 18.04 bionic、および SLES 12 の ``zstd`` システムパッケージは、以前のバージョンをインストールします。`backup_compression = zstd` はこれらのパッケージでは動作しません。

---



---

**注釈:** `lz4` および ``zstd`` は、PostgreSQL バージョン 15 以降でのみ使用できます。

---



---

**重要:** `backup_compression` を使用している場合、`barman recover` が圧縮バックアップをリカバリできるように ``recovery_staging_path`` も設定する必要があります。[圧縮バックアップのリカバリー](#) を参照してください。

---

詳細については、セクションをご覧ください。

### 21.1.8 圧縮ワーカー

このオプションのパラメーターを使用すると、マルチプルのスレッドを使用して圧縮して、圧縮速度を向上させることができますデフォルトは 0 です。

```
backup_compression_workers = 2
```

---

**注釈:** このオプションは、`zstd` 圧縮でのみ使用できます。

---



---

**注釈:** `zstd` バージョンは 1.5.0 以降である必要があります。またはマルチスレッドオプションでコンパイルされた 1.4.4 以上。

---

### 21.1.9 圧縮レベル

圧縮レベルは、`backup_compression_level` オプションを使用して指定できます。これは、`backup_compression` で指定された圧縮アルゴリズムでサポートされている整値に設定する必要があります。定義されていない場合、圧縮アルゴリズムのデフォルト値が使用されます。

`none` 圧縮は `backup_compression_level=0` のみをサポートしています。

---

**注釈:** `backup_compression_level` 使用可能な値とデフォルト値は、使用される圧縮アルゴリズムによって異なります。詳細については、圧縮アルゴリズムのドキュメントを確認してください。

---

---

**注釈:** 15 より前の PostgreSQL バージョンでは、`gzip` は ``backup_compression_level=0`` をサポートしました。デフォルトの圧縮レベルを使用した結果

---

### 21.1.10 圧縮場所

PostgreSQL バージョン 15 以降で Barman を使用する場合、サーバーで発生する圧縮つまり PostgreSQL がバックアップを圧縮するか、クライアントつまり `pg_basebackup` がバックアップを圧縮するように指定できます。これは、`backup_compression_location` オプションを使用して実現できます。

---

**重要:** `backup_compression_location` オプションは、PostgreSQL 15 以降で実行されている場合にのみ使用できます。

---

`backup_compression_location = server|client`

`backup_compression_location = server` を使用すると、圧縮作業を PostgreSQL サーバーに移動するというコストで、バックアップに必要なネットワーク帯域幅が削減されます。

`backup_compression_location` が `server` に設定されている場合、追加オプション `backup_compression_format` を `plain` に設定して、ディスクに書き込む前に `pg_basebackup` にデータを解凍させることができます。

### 21.1.11 圧縮形式

`backup_compression_format = plain|tar`

`backup_compression_format` が設定されていない場合、または値が `tar` である場合、バックアップは圧縮された tar ボールとしてディスクに書き込まれます。`plain` および `tar` 形式の説明は、`[pg_basebackup documentation][pg_basebackup-documentation]` にあります。

---

**重要:** Barman は外部ツールを使用して圧縮バックアップを管理します。`backup_compression` および ``backup_compression_format`` によっては Postgres サーバーおよび Barman サーバーに 1 つ以上のツールをインストールする必要がある場合があります。次の表は、構成に応じて選択するのに役立ちます。

---

| backup_compression | backup_compress | Postgres server  | Barman server    |
|--------------------|-----------------|------------------|------------------|
| gzip               | plain           | <b>tar</b>       | None             |
| gzip               | tar             | <b>tar</b>       | <b>tar</b>       |
| lz4                | plain           | <b>tar, lz4</b>  | None             |
| lz4                | tar             | <b>tar, lz4</b>  | <b>tar, lz4</b>  |
| zstd               | plain           | <b>tar, zstd</b> | None             |
| zstd               | tar             | <b>tar, zstd</b> | <b>tar, zstd</b> |
| none               | tar             | <b>tar</b>       | <b>tar</b>       |

### 21.1.12 同時バックアップ

通常、バックアップ操作中に、Barman は `_concurrentbackup_` に PostgreSQL ネイティブファンクション `pg_start_backup` および `pg_stop_backup` を使用します。[`^ABOUT_CONCURRENT_BACKUP`] これは、PostgreSQL 9.6 以降のバックアップを取る推奨方法ですが、PostgreSQL ではファンクションの名前が `pg_backup_start` および `pg_backup_stop` に変更されていることに注意してください (15 ベータ)。

[`^ABOUT_CONCURRENT_BACKUP`] コンカレントバックアップは、\_ストリーミングレプリケーション\_ プロトコル\_たとえば、`pg_basebackup` のようなツールを使用するテクノロジーです。

コンカレントバックアップでは、推奨されるバックアップアプローチであるだけでなく、Barman を使用した次のアーキテクチャシナリオも可能になります。rsync を使用した **スタンバイサーバーからのバックアップ**。

デフォルトでは、`backup_options` は `concurrent_backup` に設定されます。バージョン 15 より前の PostgreSQL サーバーで排他的バックアップが必要な場合、ユーザーは `backup_options` を `exclusive_backup` に設定する必要があります。

`backup_options` が `concurrent_backup` に設定されている場合、Barman はサーバーの\_コンカレントバックアップモード\_をアクティブにし、次の 2 つの単純なルールに従います。

- `ssh_command` は、宛先の Postgres サーバーを指している必要があります
- `conninfo` は、宛先の Postgres データベース上のデータベースを指している必要があります。

---

**重要:** コンカレントバックアップの場合、現在 Barman は、フルバックアップの終了 WAL ファイルが実際に出荷されたかどうかを判断できません。PostgreSQL 自分自身が WAL ファイルが正しくアーカイブされていることを確認する排他的バックアップとは反対です。その WAL ファイルが Barman によって受信およびアーカイブされるまで、フルバックアップは一貫しているとは見なされないことに注意してください。Barman 2.5 では、`WAITING_FOR_WALS` と呼ばれる新しい状態を導入しました。これは、`check-backup` コマンド `cron` コマンドによって実行される通常のメンテナンスジョブの一部によって管理されます。Barman 2.10 から、`barman backup` コマンドで `--wait` オプションを使用できます。

---

### 21.1.13 スタンバイの同時バックアップ

スタンバイをバックアップする場合、次の [configuration options][config-options] はスタンバイサーバーを指す必要があります。

- `conninfo`
- `streaming_conninfo backup_method = postgres` または `streaming_archiver = on` を使用する  
場合
- `ssh_command backup_method = rsync` を使用する場合

次の構成オプションは、プライマリサーバーを指す必要があります。

- `primary_conninfo`

Barman は `primary_conninfo` を使用してプライマリで新しい WAL に切り替えるため、WAL スイッチが自然に発生するのを待たずにスタンバイに対する同時バックアップを完了できます。

---

**注釈:** プライマリへの書き込みトラフィックがほとんどまたは全くないときにスタンバイをバックアップする場合、`primary_conninfo` を設定することが特に重要です。

---

Barman 3.8.0 現在、`primary_conninfo` が設定されている場合、サーバーに `primary_checkpoint_timeout` オプションを追加できます。これは、新しい WAL ファイルが生成されるのを強制的に実行する前に Barman が待機する最大時間秒です。プライマリのチェックポイント。 `primary_checkpoint_timeout` オプションは、プライマリサーバーで設定された `archive_timeout` オプションの値より大きい秒数に設定する必要があります。

`primary_conninfo` が設定されていない場合、バックアップは引き続き実行されますが、プライマリの現在の WAL セグメントがバックアップに必要な最新の WAL よりも新しくなるまで、停止バックアップステージで待機します。

Barman では、現在 WAL ファイルとバックアップデータが同じ PostgreSQL サーバーから取得することを必要としています。スタンバイがプライマリに昇格する場合、バックアップと WAL は引き続き有効ですが、バックアップの取得と WAL の受信に新しいスタンバイを使用するように Barman 構成を更新することができます。

WAL は、WAL ストリーミングまたは WAL アーカイブを使用してスタンバイから取得できます。WAL ストリーミングを使用するには、[WAL ストリーミング](#) セクションの指示に従ってください。

スタンバイから WAL アーカイブを使用するには、WAL archiving via `archive_command` セクションの指示に従って、さらに、スタンバイサーバーの PostgreSQL 構成で `archive_mode = always` を設定します。

---

**注釈:** PostgreSQL 10 以前では、Barman は、スタンバイで同時に有効になっている WAL ストリーミングと WAL アーカイブを処理できません。したがって、WAL ストリーミングを使用する場合は WAL アーカイブを無効にする必要があります。その逆も可能です。これは、PostgreSQL 10 以前のバージョンで生成された WAL が論理的に等価でもバイナリレベルで異なる可能性があり、Barman が 2 つの WAL が同一であることの検出に失敗するためです。

---

### 21.1.14 即時チェックポイント

バックアップを開始する前に、Barman はチェックポイントを要求します。これにより、追加のワークロードが生成されます。通常、そのチェックポイントは、PostgreSQL サーバーのワークロード制御の設定に従って調整されます。これは、バックアップが遅延する可能性があることを意味します。

このデフォルトの動作は、`immediate_checkpoint` 構成グローバル/サーバーオプションデフォルトで `false` に設定を介して変更できます。

`immediate_checkpoint` が `true` に設定されている場合、PostgreSQL はワークロードを制限しようとせず、チェックポイントは最大速度で発生し、できるだけ早くバックアップを開始します。

次の 2 つのオプションのいずれかを指定して `barman backup` を発行することにより、いつでも構成オプションの動作をオーバーライドできます。

- `--immediate-checkpoint`、即時チェックポイントを強制します。
- `--no-immediate-checkpoint`、チェックポイントが発生するのを強制的に待機します。

### 21.1.15 ローカルバックアップ

**警告:** Barman と PostgreSQL は同じサーバー上にあり、同じ単一障害点の一部であるため、実稼働環境での使用はお勧めできません。一部の EnterpriseDB 顧客は、特定の状況で、最も重要なことに、会社が提供する 24 時間 365 日の運用サービスの下で使用するために、Barman にローカルバックアップのサポートを追加することを要求しました。現在、この機能を使用するには、ソースからインストールするか、権限、ロギング、cron 構成の点で `postgres` ユーザーの環境をカスタマイズする必要があります。

特別な状況では、Barman は PostgreSQL インスタンスが存在するのと同じサーバーにインストールでき、バックアップデータは PGDATA および該当する場合はテーブルスペースとは別のボリュームに保存されます。通常、これらのボリュームは、NFS のようなファイルシステムを備えたネットワークストレージアプリケーションにあります。

このアーキテクチャは EnterpriseDB によって承認されていません。RPO と RTO の点でより良い結果を伴う、PostgreSQL のビジネス継続性エクスペリエンスを強化するために、EnterpriseDB は引き続き、レプリケーションと監視を目的とした監視ウィットネスサーバのように機能できる Barman のリモートインストールを使用したシェアードナッシングアーキテクチャをお勧めします。

ローカルバックアップの唯一の要件は、Barman が PostgreSQL サーバーと同じユーザーで実行されることです。通常は `postgres` です。Community パッケージはデフォルトで Barman を `barman` ユーザーの下にインストールすることを考えると、このユースケースでは以下を含む手動インストール手順が必要です。

- cron 構成
- logrotate を含むログ構成

Barman で特定のサーバーのローカルバックアップを使用するには、`backup_method` から `local-rsync` に設定する必要があります。この機能は本質的に同等の `rsync` と同じであり、代わりに SSH に依存し、リ

モートで動作します。local-rsync では、ファイルシステムのコピーはローカルで `rsync` コマンドを発行して実行されますこのため、Barman は PostgreSQL と同じユーザーで実行する必要があります。

local-pg13 という名前のサーバーのローカルバックアップの構成の抜粋は次のとおりです。

```
[local-pg13]
description = "Local PostgreSQL 13"
backup_method = local-rsync
...
```

## 21.2 アーカイブ機能

### 21.2.1 WAL 圧縮

構成ファイルで `compression` オプションが設定されている場合、`barman cron` コマンドは WAL ファイルを圧縮します。このオプションでは、5 つの値を使用できます。

- `bzip2` Bzip2 圧縮用 `bzip2` ユーティリティが必要
- `gzip` Gzip 圧縮の場合 `gzip` ユーティリティが必要
- `pybzip2` Bzip2 圧縮用 Python の内部圧縮モジュールを使用
- `pygzip` Gzip 圧縮の場合 Python の内部圧縮モジュールを使用します
- `pigz` Pigz 圧縮用 `pigz` ユーティリティが必要
- `custom` カスタム圧縮用。次のオプションも設定する必要があります。
  - `custom_compression_filter` 圧縮フィルタ
  - `custom_decompression_filter` 解凍フィルタ
  - `custom_compression_magic` カスタム圧縮 wal ファイルを識別する 16 進文字列

注: `pybzip2` および `pygzip` を除くすべてのメソッドでは、`barman archive-wal` が新しいプロセスをフォークする必要があります。

### 21.2.2 同期 WAL ストリーミング

Barman は、同期スタンバイサーバーのようにトランザクション WAL ファイルを収集することにより、目標復旧ポイントをゼロにすることもできます。

このようなシナリオを構成するには、Barman サーバーは *PostgreSQL ストリーミング接続* を介して WAL をアーカイブするように構成し、`receive-wal` プロセスは PostgreSQL サーバーの同期スタンバイとして存在する必要があります。

まず、`show-servers` コマンドを使用して Barman `receive-wal` プロセスのアプリケーション名を取得する必要があります。

```
barman@backup$ barman show-servers pg|grep streaming_archiver_name
streaming_archiver_name: barman_receive_wal
```

次に、アプリケーション名を同期スタンバイとして `postgresql.conf` ファイルに追加する必要があります。

```
synchronous_standby_names = barman_receive_wal
```

**重要:** これは単なる構成例であり、Barman が同期スタンバイノードになる資格があることを示しています。Barman のみを使用することをお勧めするわけではありません。このトピックの詳細については、PostgreSQL ドキュメントから `["Synchronous Replication"]` を読むことができます。

構成をリロードするには、PostgreSQL サーバーを再起動する必要があります。

サーバーが正しく構成されている場合、`replication-status` コマンドは、同期ストリーミングクライアントとして `receive_wal` プロセスを表示します。

```
[root@backup ~]# barman replication-status pg
Status of streaming clients for server pg:
Current xlog location on master: 0/90000098
Number of streaming clients: 1

1. #1 Sync WAL streamer
   Application name: barman_receive_wal
   Sync stage      : 3/3 Remote write
   Communication   : TCP/IP
   IP Address      : 139.59.135.32 / Port: 58262 / Host: -
   User name       : streaming_barman
   Current state   : streaming (sync)
   Replication slot: barman
   WAL sender PID  : 2501
   Started at      : 2016-09-16 10:33:01.725883+00:00
   Sent location   : 0/90000098 (diff: 0 B)
   Write location  : 0/90000098 (diff: 0 B)
   Flush location  : 0/90000098 (diff: 0 B)
```

## 21.3 カタログ管理機能

### 21.3.1 最小限の冗長性の安全性

`minimum_redundancy` と呼ばれるグローバル/サーバーごとの構成オプションを使用して、PostgreSQL サーバーの定期バックアップの最小数を定義できます。デフォルトでは 0 に設定されています。

この値を 0 より大きい数値に設定することにより、Barman は常にサーバーカタログに少なくともその数のバックアップがあることを保証します。

これにより、誤った `barman delete` 操作から保護されます。

---

**重要:** 保持ポリシー設定が最小冗長要件と衝突しないことを確認します。Barman のログで、このトピックに関するメッセージがないかを定期的に確認します。

---

## 21.3.2 リテンションポリシー

Barman は、バックアップの **リテンションポリシー** をサポートしています。

バックアップ保持ポリシーは、バックアップおよび関連アーカイブログ書き込み先行ログセグメントを、リカバリ手順のために保持する必要がある期間を決定するユーザー定義のポリシーです。

ユーザーの要求に基づいて、Barman は、現在の保持ポリシーを満たすために必要な定期バックアップと、これらのバックアップの完全なリカバリに必要なアーカイブ WAL ファイルを保持します。

Barman ユーザーは、**バックアップ冗長性** 定期バックアップの数またはリカバリウィンドウ長さの観点からリテンションポリシーを定義できます。

冗長性に基づいたリテンションポリシー

冗長性ベースのリテンションポリシーでは、ユーザーは保持する定期バックアップの数を決定します。冗長ベースのリテンションポリシーは、リカバリウィンドウを使用するリテンションポリシーと対比されます。

リカバリウィンドウに基づいたリテンションポリシー

リカバリウィンドウは、Barman バックアップ保持ポリシーの 1 つであり、DBA が期間を指定し、Barman がポイントインタイムリカバリーに必要なバックアップおよび/またはアーカイブされた WAL ファイルのリカバリウィンドウ中の任意の時点での保持を保証します。間隔は常に現在の時刻で終了し、ユーザーが指定した日数だけ過去に戻ります。たとえば、リテンションポリシーが 7 日間のリカバリウィンドウに設定されており、現在時刻が金曜日の午前 9 時 30 分である場合、Barman は 1 日の午前 9 時 30 分にポイントインタイムリカバリーを可能にするために必要なバックアップを保持します。前の金曜日。

---

**重要:** ブロックレベルのインクリメンタルバックアップは、保持ポリシー処理中に考慮されません。これは、この種のインクリメンタルバックアップが回復可能であるためには、チェーンを生成するフルバックアップまでのすべての親バックアップに依存するためです。バックアップチェーンの一貫性を維持するために、保持ポリシーを適用するときにフルバックアップのみが考慮されます。> 仕組み> リテンションポリシーが適用されると、Barman はブロックレベルのインクリメンタルバックアップを無視し、フルバックアップのステータスにのみ焦点を当てます。> フルバックアップが `KEEP:FULL`、`KEEP:STANDALONE`、または `VALID` としてマークされている場合、ステータス `VALID` はすべての依存ブロックレベルのインクリメンタルバックアップにマークされます。フルバックアップが `OBSOLETE` としてマークされている場合、それに依存するすべてのブロックレベルのインクリメンタルバックアップも `OBSOLETE` としてマークされ、削除されます。

---

### 21.3.3 スコープ

リテンションポリシーは以下に対して定義できます。

- PostgreSQL の定期的なベースバックアップ `retention_policy` 構成オプションを介して
- アーカイブログ、ポイントインタイムリカバリー用 `wal_retention_policy` 構成オプションを介して

---

**重要:** 時間ディメンションでは、アーカイブログは定期バックアップの時間ウィンドウに含まれる必要があります。

---

ここには、完全または部分的なポイントインタイムリカバリーという 2 つの一般的なユースケースがあります。

完全なポイント インタイム リカバリーシナリオ

ベースバックアップとアーカイブログは同じ保持ポリシーを共有するため、最初に使用可能なバックアップから任意の時点で復旧できます。

部分的なポイント インタイム リカバリーシナリオ

ベースバックアップの保持ポリシーはアーカイブログの保持ポリシーよりも広く、たとえば、ユーザーは過去 6 か月の完全な週次バックアップを保持できますが、過去 4 週間のアーカイブログ最後のものから開始する任意の時点でのリカバリを許可 4 つの定期的な毎週のバックアップ)。

---

**重要:** 現在、Barman は、`wal_retention_policy` オプションを `main` にコンストレインすることにより、\*\*完全なポイントインタイムリカバリー\*\*シナリオのみを実装しています。

---

### 21.3.4 仕組み

Barman のリテンションポリシーは次のとおりです。

- 自動 `barman cron` によって適用
- マニュアル Barman は、古いバックアップを単に報告し、削除できます。

---

**重要:** 現在、Barman は手動適用を実装していません。この機能は将来のバージョンで利用できるようになる予定です。

---

### 21.3.5 構成と構文

リテンションポリシーは、次の構成オプションを介して定義できます。

- `retention_policy` ベースバックアップ保持用
- `wal_retention_policy` アーカイブログ保持用
- `retention_policy_mode auto` にのみ設定できますリテンションポリシーは `barman cron` コマンドによって自動的に適用されます。

これらの構成オプションは、グローバルレベルとサーバーレベルの両方で定義できるため、ユーザーはマルチサーバー環境で最大限の柔軟性を実現します。

#### `retention_policy` の構文

`retention_policy` を介したベースバックアップリテンションポリシーの一般的な構文は次のとおりです。

```
retention_policy = {REDUNDANCY value | RECOVERY WINDOW OF value {DAYS | WEEKS |  
↪MONTHS}}
```

そこで

- 構文は大文字と小文字を区別します
- `value` は整数であり、`> 0` です
- 冗長リテンションポリシー の場合
  - `value` はサーバーの最小冗長レベル以上である必要がありますその値が割り当てられていない場合、警告が生成されます
  - 最初の有効なバックアップは値番目のバックアップです逆順の時系列で
- リカバリウィンドウポリシー
  - の場合、復旧可能ポイントは次のとおりです。現在時間 - window
  - 最初の有効なバックアップは、復旧可能ポイントの前の最初に使用可能なバックアップです。逆順の時系列の値はサーバーの最小冗長レベル以上である必要がありますその値に割り当てられず、警告が生成された場合

デフォルトでは、`retention_policy` は空です保持期間は適用されません。

## wal\_retention\_policy の構文

現在、wal\_retention\_policy の唯一の許可値は特別な値 `main` であり、アーカイブログの保持ポリシーをベースバックアップの保持ポリシーにマップします。

## 21.4 フックスクリプト

Barman を使用すると、データベース管理者は次の 2 つのイベントでフックスクリプトを実行できます。

- バックアップの前後
- バックアップの削除の前後
- WAL ファイルがアーカイブされる前と後
- WAL ファイルが削除される前と後

Barman が管理できるフックスクリプトには 2 種類があります。

- 標準フックスクリプト
- リトライフックスクリプト

これら 2 種類のフックスクリプトの唯一の違いは、Barman はリターンコードをチェックせずに標準フックスクリプトを 1 回だけ実行するのに対し、再試行フックスクリプトは、リターンコードによっては複数回実行される場合があることです。

具体的には、再試行フックスクリプトを実行するときに、Barman はリターンコードをチェックし、スクリプトが `SUCCESS` 標準のリターンコード `0`、または `ABORT_CONTINUE` リターンコード `62`、または `ABORT_STOP` リターンコード `63` のいずれかを返すまで、無期限に再試行します。Barman は、他の戻りコードを再試行の一時的な失敗として扱います。ユーザーにはより多くの権限が与えられます。フックスクリプトは、障害が一時的なものであるかどうかを指定することにより、ワークフローを制御できます。また、'pre' フックスクリプトの場合、`ABORT_STOP` を返すことにより、ユーザーは Barman に失敗してメインオペレーションを中断するように要求できます。

フックスクリプトは次の順序で実行されます。

1. 標準の 'pre' フックスクリプト存在する場合
2. retry 'pre' フックスクリプト存在する場合
3. 実際のイベントつまりバックアップオペレーションまたは WAL アーカイブ、再試行 'pre' フックスクリプトが `ABORT_STOP` で中止されなかった場合
4. 再試行 'post' フックスクリプト存在する場合
5. 標準の 'post' フックスクリプト存在する場合

フックスクリプトによって生成された出力は、Barman のログファイルに書き込まれます。

---

**注釈:** 現在、`ABORT_STOP` は、再試行'post' フックスクリプトによって無視されます。これらの場合、追加の警告をログに記録することとは別に、`ABORT_STOP` は `'ABORT_CONTINUE'` のように動作します。

---

### 21.4.1 バックアップスクリプト

これらのスクリプトは、次のグローバル構成オプションサーバーごとにオーバーライドできます。

- `pre_backup_script_hook script_` はベースバックアップの `_before_` 実行されました。1 回のみ、終了コードはチェックなし
- `pre_backup_retry_script_retry` フックスクリプト\_ベースバックアップの `_before_` 実行され、成功または中止されるまで繰り返し
- `post_backup_retry_script_retry` フックスクリプト\_ベースバックアップの `_after_` 実行され、成功または中止されるまで繰り返し
- `post_backup_script_hook script_` ベースバックアップの後に実行されました 1 回のみ、終了コードはチェックなし

スクリプト定義はシェルに渡され、任意の終了コードを返すことができます。`_retry_` スクリプトの場合にのみ、Barman はリターンコードをチェックします [フックスクリプト](#) を参照してください。

シェル環境には次の変数が含まれます。

- `BARMAN_BACKUP_DIR` バックアップ先ディレクトリ
- `BARMAN_BACKUP_ID` バックアップの ID
- `BARMAN_CONFIGURATION` Barman が使用する構成ファイル
- `BARMAN_ERROR` エラーメッセージある場合 `post` フェーズのみ
- `BARMAN_PHASE` スクリプトのフェーズ、`pre` または `post` のいずれか
- `BARMAN_PREVIOUS_ID` 以前のバックアップの ID 存在する場合
- `BARMAN_RETRY` 再試行スクリプトの場合は 1、それ以外の場合は 0
- `BARMAN_SERVER` サーバーの名前
- `BARMAN_STATUS` バックアップのステータス
- `BARMAN_VERSION` Barman のバージョン

### 21.4.2 バックアップ削除スクリプト

バージョン 2.4 では、バックアップ前および後の削除スクリプトを導入しました。

以前のスクリプトと同様に、バックアップ削除スクリプトはグローバル構成オプション内で構成でき、サーバーごとにそれらをオーバーライドすることが可能です。

- `pre_delete_script_hook script_` は、バックアップの削除の `_before_` に起動され、1 回のみ、終了コードはチェックされません
- `pre_delete_retry_script_retry` フックスクリプト\_バックアップの削除の `_before_` 実行され、成功または中止されるまで繰り返し

- `post_delete_retry_script_retry` フックスクリプト\_バックアップの削除の後\_実行され、成功または中止されるまで繰り返し
- `post_delete_script_hook script_`は、バックアップの削除後に 1 回だけ、終了コードはチェックせずに起動されました

スクリプトはシェルを介して実行され、任意の終了コードを結果ことができます。\_retry\_スクリプトの場合にのみ、Barman はリターンコードをチェックします上部のセクションを参照してください。

削除スクリプトは、バックアップスクリプトと同じ環境変数に加えて、以下を使用します。

- `BARMAN_NEXT_ID` 次のバックアップの ID 存在する場合

### 21.4.3 WAL アーカイブスクリプト

バックアップスクリプトと同様に、アーカイブスクリプトはグローバル構成オプションサーバーごとにオーバーライドできます。

- `pre_archive_script` WAL ファイルがメンテナンス通常 `barman cron` によってアーカイブされる前に `_hook script_` 実行され、1 回のみ、終了コードはチェックされません
- `pre_archive_retry_script_retry` フックスクリプト\_は、WAL ファイルがメンテナンス通常 `barman cron` によってアーカイブされる `_before_` 実行され、成功または中止されるまで繰り返します
- `post_archive_retry_script` WAL ファイルがメンテナンスによってアーカイブされた後、\_retry フックスクリプト\_実行され、成功または中止されるまで繰り返し
- `post_archive_script` WAL ファイルがメンテナンスによってアーカイブされた後、\_hook script\_ 実行されました。1 回のみ、終了コードはチェックされません

スクリプトはシェルを介して実行され、任意の終了コードを結果ことができます。\_retry\_スクリプトの場合にのみ、Barman はリターンコードをチェックします上部のセクションを参照してください。

アーカイブスクリプトは、バックアップスクリプトといくつかの環境変数を共有します。

- `BARMAN_CONFIGURATION` Barman が使用する構成ファイル
- `BARMAN_ERROR` エラーメッセージある場合 `post` フェーズのみ
- `BARMAN_PHASE` スクリプトのフェーズ、`pre` または `post` のいずれか
- `BARMAN_SERVER` サーバーの名前

次の変数はアーカイブスクリプトに固有です。

- `BARMAN_SEGMENT` WAL ファイルの名前
- `BARMAN_FILE` WAL ファイルのフルパス
- `BARMAN_SIZE` WAL ファイルのサイズ
- `BARMAN_TIMESTAMP` WAL ファイルのタイムスタンプ
- `BARMAN_COMPRESSION` WAL ファイルに使用される圧縮の種類

## 21.4.4 WAL 削除スクリプト

バージョン **2.4** では、WAL 削除前後のスクリプトを導入しています。

他のフックスクリプトと同様に、wal delete スクリプトはグローバル構成オプションで構成でき、サーバーごとにそれらをオーバーライドすることが可能です。

- `pre_wal_delete_script` WAL ファイルの削除の `_before_` に `_hook script_` 実行されました
- `pre_wal_delete_retry_script` `_retry` フックスクリプトは、WAL ファイルの削除の `_before_` 実行され、成功または中止されるまで繰り返します
- `post_wal_delete_retry_script` `_retry` フックスクリプト WAL ファイルの削除の後 `_` 実行され、成功または中止されるまで繰り返し
- `post_wal_delete_script` WAL ファイルの削除の後に `_` フックスクリプト `_` 実行されました

スクリプトはシェルを介して実行され、任意の終了コードを結果ことができます。`_retry_` スクリプトの場合にのみ、Barman はリターンコードをチェックします上部のセクションを参照してください。

WAL 削除スクリプトは、WAL アーカイブスクリプトと同じ環境変数を使用します。

## 21.4.5 リカバリスクリプト

バージョン **2.4** では、リカバリー前後のスクリプトを導入しています。

以前のスクリプトと同様に、リカバリスクリプトはグローバル構成オプション内で構成でき、サーバーごとにオーバーライドすることが可能です。

- `pre_recovery_script` `_hook script_` は、バックアップのリカバリーの `_before_` に起動された、1 回のみ、終了コードはチェックされません
- `pre_recovery_retry_script` `_retry` フックスクリプト バックアップのリカバリーの `_before_` 実行、成功または中止するまで繰り返します
- `post_recovery_retry_script` `_retry` フックスクリプト バックアップのリカバリーの後に実行され、成功または中止されるまで繰り返します
- `post_recovery_script` `_hook script_` は、バックアップのリカバリーの後に起動され、1 回のみ、終了コードはチェックされません

スクリプトはシェルを介して実行され、任意の終了コードを結果ことができます。`_retry_` スクリプトの場合にのみ、Barman はリターンコードをチェックします上部のセクションを参照してください。

リカバリスクリプトは、バックアップスクリプトと同じ環境変数に加えて、以下を使用します。

- `BARMAN_DESTINATION_DIRECTORY` 新しいインスタンスがリカバリされるディレクトリ
- `BARMAN_TABLESPACES` テーブルスペース再配置マップ JSON 存在する場合
- `BARMAN_REMOTE_COMMAND` リカバリーで使用される secure Shell コマンド存在する場合
- `BARMAN_RECOVER_OPTIONS` リカバリー追加オプション JSON 存在する場合

## 21.5 カスタマイズ

### 21.5.1 ロックファイルディレクトリ

Barman では、`barman_lock_directory` グローバルオプションを介してロックファイルのディレクトリを指定できます。

ロックファイルは、グローバルおよびサーバーレベルで同時作業を調整するために使用されますたとえば、`cron` 操作、バックアップ操作、WAL アーカイブへのアクセスなど。

デフォルトでは、下位互換性の理由から、`barman_lock_directory` は `barman_home` に設定されます。

---

**Tip:** ユーザーは、実行時変数データ専用のディレクトリ、たとえば `/var/run/barman` などの揮発性パーティションのディレクトリを使用することをお勧めします。

---

### 21.5.2 バイナリパス

バージョン 1.6.0 の時点で、Barman では、`global/server` オプション `path_prefix` を使用して、Barman が実行可能ファイルを検索する 1 つ以上のディレクトリを指定できます。

`path_prefix` が提供される場合、コロンで区切られた 1 つ以上のディレクトリのリストが含まれている必要があります。Barman は、最初にこれらのディレクトリ内を検索し、次に `PATH` 環境変数で指定されたディレクトリを検索します。

デフォルトでは、`path_prefix` オプションは空です。

## 21.6 クラスタ管理システムとの統合

Barman は、スタンバイサーバーストリーミングレプリケーションまたは従来のファイルベースのログ配布を使用および `[repmgr][repmgr]` のような高可用性ツールと統合するように設計されています。

アーキテクチャの観点から、WAL ファイルを Barman サーバーに直接アーカイブするように PostgreSQL を構成する必要があります。Barman は、`get-wal` フレームワークのおかげで、WAL ハブとして使用することもできます。このために、すべてのスタンバイサーバーで `barman-cli` パッケージの一部である `barman-wal-restore` スクリプトを使用できます。

`replication-status` コマンドを使用すると、管理対象サーバー、特にホットスタンバイサーバーと WAL ストリーマーに接続されているストリーミングクライアントに関する情報を取得できます。

## 21.6.1 構成モデル

構成モデルは、構成オプションのオーバーライドのセットを定義します。これらのオーバーライドは、構成モデルと同じクラスターの一部である Barman サーバーに適用できます。これらはクラスター環境を処理する場合に役立つため、たとえば、フェールオーバーイベントに応答して Barman サーバーの構成を変更できます。

例として、次のノードを含む PostgreSQL クラスターがあるとします。

- pg-node-1 プライマリ
- pg-node-2 スタンバイ
- pg-node-3 スタンバイ

プライマリノードからバックアップしており、次のオプションを含む構成があるとします。

```
[my-barman-server]
cluster = my-cluster
conninfo = host=pg-node-1 user=barman database=postgres
streaming_conninfo = host=pg-node-1 user=streaming_barman
; other options...
```

たとえば、次のようなそのクラスターの構成モデルを作成できます。

```
[my-barman-server:backup-from-pg-node-2]
cluster = my-cluster
model = true
conninfo = host=pg-node-2 user=barman database=postgres
streaming_conninfo = host=pg-node-2 user=streaming_barman
```

これは、次のコマンドで pg-node-1 から pg-node-2 へのフェイルオーバーに適用できるため、新しいプライマリノードからバックアップを開始します。

```
barman config-switch my-barman-server my-barman-server:backup-from-pg-node-2
```

これにより、クラスター構成オプションが選択したモデルで定義された値でオーバーライドされます。

**注** モデルからすべてのオプションを設定できるわけではありません。[section 5 of the ‘man’ page][man5] を参照して、モデルに適用されるスコープの設定を確認してください。

**注意** pg-backup-api をチェックすることに興味があるかもしれません。これは、REST API をスタートし、barman config-switch を含む barman コマンドを実行するためのリモート要求をリッスンすることができます。

## 21.7 パラレルジョブ

デフォルトでは、Barman はバックアップとリカバリ操作の両方でファイルコピーに 1 つのワーカーのみを使用します。バージョン 2.2 から、ファイルコピーを実行するワーカーの数をカスタマイズできるようになりました。この場合、コピーされるファイルはすべてのパラレルワーカーに等しく分散されます。

グローバルスコープとサーバースコープで構成し、対応する構成ファイルにこれらを追加できます。

```
parallel_jobs = n
```

`n` は、ファイルコピー操作で使用されるパラレルワーカーの目的の数です。デフォルト値は 1 です。

いずれの場合も、ユーザーは実行時に `backup` または `recover` コマンドを実行するときにこの値をオーバーライドできます。たとえば、次のように 4 つのパラレルワーカーを使用できます。

```
barman backup --jobs 4 server1
```

または、次のとおりです。

```
barman backup --j 4 server1
```

このパラレルジョブ機能は、`rsync` /SSH を介して構成されたサーバーでのみ利用可能であることに注意してください。ストリーミングプロトコルを介して構成されたサーバーの場合、Barman は現在 1 つのワーカーのみに制限されている `pg_basebackup` に依存します。

### 21.7.1 パラレルジョブと `sshd MaxStartups`

Barman は、SSH サーバーで許可される非認証の同時接続の最大数を超えないように、並列 `Rsync` ジョブが開始されるレートを制限します。この最大値は、`sshd` パラメーター `MaxStartups` で定義されます。`MaxStartups` を超える接続が作成されたが、まだ認証されていない場合、SSH サーバーは一部またはすべての接続をドロップする場合があります、その結果、バックアップまたはリカバリーが失敗する場合があります。

ほとんどのプラットフォームでの `sshd MaxStartups` のデフォルト値は 10 です。したがって、Barman は 10 のバッチで並列ジョブを開始し、1 秒の期間内に複数のジョブのバッチを開始しません。これにより、1 秒あたり 10 ジョブの有効なレート制限が得られます。

この制限は、次の 2 つの構成オプションを使用して変更できます。

- `parallel_jobs_start_batch_size` 単一のバッチで開始する並列ジョブの最大数。
- `parallel_jobs_start_batch_period` ジョブの単一バッチが開始される秒単位の時間。

たとえば、2 秒の期間内に 5 つ以下の新しい `Rsync` ジョブが作成されないようにするには、次のとおりです。

```
parallel_jobs_start_batch_size = 5
parallel_jobs_start_batch_period = 2
```

`barman backup` および `barman recover` コマンドの両方で次の引数を使用して、構成オプションをオーバーライドできます。

- `--jobs-start-batch-size`
- `--jobs-start-batch-period`

## 21.8 地理的な冗長性

Barman を使用して **カスケードバックアップアーキテクチャ** をセットアップすることが可能です。バックアップサーバーのソースは、PostgreSQL サーバーではなく Barman インストールです。

この機能を使用すると、ユーザーは PostgreSQL バックアップの `_地理的に分散_コピー` を透過的に保持できます。

Barman の用語では、PostgreSQL サーバーではなく Barman インストールに接続されているバックアップサーバーは **パッシブノード** と定義されます。パッシブノードは `primary_ssh_command` オプションを介して構成され、グローバルプライマリ Barman インストールのフルレプリカの場合とサーバーレベル `_direct_` および `_passive_` サーバーの両方を持つ混合シナリオの場合の両方で使用できます。

### 21.8.1 情報の同期

`barman sync-info` コマンドは、Barman サーバーの現在のステータスに関する情報を収集するために使用されます。同期に役立ちます。使用可能な構文は次のとおりです。

```
barman sync-info [--primary] <server_name> [<last_wal> [<last_position>]]
```

コマンドは、以下を含む JSON オブジェクトを結果ます。

- そのサーバーのステータスが **DONE** であるすべてのバックアップを含むマップ
- アーカイブされたすべての WAL ファイルのリスト
- サーバーの構成
- 最後の読み取り位置 `_xlog データベースファイル_内`
- 最後に読み取られた WAL ファイルの名前

JSON 応答には、**master** ノードと **passive** ノード間の同期に必要なすべての情報が含まれています。

`--primary` が指定されている場合、コマンドはローカルにではなく、定義されたプライマリノードで実行されます。

## 21.8.2 構成

サーバーを **passive node** として構成するのは簡単な操作です。サーバー構成に次のオプションを追加するだけです。

```
primary_ssh_command = ssh barman@primary_barman
```

このオプションは、プライマリサーバーへの SSH 接続パラメーターを指定し、パッシブサーバーのバックアップデータのソースを指定します。

-c/--config オプションを使用して **barman** を呼び出しており、パッシブノードがプライマリノードで **barman** を呼び出すときに同じオプションを使用する場合、次のオプションを追加します。

```
forward_config_path = true
```

## 21.8.3 ノードの同期

ノードが **passive** としてマークされている場合、Barman によって特別な方法で扱われます。

- 標準のメンテナンス操作から除外されます
- **barman backup** を含む PostgreSQL への直接操作は禁止されています

パッシブサーバーとそのプライマリ間の同期は、以下を透過的に呼び出す **barman cron** によって自動的に管理されます。

1. **barman sync-info --primary**、同期情報を収集するため
2. **barman sync-backup**、プライマリノードで使用可能なすべてのバックアップのローカルコピーを作成するため
3. **barman sync-wals**、プライマリノードで使用可能なすべての WAL ファイルをローカルにコピーするため

## 21.8.4 手動同期

**barman cron** はパッシブ/プライマリノードの同期を自動的に管理しますが、次の方法でバックアップの同期を手動でトリガーすることが可能です。

```
barman sync-backup <server_name> <backup_id>
```

**sync-backup barman** を起動すると、**primary\_ssh\_**コマンドを使用してマスターサーバーに接続し、リモートマシンにバックアップが存在する場合、**rsync** を使用してすべてのファイルのコピーを開始します。バックアップごとに 1 つの同期プロセスのみが許可されます。

WAL ファイルは、次の方法で同期することもできます。

```
barman sync-wals <server_name>
```

## 21.9 クラウドスナップショットバックアップ

スナップショットバックアップは、クラウドストレージボリュームの 1 つ以上のスナップショットで構成されるバックアップです。

次のいずれかのコマンドを使用して、suitable PostgreSQL server のスナップショットバックアップを取得できます。

- Barman サーバーが WAL の保存とメタデータのバックアップに使用されている場合、`barman backup` と `required configuration operations for snapshots`。
- Barman サーバーがなく、代わりにクラウドオブジェクトストアが WAL とバックアップメタデータに使用されている場合に、必要な `command line arguments` を備えた `barman-cloud-backup`。

### 21.9.1 スナップショットバックアップの詳細

スナップショットバックアップを取得する高レベルのプロセスは次のとおりです。

1. Barman は一連のプリフライトチェックを実行して、スナップショットオプション、インスタンス、ディスクを検証します。
2. Barman は、[PostgreSQLbackup API][postgres-low-level-base-backup] を使用してバックアップを開始します。
3. クラウドプロバイダー API を使用して、指定された各ディスクのスナップショットをトリガーします。Barman は、次のディスクに移動する前に、アプリケーションの一貫性を保証するためにスナップショットが必要な状態に到達するまで待機します。
4. 各ディスクのデバイス名などの追加のプロバイダー固有データは、バックアップメタデータに保存されます。
5. 各ディスクのマウントポイントとマウントオプションは、バックアップメタデータに保存されます。
6. Barman は、PostgreSQL バックアップ API を使用してバックアップを停止します。

クラウドプロバイダーの API 呼び出しは、`backup` コマンドが実行されるノードで行われます。これは、Barman サーバー `barman backup` を使用する場合 または PostgreSQL サーバー `barman-cloud-backup` を使用する場合。

次のプリフライトチェックは、各バックアップの前に、およびスナップショットバックアップ用に構成されたサーバーに対して `barman check` が実行されたときに実行されます。

- `snapshot_instance` およびプロバイダ固有の引数で指定されたコンピューティングインスタンスが存在します。
- `snapshot_disks` で指定されたディスクが存在する。
  - `snapshot_disks` で指定されたディスクは `snapshot_instance` に接続されています。
  - `snapshot_disks` で指定されたディスクは `snapshot_instance` にマウントされます。

## 21.9.2 スナップショットバックアップからのリカバリー

Barman は現在、スナップショットバックアップからの完全に自動化されたリカバリーを実行しません。これは、スナップショットからの復旧には、新しいインフラストラクチャのプロビジョニングと管理が必要であり、Terraform などの専用のインフラストラクチャアズコードソリューションでより適切に処理されるためです。

ただし、`barman recover` コマンドは、スナップショットリカバリインスタンスの検証、安全でないオプションの PostgreSQL 構成のチェックなどのリカバリ後タスクを実行し、必要な PITR オプションを設定するために使用できます。また、`backup_label` ファイルを適切な場所にコピーしバックアップラベルがボリュームスナップショットに保存されないため、必要な WAL 全体をコピーします--`get-wal` 復旧オプションを使用しない場合、取得するように PostgreSQL `restore_command` を構成します WAL)。

`barman-cloud-backup` で作成されたバックアップを復元する場合は、`barman recover` の代わりにより制限的な `barman-cloud-restore for スナップショット` コマンドを使用する必要があります。

スナップショットバックアップからのリカバリーは、次の手順で構成されます。

1. バックアップ中に作成されたスナップショットごとに、新しいディスクをプロビジョニングします。
2. 手順 1 でプロビジョニングされた各ディスクがバックアップメタデータに従って接続およびマウントされるコンピューティングインスタンスをプロビジョニングします。
3. `barman recover` または `barman-cloud-restore for スナップショット` コマンドを使用して、リカバリーを検証およびファイナライズします。

ステップ 1 および 2 は、既存のインフラストラクチャアズコードシステムで最適に処理されますが、これらの手順を手動でまたはカスタムスクリプトを使用して実行することもできます。

これらの手順を実行する際には、次のリソースが役立つ場合があります。

- 例 [recovery script for GCP][snapshot-recovery-script]。
- 例 [runbook for Azure][snapshot-recovery-runbook-azure]。

上記のリソースは、バックアップ/リカバリー環境に関する仮定を立てているため、さらにカスタマイズせずに本番使用に適していると考えられるべきではありません。

リカバリインスタンスがプロビジョニングされ、バックアップスナップショットからクローンされたディスクが接続およびマウントされたら、次の追加引数を使用して `barman recover` を実行します。

- `--remote-ssh-command` リカバリインスタンスにログインするために必要な ssh コマンド。
- `--snapshot-recovery-instance` クラウドプロバイダーの必要なリカバリインスタンスの名前。
- スナップショットプロバイダーに固有の追加引数。

例

```
barman recover SERVER_NAME BACKUP_ID REMOTE_RECOVERY_DIRECTORY \
  --remote-ssh-command ssh USER@HOST \
  --snapshot-recovery-instance INSTANCE_NAME
```

Barman は、バックアップがスナップショットバックアップであることを自動的に検出し、接続されたディスクがそのバックアップのスナップショットからクローンされたことを確認します。Barman は、バック

アップラベルと WAL を適切な場所にコピーし、PostgreSQL 構成に必要な復旧オプションを設定することにより、PostgreSQL の復旧を準備します。

gcp プロバイダーでは、次の追加の `barman recover` 引数を使用できます。

- `--gcp-zone` リカバリインスタンスが存在するアベイラビリティゾーンの名前。提供されない場合、Barman はサーバー構成で設定された `gcp_zone` の値を使用します。

azure プロバイダーでは、次の追加の `barman recover` 引数を使用できます。

- `--azure-resource-group` リカバリインスタンスが属するリソースグループ。提供されない場合、Barman はサーバー構成で設定された `azure_resource_group` の値を使用します。

aws プロバイダーでは、次の追加の `barman recover` 引数を使用できます。

- `--aws-region` リカバリインスタンスが存在する AWS リージョン。提供されない場合、Barman はサーバー構成で設定された `aws_region` の値を使用します。

スナップショットバックアップを復旧する場合、次の `barman recover` 引数/config 変数は使用できないことに注意してください。

| Command argument                     | Config variable                    |
|--------------------------------------|------------------------------------|
| <code>--bwlmit</code>                | <code>bandwidth_limit</code>       |
| <code>--jobs</code>                  | <code>parallel_jobs</code>         |
| <code>--recovery-staging-path</code> | <code>recovery_staging_path</code> |
| <code>--tablespace</code>            | N/A                                |

### 21.9.3 スナップショットバックアップのバックアップメタデータ

リカバリディスクとインスタンスがインフラストラクチャコード、アドホック自動化、または手動でプロビジョニングされるかどうかにかかわらず、Barman を照会して、特定のバックアップに必要なスナップショットを見つける必要があります。これは、バックアップの各スナップショットの詳細を提供する `barman show-backup` を使用して実現できます。例

```
$ barman show-backup primary 20230123T131430
Backup 20230123T131430:
  Server Name      : primary
  System Id        : 7190784995399903779
  Status           : DONE
  PostgreSQL Version : 140006
  PGDATA directory : /opt/postgres/data

  Server information:
    Checksums       : on

  Snapshot information:
    provider        : gcp
```

(次のページに続く)

(前のページからの続き)

```

project                : project_id

device_name            : pgdata
snapshot_name          : barman-av-ubuntu20-primary-pgdata-20230123t131430
snapshot_project       : project_id
Mount point           : /opt/postgres
Mount options          : rw,noatime

device_name            : tbs1
snapshot_name          : barman-av-ubuntu20-primary-tbs1-20230123t131430
snapshot_project       : project_id
Mount point           : /opt/postgres/tablespaces/tbs1
Mount options          : rw,noatime

```

--format=json オプションは、外部ツールと統合する場合に使用できます。例

```

$ barman --format=json show-backup primary 20230123T131430
...
"snapshots_info": {
  "provider": "gcp",
  "provider_info": {
    "project": "project_id"
  },
  "snapshots": [
    {
      "mount": {
        "mount_options": "rw,noatime",
        "mount_point": "/opt/postgres"
      },
      "provider": {
        "device_name": "pgdata",
        "snapshot_name": "barman-av-ubuntu20-primary-pgdata-20230123t131430",
        "snapshot_project": "project_id"
      }
    },
    {
      "mount": {
        "mount_options": "rw,noatime",
        "mount_point": "/opt/postgres/tablespaces/tbs1"
      },
      "provider": {
        "device_name": "tbs1",
        "snapshot_name": "barman-av-ubuntu20-primary-tbs1-20230123t131430",
        "snapshot_project": "project_id",
      }
    }
  ]
}

```

(次のページに続く)

```
}  
]  
}  
...
```

`barman-cloud-backup` で取得されたバックアップの場合、同様の `[barman-cloud-backup-show][pgbarman-cloud-backup-show]` コマンドがあり、`barman-cloud-backup-list` とともに使用して、クラウドオブジェクトストアのバックアップメタデータを照会できます。

`snapshots_info/provider_info` および `snapshots_info/snapshots/*/provider` で使用可能なメタデータは、以降のセクションで説明するように、クラウドプロバイダーによって異なります。

### 21.9.4 GCP プロバイダー固有のメタデータ

`snapshots_info/provider_info` では、次のフィールドが使用できます。

- `project` バックアップとリカバリーに関係するリソースを所有するプロジェクトの GCP プロジェクト ID。

`snapshots_info/snapshots/*/provider` では、次のフィールドが使用できます。

- `device_name` スナップショットのソースディスクがバックアップ時にバックアップ VM に接続された短いデバイス名。
- `snapshot_name` スナップショットの名前。
- `snapshot_project` スナップショットを所有する GCP プロジェクト ID。

### 21.9.5 Azure プロバイダー固有のメタデータ

`snapshots_info/provider_info` では、次のフィールドが使用できます。

- `subscription_id` バックアップとリカバリーに関係するリソースを所有する Azure サブスクリプション ID。
- `resource_group` バックアップに関係するリソースが属する Azure リソースグループ。

`snapshots_info/snapshots/*/provider` では、次のフィールドが使用できます。

- `location` スナップショットが取得されたディスクの Azure の場所。
- `lun` バックアップ時にスナップショットが取得されたディスクを識別する LUN。
- `snapshot_name` スナップショットの名前。

### 21.9.6 AWS プロバイダー固有のメタデータ

`snapshots_info/provider_info` では、次のフィールドが使用できます。

- `account_id` バックアップの作成に使用されたリソースを所有する AWS アカウントの ID。
- `region` バックアップに関係するリソースが存在する AWS リージョン。

`snapshots_info/snapshots/*/provider` では、次のフィールドが使用できます。

- `device_name` バックアップ時にバックアップ VM でソースディスクがマッピングされたデバイス。
- `snapshot_id` AWS によって割り当てられたスナップショットの ID。
- `snapshot_name` スナップショットの名前。



## 第22章 Barman クライアントユーティリティ `barman-cli`

以前は別のオープンソースプロジェクトであった `barman-cli` は、バージョン 2.8 から Barman のコアにマージされ、RPM / Debian パッケージとして配布されています。 `barman-cli` には、PostgreSQL サーバーとともにインストールする推奨されるクライアントユーティリティのセットが含まれています。

- `barman-wal-archive` 「`barman-wal-archive` を介した WAL アーカイブ」セクションで説明しているように、`archive_command` として使用するアーカイブスクリプト。
- `barman-wal-restore` 上記の「`get-wal`」セクションで説明したように、スタンバイサーバーと復旧サーバーで `restore_command` 復旧オプションの一部として使用される WAL 復元スクリプト。

詳細については、特定のマニュアルページまたは `--help` オプションを参照してください。

### 22.1 インストール

Barman クライアントユーティリティは、通常、PostgreSQL がインストールされている場所にインストールされます。私たちのお勧めは、プライマリまたはスタンバイであるすべての PostgreSQL サーバーに `barman-cli` パッケージをインストールすることです。

リポジトリをインストールするには、メインの「インストール」セクションを参照してください。

`root` として RedHat / CentOS システムにパッケージをインストールするには、次のように入力します。

```
yum install barman-cli
```

Debian / Ubuntu では、`root` ユーザーとして次のように入力します。

```
apt-get install barman-cli
```



## 第23章 クラウド用の Barman クライアントユーティリティ `barman-cli-cloud`

Barman クライアントユーティリティは、オブジェクトストレージの統合をサポートし、WAL ファイルとバックアップをサポートされているクラウドプロバイダーに中継することにより、PostgreSQL データベースのディザスターリカバリー機能を強化するように拡張されました。

サポートされているクラウドプロバイダーは次のとおりです。

- AWS S3 または S3 互換オブジェクトストア
- Azure Blob Storage
- Google Cloud Storage Rest API

これらのユーティリティは `barman-cli-cloud` RPM/Debian パッケージで配布され、PostgreSQL サーバーと一緒にインストールできます。

- `barman-cloud-wal-archive archive_command` として使用して、Barman サーバーをバイパスして WAL ファイルをクラウドストレージに直接出荷するアーカイブスクリプト。または、WAL アーカイブ `pre_archive_retry_script` のフックスクリプトとして
- `barman-cloud-wal-restore restore_command` として使用するスクリプトで、クラウドストレージから WAL ファイルを取得し、Barman サーバーをバイパスし、PostgreSQL スタンバイに直接保存します。
- `barman-cloud-backup` PostgreSQL サーバーで直接ローカルバックアップをとり、Barman サーバーをバイパスして、サポートされているクラウドプロバイダーに発送するために使用されるバックアップスクリプト。または、バーマンのバックアップをクラウドにコピーするためのフックスクリプトとして `post_backup_retry_script`)
- `barman-cloud-backup-delete barman-cloud-backup` で取得した 1 つ以上のバックアップをクラウドストレージから削除し、関連する WAL を削除するために使用されるスクリプト。
- `barman-cloud-backup-keep` クラウドストレージのバックアップにアーカイブバックアップとしてフラグを立てるために使用されるスクリプト。このようなバックアップは、適用される保持ポリシーに関係なく、永久に保持されます。
- `barman-cloud-backup-list` クラウドストレージから `barman-cloud-backup` で取得した Barman バックアップのコンテンツをリストするために使用されるスクリプト。
- `barman-cloud-backup-show barman-cloud-backup` で取得した Barman バックアップのメタデータを表示するために使用されるスクリプト。
- `barman-cloud-restore` クラウドストレージから `barman-cloud-backup` で直接取得したバックアップを復元するために使用するスクリプト。

これらのコマンドには、使用するクラウドプロバイダーの適切なライブラリが必要です。

- AWS S3 [boto3][boto3]
- Azure Blob Storage [azure-storage-blob][azure-storage-blob] およびオプションで [azure-identity][azure-identity]
- Google Cloud Storage [google-cloud-storage][google-cloud-storage]

aws-s3 クラウドプロバイダーの資格情報をセットアップする方法については、[“Credentials” セクションの Boto 3 documentation][boto3creds] を参照してください。

azure-blob-storage クラウドプロバイダーの資格情報については、[“Environment variables for authorization parameters” セクションの Azure documentation][azure-storage-auth] を参照してください。次の環境変数がサポートされています。AZURE\_STORAGE\_CONNECTION\_STRING、AZURE\_STORAGE\_KEY、および AZURE\_STORAGE\_SAS\_TOKEN。Azure Active Directory を介して認証するために、`--credential` オプションを使用して `azure-cli` または `managed-identity` 資格情報を指定することもできます。

## 23.1 インストール

クラウド用の Barman クライアントユーティリティは、Barman をバイパスしてクラウドプロバイダーに直接バックアップする PostgreSQL サーバーにインストールする必要があります。

`barman-cloud-backup` および/または `barman-cloud-wal-archive` をフックスクリプトとして使用する場合、Barman サーバーに `barman-cli-cloud` パッケージをインストールすることもできます。

リポジトリをインストールするには、メインの「インストール」セクションを参照してください。

`root` として RedHat / CentOS システムにパッケージをインストールするには、次のように入力します。

```
yum install barman-cli-cloud
```

Debian / Ubuntu では、`root` ユーザーとして次のように入力します。

```
apt-get install barman-cli-cloud
```

## 23.2 barman-cloud フックスクリプト

上記のように、Barman サーバーに `barman-cli-cloud` パッケージをインストールします。

`barman-cloud-backup` は、次の Barman バックアップフレーバーのバックアップ後スクリプトとして使用できます。

- `backup_method = rsync` で取得したバックアップ。
- `backup_compression` を使用しない `backup_method = postgres` で取得したバックアップ。

これを行うには、Barman のサーバー構成に次を追加します。

```
post_backup_retry_script = barman-cloud-backup [ *OPTIONS* ] *DESTINATION_URL* $
→ {BARMAN_SERVER}
```

**警告:** フックスクリプトとして実行する場合、`barman-cloud-backup` は、バックアップのステータスが `DONE` である必要があり、バックアップが他のステータスの場合、失敗します。このため、バックアップは `-w/--wait` オプションを使用して実行して、バックアップのステータスが `WAITING_FOR_WALS` であるときにフックスクリプトが実行されないようにすることをお勧めします。

PostgreSQL サーバーの Barman 構成に次を追加することにより、プレ WAL アーカイブスクリプトとして `barman-cloud-wal-archive` を構成します。

```
pre_archive_retry_script = barman-cloud-wal-archive [ *OPTIONS* ] *DESTINATION_URL*
→ ${BARMAN_SERVER}
```

## 23.3 クラウドプロバイダーの選択

`--cloud-provider` オプションを使用して、バックアップと WAL のクラウドプロバイダーを選択します。これは、次のいずれかに設定できます。

- `aws-s3` [DEFAULT] AWS S3 または S3 互換オブジェクトストア。
- `azure-blob-storage` Azure Blob Storage サービス。
- `google-cloud-storage` Google Cloud Storage サービス。

## 23.4 プロバイダーによる特異性

### 23.4.1 Google Cloud Storage

#### 23.4.2 セットアップ

`google_storage_client` 依存関係が必要です。

```
pip3 install google-cloud-storage
```

資格情報を設定するには

- [Create a service account](#)

そして、サービスアカウントキーを作成します。

- バケットのアクセス権限を設定します。

バケットのサービスアカウントに [Storage Admin Role](#) を付与することをお勧めします。

- `barman_cloud` を使用する場合バケットが存在しない場合、作成されます。デフォルトオプションはバケットの作成に使用されます。バケットに特定のオプションリージョン、ストレージクラス、ラベルを含める必要がある場合は、ニーズをすべて満たすようにバケットを作成および設定することをお勧めします。
- `env variable` `GOOGLE_APPLICATION_CREDENTIALS` をサービスアカウントキーファイルパスに設定します。

`postgresarchive_command` または `restore_command` から `barman` クラウドを実行している場合、`postgres` 環境ファイルで `GOOGLE_APPLICATION_CREDENTIALS` を設定することを忘れないでください。

### 23.4.3 使用法

一部の詳細は、すべてのバーマンクラウドコマンドに固有です。

- Google Cloud Storage `--cloud-provider=google-cloud-storage` を選択します
- `SOURCE_URL` は、`gs` と `https` 形式の両方をサポートしています。例

```
gs://BUCKET_NAME/path
or
https://console.cloud.google.com/storage/browser/BUCKET_NAME/path
```

## 23.5 barman-cloud とスナップショットのバックアップ

`barman-cloud` クライアントユーティリティは、クラウドオブジェクトストアにアップロードする代わりに、クラウドスナップショットを使用してバックアップを作成および管理することもできます。

この方法で `barman-cloud` を使用する場合、バックアップデータはボリュームスナップショットとしてクラウドプロバイダーによって保存され、`WAL` と `backup_label` を含むバックアップメタデータはクラウドオブジェクトストレージに保存されます。

前提条件は `same as for snapshot backups using Barman` で、`barman-cloud` によって使用される資格情報がオブジェクトストアに対して読み取り/書き込み/更新操作を実行できる必要があるという要件が追加されています。

スナップショットの### `barman-cloud-backup`

`barman-cloud` でスナップショットバックアップを取得するには、次の追加引数とともに `barman-cloud-backup` を使用します。

- `--snapshot-disk` 複数のディスクに複数回使用できます
- `--snapshot-instance`

`--cloud-provider` が `google-cloud-storage` の場合、次の引数も必要です。

- `--gcp-project`
- `--gcp-zone`

--cloud-provider が azure-blob-storage の場合、次の引数も必要です。

- --azure-subscription-id
- --azure-resource-group

--cloud-provider が aws-s3 の場合、次のオプションの引数を使用できます。

- --aws-profile
- --aws-region

クラウドスナップショットが要求される場合、次のオプションは `barman-cloud-backup` では使用できません。

- --bzip2、--gzip または --snappy
- --jobs

バックアップが作成されると、`barman-cloud-backup-delete` や `barman-cloud-backup-keep` などの標準の `barman-cloud` コマンドを使用して管理できます。

### 23.5.1 barman-cloud-restore for スナップショット

`barman-cloud` を使用してスナップショットバックアップからリカバリーするプロセスは、`barman backups` のプロセスと非常に似ていますが、リカバリインスタンスがプロビジョニングされたら、`barman recover` の代わりに `barman-cloud-restore` を実行する必要があります。これは、`barman recover` と同じリカバリ前チェックを実行し、バックアップラベルをリカバリインスタンスの所定の場所にコピーします。

リカバリインスタンスのプロビジョニングに必要なスナップショットメタデータは、`barman-cloud-backup-show` を使用して照会できます。

オブジェクトストア型バックアップで `barman-cloud-restore` を使用する場合と同様に、コマンドはリカバリのために PostgreSQL を準備しないことに注意してください。PostgreSQL が起動する前に必要な PITR オプション、カスタム `restore_command` 値、または WAL ファイルは、手動または外部ツールで処理する必要があります。

クラウドスナップショットで作成されたバックアップを復元するときは、`barman-cloud-restore` とともに次の追加引数を使用する必要があります。

- --snapshot-recovery-instance

gcp プロバイダーには、次の追加引数が必要です。

- --gcp-zone

azure プロバイダーには、次の追加引数が必要です。

- --azure-resource-group

aws-s3 プロバイダーでは、次の追加引数を使用できます。

- --aws-region

クラウドスナップショットバックアップを復元する場合、`--tablespace` オプションは `barman-cloud-restore` とともに使用できません。



## 第24章 障害対応

### 24.1 Barman インストールを診断する

次を使用して、構成されているすべてのサーバーのステータスに関する重要な情報を収集できます。

```
barman diagnose
```

`diagnose` コマンド出力は `barman` サーバーの完全なスナップショットであり、グローバル構成、SSH バージョン、Python バージョン、`rsync` バージョン、PostgreSQL クライアントバージョン、すべてのサーバーの現在の構成とステータスなどの役立つ情報を提供します。

`diagnose` コマンドは、Barman インストールのステータスに関するグローバルビューを提供するため、問題のトラブルシューティングに非常に役立ちます。

### 24.2 ヘルプの要求

Barman は広範囲に文書化されていますが、カバーされていないシナリオがたくさんあります。

Barman および Barman を使用したディザスタリーカバリーシナリオに関するご質問は、コミュニティメーリングリストを使用して開発チームに連絡できます。

<https://groups.google.com/group/pgbarman>

またはフリーノードの IRC チャンネル `irc://irc.freenode.net/barman`

バグを発見したイベントは、GitHub を使用してチケットを開くことができます。 <https://github.com/EnterpriseDB/barman/issues>

EnterpriseDB は、Barman に 24 時間 365 日サービスを含む専門的なサポートを提供します。

#### 24.2.1 バグの送信

Barman は広範囲にテストされ、現在いくつかの運用環境で使用されています。ただし、他のソフトウェアと同様に、Barman にもバグがないわけではありません。

バグを発見した場合は、次の手順に従ってください。

- `barman diagnose` コマンドを実行します
- 上記の診断コマンド `barman diagnose`

**警告:** fence コマンドの出力は、セキュリティの観点から潜在的に危険な情報を開示する可能性があるため、送信するときに注意してください。

## 第25章 Barman プロジェクト

### 25.1 サポートとスポンサーの機会

Barman は、EnterpriseDB によって作成および保守されているフリーソフトウェアです。Barman の使用に関するサポートが必要な場合、または新機能が必要な場合は、EnterpriseDB に連絡してください。あなたは、オープンソースとして一般に公開される Barman および PostgreSQL の新機能の開発をスポンサーすることができます。

詳細については、次をご覧ください。

- [Barman website ][11]
- [サポートセクション][12]
- [EnterpriseDB website][13]
- [Barman FAQs ][14]
- [2ndQuadrant blog: Barman][15]

### 25.2 Barman への貢献

EnterpriseDB には、Barman のコードの改善に時間と専門知識を捧げるソフトウェアエンジニア、アーキテクト、データベース管理者、システム管理者、QA エンジニア、開発者、マネージャーのチームがいます。私たちはソフトウェア開発にリーンでアジャイルな方法論を採用し、クロスファンクションのコラボレーションを介して厳密なテスト手順を実装できる\_devops\_文化を信じています。すべての Barman コミットは、制作パイプラインのさまざまな段階での複数の個人の貢献です。

これは Barman を開発する私たちの推奨する方法ですが、次の場合に限り、外部開発者からのパッチを喜んで受け入れます。

- ユーザードキュメントチュートリアルとマニュアルページが提供されます。
- ソースコードが適切に文書化され、関連するコメントが含まれています。
- 提供されたコードは単体テストの対象です。
- 無関係な機能が侵害または破損することはありません。
- ソースコードは、現在のマスターブランチに基づいてリベースされています。
- コミットとプル要求は単一の機能に制限されますマルチ機能パッチはテストとレビューが困難です。
- ユーザーインターフェイスの変更は、EnterpriseDB と事前に議論されます。

また、投稿には、著作権の譲渡と、開発者の雇用主からのレンタル作品の所有権請求の免責事項を提供することを求めています。

この目的に、GitHub のプル要求システムを使用できます。

## 25.3 著者

アルファベット順

- アンドレ・マルケジーニ
- バーバラ・ライデンス
- ジュリオ・カラコッチ
- グスタボ・オリベイラ
- イスラエル・パース
- マルティン・マルケス

過去の寄稿者アルファベット順

- アビジット・メノンセン 建築家
- Anna Bellandi QA/テスト
- Britt Cole ドキュメントレビューアー
- Carlo Ascani 開発者
- Didier Michel 開発者
- Francesco Canovai QA/テスト
- ガブリエレ・バルトリーニ 建築家
- Gianni Ciolli QA/テスト
- Giulio Calacoci 開発者
- Giuseppe Broccolo 開発者
- Jane Threefoot 開発者
- Jonathan Battiato QA/テスト
- Leonardo Cecchi 開発者
- Marco Nenciarini プロジェクトリーダー
- Michael Wallace 開発者
- ニッコロ・フェイ QA/テスト
- Rubens Souza QA/テスト
- Stefano Bianucci 開発者

## 25.4 リンク

- [check-barman][16] Barman 用の Nagios プラグイン、Holger Hamann によって書かれました MIT ライセンス
- [puppet-barman][17] Puppet の Barman モジュール GPL
- [チュートリアル on “CentOS 7 で Barman を使用して PostgreSQL データベースをバックアップ、復元、および移行する方法”][26]、Sadequl Hussain DigitalOcean コミュニティで入手可能
- [BarmanAPI][27] Mehmet Emin KarakaşGPL によって書かれた Barman 用の RESTful API

## 25.5 ライセンスと貢献

Barman は EnterpriseDB UK Limited の所有物であり、そのコードは GNU General Public License 3 に基づいて配布されています。

© Copyright EnterpriseDB UK Limited 2011-2023

Barman は、欧州委員会の第 7 次フレームワークプログラムの資金提供による研究プロジェクト [4CaaSt][18] を介して部分的に資金提供されています。

Barman への貢献は大歓迎です。AUTHORS ファイルにリストされます。EnterpriseDB UK Limited では、投稿には著作権の譲渡と、開発者の雇用主からのレンタル作品の所有権の免責事項を提供することを求めています。これにより、すべての Barman ディストリビューションがフリーコードのままであることを確認できます。関連する著作権譲渡フォームのコピーについては、[barman@enterprisedb.com](mailto:barman@enterprisedb.com) までご連絡ください。



## 第26章 参考リンク

[rpo] [https://en.wikipedia.org/wiki/Recovery\\_point\\_objective](https://en.wikipedia.org/wiki/Recovery_point_objective)

[rto] [https://en.wikipedia.org/wiki/Recovery\\_time\\_objective](https://en.wikipedia.org/wiki/Recovery_time_objective)

[repmgr] <https://www.repmgr.org/>

[sqldump] <https://www.postgresql.org/docs/current/static/backup-dump.html>

[物理的バックアップ] <https://www.postgresql.org/docs/current/static/backup-file.html>

[adminbook] <https://www.2ndquadrant.com/en/books/postgresql-10-administration-cookbook/>

[wal] <https://www.postgresql.org/docs/current/static/wal.html>

[49340627f9821e447f135455d942f7d5e96d] <https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=49340627f9821e447f135455d942f7d5e96cae6d>

[requirements\_recovery] <https://www.postgresql.org/docs/current/static/warm-standby.html#STANDBY-PLANNING>

[yumpgdg] <https://yum.postgresql.org/>

[aptpgdg] <https://apt.postgresql.org/>

[aptpgdgwiki] <https://wiki.postgresql.org/wiki/Apt>

[epel] <https://fedoraproject.org/wiki/EPEL>

[man5] <https://docs.pgbarman.org/barman.5.html>

[setup\_user] <https://docs.python.org/3/install/index.html#alternate-installation-the-user-scheme>

[pypi] <https://pypi.python.org/pypi/barman/>

[pgpass] <https://www.postgresql.org/docs/current/static/libpq-pgpass.html>

[pghba] <https://www.postgresql.org/docs/current/static/client-authentication.html>

[authpghba] <https://www.postgresql.org/docs/current/static/auth-pg-hba-conf.html>

[streamprot] <https://www.postgresql.org/docs/current/static/protocol-replication.html>

[replication-slots] <https://www.postgresql.org/docs/current/static/warm-standby.html#STREAMING-REPLICATION-SLOTS>

[synch] <https://www.postgresql.org/docs/current/static/warm-standby.html#SYNCHRONOUS-REPLICATION>

[target] <https://www.postgresql.org/docs/current/static/runtime-config-wal.html#RUNTIME-CONFIG-WAL-RECOVERY-TARGET>

[2ndrpmrepo] <https://rpm.2ndquadrant.com/>

[2ndqdebrepo] <https://apt.2ndquadrant.com/>

[boto3] <https://github.com/boto/boto3>

[boto3creds] <https://boto3.amazonaws.com/v1/documentation/api/latest/guide/configuration.html>

[azure-identity] <https://docs.microsoft.com/en-us/python/api/azure-identity/?view=azure-python>

[azure-storage-blob] <https://docs.microsoft.com/en-us/python/api/azure-storage-blob/?view=azure-python>

[azure-storage-auth] <https://docs.microsoft.com/en-us/azure/storage/blobs/authorize-data-operations-cli#set-environment-variables-for-authorization-parameters>

[google-cloud-storage] <https://cloud.google.com/storage/docs/reference/libraries>

[pg\_basebackup-documentation] <https://www.postgresql.org/docs/current/app-pgbasebackup.html>

[pg-backup-api] <https://github.com/EnterpriseDB/pg-backup-api>

[config-options] <https://docs.pgbarman.org/barman.5.html#options>

[barman-downloads] <https://pgbarman.org/downloads/>

[python-2-sunset] <https://www.python.org/doc/sunset-python-2/>

[psql] <https://www.postgresql.org/docs/current/app-psql.html>

[snapshot-recovery-runbook-azure] [https://github.com/EnterpriseDB/barman/blob/master/doc/runbooks/snapshot\\_recovery\\_azure.md](https://github.com/EnterpriseDB/barman/blob/master/doc/runbooks/snapshot_recovery_azure.md)

[snapshot-recovery-script] [https://github.com/EnterpriseDB/barman/blob/master/scripts/prepare\\_snapshot\\_recovery.py](https://github.com/EnterpriseDB/barman/blob/master/scripts/prepare_snapshot_recovery.py)

[postgres-low-level-base-backup] <https://www.postgresql.org/docs/current/continuous-archiving.html#BACKUP-LOWLEVEL-BASE-BACKUP>

[pgbarman-barman-cloud-backup-show] <https://docs.pgbarman.org/release/latest/barman-cloud-backup-show.1.html>

[8] [https://en.wikipedia.org/wiki/Hard\\_link](https://en.wikipedia.org/wiki/Hard_link)

[11] <https://www.pgbarman.org/>

[12] <https://www.pgbarman.org/support/>

[13] <https://www.enterprisedb.com/>

[14] <https://www.pgbarman.org/faq/>

[15] <https://blog.2ndquadrant.com/tag/barman/>

[16] <https://github.com/hamann/check-barman>

[17] <https://github.com/2ndquadrant-it/puppet-barman>

[18] <https://4caast.morfeo-project.org/>

[20] <https://www.postgresql.org/docs/current/static/functions-admin.html>

- [24] <https://www.postgresql.org/docs/current/static/warm-standby.html#STREAMING-REPLICATION>
- [25] <https://www.postgresql.org/docs/current/static/app-pgreceivewal.html>
- [26] <https://goo.gl/218Ghl>
- [27] <https://github.com/emin100/barmanapi>
- [31] <https://www.postgresql.org/>