



EDB

Barman Manual

リリース **3.4.0**

EnterpriseDB UK Limited

2023 年 02 月 10 日

目次

第 1 章 Introduction	3
第 2 章 Before you start	5
第 3 章 Design and architecture	7
3.1 Barman をインストールする場所	7
3.2 1 つの Barman、多くの 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 によるバックアップ	12
第 4 章 System requirements	15
4.1 バックアップの要件	15
4.2 回復の要件	16
第 5 章 Installation	17
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	17
5.2 パッケージを使用した Debian / Ubuntu へのインストール	18
5.3 パッケージを使用した SLES へのインストール	18
5.4 ソースからのインストール	19
5.5 PostgreSQL クライアント/サーバーバイナリ	19
第 6 章 Upgrading Barman	21
6.1 Barman 3.0.0 へのアップグレード	21
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	21
6.1.2 メタデータの変更	21
6.2 Barman 2.10 からのアップグレード	22
6.3 Barman 2.X (2.8 より前) からのアップグレード	22
6.4 Barman 1.X からのアップグレード	22
第 7 章 Configuration	23
7.1 オプションの範囲	23
7.2 構成例	24
第 8 章 Setup of a new server in Barman	27
第 9 章 準備手順	29
9.1 PostgreSQL 接続	29

9.2	PostgreSQL WAL アーカイブとレプリケーション	30
9.3	PostgreSQL ストリーミング接続	31
9.4	SSH 接続	32
9.5	postgres ユーザーの SSH 構成	32
9.6	barman ユーザーの SSH 設定	32
9.7	PostgreSQL から Barman へ	32
9.8	Barman から PostgreSQL へ	33
第 10 章 サーバー構成ファイル		35
第 11 章 WAL ストリーミング		37
11.1	レプリケーションスロット	38
11.2	WAL ストリーミングの構成方法	38
11.3	リカバリによる部分的な WAL ファイルの制限	39
第 12 章 archive_command を介した WAL アーカイブ		41
12.1	barman-wal-archive を介した WAL アーカイブ	41
12.2	rsync / SSH を介した WAL アーカイブ	42
第 13 章 WAL アーカイブ構成の検証		45
第 14 章 ストリーミングバックアップ		47
第 15 章 rsync /SSH でバックアップ		49
第 16 章 クラウドスナップショットによるバックアップ		51
16.1	クラウドスナップショットの前提条件	51
16.2	Google Cloud Platform のスナップショットの前提条件	51
16.3	スナップショットバックアップの構成	52
16.4	Google Cloud Platform のスナップショットの構成	53
16.5	スナップショットバックアップの作成	53
第 17 章 Windows ベースのサーバーのセットアップ方法		55
第 18 章 General commands		57
18.1	cron	57
18.2	diagnose	58
18.3	list-servers	58
第 19 章 Server commands		59
19.1	archive-wal	59
19.2	backup	59
19.3	check	60
19.4	generate-manifest	60
19.5	get-wal	60
19.6	list-backups	62
19.7	rebuild-xlogdb	62
19.8	receive-wal	63
19.8.1	receive-wal プロセス管理	63
19.8.2	レプリケーションスロット管理	63

19.9	replication-status	64
19.10	show-servers	64
19.11	status	64
19.12	switch-wal	64
19.13	verify	65
第 20 章	Backup commands	67
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	delete	68
20.4	keep	68
20.5	list-files	69
20.6	recover	69
20.6.1	リモートリカバリー	70
20.6.2	表領域のリマッピング	71
20.6.3	ポイントインタイムリカバリ	71
20.6.4	Barman サーバーから WAL を取得する	72
20.6.5	圧縮されたバックアップの回復	72
20.7	show-backup	73
第 21 章	Features in detail	75
21.1	バックアップ機能	75
21.1.1	インクリメンタルバックアップ	75
21.1.2	帯域幅の使用を制限する	76
21.1.3	ネットワーク圧縮	77
21.1.4	バックアップ圧縮	77
21.1.5	圧縮アルゴリズム	77
21.1.6	コンプレッションワーカー	78
21.1.7	圧縮レベル	78
21.1.8	圧縮場所	78
21.1.9	圧縮形式	79
21.1.10	同時バックアップ	79
21.1.11	スタンバイの同時バックアップ	80
21.1.12	即時チェックポイント	81
21.1.13	ローカルバックアップ	81
21.2	アーカイブ機能 ### WAL 圧縮	82
21.2.1	同期 WAL ストリーミング	82
21.3	カタログ管理機能 ### 最小限の冗長性の安全性	83
21.3.1	保持ポリシー	84
21.3.2	スコープ	84
21.3.3	それらの仕組み	85
21.3.4	構成と構文	85
21.4	フックスクリプト	86
21.4.1	バックアップスクリプト	87
21.4.2	バックアップ削除スクリプト	88
21.4.3	WAL アーカイブスクリプト	88
21.4.4	WAL 削除スクリプト	89

21.4.5	リカバリースクリプト	89
21.5	カスタマイズ	90
21.5.1	ロックファイルディレクトリ	90
21.5.2	バイナリパス	90
21.6	クラスター管理システムとの統合	91
21.7	パラレルジョブ	91
21.8	地理的な冗長性	92
21.8.1	情報を同期	92
21.8.2	構成	92
21.8.3	ノードの同期	93
21.8.4	手動同期	93
21.9	クラウドスナップショットのバックアップ	93
21.9.1	スナップショットバックアップの詳細	94
21.9.2	スナップショットバックアップからのリカバリ	94
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	96
第 22 章	Barman client utilities (barman-cli)	99
22.1	インストール	99
第 23 章	Barman client utilities for the Cloud (barman-cli-cloud)	101
23.1	インストール	102
23.2	barman-cloud フックスクリプト	103
23.3	クラウドプロバイダーの選択	103
23.4	プロバイダー別の特異性	103
23.4.1	Google Cloud Storage	103
23.4.2	設定	103
23.4.3	使用法	104
23.4.4	使用法	104
23.5	barman-cloud とスナップショットのバックアップ	105
23.5.1	スナップショットの barman-cloud-backup	105
23.5.2	スナップショットの barman-cloud-restore	105
第 24 章	Troubleshooting	107
24.1	Barman インストールを診断する	107
24.2	助けを求める	107
24.2.1	バグの送信	107
第 25 章	The Barman project	109
25.1	サポートとスポンサーの機会	109
25.2	Barman への貢献	109
25.3	著者	110
25.4	リンク	111
25.5	ライセンスと貢献	111
第 26 章	Reference links	113

EnterpriseDB UK Limited 2023 年 1 月 26 日 (3.4.0)

Barman (Backup and Recovery Manager) は、Python で記述された PostgreSQL サーバーのディザスタリカバリー用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、復旧フェーズ中にリスクを軽減し、DBA を支援できます。

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

!!!重要

このマニュアルは、読者が理論的なディザスタリカバリーの概念に精通していること、および物理バックアップとディザスタ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「始める前に」のセクションを参照してください。

第1章 Introduction

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、「不測の事態」が発生した場合に備えておくことが重要です。データベースのシナリオでは、予期せぬことが次のいずれかの形で発生する可能性があります。

- データ破損
- システム障害（ハードウェア障害を含む）
- ヒューマンエラー
- 自然災害

このような場合、ICT マネージャーまたは DBA は、可能な限り短時間でインシデントを修正し、データベースを復旧できるはずです。通常、この規律をディザスターリカバリー、より広義にはビジネス継続性と呼びます。

ビジネス継続性では、ウィキペディアで定義されている 2 つの基本的なメトリックを理解することが重要です。

- **Recovery Point Objective (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対象期間」
- **Recovery Time Objective (RTO)**
：「ビジネス継続性の中断に関連する容認できない結果を回避するために、災害（または混乱）後にビジネスプロセスを復元する必要があるターゲット期間とサービスレベル」

簡単に言うと、RPO は損失しても許容される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、私たちは皆、**RPO=0**（「データ損失ゼロ」）と **RTO=0**（ダウンタイムゼロ、ユートピア）を望んでいます-たとえそれが祖母のレシピ Web サイトであっても。実際には、慎重なコスト分析フェーズにより、ビジネス継続性の要件を判断できます。

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では、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** のような高可用性ソリューションの焦点です。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 と repmgr を使用した PostgreSQL オープンソースクラスターは、適切に構成して監視した場合、1 年で 99.99 % を超えるアップタイムを簡単に達成できることを確認できます。

いずれにせよ、実際のツールよりも、災害復旧に関連する文化的側面を強調することが重要です。人なくして道具は役に立たない。

Barman の使命は、次のようなディザスタ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る
- 回復手順にさらに焦点を当てています
- チームメンバーのための PostgreSQL のクラッシュリカバリ、バックアップ、ポイントインタイムリカバリ、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と見なされます）（Barman のフックスクリプトで創造的にしてください！）
- Devops チームのすべてのメンバー（はい、システム管理者や DBA だけでなく、開発者）による回復手順の定期的な実践を促進します
- 3〜6 か月ごとにチームと定期的にスケジュールされた訓練と災害復旧シミュレーションを要請します
- PostgreSQL と Barman の継続的な監視に依存しており、異常を迅速に特定できます

さらに、災害が発生したとき（はい、いつ）に備えて、自分とチームにできることをすべて行います。

- それは金曜日の夜になるでしょう。ほとんどの場合、ちょうどオフィスを出ようとしています。
- あなたが休暇中（世界一周クルーズの真っ最中）で、他の誰かが対処しなければならないとき。
- 確かにストレスになります。
- 利用可能な最後のバックアップが有効かどうかを確認できないことを後悔するでしょう。
- 回復するのにおよそどれくらい時間がかかるか分からない限り、毎秒は永遠のように思えます。

怖がらないで、準備してください。

2011 年、これらの目標を念頭に置いて、2ndQuadrant は、現在 PostgreSQL の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 Before you start

Barman を使い始める前に、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、レプリケーションに関する概念に精通することが基本です。

以下に、読むことをお勧めするリソースの非包括的なリストを示します。

PostgreSQL ドキュメント：

- [SQL Dump \[pgdump\]](#)
- [File System Level Backup](#)
- [Continuous Archiving and Point-in-Time Recovery \(PITR\)](#)
- [Reliability and the Write-Ahead Log](#)

Book： [PostgreSQL 10 Administration Cookbook](#)

- **[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理解することが重要です。

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。1 年中いつでも、EnterpriseDB などの PostgreSQL 企業が提供する世界中で多くのコースを見つけることができます。

第3章 Design and architecture

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する機能です。

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場所に置くことができます。現実的には、Barman サーバーを PostgreSQL サーバーから離れすぎないようにして、バックアップとリカバリの両方の時間を制御します。

Barman をセットアップする「フリーサイズ」の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
- 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 を監視インフラストラクチャ [nagios] と統合
- 実稼働に展開する前にすべてをテストします

[nagios]: Barman の最も重要な機能の 1 つであり、真の命の恩人である `barman check --nagios` コマンドのおかげで、Nagios / Icinga との統合は簡単です。

ディザスター リカバリーアーキテクチャのモデル化を開始する合理的な方法は次のとおりです。

- PostgreSQL および Barman に関して、次のようないくつかの可能なアーキテクチャを設計します。

1. 同じデータセンター
2. 同じ大都市圏の別のデータセンター
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する
- 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、最初の解決策を実装する

そうは言っても、Barman の非常に一般的なセットアップは、PostgreSQL サーバーがあるのと同じデータセンターにインストールすることです。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、Barman が提供するバックアップ層の数を増やす 2 つの機能のおかげで軽減できます。

1. 地理的な冗長性（Barman 2.6 で導入）
2. フックスクリプト

地理的な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 インスタンスを利用して、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 では地理的冗長性をグローバルレベルだけでなくサーバーレベルでも構成できるため、一部のサーバーがローカルの PostgreSQL サーバーに直接接続され、他のサーバーがサブセットをバックアップする 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）を示しています。それぞれには、ローカルの Barman インストールでバックアップされ、他の Barman サーバーで中継されます（パッシブ）rsync / SSH を介した多層バックアップの場合。geo 冗長性の詳細については、特定のセクションを参照してください。

Backup architecture for PostgreSQL

Example of geographical redundancy

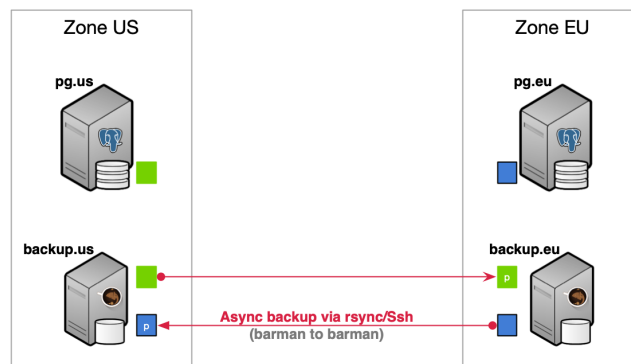


図 1: An example of architecture with geo-redundancy:raw-latex:label{georedundancy-design}

代わりにフックスクリプトのおかげで、Barman のバックアップは、tar を介したテープ、または Amazon クラウドの S3 バケットなどのさまざまなメディアにエクスポートできます。

決定は永遠ではないことを忘れないでください。この方法で始めて、時間をかけて自分に最適なソリューションに適応できます。ただし、最初はシンプルに保つようにしてください。

3.2 1つの Barman、多くの PostgreSQL サーバー

Barman によって最初に導入された別の関連機能は、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的に保存できます。[recver]

.....

[recver] は、セクション「リカバリーの要件」で詳しく説明されているように、リカバリーに適用されます。

その結果、複雑なディザスタリカバリーアーキテクチャをモデル化して、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には、独自の方法があります。実際の実験とテストに基づいて、あなたの心に響くものを選択してください。最も重要なことです。

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ています。

- 1 つの PostgreSQL インスタンス（ホスト名 `pg`）
- Barman を備えた 1 つのバックアップサーバー（ホスト名 `backup`）

3.3 ストリーミングバックアップと `rsync` / `SSH`

Barman は、転送メカニズムとして `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必要がありますが、一般的な使用法では、現在サポートされているすべてのバージョンの PostgreSQL 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。

重要: Barman は `pg_basebackup` を透過的に利用するため、増分バックアップ、並列バックアップ、重複排除などの機能は現在使用できません。この場合、`rsync` を介した従来の方法と比較して、帯域幅の制限にはいくつかの制限があります。

`pg_basebackup` の制限が発生するすべての場合（たとえば、増分バックアップと重複排除の恩恵を受けることができる非常に大規模なデータベース）には、`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、私たちの経験に基づく、従来のバックアップよりもセットアップが簡単だからです。また、ストリーミングバックアップを使用すると、Windows [windows] で PostgreSQL サーバーをバックアップでき、Docker での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、まだ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、「Windows ベースのサーバーのセットアップ方法」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 ファイルとも呼ばれます）の再生に依存しています。したがって、Barman は WAL ファイルをベースバックアップと一緒に保存して、回復時に使用できるようにすることが不可欠です。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 ストリーミングでは、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します。WAL ストリーミングはデータ損失のリスクを減らし、RPO をゼロに近い値に下げることができます。PostgreSQL クラスターに Barman を同期 WAL レシーバーとして追加し、**データ損失なし**（RPO = 0）を達成することもできます。

Barman は、PostgreSQL の `archive_command`（`rsync` / `SSH` を介して、または `barman-cli` パッケージの `barman-wal-archive` を介して）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 ファイルに切り替えた場合にのみ 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 相当のデータ変更ごとに発生します。

WAL ストリーミングまたは WAL アーカイブのいずれかが設定されていることが**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できます。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するため、WAL ストリーミングが失敗した場合でも WAL が Barman のアーカイブにコピーされます。

一般的な使用法では、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は、WAL アーカイブ*と*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かったため、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。サポートされているすべてのバージョンの PostgreSQL には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れば十分です。

3.5 バックアップの2つの一般的なシナリオ

作業を楽にするために、Barman の特定の PostgreSQL サーバーの最も一般的な2つのシナリオを以下にまとめます。

これは、Barman でバックアップすることを決定したすべてのサーバーに対して行う必要が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の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) のみについて考えます。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あります。

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。

このシナリオでは、次を構成する必要があります。

1. 管理、調整、および監視を目的とした 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とアーカイブの操作に SSH 接続を使用しないため、**ストリーミング専用**のセットアップと呼ばれます。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。

SSH 経由の WAL アーカイブには以下が必要です。

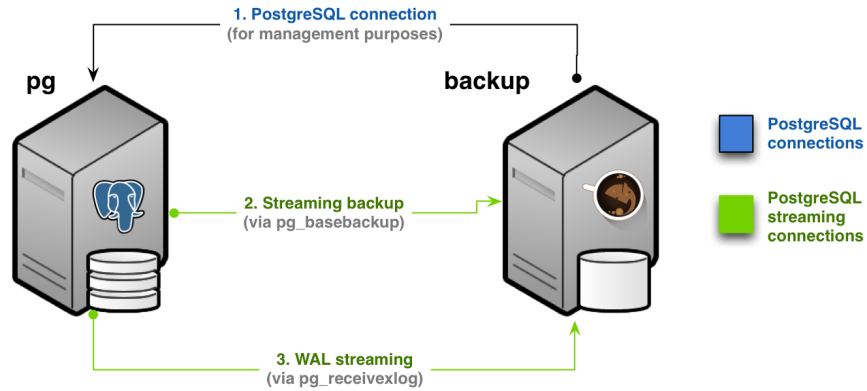


図 2: Streaming-only backup (Scenario 1):raw-latex:label{scenario1-design}

Backup architecture for PostgreSQL

Scenario 1b - Streaming backup with fallback WAL archiving

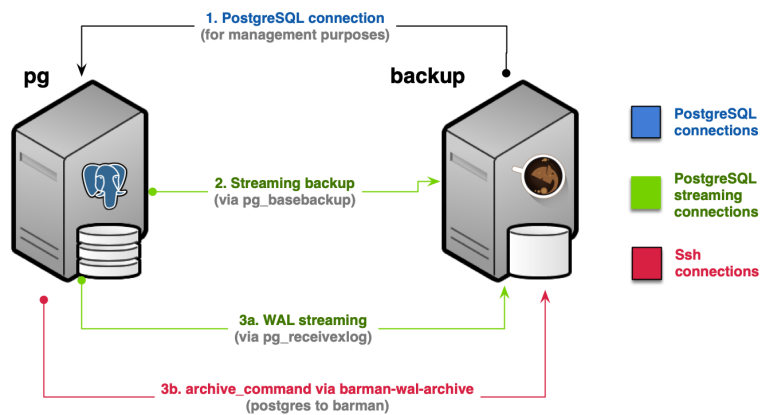


図 3: Streaming backup with WAL archiving (Scenario 1b):raw-latex:label{scenario1b-design}

- PostgreSQL サーバーの postgres ユーザーが Barman サーバーに barman ユーザーとして接続できるようにする追加の 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されています

3.5.2 シナリオ 2 : rsync /SSH によるバックアップ

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

- ファイルレベルの増分バックアップ
- 並列バックアップ
- テーブルスペースごとを含む、帯域幅使用量のより細かい制御

Backup architecture for PostgreSQL

Scenario 2 - Traditional Barman setup

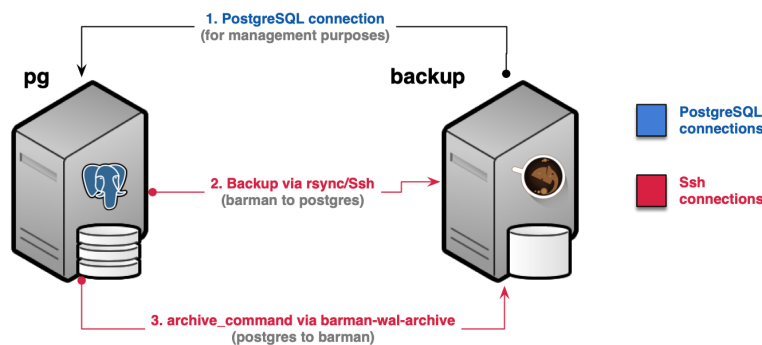


図 4: Scenario 2 - Backup via rsync/SSH

このシナリオでは、次を構成する必要があります。

1. 管理、調整、および監視を目的とした PostgreSQL への標準接続
2. Barman サーバーの barman ユーザーが PostgreSQL サーバーの postgres ユーザーとして接続できるようにする rsync が使用するベースバックアップ操作の SSH 接続
3. PostgreSQL の archive_command が使用し、PostgreSQL サーバーの postgres ユーザーが Barman サーバーに barman ユーザーとして接続できる WAL アーカイブ用の SSH 接続

ステップ 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**で説明されているように 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 2b - Traditional Barman setup with WAL streaming

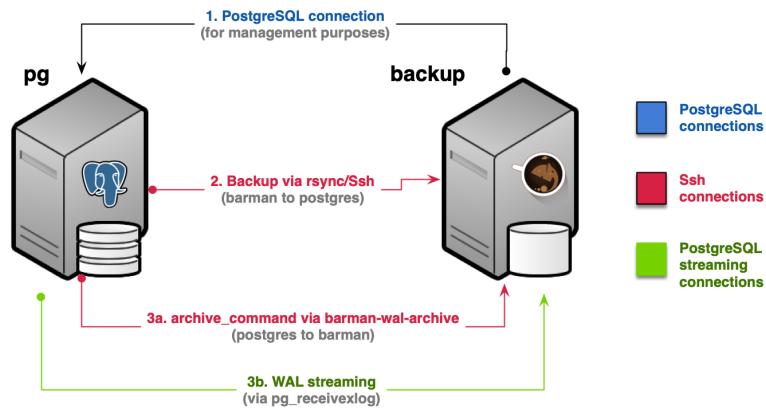


図 5: Backup via rsync/SSH with WAL streaming (Scenario 2b):raw-latex:label{scenario2b-design}

第4章 System requirements

- Linux/Unix
- Python ≥ 3.6
- Python モジュール：
 - argcomplete
 - psycpg2 $\geq 2.4.2$
 - python-dateutil
- セットアップツール
- PostgreSQL ≥ 10
- rsync $\geq 3.0.4$ (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 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 のサポートは終了しました。Python 2.7 のサポートは非推奨であり、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 で終了しました。

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、使用可能なディスク領域です。クラスターのサイズ、1日に生成される 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 の開発者は、XFS と ext4 で Barman を定期的にテストしています。

1つの Barman、多くの PostgreSQL サーバー と同様に、Barman は NFS に対して特別なことはしません。

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、hard および synchronous 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 回復の要件

Barman を使用すると、ローカル (Barman が存在する場所) またはリモート (別のサーバー上) で PostgreSQL インスタンスを回復できます。

リモートリカバリは、間違いなく Barman で PostgreSQL サーバーを復元する最も一般的な方法です。

いずれにせよ、同じ requirements for PostgreSQL's Log shipping and Point-In-Time-Recovery apply :

- 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、できるだけ似た回復環境を作成することを強くお勧めします。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 サーバーに専用のリカバリ環境を (オンデマンドでも) 使用すると、チームのディザスターリカバリ文化を育むことができます。復旧作業を練習し、慣れておくことで、不測の事態に備えることができます。

私たちの経験に基づいて、指定された復旧環境は実際の障害状況でのストレスの影響を軽減するため、復旧操作の有効性が向上します。

最後に、NTP などを使用して、サーバー間で時刻を同期することが重要です。

第5章 Installation

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

重要: Barman をインストールする推奨される方法は、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 派生物の 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](#) 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、その Web サイトにある指示に従って、EnterpriseDB から Yum を介して [public RPM repository](#) を介して配布されます。

次に、`root` として次のように入力します。

```
yum install barman
```

注釈: Barman 関連のパッケージは、PGDG リポジトリを介して更新されないようにすることをお勧めします。これを行うには、`/etc/yum.repos.d/pgdg-*.repo` ファイル内の Barman サーバーに含まれる PGDG リポジトリ定義に次の行を追加します。

edb_notranlate_2 これを行うことで、Barman ソフトウェアのパッケージ管理を EnterpriseDB の 2ndQuadrant リポジトリのみに依存します。

edb_notranlate_2 これを行うことで、Barman ソフトウェアのパッケージ管理を EnterpriseDB の 2ndQuadrant リポジトリのみに依存します。

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は、次の両方のリポジトリをインストールすることをお勧めします。

- [Public APT repository](#)、Barman 開発者が直接保守
- [PostgreSQL Community APT repository](#)、[APT section of the PostgreSQL Wiki](#)

注釈: Barman 開発者が PostgreSQL Community APT リポジトリプロジェクトに直接関与したおかげで、常に Barman の最新バージョンにアクセスできます。

Barman のインストールは簡単です。root ユーザーとして次のように入力します。

```
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](#) にある指示に従って、必要なりポジトリをインストールします。

サポートされている SLES バージョンは SLES 12 SP5 および SLES 15 SP3 です。

- SLES 12 のみ*: 次のように OpenSUSE python バックポートリポジトリを有効にする必要があります (この手順は SLES 15 では必要ありません)。

```
zypper addrepo https://download.opensuse.org/repositories/  
→devel:languages:python:backports/SLE_12_SP5/devel:languages:python:backports.repo  
zypper addrepo https://download.opensuse.org/repositories/devel:/languages:/python:/  
→backports/SLE_12_SP4 devel_languages_python_backports_sp4  
zypper refresh
```

必要なりポジトリがインストールされたら、root ユーザーとして Barman を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、熟練した GNU/Linux ユーザーのみが実行してください。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 サーバーで *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。

システム全体にインストールするには、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 アプリケーションがユーザーディレクトリ（*make sure that your 'PATH' environment variable is set properly* <<https://docs.python.org/3/install/index.html#alternate-installation-the-user-scheme>>`\_\_`）にインストールされます。

Barman is also available on the Python Package Index (PyPI) であり、*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。

- *backup_method = postgres* を使用した **ストリーミングバックアップ**と *rsync/SSH* (*pg_basebackup* が必要)
- *streaming_archiver = on* を使用した Streaming WAL archiving (*pg_receivewal* または *pg_receivexlog* 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で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。

*すべてのバイナリは/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、そのパスに適切な PG_MAJOR_VERSION の bin ディレクトリを持っています。
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 ディレクトリを指します。

上記のバイナリに加えて、(<https://www.postgresql.org/docs/current/app-psql.html>) [psql] プログラムをお勧めします。Barman は直接使用しませんが、ドキュメントには、PostgreSQL 接続が意図したとおりに機能していることを確認する方法の例が記載されています。psql バイナリは、PostgreSQL クライアントパッケージにあります。

第6章 Upgrading Barman

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、安定した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 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ついて、何千もの自動テストを実行します。

また、すべてのバージョンは以前のバージョンと互換性があります。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簡単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 は同時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まったくサポートされないことに注意してください。

6.1.2 メタデータの変更

バージョン 3.0.0 で作成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、`compression` という名前の新しいフィールドが追加されます。これは、`pg_basebackup` の組み込みの圧縮機能を使用して取得したバックアップから回復するときに使用されます。

このフィールドが存在することは、Barman の以前のバージョンでは Barman 3.0.0 で作成されたバックアップを読み取れ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 で取っ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が (<https://github.com/EnterpriseDB/pg-backup-api>) [`pg-backup-api`] に影響するため、`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` パッケージに移動されていることに注意してください。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、通常どおり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 Barman (PostgreSQL 9.6 以降に推奨) または `exclusive_backup` (現在のデフォルト) に設定する必要があります。。

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、次の行をグローバルに、またはそれを必要とするサーバーに追加してください。

```
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透過的に `archiver = on` を設定し、警告を出力します。

第7章 Configuration

Barman には 2 種類の構成ファイルがあります。

- グローバル/一般的な構成
- サーバー構成

メイン構成ファイル（デフォルトで `/etc/barman.conf` に設定）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。

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

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の両方を含む単一の構成ファイルを持つことができます。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 フォーマットに従います。

構成ファイルは、異なる種類のパラメーターを受け入れます。

- 文字列
- 列挙型
- 整数
- boolean、on/true/1 も受け入れられ、off/false/0 も受け入れられます。

それらのどれも引用する必要はありません。

注：一部の enum は off を許可しますが、false は許可しません。

7.1 オプションの範囲

すべての構成オプションにはスコープがあります。

- グローバル
- サーバー
- global/server : 一般的に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]` ラベルで識別される 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、構成ファイル内の角括弧で囲まれた行で指定されるサーバーセクションでのみ指定できます ([および])。server セクションは、そのサーバーの ID を Barman で表します。次の例では、pg という名前のサーバーのセクションを指定します。

```
[pg]
; Configuration options for the
; server named pg go here
```

Barman にはサーバー名として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、柔軟性を失うことなく構成が必要なオプションの数を削減しようとする**構成よりも規約**の設計パラダイムを実装します。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てサーバーレベルでオーバーライドできるため、ユーザーは一般的な動作を指定して 1 つ以上のサーバーに対して調整できます。これらのオプションには、global/server スコープがあります。

使用可能なすべての構成とその範囲のリストについては、[section 5 of the 'man' page](#) を参照してください。

```
man 5 barman
```

7.2 構成例

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、以下の例は、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
```

(次のページに続く)

(前のページからの続き)

```
streaming_archiver = on  
slot_name = 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、配布されている `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 Setup of a new server in Barman

「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ているサーバー ID とホスト名として `pg`
- Barman が置かれているホスト名として `backup`
- `backup` サーバーで Barman を実行しているユーザーとしての `barman`（構成の `barman_user` パラメーターで識別）
- `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、バックアップと復元の履歴全体で同じ PostgreSQL インスタンスを参照する必要があります（つまり、同じシステム識別子）。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しないでください。共通点がないため、別のサーバー定義を使用します。

第9章 準備手順

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行う必要があるいくつかの準備手順が含まれています。

!!!重要

続行する前に、「設計とアーキテクチャ」セクションで説明されているように、WAL アーカイブとバックアップ戦略を決定することが重要です。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 pg 上の PostgreSQL サーバーにスーパーユーザーとして接続できること、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。

次のように、PostgreSQL に 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、次のように必要な権限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_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;
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 15 ベータ版およびそれ以降の PostgreSQL バージョンでは、関数 `pg_start_backup` および `pg_stop_backup` の名前が変更され、シグネチャが異なります。したがって、上記のブロックの最初の 3 行を次のように置き換える必要があります。

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

実際のスーパーユーザーがいないと、`barman switch-wal` コマンドの`--force` オプションが機能しないことに注意してください。

!!!重要

上記の `createuser` コマンドはパスワードの入力を求めます。次に、パスワードを `backup` サーバー上の `~barman/.pgpass` ファイルに追加するように勧められます。詳細については、[The Password File](#) 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したり、監視したりするために必要です。

PostgreSQL が提供するものから好きな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](#) を参照してください。

`backup` ホストが `pg` ホスト上の PostgreSQL に接続できることを確認するには、`backup` ホストで `barman` ユーザとして次のコマンドを実行します。

```
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）を書き留めて、後で使用します。次の例のように、サーバー構成の `conninfo` オプションで必要になります。

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

注釈: `application_name` はオプションです。

9.2 PostgreSQL WAL アーカイブとレプリケーション

続行する前に、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ように、`pg` で PostgreSQL を適切に構成する必要があります。PostgreSQL ドキュメントの次のセクションをお読みください。

- [Role attributes](#)
- [The `pg\_hba.conf` file](#)
- [Setting up standby servers using streaming replication](#)

非常に重要な構成パラメーターの 1 つは、`wal_level` パラメーターです。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、このパラメーターを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場合は、ストリーミング接続を設定する必要があります。次のように、PostgreSQL に `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。パスワードを `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](#) 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す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、システム識別子、現在のタイムライン ID、および現在の WAL フラッシュ場所を含む応答が表示されます。

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、サーバーが管理できるレプリケーションスロット `[replslot94]` 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ストリーミング接続を介して 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]`: PostgreSQL 9.4 でレプリケーションスロットが導入されました。詳細については、「WAL ストリーミング/レプリケーションスロット」セクションを参照してください。

`max_replication_slots` および `max_wal_senders` に対して提案された値は例として検討する必要があります。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 の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ようにするプロトコルおよびツールのセットです。SSH の使用に関するその他のドキュメントは、Digital Ocean の記事

SSH Essentials にあります。

SSH キー交換は、異なる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方法であり、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: 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のみにストリーミング接続を使用する場合、この手順は必要ありません。

[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

以前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入力せずにホストから接続するために使用するため、キーペアの作成中にパスフレーズを入力しないでください。

9.6 barman ユーザーの SSH 設定

前の段落のように、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、パス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済みキーに構成する必要があります。

認証される公開鍵は、`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、その内容は、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、`600` をパーミッションとして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します。

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

`archive_command` 構成パラメーターの値については、「`archive` コマンドによる WAL アーカイブ」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、PostgreSQL サーバーで `postgres` ユーザーの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 `.ssh/id_rsa.pub` ファイルの内容は、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します。

```
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、次の内容で `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」に従って設定されています。

`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 アーカイブにストリーミング接続を使用するか、サーバーのバックアップを作成する予定がある場合は、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
```

この値は、セクション「準備手順: PostgreSQL 接続」の説明に従って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 サーバーからの継続的な WAL ストリーミングを追加できるようにすることにより、目標復旧時点（RPO）を削減できます。

Barman は `pg_receivewal` に依存しており、ネイティブのストリーミングレプリケーションプロトコルを活用し、PostgreSQL サーバー（マスターまたはスタンバイ）からトランザクションログを継続的に受信します。PostgreSQL 10 より前は、`pg_receivewal` は `pg_receivexlog` と命名されました。

重要: Barman では、同じサーバーに ``pg_receivewal`` がインストールされている必要があります。`pg_receivewal` には後方互換性があるため、利用可能な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する
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` コマンドの実行を通じ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` コマンドによってアーカイブされます。

`streaming_archiver_name` パラメーターで特に指定されない限り、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` システムビューでステータスを監視できるようにします。

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 WAL ファイルを受信するまで削除しないことを保証する自動化された方法です。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 manual](#) を参照してください。

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作成することもできます。このシナリオは、Docker ベースの PostgreSQL サーバーを構成したり、Windows で実行されている PostgreSQL サーバーを操作したりする場合にも役立ちます。

重要: 現時点では、Windows のサポート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。

11.2 WAL ストリーミングの構成方法

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、既に述べたように、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を使用する場合（推奨）、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不可欠なオプションは `slot_name` 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は必須です。

レプリケーションスロット名を構成する場合、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、次のように設定して、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 リカバリによる部分的な WAL ファイルの制限

`pg_receivewal` の標準的な動作は、WAL セグメント名の後に `.partial` サフィックスを付けて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、部分的な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ファイルを Barman の `archive-wal` コマンドに配信します。

マスター PostgreSQL サーバーで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）が 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 では、`get-wal` コマンドは、`--partial/-P` オプションを介して現在の `.partial` WAL ファイルの内容を返すことができます。これは、完全または特定の時点へのリカバリ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` なしで `recover` コマンドを実行した場合、Barman は `restore_command` リカバリーオプションで `barman-wal-restore` に `-P` オプションを自動的に追加します（その後、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 に出荷されたがまだアーカイブされていない場合に備えて、`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、WAL ファイルを Barman 着信ディレクトリにコピーするために PostgreSQL サーバーによって実行されるシェルコマンドである必要があります。

これは2つの方法で実行できます。どちらも SSH 接続が必要です。

- barman-wal-archive ユーティリティを介して（Barman 2.6 から）
- rsync / SSH 経由（Barman 2.6 より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、「スタンバイからの同時バックアップ」セクションをお読みください。

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ための**推奨される方法**は、EnterpriseDB パブリックリポジトリを介して配布され、GNU GPL 3 ライセンスで利用可能な 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。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 サーバーに出荷された WAL ファイルのデータが破損するリスクが軽減されます。rsync/SSH を archive_command WAL ファイルとして使用する場合、ファイルの内容が宛先のディスクにフラッシュされて 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、Barman の put-wal コマンド（2.6 で導入）とネイティブに通信する barman-wal-archive ユーティリティを開発しました。したがって、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減らします。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。

次のコマンドを使用して、`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 Barman で着信 WAL ファイルを受け入れるように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、`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、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ます。

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

Barman サーバーとの通信に SSH を使用するため、`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 のデフォルトの 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有効にし、`archive_command` が `barman-wal-archive` を使用するように設定します。

```
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有効にします。

```
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、`archive_command` プロセスに厳密なチェックを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ています。

```
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。このトリックは、サーバーが複製され、復元された PostgreSQL インスタンスから WAL ファイルを受信しないように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

継続的アーカイブがオンになっていて正常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確認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。

この目的のために、および 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 でアーカイブプロセスをトリガーします。Barman は 30 秒以内に 1 つのファイルが到着するのを待ちます（タイムアウトは `--archive-timeout` オプションで変更できます）。WAL ファイルを受信しない場合、エラーが返されます。

`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。

重要: Barman では、同じサーバーに `pg_basebackup` がインストールされている必要があります。`pg_basebackup` には下位互換性があるため、利用可能な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。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で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: `archiver` または `streaming_archiver` を介して WAL が 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できません

サーバー構成が有効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でバックアップ

rsync を介したバックアップは、2.0 より前の使用可能な唯一の方法であり、現在、増分バックアップ機能をサポートする唯一のバックアップ方法です。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を作成するには、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: *archiver* または *streaming_archiver* を介して WAL が 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できません

サーバー構成が有効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
```

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

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


第16章 クラウドスナップショットによるバックアップ

Barman は、ストレージボリュームのスナップショットを作成することにより、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。このように構成すると、PostgreSQL ファイルの物理バックアップはクラウドに保存されたボリュームスナップショットになり、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身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 [スナップショットの `barman-cloud-backup` コマンド](#) を直接使用して、Barman サーバーなしで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す必要があります。

- 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。
- PostgreSQL は、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構成する必要があります。
- `findmnt` コマンドが PostgreSQL ホストで使用できる必要があります。

!!!重要

PGDATA の外部に保存された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 のスナップショットの前提条件

Barman が使用する Python ディストリビューションで `google-cloud-compute` および `grpcio` ライブラリを利用する必要があります。これらのライブラリはオプションの依存関係であり、Barman パッケージでは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 grpcio google-cloud-compute
```

注釈: google-cloud-compute ライブラリに必要な 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。

Google Cloud Platform でのスナップショットバックアップには、次の追加の前提条件が適用されます。

- スナップショットバックアップに含まれるディスクはすべて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。
- Barman は、必要な権限セットを持つサービスアカウントを利用できる必要があります。これを行うには、そ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アタッチするか（推奨）、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します。

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 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`) のみがサポートされています。

次のパラメーターは、クラウドプロバイダーに関係なく設定する必要があります。

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

`snapshot_instance` が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ている場合、`snapshot_zone` はインスタンスが配置されている使用可能なゾーンであり、`snapshot_disks` はバックアップに含める必要があるディスクのコンマ区切りリストです。

重要: `snapshot_disks` に、PostgreSQL が必要とするデータを格納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、回復時に利用できません。

16.4 Google Cloud Platform のスナップショットの構成

GCP を使用する場合は、次の追加パラメーターを設定する必要があります。

```
snapshot_gcp_project = GCP_PROJECT_ID
```

これは、`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。

16.5 スナップショットバックアップの作成

構成オプションを設定し、Barman が適切な資格情報を利用できるようになったら、`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` コマンド中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詳細については、

`cloud snapshots section in feature details` を参照してください。

第17章 Windowsベースのサーバーのセットアップ方法

WAL アーカイブとバックアップの両方に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: この機能はまだ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` 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を取得する際に問題があり、PostgreSQL のロケールが英語でない場合、この問題の回避策は、英語でメッセージを出力するように PostgreSQL に指示することです。これを行うには、`postgresql.conf` ファイルに次のパラメーターを追加します。

```
lc_messages = English
```

```
lc_messages = English
```

これにより問題が解決されることが報告されています。

通常どおりサーバーをバックアップできます。

リモート回復は Windows サーバーでは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回復してから、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 サーバー間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 の実行を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を持ってい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 General commands

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

一般的なコマンドの範囲は Barman サーバー全体であり、多くの PostgreSQL サーバーをバックアップできます。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されたバックアップで機能します。

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、長時間実行されるデーモンまたはサービスファイルが含まれていません (`systemctl start`、`service start` などはありません)。代わりに、barman のバックグラウンド「定常状態」バックアップ操作を実行する `barman cron` サブコマンドが提供されています。

`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 スクリプトで実行する必要があります。 `barman cron` を毎分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、毎分実行される 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ます。これにより、次のサーバーにも保持ポリシーが適用されます。

- `retention_policy` は空でも有効でもありません。
- `retention_policy_mode` を `auto` に設定。

`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できることを確認してください。

注釈: *barman cron* はバックグラウンド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を担当しません。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 エントリでスケジュールする必要があります。

18.2 diagnose

diagnose コマンドは、診断とサポートの目的に役立つ 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。

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ースの一部とは見なされません。フォーマットは、異なる Barman バージョン間で変更できます。

18.3 list-servers

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 commands

前のセクションで述べたように、サーバーコマンドは PostgreSQL サーバーまたはその領域で動作し、ステータスの確認、メンテナンスオペレーションの実行、バックアップ、Barman 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` 一般コマンドから起動するように設計されています。

19.2 backup

`backup` コマンドは、指定されたサーバーの完全バックアップ (*base backup*) 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オプションがいくつかあります。詳細については、マニュアルページを参照してください。

次を使用して、特定のサーバーの完全バックアップを実行できます。

```
barman backup <server_name>
```

注釈: `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。

注釈: `barman backup <server_1_name> <server_2_name>` を使用して、`<server_1_name>` と `<server_2_name>` サーバーの両方を順次バックアップできます。

Barman 2.10 では、`backup` コマンドに `-w/--wait` オプションが導入されました。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ブされるのを待ってから、状態を `DONE` に設定し、バックアップ後の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ている場合、Barman は指定された秒数後に 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、欠落している WAL ファイルがアーカイブされていることを引き続き確認し、バックアップに `DONE` のラベルを付けます。

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>
```

注釈: `barman check all` を使用して、構成済みの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 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` が関与せず、`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、生成コマンドは中断されます。

コマンド例：

```
barman generate-manifest <server_name> <backup_id>
``` backup_id [backup idshortcuts]{#backup-id-shortcuts}のいずれかを使用できます。

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。

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

19.5 get-wal

Barman では、ユーザーが `get-wal` コマンドを使用して 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]() <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、特に指定がない限り、非圧縮のコンテンツが STDOUT に返されます。

get-wal コマンドでは、次のオプションを使用できます。

- -o により、ユーザーは Barman が要求された WAL ファイルを保存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始めて、アーカイブから WAL ファイルまでをのぞきます

リカバリオペレーション中に get-wal を使用して、Barman サーバーをサーバーの WAL ハブに変換できます。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で自動的に実現できます。

```
recovery_options = get-wal
```

recovery_options は、コンマ区切りの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。リカバリ操作中にキーワード 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 を使用して必要な WAL ファイルをフェッチします。同様に、実行時に recover コマンドに --get-wal オプションを使用できます。

get-wal が recovery_options に設定されているが、リカバリ操作中には必要ない場合、--no-get-wal オプションを recover コマンドで使用して、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り、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、例で sudo -u barman を使用している理由です。

リモートリカバリのために recovery_options を get-wal に設定すると、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、より回復力のある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と解凍や、PostgreSQL がそれらのいずれかを適用しているときに次の WAL ファイルを取得できるピーク機能など、多くの便利なオプションがあります。これは、PostgreSQL と Barman 間の帯域幅使用を最適化する優れた方法です。

barman-wal-restore は、barman-cli パッケージで利用できます。

これは、リモートリカバリの restore_command の例です。

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

Barman サーバーとの通信に SSH を使用するため、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 のデフォルトの 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 Barman で次のコマンドで WAL ファイルを送信するように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、`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に 2 つの引数が必要ですが、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ます。

```
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。

19.6 list-backups

次を使用して、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>
```

注釈: 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。

機械可読な出力を得るには、`--minimal` オプションを使用します。

19.7 rebuild-xlogdb

いつでも、特定のサーバー（または `all` ショートカットを使用してすべてのサーバー）の WAL アーカイブのコンテンツを再生成できます。WAL アーカイブは `xlog.db` ファイルに含まれており、Barman が管理す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>
```

19.8 receive-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。

19.8.1 receive-wal 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` プロセスが起動されます。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` オプションを指定してコマンドを実行すると、現在実行中の `receive-wal` プロセスが停止します。

`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は、`--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 以降の `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ピークが高い場合、`receive-wal` プロセスが遅れて同期しなくなる場合がありますことに注意してください。予防措置として、Barman では現在、誤った仮定を避けるために、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する必要があります。

19.8.2 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とするレプリケーションスロットを作成または削除する場合にも役立ちます。

`--create-slot` オプションを使用すると、`slot_name` 構成オプションにちなんで名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。

19.9 replication-status

`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、現在 PostgreSQL サーバーに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>
```

サーバー名として ``all`` を使用するすべてのサーバーの.. **NOTE You can request a full status report of the replica::**

- 。

機械可読な出力を得るには、`--minimal` オプションを使用します。

19.10 show-servers

次のコマンドで、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>
```

注釈: 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

19.11 status

`status` コマンドは、PostgreSQL サーバーまたはサーバー名として `all` を使用してい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>
```

19.12 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、受信して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切り替え以降、トランザクションアクティビティが発生していない場合は、`--force` オプションを使用して切り替え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。

19.13 verify

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

edb_notranlate_17 このコマンドは pg_verifybackup <path_to_backup_manifest> -n を呼び出します (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 idshortcuts のいずれかを使用できます。

第20章 Backup commands

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で直接機能するコマンドです。

注釈: `barman list-backups <server_name>` でバックアップ ID を取得できることに注意してください

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

これらのキーワードを Barman コマンドで使用すると、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。たとえば、次を発行できます。

```
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、`--name <friendly_name>` オプションを使用してバックアップを取得した場合、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、完全バックアップの一貫性に必要なすべての 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`` 操作の終了時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。

バックアップの最初から最後まで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引き続き確認し、バックアップに `DONE` のラベルを付けます。

バックアップの最後に必要な最初の WAL ファイルがない場合、そ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、特にスタンバイサーバーからバックアップする場合、WAL アーカイブ（ストリーミングまたは `archive_command` を介して）が適切に機能していることを確認することが重要です。

20.3 delete

次のコマンド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、バックアップを識別するために **バックアップ ID のショートカット** を受け入れます。

20.4 keep

サーバーの保持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、次のようにアーカイブバックアップにすることができます。

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

`TARGET` の可能な値は次のとおりです。

- `full` : バックアップでいつでも最新の状態に復元できます。これを達成するために、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が取られた時点の状態に回復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` オプションが指定されている場合、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、アーカイブとしてフラグが付けられていないバックアップの場合は `full` または `standalone` 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。

バックアップにアーカイブバックアップとしてフラグが付けられると、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と失敗します。
- 保持ポリシーはそのバックアップを `OBSOLETE` と見なさないため、`barman cron` はそのバックアップを削除しません。

- そのバックアップに必要な WAL は永久に保持されます。指定されたり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で元に戻すことができます。

警告: サーバーの保存方針で `standalone` アーカイブバックアップが必要なくなると、`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` 値の間の 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、最新のポイントインタイムへのリカバリに必要な WAL がまだ利用できます。

20.5 list-files

次を使用して、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で（またはログの末尾まで）のすべての 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 をアーカイブ手順に統合するのに非常に役立ちます。

20.6 recover

`recover` コマンドは、`backup` コマンドを使用してバックアップを実行した後、サーバー全体を回復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。その場合は、リカバリーを発行する前に停止することを忘れないでください。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。

リカバリーの実行の最後に、選択したバックアップがローカルにリカバリされ、宛先パスには PostgreSQL インスタンスの起動に使用できるデータディレクトリが含まれます。

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、`:ref:`list-backups`<list-backups>` 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ません (pg_tblspc 内のテーブルスペースを除く)。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備えて、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オプションがいくつかあります。

20.6.1 リモートリカバリー

リカバリーコマンドの呼び出しに `--remote-ssh-command <COMMAND>` オプションを追加します。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: リモートホストでリカバリを実行するには、`postgres` ユーザーを使用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。その場合は、リカバリーを発行する前に停止することを忘れないでください。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 `:ref:`get-wal`<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、公開キー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リカバリに必要な 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 表領域のリマッピング

Barman は、recover コマンドに `--tablespace` オプションを使用して、1 つ以上のテーブルスペースを自動的にリマップできます。このオプションは、`NAME:DIRECTORY` 形式を使用して、値のペアを引数として受け入れます。

- `NAME` は表領域の識別子
- `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 は作成しようとします（必要な権限があると仮定）。

20.6.3 ポイントインタイムリカバリ

Barman は PostgreSQL のポイントインタイムリカバリ (PITR) をラップしているため、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最後です。バックアップの開始から終了までの任意の時点を回復する場合は、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、一貫性に達したときにリカバリを終了できます。

リカバリターゲットは、次の相互に排他的な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) を指定するには PostgreSQL 10 以降が必要
- `--target-name TARGET_NAME`: `pg_create_restore_point(name)` 関数で以前に作成した名前付け復元ポイントを指定します
- `--target-immediate`: 一貫した状態に到達すると、リカバリが終了します（つまり、ベースバックアッププロセスが終了します）

重要: 時間、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にある必要があります。バックアップの開始から終了までの時点に回復する場合は、カタログ内の以前のバックアップから回復する必要があります。

`--exclusive` オプションで、リカバリ対象の直前で停止するか直後で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、回復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 アーカイブ内の最新のタイムラインに回復する場合) および `current` (バックアップが作成されたときのタイムラインに回復する場合) 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` タイムラインに回復し、12 未満の PostgreSQL バージョンは `current` タイムラインに回復します。「始める前に」セクションで説明されているように、タイムラインの詳細については、PostgreSQL のドキュメントをご覧ください。

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

- `shutdown`: 回復目標に到達したら、PostgreSQL をシャットダウンする
- `pause`: 復旧目標に到達すると、PostgreSQL を一時停止状態で起動し、ユーザーがインスタンスを検査できるようにします
- `promote`: リカバリターゲットに到達すると、PostgreSQL はリカバリを終了し、マスターに昇格します

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（後方互換性のため）。`--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](#) を参照してください。

Barman 2.4 は、`recover` コマンドの `--standby-mode` オプションも追加します。指定された場合、`standby.signal` ファイル（PostgreSQL 12 から）を作成するか、生成されたリカバリー構成に `standby_mode = on` を追加することにより、リカバリー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。スタンバイモードの詳細については、PostgreSQL のドキュメントをご覧ください。

20.6.4 Barman サーバーから WAL を取得する

`barman recover` コマンドは、リカバリ中に Barman から WAL をフェッチするように PostgreSQL を構成できます。これは、

`get-wal` section で説明されているように、`recovery_options`

グローバル/サーバー構成オプションを `'get-wal'` に設定することで有効になります。`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ーに必要な WAL をコピーします。

`--get-wal` および `--no-get-wal` オプションを使用して、`recovery_options` で定義された動作をオーバーライドできます。`--get-wal` と `barman recover` を使用して、Barman サーバーからの WAL のフェッチを有効にするか、`--no-get-wal` を使用して無効にします。

20.6.5 圧縮されたバックアップの回復

`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上のステージングディレクトリにコピーされます。
2. 圧縮ファイルがターゲットディレクトリに解凍されます。

3. Barman による特別な処理が必要な構成ファイルは、復元先からコピーされ、必要に応じて分析または編集され、Rsync を使用して復元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、ステージングディレクトリに適切な場所を選択するために `recovery_staging_path` オプションに依存しています。

`backup_compression` オプションを使用している場合は**する必要があります**したがって、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` オプションを使用します。これらのいずれも行わずに圧縮されたバックアップを復元しようとする、Barman は適切な場所を推測しようとせずに失敗します。

20.7 show-backup

次を使用して、特定のサーバーの特定のバックアップについて使用可能なすべての情報を取得できます。

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

`show-backup` コマンドは、バックアップを識別するために **バックアップ ID のショートカット** を受け入れます。

第21章 Features in detail

このセクションでは、いくつかの Barman 機能を紹介し、その適用性と使用に必要な構成について説明します。

Barman 構成で作業して多くのシナリオを作成できるため、このリストは完全ではありません。それでも、一般的なパターンについて説明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

Barman は **ファイルレベルのインクリメンタルバックアップ**を実装しています。増分バックアップは、特定の PostgreSQL サーバーのカタログで利用可能な最新の完全バックアップからのデータ変更のみを保存する完全な定期バックアップです。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。

注釈: ブロックレベルの増分バックアップは将来のバージョンで利用できるようになります。

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

Barman の増分バックアップの主な目標は次のとおりです。

- フルバックアッププロセスにかかる時間を短縮します
- 複数の定期的なバックアップが占めるディスク領域を削減する（**データ重複排除**）

この機能は `rsync` と `hard links` 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で前回のバックアップ以降に変更されていないファイルを共有し、関連するディスク使用量を節約することです。これは、VLDB コンテキストと、読み取り専用の履歴テーブルを高い割合で含むデータベースに特に当てはまります。

Barman 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増分バックアップを実装します。これは3つの値を受け入れます。

- `off` : 標準の完全バックアップ（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、変更のないファイルのハードリンクを作成することによる増分バックアップ（バックアップスペースと時間の削減）
- **copy** : サーバーの最後のバックアップを再利用し、変更のないファイルのコピーを作成することによる増分バックアップ（バックアップ時間の短縮のみ）

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増分バックアップが自動的に有効になります。

最後に、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つの値を受け入れます: `off`、`link`、および `copy`。たとえば、次のように1回限りの増分バックアップを実行できます。

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

21.1.2 帯域幅の使用を制限する

`bandwidth_limit` オプション（`global/per server`）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0、つまり無制限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。

複数のテーブルスペースがあり、1つ以上のテーブルスペースでバックアップ手順のI/Oワークロードを制限したい場合は、`tablespace_bandwidth_limit` オプション（`global/per server`）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、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。そうでない場合、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。

21.1.3 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減らすことができます。`network_compression` オプション (global/per server) 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有効になります (バックアップとリカバリの両方)。デフォルトでは `false` に設定されています。

21.1.4 バックアップ圧縮

Barman は、バックアッププロセス中にバックアップデータを圧縮するために `pg_basebackup` の圧縮機能を使用できます。これは、`backup_compression` 構成オプション (global/per server) 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み。

21.1.5 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次のとおりです。 `gzip` `lz4` および `zstd`。

```
backup_compression = gzip|lz4|zstd
```

Barman では、選択した圧縮アルゴリズムの CLI ユーティリティが Barman サーバーとの両方で使用できる必要があります。CLI ユーティリティを使用して、圧縮された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します。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インストールできます。

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

注釈: `zstd` バージョンは 1.4.4 以上である必要があります。Debian 10 (buster)、Ubuntu 18.04 (bionic) および SLES 12 の ``zstd`` の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は、`recovery_staging_path` も設定して、`barman recover` が圧縮されたバックアップを回復できるようにする必要があります。 [圧縮されたバックアップの回復](#) を参照

詳細については、セクションをご覧ください。

21.1.6 コンプレッションワーカー

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を行うと、圧縮速度が向上します（デフォルトは0）。

```
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。

21.1.7 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数値に設定する必要があります。

21.1.8 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（PostgreSQL はバックアップを圧縮）またはクライアント（`pg_basebackup` はバックアップを圧縮）で圧縮を指定できます。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 以降で実行し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ことを犠牲にして、バックアップに必要なネットワーク帯域幅を削減できます。

`backup_compression_location` が `server` に設定されている場合、追加のオプション `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。

21.1.9 圧縮形式

```
backup_compression_format = plain|tar
```

`backup_compression_format` が設定されていないか、値が `tar` の場合、バックアップは圧縮された tar-ball としてディスクに書き込まれます。`plain` と `tar` の両方のフォーマットの説明は、`[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` に応じて Postgres サーバーと 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。

<code>backup_compression</code>	<code>backup_compression_format</code>	Postgres server	Barman server
gzip	plain	tar	None
gzip	tar	tar	tar
lz4	plain	tar, lz4	None
lz4	tar	tar, lz4	tar, lz4
zstd	plain	tar, zstd	None
zstd	tar	tar, zstd	tar, zstd

21.1.10 同時バックアップ

通常、バックアップ操作中、Barman は `_concurrent_backup` に PostgreSQL ネイティブ関数 `pg_start_backup` および `pg_stop_backup` を使用します。[ABOUT_CONCURRENT_BACKUP] これは、PostgreSQL 9.6 以降で推奨されるバックアップ方法です（ただし、PostgreSQL 15 ベータでは関数の名前が `pg_backup_start` および `pg_backup_stop` に変更されました）。

[ABOUT_CONCURRENT_BACKUP] 同時バックアップは、_ストリーミングレプリケーションプロトコル_を使用するテクノロジーです（たとえば、`pg_basebackup` などのツールを使用します）。

推奨されるバックアップアプローチであるだけでなく、同時バックアップでは、Barman を使用した次のアーキテクチャシナリオも使用できます。`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 より古い 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 はサーバーの同時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 Postgres サーバーを指す必要があります
- `conninfo` は、宛先 Postgres データベース上のデータベースを指す必要があります。

重要: 同時バックアップの場合、現在 Barman は完全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の反対。完全バックアップは、その Barman ファイルを受信してアーカイブするまで一貫性があ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が導入されています。これは、`check-backup` コマンド（`cron` コマンドで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。

21.1.11 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` 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` 使用時)

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、WAL 切り替え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るようにします。

!!!注

プライマリで書き込みトラフィックがほとんどまたはまったくないときにスタンバイをバックアップする場合は、`primary_conninfo` を設定することが特に重要です。`primary_conninfo` が設定されていない場合でも、バックアップは実行されますが、プライマリ上の現在の WAL エージェ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、バックアップの停止段階で待機します。

Barman は現在、WAL ファイルとバックアップデータが同じ PostgreSQL サーバーからのものであ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した場合、バックアップと WAL は引き続き有効ですが、バックアップの作成と 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 WAL ストリーミングと有効になっている 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。これは、PostgreSQL 10 以前で生成された WAL は論理的には同等であるがバイナリレベルで異なる可能性があるため、Barman は 2 つの WAL が同一であることを検出できないためです。

21.1.12 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るため、バックアップが遅延する可能性があります。

このデフォルトの動作は、`immediate_checkpoint` 構成のグローバル/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してず、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使用して `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。

21.1.13 ローカルバックアップ

注釈: 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、この機能は実稼働での使用はお勧めしません。EnterpriseDB の一部のお客様は、特定の状況で、最も重要なことに、会社が提供する 24 時間年中無休の運用サービスの下で使用するローカルバックアップのサポートを追加することを Barman しています。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、ロギングおよび `cron` 構成の観点から `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 を 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し、バックアップデータを PGDATA、および必要に応じてテーブルスペースとは別のボリュームに保存できます。通常、これらのボリュームは NFS などのファイルシステムを備えたネットワークストレージアブライアンスにあります。

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もたらす 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、レプリケーションと監視の目的で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ードナッシングアーキテクチャを引き続きお勧めします。

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（通常は `postgres`）で実行されることです。コミュニティパッケージはデフォルトで `barman` ユーザーに Barman をインストールするため、このユースケースでは次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` を `local-rsync` に設定する必要があります。この機能は、代わりに SSH に依存してリモートで動作する `rsync` 同等の機能と本質的に同じです。`local-rsync` では、ローカルで `rsync` コマンドを発行してファイルシステムのコピーが実行されます（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 ### WAL 圧縮

`barman cron` コマンドは、構成ファイルで `compression` オプションが設定されている場合、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 圧縮の場合 (`bzip2` ユーティリティが必要)
- `gzip` : Gzip 圧縮用 (`gzip` ユーティリティが必要)
- `pybzip2` : Bzip2 圧縮の場合 (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)
- `pigz` : Pigz 圧縮用 (`pigz` ユーティリティが必要)
- `custom` : カスタム圧縮の場合、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ための 16 進文字列
- 注: `pybzip2` および `pygzip` を除くすべてのメソッドでは、新しいプロセスをフォークするために `barman archive-wal` が必要です。

21.2.1 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減らすこともできます。

このようなシナリオを構成するには、[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 Barman サーバーを構成する必要があります、`receive-wal` プロセスは PostgreSQL サーバーの同期スタンバイとして認識される必要があります。

まず、`show-servers` コマンドで 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: これは構成の一例に過ぎません Barman が同期スタンバイノードとして使用できることを示します。Barman のみを使用することは推奨していません。このトピックの詳細については、PostgreSQL のドキュメントから [Synchronous Replication](#) をお読みください。

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 `receive_wal` プロセスを同期ストリーミングクライアントとして表示します。

```
[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 ### 最小限の冗長性の安全性

デフォルトでは 0 に設定されている `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。

この値を 0 より大きい任意の数に設定することにより、Barman は、サーバーカタログに常に少なくともその数のバックアップがあることを確認します。

これにより、偶発的な `barman delete` 操作から保護されます。

重要: アイテム保持ポリシー設定が最小冗長要件と競合しないことを確認してください。このトピックに関するメッセージについては、Barman のログを定期的に確認してください。

21.3.1 保持ポリシー

Barman は、バックアップの **保持ポリシー** をサポートしています。

バックアップ保持ポリシーは、復旧手順のためにバックアップと関連するアーカイブログ（ログ先行書き込みセグメント）を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的なバックアップと、そ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された WAL ファイルを保持します。

Barman ユーザーは、**バックアップの冗長性**（定期的なバックアップの数）または **リカバリウィンドウ**（どれくらいの期間）に関して保持ポリシーを定義できます。

冗長性に基づく保持ポリシー

：冗長性ベースの保持ポリシーでは、ユーザーは保持する定期的なバックアップの数を決定します。冗長性ベースの保持ポリシーは、リカバリ期間を使用する保持ポリシーとは対照的です。

リカバリウィンドウに基づく保持ポリシー

：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 が期間を指定すると、Barman はリカバリウィンドウ中の任意の時点へのポイントインタイムリカバリ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 ファイルの保持を保証します。間隔は常に現在の時刻で終わり、ユーザーが指定した日数だけさかのぼります。たとえば、保持ポリシーが 7 日間の復旧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、前の金曜日。

21.3.2 スコープ

保持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用** : `wal_retention_policy` 構成オプションを介して

重要: 時間次元では、定期的なバックアップの時間枠にアーカイブログを含める必要があります。

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の 2 つの一般的なユースケースがあります。

フルポイントインタイムリカバリのシナリオ：

：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、使用可能な最初のバックアップからいつでも復旧できます。

部分的なポイントインタイムリカバリのシナリオ：

：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がっています。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 週間のログをアーカイブできます（最後の 4 回の定期的な週次バックアップ）。

重要: 現在、Barman は `wal\_retention\_policy` オプションを `main` に制限することにより、**フルポイントインタイムリカバリ**シナリオのみを実装しています。

21.3.3 それらの仕組み

Barman のリテンションポリシーは次のとおりです。

- **自動化** : `barman cron` によって強制
- **手動** : Barman は古いバックアップを報告し、削除できます

重要: 現在、Barman は手動による強制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ります。

21.3.4 構成と構文

保持ポリシーは、次の構成オプションを使用して定義できます。

- `retention_policy` : ベースバックアップ保持用
- `wal_retention_policy` : アーカイブログ保存用
- `retention_policy_mode` : `auto` にのみ設定できます（保持ポリシーは `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発揮します。

`retention_policy` の構文

`retention_policy` を介したベースバックアップ保持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せん
- `value` は整数で > 0

- **冗長性保持ポリシー** の場合： - value はサーバーの最小冗長性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）逆順の時系列で
- **リカバリウィンドウポリシー** の場合： - リカバリ可能性のポイントは次のとおりです。逆順の時系列におけるその値は、サーバーの最小冗長性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ておらず、警告が生成された場合）

デフォルトでは、`retention_policy` は空です（保持は強制されません）。

`wal_retention_policy` の構文

現在、`wal_retention_policy` に許可されている値は特別な値 `main` です。これは、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 ファイルを削除する前と後

Barman が管理できるフックスクリプトは 2 種類あります。

- 標準フックスクリプト
- フックスクリプトを再試行

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のフックスクリプトを 1 回のみ実行しますが、リトライフックスクリプトはリターンコードに応じて複数回実行される場合があります。

具体的には、リトライフックスクリプトを実行すると、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）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する一時的な障害として扱います。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` を返すことで、ユーザーは Barman にメインオペレーションをエラーで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. リトライ「pre」フックスクリプト（存在する場合）
3. リトライ「pre」フックスクリプトが `ABORT_STOP` で中止されなかった場合の実際のイベント（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）

4. リトライ「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` はリトライ「post」フックスクリプトで無視されます。これらの場合、追加の警告をログに記録することを除いて、`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を使用して構成できます（サーバーごとにオーバーライドできます）。

- `pre_backup_script` : ベースバックアップの前に 1 回だけ実行され、終了コードをチェックしないフックスクリプト
- `pre_backup_retry_script` : ベースバックアップの前に実行されるフックスクリプトを、成功するか中止するまで繰り返し
- `post_backup_retry_script` : ベースバックアップ後に実行されるフックスクリプトを、成功するか中止するまで繰り返し
- `post_backup_script` : ベースバックアップ後に 1 回だけ実行され、終了コードをチェックしないフックスクリプト

スクリプト定義はシェルに渡され、終了コードを返すことができます。再試行スクリプトの場合にのみ、Barman はリターンコードを確認します（hook script section を参照）。

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（post フェーズのみ）
 - `BARMAN_PHASE` : スクリプトのフェーズ、pre または post
 - `BARMAN_PREVIOUS_ID` : 以前のバックアップの ID（存在する場合）
 - `BARMAN_RETRY` : リトライスクリプトなら 1、そうでなければ 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、バックアップの前後の削除スクリプトが導入されています。

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、サーバーごとにオーバーライドできます。

- **pre_delete_script** : バックアップを削除する前に一度だけ、終了コードをチェックせずにフックスクリプトが起動されました
- **pre_delete_retry_script** : バックアップの削除前に実行されるフックスクリプトを、成功するか中止するまで繰り返し
- **post_delete_retry_script** : バックアップの削除後に実行されるフックスクリプトを、成功または中止するまで繰り返し
- **post_delete_script** : バックアップの削除後に一度だけ、終了コードをチェックせずに起動されたフックスクリプト

スクリプトはシェルを介して実行され、終了コードを返すことができます。再試行スクリプトの場合にのみ、Barman はリターンコードを確認します（上のセクションを参照）。

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：

- **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で構成できます（サーバーごとにオーバーライドできます）。

- **pre_archive_script** : メンテナンス（通常は `barman cron`）によって WAL ファイルがアーカイブされる前に 1 回だけ実行され、終了コードをチェックしないフックスクリプト
- **pre_archive_retry_script** : メンテナンス（通常 `barman cron`）によって WAL ファイルがアーカイブされる前に、成功するか中止されるまで繰り返し実行されるフックスクリプト
- **post_archive_retry_script** : メンテナンスによって WAL ファイルがアーカイブされた後、成功するか中止されるまで繰り返し実行されるフックスクリプト
- **post_archive_script** : メンテナンスで WAL ファイルがアーカイブされた後に一度だけ実行されるフックスクリプト、終了コードのチェックなし

スクリプトはシェルを介して実行され、終了コードを返すことができます。再試行スクリプトの場合にのみ、Barman はリターンコードを確認します（上のセクションを参照）。

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

- **BARMAN_CONFIGURATION** : Barman が使用する構成ファイル
- **BARMAN_ERROR** : エラーメッセージ（`post` フェーズのみ）
- **BARMAN_PHASE** : スクリプトのフェーズ、`pre` または `post`

- `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 の前後の削除スクリプトが導入されています。

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、サーバーごとにオーバーライドできます。

- `pre_wal_delete_script` : WAL ファイル削除前に実行されるフックスクリプト
- `pre_wal_delete_retry_script` : WAL ファイルの削除の前に実行されるフックスクリプトを、成功するか中止されるまで繰り返し
- `post_wal_delete_retry_script` : WAL ファイルの削除後に実行されるフックスクリプト、成功するか中止されるまで繰り返し
- `post_wal_delete_script` : WAL ファイル削除後に実行されるフックスクリプト

スクリプトはシェルを介して実行され、終了コードを返すことができます。再試行スクリプトの場合にのみ、Barman はリターンコードを確認します（上のセクションを参照）。

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 リカバリースクリプト

バージョン 2.4 では、回復前および回復後のスクリプトが導入されています。

以前のスクリプトと同様に、回復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できます。

- `pre_recovery_script` : バックアップの回復前に起動されたフックスクリプト、終了コードをチェックせずに 1 回のみ
- `pre_recovery_retry_script` : バックアップの復元の前に実行されるフックスクリプトを、成功するか中止するまで繰り返し
- `post_recovery_retry_script` : バックアップの回復後に実行されるフックスクリプトを、成功するか中止するまで繰り返し

- `post_recovery_script` : バックアップの回復後に一度だけ、終了コードをチェックせずに起動されたフックスクリプト

スクリプトはシェルを介して実行され、終了コードを返すことができます。再試行スクリプトの場合にのみ、Barman はリターンコードを確認します（上のセクションを参照）。

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：

- `BARMAN_DESTINATION_DIRECTORY` : 新しいインスタンスが回復されたディレクトリ
- `BARMAN_TABLESPACES` : テーブルスペースリロケーションマップ (JSON、存在する場合)
- `BARMAN_REMOTE_COMMAND` : リカバリで使用するセキュアシェルコマンド (存在する場合)
- `BARMAN_RECOVER_OPTIONS` : リカバリ追加オプション (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使用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の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（下位互換性のため）、`barman_lock_directory` は `barman_home` に設定されます。

注釈: ユーザーは、実行時変数データ（たとえば、`/var/run/barman`）専用の揮発性パーティション内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 以降、Barman では、グローバル/サーバーオプション `path_prefix` を使用して、Barman が実行可能ファイルを探す 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 クラスター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 SHIPPING を使用）および (<https://www.repmgr.org/>) [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 ハブとしても使用できます。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。

21.7 パラレルジョブ

デフォルトでは、Barman はバックアップと回復の両方の操作中にファイルのコピーに 1 つのワーカーのみを使用します。バージョン 2.2 以降、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ます。この場合、コピーされるファイルはすべての並列ワーカーに均等に分散されます。

グローバルおよびサーバースコープで構成でき、対応する構成ファイルにこれらを追加します。

```
parallel_jobs = n
```

ここで、`n` は、ファイルコピー操作で使用する並列ワーカーの数です。デフォルト値は 1 です。

いずれの場合も、ユーザーは `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 つの並列ワーカーを使用できます。

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

または、代わりに：

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

この並列ジョブ機能は、`rsync` /SSH を介して構成されたサーバーでのみ使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。

21.8 地理的な冗長性

バックアップサーバーのソースが PostgreSQL サーバーではなく Barman インストールである 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できます。

この機能により、ユーザーは PostgreSQL バックアップの地理的に分散したコピーを透過的に保持できます。

Barman の専門用語では、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** と定義されています。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（ダイレクトサーバーとパッシブ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 情報を同期

`barman sync-info` コマンドは、同期の目的で役立つ 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。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>]]
```

このコマンドは、以下を含む JSON オブジェクトを返します。

- 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のマップ
- 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
- 最後に読み取った位置（xlog データベースファイル内）
- 最後に読み取った WAL ファイルの名前

JSON 応答には、**master** と **passive** ノード間の同期に必要な情報がすべて含まれています。

`--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識別します。

`-c/--config` オプションで `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` を呼び出す場合は、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。

- 標準保守作業から除外されます
- `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、透過的に以下を呼び出す `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できます。

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

`sync-backup barman` を起動すると、`primary_ssh_command` 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は、`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、次の方法でも同期できます。

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

21.9 クラウドスナップショットのバックアップ

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作成できます。

- Barman サーバーが WAL とバックアップメタデータの保存に使用されている場合、`required configuration operations for snapshots` を使用した `barman backup`。

- Barman サーバーがなく、代わりに WAL とバックアップメタデータにクラウドオブジェクトストアが使用されている場合、必要な command line arguments を使用した `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作成するための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、 [PostgreSQL backup API](#) 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、スナップショットがアプリケーションの一貫性を保証するために必要な状態に達するまで待つてから、次のディスクに進みます。
4. 各ディスクのデバイス名など、追加のプロバイダー固有のデータが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 API 呼び出しは、バックアップ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）のいずれかです。

次のプリフライトチェックは、各バックアップの前に実行されます。また、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` が実行されます。

- `snapshot_zone` で指定されたアベイラビリティゾーンに `snapshot_instance` で指定されたコンピュータインスタンスが存在すること。
- `snapshot_zone` で指定されたアベイラビリティゾーンに `snapshot_disks` で指定されたディスクが存在すること。
- `snapshot_disks` で指定されたディスクが `snapshot_instance` に接続されています。
- `snapshot_disks` で指定されたディスクが `snapshot_instance` にマウントされている。

21.9.2 スナップショットバックアップからのリカバリ

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、これは、Terraform などの専用の Infrastructure-as-Code 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` コマンドを使用して、スナップショット復旧インスタンスを検証したり、安全でないオプションの PostgreSQL 構成の確認などの復旧後のタスクを実行したり、必要な PITR オプションを設定したりできます。また、`backup_label` ファイルを所定の場所にコピーし（バックアップラベルはボ

リユーmsナップショットに保存されないため)、必要な WAL 間でコピーします (--get-wal 回復オプションが使用されている場合、フェッチするように 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、barman recover の代わりに、より制限された **スナップショットの barman-cloud-restore** 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. バックアップメタデータに従って、手順 1 でプロビジョニングした各ディスクがアタッチおよびマウントされた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 **スナップショットの barman-cloud-restore** コマンドを使用して、リカバリを検証して終了します。

手順 1 と 2 は、既存のコードとしてのインフラストラクチャシステムで処理するのが最適ですが、これらの手順を手動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。このようなスクリプトの例は **provided with Barman** ですが、このスクリプトは実行される環境についてさまざまな仮定を行うため、実稼働での使用には適していません。

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されてマウントされたら、次の追加の引数を使用して barman recover を実行します。

- --remote-ssh-command : リカバリインスタンスへのログインに必要な ssh コマンド。
- --snapshot-recovery-instance : クラウドプロバイダーが必要とするリカバリインスタンスの名前。
- --snapshot-recovery-zone : リカバリインスタンスが配置されているアベイラビリティゾーンの名前。

例：

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

スナップショットバックアップのリカバリ時には、次の barman recover 引数/構成変数を使用できないことに注意してください。

Command argument	Config variable .
--bwlmit	bandwidth_limit
--jobs	parallel_jobs
--recovery-staging-path	recovery_staging_path
--tablespace	N/A

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、接続されているディスクがそのバックアップのスナップショットから複製されたことを確認します。Barman は、バックアップラベルと WAL を所定の場所にコピーし、PostgreSQL 構成で必要なリカバリオプションを設定することにより、PostgreSQL をリカバリ用に準備します。

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コードとしてのインフラストラクチャ、アドホックな自動化、または手動のいずれかでプロビジョニングされたかにかかわらず、Barman にクエリを実行して、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、バックアップ内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

Snapshot information:
  provider          : gcp
  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ときに使用できます。

```
$ 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"
      },

```

(次のページに続く)

(前のページからの続き)

```
"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",
  }
}
]
}
...
```

`barman-cloud-backup` で取得したバックアップには、`barman-cloud-backup-list` と共に使用してクラウドオブジェクトストア内のバックアップメタデータを照会できる類似の `barman-cloud-backup-show` コマンドがあります。

第22章 Barman client utilities (barman-cli)

以前は別のオープンソースプロジェクトでしたが、`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 サーバーと一緒にインストールするための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」セクションで説明されているように `archive_command` として使用され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およびリカバリサーバーで `restore_command` リカバリ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。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をお勧めします。

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。

パッケージを RedHat/CentOS システムにインストールするには、`root` として次のように入力します。

```
yum install barman-cli
```

Debian/Ubuntu では、`root` ユーザとして次のように入力します。

```
apt-get install barman-cli
```


第23章 Barman client utilities for the Cloud (barman-cli-cloud)

Barman クライアントユーティリティが拡張され、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、オブジェクトストレージの統合をサポートし、PostgreSQL データベースのディザスターリカバリー機能を強化しています。

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 互換のオブジェクトストア）
- Azure Blob ストレージ
- 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ており、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** として使用され、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送信するアーカイブスクリプト。または、WAL アーカイブ用のフックスクリプトとして (**pre_archive_retry_script**) ;
- **barman-cloud-wal-restore** : **restore_command** として使用して、Barman サーバーをバイパスしてクラウドストレージから WAL ファイルを取得し、PostgreSQL スタンバイに直接保存するスクリプト。
- **barman-cloud-backup** : PostgreSQL サーバーで直接ローカルバックアップを作成し、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送信するために使用されるバックアップスクリプト。または、バーテンダー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** で作成された 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内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付けるために使用されるスクリプト-そ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** で取られた 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** : **barman-cloud-backup** でクラウドストレージから直接取得したバックアップを復元するために使用されるスクリプト;

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 : (<https://github.com/boto/boto3>) [boto3]
- Azure Blob Storage : (<https://docs.microsoft.com/en-us/python/api/azure-storage-blob/?view=azure-python>) [azure-storage-blob] および (オプションで) (<https://docs.microsoft.com/en-us/python/api/azure-identity/?view=azure-python>) [azure-identity]
- Google Cloud Storage : (<https://cloud.google.com/storage/docs/reference/libraries>) [google-cloud-storage]
- boto3<1.18.0
- azure-storage-blob<12.10.0 および azure-identity<1.8.0
- google-cloud-storage<2.0.0

aws-s3 クラウドプロバイダーの資格情報を設定する方法については、

[Credentials](#) 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](#)

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。--credential オプションを使用して、Azure Active Directory を介して認証するために、azure-cli または managed-identity の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は、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。

パッケージを RedHat/CentOS システムにインストールするには、root として次のように入力します。

```
yum install barman-cli-cloud
```

Debian/Ubuntu では、root ユーザ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

上記の説明に従って、Barman サーバーに `barman-cli-cloud` パッケージをインストールします。

以下を PostgreSQL サーバーの Barman 構成に追加して、`barman-cloud-backup` をポストバックアップスクリプトとして構成します。

```
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は失敗します。このため、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、`-w` / `--wait` オプションを使用してバックアップを実行することをお勧めします。

以下を PostgreSQL サーバーの Barman 構成に追加して、`barman-cloud-wal-archive` をプレ WAL アーカイブスクリプトとして構成します。

```
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` [デフォルト]: AWS S3 または S3 互換の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 プロバイダー別の特異性

23.4.1 Google Cloud Storage

23.4.2 設定

`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は作成されます。バケットの作成にはデフォルトのオプションが使用されます。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）が必要な場合は、必要なものすべてに一致するようにバケットを作成して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。

`postgres` (`archive_command` または `restore_command`) から `barman cloud` を実行している場合は、`postgres` 環境ファイルに `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 `barman cloud` コマンドに固有です。

- `Google Cloud Storage--cloud-provider=google-cloud-storage` を選択
- `SOURCE_URL` は `gs` と `https` の両方のフォーマットをサポートします。例：

```
gs://BUCKET_NAME/path
or
https://console.cloud.google.com/storage/browser/BUCKET_NAME/path
```

23.4.4 使用法

一部の詳細は、すべての `barman cloud` コマンドに固有です。

- `Google Cloud Storage--cloud-provider=google-cloud-storage` を選択
- `SOURCE_URL` は `gs` と `https` の両方のフォーマットをサポートします。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を使用して、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 を使用すると、バックアップデータはクラウドプロバイダーによってボリュームスナップショットとして保存され、backup_label を含む WAL と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 であり、barman-cloud によって使用される資格情報は、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ります。

23.5.1 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作成するには、次の追加の引数を指定して barman-cloud-backup を使用します。

- `--snapshot-disk` (複数のディスクで複数回使用できます)
- `--snapshot-instance`
- `--snapshot-zone`

`--cloud-provider` が `google-cloud-storage` の場合、次の引数も必要です。

- `--snapshot-gcp-project`

クラウド スナップショットが要求された場合、次のオプションは `barman-cloud-backup` で使用できません。

- `--bzip2`、`--gzip` または `--snappy`
- `--jobs`

バックアップを作成したら、`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。

23.5.2 スナップショットの barman-cloud-restore

barman-cloud を使用してスナップショットバックアップから回復するプロセスは、

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のメタデータは、barman-cloud-backup-show を使用して照会できます。

オブジェクトが保存されたバックアップで `barman-cloud-restore` を使用する場合と同様に、このコマンドは PostgreSQL を回復用に準備しないことに注意してください。PostgreSQL の起動前に必要な 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、次の追加の引数を `barman-cloud-restore` と共に使用する必要があります。

- `--snapshot-recovery-instance`
- `--snapshot-recovery-zone`

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 Troubleshooting

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` コマンドの出力は `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 クライアントのバージョン、およびすべてのサーバーの現在の構成とステータスを提供します。

`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な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 助けを求める

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もたくさんあります。

Barman と Barman を使用したディザスターリカバリーのシナリオに関する質問については、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してください。

<https://groups.google.com/group/pgbarman>

または freenode の IRC チャンネル：<irc://irc.freenode.net/barman>

バグを発見した場合は、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、24 時間年中 Barman のサービスを含む専門家のサポートを提供します。

24.2.1 バグの送信

Barman は広範にテストされ、現在いくつかの実稼働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 にはバグがありません。

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`) で取得した出力を添付して、GitHub の問題トラッカーを介してバグを報告します

警告: セキュリティの観点から危険な情報を開示する可能性があるため、診断コマンドの出力を送信するときは注意してください。

第25章 The Barman project

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メンテナンス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。オープンソースとして一般公開される Barman と PostgreSQL の新機能の開発を後援できます。

詳細については、次の Web サイトをご覧ください。

- [Barman website](#)
- [Support section](#)
- [EnterpriseDB website](#)
- [Barman FAQs](#)
- [2ndQuadrant blog: Barman](#)

25.2 Barman への貢献

EnterpriseDB には、バーマンのコードを改善するため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あります。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ており、クロスファンクショナルなコラボレーションを通じ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。Barman のすべてのコミットは、生産パイプラインのさまざまな段階にある複数の個人の貢献です。

これは Barman を開発するための推奨される方法ですが、次の場合に限り、外部の開発者からのパッチを喜んで受け入れます。

- ユーザドキュメント（チュートリアルとマニュアルページ）が提供されています。
- 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していません。
- ソースコードは現在のマスターブランチに基づいています。
- コミットとプルリクエストは単一の機能に制限されています（マルチ機能の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、コントリビューションには、著作権の割り当てと、デベロッパーの雇用主からの委託作品の所有権の免責事項を提供する必要があります。

この目的には、GitHub のプルリクエストシステムを使用できます。

25.3 著者

アルファベット順：

- アビジット・メノンセン

*ディディエ・ミシエル

- マイケル・ウォレス

過去の貢献者（アルファベット順）：

- Anna Bellandi（QA /テスト）
- ブリット・コール（ドキュメントレビュアー）
- カルロ・アスカニ（開発者）

- Francesco Canovai（QA /テスト）

• ガブリエレ・バルトリニ（建築家）

- Gianni Ciolli（QA /テスト）

- Giulio Calacoci（開発者）

*ジュゼッペ・ブロッコロ（開発者）

*ジェーン・スリーフト（開発者）

- Jonathan Battiato（QA /テスト）
- レオナルド・チェッキ（開発者）
- Marco Nenciarini（プロジェクトリーダー）
- 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](#) : Holger Hamann (MIT ライセンス) によって書かれた Barman の Nagios プラグイン
- [puppet-barman](#) : Puppet (GPL) の Barman モジュール
- [Tutorial on "How To Back Up, Restore, and Migrate PostgreSQL Databases with Barman on CentOS 7"](#)、Sadequl Hussain (DigitalOcean コミュニティで入手可能)
- [BarmanAPI](#) : Barman の RESTFul API、Mehmet Emin Karakas (GPL)

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 の下で配布されています。

© Copyright EnterpriseDB UK Limited 2011-2023

Barman は、欧州委員会の第 7 フレームワークプログラムによって資金提供された研究プロジェクト [4CaaSt](#) を通じ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 は、コントリビューションが著作権の譲渡と、開発者の雇用者からの委託仕事の所有権の請求の免責事項を提供することを要求しています。これにより、すべての Barman ディストリビューションが無料の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 にお問い合わせください。

第26章 Reference links

[rpo] : https://en.wikipedia.org/wiki/Recovery_point_objective

[rto] : https://en.wikipedia.org/wiki/Recovery_time_objective

[repmgr] : <https://www.repmgr.org/>

[sqldump] : <https://www.postgresql.org/docs/current/static/backup-dump.html>

[physicalbackup] : <https://www.postgresql.org/docs/current/static/backup-file.html>

[pitr] : <https://www.postgresql.org/docs/current/static/continuous-archiving.html>

[adminbook] : <https://www.2ndquadrant.com/en/books/postgresql-10-administration-cookbook/>

[wal] : <https://www.postgresql.org/docs/current/static/wal.html>

[49340627f9821e447f135455d942f7d5e96cae6d] : <https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=49340627f9821e447f1352455>

[requirements_recovery] : <https://www.postgresql.org/docs/current/static/warm-standby.html#STREAMING-REPLICATION-SLOTS>

[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>

[roles] : <https://www.postgresql.org/docs/current/static/role-attributes.html>

[レプリケーションスロット] : <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>

[ターゲット] : <https://www.postgresql.org/docs/current/static/runtime-config-wal.html#RUNTIME-CONFIG-WAL-RECOVERY-TARGET>

[2ndqrpmrepo] : <https://rpm.2ndquadrant.com/>

[2ndqdebrepo] : <https://apt.2ndquadrant.com/>

[boto3] : <https://github.com/boto/boto3>

[oto3creds] : <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-script] : 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

[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/>