



Barman User Guide

リリース **3.17.0**

EnterpriseDB UK Limited

2026 年 02 月 02 日

目次

第 1 章	コンセプト	3
1.1	はじめに	3
1.2	一般的なバックアップの概念	4
1.3	Postgres バックアップの概念と用語	7
1.4	Barman の概念と用語	9
第 2 章	インストールする	13
2.1	システム要件	13
2.2	RHEL ベースのディストリビューション	14
2.3	Debian ベースのディストリビューション	15
2.4	SLES ベースのディストリビューション	16
2.5	追加要件	17
第 3 章	クイックスタート	19
3.1	最初のサーバーの構成	19
3.2	最初のバックアップの取得	24
3.3	バックアップの復元	24
第 4 章	barman を使用したアーキテクチャバックアップデザイン	27
4.1	Barman をインストールする場所	27
4.2	1 人の Barman、多数の Postgres サーバー	29
4.3	バックアップ戦略	29
4.4	WAL アーカイブ戦略	29
4.5	バックアップの 2 つの一般的なシナリオ	30
4.6	地理的な冗長性	34
4.7	クラウド スナップショット バックアップ	35
第 5 章	前提条件	37
5.1	Postgres ユーザー	37
5.2	Postgres 接続	38
5.3	Postgres クライアントツール	38
5.4	Postgres ストリーミングレプリケーション接続	39
5.5	SSH 接続	40
5.6	archive_command を介した WAL アーカイブ	41
第 6 章	Barman チェック	45
6.1	使用法	45
6.2	出力を理解する	46
第 7 章	バックアップ	49
7.1	概要	49

7.2	バックアップの要件	50
7.3	インクリメンタルバックアップ	50
7.4	帯域幅使用量の管理	50
7.5	ネットワーク圧縮	51
7.6	バックアップ圧縮	51
7.7	バックアップの暗号化	53
7.8	即時チェックポイント	54
7.9	ストリーミングバックアップ	55
7.10	SSH を介した Rsync を使用したバックアップ	56
7.11	ローカルで Rsync を使用したバックアップ	57
7.12	スタンバイのコンカレント バックアップ	58
7.13	外部構成ファイルの管理	59
7.14	バックアップに不変ストレージを使用する	59
7.15	クラウド スナップショット バックアップ	61
第 8 章	WAL アーカイブ	71
8.1	WAL 圧縮	71
8.2	WAL 暗号化	72
8.3	同期 WAL ストリーミング	73
第 9 章	カタログ情報	75
9.1	目的	75
9.2	使用法	76
第 10 章	リカバリー	79
10.1	ローカルリカバリー	80
10.2	リモートリカバリー	80
10.3	デルタリカバリー	80
10.4	テーブルスペースのリマッピング	81
10.5	ポイント インタイム リカバリー	81
10.6	Barman から WAL の取得	82
10.7	暗号化されたバックアップの復元	84
10.8	圧縮バックアップの復元	86
10.9	ブロックレベルのインクリメンタルバックアップのリカバリー	86
10.10	マルチフォーマットバックアップのリカバリパイプライン	88
10.11	.partial WAL ファイルの制限	89
10.12	外部構成ファイルの管理	89
10.13	スナップショットバックアップからの復旧	89
第 11 章	リテンション ポリシー	93
11.1	概要	93
11.2	リテンションポリシーの主要コンポーネント	93
11.3	リテンションポリシーの主な目的	94
11.4	最小限の冗長性の安全性	94
11.5	リテンションポリシーの範囲	95
11.6	ユースケース	96
11.7	保持ポリシーが適用される方法	96
11.8	構成と構文	96

11.9	ブロックレベルのインクリメンタルバックアップのリテンションポリシー	97
11.10	クラウドバックアップのリテンションポリシー	97
第 12 章	診断と障害対応	99
12.1	Barman ステータス	99
12.2	Barman ショーサーバー	100
12.3	Barman 複製ステータス	100
12.4	Barman 診断	100
第 13 章	構成リファレンス	103
13.1	使用法	103
13.2	オプション	104
13.3	構成モデル	127
第 14 章	コマンドリファレンス	129
14.1	barman	129
14.2	ショートカット	131
14.3	終了ステータス	131
14.4	サブコマンド	131
14.5	barman-cli コマンド	158
第 15 章	地理的な冗長性	163
15.1	同期情報	163
15.2	構成	163
15.3	ノードの同期	164
15.4	手動同期	164
第 16 章	フックスクリプト	165
16.1	バックアップを作成する前と後	166
16.2	バックアップを削除する前と後	167
16.3	WAL がアーカイブされる前と後	167
16.4	WAL ファイルが削除される前と後	168
16.5	バックアップを復元する前と後	168
16.6	barman でフックとして barman-cloud-* スクリプトを使用する	168
第 17 章	クラウドの Barman	171
17.1	Barman クラウドクライアントパッケージ	172
17.2	インストール	173
17.3	オブジェクトロックのサポート	173
17.4	コマンドリファレンス	174
第 18 章	用語集	201
第 19 章	リリースノート	203
19.1	Barman release notes	203
第 20 章	よくある質問	259
20.1	一般	259
20.2	バックアップ	260
20.3	インストールと構成	262

20.4 リカバリー	262
20.5 要件	263
第 21 章 ライセンス	265
索引	267

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

Barman is distributed under GNU GPL 3 and maintained by [EnterpriseDB](#), a platinum sponsor of the [PostgreSQL](#) project.

第1章 コンセプト

ディザスタリカバリー計画の作成は、特にバックアップ管理に関係するさまざまな概念に慣れていない人にとって、難しい場合があります。バックアップにはさまざまな方法があり、それぞれに独自の長所、短所、および技術的要件があります。適切なアプローチの選択は、リソース、環境、および技術的知識によって異なります。誰もがこのコンテキストで十分な基礎を持っているわけではないことを承知で、このセクションでは、特に Postgres と Barman のコンテキストで、データベースのバックアップに関する最も基本的な概念を説明することに専念しています。

バックアップの概念、Postgres の論理的および物理的バックアップに既に慣れている場合は、*Barman concepts and terminology* セクションに進んでください。

1.1 はじめに

完璧な世界では、バックアップは必要ありません。ただし、特に重要なビジネス環境では、予期せぬことが発生した場合に備えることが重要です。データベースシナリオでは、*/"unexpected/"* は次のいずれかの形式を取ることができます。

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

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

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

- 目標復旧ポイント RPO 重大なインシデントにより IT サービスからデータが失われる可能性のある最大目標期間。
- 目標復旧時間 RTO 災害または中断後に、ビジネス継続性の中断に関連する受け入れがたい結果を回避するために、ビジネスプロセスを復元する必要がある目標期間とサービスレベル。

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

当然のことながら、たとえそれが祖母のレシピウェブサイトであったとしても、私たち全員が RPO=0 データ損失ゼロと RTO=0 ダウンタイムゼロ、理想郷を望んでいます。実際には、ビジネス継続性の要件を決定するには、注意深いコスト分析フェーズが必要です。

幸いなことに、Barman と Postgres で構成されるオープンソーススタックを使用すると、同期ストリーミングレプリケーションのおかげで RPO=0 を達成できます。RTO は、Patroni や 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 をこれらのツールのいずれかと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。

いずれにせよ、実際のツールではなく、ディザスタリカバリーに関連する文化的な側面をより強調することが重要です。人間のいない道具は役に立ちません。Barman との私たちの使命は、次のようなディザスタリカバリーの文化を促進することです。

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

さらに、災害が発生した場合に自分自身とチームを準備するためにできる限りのことを行ってください。なぜなら、それが発生した場合

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

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

準備をしてください、怖がらないでください。

1.2 一般的なバックアップの概念

各データベースシステムには独自の用語がある場合がありますが、すべてのリレーショナルデータベースで一貫した基本的なバックアップ原則があります。このセクションでは、バックアップがどのように動作するかを理解するために必要な中核概念の概要を提供します。

1.2.1 物理的および論理的バックアップ

リレーショナルデータベースのコンテキストでは、論理バックアップは、実行時に、バックアップが取られたときの正確な状態でデータを再作成する一連の操作に他なりません。簡単に言うと、データベースの構造とデータを再構成する一連の SQL ステートメントです。この方法は、データベースバージョンやシステムアーキテクチャなどの特定の環境仕様に依存しないため、互換性のないシステム間でデータを移行する場合に適した選択です。また、特定のデータベースオブジェクトまたはテーブルの復元を許可することにより、柔軟性が向上します。

ただし、論理バックアップには時間がかかる場合があります。データベースのサイズによっては、数時間または数日かかる場合があります。また、バックアップはバックアッププロセスの開始時のデータベースの状態を反映するため、バックアップ中に行われた変更はキャプチャされず、データ損失の潜在的なウィンドウが発生します。その結果、この方法は、通常、小さくて複雑でないデータベースに推奨されます。Postgres では、論理バックアップは `pg_dump` および `pg_dumpall` ユーティリティを介して実装されます。

一方、物理バックアップは、ファイルシステムからデータベースファイルを直接コピーすることにより機能します。したがって、このメソッドは通常、環境仕様に関連付けられます。物理的バックアップには、`cp` のような基本的な Unix ツールを使用したものから、Barman のようなバックアップマネージャーを使用したより洗練されたソリューションに至るまで、さまざまなアプローチがあります。バックアップ管理ツールは、ファイルがデータベースの一貫した状態を表すことを保証し、サーバーを通常に実行したままにすることは、手動で行うと困難であるため、物理的バックアップにおいて重要な役割を果たします。

物理的バックアップは、データベースを再作成するために操作を完全に再生する必要がないため、バックアッププロセス中だけでなく、特にリカバリー中にも論理バックアップよりもはるかに高速です。また、インクリメンタルバックアップの可能性が有効になり、時間とストレージの使用量が大幅に削減され、より頻繁なバックアップが可能になります。最後に、このアプローチの最大の利点の 1 つは、ポイントインタイムリカバリー PITR を実行できることです。これにより、データベースを現在時刻とバックアップ時刻の間の特定の時点に復元できます。この機能は、トランザクションログが物理的バックアップとともにアーカイブされる場合にのみ可能です。

お気づきのとおり、物理バックアップはより堅牢ですが、より複雑です。このため、Postgres の Barman のような補助バックアップ管理ツールは、ディザスターリカバリー計画でこのプロセスが効果的かつ確実に処理されるようにする上で重要な役割を果たします。

1.2.2 バックアップの種類

物理バックアップに関しては、基本的に 3 つの異なるタイプに分類できます。フル、インクリメンタル、差分。

フルバックアップはベースバックアップとも呼ばれ、特定の時点ですべてのデータをキャプチャし、基本的にデータベース全体の完全なスナップショットを作成します。このタイプのバックアップには、システムをバックアップが取られたときの正確な状態に復元するために必要なすべての情報が含まれています。この意味で、一貫した完全な物理バックアップからのリカバリーは、本質的に完全であるため、可能な限り最速です。

一方、インクリメンタルバックアップは、以前のバックアップ以降に発生した変更のみをキャプチャするように設計されています。以前のバックアップは、完全バックアップまたは別のインクリメンタルバックアップのいずれかです。インクリメンタルバックアップにより、時間とストレージの使用量が大幅に削減され、より頻繁なバックアップが可能になり、その結果、データ損失のリスクが削減されます。通常、インクリメンタルバックアップは、ファイルや内部データブロックなどの低レベルのデータ構造に依存して、

実際に何が変更されたかを判断するため、物理的バックアップでのみ可能です。インクリメンタルバックアップからのリカバリーには、ベースから最新のインクリメンタルまでのすべてのバックアップのチェーンが必要です。

最後に、差分バックアップは、以前のバックアップ以降に行われた変更のみをキャプチャするという点でインクリメンタルバックアップに似ています。ただし、主な違いは、差分バックアップは常に完全バックアップを基準として変更を記録し、インクリメンタルまたは別の差分バックアップとは決して相対しないことです。実際、すべての差分バックアップは増分バックアップですが、すべての増分バックアップが差分バックアップであるわけではありません。この場合のリカバリーには、最新の差分バックアップとそれに関連するベースバックアップのみが必要です。

1.2.3 トランザクションログ

トランザクションログは、ほとんどのリレーショナルデータベースの基本的な部分です。これは、データベース内のすべての操作を実行前に記録する一連の連続したファイルで構成されています。その結果、彼らは一定期間にデータベースで発生したすべての変更を所有します。これは、主に、クラッシュ前にディスクにフラッシュされていない操作を再生できることにより、データベースがクラッシュから効果的に回復できることを保証し、データの損失を防ぎます。これは、データベースレプリケーションの多くの実装のキーコンポーネントでもあります。

トランザクションログは、すべての操作が永続化された後、リサイクルされます。ただし、これらのログを事前にアーカイブできる場合は、基本的にデータベースに行われたすべての変更の完全な記録を保持し、いつでも再生できます。トランザクションログのアーカイブとともにベースバックアップがあることにより、継続的なバックアップが可能になります。これは、完全なバックアップを定期的に取得することができない大規模なデータベースで特に役立ちます。差分バックアップと同様の目標を達成しますが、ポイントインタイムリカバリーなどのより堅牢な機能も有効にするため、さらに多くの機能を備えていることに気づくでしょう。

1.2.4 ポイントタイム リカバリー

ポイントインタイムリカバリーを使用すると、ベースバックアップの終了時間からアーカイブされたトランザクションログでカバーされる最も遠いポイントまでの特定の瞬間にデータベースを復元できます。トランザクションログの継続的なアーカイブを維持することにより、現時点までにデータベースに加えられたすべての変更を再生できます。これは、ベースバックアップに基づいてすべてのトランザクションログを再生することにより実行され、たとえば、目的のタイムスタンプまたはトランザクション ID に基づいて、任意の時点で再生を停止する機能も提供します。PITR では、秒未満の精度が可能で、通常毎日実行される標準の完全バックアップと比べて、大きな利点があります。

この機能は、誤った削除や変更など、人的エラーまたは意図しない変更が発生した状況で特に役立ちます。PITR は、データベースを不要なイベントの直前の正確な状態に復元することにより、RPO を大幅に削減します。強力な安全装置を提供し、データベースを以前の完全バックアップに戻したり、その後のすべての正当な変更を失うリスクを負うことなく、重要なデータを迅速かつ正確にリカバリーできます。

ただし、PITR がすべての問題の解決策であることを意味するわけではありません。トランザクションログの再生は、ベースバックアップからの距離によっては、依然として時間がかかる場合があります。したがって、最適なソリューションは、実際にはすべての戦略の組み合わせです。完全バックアップと頻繁なインクリメンタルバックアップ、およびトランザクションログのアーカイブ。このように、最新の状態への復元は、最新のバックアップを復元し、その後に後続のトランザクションログを再生することになりま

す。同様に、特定のポイントへの復元は、ターゲットポイントに最も近い以前のバックアップを復元し、その後目的のターゲットまでの後続のトランザクションログを再生することです。

1.3 Postgres バックアップの概念と用語

このセクションでは、Postgres、その実装、および特定の特性のコンテキストでバックアップの概念を説明します。コンテンツは主に [Backup and Restore section from the Postgres official documentation](#) に基づいていますので、Postgres がバックアップを処理する方法についてのより詳細な説明が必要な場合は、それを読むことを強くお勧めします。

1.3.1 pg_dump vs pg_basebackup

Postgres でバックアップを取るための基本的に 2 つの主要ツール `pg_dump` および `pg_basebackup`。それらの違いは、基本的に、論理バックアップと物理バックアップの違いです。つまり、`pg_dump` `pg_dumpall` を含むは論理バックアップを取得し、`pg_basebackup` は物理的バックアップを取得します。

注釈

Barman は、論理バックアップで動作しないため、`pg_dump` または `pg_dumpall` をまったく利用しません。`pg_basebackup` は、構成されたバックアップ方法に応じて Barman によって使用されます。

`pg_basebackup` は、基本的に、`pg_basebackup` を使用して、Postgres クラスターから宛先ディレクトリにテーブルスペースがある場合を含むすべてのファイルをコピーします。クラスター全体をバックアップすることのみができ、特定のデータベースまたはオブジェクトをバックアップすることはできません。`pg_basebackup` は、データベースサーバーをバックアップモードにおよび解除し、一貫性のために必要なすべてのトランザクションログがベースバックアップとともに保存されていることを確認します。そのため、`pg_dump` とは異なり、`pg_basebackup` で取得されたバックアップには、バックアップの進行中に発生した変更も含まれます。これは、頻繁に高負荷が発生するデータベースにとって大きな利点です。`pg_basebackup` の詳細については、`pg_basebackup` で読むことができます。

注釈

実際には、Postgres の物理バックアップは、少なくともバックアッププロセス中に生成されたトランザクションログ Postgres の WAL もある場合にのみ完全/自己完結型です。それ以外の場合、バックアップ自分自身は、新しい Postgres インスタンスを復元して起動するには不十分です。

`pg_basebackup` を使用して、`pg_basebackup` と同様の結果を達成することもできます。これは、Postgres で物理的バックアップを取るさらに別の方法です。低レベル API は、代替コピーツールを使用して物理的バックアップを手動で取得する場合に使用されます。このシナリオでは、データベースサーバーを手動でバックアップモードにおよび終了し、一貫性のために必要なすべてのトランザクションログが正しくアーカイブされていることを確認する必要があります。

注釈

Barman は、構成されたバックアップ方法に応じて Postgres の低レベル API を使用します `backup_method = rsync`。

1.3.2 先行書き込みログ

ログの先行書き込み WAL は、Postgres およびその他のデータベースがトランザクションログを参照する方法です。Postgres では、各 WAL ファイルは 16 MB 相当の変更構成可能をサポートします。WAL ファイルは次々にシークエンシャルに書き込まれ、チェックポイントが実行されるまで同時に維持されます。Postgres のチェックポイントは、すべての変更をディスクに永続化して、後で WAL をリサイクルできるようにする行為です。チェックポイントは通常 5 分ごと、または 1GB の WAL ファイルが生成された後に発生し、両方のオプションを構成可能です。WAL は、クラッシュリカバリー、データベースレプリケーション、PITR に役立つだけでなく、良好なパフォーマンスを保証するための重要なコンポーネントでもあります。そうしないと、各トランザクションがコミットした後に変更をディスクに同期する必要があり、巨大な I/O が発生します。WAL では、データベースの一貫性を保証するために十分であるため、変更をチェックポイント時に延期できます。

1.3.3 WAL アーカイブと WAL ストリーミング

トランザクションログのアーカイブは、Postgres では"継続的アーカイブ"または"WAL アーカイブ"として知られています。WAL アーカイブとは、本質的に、WAL ファイルがリサイクルされる前に他の場所に保存できることを意味します。Postgres では、これを行う伝統的な方法は、サーバー構成の `archive_command` パラメーターを使用することです。

`archive_command` は、任意のシェルコマンドを値として受け入れ、完全に入力されると WAL ファイルごとに実行されます。このようなコマンドは、各ファイルが目的の場所に安全にコピーされることを確認します。これにより、Postgres がこれらのファイルを保存する方法または場所について仮定を行わないという意味で、非常に高い柔軟性を提供し、必要なコマンドまたはライブラリを使用できます。このコマンドは、成功を示すゼロの終了ステータスを返す必要があります。それ以外の場合、Postgres はアーカイブが失敗したことを理解し、正常にアーカイブされるまでこれらのファイルをリサイクルしません。これは安全性の確保には役立ちますが、再び機能するか、ディスク領域が不足するまで WAL ファイルは蓄積され続けるため、コマンドが何らかの理由で失敗し始めると悪夢になる可能性があります。`archive_command` の詳細については、こちらをご覧ください。

WAL をアーカイブする別の方法は、ストリーミングレプリケーションプロトコルを使用して WAL ファイルを目的の場所に転送するために使用されるネイティブ Postgres ユーティリティである `pg_receivewal` を使用することです。一般に WAL ストリーミングとして知られるこの方法の大きな利点は、従来の `archive_command` 方法と比較して、ファイルがリアルタイムで転送されることです。つまり、転送を開始するために WAL セグメントが完全に満たされるまで待機する必要がないことですこれにより、データ損失の可能性が大幅に減少します。

`archive_command` とは異なり、デフォルトでは、この方法だけでは、WAL ファイルがリサイクルされる前に正常にアーカイブされることは保証されません。これは、WAL ファイルをアーカイブする前にリサイクルでき、基本的にログが永久に失われることを意味します。このため、このシナリオでは、レプリケーションスロットの使用を強くお勧めします。レプリケーションスロットは、主にデータベースレプリケーションのコンテキストで使用され、プライマリサーバーが後続のレプリカが必要とする WAL ファイルを正常に受信するまで保持し、レプリカがオフラインになるか切断された場合の安全性を高めます。`pg_receivewal`

とともに使用すると、同じ目標を達成します。つまり、レシーバーに正常に転送されるまで、WAL ファイルがリサイクルされないようにします。

1.3.4 リカバリー

Postgres の復旧プロセスは、バックアップの種類によって異なります。論理バックアップの場合、このプロセスは、バックアップファイルの形式に応じて、`pg_restore` を実行するか、バックアップファイルからすべての SQL コマンドを実行するだけで簡単です。ただし、物理的バックアップの場合、プロセスは少し複雑です。

物理バックアップから正常にリカバリーするには、クラスターファイルとその WAL アーカイブの両方が必要です。少なくとも、バックアッププロセス中に生成された WAL ファイルが必要です。バックアップが `pg_basebackup` で取得された場合、特に指定しない限り、必要な WAL ファイルは既に出力ディレクトリに含まれています。ただし、Postgres 低レベル API を使用して手動で取得した場合、リカバリ中に必要なすべての WAL ファイルが利用できることを確認するのは自分の責任です。

リカバリーの準備をするには、いくつかの手順に従う必要があります。これには、`:term: PITR`, among others. For a detailed explanation of this process, refer to the [Postgres official documentation](#) を実行する場合、WAL アーカイブとターゲットポイントから WAL ファイルを取得するコマンドなど、バックアップクラスターディレクトリの構成ファイルでのいくつかのパラメーターの指定が含まれます。すべてが正しければ、バックアップから新しいインスタンスを起動できます。Postgres は、必要な WAL がすべて適用されていることを確認します。

リカバリーに Postgres のインクリメンタルバックアップが含まれる場合、最初に `pg_combinebackup` を使用してすべてのバックアップを結合する必要があります。合成フルバックアップが生成され、標準のフルバックアップと同じ方法でリカバリーに使用できます。

1.4 Barman の概念と用語

このセクションでは、Barman の重要な概念の概要を提供し、以前のセクションで説明した概念の一部を Barman が利用する方法を示します。

1.4.1 サーバー

Barman は、複数のデータベースサーバーのバックアップを同時に管理できます。このため、バックアップサーバーの論理的分離が必要になります。Barman では、バックアップサーバーまたは単にサーバーは、特定のデータベースサーバーのバックアップコンテキストを表します。Barman がデータベースインスタンスと対話する方法、バックアップが管理される方法、その保持ポリシーなどを定義します。各サーバーには、すべてのバックアップと WAL ファイルが保存される独自の専用ディレクトリと、ほとんどの Barman で指定する必要がある一意の名前があります。コマンドを使用して、どのコンテキストで実行するかを指定します。

1.4.2 バックアップ方法

Postgres backup concepts and terminology で概要を示したように、Postgres サーバーをバックアップする方法は複数あります。Barman のコンテキストでは、これらはバックアップ方法と呼ばれます。Barman は、さまざまなバックアップ方法をサポートしており、それぞれが異なる Postgres 機能に依存し、独自の要件、

長所、短所があります。サーバーの構成ファイルの `backup_method` パラメーターを使用して、目的のバックアップ方法を指定できます。

注釈

Barman サーバーを管理するときは、単一のバックアップ方法を使用することを強くお勧めします。バックアップ方法を変更する必要がある場合は、新しい Barman サーバーをセットアップすることをお勧めします。

1.4.3 Rsync バックアップ

`backup_method = rsync` で取得されたバックアップ。このバックアップ方法を使用する場合、Barman は Postgres 低レベル API と Rsync を使用して、SSH 接続を介してクラスターファイルを手動で転送します。Rsync は、2つの場所間でファイルとディレクトリを同期できる強力なコピーツールです。同じホスト上またはネットワーク上の別のホスト上。Barman は、低レベル API を使用してサーバーをバックアップモードにおよびオフにし、同時に Rsync を使用してすべての関連ファイルを Barman のサーバーの指定されたディレクトリにコピーします。このプロセスの最後に、Barman はデータベースサーバーで WAL スイッチを強制して、必要なすべての WAL ファイルがアーカイブされるようにします。最後に、完全性チェックが実行されて、バックアップが一貫していることを確認します。

1.4.4 ストリーミングバックアップ

`backup_method = postgres` で取得されたバックアップ。このバックアップ方法を使用する場合、Barman はデータベースサーバーをバックアップするために `pg_basebackup` を呼び出します。Barman は、すべてのテーブルスペースを Barman 上のサーバーの専用ディレクトリにマップします。このプロセスの最後に、Barman はデータベースサーバーで WAL スイッチを強制して、必要なすべての WAL ファイルがアーカイブされるようにします。最後に、完全性チェックが実行されて、バックアップが一貫していることを確認します。

1.4.5 スナップショットバックアップ

スナップショットバックアップは、`backup_method = snapshot` を設定するか、Barman のクラウド CLI ツールを直接使用して実行できます。これらのバックアップは、Barman をデータベースサーバーが存在するクラウドプロバイダーと統合することにより機能します。データベースのストレージボリュームのスナップショットが物理的バックアップとして取得されます。このセットアップでは、Barman はクラウドでバックアップを管理し、主に WAL ファイル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。

1.4.6 ファイルレベルのインクリメンタル バックアップ

rsync バックアップを使用する場合、ファイルレベルのインクリメンタル バックアップが可能です。これは、ファイルシステムのハードリンクに依存する重複排除の Rsync ネイティブ機能を使用します。ファイルレベルのインクリメンタル バックアップを実行する場合、Barman は最初に使用可能な最新のサーバーバックアップへのハードリンクを作成し、基本的に追加のディスク領域を消費せずにそのコンテンツを別のディレクトリにレプリケートします。次に、Rsync を使用してその内容を Postgres クラスターのコンテンツと同期し、変更されたファイルのみをコピーします。ハードリンクを使用せずにファイルレベルのイ

ンクリメンタルバックアップを持つこともできます。この場合、Barman は最初に以前のバックアップの内容を新しいバックアップディレクトリにコピーします、基本的にそれを複製して追加のディスク領域を消費しますが、変更されたファイルのみをコピーしますデータベースサーバー。

1.4.7 ブロックレベルのインクリメンタル バックアップ

streaming backups を使用する場合、ブロックレベルのインクリメンタルバックアップが可能です。Postgres 17 で導入された、インクリメンタルバックアップのネイティブ `pg_basebackup` 機能を活用します。この機能には、ネイティブのインクリメンタルバックアップ用に適切に構成されたバージョン 17 以降の Postgres インスタンスが必要です。ブロックレベルのインクリメンタルバックアップでは、有効な `backup_manifest` ファイルを含むバックアップを重複排除のリファレンスとして使用できます。重複排除はブロックレベル Postgres のページで発生するため、ブロックレベルのインクリメンタルバックアップはファイルレベルのインクリメンタルバックアップよりも効率的です。

1.4.8 `archive_command` を介した WAL アーカイブ

これは、WAL ファイルを Barman に転送する 2 つの方法の 1 つです。一般的に *rsync* バックアップとともに使用されるこのアプローチには、Postgres の `archive_command` パラメーターを構成して、WAL ファイルを Barman のサーバーの専用ディレクトリに直接アーカイブすることが含まれます。コマンドは、Barman ホストでサーバーの WAL ディレクトリを手動で指定する Rsync コマンド、またはサーバー名のみを必要とし、Barman が残りを処理する `barman-wal-archive` ユーティリティのいずれかです。さらに、`barman-wal-archive` は、ファイルを受信するとすぐに `fsync` を実行することにより、安全性を向上させます。

1.4.9 WAL ストリーミング

これは、WAL ファイルを Barman に転送する 2 つの方法の 1 つです。一般的に *streaming backups* とともに使用されるこのアプローチは、`pg_receivewal` ユーティリティに依存して WAL ファイルを転送します。データベースサーバーで手動構成が必要ないため、構成が非常に簡単です。*streaming backups* で述べたように、WAL ストリーミングを使用する場合、レプリケーションスロットをお勧めします。バックアップサーバー構成で `create_slot` から `auto` を設定することにより、事前に手動でスロットを作成するか、Barman にスロットを作成させることができます。

1.4.10 フックスクリプト

Barman を使用すると、開発者は操作前および/または後として特定の操作に沿ってフックスクリプトを実行できます。この機能は、開発者に、調整された多様な動作を実装する柔軟性を提供します。`rsync` を使用する場合、バックアップ後スクリプトを利用してバックアップのマニフェストを生成できます。または、バックアップ後 *hook scripts with barman-cloud* コマンドを組み合わせることで、バックアップをクラウドストレージにコピーできるハイブリッド分散アーキテクチャを作成できます。

このトピックのより詳細な説明については、*hook scripts* のメインセクションを参照してください。

1.4.11 リストアとリカバリー

Barman では、リカバリーは、新しい場所に必要なすべての WAL ファイルとともにバックアップを復元し、リカバリーのために Postgres インスタンスを効果的に準備するプロセスです。

リカバリー で概要を示したように、Postgres の復旧プロセスは、ベースディレクトリの準備からサーバー自分自身の起動までのいくつかの手順で構成されています。Barman は、バックアップを復旧する準備に必要なすべての手順を実行できます。これは、Barman の用語で `"restore/"` と呼ばれます。この場合、復旧の完了は、通常、サーバーを起動するだけで済みますので、Postgres は必要な WAL を適用してライブ開始できます。

第2章 インストールする

Barman 公式パッケージは、`PGDG`によって提供されています。これらのパッケージは、オペレーティングシステムに付属する Python 3 のデフォルトバージョンを使用します。

Barman 機能スイートを構成する 3 つのパッケージ `barman`、`barman-cli` および `barman-cli-cloud`。

- `barman` はメインパッケージであり、インストールする必要があります。
- `barman-cli` は、`barman-wal-restore` および `barman-wal-archive` ユーティリティを保持するオプションパッケージです。このパッケージは、これらのユーティリティを `archive_command` または `restore_command` として使用する予定がある場合に必須です。Barman クラスターの一部である各 Postgres サーバーにインストールする必要があります。
- `barman-cli-cloud` は、クラウドプロバイダーでバックアップを管理するために使用できる `barman-cloud-*` クライアントスクリプトを保持するオプションパッケージです。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 Postgres サーバーにインストールする必要があります。

注釈

Barman パッケージは、いくつかの異なるリポジトリで見つけることができます。互換性、安定性、最新の更新へのアクセスが保証されるため、PGDG リポジトリの使用をお勧めします。

警告

別のリポジトリを使用して Barman をアップグレードしないでください。そうすることで、各ソースリポジトリが異なる構成レイアウトを使用する異なるパッケージを提供するため、構成が失われるリスクがあります。

2.1 システム要件

Barman サーバーを実行するために必要な最小システム要件は次のとおりです。

- Linux オペレーティングシステム Debian、Ubuntu、RHEL、Rocky、Fedora などまたは UNIX 系オペレーティングシステム FreeBSD、OpenBSD など
- Python 3.8 以降
- Python モジュール
 - `psycopg2` `>= 2.4.2` Postgres サーバーに接続するために必要です

- python-dateutil
- setuptools
- argcomplete オプション
- PostgreSQL クライアントツール Postgres サーバーと対話するために必要です
- PostgreSQL サーバー >= 13
- rsync >= 3.1.0 リカバリーと Rsync バックアップに必要
- boto3 >= 1.29.1 AWS のスナップショットロック機能と `backup_method = snapshot` を併用する場合に必要
- file POSIX コマンド、通常は file パッケージで提供されます

バージョン 3.14 で非推奨: Python のバージョン 3.6 および 3.7 のサポートは非推奨になりました。Barman 3.14 は Python 3.6 では動作しないことが知られています。Python 3.7 で動作する可能性がありますが、3.8 より前のバージョンの Python ではテストされておらず、サポートされていません。

注釈

RedHat Enterprise Linux、RockyLinux、および AlmaLinux のユーザーは、`Extra Packages Enterprise Linux EPEL` リポジトリ<<https://fedoraproject.org/wiki/EPEL>>をインストールする必要があります。

2.2 RHEL ベースのディストリビューション

RHEL システムおよび AlmaLinux、Oracle Linux、Rocky Linux などの同様の RHEL ベースのシステムに:term:`RPM`パッケージを使用して、`barman`、`barman-cli` および `barman-cli-cloud` をインストールできます。

インストールを始めるには、最初に [PGDG RPM repository](#) をインストールします。

重要

`barman-cli-cloud` スクリプトは、PGDG からの RHEL ベースのディストリビューションの `barman-cli` パッケージの一部です。したがって、クラウドスクリプトを使用するには、`barman-cli` をインストールするだけで済みます。

2.2.1 バートンダー

`barman` パッケージをインストールするには `barman` として実行します。

```
dnf install barman
```

2.2.2 barman-cli

barman-cli パッケージをインストールするには、Postgres サーバーで barman-cli として実行します。

```
dnf install barman-cli
```

注釈

barman-cloud ユーティリティを *hook scripts* として使用する場合、Barman サーバーに barman-cli パッケージをインストールする必要があります。

2.3 Debian ベースのディストリビューション

Debian システムおよび Ubuntu のような Debian ベースのシステムでは、DEB パッケージを使用して ``barman`、barman-cli、および barman-cli-cloud をインストールできます。

インストールをはじめに、PGDG APT リポジトリをインストールします。これはシステムによって異なります。

- Debian の場合 [PGDG Debian repository](#)。
- Ubuntu の場合 [PGDG Ubuntu repository](#)。

重要

barman-cli-cloud パッケージは、barman-cli のインストール時に推奨パッケージに含まれます。

インストールを開始する前に、ユースケースを評価することが重要です。barman-cloud-backup などの barman-cloud クライアントスクリプトを使用する予定がない場合は、barman-cli のインストール時に推奨パッケージとして barman-cli-cloud のインストールをスキップできます。ただし、barman-cloud クライアントスクリプトのみを使用する場合は、barman-cli-cloud パッケージを自分自身でインストールできます。

2.3.1 バートンダー

barman パッケージをインストールするには barman として実行します。

```
apt install barman
```

2.3.2 barman-cli

barman-cli パッケージをインストールするには、Postgres サーバーで barman-cli として実行します。

```
apt install barman-cli
```

2.3.3 barman-cli-cloud

barman-cli-cloud パッケージをインストールするには、Postgres サーバーで barman-cli-cloud として実行します。

```
apt install barman-cli-cloud
```

注釈

barman-cloud ユーティリティを *hook scripts* として使用する場合、このパッケージを Barman サーバーにインストールする必要があります。

2.4 SLES ベースのディストリビューション

barman は、`barman` にインストールできます。

インストールを開始するには、[PGDG SLES Repository Configuration](#) で入手可能な詳細な手順に従って、適切なリポジトリを追加する必要があります。

The current supported version for installation is SLES 15 SP6.

重要

barman-cli-cloud ユーティリティは、PGDG からの SLES ベースのディストリビューションの `barman-cli` パッケージの一部です。したがって、クラウドスクリプトを使用するには、`barman-cli` をインストールするだけで済みます。

2.4.1 バーテンダー

barman パッケージをインストールするには `barman` として実行します。

```
zypper install barman
```

2.4.2 barman-cli

barman-cli パッケージをインストールするには、Postgres サーバーで barman-cli として実行します。

```
zypper install barman-cli
```

注釈

barman-cloud ユーティリティを *hook scripts* として使用する場合、このパッケージを Barman サーバーにインストールする必要があります。

2.5 追加要件

クラウドストレージ統合やデータ圧縮などの一部の Barman 機能は、オプションの Python ライブラリに依存しています。これらの依存関係は、システムのパッケージマネージャーを介して Barman をインストールする場合、デフォルトでは含まれません。

barman-cloud、スナップショットバックアップ、または特定の圧縮アルゴリズムなどの機能を有効にするには、以下の表にリストされている適切な追加依存関係をインストールする必要があります。これらの依存関係が、Barman が使用するのと同じ Python インタープリターにインストールされていることを確認します。

Optional Libraries

機能/プロバイダー	必要なライブラリ
AWS S3	boto3
AWS EC2 スナップショット	boto3
Azure Blob Storage	azure-identity、azure-storage-blob
Azure VM のスナップショット	azure-identity、azure-storage-blob、azure-mgmt-compute
Google Cloud Storage	google-cloud-storage
GCP コンピューティングスナップショット	grpcio、google-cloud-storage、google-cloud-compute
スナッピー圧縮	python-snappy、cramjam >= 2.7.0
Zstandard 圧縮	zstandard
LZ4 圧縮	lz4
オートコンプリート	argcomplete

Tip

barman-cloud の使用時に `ModuleNotFoundError` が発生した場合は、次のことを確認します。

1. Barman が使用するのと同じ Python インタープリターを使用してライブラリをインストールしました。
2. インストールは、許可やネットワークエラーなしで正常に完了しました。

第3章 クイックスタート

barman を使用したアーキテクチャバックアップデザインで述べられているように、専用ホストで Barman をセットアップすることをお勧めします。ただし、このチュートリアル例では、次のホストを前提としています。

- *pghost* Postgres が実行されているホスト。
- *barmanhost* Barman がセットアップされるホスト。

barmanhost に従って、Barman が *barmanhost* に既にインストールされていると仮定すると、次の手順に進むことができます。

3.1 最初のサーバーの構成

Barman は、さまざまなバックアップおよび WAL アーカイブ戦略をサポートしています。ここでは、最も一般的に使用される 2 つのアーキテクチャをセットアップするための簡単なレシピを見つけます。ニーズに最も適したものを選択し、次のセクションに進みます。

3.1.1 WAL ストリーミングを使用したストリーミングバックアップ

この戦略では、バックアップと WAL アーカイブの両方に Postgres 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ため、実装するには *pghost* と *barmanhost* 間のネイティブ Postgres 接続のみが必要です。このアプローチの主な利点は、SSH 接続が必要ないため、他の戦略と比較してセットアップが簡単なオプションになることです。

バックアップは *pg_basebackup* ユーティリティに、WAL ファイルの転送は *pg_receivewal* に依存します。したがって、事前に両方のツールを *barmanhost* にインストールしておく必要があります。これらのツールのインストール方法の詳細が必要な場合は、*pg_basebackup* セクションを確認してください。

1. 最初のステップとして、Postgres サーバーで必要なユーザーを作成しましょう。 *pghost* で、次のコマンドを実行します。

```
createuser -s -P barman
```

このコマンドは、**barman** と呼ばれる新しい Postgres スーパーユーザーを作成します。これは、Postgres サーバーのメンテナンスタスクに Barman が使用します。または、スーパーユーザー権限は持たないが、**barman** のレシピに従って、必要な操作を実行するために必要な権限を持つユーザーを作成できます。

```
createuser -P --replication streaming_barman
```

このコマンドは、レプリケーション特権を持つ **streaming_barman** という新しい Postgres ユーザーを作成します。これは、Barman サーバーにファイルを転送するために `pg_receivewal` および `pg_basebackup` を呼び出すときに Barman が使用します。

どちらの `createuser` コマンドも、パスワードの入力を求めます。パスワードは、`barmanhost` の Barman ホームディレクトリの下の `.pgpass` という名前の `createuser` に追加することをお勧めします。ストリーミング接続の構成方法の詳細が必要な場合は、`createuser` セクションを確認してください。

これ以降、このセクションでは、Postgres ユーザーとバックアップする Postgres サーバーの両方が既に作成されていることを前提としています。また、`postgres` という名前のデータベースが利用できることを想定しているため、Barman はそのデータベースを介して Postgres サーバーに接続できます。

2. 次に、以前に作成したユーザーへのアクセスを許可してください `barmanhost`。 `pghost` で、次の HBA ルールを `pg_hba.conf` ファイルに追加します。

```
# allows access to the barman user from barmanhost
host    all    barman    barmanhost/32    md5
# allows access to the streaming_barman user from barmanhost
host    replication    streaming_barman    barmanhost/32    md5
```

次に、Postgres 構成をリロードして、新しい HBA ルールを有効にします。 `pghost` で、次を実行します。

```
psql -c "SELECT pg_reload_conf();"
```

3. Still on `pghost`, make sure your Postgres server is properly configured for WAL streaming. On its `postgresql.conf` file, assert that `wal_level` is set to `replica` or `logical`:

```
wal_level = replica
```

`wal_level` 構成値が変更された場合は、Postgres サーバーを再起動して変更を有効にします。

4. 次に、Barman で最初のバックアップサーバーを構成します。 `barmanhost` で、次の内容を使用してファイルを `/etc/barman.d/streaming-backup-server.conf` に作成します。

```
[streaming-backup-server]
description = "Postgres server using streaming replication"
streaming_archiver = on
backup_method = postgres
streaming_conninfo = host=pghost user=streaming_barman dbname=postgres
slot_name = barman
create_slot = auto
conninfo = host=pghost user=barman dbname=postgres
```

そこで

- `[streaming-backup-server]` は、Barman 上のバックアップサーバーに選択した名前です。
- `description` は、バックアップサーバーの説明テキストです。
- `streaming_archiver = on` は、このバックアップサーバーの WAL ファイルがストリーミングレプリケーションを使用して Postgres から Barman に転送されることを Barman に指示します。

- `backup_method = postgres` は、このサーバーがバックアップ方法として `postgres` を使用することを Barman に伝えます。これは、本質的に `pg_basebackup` を使用してバックアップを取ることを意味します。
- `streaming_conninfo` は、Postgres サーバーへの `libpq` 接続の接続文字列です。これは、`pg_receivewal` および `pg_basebackup` が Barman サーバーにファイルを転送するために使用する接続です。
- `slot_name` は、このバックアップサーバーが `pg_receivewal` を介して WAL をストリーミングするために使用される Postgres の物理レプリケーションスロットの名前です。
- `create_slot = auto` は、Postgres でレプリケーションスロットを自動的に作成する必要があることを Barman に指示し、事前の手動作成は必要ありません。
- `conninfo` は、Barman がメンテナンスの目的で使用する Postgres サーバーへの `libpq` 接続の接続文字列です。

`barmanhost` で、次を実行します。

```
barman list-servers
```

Barman に構成されたすべてのバックアップサーバーを含む出力が表示され、新しいサーバーを認識したことを確認します。

```
streaming-backup-server - Postgres server using streaming replication
```

5. Once finished with the configuration of both Barman and Postgres servers, you should be ready to go! Execute the following command on `barmanhost` to check that everything is OK with your server:

```
barman check streaming-backup-server
```

WAL アーカイブに関連する失敗したチェックが表示された場合でも、心配する必要はありません。これは、Barman がまだ完全な WAL ファイルを受け取っていないことを意味します。おそらく、サーバーが最初に作成されてから Postgres サーバーで WAL セグメントがスイッチされていないためです。次のコマンドを使用して、`barmanhost` から WAL スイッチを強制できます。

```
barman switch-wal --force streaming-backup-server
```

また、レプリケーションスロットと `pg_receivewal` に関連するチェックが失敗した場合は、次のコマンドを実行します。

```
barman cron
```

このコマンドは、Barman サーバーでメンテナンスタスクを実行するバックグラウンドプロセスを開始します。これらのタスクには、Postgres のレプリケーションスロットの作成、`create_slot = auto` の場合、および `pg_receivewal` プロセスの起動が含まれます。

`check` コマンドを再度実行し、失敗したチェックが表示されないことを確認します。

```
barman check streaming-backup-server
```

このサーバーは、バックアップを取得し、Postgres サーバーから WAL ファイルを受信する準備が整いました。 *taking your first backup* セクションに移動します。

3.1.2 WAL アーカイブを使用した Rsync バックアップ

この戦略は、Rsync および SSH 接続に依存して、バックアップファイルと WAL ファイルを Barman サーバーに転送します。

SSH 接続に依存するため、`pghost` と `barmanhost` の間で双方向のパスワードなしの SSH 接続が必要です。これを設定する方法の詳細については、`pghost` セクションを参照してください。

1. 最初のステップとして、Postgres サーバーで必要なユーザーを作成しましょう。`pghost` で、次のコマンドを実行します。

```
createuser -s -P barman
```

このコマンドは、**barman** と呼ばれる新しい Postgres スーパーユーザーを作成します。これは、Barman がメンテナンスタスクと、Postgres 低レベル API を使用してバックアップコマンドを発行するために使用されます。または、スーパーユーザー権限は持たないが、**barman** のレシピに従って、必要な操作を実行するために必要な権限を持つユーザーを作成できます。

`createuser` コマンドは、パスワードの入力を求めます。パスワードは、`barmanhost` の Barman ホームディレクトリの下に `.pgpass` という名前の `createuser` に追加することをお勧めします。

これ以降、このセクションでは、この Postgres ユーザーとバックアップする Postgres サーバーが既に作成されていることを前提としています。また、`postgres` という名前のデータベースが利用できることを想定しているため、Barman はそのデータベースを介して Postgres サーバーに接続できます。

2. 次に、以前に作成したユーザーへのアクセスを許可してください `barmanhost`。`pghost` で、次の HBA ルールを `pg_hba.conf` ファイルに追加します。

```
# allows access to the barman user from barmanhost
host    all    barman    barmanhost/32    md5
```

次に、Postgres 構成をリロードして、新しい HBA ルールが有効になります。`pghost` で、次を実行します。

```
psql -c "SELECT pg_reload_conf();"
```

3. Still on `pghost`, make sure your Postgres server is properly configured for WAL archiving. On its `postgresql.conf` file, assert the following parameters are properly set:

```
wal_level = replica
archive_mode = on
archive_command = 'barman-wal-archive barmanhost rsync-backup-server %p'
```

注釈

Barman 2.6 以降、`archive_command` を介して WAL ファイルをアーカイブする推奨される方法は、上記の例のように、`barman-wal-archive` ユーティリティを使用することです。このユーティリティを

使用するには、`pghost` に `barman-cli` パッケージもインストールしていることを確認します。詳細および代替コマンドオプションについては、`archive_command` セクションを確認してください。

4. 次に、Barman で最初のバックアップサーバーを構成します。`barmanhost` で、次の内容を使用して構成ファイルを `/etc/barman.d/rsync-backup-server.conf` に作成します。

```
[rsync-backup-server]
description = "Postgres server using Rsync and WAL archiving"
archiver = on
backup_method = rsync
reuse_backup = link
backup_options = concurrent_backup
ssh_command = ssh postgres@pghost
conninfo = host=pghost user=barman dbname=postgres
```

そこで

- `[rsync-backup-server]` は、Barman 上のバックアップサーバーに選択した名前です。
- `description` は、バックアップサーバーの説明テキストです。
- `archiver = on` は、このバックアップサーバーの WAL ファイルが、Postgres で構成された `archive_command` を使用して Postgres から Barman に転送されることを Barman に指示します。
- `backup_method = rsync` は、このバックアップサーバーがバックアップ方法として `rsync` を使用することを Barman に伝えます。これは本質的に、Rsync でクラスターファイルをコピーすることを意味します。
- `reuse_backup = link` は、以前のバックアップのファイルを再利用し、このサーバーの新しいバックアップを取るたびにストレージとネットワークリソースを節約することにより、データ重複排除を行うことを Barman に指示します。詳細については、`reuse_backup = link` セクションを確認してください。
- `backup_options = concurrent_backup` は、Barman がバックアップを行うときに Postgres サーバーで非排他的なバックアップコマンドを発行することを示します。
- `ssh_command` は、`barmanhost` から `pghost` に接続するために使用される SSH コマンドです。この構成値をそれに応じて置き換えます。
- `conninfo` は、Barman がメンテナンスの目的で使用する Postgres サーバーへの `:term:libpq` 接続の接続文字列です。

`barmanhost` で、次を実行します。

```
barman list-servers
```

Barman に構成されたすべてのバックアップサーバーを含む出力が表示され、新しいサーバーを認識したことを確認します。

```
rsync-backup-server - Postgres server using Rsync and WAL archiving
```

5. Once finished with the configuration of both Barman and Postgres servers, you should be ready to go! Execute the following command on `barmanhost` to check that everything is OK with your server:

```
barman check rsync-backup-server
```

WAL アーカイブに関連する失敗したチェックが表示された場合でも、心配する必要はありません。これは、Barman がまだ WAL ファイルを受信していないことを意味します。おそらく、サーバーが最初に作成されてから Postgres サーバーで WAL セグメントがスイッチされていないためです。次のコマンドを使用して、`barmanhost` から WAL スイッチを強制できます。

```
barman switch-wal --force rsync-backup-server
```

次に、次のコマンドを実行します。これにより、Barman サーバーでメンテナンスタスクを実行するバックグラウンドプロセスが開始されます。

```
barman cron
```

`check` コマンドを再度実行し、失敗したチェックが表示されないことを確認します。

```
barman check rsync-backup-server
```

このサーバーは、バックアップを取得し、Postgres サーバーから WAL ファイルを受信する準備が整いました。 *taking your first backup* セクションに移動します。

3.2 最初のバックアップの取得

バックアップサーバーにどの戦略を選択するかに関係なく、前の手順を完了すると、準備が整います。次のコマンドを実行して、バックアップを取得できます。

```
barman backup --name first-backup <server_name>
```

コマンドが完了したら、次のコマンドを使用してバックアップサーバーのすべてのバックアップをリストできます。

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

そして、次のコマンドで特定のバックアップの詳細を表示します。

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

3.3 バックアップの復元

バックアップを復元する必要がある場合は、次のコマンドで行うことができます。

```
barman restore <server_name> first-backup /path/to/restore
```

リモートサーバーに復元する場合、Barman ホストから宛先ホストへのパスワードなしの SSH 接続が必要で、その SSH コマンドは `--remote-ssh-command` オプションを使用して指定する必要があります。

```
barman restore --remote-ssh-command="ssh user@host" <server_name> first-backup /path/  
→to/restore
```


第4章 barmanを使用したアーキテクチャバックアップデザイン

効果的なディザスタリカバリー計画は、注意深く設計されたアーキテクチャから始まります。重要な決定には、他の考慮事項の中でも、バックアップサーバーをホストする場所とバックアップと WAL ファイルを転送する方法の決定が含まれます。それを念頭に置いて、このセクションでは、Barman でバックアップを展開および管理するためのいくつかの異なるセットアップについて説明します。

4.1 Barman をインストールする場所

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

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

Barman をセットアップするための「フリーサイズ」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。

- Barman を専用サーバーにインストールします。
- Postgres サーバーと同じストレージを共有しないでください。
- Barman をインフラ監視ツールと統合します。
- 運用に展開する前にすべてをテストします。

ディザスタリカバリーアーキテクチャのモデリングを開始するための良い方法には、次のものが含まれます。

- 次のような Postgres と Barman の場所に関して、いくつかの可能なアーキテクチャをデザインします。
 1. 同じデータセンター。
 2. 同じ大都市圏内の異なるデータセンター。
 3. さまざまな大都市圏のさまざまなデータセンター。
- 各仮説の長所と短所を詳しく説明します。
- 費用対効果分析を使用して、システムの `SPOF` を評価します。
- 決定を下し、初期ソリューションを実装します。

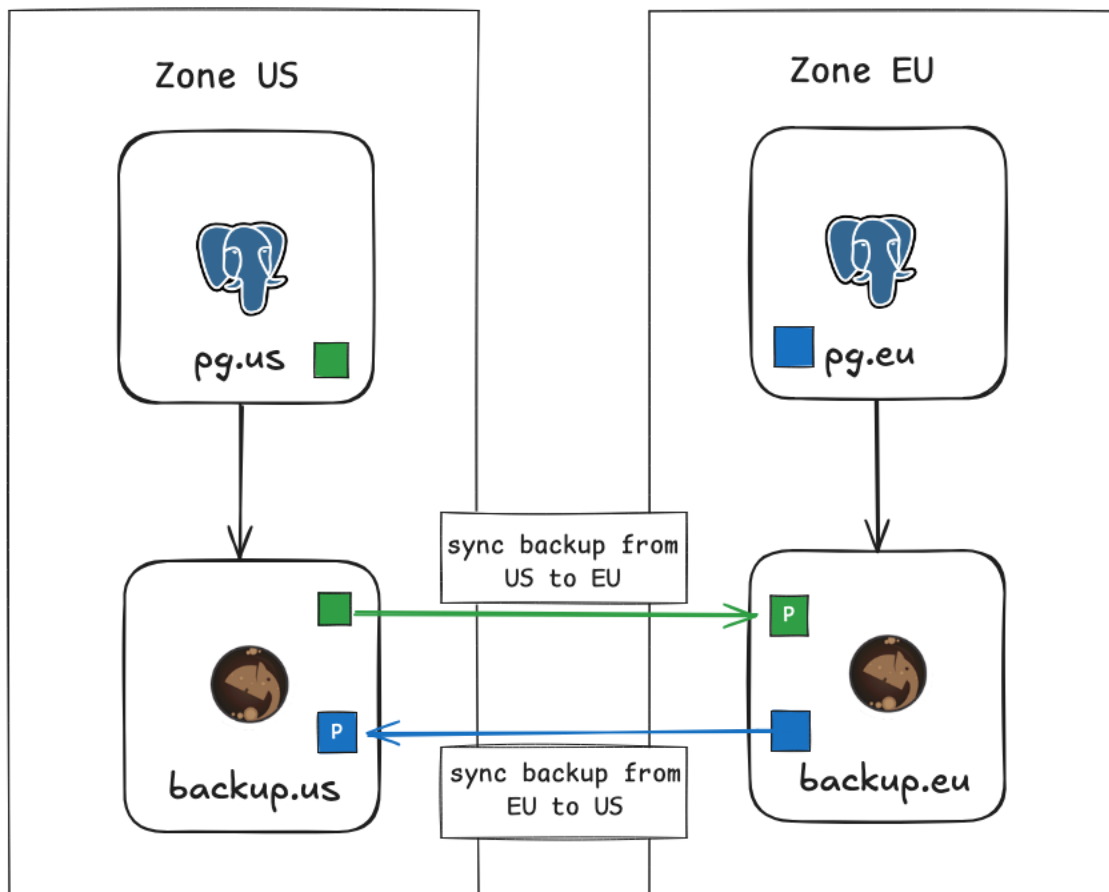
それを念頭に置いて、Barman の非常に一般的なセットアップは、Postgres サーバーと同じデータセンターにインストールすることです。その場合、SPOF はデータセンター自身自身です。ただし、このよ

うな :term:`SPOF` のインパクトは、:ref:`geographical redundancy` Barman 2.6 で導入された *geographical redundancy* などの機能で大幅に軽減できます。

地理的冗長性を使用すると、別のデータセンター/アベイラビリティゾーンにある Barman インスタンスを依存して、Postgres ノードと同じデータセンターにある Barman サーバーのコンテンツ全体を同期できます。また、Barman で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ことを考えると、一部のサーバーがローカルの Postgres 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 インストールクロスサイトバックアップ。以下の図は、2つのアベイラビリティゾーン1つはヨーロッパに、もう1つは米国にあり、それぞれにプライマリ Postgres サーバーがあり、ローカルの Barman インストールでバックアップされ、多層バックアップのために他の Barman サーバーパッシブとして定義に中継されます。Rsync / SSH 経由。地理的冗長性の詳細については、*geographical redundancy* セクションで入手できます。

Backup architecture for Postgres

Multi-location geographical redundancy



Hook Scripts のおかげで、Barman のバックアップは、tar を介したテープなどのさまざまなメディア、またはクラウドプロバイダーのオブジェクトストレージバケットのような場所にエクスポートできます。

どのような決定も永遠ではないことに注意してください。一方の方法で開始し、時間の経過とともに自分に最適なソリューションに適応できます。ただし、最初はシンプルにするようにしてください。

4.2 1人のBarman、多数のPostgresサーバー

Barmanによって最初に紹介されたもう1つの関連機能は、マルチプルの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Postgresインスタンスからのバックアップデータを一元化した方法で保存できます。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、Postgresサーバーが中央のBarmanサーバーを中心にローテーションする/「スタースキーマ」を形成できます。

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

4.3 バックアップ戦略

バックアップ戦略の選択も、セットアップにおいて重要な役割を果たします。Barmanは、トランスポートメカニズムとしてSSHを使用するRsync、またはPostgresストリーミングレプリケーションプロトコルを使用するpg_basebackupを使用してバックアップを取得できます。これら2つの方法のいずれかを選択することは、決定する必要がありますが、一般的な使用のために、現在サポートされているPostgresのすべてのバージョンでストリーミングレプリケーションを使用することをお勧めします。

注釈

Barmanはストリーミングバックアップを使用するときにpg_basebackupを利用するため、現在パレルバックアップなどの機能は利用できません。この場合、帯域幅制限にはいくつかの制限があります-Rsyncを介した従来の方法と比較して。17より前のPostgresバージョンでは、この方法を使用する場合、インクリメンタルバックアップも利用できません。

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

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

4.4 WAL アーカイブ戦略

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

1. WAL streaming involves transferring WAL files from the Postgres server with `pg_receivewal` using the Postgres streaming replication protocol. With WAL streaming, WALs are transferred while they are still being generated, which means that Barman doesn't have to wait for WAL segments to be completely filled in order to receive them. Such mechanism makes WAL streaming able to significantly reduce the risk of data loss, bringing *RPO* down to near zero values. It is also possible to add Barman as a synchronous WAL receiver in your Postgres cluster and achieve zero data loss (RPO=0). With the use of replication slots, we can also assure that no WAL file is recycled before being successfully received by Barman.

pg_receivewal のインストール方法の詳細については、[pre-requisites for wal streaming](#) を参照してください。

i 注釈

WAL ストリーミングを使用する場合、常にプライマリノードからストリーミングすることをお勧めします。これは、フェイルオーバーが発生した場合でも、すべての WAL が Barman によって受信されるようにするためです。

2. Barman also supports standard WAL file archiving, which is achieved using the Postgres `archive_command`, either using Rsync/SSH or `barman-wal-archive` from the `barman-cli` package. With this method, WAL files are archived only when Postgres switches to a new WAL file, which normally happens every 16MB worth of data changes. This approach offers more flexibility by allowing you to pick a tool of your choice for transferring the WAL files.

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

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

i 注釈

Barman の以前のバージョンでは、WAL アーカイブと WAL ストリーミングの両方を使用することを推奨していました。これは、9.4 より前の Postgres バージョンがレプリケーションスロットをサポートしていないため、WAL ストリーミングだけではすべての WAL が Barman の WAL アーカイブに安全に保存されることを保証できないためです。Postgres のサポートされているすべてのバージョンにレプリケーションスロットがあるため、WAL ストリーミングのみを構成するだけで十分です。

4.5 バックアップの2つの一般的なシナリオ

あなたの人生を楽にするために、このセクションでは、Barman の特定の Postgres サーバーの 2 つの最も一般的なシナリオをまとめます。これは、Barman でバックアップすることにしたすべてのサーバーに対して行う必要がある決定であることに注意してください。これは、同じ Barman サーバー内で異種セットアップを使用できることを意味します。

pg および backup を使用して、それぞれ Postgres および Barman サーバーを参照します。ただし、実際には、アーキテクチャには、repmgr、pgBouncer、Nagios/Icinga などの他のテクノロジーが含まれる可能性が高くなります。

4.5.1 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

Streaming Backups で述べたように、このアプローチでは、Postgres ストリーミングプロトコルを使用してクラスターファイルを Barman サーバーに転送します。これは、`pg_basebackup` ユーティリティを使用して実行されます。Barman では、Barman サーバー構成に `backup_method = postgres` を含めることにより、この方法を設定できます。

このアプローチでは、Postgres 17 以降で利用可能な `pg_basebackup` が提供する *block-level incremental backups* サポートを活用できます。ブロックレベルのインクリメンタルバックアップは、重複排除率の点で

Rsync 戦略が提供する *block-level incremental backups* よりもはるかに効率的である傾向があります。

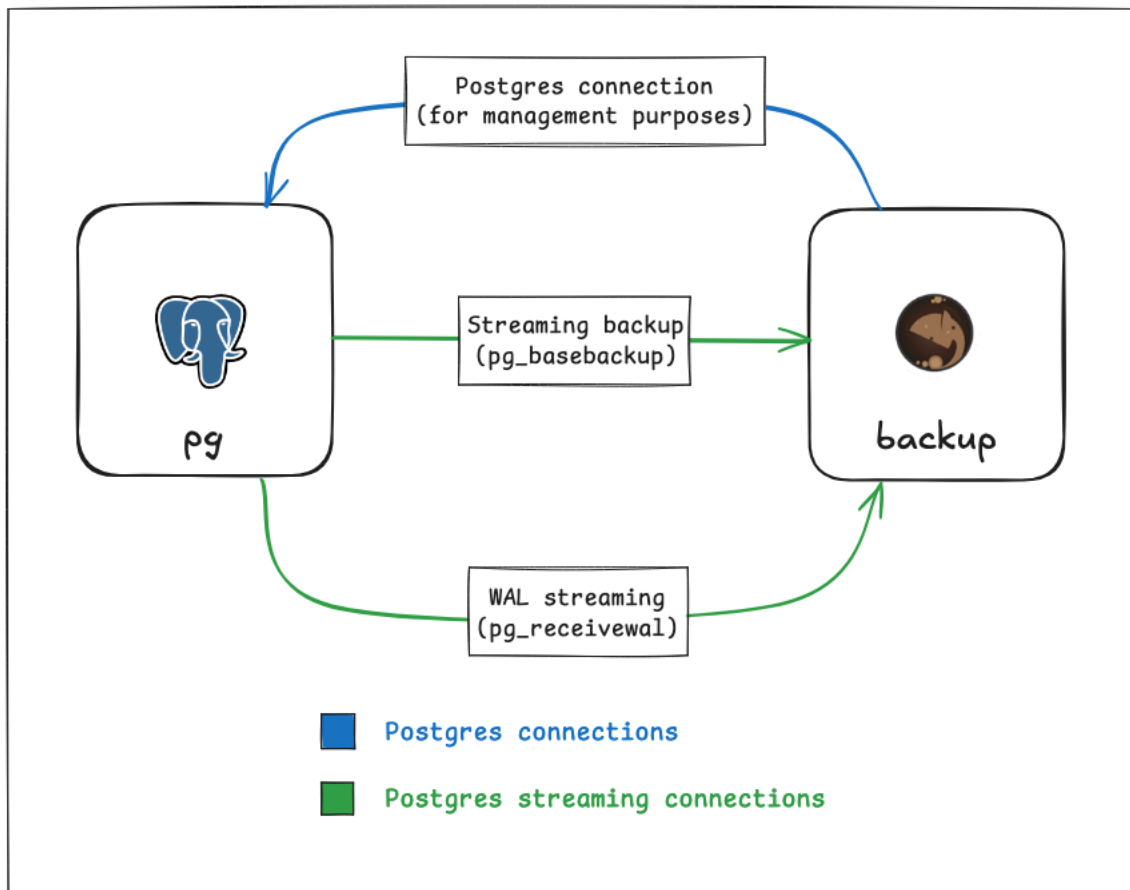
この方法は、WAL ファイルの WAL ストリーミングと組み合わせて使用されます。Barman の用語では、このセットアップは、バックアップおよびアーカイブ操作に SSH 接続を使用しないため、ストリーミング専用のセットアップと呼ばれます。これは、Docker 環境や高度に規制された環境などに特に適しており、非常に実用的です。

ストリーミングバックアップ方法は、通常、ほとんどのユースケースで推奨されるアプローチです。

次の図は、この設定が実際にどのように機能するかを示しています。

Backup architecture for Postgres

Full streaming architecture



構成するには、次のものがが必要です。

1. 管理、調整、およびモニタリングを目的とした Postgres への標準接続。
2. A streaming replication connection to be used by both `pg_basebackup` (for base backup operations) and `pg_receivewal` (for WAL streaming).

4.5.2 シナリオ 2 rsync / SSH を介したバックアップ

rsync backups 概念で述べたように、このアプローチは Rsync に依存してバックアップファイルを Barman サーバーに転送します。これは、サーバーをバックアップモードにし、Rsync を使用してクラスターファイルを転送することにより実行されます。

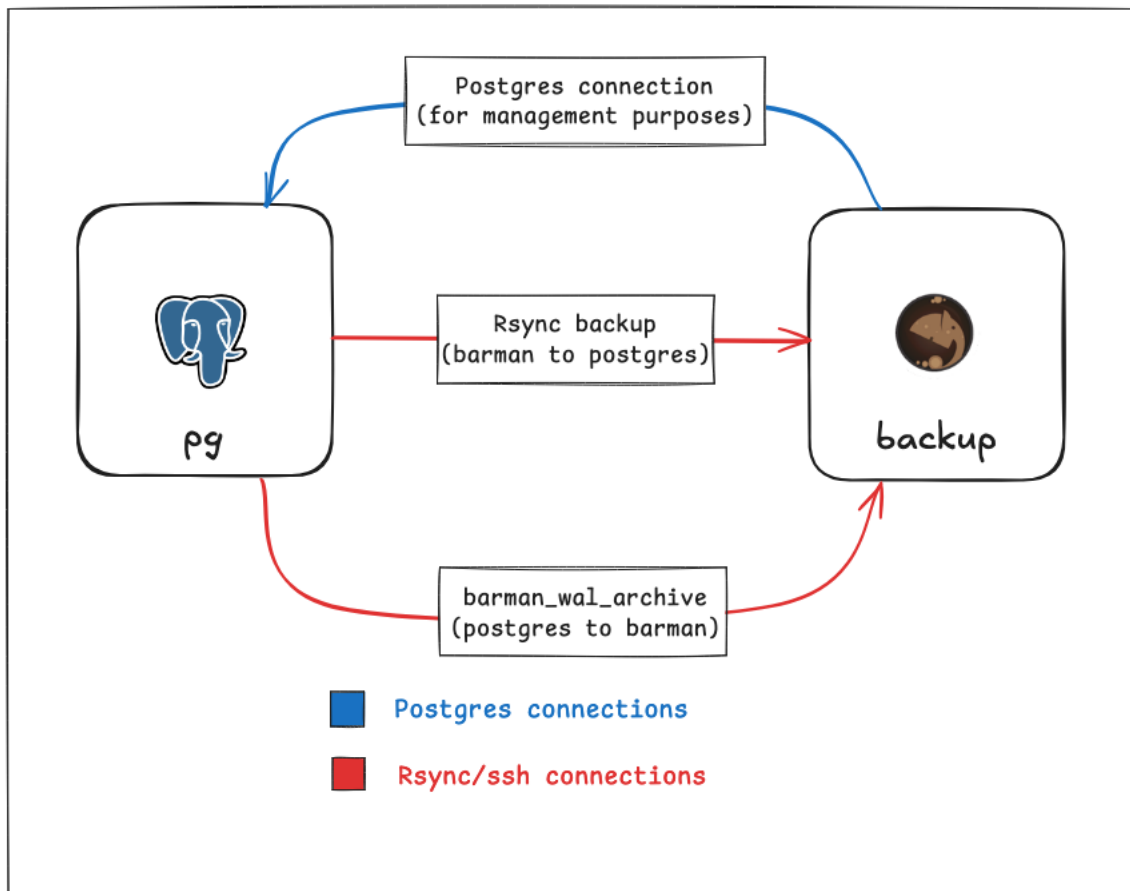
このアプローチの主な利点は、バックアップ操作の実行時に *parallel jobs* を使用できることです。これにより、バックアップの全体的な時間を大幅に短縮できます。また、*parallel jobs* を取得する機能も提供します。これは、重複排除のために以前のバックアップのファイルを再利用します。ファイルレベルのインクリメンタルバックアップは、各バックアップが他から完全に独立しているため、*parallel jobs* よりも柔軟です。つまり、インクリメンタルバックアップに影響を与えることなくルートバックアップを削除できます。

この方法のもう 1 つの利点は、テーブルスペースごとなど、帯域幅の使用をより詳細に制御できることです。詳細については、*Managing Bandwidth Usage* を確認してください。

次の図は、この設定が実際にどのように機能するかを示しています。

Backup architecture for Postgres

Remote copy architecture



構成するには、次のものがが必要です。

1. 管理、調整、およびモニタリングを目的とした Postgres への標準接続。
2. An SSH connection to be used by Rsync for base backup operations that allow the **barman** user on the Barman server to connect as the **postgres** user on the Postgres server.
3. An SSH connection for WAL archiving to be used by the `archive_command` in Postgres that allows the **postgres** user on the Postgres server to connect as **barman** user on the Barman server.

4.5.3 ハイブリッドシナリオ

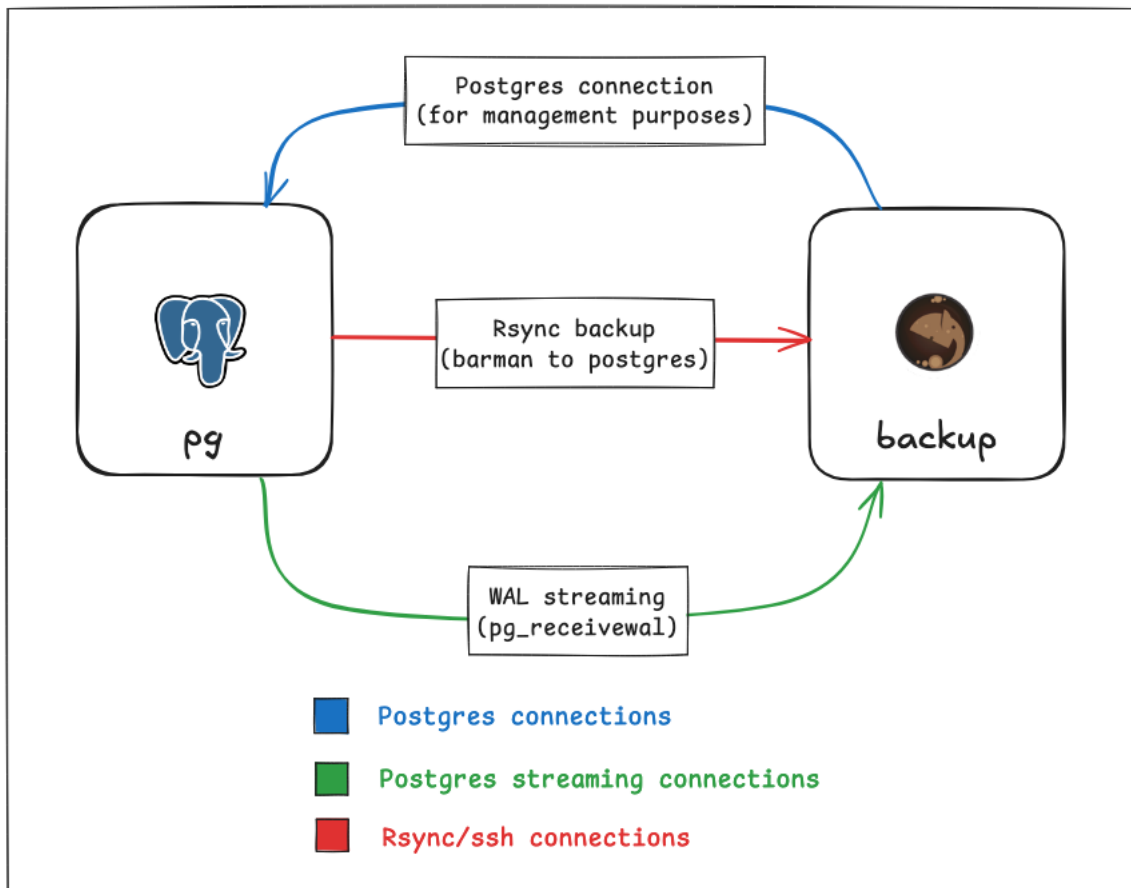
また、rsync バックアップと WAL ストリーミングを組み合わせたハイブリッドアプローチを使用して、rsync バックアップファイルレベルのインクリメンタルバックアップ、パラレルジョブなどと WAL ストリーミングのものより効率的な WAL 転送の利点を備えた結果を達成することもできます。、オプションの RPO ゼロ)。

次の図は、この設定が実際にどのように機能するかを示しています。

Backup architecture for Postgres

Frame

Rsync backup with WAL streaming architecture



これを行うには、Barman サーバー構成で `backup_method` を `rsync` として構成し、`streaming_archiver` を `on` に設定する必要があります。また、WAL アーカイブに `pg_receivewal` が使用するストリーミング

レプリケーション接続と、Rsync がベースバックアップ操作に使用する SSH 接続も必要です。

4.5.4 WAL アーカイブフォールバック冗長性

Barman を使用すると、WAL ストリーミングに加えて WAL アーカイブを構成して、WAL ストリーミングが失敗した場合のフォールバックメカニズムを設けることができます。これは、上記で説明した `backup_method` 構成のいずれかで実行できます。

1. When using the streaming-only setup, described in the [Scenario 1](#), you can also configure WAL archiving via SSH in addition to WAL streaming. In such scenarios, WAL archiving would act as a fallback mechanism in case WAL streaming failed.
2. When using the Rsync backup method, described in [Scenario 2](#), you can also configure WAL streaming instead of using the `archive_command` in order to have a lower *RPO*. You can also opt for configuring WAL streaming in addition to WAL archiving and have both options.

いずれの場合も、Barman は WAL ファイルがアーカイブで重複していないことを自動的に確認し、1 回のみ保存します。

4.6 地理的な冗長性

Barman の非常に一般的なセットアップは、PostgreSQL サーバーが存在するのと同じデータセンターにインストールすることです。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、バックアップ層の数を増やすために Barman が提供する 2 つの機能のおかげで、緩和できます。

1. 地理的冗長性 Barman 2.6 で導入
2. フックスクリプト

地理的冗長性を使用すると、別のデータセンター/アベイラビリティゾーンにある別の Barman インスタンスに依存し、プライマリ Barman サーバーのコンテンツ全体を同期できます。

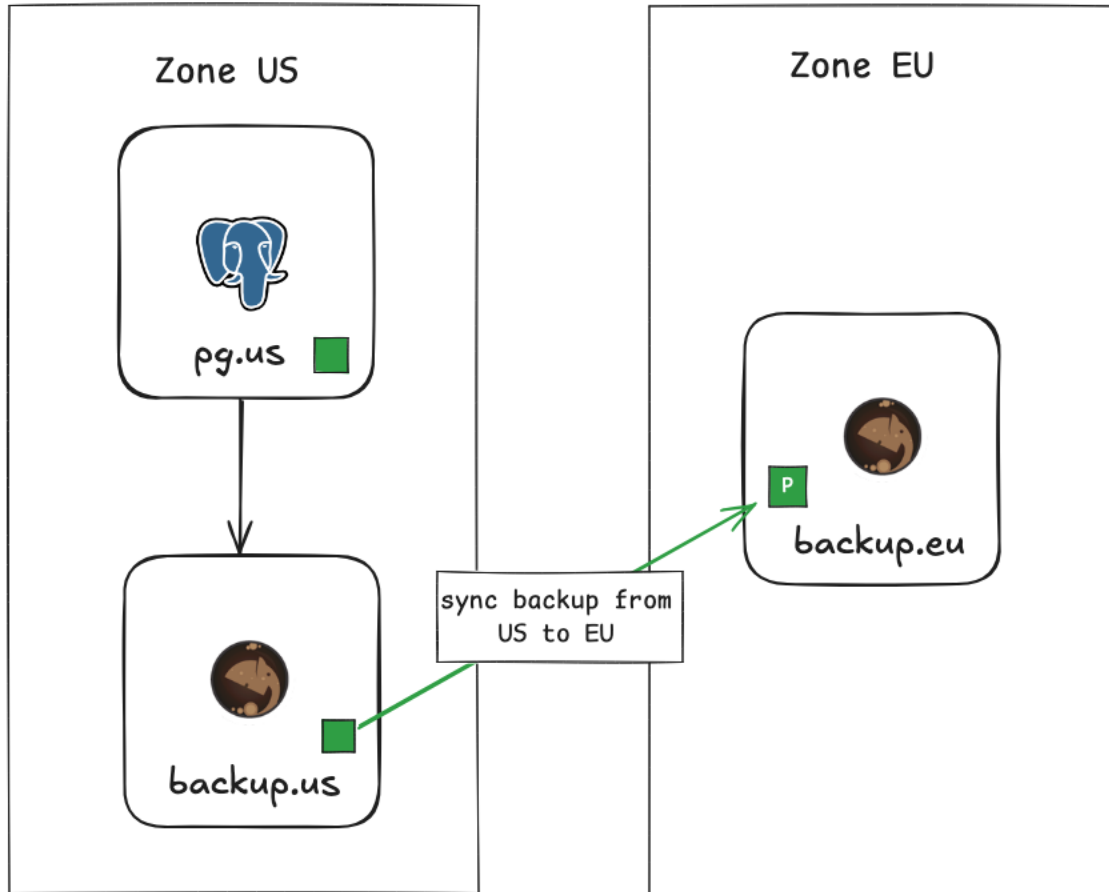
さらにあります。Barman で地理的冗長性をグローバルレベルだけでなくサーバーレベルでも構成できることを考えると、一部のサーバーがローカルの PostgreSQL サーバーに直接接続し、他のサーバーがサブセットをバックアップする Barman のハイブリッドインストールを作成できます。さまざまな Barman インストールのクロスサイトバックアップ。

以下の図は、2 つのアベイラビリティゾーン 1 つはヨーロッパに、もう 1 つは米国にあり、それぞれにプライマリ PostgreSQL サーバーがあり、ローカルの Barman インストールでバックアップされ、多層バックアップのために他の Barman サーバーパッシブとして定義に中継されます。rsync / SSH 経由。地理的冗長性の詳細については、[Geographical Redundancy](#) セクションで入手できます。

次の画像は、この設定が実際にどのように機能するかを示しています。

Backup architecture for Postgres

Geographical redundancy



4.7 クラウド スナップショット バックアップ

Barman は、クラウドスナップショットバックアップもサポートしており、クラウド内の Postgres サーバーが存在するストレージボリュームのスナップショットを取得します。Barman は現在、Azure、Google、および AWS でこの方法をサポートしています。この方法の前提条件は、Postgres サーバーが存在するクラウドプロバイダーによって異なるため、詳細については [クラウド スナップショット バックアップ](#) セクションを確認することをお勧めします。

第5章 前提条件

このセクションでは、ユースケースに応じて Barman 環境をセットアップするために必要ないくつかの要件と構成について詳しく説明します。

このセクションでは、次のホストを想定しています。

- `pghost` Postgres が実行されているホスト。
- `barmanhost` Barman がセットアップされるホスト。

5.1 Postgres ユーザー

Barman でサーバーに関する情報を収集するには、Postgres インスタンスへの接続が必要です。この接続をセットアップするための推奨される方法は、Postgres で `barman` という名前の専用ユーザーを作成することです。このユーザーには、必要な特権が必要です。

注釈

以下で実行する `createuser` コマンドは、パスワードの入力を求めます。パスワードは、`barmanhost` の Barman ホームディレクトリの下に `.pgpass` という名前の `createuser` に追加することをお勧めします。さらに、Postgres が提供するものの中から希望のクライアント認証方法を選択できます。詳細については、`createuser` を確認してください。

Postgres で `barman` という名前のスーパーユーザーを作成するには、`pghost` で次のコマンドを実行します。

```
createuser -s -P barman
```

または、必要な特権のみを持つユーザーを選択する場合、次の手順を実行します。

1. `pghost` で、次のコマンドを実行して、Postgres に `barman` という名前のユーザーを作成します。

```
createuser -P barman
```

2. `pghost` で、`psql` インターフェイスで、次のステートメントを実行します。

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

Postgres バージョン 14 以前のバージョンを使用する場合、ファンクション `pg_backup_start` および `pg_backup_stop` の名前と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 2 行を次のように置き換える必要があります。

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

注釈

Postgres バージョン 13 以下では、`barman switch-wal` コマンドの `--force` オプションは、スーパーユーザーがないと機能しません。Postgres バージョン 15 以降では、ステートメント `GRANT pg_checkpoint TO barman;` を実行することにより、`pg_checkpoint` ロールを付与してスーパーユーザーなしで使用できます。

5.2 Postgres 接続

どのバックアップ方法を使用しているかに関係なく、Postgres インスタンスへの接続が必要です。この接続は、Barman がデータベースサーバーとのアクティビティを調整するために、およびモニタリングのために必要です。

`barmanhost` がスーパーユーザーとして、または必要な特権を持つユーザーとしてデータベースサーバーに接続できることを確認します。Postgres 接続のセットアップに関する詳細情報は、`barmanhost` で見つけることができます。

ユーザーを作成したら、`barmanhost` で次のコマンドを実行して、Postgres インスタンスに接続できることをアサートします。

```
psql -c 'SELECT version()' -U barman -h pghost postgres
```

`postgres` は、Barman が Postgres インスタンスに接続できる利用可能なデータベースです。

上記のコマンドが成功した場合、それはデータベースサーバーに正常に接続できることを意味します。上記の接続パラメーターは、サーバーの構成ファイルの `conninfo` パラメーターに書き込む必要があるものであるため、覚えておいてください。このコンテキストでは、このパラメーターは次のようになります。

```
[my-server]
; ...
conninfo = host=pghost user=barman dbname=postgres application_name=myapp
```

`application_name` はオプションのパラメーターです。

5.3 Postgres クライアントツール

Postgres サーバーと対話するには、Postgres クライアントツールが必要です。Barman が最も一般的に使用するツールは `pg_basebackup` および `pg_receivewal` です。これらは Postgres クライアントパッケージによって提供されます。

Debian または Ubuntu に Postgres クライアントパッケージをインストールするには、`barmanhost` で次のコマンドを実行します。

```
sudo apt-get install postgresql-client
```

または、`barmanhost` が RHEL、Rocky Linux、Alma Linux を使用している場合、次のレシピに従います。

```
sudo dnf install postgresql
```

5.4 Postgres ストリーミングレプリケーション接続

ストリーミングバックアップまたは WAL ファイルのストリーミングを使用する予定がある場合は、ストリーミング接続をセットアップする必要があります。さらに、[pre-requisites](#) セクションで共有されているように、Postgres クライアントツールをインストールする必要があります。

Postgres で `streaming_barman` という名前の専用ユーザーを作成することをお勧めします。次のコマンドでこれを行うことができます。

```
createuser -P --replication streaming_barman
```

注釈

以下で実行する `createuser` コマンドは、パスワードの入力を求めます。パスワードは、`barmanhost` の Barman ホームディレクトリの下の `.pgpass` という名前の `createuser` に追加することをお勧めします。さらに、Postgres が提供するものの中から希望のクライアント認証方法を選択できます。詳細については、`createuser` を確認してください。

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確認できます。

```
psql -U streaming_barman -h pghost -c "IDENTIFY_SYSTEM" replication=1
```

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

systemid	timeline	xlogpos	dbname
7139870358166741016	1	1/330000D8	

(1 row)

Postgres 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 Postgres 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、Postgres が管理できる同時ストリーミング接続の最大数を表します。

もう 1 つの重要なパラメーターは `max_replication_slots` で、これは Postgres が管理できるレプリケーションスロットの最大数を表します。このパラメーターは、ストリーミング接続を使用して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関連します。

```
max_replication_slots = 2
```

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

5.5 SSH 接続

`archive_command` を介して Rsync バックアップまたは WAL アーカイブを使用する予定がある場合、SSH 接続が必要です。

SSH は、リモートサーバーへのリモートシェルを開き、サーバーとローカルシステム間でファイルをコピーできるプロトコルとツールセットです。SSH の使用に関する詳細ドキュメントは [article /"SSH Essentials/" by Digital Ocean](#) で見つけることができます。

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

5.5.1 postgres ユーザーの SSH 構成

これまでに行ったことがない限り、**postgres** ユーザーの SSH キーを作成する必要があります。pgghost に **postgres** としてログインし、実行します。

```
ssh-keygen -t rsa
```

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

5.5.2 barman ユーザーの SSH 構成

barman ユーザーの SSH キーも作成する必要があります。barmanhost に **barman** としてログインし、実行します。

```
ssh-keygen -t rsa
```

繰り返しますが、パスフレーズは入力する必要はありません。

5.5.3 Postgres から Barman へ

pgghost から barmanhost への SSH 接続は、`archive_command` を使用して WAL ファイルを正しくアーカイブするために必要です。

pgghost から barmanhost に正常に接続するには、pgghost ユーザーの公開キーを、barmanhost の pgghost ユーザーの許可されたキーに保存する必要があります。このキーは、`.ssh/id_rsa.pub` という名前のファ

イルの `pghost` ユーザーホームディレクトリーにあり、そのコンテンツは、`barmanhost` の `pghost` ユーザーのホームディレクトリー内の `.ssh/authorized_keys` という名前のファイルに含まれる必要があります。`authorized_keys` ファイルが存在しない場合は、権限として `600` を使用して作成します。

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

```
ssh barman@barmanhost -C true
```

5.5.4 Barman から Postgres へ

`barmanhost` から `pghost` 間の SSH 接続は、`Rsync` を使用した従来のバックアップに使用されます。

`barmanhost` から `pghost` に正常に接続するには、`barmanhost` ユーザの公開キーを、`pghost` の `barmanhost` ユーザの許可されたキーに保存する必要があります。このキーは、`barmanhost` ユーザーホームディレクトリーの `.ssh/id_rsa.pub` という名前のファイルにあります。そのコンテンツは、`pghost` の `barmanhost` ユーザーのホームディレクトリー内の `.ssh/authorized_keys` という名前のファイルに含まれる必要があります。`authorized_keys` ファイルが存在しない場合は、権限として `600` を使用して作成します。

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

```
ssh postgres@pghost -C true
```

5.6 archive_command を介した WAL アーカイブ

WAL アーカイブ戦略 セクションで述べたように、`Barman` で `wals` をアーカイブするには2つのオプションがあります。ストリーミングレプリケーションプロトコルを使用して `WAL` ファイルをアーカイブする場合は、**WAL アーカイブ戦略** 概念と **WAL アーカイブ戦略** セクション、特に `WAL` ストリーミングによるストリーミングバックアップサブセクションを参照してください。それ以外の場合は、**WAL アーカイブ戦略** または `Rsync` / `SSH` で `archive_command` を使用して `WAL` アーカイブを構成できます。

5.6.1 barman-wal-archive を使用する

`Barman 2.6` 以降、書き込み先行ログファイルを安全にアーカイブするための推奨アプローチは、`barman-cli` パッケージの `barman-wal-archive` コマンドを利用することです。このパッケージをインストールする方法については、`barman-wal-archive` セクションを参照してください。

`rsync` や `SSH` などの従来の方法の代わりに `barman-wal-archive` を使用すると、`WAL` ファイルの `Barman` サーバーへの転送中のデータ破損のリスクを最小限に抑えます。従来の方法には、ファイルのコンテンツが宛先のディスクに適切にフラッシュおよび `fsync` されるという保証がありません。

`barman-wal-archive` ユーティリティは、`barman-wal-archive` コマンドと直接対話します。このコマンドは、受信した `WAL` ファイルが `fsync` され、各サーバーの正しい受信ディレクトリーに保存されるようにします。`archive_command` に必要なパラメーターはサーバーの名前だけであるため、誤って配置される可能性が減少します。

i 注釈

`barman-wal-archive` クライアントは、障害を最小限に抑え、PostgreSQL サーバーに WAL ファイルが蓄積するのを防ぐように設計されています。重複した WAL ファイルが Barman ホストに送信されると、コマンドは成功コード 0 で終了し、メッセージがログに記録されます。

複製ファイルの内容が Barman ホストに既に保存されているものと一致する場合、ファイルは無視されます。ただし、コンテンツが異なる場合、ファイルは Barman サーバーの `errors` ディレクトリに移動します。これにより、今後の `barman check` 実行では、不一致による障害が報告されます。

`barman-wal-archive` が Barman サーバーに接続できること、および Postgres サーバーが着信 WAL ファイルを受け入れるように正しく構成されていることを確認するには、次のコマンドを実行します。

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

ここで、`backup` は Barman ホストを指し、`pg` は Barman で構成された Postgres サーバーの名前であり、`DUMMY` は `-t` オプションを使用する場合に無視される WAL ファイル名のプレースホルダーです。

設定が正しい場合は、次が表示されます。

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

ユーティリティは SSH を介して通信するため、postgres ユーザーがバックアップサーバーに `barman` としてログインするように SSH キー認証が設定されていることを確認します。SSH 接続がデフォルト 22 以外のポートを使用する場合、`--port` オプションを使用してポートを指定できます。

WAL アーカイブを使用した [Rsync バックアップ](#) を参照して、作業を開始します。

5.6.2 Rsync / SSH を使用する

`archive_command` を構成するための **alternative approach** は、SSH を介して `rsync` コマンドを利用することです。ここでは、`pg` という名前の Postgres サーバー、`backup` という名前の Barman サーバー、および `barman` という名前のユーザーを効果的に設定するための初期手順を示します。

着信 WAL ディレクトリを見つけるには、次のコマンドを使用して、`incoming_wals_directory` 値を確認します。

```
barman show-servers pg | grep incoming_wals_directory

incoming_wals_directory: /var/lib/barman/pg/incoming
```

次に、`pg` ホスト上の Postgres インスタンスの `postgresql.conf` ファイルを編集して、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、前のコマンドから取得した実際のパスに置き換えてください。これらの変更を行った後、Postgres サーバーを再起動します。

archive_command プロセスのセキュリティを強化するには、より厳密なチェックを実装することを検討します。たとえば、次のコマンドは、rsync を実行する前にホスト名が一致することを確認します。

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

HOSTNAME を hostname --fqdn からの出力に置き換えます。このアプローチは、サーバーがクローン化される際の潜在的な問題に対するセーフガードとして機能し、復旧した Postgres インスタンスから WAL ファイルが送信されないようにします。

第6章 Barman チェック

check コマンドは、指定されたサーバーへの接続を検証し、サーバー構成が一貫していることを確認します。このコマンドは、一連のチェックを実行して、SSH の適切なセットアップと Postgres 接続が機能していること、バックアップディレクトリの存在、および WAL アーカイブの整合性を検証します。また、Postgres 構成、保持ポリシー、バックアップの有効性、および WAL アーカイブプロセスも検証します。検出された問題を特定して報告するため、Barman が提供する最も重要な機能の 1 つとなります。

このコマンドを実行すると、次のことが行われます。

- WAL アーカイブが正しく構成されていることを確認します。
- Postgres 接続と必要な特権を検証します。
- 必要なディレクトリがすべて存在することを確認します。
- 保持ポリシーが適切に設定されていることを確認します。
- バックアップと WAL ファイルの有効性を確認します。
- 構成エラーとアーカイバーの問題を検出します。

6.1 使用法

check コマンドを使用するには、次のコマンドを実行します。

```
barman check <server_name>
```

構成されているすべてのサーバーでチェックを実行するには、次を使用します。

```
barman check all
```

Tip

check コマンドをアラートおよびモニタリングインフラストラクチャと統合して、バックアップ環境の継続的な監視を保証します。--nagios オプションは、Nagios / Icinga のプラグインを作成するのに役立ち、サーバーのステータスをシームレスに監視できます。例、次のことを実行できます。

```
barman check --nagios <server_name>
```

これにより、Nagios と互換性のある形式で結果が出力され、既存のモニタリングセットアップへの統合が容易になります。

 Tip

他のバックアップおよびサーバー固有のチェックには、次のコマンドを使用できます。

- `barman check-backup` 物理バックアップの一貫性を検証するために必要なすべての WAL ファイルが適切にアーカイブされていることを確認します。このコマンドは `cron` ジョブによって使用され、各バックアップ操作の後に自動的に実行されます。詳細については、`barman check-backup` を参照してください。

6.2 出力を理解する

`barman check` を実行すると、次のような出力が表示されます。

```
$ barman check example
Server example:
  PostgreSQL: OK
  superuser or standard user with backup privileges: OK
  PostgreSQL streaming: OK
  wal_level: OK
  replication slot: OK
  directories: OK
  retention policy settings: OK
  backup maximum age: OK (no last_backup_maximum_age provided)
  compression settings: OK
  configuration files: OK
  pg_basebackup: OK
  pg_basebackup compatible: OK
  received WAL files: OK
  archiving: OK
  archive_mode: OK
  archive_command: OK
  continuous archiving: OK
  archive_timeout: OK
```

出力ラベルは、各チェックのステータスによって異なる場合があります。たとえば、チェックが失敗した場合、**FAILED** としてマークされ、問題のトラブルシューティングに役立つヒントが含まれる場合があります。チェックは合格したが警告がある場合、追加のコンテキストとともに **WARNING** としてマークされます。

`barman check` コマンドは、Barman 機能の次の各側面のチェックを実行します。

WAL Archive

- WAL アーカイブが正しく設定されていることを確認します。
- 受信ディレクトリとストリーミングディレクトリにある WAL ファイルの数を確認します。

 注釈

Barman 構成に `archiver = off` があり、受信ディレクトリに WAL が存在する場合、チェックは失敗します。これは、着信ディレクトリの WAL が、Postgres がまだ `archive_command` を使用しているか、ユーザーが `archiver` から `streaming_archiver` に切り替え、Barman のアーカイブに WAL が保存されていないことを示唆しているために発生します。

これを解決するには、着信の WAL が必要か、または安全に削除できるかどうかを判断する必要があります。これは次の方法で実行できます。

1. WAL が最も古いバックアップの `begin_wal` より新しいかどうかの確認。
2. これらの WAL が既に Barman のアーカイブに存在するかどうかの確認。

`streaming_archiver = off` と WAL がストリーミングディレクトリで見つかった場合、同じ問題が適用されます。

Postgres Connection

- Postgres 接続を検証します。
- サーバーバージョンがサポートされていることを確認します。
- 必要な特権とストリーミングサポートを確認します。

Local Tools Validity

- バックアップを取るための `pg_basebackup`、WAL ファイルのストリーミング用の `pg_receivewal` など、バックアップを取得し WAL を受信するためのローカルツールがデータベースサーバーのバージョンで正しく動作することを確認します。

Directory

- 必要なバックアップディレクトリがすべて存在することを確認します。
- ディレクトリが存在しない場合は作成します。

Retention Policy

- 保持ポリシー設定を検証します。

Backup Validity

- バックアップの有効性要件が満たされていることを確認します。
- バックアップの最大存続期間と最小サイズを確認します。

WAL Validity

- WAL アーカイブ要件が満たされていることを確認します。
- WAL ファイルの最大存続期間とサイズを確認します。

Configuration

- サーバーのメッセージリストのエラーメッセージを検査します。
- 見つかったエラーを出力します。

Identity

- ストリーミング接続から取得したシステム ID が、標準接続からのシステム ID およびディスクに保存されているものと一致することを確認します。

Archiver Errors

- error ディレクトリでアーカイブエラーの存在を検査します。

第7章 バックアップ

7.1 概要

backup コマンドは、構成ファイルパラメーターに従って、Postgres サーバー全体をバックアップするために使用されます。それを使用するには、次を実行します。

```
barman backup [OPTIONS] SERVER_NAME
```

注釈

backup コマンドの詳細については、backup コマンドリファレンスを参照してください。

重要

Barman との相互作用が予定されている場合、サーバーが正しく構成されていることを確認する必要があります。バックアップを作成する前にカバーする必要がある手順については、[Quickstart](#) および [Quickstart](#) セクションを参照してください。

警告

WAL ファイルが `archiver` または `streaming_archiver` オプションを介して Barman に正しくアーカイブされていない場合、バックアップの開始は失敗します。

Barman は、Postgres サーバーの複数のバックアップ方法を提供しており、それぞれに独自のアプローチと要件があります。

バージョン 2.0 より前のバージョンでは、Barman は標準バックアップとファイルレベルのインクリメンタルバックアップの両方を `rsync` にのみ依存していました。ストリーミングバックアップはこのバージョンで導入されました。バージョン 3.11 以降、Barman は、ストリーミング接続を介したブロックレベルのインクリメンタルバックアップもサポートします。

重要

Postgres 15 以降では、`exclusive` バックアップはサポートされなくなりました。バックアップを取る唯一の方法は、`concurrent` バックアップを使用することです。`backup_options` が設定されていない場合、Barman は自動的に `concurrent_backup` に設定します。

7.2 バックアップの要件

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

Barman 開発者は、XFS および ext4 ファイルシステムを使用して Barman を定期的にテストします。PostgreSQL と同様に、Barman は、バックアップと WAL の保存に使用される NFS マウントポイントに対して特別なことは何もしません。NFS で Barman を安全に使用するには、次の点が必要です。

- `barman_lock_directory` はローカルファイルシステム上にある必要があります。
- 少なくとも NFS プロトコルバージョン 4 を使用してください。
- ファイルシステムは、ハードおよび同期オプション `hard`、`sync` を使用してマウントする必要があります。

7.3 インクリメンタル バックアップ

インクリメンタルバックアップには、既存のバックアップを参照として使用し、Postgres サーバーでの最後のバックアップ以降に発生したデータ変更のみをコピーすることが含まれます。

Barman でのインクリメンタルバックアップの主な目的は次のとおりです。

- 完全なバックアッププロセスの時間を短縮します。
- 定期的なバックアップデータの重複排除にわたって冗長データを削除することにより、ディスク領域の使用量を削減します。

Barman は、2 種類のインクリメンタルバックアップをサポートしています。

- ファイルレベルのインクリメンタルバックアップ `rsync` を使用
- ブロックレベルのインクリメンタルバックアップ PostgreSQL 17 で `pg_basebackup` を使用

注釈

異なる種類のインクリメンタルバックアップには、相互互換性がありません。たとえば、`rsync` バックアップに加えてブロックレベルのインクリメンタルバックアップを取得することはできません。また、`pg_basebackup` で作成されたストリーミングバックアップに基づいてファイルレベルのインクリメンタルバックアップを取得することはできません。

7.4 帯域幅使用量の管理

`bandwidth_limit` オプショングローバルまたはサーバーごとに、1 秒あたりのキロバイト単位の最大レート指定することにより、I/O 帯域幅の使用を制御できます。デフォルトでは、このオプションは 0 に設定されています。つまり、帯域幅制限がないことを意味します。

特定の表領域の I/O ワークロードを管理する必要がある場合は、`tablespace_bandwidth_limit` オプショングローバルまたはサーバーごとを使用して、個々の表領域の制限を設定します。

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

このオプションは、テーブルスペース名と帯域幅制限のペア 1 秒あたりのキロバイト単位のコンマ区切りリストを取得します。

サーバーをバックアップするときに、Barman はこのオプションにリストされているテーブルスペースをチェックします。一致するテーブルスペースが見つかった場合、指定された帯域幅制限が適用されます。一致するものが見つからない場合、サーバーのデフォルトの帯域幅制限が使用されます。

重要

`bandwidth_limit` オプションは、`rsync` および `postgres` バックアップ方法で使用できますが、`tablespace_bandwidth_limit` オプションは `rsync` を使用する場合にのみ適用されます。

7.5 ネットワーク圧縮

ネットワーク圧縮を使用することにより、ネットワーク上で転送されるデータのサイズを削減できます。これは、`network_compression` オプショングローバルまたはサーバーごとを使用して有効にできます。

```
network_compression = true | false
```

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。

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

7.6 バックアップ圧縮

Barman は、`pg_basebackup` ツールを使用したバックアップ圧縮をサポートしています。この機能は、`backup_compression` オプショングローバルまたはサーバーごとを使用して有効にできます。

重要

`backup_compression` オプションは、ここで説明する他のオプションと同様に、`postgres` バックアップ方法でのみ使用できます。

7.6.1 圧縮アルゴリズム

`backup_compression` オプションを設定すると、指定されたアルゴリズムを使用してバックアップが圧縮されます。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、zstd、および none 非圧縮のバックアップになります。

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

Barman、選択した圧縮アルゴリズムに対応する CLI ユーティリティを Barman サーバーと Postgres サーバーの両方にインストールする必要があります。これらのユーティリティは、Debian、Ubuntu、RedHat、CentOS、および SLES システムに `gzip`、`lz4`、および `zstd` という名前のシステムパッケージを介してインストールできます。

- Ubuntu 18.04 bionic では、`lz4` ユーティリティは `liblz4-tool` パッケージで利用できます。
- `lz4` および `zstd` は、Postgres 15 以降でサポートされています。

重要

`backup_compression` を使用する場合、`staging_path` および `staging_location` を設定して圧縮バックアップのリカバリーを有効にする必要があります。詳細については、`backup_compression` セクションを参照してください。

7.6.2 コンプレッションワーカー

`backup_compression_workers` オプションデフォルトは `0` を設定することにより、マルチプルのスレッドを使用して圧縮を高速化できます。

```
backup_compression_workers = 2
```

注釈

このオプションは、`zstd` 圧縮でのみ使用できます。`zstd` バージョンは 1.5.0 以上、またはマルチスレッドが有効になっている場合は 1.4.4 以上である必要があります。

7.6.3 圧縮レベル

`backup_compression_level` オプションで圧縮レベルを指定します。これは、選択した圧縮アルゴリズムでサポートされている整値である必要があります。指定しない場合、アルゴリズムのデフォルト値が使用されます。

- `none` 圧縮の場合、`backup_compression_level` を `0` に設定する必要があります。
- 使用可能なレベルとデフォルト値は、選択した圧縮アルゴリズムによって異なります。詳細については、*backup configuration options* セクションを確認してください。
- 15 より前の Postgres バージョンの場合、`gzip` は、デフォルトの圧縮レベルを使用する `backup_compression_level = 0` のみをサポートしています。

7.6.4 圧縮場所

Postgres 15 以降では、圧縮が発生する場所 `server` または `client` で選択できます。`backup_compression_location` オプションを設定します。

```
backup_compression_location = server | client
```

- `server` 圧縮は Postgres サーバーで発生し、ネットワーク帯域幅は削減されますが、サーバーのワークロードが増加します。
- `client` 圧縮はクライアントサイドで `pg_basebackup` によって処理されます。

`backup_compression_format` を使用してバックアップ形式を指定することもできます。

```
backup_compression_format = plain | tar
```

- `plain pg_basebackup` は、ディスクに書き込む前にデータを解凍します。
- `tar` バックアップは圧縮された `tar` ボールとして書き込まれますデフォルト。

注釈

`backup_compression_location = server` および `backup_compression_format = plain` を設定すると、ファイルがサーバー側で圧縮され、クライアント側で解凍されることを考慮して、ネットワーク使用量を削減できます。これは、ネットワーク帯域幅は制限されているが CPU は制限されておらず、バックアップを非圧縮で保存する必要がある場合に役立ちます。

選択した `backup_compression` および `backup_compression_format` によっては、Postgres サーバーと Barman サーバーの両方に追加ツールをインストールする必要がある場合があります。

以下の表を参照して、構成に適切なツールを選択します。

<code>backup_compression</code>	<code>backup_compression_</code>	Postgres	Barman
<code>gzip</code>	プレーン	タール	なし
<code>gzip</code>	タール	タール	タール
<code>lz4</code>	プレーン	<code>tar</code> 、 <code>lz4</code>	なし
<code>lz4</code>	タール	<code>tar</code> 、 <code>lz4</code>	<code>tar</code> 、 <code>lz4</code>
<code>zstd</code>	プレーン	<code>tar</code> 、 <code>zstd</code>	なし
<code>zstd</code>	タール	<code>tar</code> 、 <code>zstd</code>	<code>tar</code> 、 <code>zstd</code>
なし	タール	タール	タール

7.7 バックアップの暗号化

Barman は、バックアップと WAL ファイルの両方の暗号化をサポートしています。この機能は、`encryption` オプショングローバルまたはサーバーごとを使用して有効にできます。

7.7.1 要件

バックアップの現在の暗号化実装は、tar 形式でバックアップを取得する `pg_basebackup` 機能に依存しています。これを行うには、次のように構成を設定する必要があります。

- `backup_method = postgres`
- `backup_compression = <compression_method>` 圧縮なしの場合は `none`
- `backup_compression_format = tar`

バックアップされた tar ファイルは、`pg_basebackup` が Barman サーバーディスクへの書き込みを終了するとすぐに暗号化されます。

7.7.2 暗号化方法

`encryption` オプションの設定は、ベースバックアップと WAL に使用される暗号化方法を指定します。現在、`gpg` および `none` 暗号化なしのみが受け入れられる値です。

注釈

WAL 暗号化の詳細については、[WAL 暗号化](#) を参照してください。

注釈

復号の詳細については、[暗号化されたバックアップの復元](#) を参照してください。

GPG

この方法は、構成ファイルで `encryption = gpg` を設定することにより有効にされます。

暗号化に `GPG` を使用するには、サーバーに `gpg` バージョン 2.1 以降をインストールする必要があります。また、事前に GPG キーペアを生成し、生成された公開キーの ID または受信者のメールアドレスで `encryption_key_id` オプションを構成する必要があります。対応するプライベートキーは GPG のキーリングに存在し、強力なパスフレーズで保護されている必要があります。

7.8 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。これは、追加のワークロードを生成する可能性があります。デフォルトでは、このチェックポイントは Postgres のワークロード制御設定に従って管理され、バックアップが遅延する場合があります。

`immediate_checkpoint` 構成オプションデフォルトは `false` を使用して、このデフォルトの動作を変更できます。

`immediate_checkpoint` が `true` に設定されている場合、Postgres はスロットルなしで最大速度でチェックポイントを実行し、できるだけ早くバックアップを開始できます。`barman backup` コマンドで次のオプションのいずれかを使用することにより、いつでもこの構成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` バックアップを開始する前にチェックポイントが完了するのを待機します。

7.9 ストリーミングバックアップ

Barman は、`pg_basebackup` とのストリーミング接続を使用して Postgres サーバーのバックアップを実行できます。

重要

`pg_basebackup` は Barman サーバーにインストールする必要があります。下位互換性があるため、`pg_basebackup` の最新バージョンを使用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、複数のバージョンをインストールし、指定できます。

ストリーミングバックアップを構成するには、`backup_method` を `postgres` に設定します。

```
backup_method = postgres
```

7.9.1 ブロックレベルのインクリメンタル バックアップ

このタイプのバックアップは、Postgres 17 で導入されたネイティブのインクリメンタル バックアップ機能を使用します。

ブロックレベルのインクリメンタル バックアップは、Postgres のページレベルでデータを重複排除します。これは、最後のバックアップ以降に変更されたページのみを保存する必要があることを意味し、特に頻繁に書き込みを行う大規模なデータベースの場合、より効率的です。

Barman でブロックレベルのインクリメンタルバックアップを実行するには、`backup` コマンドで `--incremental` オプションを使用します。重複排除のために `backup_method=postgres` で作成された以前のバックアップ完全またはインクリメンタルを参照するバックアップ ID またはショートカットを提供する必要があります。または、`last-full` または `latest-full` を使用して、カタログ内の最新の適格なフルバックアップを参照できます。

コマンドの例

```
barman backup --incremental BACKUP_ID SERVER_NAME
```

Barman でブロックレベルのインクリメンタルバックアップを使用するには、次のことを行う必要があります。

- Postgres 17 以降を使用してください。
- この機能は WAL 要約に依存しているため、最初の完全バックアップを取得する前にデータベースサーバーで `summarize_wal` を有効にする必要があります。
- `backup_method=postgres` を使用します。

i 注釈

圧縮バックアップは、現在 Barman のブロックレベルのインクリメンタルバックアップではサポートされていません。

ⓘ 重要

ブロックレベルのインクリメンタルバックアップの間で `data_checksums` を有効にする場合、新しい完全バックアップを取ることをお勧めします。発散チェックサム構成は、リカバリ中に問題を発生させる可能性があります。

7.10 SSH を介した Rsync を使用したバックアップ

Barman は、トランスポートメカニズムとして SSH を使用する Rsync を使用して Postgres サーバーのバックアップを実行できます。

rsync を使用してバックアップを構成するには、Barman サーバー構成ファイルに次のパラメーターを含めます。

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

ここで、`backup_method` は rsync バックアップ方法をアクティブにし、`ssh_command` は Barman サーバーから Postgres サーバーへの SSH 接続の詳細を指定します。

i 注釈

Barman 3.11 以降、rsync ベースのバックアップにキープアライブメカニズムが使用されます。このメカニズムは、libpq 接続を介して単純な `SELECT 1` クエリーを送信して、アイドル接続によるファイアウォールまたはルーターの切断を防ぎます。 `keepalive_interval` 構成オプションを使用して、このメカニズムを制御または無効にできます。

7.10.1 ファイルレベルのインクリメンタル バックアップ

ファイルレベルのインクリメンタルバックアップは rsync またはハードリンクに依存するため、オペレーティングシステムとバックアップデータが保存されるファイルシステムの両方がこれらの機能をサポートする必要があります。

基本的なアイデアは、後続のベースバックアップ中に、最後のバックアップから変更されていないファイルが共有され、ディスク領域を節約するということです。これは、`:term: VLDB`` および読み取り専用の履歴テーブルの割合が高い場合に特に役立ちます。

Barman コマンドを管理する `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 rsync インクリメンタルバックアップを有効にできます。3つの値を受け入れます。

- `off` 標準のフルバックアップデフォルト。

- **link** 最後のバックアップを再利用し、変更されていないファイルのハードリンクを作成するファイルレベルのインクリメンタルバックアップ、バックアップ領域と時間の両方を削減します。
- **copy** 最後のバックアップを再利用し、変更されていないファイルのコピーを作成するファイルレベルのインクリメンタルバックアップ。バックアップ時間は削減されますが、領域は削減されません。

通常、次のように `reuse_backup` を `link` に設定します。

```
reuse_backup = link
```

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

Barman コマンドの実行時に、`--reuse-backup` ランタイムオプションでこの設定をオーバーライドできます。たとえば、1 回限りのインクリメンタルバックアップを実行するには、次を使用します。

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

注釈

ブロックレベルのインクリメンタルバックアップとは異なり、rsync ファイルレベルのインクリメンタルバックアップは自己完結型です。親バックアップが削除されても、他のバックアップの整合性は影響を受けません。rsync バックアップの重複排除はハードリンクを使用します。つまり、再利用されたバックアップが削除されるときに、新しい完全バックアップを作成する必要はありません。共有ファイルは、これらのファイルを使用した最後のバックアップも削除されるまでディスクに残ります。さらに、初期バックアップに `reuse_backup = link` または `reuse_backup = copy` を使用しても、リンクまたはコピーする既存のファイルが存在しないため完全バックアップとして扱われるため、効果はありません。

7.11 ローカルで Rsync を使用したバックアップ

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

このアーキテクチャは、Barman チームによって承認されていません。Postgres のビジネス継続性エクスペリエンスを強化し、`:term:RPO` および `:term:RTO` の点でより良い結果を得るには、引き続きウィットネス サーバのように機能する Barman のリモートインストールを使用したシェアードナッシングアーキテクチャをお勧めします。レプリケーションとモニタリングのためのサーバー。

ローカルバックアップの唯一の要件は、Barman が Postgres 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 を `barman` ユーザーの下にインストールすることを考えると、このユースケースでは以下を含む手動インストール手順が必要です。

- cron 構成
- ログ構成 `logrotate` を含む

 ヒント

`barman.conf` ファイルの `barman_user` オプションを使用して、Barman に別のユーザーを設定します。権限の問題を回避するには、新しいユーザーがアクセスできるパスに応じて、ホーム、ログ、およびロックディレクトリを変更することを忘れないでください。詳細については、`barman_user` を参照してください。

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。この機能は本質的に同等の `rsync` と同じであり、代わりに SSH に依存し、リモートで動作します。`local-rsync` では、ファイルコピーはローカルで `rsync` コマンドを発行して実行されますこのため、Barman は Postgres と同じユーザーで実行する必要があります。

7.12 スタンバイのコンカレント バックアップ

スタンバイサーバーからバックアップを実行する場合、次の構成オプションがスタンバイをポイントするように設定されていることを確認します。

- `conninfo`
- `streaming_conninfo backup_method = postgres` または `streaming_archiver = on` を使用する場合
- `ssh_command backup_method = rsync` を使用する場合
- `wal_conninfo conninfo` がスタンバイを指している場合、プライマリに接続します

`primary_conninfo` 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。Barman は `primary_conninfo` を使用して、プライマリで新しい WAL スイッチをトリガーし、自然な WAL スイッチを待たずにスタンバイからのコンカレントバックアップを完了できます。

 注釈

プライマリでの書き込みアクティビティが最小限または全くない期間にスタンバイをバックアップする場合、`primary_conninfo` を構成することが重要です。

Barman 3.8.0 以降では、`primary_conninfo` が構成されている場合、`primary_checkpoint_timeout` オプションを設定することもできます。これは、Barman がプライマリにチェックポイントを強制するまでの、新しい WAL ファイルの最大待機時間秒単位を指定します。このタイムアウトは、プライマリに設定された `archive_timeout` 値を超える必要があります。

`primary_conninfo` が設定されていない場合、バックアップは続行されますが、最後にアーカイブされた WAL セグメントがバックアップに必要な最新の WAL より新しくなるまで、`stopbackup` ステージで一時停止します。

Barman では、WAL ファイルとバックアップデータが同じ Postgres クラスターから発生する必要があります。スタンバイがプライマリに昇格しても、既存のバックアップと WAL は有効なままです。ただし、Barman 構成を更新して、今後のバックアップと WAL の取得に新しいスタンバイを使用する必要があります。

i 注釈

Postgres クラスタでフェールオーバーが発生した場合、*Configuration Models* で Barman 構成を更新できます。

WAL は、WAL ストリーミングまたは WAL アーカイブを介してスタンバイから取得できます。詳細については、*concepts* セクションを参照してください。WAL ストリーミングまたは WAL アーカイブでの作業を開始する場合は、*concepts* または *concepts* のクイックスタートセクションを参照してください。

i 注釈

Postgres 10 以前の場合、Barman はスタンバイで同時の WAL ストリーミングとアーカイブを処理できません。Postgres 10 以前の WAL はバイナリレベルで異なる場合があります、Barman での誤検出問題が発生する可能性があるため、他方を使用している場合は一方を無効にする必要があります。

7.13 外部構成ファイルの管理

Barman は、使用されるバックアップ方法に応じて、**外部構成ファイル<外部構成ファイル>**を異なる方法で処理します。``rsync`` メソッドでは、外部ファイルは PGDATA ディレクトリにコピーされます。ただし、`postgres` メソッドでは、外部ファイルはコピーされず、警告が発行されてこれらのファイルについてユーザーに通知します。

リカバリーの章の *Managing external configuration files* セクションを参照して、バックアップを復元するときに外部ファイルがどのように処理されるかを理解します。

💡 ヒント

`backup_method = postgres` の場合、Barman は PostgreSQL ホストへの SSH 接続を確立しないため、バックアップ後フックを構成し、`barman show-server` コマンドの出力を使用して、バックアップが完了した直後に外部構成ファイルを自分でバックアップすることができます。

7.14 バックアップに不変ストレージを使用する

Barman は、不変ストレージにバックアップを保存するように構成して、悪意のあるアクターまたは誤った削除から保護できます。このようなストレージは、**WORM** Write Once Read Many ストレージと呼ばれることもあります。

このタイプのストレージの主な使用例は、ランサムウェア攻撃からバックアップを保護することです。不変のストレージを使用することにより、特定の期間バックアップを削除または変更できません。

Barman が不変のバックアップを提供するには、バックアップと WAL ファイルのみを不変ストレージに配置し、復元不可能なデータを通常のストレージに残す必要があります。この方法により、Barman は、バックアップと WAL ファイルのメタデータに関する一時的な情報を維持できます。その情報は定期的な更新が必要です。

上記を踏まえて、不変のストレージにバックアップを保存するように Barman を構成するには、次の提案に従う必要があります。

- 次の 2 つのディレクトリのみが不変のストレージパスに保存されるように構成する必要があります。
 - `basebackups_directory` バックアップが保存されるディレクトリ。
 - `wal_directory` WAL ファイルが保存されるディレクトリ。
- 他のすべてのディレクトリは、Barman の内部プロセスによって使用され、クラスターの復元に重要なデータを保持しないため、通常のストレージパスに保存する必要があります。これは、グローバル構成で通常のストレージをポイントするように `barman_home` オプションを構成するか、サーバーセクションで `barman_home` オプションを構成することにより実現できます。これには、以前の箇条書きのオプションをそれに応じて設定する必要があります。
- WAL ファイルカタログは、通常のストレージパスに保存する必要があります。これは、通常のストレージをポイントするように `xlogdb_directory` オプションを構成することにより実現できます。
- `staging_path` および `staging_location` オプション詳細については `staging_path` を参照してください。インクリメンタルまたは圧縮または暗号化されたバックアップの復元に使用されるパスも、通常のストレージに存在する必要があります。
- 保持ポリシーは、少なくともバックアップファイルが不変である完全な期間をカバーする必要があります。これは、server セクションの `retention_policy` オプションを不変ストレージの不変期間よりも大きい値に設定することにより実現できます。これは、不変期間が終了する前にバックアップが削除されないようにするためです。

バックアップの不変性を構成するには、有効にする必要がある `worm_mode` オプションがあります。これにより、Barman は、バックアップと WAL ファイルが `WORM` 環境に保存されている場合に問題のあるプロセスをスキップできます。

注釈

`xlogdb` ファイルを再配置するオプションは Barman 3.12 に含まれていました。詳細については、`xlogdb` を参照してください。

7.14.1 電流制限

Barman の不変バックアップサポートの現在の実装には、次の制限があります。

- `WORM` 環境には猶予期間が必要です。猶予期間は、`WORM` 制限が有効になる前にデータを変更または削除できる事前定義されたウィンドウを提供します。この要件は、Barman が名前の変更を利用して WAL を外部パーティションに安全にコピーするため、ファイルが既に `WORM` 状態に入っている場合、これは失敗するために存在します。

一般に、Barman が必要な操作を完了するための十分な時間を提供するため、少なくとも 15 分の猶予期間をお勧めします。

注釈

バックアップの暗号化も有効になっている場合、猶予期間は暗号化の実行に必要な時間をカバーするのに十分な長さである必要があります特にバックアップにテーブルスペースが含まれ、複数の tar ボールが生成される場合。

これは、暗号化がバックアッププロセスの最後にのみ発生するため、`pg_basebackup` が完了した後です。暗号化はインプレースで実行できないため、各 tar ファイルは個別に暗号化され、完了すると非暗号化バージョンが削除されます。

これらの制約を考慮すると、ユーザーは不変のバックアップサポートを有効にする前に、現在の実装が要件を満たしているかどうかを評価する必要があります。

7.15 クラウド スナップショット 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利用して、特定のクラウド環境に展開された Postgres サーバーのバックアップを実行できます。この設定では、Postgres ファイルのバックアップはクラウドに保存されているボリュームスナップショットとして表され、Barman は書き込み先行ログ 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。バックアップデータはクラウドに保存されているにもかかわらず、Barman は、rsync または postgres バックアップ方法で作成された従来のバックアップと同様にこれらのバックアップを管理します。

注釈

さらに、Postgres サーバーで `barman-cloud-backup` コマンドを直接使用することにより、Barman サーバーを使用せずにスナップショットバックアップを作成できます。このオプションを適切に使用方法の詳細については、`barman-cloud-backup` セクションを参照してください。

重要

`backup_method=snapshot` を使用する場合、次の構成オプションと同等のコマンド引数該当する場合は使用できません。

- `backup_compression`
- `bandwidth_limit --bwlimit`
- `parallel_jobs --jobs`
- `network_compression`
- `reuse_backup --reuse-backup`

スナップショットを使用してバックアップを構成するには、Barman サーバー構成ファイルに次のパラメーターを含めます。

```
backup_method = snapshot
snapshot_provider = CLOUD_PROVIDER
```

(次のページに続く)

(前のページからの続き)

```
snapshot_instance = INSTANCE_NAME
snapshot_disks = DISK_NAME1,DISK_NAME2
```

重要

`snapshot_disks` に、Postgres データを保存するすべてのディスクが含まれていることを確認します。リストされていないディスクに保存されているデータはバックアップされず、リカバリ中に使用できません。

7.15.1 要件と構成

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、展開が次の要件を満たしている必要があります。

1. Postgres は、サポートされているクラウドプロバイダーが提供するコンピューティングインスタンスで実行されている必要があります。
2. PGDATA やテーブルスペースデータを含むすべての重要なデータは、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する必要があります。
3. `findmnt` コマンドは、Postgres 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。構成管理システムまたはその他のメカニズムを使用して、これらのファイルを個別に管理する必要があります。

7.15.2 Google Cloud Platform

Barman で `GCP` のスナップショットバックアップを使用するには、以下を確認してください。

1. Python Libraries

Barman が使用する Python ディストリビューションの `google-cloud-compute` および `grpcio` ライブラリをインストールします。これらのライブラリはオプションであり、デフォルトでは含まれません。

`pip` を使用してインストールします。

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

注釈

`google-cloud-compute` ライブラリには、Python 3.7 以降が必要です。GCP スナップショットは、以前の Python バージョンと互換性がありません。

2. Disk Requirements

snapshot バックアップで使用されるディスクは、ゾーン永続ディスクである必要があります。現時点では、リージョン永続ディスクはサポートされていません。

3. Access Control

Barman には、特定の権限を持つサービスアカウントが必要です。このアカウントを Barman を実行しているコンピューティングインスタンスに接続するか、`GOOGLE_APPLICATION_CREDENTIALS` 環境変数を使用して資格情報ファイルを指定できます。

❗ 重要

サービスアカウントに以下にリストされている権限があることを確認します。

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

For provider specific credentials configurations, refer to the [Google authentication methods](#) and [service account impersonation](#).

4. Specific Configuration

フィールド `gcp_project` および `gcp_zone` は、GCP に固有の構成オプションです。

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

7.15.3 Microsoft Azure

Barman を使用して Azure でスナップショットバックアップを使用するには、以下を確認します。

1. Python Libraries

`azure-mgmt-compute` および `azure-identity` ライブラリは、Barman が使用する Python ディストリビューションで利用できる必要があります。これらのライブラリはオプションであり、デフォルトでは含まれません。

`pip` を使用してインストールします。

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

i 注釈

azure-mgmt-compute ライブラリには、Python 3.7 以降が必要です。Azure スナップショットは、以前の Python バージョンと互換性がありません。

2. Disk Requirements

スナップショットバックアップに関係するすべてのディスクは、データディスクとして VM インスタンスに接続されている管理ディスクである必要があります。

3. Access Control

Barman は、マネージド ID または CLI ログインを介して取得した資格情報を使用して Azure にアクセスする必要があります。

次の環境変数がサポートされています。AZURE_STORAGE_CONNECTION_STRING、AZURE_STORAGE_KEY および AZURE_STORAGE_SAS_TOKEN。Azure Active Directory を介して認証するために、`--credential` オプションを使用して、`default`、`azure-cli` または `managed-identity` 資格情報を指定することもできます。

ⓘ 重要

資格情報に以下にリストされている権限があることを確認します。

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

For provider specific credential configurations, refer to the [Azure environment variables configurations](#), [Identity Package](#) and [DefaultAzureCredential documentation](#).

4. Specific Configuration

フィールド `azure_subscription_id` および `azure_resource_group` は、Azure に固有の構成オプションです。

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

7.15.4 アマゾン ウェブ サービス

Barman で `AWS` のスナップショットバックアップを使用するには、以下を確認してください。

1. Python Libraries

boto3 ライブラリは、Barman が使用する Python ディストリビューションで利用できる必要があります。このライブラリはオプションであり、デフォルトでは含まれません。

pip を使用してインストールします。

```
pip3 install boto3
```

2. Disk Requirements

スナップショットバックアップに関係するすべてのディスクは、同じ VM インスタンスに接続されている非ルート EBS ボリュームである必要があります。

3. Access Control

Barman は AWS にアクセスする必要があるため、postgres ユーザーとして awscli ツールを使用して AWS 資格情報を構成する必要があります、AWS コンソールの IAM セクションで事前に作成する必要があるアクセスキーとシークレットキーを入力します。

重要

以下にリストされている権限があることを確認します。

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

プロバイダー固有の資格情報構成については、[AWS boto3 configurations](#) を参照してください。

4. Specific Configuration

フィールド `aws_region`、`aws_profile`、および `aws_wait_snapshots_timeout` は、AWS に固有の構成オプションです。

`aws_profile` は、資格情報ファイルの AWS プロファイルの名前です。使用しない場合、デフォルトのプロファイルが適用されます。資格情報ファイルが存在しない場合、資格情報は環境から取得されます。

`aws_region` は、AWS プロファイルで定義されたリージョンをオーバーライドします。

`aws_wait_snapshots_timeout` は、スナップショットが作成されるのを待機するタイムアウトです。デフォルトは 3600 秒です。

`snapshot_instance` または `snapshot_disks` を指定すると、Barman はインスタンス/ボリューム ID またはリソースの名前を受け入れます。名前を使用する場合、Barman は AWS に照会して、一致する Name タグを持つリソースを照会します。0 または複数の一致が見つかった場合、Barman はエラーを結果ます。

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

5. Ransomware Protection

ランサムウェア保護は、データを保護し、動作の安定性を維持するために不可欠です。Amazon EBS Snapshot Lock を使用すると、スナップショットが削除から保護され、ランサムウェア攻撃から保護する不変のバックアップを提供します。スナップショットをロックすることにより、不要な削除が防止され、侵害された場合に信頼できる復旧オプションが保証されます。Barman は、バックアップの作成時にスナップショットをロックすることにより、バックアップの不要な削除を防ぐことができます。

注釈

ロックされたバックアップを削除するには、最初に AWS コンソールでロックを手動で削除する必要があります。

バックアップの作成中にスナップショットをロックするには、次のオプションを構成する必要があります。

1. スナップショットロックモード **compliance** または **governance** のいずれかを選択します。
2. ロック期間または有効期限のいずれかを設定します両方ではありません。ロックの期間は 1~36,500 の範囲で日単位で指定されます。有効期限を選択する場合、YYYY-MM-DDTHH:MM:SS.sssZ 形式を使用して、スナップショット作成日時の少なくとも 1 日後である必要があります。
3. オプションで、クールオフ期間時間単位を 1~72 の範囲で設定します。このオプションは、ロックモードが **compliance** に設定されている場合にのみ適用されます。

```
aws_snapshot_lock_mode = compliance | governance
aws_snapshot_lock_duration = 1
aws_snapshot_lock_cool_off_period = 1
aws_snapshot_lock_expiration_date = "2024-10-07T21:53:00.606Z"
```

重要

以下にリストされている権限があることを確認します。

- `ec2:LockSnapshot`

AWS スナップショットロックの概念については、[Amazon EBS snapshot lock concepts](#) を参照してください。

7.15.5 バックアッププロセス

次に、スナップショットバックアッププロセスの概要を示します。

1. Barman は、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。
各バックアップの前と `barman check` コマンドの実行中に、次のチェックが実行されます。

- `snapshot_instance` およびプロバイダ固有の引数で指定されたコンピューティングインスタンスが存在します。
 - `snapshot_disks` にリストされているディスクは存在します。
 - `snapshot_disks` にリストされているディスクは、`snapshot_instance` に接続されています。
 - `snapshot_disks` にリストされているディスクは、`snapshot_instance` にマウントされています。
2. Barman は、Postgres バックアップ 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 は、指定された各ディスクのスナップショットを作成するために使用されます。Barman は、各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状態に到達するまで待機してから、次のディスクに進みます。
 4. 各ディスクのデバイス名、各ディスクのマウントポイントとオプションなどの追加プロバイダー固有の詳細は、バックアップメタデータに記録されます。

7.15.6 メタデータ

コードとしてのインフラストラクチャ、アドホック自動化、または手動でリカバリディスクとインスタンスをプロビジョニングするかどうかに関係なく、Barman を使用して特定のバックアップに必要なスナップショットを特定する必要があります。これは、`barman show-backup` コマンドで行うことができます。このコマンドは、バックアップに含まれる各スナップショットの詳細を提供します。

例

```
Backup 20240813T200506:
  Server Name      : snapshot
  System Id       : 7402620047885836080
  Status          : DONE
  PostgreSQL Version : 160004
  PGDATA directory : /opt/postgres/data
  Estimated Cluster Size : 22.7 MiB

Server information:
  Checksums      : on

Snapshot information:
  provider      : aws
  account_id    : 714574844897
  region        : sa-east-1

  device_name    : /dev/sdf
  snapshot_id    : snap-0d2288b4f30e3f9e3
  snapshot_name  : Barman_AWS:1:/dev/sdf-20240813t200506
  Mount point    : /opt/postgres
  Mount options  : rw,noatime,seclabel
```

(次のページに続く)

(前のページからの続き)

Base backup information:

```

Backup Method      : snapshot-concurrent
Backup Size       : 1.0 KiB (16.0 MiB with WALs)
WAL Size          : 16.0 MiB
Timeline          : 1
Begin WAL         : 0000000100000000000000001A
End WAL           : 0000000100000000000000001A
Number of WALs    : 1
Begin time        : 2024-08-14 16:21:50.820618+00:00
End time          : 2024-08-14 16:22:38.264726+00:00
Copy time         : 47 seconds
Estimated throughput : 22 B/s
Begin Offset      : 40
End Offset        : 312
Begin LSN         : 0/1A000028
End LSN           : 0/1A000138

```

WAL information:

```

Number of files    : 1
Disk usage         : 16.0 MiB
WAL rate           : 5048.32/hour
Last available     : 0000000100000000000000001B

```

Catalog information:

```

Retention Policy   : not enforced
Previous Backup    : - (this is the oldest base backup)
Next Backup        : - (this is the latest base backup)

```

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

```

{
  "snapshots_info": {
    "provider": "gcp",
    "provider_info": {
      "project": "project_id"
    },
  },
  "snapshots": [
    {
      "mount": {
        "mount_options": "rw,noatime",
        "mount_point": "/opt/postgres"
      },
      "provider": {
        "device_name": "pgdata",

```

(次のページに続く)

(前のページからの続き)

```

    "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",
    }
  }
]
}
}

```

snapshots_info/provider_info および snapshots_info/snapshots/*/provider にあるメタデータは、以降のセクションで詳細に説明するように、クラウドプロバイダーによって異なります。

GCP

snapshots_info/provider_info

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

snapshots_info/snapshots/*/provider

- **device_name** スナップショットのソースディスクがバックアップ時にバックアップ VM に接続された短いデバイス名。
- **snapshot_name** スナップショットの名前。
- **snapshot_project** スナップショットを所有する GCP プロジェクト ID。

Azure

snapshots_info/provider_info

- **subscription_id** バックアップとリカバリーに関係するリソースを所有する Azure サブスクリプション ID。
- **resource_group** バックアップに関係するリソースが属する Azure リソースグループ。

snapshots_info/snapshots/*/provider

- **location** スナップショットが取得されたディスクの Azure の場所。
- **lun** バックアップ時にスナップショットが取得されたディスクを識別する LUN。

- `snapshot_name` スナップショットの名前。

AWS

`snapshots_info/provider_info`

- `account_id` バックアップの作成に使用されるリソースを所有する AWS アカウントの ID。
- `region` バックアップに関係するリソースが存在する AWS リージョン。

`snapshots_info/snapshots/*/provider`

- `device_name` バックアップ時にバックアップ VM でソースディスクがマッピングされたデバイス。
- `snapshot_id` AWS によって割り当てられたスナップショットの ID。
- `snapshot_name` スナップショットの名前。

第8章 WAL アーカイブ

Barman は、WAL アーカイブに関する追加機能も提供します。

8.1 WAL 圧縮

Barman は、Barman の WAL アーカイブに入るときに WAL ファイルを圧縮できます。このプロセスは、`barman cron` によって自動的に処理されるか、`barman archive-wal` コマンドが手動で実行されたときに処理されます。

圧縮は、構成ファイルの `compression` オプションを介して有効にされます。オプションでは、次のいずれかの値を使用できます。

- `lz4` LZ4 圧縮用 `lz4` ライブラリをインストールする必要があります。
- `xz` XZ 圧縮用 Python の内部圧縮ライブラリを使用します。
- `zstd` Zstandard 圧縮用 `zstandard` ライブラリをインストールする必要があります。
- `gzip` Gzip 圧縮用 Python の内部圧縮ライブラリを使用します。
- `pygzip` Gzip 圧縮用現在は `gzip` と同じです非推奨。
- `pigz` Pigz 圧縮用 `pigz` ユーティリティが必要です。
- `bzip2` Bzip2 圧縮用 Python の内部圧縮ライブラリを使用します。
- `pybzip2` Bzip2 圧縮用現在は `bzip2` と同じです非推奨。
- `custom` カスタム圧縮用。同様に次のオプションを設定する必要があります
`custom_compression_filter`、`custom_decompression_filter`、`custom_compression_magic`
。詳細については、`custom`を確認してください。非推奨

バージョン 3.16 で非推奨: `pygzip` および `pybzip2` 圧縮オプションは非推奨であり、将来のリリースでは削除される予定です。代わりに、同等の `gzip` および `bzip2` を使用してください。

バージョン 3.17 で非推奨: `custom` 圧縮オプションと関連する構成オプション `custom_compression_filter`、`custom_decompression_filter`、および `custom_compression_magic` は非推奨であり、将来のリリースでは削除されます。代わりに、ビルトイン圧縮アルゴリズムのいずれかを使用します。

注釈

`pigz` を使用する場合 Barman は、圧縮のための新しいプロセスをフォークします。

重要

Barman は、各 WAL ファイルが圧縮された方法や解凍方法に関するメタデータを保存しません。これは、主にサーバーの現在の構成に依存します。したがって、`compression = custom` を使用する場合、`custom_decompression_filter` がアーカイブで現在使用可能な WAL との互換性を維持していることを確認する必要があります。たとえば、`custom_decompression_filter` が途中で別の圧縮アルゴリズムに変更された場合、Barman は、復元時に、以前の圧縮アルゴリズムでアーカイブされた古い WAL を解凍できません。

`compression_level` 構成を使用して圧縮レベルを指定することもできます。有効な値は、選択したアルゴリズムまたは事前定義されたラベルのいずれか `low`、`medium`、および `high` のサポートされている範囲内の整数です。各アルゴリズムの範囲と、各事前定義されたラベルがマップするレベルは `compression_level` で見つかります。

`barman-wal-archive` を使用して `archive_command` を介してアーカイブする場合、選択したアルゴリズムとレベルで `--compression` および `--compression-level` オプションを使用して圧縮を構成することもできます。このような場合、圧縮は WAL が Barman サーバーに送信される前に、クライアント側で実行されます。 `archive_command` を参照してください。

注釈

`barman-wal-archive` で圧縮が有効になっている場合、Barman サーバーで構成された圧縮設定が異なる場合、それが優先されます。

重要

`barman-wal-archive` で圧縮が有効になっている場合、ファイルが Barman に送信される前に、クライアント側で実行されます。圧縮率を高くすると WAL の SHIPPING が遅延し、WAL ファイルが蓄積される可能性があるため、データベースサーバーの負荷と選択した圧縮アルゴリズムとレベルに注意してください。

8.2 WAL 暗号化

Barman は、WAL ファイルが Barman の WAL アーカイブに入るときに暗号化できます。このプロセスは、`barman cron` によって自動的に処理されるか、`barman archive-wal` コマンドが手動で実行されたときに処理されます。

WAL とバックアップの暗号化は両方とも、構成ファイルの `encryption` オプションを介して有効になります。現在、`gpg` および `none` 暗号化なしのみが値として受け入れられます。暗号化の構成方法に関する詳細なガイドは、`encryption` で利用できます。

8.3 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、`RPO` をゼロに削減することもできます。

`RPO` ゼロを使用してシナリオを構成するには、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 サーバーを構成し、receive-wal プロセスを Postgres サーバーへの同期接続として構成する必要があります。

最初に、show-servers コマンドを使用して Barman require-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 show-servers pg | grep streaming_archiver_name
```

出力

```
streaming_archiver_name: barman_receive_wal
```

注釈

accept-wal プロセスを開始するときに Barman が使用するアプリケーション名は、streaming_archiver_name 構成オプションで構成されます。このオプションのデフォルト値は barman_receive_wal です。

次に、アプリケーション名を postgresql.conf ファイルの synchronous_standby_names パラメーターに追加する必要があります。

```
synchronous_standby_names = 'barman_receive_wal'
```

重要

このトピックの詳細については、Barman:term: *RPO* zero adds more security to your backups and gives you more recovery options. However, it should not be considered as a substitution of a real Postgres replica. Please read the [official Postgres documentation about "Synchronous Replication"](#)。

変更を有効にするには、Postgres サーバー構成をリロード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、barman replication-status コマンドは、同期ストリーミングクライアントとして receive-wal プロセスを表示するはずです。

```
barman replication-status pg
```

出力

```
Status of streaming clients for server 'pg':
Current xlog location on master: 0/90000098
Number of streaming clients: 1
```

(次のページに続く)

(前のページからの続き)

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)
```

第9章 カタログ情報

バックアップカタログは、Barman ノード上のすべてのサーバーとバックアップを追跡する包括的なレコードです。サーバーは、Barman によって追跡およびバックアップされている Postgres データベースシステムであることに注意してください。各サーバーは Barman ノード内で構成され、データが定期的にバックアップされ、必要に応じて復元できるようにします。Barman ノードは、これらの Postgres サーバーのバックアップとリカバリプロセスを管理する Barman ソフトウェアをホストする別のサーバーです。構成されたサーバーと通信して、バックアップを実行、安全に保存し、すべてのバックアップアクティビティの詳細なカタログを維持します。

バックアップカタログは、効果的なバックアップ管理とリカバリ操作に不可欠であり、Barman ツールの重要なコンポーネントとなる統一インターフェイスを提供します。すべてのバックアップアクティビティの包括的なビューと正確な制御を提供することにより、バックアップとリカバリタスクを組織、監視、および実行において中心的な役割を果たします。この合理化されたアプローチにより、効率が向上し、管理が簡素化され、シームレスな復旧プロセスが保証されます。

グローバル構成オプション `barman_home` は、Barman が複数のサーバーのバックアップを保存および管理するディレクトリを指定します。デフォルトでは、このディレクトリは `/var/lib/barman` に設定されますが、`/etc/barman.conf` にあるグローバル構成ファイルを変更することによりカスタマイズできます。このディレクトリ内では、各サーバーからのバックアップは個別のサブディレクトリに編成され、各サーバーのバックアップは `<barman_home>/<SERVER_NAME>/base` の下にあります。この構造は、バックアップを整理し、簡単にアクセスできるようにします。

9.1 目的

すべての Postgres サーバーとバックアップ関連の情報を追跡する集中リポジトリとして機能します。

ここで、それが果たす重要な役割をいくつか示します。

- **Backup Metadata Storage** バックアップの開始時間と終了時間、ステータス、取得された特定の Postgres インスタンス、その他の多くのメトリックなど、各バックアップに関するメタデータを保存します。このメタデータは、バックアップライフサイクルの追跡と管理に役立ちます。
- **Backup Management** カタログは、バックアップを効率的に編成および管理する方法を提供します。詳細な記録を保持することにより、使用可能なバックアップのリスト、バックアップのステータスの確認、最新または特定のバックアップの特定、保持ポリシーの適用などの操作が簡素化されます。
- **Restore Operations** リストア操作中に、カタログは、使用可能なバックアップとその詳細をすばやく特定するのに役立ちます。これにより、バックアップとそれに関連するアーカイブログに関する必要な情報を提供することにより、効率的な復元が促進されます。
- **Ease of Use** 複数のバックアップとそれに関連するメタデータを管理するプロセスを簡素化し、管理者が大量のバックアップと複雑なバックアップ戦略を処理しやすくします。

9.2 使用法

Barman は、Postgres バックアップを管理し、バックアップカタログと対話するための簡単なターミナルインターフェイスを提供します。このインターフェイスは、サーバー管理とバックアップ操作の両方のためのさまざまなサブコマンドを提供します。すべての Barman サブコマンドは、`list-backups` および `show-backup` の 2 つの重要なコマンドを含む、*sub commands* セクションにあります。これらのコマンドを例とともに以下に示します。

9.2.1 list-backups

サーバーの使用可能なバックアップを表示します。このコマンドは、バックアップ ID やバックアップタイプなどの最小限だが重要な情報を含むバックアップのリストを取得するのに役立ちます。

例

```
svr_pg17 20240901T103000 - F - Mon Nov  2 15:12:03 2024 - Size: 820.0 MiB - WAL Size: 22 MiB
```

- `svr_pg17` はサーバー名です。
- `20240901T103000` はバックアップ ID です。
- `F` は、バックアップタイプのラベルです。
- `Nov 1 11:12:03 2024` は、バックアップ操作が終了した日時です。
- `Size: 820.0 MiB` は、バックアップのサイズです。
- `WAL Size: 22 MiB` は、このバックアップに関連するすべての WAL ファイルのサイズです。

注釈

バックアップタイプラベルは、フルバックアップの場合は `F`、ブロックレベルのインクリメンタルバックアップの場合は `I` です。rsync バックアップの場合は `R`、クラウドスナップショットバックアップの場合は `S`。

9.2.2 show-backup

特定のバックアップに関する詳細情報を表示します。例、ブロックレベルのインクリメンタルバックアップ

```
Backup 20240902T130000:
  Server Name      : prod_pg17
  Status           : DONE
  PostgreSQL Version : 170000
  PGDATA directory : /var/lib/pgsql/17/data
  Estimated Cluster Size : 244.7 GiB

  Server information:
    Checksums       : on
```

(次のページに続く)

(前のページからの続き)

```
WAL summarizer      : on
```

Base backup information:

```
Backup Method       : postgres
Backup Type        : incremental
Backup Size        : 3.4 GiB (36.7 GiB with WALs)
WAL Size           : 32.3 GiB
Resources saved    : 241.3 GiB (98.61%)
Timeline           : 1
Begin WAL          : 00000000100000CFD0000000AD
End WAL            : 00000000100000D0D000000008
Number of WALs     : 3932
WAL compression ratio: 79.51%
Begin time         : 2024-09-02 13:00:01.633925+00:00
End time           : 2024-09-03 10:27:06.522846+00:00
Copy time          : 1 second
Estimated throughput : 2.0 MiB/s
Begin Offset       : 1575048
End Offset         : 13853016
Begin XLOG         : CFD/AD180888
End XLOG           : D0D/8D36158
```

WAL information:

```
Number of files     : 35039
Disk usage          : 121.5 GiB
WAL rate            : 275.50/hour
Compression ratio   : 77.81%
Last available      : 00000000100000D950000000E7
```

Catalog information:

```
Retention Policy    : not enforced
Previous Backup     : 20240902T120001
Next Backup         : - (this is the latest base backup)
Root Backup         : 20240801T015504
Parent Backup       : 20240831T016504
Backup chain size   : 3
Children Backup(s)  : 20240903T018515,20240903T019515
```

注釈

`show-backup` コマンドの出力は、Postgres サーバーのバージョンとバックアップの種類によって異なる場合があります。

- フィールド `Root Backup`、`Parent Backup`、`Backup chain size` および `Children Backup(s)` は、Postgres 17 以降で `backup_method=postgres` で取得されたブロックレベルのインクリメン

タルバックアップにのみ関連します。これらのフィールドは、他の種類のバックアップまたは古い Postgres バージョンでは表示されません。

- **show-backup** コマンドは、バックアップメタデータに依存します。バックアップが Barman バージョン 3.10 以前で作成された場合、Postgres 17 のブロックレベルのインクリメンタルバックアップに関連するフィールドなど、バージョン 3.11 で導入されたフィールドは含まれません。
- フィールド **Resource Saved** は、rsync およびインクリメンタルバックアップで使用でき、**Snapshot Information** はスナップショットバックアップでのみ使用できます。
- バックアップが圧縮されている場合、使用された圧縮アルゴリズムを含む **Backup Compression** フィールドが出力に表示されます。
- バックアップが暗号化されている場合、**Encryption** フィールドが出力に表示され、使用された暗号化方法が含まれます。
- フィールド **Backup Type** の可能な値は次のとおりです。
 - **rsync** rsync で取得したバックアップの場合。
 - **full pg_basebackup** で取得したフルバックアップの場合。
 - **incremental pg_basebackup** で取得したインクリメンタルバックアップ用。
 - **snapshot** クラウドで取得されたスナップショットベースのバックアップ用。

第10章 リカバリー

resume コマンドは、backup コマンドで作成されたバックアップから Postgres サーバー全体を復元するために使用されます。バックアップを作成するときに、バックアップを一意に識別する `backup_id` が割り当てられます。resume コマンドを使用するには、次を実行します。

```
barman restore [OPTIONS] SERVER_NAME BACKUP_ID DESTINATION_PATH
```

`backup_id` を `auto` として指定することにより、バックアップを復元できます。この場合、Barman はカタログから最適なバックアップを自動的に選択します。リカバリターゲットが提供されない場合、デフォルトで最新の使用可能なバックアップが選択されます。復旧ターゲットが指定されている場合、Barman はそれを使用して、復旧基準に基づいて適切なバックアップを特定します。

- `target_time` Barman は、`target_time`、たとえば `2025-01-21 10:00:00` までで使用可能な最新のバックアップを取得します。
- `target_lsn` Barman は、`target_lsn`、たとえば `3/64000000` までで使用可能な最新のバックアップを取得します。
- `target_tli` Barman は、タイムライン `target_tli`、たとえば `2` から利用可能な最新のバックアップを取得します。
- `target_tli`、他の2つのターゲットのいずれかと組み合わせた場合 Barman は、`target_tli` に属する `target_lsn` または `target_time` までで利用可能な最新のバックアップを取得します `2025-01-21 10:00:00` タイムライン `2` から `2025-01-21 10:00:00`。

注釈

- Barman のリカバリーの概念をより明確に理解するためには、[リストアとリカバリー](#) を参照してください。
- バックアップは指定されたディレクトリに復元され、Postgres インスタンスを起動する準備が整います。
- `list-backups` コマンドを使用して、必要な特定のバックアップ ID を見つけます。
- Barman は、PGDATA 内のシンボリックリンクを追跡しませんテーブルスペースを除きます。シンボリックリンクを管理し、ディザスターリカバリー計画に含めます。
- `backup_id` を `auto` として指定する場合、Barman は `backup_id` を考慮しません。選択したバックアップが特定の引数を必要とするか、特定のオプションをサポートしていない場合、復元は失敗する場合があります。このような場合、それに応じて `restore` コマンドを調整するのに役立つエラーメッセージを受け取ります。たとえば、スナップショットバックアップを復元する場合は、`--snapshot-recovery-instance` が必要です。

10.1 ローカルリカバリー

`--remote-ssh-command` が存在しない場合、Barman はバックアップをローカルに復元します。DESTINATION_PATH で指定されたパス。

すべてのファイルとディレクトリは、`barman` ユーザーが所有します。したがって、Postgres を起動してリカバリーを実行するときは、2つのオプションがあります。

1. `barman` ユーザーとして Postgres を起動します。この場合、`--get-wal` を使用すると、`restore_command` は `sudo` を必要とせずに `get-wal` を呼び出すことができます。
2. `postgres` ユーザーとして Postgres を起動します。この場合、サーバーを起動する前に、復元されたすべてのファイルとディレクトリこれには PGDATA およびテーブルスペースを含むの所有権を `postgres` ユーザーに変更する必要があります。`--get-wal` を使用する場合、`restore_command` は `sudo` を使用して、`barman` ユーザーとして `get-wal` を実行する必要があります。

WAL リストアの設定の詳細については、[Fetching WALs](#) を確認してください。

10.2 リモートリカバリー

`--remote-ssh-command` `COMMAND` を使用して、SSH を介してリモートサーバーでリカバリーを実行します。リモート復旧のターゲットノードで `postgres` ユーザーを使用することをお勧めします。

Known Limitations

- `get-wal` が指定されない限り、システムの一時ディレクトリに少なくとも 4GB の空き領域が必要です。
- SSH は公開キー認証を使用する必要があります
- リモートユーザーは、必要な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ベースバックアップと WAL ファイル用の十分な空き領域があることを確認します。

10.3 デルタリカバリー

`--delta-restore` を使用して、`barman` がカスタムテーブルスペースとともに既存の宛先ディレクトリにバックアップを上書きして復元できるようにします。

もう1つのオプションは、リカバリ操作の前にグローバル/サーバーレベルで `recovery_options` 構成内に `delta-restore` を含めて、`--delta-restore` フラグを指定せずにデルタリカバリーを実行することです。

```
recovery_options = 'delta-restore'
```

`recovery_options` で `delta-restore` が指定されているが、特定のリカバリー中には必要ない場合、`barman restore` コマンドで `--no-delta-restore` オプションを使用して無効にできます。

i 注釈

`get-wal` を有効にしてデルタリカバリーを実行している場合、両方を `recovery_options = get-wal, delta-restore` として構成に追加できます。

ii 重要

`delta-restore` オプションは、次のシナリオでサポートされています `delta-restore`。

- **Local restore of plain backups** - インクリメンタル、圧縮、暗号化、またはマルチフォーマットのバックアップはサポートされていません。
- **Remote restore with staging_location=local** - `staging_location=remote` の使用はサポートされていません。

10.4 テーブルスペースのリマッピング

`--tablespace NAME:DIRECTORY` を使用して、表領域を新しい場所に再マップします。Barman は、宛先ディレクトリが存在しない場合、作成しようとします。

10.5 ポイント インタイム リカバリー

次のいずれかのオプションを使用してリカバリ `target` を指定します。

- `--target-time` 特定のタイムスタンプにリカバリーします。
- `--target-xid` 特定のトランザクション ID にリカバリーします。
- `--target-lsn` ログシーケンス番号にリカバリーします。
- `--target-name` 名前付けの復元ポイントにリカバリーします。
- `--target-tli` 特定のタイムラインにリカバリーします。
- `--target-immediate` 一貫した状態に到達したら、リカバリーを終了します。

i 注釈

- 復旧ターゲットは、バックアップ終了後の値である必要があります。バックアップ内のポイントインタイムにリカバリするには、以前のバックアップを使用します。
- `--target-time` のタイムスタンプで指定されない場合、タイムゾーンはデフォルトの Barman ホストになります。
- `--exclusive` を使用して、ターゲットの直前で停止するか、ターゲットを含めて停止するかを制御します。
- `--target-tli` は、ターゲットタイムラインを設定します。数値 ID またはショートカット値 `latest` または `current` を使用します。

- 少なくとも 1 つの `--target-*` オプションが指定される場合、バックアップの復元時に *Barman* によって `recovery.signal` ファイルが作成され、ターゲットリカバリを開始するようにサーバーに通知されます。
- `backup_id` を `auto` として指定する場合、許可されるリカバリターゲットはのみです `--target-time`、`--target-lsn` および `--target-tli`。 `--target-time` および `--target-lsn` は相互排他的であるが、`--target-tli` は独立してまたは `--target-time` または `--target-lsn` とともに使用できることに注意してください。

前述のターゲットは、次の値を取ることができる `--target-action` で使用できます。

- `shutdown` 目標に到達したら、Postgres をシャットダウンします。
- `pause` ターゲットに到達したら、検査のために Postgres を一時停止します。
- `promote` ターゲットに到達すると、Postgres をプライマリに昇格させます。

`--standby-mode` を呼び出すことにより、インスタンスをスタンバイとして構成することもできます。バックアップが復元された後、復元されたノードをリカバリモードで起動する前に、目的の上流ノードに接続するように構成を変更してください。

注釈

- `--standby-mode` を指定すると、`recovery.signal` ファイルの代わりに `standby.signal` ファイルが作成されます。
- `--standby-mode` を使用する場合、可能ですが、`--target-*` オプションを設定する必要はありません。

参考

For more information regarding Postgres recovery behavior, refer to [Archive Recovery](#) and [Recovery Target](#)

10.6 Barman から WAL の取得

`--get-wal` を使用して、リカバリ中に Barman から WAL を取得するように Postgres を構成します。設定されていない場合、Barman は `restore` コマンドの一部として Postgres リカバリに必要なすべての WAL をコピーします。

注釈

`--target-xid`、`--target-name`、または `--target-time` などのターゲットで `--no-get-wal` を使用する場合、Barman は WAL アーカイブ全体をコピーして可用性を保証します。

もう 1 つのオプションは、復旧オペレーションの前にグローバル/サーバーレベルで `recovery_options` 構成内に `get-wal` を含めて、リカバリプロセス中に `--get-wal` を指定せずに WAL ファイルを取得し、

Barman サーバーを効果的にサーバーの WAL ハブに変えることです。

```
recovery_options = 'get-wal'
```

復元中に `get-wal` が含まれる場合、Barman は、リカバリがローカルかリモートかに応じて、`barman get-wal` または `barman-wal-restore` のいずれかを使用して必要な WAL ファイルを取得するように `restore_command` を設定します。

`recovery_options` で `get-wal` が指定されているが、特定のリカバリー中には必要ない場合、`barman restore` コマンドで `--no-get-wal` オプションを使用して無効にできます。

注釈

`get-wal` を有効にしてデルタリカバリーを実行している場合、両方を `recovery_options = get-wal, delta-restore` として構成に追加できます。

10.6.1 ローカルリカバリーでの `get-wal` の使用

`barman get-wal` コマンドは、Barman によって管理される WAL カタログへのアクセスが必要であるため、ローカルリカバリでは常に `barman` ユーザーとして実行する必要があります。

```
restore_command = 'barman get-wal SERVER_NAME %f > %p'
```

これは、PostgreSQL サーバーが `barman` ユーザーとしても実行されることを前提としています。

別のユーザーたとえば `postgres` で PostgreSQL を実行する場合、あなたには以下の責任があります。

- サーバーを実行するユーザーに一致するように PGDATA およびすべての表領域の所有権を更新します。
- `barman` ユーザーとして `barman get-wal` を呼び出すことができるように `restore_command` を調整しますたとえば、そのユーザーになるように `sudo` または他のメカニズムを構成することにより。

言い換えれば、`barman` 以外のユーザーを使用してローカルリカバリーで PostgreSQL を実行する場合、追加の構成で WAL ファイルを Barman から取得でき、サーバーを起動できるようにする必要があります。

たとえば、`postgres` ユーザーが `barman` ユーザーとして `get-wal` コマンドを実行できるようにするには、次の行を `/etc/sudoers` ファイルに追加します `SERVER` を実際のサーバー名に置き換えます。

```
postgres ALL=(barman) NOPASSWD: /usr/bin/barman get-wal SERVER *
```

10.6.2 リモートリカバリーでの `get-wal` の使用

リモートリカバリーの場合、`recovery_options` を `get-wal` に設定すると、SSH 接続エラーをより堅牢に処理するように設計された `recovery_options` スクリプトを使用して `restore_command` が作成されます。

このスクリプトは、WAL ファイルの自動圧縮および解凍や `peek` 機能などの便利な機能を提供し、Postgres が以前の WAL ファイルを処理しているときに今後の WAL ファイルを取得でき、Postgres と Barman 間の帯域幅を最適化します。

`barman-wal-restore` は、`barman-cli` パッケージに含まれています。次に、`barman-wal-restore` の `restore_command` の例を示します。

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

ここで、`backup` は、Barman がインストールされているホストを指します。SSH 経由で通信するため、`postgres` ユーザーがバックアップサーバーに `barman` としてログインするには、SSH キー認証が必要です。デフォルト以外の SSH ポートを使用する必要がある場合は、`--port` オプションで指定できます。

`barman-wal-restore` が Barman サーバーに接続できること、および必要な Postgres サーバーが WAL ファイルを送信するように設定されていることを確認するには、次のコマンドを使用します。

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

ここで、`backup` は Barman がインストールされているホストを指し、`pg` は Barman で構成された Postgres サーバーの名前であり、`DUMMY` はプレースホルダーとして機能します。スクリプトには、WAL ファイル名と宛先ディレクトリの 2 つの引数が必要ですが、これらは無視されます。

すべてが正しく設定されている場合、次が表示されます。

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

`barman-wal-restore` コマンドの詳細については、`barman-cli` がインストールされたホストで `man barman-wal-restore` と入力するか、`barman-wal-restore` コマンドリファレンスを参照してください。

💡 Tip

`pg_wal` ディレクトリと `spool` ディレクトリの両方が同じファイルシステムにある場合、ファイルはコピーされるのではなく名前が変更されるため、WAL ファイルの提供が速くなります。ただし、これらのディレクトリが別のファイルシステム上にある場合、操作にはファイルのコピーと元の削除の両方が含まれるため、パフォーマンスの向上はありません。WAL ファイル管理効率を最適化するために、ファイルシステムの場所に注意してください。

10.7 暗号化されたバックアップの復元

暗号化されたバックアップと WAL は、復元フェーズで最終的な宛先にコピーされる前に復号されます。リストア中に、プライベートキーのパスフレーズを取得するコマンドが `encryption_passphrase_command` に存在する必要があります。このコマンドは、パスフレーズを標準出力に出力する必要があり、パスワードボールド、外部キー管理サービス、ファイルなどの安全な場所からパスフレーズを取得するために使用できます。

これらは、パスフレーズコマンドを設定する方法のいくつかの例です。

- 環境変数からの読み取りの例

```
encryption_passphrase_command="echo $BARMAN_PASSPHRASE"
```

- ファイルからの読み取りの例

```
encryption_passphrase_command="cat /path/to/barman_passphrase"
```

- HashiCorp Vault からの読み取り例

```
encryption_passphrase_command="vault kv get -field=<FIELD> <KEY>"
```

- AWS Secret Manager からの読み取り例

```
encryption_passphrase_command="aws secretsmanager get-secret-value --secret-id  
<SECRET_NAME> --profile <AWS_PROFILE> --output text --query SecretString | jq -r  
'<SECRET_KEY>'"
```

バックアップの復号化は次のように発生します。

1. バックアップは、`staging_location=remote` が設定されているかどうかに関係なく、Barman サーバー上のステージングディレクトリに復号されます。これは、復号化が Barman ノードでローカルに発生するように設計されているためです。復号化に使用されるディレクトリは、`staging_path` オプションでも定義されます。
2. 解凍や組み合わせなど、インクリメンタルバックアップの場合など、追加の操作が必要な場合、それらはステージングディレクトリのコンテンツをソースとして使用して実行されます。それ以外の場合、復号化されたファイルは最終的な宛先に直接コピーされます。
3. 追加の操作が必要な場合、後続の操作が完了した後にステージングディレクトリは削除されます。

WAL ファイルの復号化は、リカバリ中にファイルが取得される方法によって異なります。

1. `--no-get-wal` オプションデフォルトを使用する場合、必要な WAL ファイルはすべてステージングディレクトリに復号され、最終的な宛先にコピーされます。これは、`encryption_passphrase_command` が 1 回呼び出され、出力がすべての WAL ファイルに再利用されることを意味します。また、コマンドは `barman restore` コマンドの実行中にのみ必要であり、Postgres リカバリプロセス中には必要ありません。
2. `--get-wal` オプションを使用すると、復旧プロセス中に必要に応じて、WAL ファイルが Postgres サーバーに提供されます。このシナリオでは、Barman は Postgres サーバーに送信する前に、各 WAL ファイルをローカルで復号化します。これは、`restore_command` を介して取得される WAL ファイルごとに `encryption_passphrase_command` が 1 回呼び出されることも意味します。

重要

`staging_location=remote` を使用する場合、`staging_path` は、ローカルノードでの読み取りおよび書き込みのためにアクセス可能な場所を指す必要があります。これは、復号化が常にローカルノードで発生するためです。

10.8 圧縮バックアップの復元

`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 はリストア中にそれを解凍できます。

このプロセスには、いくつかの手順が含まれます。

1. ローカルに復元する場合、圧縮されたバックアップファイルは復元先ディレクトリに直接解凍されます。リモートで復元する場合、動作は `staging_location` 設定によって異なります。
 - `staging_location=local` 圧縮されたバックアップは、Barman サーバー上の `staging_path` で解凍され、`rsync` を使用してリモートの復元先にコピーされます。
 - `staging_location=remote` 圧縮バックアップは、`rsync` を使用してリモートサーバーの `staging_path` にコピーされ、リモートの復元先にて解凍されます。
2. リモートリカバリーの場合、特別な処理を必要とする構成ファイルは、復元先ディレクトリから `barman` ノードのローカル一時ディレクトリにコピーされ、必要に応じて編集および変更されて、`Rsync` を使用してリストアディレクトリに戻されます。ローカル復旧の場合、ローカル一時ディレクトリは復元先自分自身であるため、編集とマングリング操作はインプレースで実行されます。バックアップディレクトリには圧縮された `tarball` ファイルのみが含まれているため、Barman は復元ディレクトリ内の個々のファイルにのみアクセスできるため、この中間手順が必要です。
3. 追加の操作が必要な場合、後続の操作が完了した後にステージングディレクトリは削除されます。

Barman はデプロイ環境の知識がないため、`staging_path` および `staging_location` オプションに依存して、ステージングディレクトリの適切な場所を決定します。グローバル/サーバー構成でオプションを設定するか、`barman restore` コマンドで `--staging-path` および `--staging-location` オプションを使用します。Barman は自分自身で適切な場所を推測できないため、そうしないとエラーが発生します。

10.9 ブロックレベルのインクリメンタル バックアップのリカバリー

ブロックレベルのインクリメンタルバックアップから復旧する場合、Barman は `pg_combinebackup` を使用してバックアップチェーンを結合します。このチェーンは、ルートバックアップと、リカバリされるものまでの後続のすべての増分バックアップで構成されています。

ブロックレベルのインクリメンタルバックアップから正常にリカバリーするには、グローバル/サーバー構成で `staging_path` および `staging_location` オプションを指定するか、`barman restore` コマンドで同等の `--staging-path` および `--staging-location` オプションを使用する必要があります。Barman は適切なステージング場所を自動的に決定できないため、そうしないとエラーが発生します。

このプロセスには、次の手順が含まれます。

1. Barman は、バックアップのチェーンを組み合わせて、合成バックアップを作成します。ローカルに復元する場合、バックアップのチェーンは復元先ディレクトリに直接結合されます。リモートで復元する場合、動作は `staging_location` 設定によって異なります。
 - `staging_location=local` バックアップのチェーンは Barman サーバー上の `staging_path` で結合され、`rsync` を使用してリモートの復元先にコピーされます。
 - `staging_location=remote` バックアップのチェーンは、`rsync` を使用してリモートサーバーの `staging_path` にコピーされ、リモートの復元先で結合されます。

2. 追加の操作が必要な場合、後続の操作が完了した後にステージングディレクトリは削除されます。

重要

チェーン内のバックアップがチェックサムを無効にして取得されたが、最終的なバックアップでチェックサムが有効になっている場合、結果の合成バックアップには、無効なチェックサムを含むページが含まれる場合があります。詳細については、[pg_combinebackup documentation](#) の制限を参照してください。

10.9.1 結合モード

結合バックアップにファイルを作成するために使用される方法は、`--combine-mode` コマンドの `--combine-mode` オプションで制御できます。

サポートされているモードは次のとおりです。

copy — チェーン内のファイルごとに標準のファイルコピーを実行します。これはデフォルトモードであり、すべてのシステムで動作します。生成されるバックアップのファイルは元のファイルから完全に独立しているため、最高レベルの安全性を提供しますが、より多くの時間、追加のディスク領域と I/O が犠牲になります。

copy-file-range — `copy_file_range` システムコールを使用して、効率的なファイルコピーを行います。一部のファイルシステムでは、これは物理ディスクブロックを共有することにより `--clone` と同様の結果を達成できますが、他のファイルシステムでは、最適化されたパスを介してフルコピーを実行します。このモードは、現在 Linux と FreeBSD でサポートされています。バックアップマニフェストが利用できないか、互換性のあるチェックサムが含まれていない場合でも、ファイルはコピーされますが、チェックサム検証のためにブロックごとに読み取られます。

link — 可能な場合、ファイルを合成バックアップにコピーする代わりに、ハードリンクを作成します。このアプローチは場合によってはより速く、ディスク領域を節約できますが、潜在的なリスクをもたらします。復元ディレクトリに加えられた変更は、元のバックアップディレクトリにも影響を与えます。その逆も同様です。したがって、このモードは、入力ディレクトリが結合後に削除される使い捨てコピーである場合にのみ使用する必要があります。入力バックアップと出力ディレクトリは、同じファイルシステムに存在する必要があります。バックアップマニフェストに互換性のあるチェックサムが含まれていない場合、ハードリンクは作成されますが、ファイルはチェックサム計算のためにブロックごとに読み取られます。

clone — 一部のファイルシステムが提供するファイルクローン機能 `reflink` などを使用して、最初にデータブロックを共有するファイルの軽量コピーを作成します。このアプローチは、リンクの速度と領域効率とコピーの分離を組み合わせます。いずれかのコピーを変更しても、他方には影響しません。このモードのサポートは、基になるファイルシステムによって異なりますたとえば、XFS および Btrfs はクローンをサポートしています。

これらのモードの動作の詳細については、[pg_combinebackup documentation](#) を参照してください。

10.10 マルチフォーマットバックアップのリカバリパイプライン

バックアップは、圧縮、暗号化、インクリメンタル、またはこれらの組み合わせが可能です。Barman は、復元中に必要に応じて解凍、復号、およびインクリメンタルなバックアップチェーンの組み合わせを自動的に処理することにより、復旧プロセスを合理化します。barman restore コマンドを発行すると、Barman はバックアップタイプを検出し、必要なすべての操作を正しい順序で実行し、手動介入を削除します。

例

- バックアップが暗号化されている場合、Barman は構成された `encryption_passphrase_command` を使用してバックアップを復号します。
- バックアップが圧縮されている場合、Barman はデータファイルを復元する前にそれを解凍します。
- バックアップがブロックレベルのインクリメンタルの場合、Barman は `pg_combinebackup` を使用してバックアップチェーンを結合して、復元可能な完全なバックアップを生成します。
- バックアップが暗号化、圧縮、およびインクリメンタルの両方である場合、Barman は次の順序で自動的に処理します。復号、解凍、およびインクリメンタルチェーンの組み合わせ。ステップごとに、Barman は選択した場所に一時的なステージングディレクトリを作成します。ステップが完了すると、前のステップのステージングディレクトリが削除されます。これは、3 つまでのステージングディレクトリが同時にディスク領域を使用しないことを意味します。

構成ファイルまたはコマンドライン引数として `staging_path` および `staging_location` オプションを使用して、これらの操作の場所を制御できます。この柔軟性により、特にリモート復旧シナリオで、使用可能なディスク領域とネットワーク帯域幅を最適化できます。

たとえば、バックアップが暗号化、圧縮、およびインクリメンタルであり、`staging_location=remote` および `staging_path=/tmp` を使用したリモートリカバリーの場合

1. チェーン内のすべてのバックアップは、Barman ノードのステージングパスたとえば `/tmp/decrypt-staging-location` に復号されます。
2. 次に、それらは `rsync` でリモートノードたとえばリモートノードの `/tmp/rsync-staging-location` のステージングディレクトリにコピーされ、コピーが完了すると、Barman ノードのステージングパス `/tmp/decrypt-staging-location` が削除されます。
3. 次に、コピーされたバックアップは別のリモートステージングディレクトリなどに解凍されます。リモートノード上の `/tmp/decompress-staging-location`、解凍後、リモートノード上のステージングパス `/tmp/rsync-staging-location` が削除されます。
4. 最後に、解凍されたファイルは、復元先を出力として使用する合成バックアップに結合されます。
5. この後、解凍ステージングディレクトリ `/tmp/decompress-staging-location` も削除されます。

`staging_location=local` の場合、すべての操作は Barman サーバーで実行されますが、順序はわずかに異なります。`rsync` コピーは最後に遅延されます。結合オペレーションは独自の `staging_path` を使用し、最後のステップは、合成バックアップをリモートノードのリストア先に転送することです。

10.11 .partial WAL ファイルの制限

`streaming_archiver` を使用する場合、Barman は `pg_receivewal` を利用して、ネイティブストリーミングレプリケーションプロトコルを介して Postgres 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。デフォルトでは、`pg_receivewal` はこれらのログを `.partial` サフィックスを持つファイルに書き込み、これらがまだ完了していないことを示します。Barman は、`streaming_wals_directory` でこれらの `.partial` ファイルを探します。`pg_receivewal` はファイルが完了すると、`.partial` サフィックスを削除し、永続的なストレージと圧縮のために Barman の `archive-wal` コマンドに渡します。

マスター Postgres サーバーに突然障害が発生し、復旧できない場合、Barman にストリーミングされた `.partial` ファイルには、アーカイブプロセスに配信されなかった重要なデータが含まれている可能性があります。

Barman バージョン 2.10 以降、`get-wal` コマンドは、`--partial` または `-P` オプションを使用して、現在の `.partial` WAL ファイルのコンテンツを取得できます。これは、完全な復元を実行するか、ポイントインタイムリカバリーを実行するかにかかわらず、リカバリーに役立ちます。`get-wal` を使用し、`--standby-mode` を使用せずに `restore` コマンドを開始すると、Barman は `barman-wal-restore` コマンドに `-P` オプションを自動的に含めて、`.partial` ファイルを処理します。

さらに、`get-wal` は、`incoming` ディレクトリをチェックして、Barman に送信されたがまだアーカイブされていない WAL ファイルがないかを確認します。

`no-get-wal` でリカバリーする場合、Barman はアーカイブされたすべての WAL を宛先ノードにコピーします。この場合、部分的な WAL ファイルはコピーされず、トランザクションからの最終的な損失データは部分ファイルに記録されます。

このような制限を回避するには、`streaming_wals_directory` にある部分 WAL ファイルを宛先ノードの `streaming_wals_directory` にコピーし、`.partial` サフィックスなしで名前を変更できます。

10.12 外部構成ファイルの管理

Barman は、バックアップが最初に取得された方法に応じて、異なる方法で外部構成ファイルを復元します。`rsync` バックアップを復元する場合、外部ファイルは元の場所ではなく、`rsync` を介して `'PGDATA'` ディレクトリに復元されます。構成ファイルに関連するものを含む、潜在的に危険な設定に関する警告が発行されます。対照的に、`postgres` バックアップを復元する場合、外部ファイルはバックアップされていないため、復元されません。警告は、復元されなかったファイルについてユーザーに通知します。

バックアップの章の *Managing external configuration files* セクションを参照して、バックアップを作成するときに外部ファイルがどのように処理されるかを理解します。

10.13 スナップショットバックアップからの復旧

Barman は現在、スナップショットバックアップからの完全な自動リカバリーをサポートしていません。この制限は、スナップショットのリカバリーには、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、このタスクは、Terraform や OpenTofu のような専用の `'IAC'` ソリューションで処理するのに最適なタスクであるために発生します。

ただし、`barman resume` コマンドを使用して、スナップショットリカバリインスタンスを検証し、安全でない設定の Postgres 構成を確認したり、必要な PITR オプションの構成などのリカバリ後のタスクを実行することはできます。また、このコマンドは、`backup_label` ファイルはボリュームスナップショットに含ま

れていないため、`backup_label` ファイルをコピーし、必要な WAL ファイルを転送します。 `--get-wal` リカバリオプションが指定されていない場合、WAL。

`barman-cloud-backup` で作成したバックアップから復元する場合、`barman restore` の代わりに `barman-cloud-restore` コマンドを使用する必要があります。

注釈

クラウドプロバイダーを使用する場合、同じ要件と構成が復元に適用されます。 **Requirements and Configuration** セクションと、 **Requirements and Configuration** セクションで作業している特定のクラウドプロバイダーを参照してください。

10.13.1 リカバリーの手順

1. バックアップ中に作成される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. ステップ 1 の各ディスクがバックアップメタデータに従って接続およびマウントされるコンピューティングインスタンスをプロビジョニングします。
3. `barman restore` または `barman-cloud-restore` コマンドを使用して、リカバリーを検証およびファイナライズします。

ステップ 1 および 2 は、既存の IAC システムで管理するのが理想的ですが、手動またはカスタムスクリプトを介して実行することもできます。

10.13.2 役立つリソース

[Example recovery script for GCP](#)。

[Example runbook for Azure](#)。

これらのリソースは、バックアップおよびリカバリ環境を想定しているため、運用環境で使用する前にカスタマイズする必要があります。

10.13.3 resume コマンドの実行

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作成されたディスクが接続およびマウントされたら、次の追加引数を使用して `barman resume` コマンドを実行します。

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

コマンドの例

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

Barman は、バックアップをスナップショットとして自動的に認識し、接続されたディスクが対応するスナップショットからクローンされたことを確認します。次に、バックアップラベルと WAL をコピーし、必要な復旧オプションで Postgres 構成を調整することにより、Postgres の復旧の準備をします。

プロバイダー固有の引数

GCP の場合

- `--gcp-zone` リカバリインスタンスが存在するアベイラビリティゾーン。省略すると、Barman はサーバー構成で設定された `gcp_zone` 値を使用します。

Azure の場合

- `--azure-resource-group` リカバリインスタンスのリソースグループ。提供されない場合、Barman はサーバー構成の `azure_resource_group` 値を参照します。

AWS の場合

- `--aws-region` リカバリインスタンスの AWS リージョン。指定しない場合、Barman はデフォルトでサーバー構成で設定された `aws_region` 値を使用します。

第11章 リテンションポリシー

11.1 概要

バックアップの保持ポリシーは、データのバックアップコピーが時間の経過とともに処理される方法を管理するように設計された一連の戦略的ガイドラインです。このポリシーは、バックアップを保持する期間、アーカイブまたは削除する時期、および編成方法に関するルールとガイドラインの概要を示しています。明確に定義された保持ポリシーの実装は、データ保護を保証、ストレージ使用の最適化、およびコンプライアンス要件を満たすために重要です。

注釈

保持ポリシーは、ステータスが **DONE** のバックアップにのみ適用されます。不完全なバックアップたとえば、**WAITING_FOR_WALS** 状態または **FAILED** は、保持ルールによって考慮されません。

11.2 リテンションポリシーの主要コンポーネント

これらのポリシーの主要コンポーネントを理解することは、データ保護、ストレージ効率、コンプライアンスのバランスを保つシステムを設計するために重要です。

11.2.1 リテンション期間

- **Time-Based Retention** このコンポーネントは、バックアップを固定期間 30 日、1 年など、バックアップを保持する期間を指定します。時間ベースの保持は簡単で、特定の期間より古いバックアップが自動的に削除されます。
- **Quantity-Based Retention** あるいは、保持ポリシーは、保持されたバックアップの数たとえば、最後の 10 バックアップなどに基づくことができます。この方法は、経過期間に関係なく、特定の数の最近のバックアップを維持するのに役立ちます。

11.2.2 バックアップの種類

- **Full Backups** : これらのバックアップはデータセット全体をキャプチャし、その包括的な性質により長く保持されることがよくあります。他の種類と比べて、フルバックアップには異なる保持ポリシーが適用される場合があります。
- **Incremental Backups** インクリメンタルバックアップは、最後のバックアップ以降の変更をキャプチャします。これらのバックアップの保持ポリシーは、バックアップチェーンでのロールと他のバックアップへの依存関係を反映して、異なる場合があります。

11.2.3 クリーンアップルール

- **Automated Cleanup** 効果的な保持ポリシーには、事前定義されたルールに従って古いバックアップを特定および削除する自動クリーンアップメカニズムが含まれます。これにより、手動介入が削減され、不要なデータが保持されるリスクが最小限に抑えられます。
- **Archiving and Deletion** クリーンアップルールでは、古いバックアップを削除前にアーカイブするか、直接削除するかを指定できます。アーカイブは、コンプライアンスまたはその他の目的で履歴データを維持するのに役立ちます。

11.3 リテンションポリシーの主な目的

堅牢なリテンションポリシーの実装は、効果的なバックアップ管理の基礎であり、いくつかの重要な目的を含みます。

11.3.1 十分なデータ保護の確保

- **Historical Recovery** 保持ポリシーは、さまざまな時点からのリカバリーを促進するためにバックアップを保持する期間を定義します。これは、データの損失、破損、または誤った変更が発生した場合に、最近のバックアップからだけでなく、古いバックアップからデータを復旧するためにも重要です。
- **Recovery Flexibility** さまざまな期間にわたってバックアップを保持することにより、最新のデータの復元、破損の問題への対処、または誤った操作の元に戻すなど、さまざまな種類のデータ復旧シナリオに対応できます。

11.3.2 ストレージ使用の最適化

- **Efficient Storage Management** 保持ポリシーは、貴重なストレージ領域を消費する古いバックアップの蓄積を防ぐのに役立ちます。これは、バックアップの数または保持期間に制限を設定することにより実現され、それによりストレージ使用率が最適化され、コストが効果的に管理されます。
- **Cost Control** 古いバックアップのクリーンアップを自動化することにより、組織はストレージインフラストラクチャと関連するメンテナンスに関連する不必要な費用を回避できます。

11.3.3 コンプライアンスとレギュレーション

- **Meeting Legal Requirements** 多くの業界には、データ保持を管理する特定の規制があり、バックアップの最小保持期間を規定する場合があります。明確に定義された保持ポリシーにより、これらの規制要件が満たされることが保証され、組織が法的および業界標準に準拠し続けるのに役立ちます。
- **Audit Readiness** 適切な保持ポリシーは、保持規制へのコンプライアンスを示す明確で編成されたバックアップ履歴を維持することにより、監査を容易にします。

11.4 最小限の冗長性の安全性

グローバルまたはサーバーごとの構成で `minimum_redundancy` オプションを使用して、Postgres サーバーのバックアップの最小数を設定できます。デフォルトでは、このオプションは 0 に設定されています。

`minimum_redundancy` を 0 より大きい数値に設定すると、Barman は常に少なくともその数のバックアップがサーバー上で利用できるようにします。

この設定は、バックアップの誤った削除からの保護に役立ちます。

注釈

リテンションポリシーが最小冗長設定と競合しないことを確認します。Barman のログに関連するメッセージがないか定期的にチェックしてください。

11.5 リテンションポリシーの範囲

Barman、2 つの方法で保持ポリシーを定義できます。

11.5.1 バックアップの冗長性

保持するバックアップの数を指定します。Barman は、最新のバックアップを指定された数まで保持します。このタイプのポリシーは、保持期間は考慮せず、バックアップ数に焦点を当てます。

たとえば、冗長性を 3 に設定すると、Barman は 3 つの最新のバックアップを保持し、古いものは破棄します。

11.5.2 リカバリーウィンドウ

そのウィンドウ内の任意のポイントにリカバリーできるように、バックアップを保持する期間を指定します。間隔ウィンドウは常に現在の時刻に終了し、指定された期間逆方向にスパンします。Barman は、このウィンドウ内の任意の時点へのポイントインタイムリカバリーに必要なバックアップとアーカイブログを保持します。

たとえば、7 日間の復旧ウィンドウを設定すると、Barman はバックアップと WAL ファイルを保持して、過去 7 日以内の任意のポイントにリカバリーできるようにします。これは、ウィンドウの外にある最初のバックアップは対応する WAL で保持されますが、このバックアップとすべての古い WAL より前のバックアップは時代遅れとしてマークされ、最終的に削除されることを意味します。

11.5.3 Keep コマンド

`keep` コマンドを使用して、特定のバックアップをマークして、無期限に保持できます。これは、そのバックアップについて前述した保持ポリシーをオーバーライドします。`keep` コマンドの詳細については、`keep` を参照してください。

注釈

インクリメンタルバックアップは、`keep` コマンドの対象外です。この制限は、インクリメンタルバックアップが親バックアップに依存し、独自にリカバリーに使用できないために存在します。独立して使用できないため、ベースなしでインクリメンタルを保持しても意味がありません。

インクリメンタルバックアップに `keep` を適用しようとする、Barman は、インクリメンタルバックアップを保持できないことを示すエラーを返します。

Only the full base backup — the one that starts the backup chain— can be marked with keep.

11.6 ユースケース

11.6.1 ポイント インタイム リカバリー

ベースバックアップとアーカイブされた WAL ファイルには、同じ保持ポリシーがあります。この設定により、Postgres サーバーからデータを最も古いバックアップの終了時間からの任意の時点に復旧できます。

11.6.2 業務効率とスペース管理

特にリソースが制限されている環境では、古いバックアップを定期的に削除して、ストレージコストを節約し、ストレージ領域を効果的に管理しながら、特定の数の最近のバックアップを維持することができます。

11.6.3 長期アーカイブ

コンプライアンスまたは履歴の目的で、通常の操作要件を超えて長期間バックアップを保持する必要がある場合があります。これは、データを特定の期間保持する必要がある規制業界で多くの場合必要です。

11.7 保持ポリシーが適用される方法

Barman のリテンションポリシーは、`barman cron` によって実行される Barman のメンテナンスタスクによって自動的に適用されます。

11.8 構成と構文

リテンションポリシーは、マルチサーバー環境での柔軟性を提供する `retention_policy` オプションを使用してグローバルまたはサーバーごとに構成されます。デフォルトでは、`retention_policy` オプションの値が設定されていないため、保持は強制されません。

リテンションポリシーの構文は次のとおりです。

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

- 値は 0 より大きい整数である必要があります。
- バックアップ冗長性の場合、値はサーバーの最小冗長レベル以上である必要があります。
- リカバリウィンドウの場合、値は逆の順序でサーバーの最小冗長レベルと少なくとも同じ高さである必要があります。
- 値が割り当てられていない場合、警告が生成されます。

重要

ブロックレベルのインクリメンタルバックアップは、親バックアップとルートバックアップに依存するため、リテンションポリシーでは考慮されません。ルートバックアップのみが保持期間を決定するために使用されます。

11.9 ブロックレベルのインクリメンタルバックアップのリテンションポリシー

保持ポリシーが適用される場合

- Barman はルートバックアップに焦点を当てます。
- ルートバックアップが **KEEP:FULL** としてマークされている場合、ルートバックアップが保持ポリシー内にあるかどうかに関係なく、関連するすべてのインクリメンタルバックアップは **VALID** としてマークされます。
- ルートバックアップが **KEEP:STANDALONE** としてマークされ、保持ポリシー内にある場合、関連するすべてのインクリメンタルバックアップは **VALID** としてマークされます。ただし、ルートバックアップが保持ポリシーの対象外である場合、関連するすべてのインクリメンタルバックアップは **OBSOLETE** としてマークされます。
- ルートバックアップが **KEEP** フラグでマークされていない場合、関連するすべてのインクリメンタルバックアップは同じラベルを継承します。たとえば、ルートバックアップが **OBSOLETE** としてマークされている場合、関連するすべてのインクリメンタルバックアップも **OBSOLETE** としてマークされます。

11.10 クラウドバックアップのリテンションポリシー

クラウドバックアップには2つのシナリオがあります。

1. Barman サーバーで *snapshots backups* を一元化されたバックアップおよびリカバリーマネージャーとして使用する。
2. クラウドオブジェクトストレージで *cloud backups* を使用して、Barman サーバーを使用せずにバックアップを管理します。

最初のシナリオでは、Barman は **cron** で説明しているように、メンテナンス操作と保持ポリシーの適用に **cron** を使用します。この場合、**snapshot** バックアップは他の **rsync** または **postgres** バックアップと同じように扱われます。

2 番目のシナリオでは、Barman サーバーが存在しないため、保持ポリシーを適用するためのメンテナンス操作の **cron** がありません。代わりに、**barman-cloud-backup-delete** を **-r RETENTION_POLICY** オプションとともに使用する必要があります **barman-cloud-backup-delete** を参照してください。これにより、指定された保持ポリシーを満たさないバックアップが削除されます。さらに、フックスクリプトまたはカスタムスクリプトを使用してこれらのコマンドをスケジュールして、クラウドバックアップの **cron** メンテナンスをシミュレートすることもできます。

第12章 診断と障害対応

ツールに必要な機能の1つは、効率的な方法で問題をトラブルシューティングする機能です。Barman は、問題を診断およびトラブルシューティングする複数の方法を提供します。

通常、問題は `barman check` から返されたエラーまたは警告メッセージから発生しますが、他のソースから発生する場合があります。

12.1 Barman ステータス

`barman status` コマンドを使用して、特定の Postgres サーバーのステータスを確認できます。このコマンドは、PostgreSQL バージョン、現在のデータサイズ、PostgreSQL データディレクトリ、現在の WAL、archive コマンド、最後にアーカイブされた WAL、使用可能なバックアップの数など、サーバーのステータスの詳細ビューを提供します:

```
$ barman status pg16
Server pg16:
  Description: PostgreSQL 16 Database (via SSH)
  Active: True
  Disabled: False
  PostgreSQL version: 16.4
  Cluster state: in production
  Current data size: 31.9 MiB
  PostgreSQL Data directory: /var/lib/pgsql/16/data
  Current WAL segment: 000000010000003B000000010
  PostgreSQL 'archive_command' setting: barman-wal-archive barman_server pg16 %p
  Last archived WAL: 000000010000003B0000000F, at Mon Nov 18 07:54:54 2024
  Failures of WAL archiver: 0
  Passive node: False
  Retention policies: not enforced
  No. of available backups: 2
  First available backup: 20240911T085206
  Last available backup: 20240911T085736
  Minimum redundancy requirements: satisfied (2/0)
```

これは、サーバーの構成が barman サーバーの構成と連携するための良いツールです。

12.2 Barman ショーサーバー

barman show-servers コマンドを使用して、バックアップ用に構成された特定の Postgres サーバーのすべての構成オプションと値をリストできます。

```
$ barman show-servers pg16
Server local16:
  active: True
  archive_command: barman-wal-archive barman-server pg16 %p
  archive_mode: on
  archive_timeout: 0
  archived_count: 1
  ...
```

12.3 Barman 複製ステータス

barman replication-status コマンドを使用して、特定の Postgres サーバーのレプリケーションステータスをすばやく確認できます。このコマンドは、レプリケーションのステータスの詳細なビューを提供します。

```
$ barman replication-status pg16
Status of streaming clients for server 'pg16':
Current LSN on master: 3B/100000148
Number of streaming clients: 1

1. Async WAL streamer
  Application name: barman_receive_wal
  Sync stage      : 3/3 Remote write
  Communication   : Unix domain socket
  User name       : barman
  Current state   : streaming (async)
  Replication slot: standby
  WAL sender PID  : 165959
  Started at      : 2024-11-18 07:49:01.837787+00:00
  Sent LSN       : 3B/100000148 (diff: 0 B)
  Write LSN      : 3B/100000148 (diff: 0 B)
  Flush LSN      : 3B/100000000 (diff: -328 B)
```

12.4 Barman 診断

barman diagnose コマンドは、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します。これは、Barman によってバックアップされている構成済みの Postgres サーバーの全体図です。

```
barman diagnose
```

diagnose コマンド出力は barman サーバーの完全なスナップショットであり、グローバル構成、SSH バージョン、Python バージョン、rsync バージョン、PostgreSQL クライアントバージョン、すべてのサーバー

の現在の構成とステータスなどの役立つ情報を提供します。

diagnose コマンドは、Barman インストールのステータスに関するグローバルビューを提供するため、問題のトラブルシューティングに非常に役立ちます。

第13章 構成リファレンス

Barman は、構成より慣例のアプローチに従います。これにより、一部のオプションをグローバルに定義し、サーバーレベルでオーバーライドできることにより構成が簡素化されます。これは、サーバーのデフォルトの動作を設定し、必要に応じて特定のサーバーをカスタマイズできることを意味します。このデザインにより、柔軟性を維持しながら、過剰な構成の必要性が削減されます。

13.1 使用法

効果的な操作には、適切な構成が重要です。Barman は、さまざまな種類の構成ファイルを使用して、グローバル設定、サーバー固有の設定、および3つのスコープで構成されるモデル固有の設定を管理します。

1. **Global Configuration:** It comprises one file with a set of configurations for the barman system, such as the main directory, system user, log file, and other general options. Default location is `/etc/barman.conf` and it can be overridden on a per-user level by `~/.barman.conf` or by specifying a `.conf` file using the `-c / --config` with the barman command directly in the CLI.

Barman は、事前に定義された順序で構成ファイルを検索し、最初に見つけたものを使用します。検索順序は次のとおりです。

- `~/.barman.conf` - ユーザーごとの構成ファイル。
- `/etc/barman.conf` - メインのグローバル構成ファイル。
- `/etc/barman/barman.conf` - 代替グローバル構成ファイル。

2. **Server Configuration:** It comprises one or more files with a set of configurations for a Postgres server that you want to keep track and interact for backup, recovery and/or replication. Default location is `/etc/barman.d` and must use the `.conf` suffix. You may have one or multiple files for servers. You can override the default location by setting the `configuration_files_directory` option in the global configuration file and placing the files in that particular location.

3. **Model Configuration:** It comprises one or more files with a set of configurations overrides that can be applied to Barman servers within the same cluster as the model. These overrides can be implemented using the `barman config-switch` command. Default location is `/etc/barman.d` and must use the `.conf` suffix. The same `configuration_files_directory` override option from the server configuration applies for models. You may have one or multiple files for models.

注釈

歴史的には、グローバル、サーバー、およびモデルオプションを含む単一の構成ファイルを使用できますが、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。

構成ファイルは INI フォーマットに従い、角かっこ内のヘッダーで示されるセクションで構成されています。各セクションにはさまざまなオプションを含めることができます。

モデルとサーバーには一意の識別子が必要で、予約語は名前として使用できません。

Reserved Words

次の予約語は、サーバー名またはモデル名として使用できません。

- **barman** グローバルセクションを識別します。
- **all** すべての管理対象サーバーでコマンドを実行するための特別なショートカット。

Parameter Types

構成オプションには次のタイプがあります。

- **String** テキストデータファイルパス、名前など。
- **Enum** 列挙値、多くの場合事前定義された選択に制限されます。
- **Integer** 数値。
- **Boolean** `on`、`true`、`1` `true` または `off`、`false`、`0` `false` です。

注釈

一部の列挙型では、`off` は許可されますが、`false` は許可されません。

13.2 オプション

構成ファイルのオプションには、特定のスコープまたは共有スコープを持つことができます。次の構成オプションは、Barman がバックアップとリカバリーを実行する方法を構成するだけでなく、Barman が構成済みの Postgres サーバーと相互作用して、バックアップとリカバリー、および高可用性戦略を適用できる方法のさまざまな側面を構成するために使用されます。

13.2.1 一般

これらは一般的な構成オプションです。

active

このオプションが `true` デフォルトに設定されている場合、サーバーは完全に動作します。 `false` に設定されている場合、サーバーは診断使用のみに制限されます。つまり、バックアップ実行や WAL アーカイブなどの操作コマンドが一時的に無効になります。新しいサーバーを Barman に組み込む場合、最初に `active = false` を設定することをお勧めします。サーバーをアクティブにする前に、バーマンチェックで問題が表示されないことを確認します。このアプローチは、初期セットアップ中に Barman での過度のエラーログを防ぐのに役立ちます。

スコープサーバー/モデル。

archiver

このオプションは、サーバーの Postgres の `archive_command` を介したログファイルの配布を有効にします。true に設定すると、Barman は継続的アーカイブが構成されることを想定し、Postgres が受信ディレクトリ `incoming_wals_directory` に保存する WAL ファイルを、チェック、処理、圧縮を含めて管理します。false デフォルトに設定されている場合、継続的なアーカイブは無効になります。

注釈

`archiver` も `streaming_archiver` も構成されていない場合、Barman はこのオプションを true に自動的に設定して、デフォルトでアーカイブが有効になっていた以前のデフォルト動作との互換性を維持します。

スコープグローバル/サーバー/モデル。

archiver_batch_size

このオプションは、0 より大きい値に設定することにより、アーカイバプロセスの WAL ファイルのバッチ処理を有効にします。設定されていない場合、アーカイバは WAL キューに無制限のデフォルトの処理モードを使用します。バッチ処理が有効になると、アーカイバプロセスは実行ごとに最大 `archiver_batch_size` WAL セグメントを処理します。この値は整数である必要があります。

スコープグローバル/サーバー/モデル。

bandwidth_limit

バックアップおよびリカバリー操作の最大転送速度を 1 秒あたりのキロバイト単位で指定します。0 の値は、制限なしデフォルトを示します。

注釈

`backup_method = postgres | rsync` の場合にのみ適用されます。

スコープグローバル/サーバー/モデル。

barman_home

Barman のメインデータディレクトリを指定します。デフォルトの `/var/lib/barman`。

範囲 グローバル。

barman_lock_directory

ロックファイルのディレクトリを指定します。デフォルトは `barman_home` です。

注釈

`barman_lock_directory` は、非ネットワークローカルファイルシステム上にある必要があります。NFS またはその他のネットワークファイルシステムを使用してバックアップと WAL ファイルを保存する場合、lock ディレクトリがローカルファイルシステムにあることを確認してください。

範囲 グローバル。

check_timeout

サーバーごとに Barman check コマンドの最大実行時間を秒単位で設定します。タイムアウトを無効にするには、`0` に設定します。デフォルトは `30` 秒です。負でない整数である必要があります。

スコープ グローバル/サーバー/モデル。

cluster

サーバーまたはモデルを、関連するクラスター名にタグ付けします。Barman は、この関連付けを使用して、このクラスター内のすべてのサーバー/モデルの構成をオーバーライドします。サーバーで省略した場合、デフォルトのサーバーの名前が使用されます。

i 注釈

該当するサーバーをグループ化するには、構成モデルに指定する必要があります。

スコープ サーバー/モデル。

config_changes_queue

`barman config-update` コマンドを介して要求された構成の変更を処理する Barman のキューのファイルシステムの場所を指定します。このキューは、構成変更要求のシリアル化と再試行を管理します。デフォルトでは、Barman は `barman_home` の下の `cfg_changes.queue` という名前のファイルに書き込みます。

範囲 グローバル。

configuration_files_directory

サーバー/モデル構成ファイルが Barman によって読み取られるディレクトリを指定します。デフォルトの `/etc/barman.d/`。

範囲 グローバル。

conninfo

Barman が PostgreSQL サーバーに接続するために使用する接続文字列を指定します。これは libpq 接続文字列です。一般的に使用されるキーには、`host`、`hostaddr`、`port`、`dbname`、`user` および `password` が含まれます。詳細については、`host PostgreSQL` ドキュメントを参照してください。

スコープ サーバー/モデル。

create_slot

ストリーミング WAL ファイルにレプリケーションスロットがまだ存在しない場合、Barman がレプリケーションスロットを自動的に作成するかどうかを決定します。`auto` に設定され、`slot_name` が定義されている場合、Barman はスロットを自動的に作成しようとします。`manual` デフォルトに設定している場合、レプリケーションスロットを手動で作成する必要があります。

スコープ グローバル/サーバー/モデル。

description

人間が読めるサーバーの説明を提供します。

スコープサーバー/モデル。

errors_directory

Barman によるアーカイブ中にエラーが発生した WAL ファイルが保存されるディレクトリ。これには、重複した WAL ファイルたとえば、既にストリーミングされたが、異なるハッシュを持つアーカイブされた WAL ファイル、および WAL アーカイブディレクトリで見つかった予期しないファイルが含まれます。

このディレクトリにファイルを配置する目的は、誰かが後でアーカイブされなかった理由を確認し、適切なアクションを実行できるようにすることです。破棄、どこか別の場所に保存、以前にアーカイブされた重複ファイルを置き換えるなど

スコープサーバー。

encryption

バックアップと WAL ファイルの暗号化に使用される暗号化方法を指定します。サポートされている値は次のとおりです。

- **none** デフォルト 暗号化は適用されません。
- **pgp** 暗号化に `GPG` を使用します。システムに `GPG` をインストールし、適切に構成する必要があります。

スコープグローバル/サーバー/モデル。

encryption_key_id

バックアップと WAL ファイルの暗号化に使用される暗号化キー ID を指定します。このオプションは `encryption = pgp` の場合に必要で、システムで使用可能な有効な `term:GPG` キー ID に対応する必要があります。

スコープグローバル/サーバー/モデル。

encryption_passphrase_command

バックアップと WAL ファイルを復号するための暗号化パスフレーズを取得するために使用されるコマンドを指定します。コマンドはパスフレーズを標準出力に書き込む必要があります。

スコープグローバル/サーバー/モデル。

forward_config_path

`cron` または `sync-info` コマンド中に、パッシブノードが構成ファイルパスをプライマリノードに転送するかどうかを決定します。Barman が `-c / --config` オプションで呼び出され、構成パスがパッシブとプライマリ Barman サーバーの両方で同じである場合、`true` に設定します。デフォルトの `false`。

スコープグローバル/サーバー/モデル。

immediate_checkpoint

Postgres がバックアップの開始時にチェックポイントを処理する方法を制御します。`false` デフォルトに設定して、`checkpoint_completion_target` に従ってチェックポイントを完了できます。即時チェックポイントの場合は `true` に設定します。Postgres はできるだけ早くチェックポイントを完了します。

スコープグローバル/サーバー/モデル。

keepalive_interval

ハートビートクエリーを送信する間隔を秒単位で設定して、rsync バックアップ中に libpq 接続をアクティブに保ちます。デフォルトは 60 秒です。これを 0 に設定すると、ハートビートが無効になります。

スコープ グローバル/サーバー/モデル。

lock_directory_cleanup

barman_lock_directory の未使用のロックファイルの自動クリーンアップを有効にします。

範囲 グローバル。

log_file

Barman のログファイルの場所を指定します。デフォルトの /var/log/barman/barman.log。

範囲 グローバル。

log_level

ロギングのレベルを設定します。オプションには次のものが含まれます DEBUG、INFO、WARNING、ERROR、および CRITICAL。

範囲 グローバル。

minimum_redundancy

保持するバックアップの最小数を指定します。デフォルトは 0 です。

スコープ グローバル/サーバー/モデル。

model

true に設定すると、サーバーセクションを構成ファイルからクラスターのモデルに変更します。この場合、false オプションはありません。false オプションをシミュレートする場合は、構成内のオプションをコメントアウトする #model=true または削除します。サーバー名のデフォルト。

スコープ モデル。

network_compression

ネットワーク転送のデータ圧縮を有効または無効にします。圧縮を無効にするには false デフォルトに設定し、有効にしてネットワーク使用量を削減するには true に設定します。

スコープ グローバル/サーバー/モデル。

parallel_jobs

バックアップまたはリカバリー中にファイルのコピーに使用されるパラレルワーカーの数を制御します。正の整数である必要があります。デフォルトは 1 です。

注釈

backup_method = rsync の場合にのみ適用されます。

スコープ グローバル/サーバー/モデル。

parallel_jobs_start_batch_period

並列ジョブの単一バッチが開始される時間間隔を秒単位で指定します。デフォルトは 1 秒です。これは、`parallel_jobs_start_batch_size` が 10 であり、`parallel_jobs_start_batch_period` が 1 である場合、1 秒以内に開始できる 10 ジョブの最大数があるため、1 秒あたりの 10 ジョブの有効レート制限が得られることを意味します。

i 注釈

`backup_method = rsync` の場合にのみ適用されます。

スコープグローバル/サーバー/モデル。

parallel_jobs_start_batch_size

単一のバッチで開始する並列ジョブの最大数を定義します。デフォルトは 10 ジョブです。これは、`parallel_jobs_start_batch_size` が 10 であり、`parallel_jobs_start_batch_period` が 2 である場合、これは、2 秒以内に開始できる最大 10 ジョブを生成することを意味します。

i 注釈

`backup_method = rsync` の場合にのみ適用されます。

スコープグローバル/サーバー/モデル。

path_prefix

コロンで区切られた 1 つ以上の絶対パスをリストします。Barman は、PostgreSQL バイナリ `PG_MAJOR_VERSION` の適切な `bin` ディレクトリから `rsync`、暗号化、圧縮ツールなどの実行可能ファイルを検索します。これらのパスは `PATH` 環境変数の先頭に追加され、他のパスより前にチェックされます。

スコープグローバル/サーバー/モデル。

primary_checkpoint_timeout

プライマリサーバーにチェックポイントを強制する前に、新しい WAL ファイルを待機する時間。デフォルトの 0。

スコープサーバー/モデル。

primary_conninfo

スタンバイバックアップ中に Barman がプライマリ Postgres サーバーに接続するための接続文字列。

スコープサーバー/モデル。

primary_ssh_command

Barman がパッシブの場合、プライマリ Barman サーバーに接続するための SSH コマンド。

スコープグローバル/サーバー/モデル。

slot_name

`streaming_archiver` が有効になっている場合の `receive-wal` コマンドのレプリケーションスロット名。

スコープグローバル/サーバー/モデル。

ssh_command

Barman が rsync バックアップのために Postgres サーバーに接続するために使用する SSH コマンド。

スコープサーバー/モデル。

streaming_archiver

WAL ファイルの Postgres のストリーミングプロトコルを有効にします。デフォルトの **false**。

i 注釈

archiver も **streaming_archiver** も構成されていない場合、Barman は **archiver** オプションを **true** に自動的に設定して、デフォルトでアーカイブが有効になっていた以前のデフォルト動作との互換性を維持します。

スコープグローバル/サーバー/モデル。

streaming_archiver_batch_size

ストリーミングアーカイバーで WAL ファイルを処理するためのバッチサイズ。デフォルトの **0**。

スコープグローバル/サーバー/モデル。

streaming_archiver_name

receive-wal コマンドのアプリケーション名。デフォルト **barman_receive_wal**。

スコープグローバル/サーバー/モデル。

streaming_backup_name

pg_basebackup コマンドのアプリケーション名。デフォルト **barman_streaming_backup**。

スコープグローバル/サーバー/モデル。

streaming_conninfo

ストリーミングレプリケーションプロトコルの接続文字列。デフォルトの **conninfo**。

スコープサーバー/モデル。

tablespace_bandwidth_limit

バックアップおよびリカバリー操作の特定のテーブルスペースの最大転送速度。 **0** の値は、制限なしデフォルトを示します。

i 注釈

backup_method = rsync の場合にのみ適用されます。

スコープグローバル/サーバー/モデル。

13.2.2 バックアップ

これらの構成オプションは、Barman がバックアップを実行する方法に関連しています。

autogenerate_manifest

これは、バックアップマニフェストファイルの自動作成を許可するブールオプションです。JSON ドキュメントであるマニフェストファイルには、バックアップに含まれるすべてのファイルがリストされています。バックアップの完了時に生成され、backup ディレクトリに保存されます。マニフェストファイルの形式は、`backup manifest format` PostgreSQL ドキュメントで概要を示した仕様に準拠しており、`pg_verifybackup` ツールと互換性があります。デフォルトは `false` です。

注釈

`backup_method` が `rsync` でない場合、このオプションは無視されます。

スコープグローバル/サーバー/モデル。

backup_compression

バックアッププロセスの圧縮方法を指定します。`gzip`、`lz4`、`zstd`、または `none` に設定できます。選択した圧縮方法の CLI ツールが Barman と Postgres サーバーの両方で利用できることを確認します。

注釈

`lz4` および `zstd` には、Postgres バージョン 15 以降が必要であることに注意してください。このオプションの設定を解除するか、`none` を使用すると、非圧縮アーカイブデフォルトになります。`backup_method = postgres` の場合のみサポートされます。

スコープグローバル/サーバー/モデル。

backup_compression_format

圧縮バックアップを保存するときに `pg_basebackup` が使用する形式を決定します。オプションは `plain` または `tar` で、設定されていない場合はデフォルトで `tar`。`plain` 形式は、Postgres バージョン 15 以降を使用しており、`backup_compression_location` が `server` に設定されている場合にのみ使用できます。

注釈

`backup_method = postgres` の場合のみサポートされます。

スコープグローバル/サーバー/モデル。

backup_compression_level

バックアップの圧縮レベルを整数として定義します。許容値は、`backup_compression` で指定された圧縮方法によって異なります。

i 注釈

`backup_method = postgres` の場合のみサポートされます。

スコープグローバル/サーバー/モデル。

backup_compression_location

バックアップ中に圧縮を発生する場所を指定します `client` または `server` のいずれか。 `server` オプションは、Postgres バージョン 15 以降を使用している場合にのみ使用できます。

i 注釈

`backup_method = postgres` の場合のみサポートされます。

スコープグローバル/サーバー/モデル。

backup_compression_workers

バックアッププロセス中の圧縮に使用されるスレッドの数を設定します。これは、`backup_compression=zstd` の場合にのみ適用されます。デフォルト値は 0 で、標準の圧縮動作を使用します。

i 注釈

`backup_method = postgres` の場合のみサポートされます。

スコープグローバル/サーバー/モデル。

backup_directory

サーバーのバックアップデータを保存するディレクトリを指定します。デフォルトの `<barman_home>/<server_name>`。

スコープサーバー。

backup_method

Barman がバックアップを実行するために使用する方法を定義します。オプションには次のものが含まれます

- `rsync` デフォルト SSH で `rsync` コマンドを使用してバックアップを実行します `ssh_command` が必要。
- `postgres` バックアップに `pg_basebackup` コマンドを使用します。
- `local-rsync` Barman は Postgres データベースと同じサーバーで同じユーザーとして実行され、`rsync` ファイルシステムコピーを実行すると仮定します。
- `snapshot` `snapshot_provider` オプションで指定されたクラウドプロバイダーの API を利用して、`snapshot_disks` で定義されたディスクスナップショットを作成し、バックアップラベルとメタデー

タのみを独自のストレージに保存します。

スコープグローバル/サーバー/モデル。

backup_options

バックアップ中に Barman が Postgres と対話する方法を制御します。これは、次のものを含めることができるコンマ区切りリストです。

- **concurrent_backup** デフォルト 同時バックアップを使用し、Postgres バージョン 9.6 以降で推奨され、スタンバイサーバーからのバックアップをサポートします。
- **exclusive_backup** 非推奨の排他的バックアップ方法を使用します。15 より前の Postgres バージョンのみ。このオプションは、将来的に削除される予定です。
- **external_configuration** バックアップ実行中の外部構成ファイルに関する警告を抑制します。

注釈

exclusive_backup と **concurrent_backup** は一緒に使用できません。

スコープグローバル/サーバー/モデル。

basebackups_directory

ベースバックアップが保存されるディレクトリを指定します。デフォルトの <backup_directory>/base。

スコープサーバー。

basebackup_retry_sleep

ベースバックアップコピーが失敗した後、再試行するまでに待機する秒数を設定します。デフォルトは 30 秒です。負でない整数である必要があります。

注釈

これは、バックアップ操作とリカバリー操作の両方に適用されます。

スコープグローバル/サーバー/モデル。

basebackup_retry_times

エラーが発生した後のベースバックアップコピーの再試行回数を定義します。デフォルトは 0 再試行なしです。負でない整数である必要があります。

注釈

これは、バックアップ操作とリカバリー操作の両方に適用されます。

スコープグローバル/サーバー/モデル。

reuse_backup

`backup_method=rsync` を使用する場合、最後に使用可能なバックアップを再利用することにより、インクリメンタルバックアップのサポートを制御します。オプションは次のとおりです。

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

i 注釈

このオプションは `backup_method=postgres` の場合無視されます。

スコープグローバル/サーバー/モデル。

worm_mode

`on` に設定している場合、WORM ライトワンスリードメニーストレージのサポートを有効にし、Barman が不変ストレージのバックアップを正しく処理できるようにします。デフォルトは `off` です。

スコープグローバル/サーバー/モデル。

13.2.3 クラウド バックアップ

これらの構成オプションは、Barman がクラウドでバックアップを実行する方法に関連しています。

aws_await_snapshots_timeout

タイムアウトが発生する前に AWS スナップショットが作成されるのを待機する時間を秒単位で指定します。デフォルト値は 3600 秒です。これは正の整数である必要があります。

i 注釈

`backup_method = snapshot` および `snapshot_provider = aws` の場合にのみサポートされます。

スコープグローバル/サーバー/モデル。

aws_profile

AWS で認証するときに使用する AWS プロファイルの名前たとえば、AWS 資格情報ファイルの INI セクション。

i 注釈

`backup_method = snapshot` および `snapshot_provider = aws` の場合にのみサポートされます。

スコープグローバル/サーバー/モデル。

aws_region

snapshot_instance および snapshot_disks で定義された EC2 VM とストレージボリュームが存在する AWS リージョンを示します。

i 注釈

backup_method = snapshot および snapshot_provider = aws の場合にのみサポートされます。

スコープグローバル/サーバー/モデル。

aws_snapshot_lock_mode

スナップショットのロックモード。これは、snapshot_instance および snapshot_disk が設定されている場合にのみ有効です。

許可されるオプション

- compliance。
- governance。

i 注釈

backup_method = snapshot および snapshot_provider = aws の場合にのみサポートされます。

スコープグローバル/サーバー/モデル。

aws_snapshot_lock_duration

ロック期間は、スナップショットがロックされた期間日単位であり、1～36,500 の範囲です。ロック期間または有効期限のいずれかを設定します両方ではありません。

i 注釈

backup_method = snapshot および snapshot_provider = aws の場合にのみサポートされます。

スコープグローバル/サーバー/モデル。

aws_snapshot_lock_cool_off_period

クーリングオフ期間は、compliance モードでスナップショットをロックするときに指定できるオプションの時間時間単位で、1～72 の範囲で指定します。

i 注釈

backup_method = snapshot および snapshot_provider = aws の場合にのみサポートされます。

スコープグローバル/サーバー/モデル。

aws_snapshot_lock_expiration_date

ロックの期間は、将来の有効期限によって決まります。YYYY-MM-DDTHH:MM:SS.sssZ 形式を使用して、スナップショット作成日時から少なくとも 1 日後である必要があります。ロック期間または有効期限のいずれかを設定します両方ではありません。

i 注釈

`backup_method = snapshot` および `snapshot_provider = aws` の場合にのみサポートされます。

スコープグローバル/サーバー/モデル。

azure_credential

認証に使用する Azure 資格情報の種類を指定します `azure-cli`、`managed-identity` または `default`。指定しない場合、デフォルトの Azure 認証方法が使用されます。

i 注釈

`backup_method = snapshot` および `snapshot_provider = azure` の場合にのみサポートされます。

スコープグローバル/サーバー/モデル。

azure_resource_group

コンピューティングインスタンスと、`snapshot_instance` および `snapshot_disks` で定義されたディスクを含む Azure リソースグループの名前を指定します。

i 注釈

`backup_method = snapshot` および `snapshot_provider = azure` の場合にのみサポートされます。

スコープグローバル/サーバー/モデル。

azure_subscription_id

`snapshot_instance` および `snapshot_disks` で定義されたインスタンスとストレージボリュームを所有する Azure サブスクリプションを識別します。

i 注釈

`backup_method = snapshot` および `snapshot_provider = azure` の場合にのみサポートされます。

スコープグローバル/サーバー/モデル。

gcp_project

`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 を指定します。

i 注釈

`backup_method = snapshot` および `snapshot_provider = gcp` の場合にのみサポートされます。

スコープグローバル/サーバー/モデル。

gcp_zone

スナップショットバックアップ用にコンピューティングインスタンスとディスクが配置されているアベイラビリティゾーンを示します。

i 注釈

`backup_method = snapshot` および `snapshot_provider = gcp` の場合にのみサポートされます。

スコープサーバー/モデル。

snapshot_disks

このオプションは、クラウドスナップショットバックアップに含めるディスクのコンマ区切りリストです。

i 注釈

`backup_method = snapshot` の場合に必須。

`snapshot_disks` リストには、Postgres データを保存するすべてのディスクが含まれていることを確認します。これらのリストされたディスクにないデータはバックアップに含まれず、リカバリ中に使用できません。

スコープサーバー/モデル。

snapshot_instance

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。

i 注釈

`backup_method = snapshot` の場合に必須。

スコープサーバー/モデル。

snapshot_provider

スナップショットの作成に使用するクラウドプロバイダーの名前。サポートされている値 `aws`、`azure` および `gcp`。

i 注釈

`backup_method = snapshot` の場合に必須。

スコープ グローバル/サーバー/モデル。

13.2.4 フックスクリプト

これらの構成オプションは、フックスクリプトの実行前または実行後に関連しています。

post_archive_retry_script

WAL ファイルがアーカイブされた後に実行するフックスクリプトを指定します。Barman は、SUCCESS (0)、ABORT_CONTINUE (62)、または ABORT_STOP (63) を返すまで、このスクリプトを再試行します。アーカイブ後のシナリオでは、ABORT_STOP は ABORT_CONTINUE と同じ効果があります。

スコープ グローバル/サーバー。

post_archive_script

WAL ファイルがアーカイブされた後に実行するフックスクリプトを指定します `post_archive_retry_script`。

スコープ グローバル/サーバー。

post_backup_retry_script

ベースバックアップの後に実行するフックスクリプトを指定します。Barman は、SUCCESS (0)、ABORT_CONTINUE (62)、または ABORT_STOP (63) を返すまで、このスクリプトを再試行します。バックアップ後シナリオでは、ABORT_STOP は ABORT_CONTINUE と同じ効果があります。

スコープ グローバル/サーバー。

post_backup_script

ベースバックアップの後に実行するフックスクリプトを指定します `post_backup_retry_script`。

スコープ グローバル/サーバー。

post_delete_retry_script

バックアップを削除した後に実行するフックスクリプトを指定します。Barman は、SUCCESS (0)、ABORT_CONTINUE (62)、または ABORT_STOP (63) を返すまで、このスクリプトを再試行します。削除後シナリオでは、ABORT_STOP は ABORT_CONTINUE と同じ効果があります。

スコープ グローバル/サーバー。

post_delete_script

`post_delete_retry_script` に従って、バックアップを削除した後に実行するフックスクリプトを指定します。

スコープ グローバル/サーバー。

post_recovery_retry_script

リカバリ後に実行するフックスクリプトを指定します。Barman は、SUCCESS (0)、ABORT_CONTINUE (62)、または ABORT_STOP (63) を返すまで、このスクリプトを再試行します。リカバリ後のシナリオでは、ABORT_STOP は ABORT_CONTINUE と同じ効果があります。

スコープ グローバル/サーバー。

post_recovery_script

post_recovery_retry_script に従って、リカバリ後に実行するフックスクリプトを指定します。

スコープ グローバル/サーバー。

post_wal_delete_retry_script

WAL ファイルを削除した後に実行するフックスクリプトを指定します。Barman は、SUCCESS (0)、ABORT_CONTINUE (62)、または ABORT_STOP (63) を返すまで、このスクリプトを再試行します。WAL 削除後のシナリオでは、ABORT_STOP は ABORT_CONTINUE と同じ効果があります。

スコープ グローバル/サーバー。

post_wal_delete_script

post_wal_delete_retry_script に従って、WAL ファイルを削除した後に実行するフックスクリプトを指定します。

スコープ グローバル/サーバー。

pre_archive_retry_script

pre_archive_script に続いて、メンテナンス中に WAL ファイルがアーカイブされる前に実行するフックスクリプトを指定します。再試行フックスクリプトとして、Barman は SUCCESS (0)、ABORT_CONTINUE (62)、または ABORT_STOP (63) のいずれかを返すまでスクリプトを繰り返し実行します。ABORT_STOP を返すと、障害がエスカレートし、WAL アーカイブプロセスが停止します。

スコープ グローバル/サーバー。

pre_archive_script

WAL ファイルがメンテナンスによってアーカイブされる前に起動されるフックスクリプトを指定します。

スコープ グローバル/サーバー。

pre_backup_retry_script

pre_backup_script に続いて、ベースバックアップの前に実行するフックスクリプトを指定します。再試行フックスクリプトとして、Barman は SUCCESS (0)、ABORT_CONTINUE (62)、または ABORT_STOP (63) を返すまで、スクリプトの実行を繰り返し試行します。ABORT_STOP を返すと、障害がエスカレートし、バックアッププロセスが中断されます。

スコープ グローバル/サーバー。

pre_backup_script

ベースバックアップを開始する前に実行するフックスクリプトを指定します。

スコープ グローバル/サーバー。

pre_delete_retry_script

`pre_delete_script` に続いて、バックアップを削除する前に実行する再試行フックスクリプトを指定します。再試行フックスクリプトとして、Barman は `SUCCESS` (0)、`ABORT_CONTINUE` (62)、または `ABORT_STOP` (63) を返すまで、スクリプトの実行を繰り返し試行します。`ABORT_STOP` を返すと、障害がエスカレートし、バックアップの削除が中断されます。

スコープ グローバル/サーバー。

pre_delete_script

バックアップを削除する前に実行するフックスクリプトを指定します。

スコープ グローバル/サーバー。

pre_recovery_retry_script

`pre_recovery_script` に従って、リカバリーの前に実行する再試行フックスクリプトを指定します。再試行フックスクリプトとして、Barman は `SUCCESS` (0)、`ABORT_CONTINUE` (62)、または `ABORT_STOP` (63) を返すまで、スクリプトの実行を繰り返し試行します。`ABORT_STOP` を返すと、障害がエスカレートし、リカバリプロセスが中断されます。

スコープ グローバル/サーバー。

pre_recovery_script

リカバリを開始する前に実行するフックスクリプトを指定します。

スコープ グローバル/サーバー。

pre_wal_delete_retry_script

`pre_wal_delete_script` の前に実行される、WAL ファイル削除の再試行フックスクリプトを指定します。再試行フックスクリプトとして、Barman は `SUCCESS` (0)、`ABORT_CONTINUE` (62)、または `ABORT_STOP` (63) を返すまで、スクリプトの実行を繰り返し試行します。`ABORT_STOP` を返すと、障害がエスカレートし、WAL ファイルの削除が中断されます。

スコープ グローバル/サーバー。

pre_wal_delete_script

WAL ファイルを削除する前に実行するフックスクリプトを指定します。

スコープ グローバル/サーバー。

13.2.5 ログ先行書き込み WAL

これらの構成オプションは、Barman が PostgreSQL サーバーの書き込み先行ログ WAL を管理する方法に関連しています。

compression

WAL ファイルの標準の圧縮アルゴリズムを指定します。オプションには次のものが含まれます。 `lz4`、`xz`、`zstd`、`gzip`、`pygzip`、`pigz`、`bzip2`、`pybzip2`、`snappy` および `custom`。

バージョン 3.16 で非推奨: `pygzip` および `pybzip2` 圧縮オプションは非推奨であり、将来のリリースでは削除される予定です。代わりに、同等の `gzip` および `bzip2` を使用してください。

バージョン 3.17 で非推奨: `custom` 圧縮オプションと関連する構成オプション `custom_compression_filter`、`custom_decompression_filter`、および `custom_compression_magic` は非推奨であり、将来のリリースでは削除されます。代わりに、ビルトイン圧縮アルゴリズムのいずれかを使用します。

注釈

これらのオプションはすべて、圧縮が発生する場所にモジュールをインストールする必要があります。

`custom` オプションはカスタム圧縮用であり、次のオプションも設定する必要があります。

- `custom_compression_filter` 圧縮フィルター。
- `custom_decompression_filter` 解凍フィルタ
- `custom_compression_magic` カスタム圧縮 wal ファイルを識別する 16 進文字列。

スコープグローバル/サーバー/モデル。

バージョン 3.17 で非推奨: この構成オプションは非推奨であり、将来のリリースでは削除される予定です。

`custom_compression_filter`

WAL ファイルのカスタム圧縮アルゴリズムを指定します。これは `bash` コマンドを作成するために内部で使われる `string` である必要があります、次の文字列 `> /"$2/" < /"$1/";` の先頭にあります。標準出力に書き込み、入力ファイルは削除しません。

Tip

```
custom_compression_filter = /"xz -c/"
```

これは、`xz -c > /"$2/" < /"$1/";` を実行するのと同じです。

スコープグローバル/サーバー/モデル。

バージョン 3.17 で非推奨: この構成オプションは非推奨であり、将来のリリースでは削除される予定です。

`custom_compression_magic`

カスタムマジック値を定義して、WAL ファイルで使われるカスタム圧縮アルゴリズムを識別します。これが設定されている場合、Barman は、指定されたアルゴリズムで既に圧縮された WAL にカスタム圧縮を適用することを回避します。構成されていない場合、Barman は事前圧縮されたものを含むすべての WAL ファイルにカスタム圧縮を適用します。

Tip

たとえば、`xz` 圧縮アルゴリズムでは、マジックナンバーを使用して `.xz` ファイルのフォーマットを検出します。

xz ファイルの場合、マジックナンバーは次のバイトシーケンスです。

マジックナンバー `FD 37 7A 58 5A 00`

16 進数表現では、これは次のように表すことができます。

16 進数文字列 fd377a585a00

Barman は `custom_compression_magic` の値の先頭に `0x` が付けられることを想定しているため、次のようにその構成オプションを設定する必要があります。

```
custom_compression_magic = 0xfd377a585a00
```

参照 [xz-file-format](#)

スコープグローバル/サーバー/モデル。

バージョン 3.17 で非推奨: この構成オプションは非推奨であり、将来のリリースでは削除される予定です。

custom_decompression_filter

圧縮 WAL ファイルのカスタム解凍アルゴリズムを指定します。これは `bash` コマンドを作成するために内部で使用される `string` である必要があり、次の文字列 `> /"$2/" < /"$1/"`; の先頭にあります。使用される圧縮アルゴリズムに対応する必要があります。

💡 Tip

```
custom_compression_filter = "xz -c -d/"
```

これは、`xz -c -d > /"$2/" < /"$1/"`; を実行するのと同じです。

スコープグローバル/サーバー/モデル。

compression_level

選択した圧縮アルゴリズムで使用される圧縮レベルを指定します。有効な値は、選択したアルゴリズムまたは事前定義されたラベルのいずれかショートカットとして機能する `low`、`medium`、および `high` のサポートされている範囲内の整数です。

- `low` 低レベルの圧縮を使用し、圧縮率よりも圧縮速度を優先します。
- `medium` 中レベルの圧縮を使用し、圧縮速度と圧縮率のバランスをとります。
- `high` 高レベルの圧縮を使用し、圧縮速度よりも圧縮率を優先します。

事前定義されたラベルは、以下に詳細を示すように、アルゴリズム固有のレベルにマッピングされます。

表 1: 圧縮レベル

アルゴリズム	レベルレンジ	低い	中	ハイ
lz4	0～16	0	6	10
xz	1～9	1	3	5
zstd	-22～22	1	4	9
gzip、pygzip および pigz	1～9	1	6	9
bzip2 および pybzip2	1～9	1	5	9

指定された圧縮レベルがアルゴリズムの最大レベルより大きい場合、その最大レベルが使用されます。同様に、最小レベルよりも低い場合、その最小レベルが使用されます。デフォルト値は `medium` です。

スコープグローバル/サーバー/モデル。

incoming_wals_directory

着信 WAL ファイルをアーカイブするディレクトリを指定します。 `archiver` を有効にする必要があります。デフォルト `<backup_directory>/incoming`。

スコープサーバー。

last_wal_maximum_age

最新のアーカイブされた WAL ファイルが該当する時間フレームを定義します。最新の WAL ファイルがこの期間よりも古い場合、`barman check` コマンドはエラーを報告します。空のままの場合デフォルト、WAL ファイルの古さはチェックされません。フォーマットは `last_backup_maximum_age` と同じです。

スコープグローバル/サーバー/モデル。

max_incoming_wals_queue

`barman check` コマンドがエラーを返す前に、着信キューストリーミングプールとアーカイブプールの両方を含む WAL ファイルの最大数を定義します。デフォルトは `None` 無効です。

スコープグローバル/サーバー/モデル。

streaming_wals_directory

ストリーミング WAL ファイルのためのディレクトリ。デフォルトの `<backup_directory>/streaming`。

注釈

このオプションは、`streaming_archiver` がアクティブな場合に適用されます。

スコープサーバー。

wal_conninfo

`wal_conninfo` 接続文字列は、Barman が WAL を受信するレプリケーションスロットのステータスを監視するために使用されます。指定すると、これらのチェックでは `wal_streaming_conninfo` より優先されます。`wal_conninfo` が設定されていないが、`wal_streaming_conninfo` が設定されている場合、`wal_conninfo` は `wal_streaming_conninfo` にフォールバックします。`wal_conninfo` も `wal_streaming_conninfo` も設定されていない場合、`wal_conninfo` は `conninfo` にフォールバックします。どちらの接続文字列も、`streaming_conninfo` および `conninfo` で定義されたものと同じクラスター内の Postgres インスタンスにアクセスする必要があります。`wal_conninfo` と `wal_streaming_conninfo` の両方が設定されている場合、`wal_conninfo` のみが設定を読み取り、レプリケーションスロットのステータスを確認するために適切な権限を必要とします。ただし、`wal_streaming_conninfo` のみが設定されている場合、これらのタスクを実行するには必要な権限が必要です。必要な権限には、`pg_monitor`、`pg_read_all_settings` と `pg_read_all_stats` の両方、またはスーパーユーザー権限などのロールが含まれます。

スコープサーバー/モデル。

wal_streaming_conninfo

この接続文字列は、Barman が Postgres サーバーに接続するために使用され、ストリーミングレプリケーションを介して WAL セグメントを受信し、`wal_conninfo` が設定されていない場合にレプリケーションスロットのステータスを確認します。指定しない場合、Barman はこれらのタスクにデフォルトで `streaming_conninfo` を使用します。`wal_streaming_conninfo` は、`streaming_conninfo` で定義されたのと同じクラスター内の Postgres インスタンスに接続する必要があり、ストリーミングレプリケーションをサポートする必要があります。`wal_streaming_conninfo` と `wal_conninfo` の両方が設定されている場合、`wal_conninfo` のみが設定を読み取り、レプリケーションスロットのステータスを確認するために必要な権限を必要とします。`wal_streaming_conninfo` のみを指定する場合、これらの権限が必要です。必要な権限には、`pg_monitor`、`pg_read_all_settings` と `pg_read_all_stats` の両方、またはスーパーユーザー権限などのロールが含まれます。

スコープサーバー/モデル。

wals_directory

WAL ファイルを含むディレクトリ。デフォルトの `<backup_directory>/wals`。

スコープサーバー。

xlogdb_directory

SERVER-xlog.db ファイルのカスタムディレクトリ。SERVER はサーバー名です。このファイルには、アーカイブされた WAL ファイルのメタデータが保存され、Barman によって内部的に使用されます。設定されていない場合、デフォルトの `wals_directory` の値が使用されます。

スコープ グローバル/サーバー。

13.2.6 リストア

これらの構成オプションは、Barman が復元バックアップを管理する方法に関連しています。

local_staging_path

リカバリ中にブロックレベルのインクリメンタルバックアップを結合するためのローカルパスを指定します。この場所には、新しい合成バックアップを一時的に保存するための十分な領域が必要です。ブロックレベルのインクリメンタルバックアップからのリカバリーに必要です。

❗ 注釈

`backup_method = postgres` の場合にのみ適用されます。

バージョン 3.15 で非推奨: `local_staging_path` は非推奨であり、将来のリリースで削除される予定です。代わりに `staging_path` および `staging_location` を使用してください。

スコープグローバル/サーバー/モデル。

recovery_options

リカバリ操作のオプション。現在、`get-wal` および `delta-restore` のみがサポートされています。`get-wal` は、リカバリ構成での基本的な `restore_command` の作成を有効にします。これは、`barman get-wal` コマンドを使用して Barman の WAL アーカイブから WAL ファイルを直接取得します。`delta-restore` を

使用すると、barman は既存のカスタムテーブルスペースとともに、既存の宛先ディレクトリにバックアップを上書きして復元できます。この設定は、コンマ区切りの値のリストを受け入れ、デフォルトで空です。

スコープグローバル/サーバー/モデル。

recovery_staging_path

圧縮バックアップからファイルをステージングするための復旧ホスト上のパスを指定します。この場所には、圧縮バックアップを一時的に保存するための十分な領域が必要です。

注釈

圧縮バックアップにのみ適用されます。

バージョン 3.15 で非推奨: `recovery_staging_path` は非推奨であり、将来のリリースで削除される予定です。代わりに `staging_path` および `staging_location` を使用してください。

スコープグローバル/サーバー/モデル。

staging_path

リストア中に中間ファイルがステージングされるパス。圧縮されたバックアップを復元する場合、最終的な宛先にコピーする前に、解凍の一時的な場所として機能します。インクリメンタルバックアップを復元する場合、最終的な宛先にコピーする前にバックアップが結合されます。この場所には、解凍/結合されたバックアップを保存するための十分な領域が必要です。デフォルトは `/tmp` です。

スコープグローバル/サーバー/モデル。

staging_location

`staging_path` がローカルパスかリモートパスであるかを指定します。有効な値は `local` および `remote` です。デフォルトは `local` です。

スコープグローバル/サーバー/モデル。

combine_mode

リストア中にインクリメンタルバックアップを組み合わせる場合、`pg_combinebackup` のコピーモードを指定します。

オプションには次のものが含まれます

- **copy** デフォルト インクリメンタルバックアップを組み合わせる場合、標準のファイルコピーを使用します。
- **link** インクリメンタルバックアップを組み合わせる場合、ハードリンクを使用します。合成バックアップの再構成はより速くなりファイルコピーなし、使用するディスク領域は少なくなります。
- **clone** 新しいデータディレクトリにファイルをコピーする代わりに、効率的なファイルクローン作成一部のシステムでは「**reflink**」としても知られますを使用します。これにより、データファイルのほぼ瞬時のコピーが行われます。
- **copy-file-range** `copy_file_range` システムコールを使用して、効率的なコピーを行います。一部のファイルシステムでは、これにより **clone** と同様の結果が得られ、物理ディスクブロックを共有

しますが、他のファイルシステムでは、ブロックをコピーする場合がありますが、最適化されたパスを介して実行します。

各モードの詳細と制限については、[pg_combinebackup documentation](#) を参照してください。

スコープグローバル/サーバー/モデル。

13.2.7 リテンション ポリシー

これらの構成オプションは、Barman がバックアップの保持ポリシーを管理する方法に関連しています。

last_backup_maximum_age

最新のバックアップが含まれる時間フレームを定義します。最新のバックアップがこの期間よりも古い場合、barman check コマンドはエラーを報告します。空のままの場合デフォルト、最新のバックアップは常に有効であるとみなされます。受け入れられる形式は `/"n {DAYS|WEEKS|MONTHS|HOURS}"/` で、`n` は 0 より大きい整数です。

スコープグローバル/サーバー/モデル。

last_backup_minimum_size

最近成功したバックアップの最小許容サイズを指定します。最新のバックアップがこのサイズより小さい場合、barman check コマンドはエラーを報告します。空のままの場合デフォルト、最新のバックアップは常に有効であるとみなされます。受け入れられるフォーマットは `/"n {k|Ki|M|Mi|G|Gi|T|Ti}"/` で、大文字と小文字が区別されます。ここで、`n` は、ゼロより大きい整数で、オプションの SI または IEC サフィックスが付きます。`k` は `k = 1000` のキロを表し、`Ki` はキロバイト `Ki = 1024` を表します。残りのオプションは、より大きな測定単位についても同じ理由があります。

スコープグローバル/サーバー/モデル。

retention_policy

バックアップと WAL ファイルを保持する期間を定義します。このオプションを空白のままにした場合、保持ポリシーは適用されません。オプションには、冗長性とリカバリウィンドウポリシーが含まれます。

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

- **retention_policy = REDUNDANCY 2** は、バックアップカタログに 2 つのバックアップのみを保持し、新しいバックアップが作成されると、古いバックアップを自動的に削除します。数値は正の整数である必要があります。
- **retention_policy = RECOVERY WINDOW OF 2 DAYS** は、過去 2 日間の任意の時点にリカバリするために必要なバックアップのみを保持し、古いバックアップは自動的に削除します。期間番号は正の整数である必要があり、次のオプションを適用できます。DAYS、WEEKS、MONTHS。

スコープグローバル/サーバー/モデル。

retention_policy_mode

保持ポリシーを適用するためのモード。現在は auto のみをサポートしています。

スコープグローバル/サーバー/モデル。

wal_retention_policy

WAL ファイルを保持するためのポリシー。現在、`main` のみが使用できます。

スコープグローバル/サーバー/モデル。

13.3 構成モデル

構成モデルは、Postgres サーバーを特定の `cluster` 名前の下で編成することにより、Postgres サーバーの構成オーバーライドを管理および適用する体系的なアプローチを提供します。

13.3.1 目的

構成モデルの主な目的は、同じ `cluster` によってグループ化された Postgres サーバーの構成設定の管理を簡素化することです。モデルを使用することにより、一般的な構成オーバーライドのセットを適用でき、業務効率が向上します。これらはクラスター環境で特に有益であり、フェールオーバーイベント中に利用できるさまざまな構成モデルを作成できます。

13.3.2 アプリケーション

モデルファイルで定義された構成は、モデルで指定された同じ `cluster` 名前を共有する Postgres サーバーに適用できます。したがって、そのモデルを利用するサーバーはこれらの設定を継承でき、すべてのサーバーにわたって一貫した適応可能な構成を促進します。

13.3.3 使用法

モデルオプションは、サーバーセクションと同じ方法で識別されるモデルセクション内でのみ定義できます。サーバーセクションとモデルセクションの識別子の間に競合がないことを確認することが重要です。

構成モデルを適用するには、`barman config-switch SERVER_NAME MODEL_NAME` を実行します。このコマンドは、指定されたクラスター名に関連付けられている関連 Barman サーバーへのモデルのオーバーライドの適用を容易にします。

オーバーライドを削除する場合は、モデル構成ファイルの削除だけでは効果がないため、次のようにコマンドで `--reset` 引数を使用して削除できます。`barman config-switch SERVER_NAME --reset`。

注釈

`config-switch` コマンドは、モデル名が存在し、サーバーと同じ `cluster` に関連付けられている場合にのみ成功します。さらに、一度にアクティブなモデルは 1 つだけです。異なるモデルでコマンドを複数回実行した場合、最後のモデルで定義されたオーバーライドのみが適用されます。

モデルからすべてのオプションを構成できるわけではありません。使用可能な構成の範囲を確認して、モデルに適用する設定を決定してください。

13.3.4 特典

- 一貫性 クラスター内の複数の Barman サーバー全体で均一な構成を保証します。
- 効率 一元的な更新とオーバーライドを許可することにより、構成管理を簡素化します。
- 柔軟性 複数のモデルファイルを使用でき、必要に応じてさまざまなオーバーライドのセットを定義できる機能を提供します。

第14章 コマンドリファレンス

Barman には、`barman` という名前のコマンドラインインターフェイスがあり、これは基本的に Barman のバックエンドと対話するために使用されます。

`barman` の各サブコマンドに入る前に、`barman` には、すべてのサブコマンドで使用可能なグローバルオプションがあることに注意してください。これらのオプションは、サブコマンドの動作を変更でき、次のように使用できます。

14.1 barman

14.1.1 概要

```
barman
  [ { -c | --config } CONFIG ]
  [ { --color | --colour } { never | always | auto } ]
  [ { -d | --debug } ]
  [ { -f | --format } { json | console } ]
  [ { -h | --help } ]
  [ --log-level { NOTSET | DEBUG | INFO | WARNING | ERROR | CRITICAL } ]
  [ { -q | --quiet } ]
  [ { -v | --version } ]
  [ SUBCOMMAND ]
```

注釈

これは、概要の構文です。

- 角かっこ間のオプションはオプションナル。
- 中括弧間のオプションは、いずれかの設定操作の選択を表します。
- `[ ... ]` を使用したオプションは複数回指定できます。
- 大文字で書かれたものは、値を与える必要があるリテラルを表します。

以下のセクションで `barman` サブコマンドを説明するときに、これと同じ構文を使用します。

また、以降のセクションでサブコマンドを説明するときは、コマンドの概要を `SUBCOMMAND` の代わりに見る必要があります。

14.1.2 パラメーター

-c / --config CONFIG

使用する構成ファイルを指定します。提供されない場合、デフォルトの `/etc/barman.conf`。

--color / --colour { never | always | auto }

出力で色を使用するかどうかを制御します。デフォルトは `auto` です。オプションは次のとおりです。

- `never` 色を使用しません。
- `always` 常にカラーを使用します。
- `auto` 出力が端末にある場合は色を使用します。

-d / --debug

デバッグ出力を有効にします。デフォルトは `false` です。トラブルシューティングのための詳細なログ情報を提供します。

-f / --format { json | console }

出力形式を指定します。オプションは次のとおりです。

- `json` JSON 形式で出力します。
- `console` 人間が判読できる形式で出力しますデフォルト。

-h / --help

ヘルプメッセージを表示して終了します。コマンドの使用法に関する情報を提供します。

--log-level { NOTSET | DEBUG | INFO | WARNING | ERROR | CRITICAL }

デフォルトのログレベルをオーバーライドします。オプションは次のとおりです。

- **NOTSET** これは、特定のロギングレベルが設定されていない場合のデフォルトレベルです。これは本質的にログメッセージの"/フィルタリングなし"を意味し、構成で設定されたレベルに従ってすべてのメッセージを処理できます。
- **DEBUG** このレベルは、詳細な診断情報に使用され、多くの場合、開発者が問題を診断する際に役立ちます。これには、プログラムの実行のトレースを支援することを目的とした、より粒度が高く詳細なメッセージが含まれています。
- **INFO** このレベルは、アプリケーションの通常の実行に関する一般的な情報を提供します。これは、アプリケーションの進行状況を示すメッセージ、または実行フローのキーポイントを強調表示するために使用されますが、問題は示していません。
- **WARNING** このレベルは、予期しない何かが発生したか、潜在的な問題がある可能性を示します。これは、重要ではないが懸念される可能性のあるメッセージに使用され、注意が必要である可能性があることを示します。
- **ERROR** このレベルは、特定の操作を正常に完了できないエラーが発生した場合に使用されます。これは、対処する必要があるが、必ずしもアプリケーションの実行を停止するわけではない重要な問題を示すために使用されます。
- **CRITICAL** これは、最高レベルの重大度であり、アプリケーションを終了させた可能性が高い、またはすぐに対処しないと重大な結果をもたらす重大なエラーを示します。緊急の注意が必要な重要な問題に使用されます。

-q / --quiet

すべての出力を抑制します。cron ジョブまたは自動スクリプトに役立ちます。

-v / --version

プログラムバージョン番号を表示して終了します。

14.2 ショートカット

一部のコマンドでは、次のショートカットまたはエイリアスを使用して、特定のサーバーのバックアップを識別できます。具体的には、`all` ショートカットを使用して、すべてのサーバーを識別できます。

Shortcut	Description
all	使用可能なすべてのサーバー
first/oldest	サーバーの使用可能な最も古いバックアップ時系列。
last/latest	サーバーで使用可能な最新のバックアップ時系列。
last-full/latest-full	rsync または postgres 方法で取得された最新のフルバックアップ。
last-failed	失敗した最新のバックアップ時系列。

14.3 終了ステータス

ステータスコード **0** は **0** を意味し、ステータスコード **1** は **1** を意味します。

14.4 サブコマンド

`barman` は、いくつかの便利な操作を公開します。このセクションでは、それぞれを説明することを目的としています。

以下のセクションでは、`barman` によって実装される各コマンドの説明を見つけます。これらのコマンドの一部には、このドキュメントの別のメインセクションに詳細な情報が含まれている場合があります。その場合、すぐに移動できるリファレンスが提供されています。

14.4.1 `barman archive-wal`

概要

```
archive-wal
[ { -h | --help } ]
SERVER_NAME
```

説明

標準の `archive_command` または `pg_receivewal` を使用したストリーミングレプリケーションから受信した WAL ファイルを取得し、サーバーの WAL アーカイブに保存します。構成ファイルで `compression` を有効にしている場合、WAL ファイルはアーカイブされる前に圧縮されます。

パラメーター

SERVER_NAME

barman ノードのサーバーの名前。

-h / --help

ヘルプメッセージを表示して終了します。コマンドの使用法に関する情報を提供します。

14.4.2 barman backup

概要

```
backup
  [ --bwlimit KBPS ]
  [ { -h | --help } ]
  [ --incremental BACKUP_ID ]
  [ --immediate-checkpoint ]
  [ { -j | --jobs } PARALLEL_WORKERS ]
  [ --jobs-start-batch-period PERIOD ]
  [ --jobs-start-batch-size SIZE ]
  [ --keepalive-interval SECONDS ]
  [ --manifest ]
  [ --name NAME ]
  [ --no-immediate-checkpoint ]
  [ --no-manifest ]
  [ --no-retry ]
  [ --retry-sleep SECONDS ]
  [ --retry-times NUMBER ]
  [ --reuse-backup { off | copy | link } ]
  [ { --wait | -w } ]
  [ --wait-timeout SECONDS ]
  SERVER_NAME [ SERVER_NAME ... ]
```

説明

PostgreSQL サーバーのバックアップを実行します。Barman は、グローバル構成ファイルとサーバー構成ファイルで指定されたパラメーターを使用します。Barman ノードで構成されているすべてのサーバーからバックアップを実行するには、サーバー名の代わりに all ショートカットを指定します。複数のサーバー名を順番に指定して、特定のサーバーのバックアップを実行することもできます。

パラメーター

SERVER_NAME

barman ノードのサーバーの名前。

--bwlimit

最大転送速度をキロバイト/秒で指定します。値 0 は制限がないことを示します。この設定は、bandwidth_limit 構成オプションをオーバーライドします。

-h / --help

ヘルプメッセージを表示して終了します。コマンドの使用法に関する情報を提供します。

--incremental

ブロックレベルのインクリメンタル バックアップを実行します。 `BACKUP_ID` または以前のバックアップへのショートカットを提供する必要があります。これは、インクリメンタルバックアップの親バックアップとして機能します。

i 注釈

バックアップと親バックアップには `backup_method=postgres` が必要です。

--immediate-checkpoint

初期チェックポイントをできるだけ早く実行させ、構成ファイルの `immediate_checkpoint` パラメーターに設定された値をオーバーライドします。

-j / --jobs

バックアップ中にファイルのコピーに使用するパラレルワーカーの数を指定します。この設定は、構成ファイルで指定されている場合、`parallel_jobs` パラメーターをオーバーライドします。

--jobs-start-batch-period

ジョブの単一のバッチを開始する時間を秒単位で指定します。この値は、構成ファイルで設定されている場合、`parallel_jobs_start_batch_period` パラメーターをオーバーライドします。デフォルトは 1 秒です。

--jobs-start-batch-size

単一のバッチで開始するパラレルワーカーの最大数を指定します。この値は、構成ファイルで定義されている場合、`parallel_jobs_start_batch_size` パラメーターをオーバーライドします。デフォルトは 10 ワーカーです。

--keepalive-interval

サーバーにハートビートクエリを送信して、Rsync バックアップ中に libpq 接続をアクティブに保つ間隔を秒単位で指定します。デフォルトは 60 秒です。0 の値はハートビートを無効にします。

--manifest

バックアップの完了時に、バックアップマニフェストファイルの作成を強制します。構成ファイルから `autogenerate_manifest` パラメーターをオーバーライドします。rsync バックアップ戦略にのみ適用されます。

--name

barman コマンドでバックアップ ID の代わりに使用できるこのバックアップのフレンドリ名を指定します。

--no-immediate-checkpoint

構成ファイルの `immediate_checkpoint` パラメーターに設定された値をオーバーライドして、チェックポイントが実行されるまでバックアップを強制的に待機します。

--no-manifest

バックアップの完了時に、バックアップマニフェストファイルの自動作成を無効にします。この設定は、構成ファイルの `autogenerate_manifest` パラメーターをオーバーライドし、rsync バックアップ

ブストラテジーにのみ適用されます。

--no-retry

エラーが発生した場合の再試行はありません。 `--retry-times 0` の設定と同じです。

--retry-sleep

コピーが失敗した後、再試行するまでに待機する秒数を指定します。この設定は、バックアップとリカバリー操作の両方に適用され、構成ファイルで定義されている場合 `basebackup_retry_sleep` パラメーターをオーバーライドします。

--retry-times

エラーが発生した場合に、ベースバックアップコピーを再試行する回数を指定します。これは、バックアップとリカバリー操作の両方に適用され、構成ファイルで設定されている場合 `basebackup_retry_times` パラメーターをオーバーライドします。

--reuse-backup

構成ファイルで構成された `reuse_backup` オプションの動作をオーバーライドします。可能な値は次のとおりです。

- **off** 最後の使用可能なバックアップを再利用しません。
- **copy** サーバーの最後に利用可能なバックアップを再利用し、変更されていないファイルのコピーを作成しますバックアップ時間を削減します。
- **link** デフォルト サーバーの最後に利用可能なバックアップを再利用し、変更されていないファイルへのハードリンクを作成しますバックアップ時間と領域の両方を節約します。

i 注釈

これは、最後に使用可能なバックアップが `backup_method=rsync` で実行された場合にのみ効果があります。

--wait / -w

ベースバックアップに必要なすべての WAL ファイルがアーカイブされるまで待ちます。

--wait-timeout

タイムアウトになる前に、必要な WAL ファイルがアーカイブされるのを待機する時間を秒単位で指定します。

14.4.3 barman check-backup

概要

check-backup

```
[ { -h | --help } ]
SERVER_NAME BACKUP_ID
```

説明

物理バックアップの一貫性を検証するために必要な WAL ファイルがすべて適切にアーカイブされていることを確認します。このコマンドは、cron ジョブによって、および各バックアップ操作の最後に自動的に実行されます。BACKUP_ID の代わりにショートカットを使用できます。

パラメーター

SERVER_NAME

barman ノードのサーバーの名前。

BACKUP_ID

barman カタログのバックアップの ID。

-h / --help

ヘルプメッセージを表示して終了します。コマンドの使用法に関する情報を提供します。

14.4.4 barman check

概要

```
check
  [ { -h | --help } ]
  [ --nagios ]
  SERVER_NAME [ SERVER_NAME ... ]
```

説明

SSH 接続、Postgres バージョン、構成とバックアップディレクトリ、アーカイブとストリーミングプロセス、レプリケーションスロットなど、サーバーに関するステータス情報を表示します。all をショートカットとして使用して、構成されているすべてのサーバーの診断情報を表示します。

パラメーター

SERVER_NAME

barman ノードのサーバーの名前。

-h / --help

ヘルプメッセージを表示して終了します。コマンドの使用法に関する情報を提供します。

--nagios

Nagios プラグイン互換出力。

14.4.5 barman config-switch

概要

```
config-switch
  [ { -h | --help } ]
  SERVER_NAME { --reset | MODEL_NAME }
```

説明

モデルから Barman のサーバーに一連の構成オーバーライドを適用します。最終的な構成は、サーバーの既存の設定をモデルで指定された設定と組み合わせるかオーバーライドします。 `--reset` 引数を使用してサーバー構成をリセットできます。

注釈

特定のサーバーで一度に 1 つのモデルのみをアクティブにできます。

パラメーター

SERVER_NAME

barman ノードのサーバーの名前。

MODEL_NAME

モデルの名前。

-h / --help

ヘルプメッセージを表示して終了します。コマンドの使用法に関する情報を提供します。

--reset

サーバーの構成をリセットします。

14.4.6 barman config-update

概要

```
config-update
  [ { -h | --help } ]
  STRING
```

説明

Barman でサーバーおよび/またはモデルの構成を作成または更新します。パラメーターは、ドキュメントの配列を含む JSON 文字列である必要があります。各ドキュメントには、サーバーまたはモデルの `scope` キーと、スコープ値に応じて `server_name` または `model_name` キーのいずれかを含める必要があります。さらに、ドキュメントには、Barman 構成オプションとその目的の値を表す他のキーを含める必要があります。

i 注釈

barman config-update コマンドは、barman_home ディレクトリにある .barman.auto.conf という名前のファイルに構成オプションを書き込みます。この構成ファイルは優先順位が高く、グローバル Barman 構成ファイル通常は /etc/barman.conf および configuration_files_directory で指定されたインクルードファイル通常は /etc/barman.d のファイルからの値をオーバーライドします。後でこれらのファイルの構成オプションを手動で変更する場合は、これに注意してください。

パラメーター**STRING**

JSON 形式の文字列のリスト。

-h / --help

ヘルプメッセージを表示して終了します。コマンドの使用法に関する情報を提供します。

例

```
JSON_STRING='[{ "scope": "server", "server_name": "my_server", "archiver": "on", "streaming_archiver": "off" }]'
```

14.4.7 barman cron**概要****cron**

```
[ { -h | --help } ]
[ --keep-descriptors ]
```

説明

保持ポリシーの適用や WAL ファイルの管理などのメンテナンスタスクを実行します。

パラメーター**-h / --help**

ヘルプメッセージを表示して終了します。コマンドの使用法に関する情報を提供します。

--keep-descriptors

Barman サブプロセスの `stdout` および `stderr` ストリームをメインプロセスに接続したままにします。これは、Docker ベースのインストールに特に役立ちます。

14.4.8 barman delete

概要

```
delete
  [ { -h | --help } ]
  SERVER_NAME BACKUP_ID
```

説明

指定されたバックアップを削除します。BACKUP_ID の代わりにショートカットを使用できます。

パラメーター

SERVER_NAME

barman ノードのサーバーの名前

BACKUP_ID

barman カタログのバックアップの ID。

-h / --help

ヘルプメッセージを表示して終了します。コマンドの使用法に関する情報を提供します。

14.4.9 barman diagnose

概要

```
diagnose
  [ { -h | --help } ]
  [ --show-config-source ]
```

説明

Barman がインストールされているサーバーである Barman ノード、および構成されているすべての Postgres サーバーに関する診断情報を表示します。これには、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、すべてのサーバーの現在の構成とステータスなどの詳細が含まれます。

パラメーター

-h / --help

ヘルプメッセージを表示して終了します。コマンドの使用法に関する情報を提供します。

--show-config-source

各構成オプションの有効な値を提供するソースファイルを含めます。

14.4.10 barman generate-manifest

概要

```
generate-manifest
  [ { -h | --help } ]
  SERVER_NAME BACKUP_ID
```

説明

バックアップ用の `backup_manifest` ファイルを生成します。BACKUP_ID の代わりにショートカットを使用できます。

パラメーター

SERVER_NAME

barman ノードのサーバーの名前

BACKUP_ID

barman カタログのバックアップの ID。

-h / --help

ヘルプメッセージを表示して終了します。コマンドの使用法に関する情報を提供します。

14.4.11 barman get-wal

概要

```
get-wal
  [ { --bzip | -j } ]
  [ { --gzip | -z | -x } ]
  [ { -h | --help } ]
  [ --keep-compression ]
  [ { --output-directory | -o } OUTPUT_DIRECTORY ]
  [ { --peek | -p } VALUE ]
  [ { -P | --partial } ]
  [ { -t | --test } ]
  SERVER_NAME WAL_NAME
```

説明

指定されたサーバーの xlog アーカイブから WAL ファイルを取得します。デフォルトでは、要求された WAL ファイルが見つかったら、非圧縮コンテンツとして STDOUT に返されます。

パラメーター

SERVER_NAME

barman ノードのサーバーの名前

WAL_NAME

barman カタログのバックアップの ID。

--bzip2 / -j

出力は bzip2 を使用して圧縮されます。

--gzip / -z / -x

出力は gzip を使用して圧縮されます。

-h / --help

ヘルプメッセージを表示して終了します。コマンドの使用法に関する情報を提供します。

--keep-compression

ファイルの内容は解凍しないでください。出力は元の圧縮ファイルになります。

--output-directory / -o

barman が WAL ファイルを保存する宛先ディレクトリ。

--peek / -p

指定された WAL ファイルからこのパラメーターで指定された値までの WAL ファイルを取得するには、1 以上の整数値を指定します。このオプションを使用する場合、`get-wal` は、行ごとに 1 つの名前を含む 0 から指定された WAL セグメント名のリストを結果ます。

-P / --partial

さらに、部分的な WAL ファイル `.partial` を収集します。

-t / --test

WAL 取得のために Barman で指定された Postgres サーバーの接続と構成の両方をテストします。このオプションを使用する場合、必須の `WAL_NAME` 引数は無視されます。

警告

`-z / --gzip` および `-j / --bzip2` オプションは非推奨であり、将来的に削除される予定です。WAL 圧縮の場合、`compression` 構成オプションを介して Barman サーバーで直接有効にしてください。

14.4.12 barman keep

概要

keep

```
[ { -h | --help } ]
{ { -r | --release } | { -s | --status } | --target { full | standalone } }
SERVER_NAME BACKUP_ID
```

説明

指定されたバックアップを無期限に保持するアーカイブバックアップとして **target** でマークし、アクティブな保持ポリシーをオーバーライドします。バックアップの **keep status** とバックアップの **keep** マーク **release** を確認することもできます。BACKUP_ID の代わりにショートカットを使用できます。

注釈

インクリメンタルバックアップは、**keep** コマンドの対象外です。この制限は、インクリメンタルバックアップが親バックアップに依存し、独自にリカバリーに使用できないために存在します。独立して使用できないため、ベースなしでインクリメンタルを保持しても意味がありません。

インクリメンタルバックアップに **keep** を適用しようとする、Barman は、インクリメンタルバックアップを保持できないことを示すエラーを返します。

Only the full base backup — the one that starts the backup chain— can be marked with keep.

パラメーター

SERVER_NAME

barman ノードのサーバーの名前

BACKUP_ID

barman カタログのバックアップの ID。

-h / --help

ヘルプメッセージを表示して終了します。コマンドの使用法に関する情報を提供します。

-r / --release

このバックアップからキープマークを解放します。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。

-s / --status

バックアップのアーカイブステータスを報告します。ステータスは、アーカイブバックアップの場合は **full** または **standalone**、アーカイブとして指定されていないバックアップの場合は **nokeep** になります。

--target

アーカイブバックアップのリカバリターゲットを定義します。可能な値は次のとおりです。

- **full** バックアップは、最新のポイント・インタイムにリカバリーするために使用できます。これをサポートするために、Barman は必要なすべての WAL を保持して、バックアップの一貫性と後続の WAL を維持します。
- **standalone** バックアップは、サーバーをバックアップ時の状態に復元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。

14.4.13 barman list-backups

概要

```
list-backups
  [ { -h | --help } ]
  [ --minimal ]
  SERVER_NAME
```

説明

サーバーの使用可能なバックアップを表示します。このコマンドは、バックアップ ID とバックアップタイプの両方を取得するのに役立ちます。このコマンドの詳細については、[Catalog usage](#) を参照してください。

パラメーター

SERVER_NAME

barman ノードのサーバーの名前

-h / --help

ヘルプメッセージを表示して終了します。コマンドの使用法に関する情報を提供します。

--minimal

機械可読出力。

14.4.14 barman list-files

概要

```
list-files
  [ { -h | --help } ]
  [ --target { data | full | standalone | wal } ]
  [ --list-empty-directories ]
  SERVER_NAME BACKUP_ID
```

説明

特定のバックアップ内のすべてのファイルをリストします。BACKUP_ID の代わりにショートカットを使用できます。

パラメーター

SERVER_NAME

barman ノードのサーバーの名前

BACKUP_ID

barman カタログのバックアップの ID。

-h / --help

ヘルプメッセージを表示して終了します。コマンドの使用法に関する情報を提供します。

--target

リストする特定のファイルを定義します。可能な値は次のとおりです。

- **standalone** デフォルト 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。
- **data** データファイルのみをリストします。
- **wal** ベースバックアップの開始とログの最後または次のベースバックアップの開始までのすべての WAL ファイルをリストします指定されたベースバックアップが使用可能な最新のものかどうかに応じて。
- **full data + wal** と同じ。

--list-empty-directories

空のディレクトリをリストに追加します。

14.4.15 barman list-processes

概要

```
list-processes
[ { -h | --help } ]
SERVER_NAME
```

説明

list-processes サブコマンドは、Barman サーバーのすべてのアクティブなサブプロセスを出力します。アクティブなサブプロセスごとにプロセス識別子 PID と、対応するバーマンタスクが表示されます。

パラメーター

SERVER_NAME

アクティブなサブプロセスをリストするサーバーの名前。

-h / --help

ヘルプメッセージを表示して終了します。

14.4.16 barman list-servers

概要

```
list-servers
[ { -h | --help } ]
[ --minimal ]
```

説明

構成されているすべてのサーバーをその説明とともに表示します。

パラメーター

-h / --help

ヘルプメッセージを表示して終了します。コマンドの使用法に関する情報を提供します。

--minimal

機械可読出力。

14.4.17 barman lock-directory-cleanup

概要

```
lock-directory-cleanup
[ { -h | --help } ]
```

説明

未使用のロックファイルを `barman_lock_directory` から自動的に削除します。

パラメーター

-h / --help

ヘルプメッセージを表示して終了します。コマンドの使用法に関する情報を提供します。

14.4.18 barman put-wal

概要

```
put-wal
[ { -h | --help } ]
[ { -t | --test } ]
SERVER_NAME
```

説明

リモートサーバーから WAL ファイルを受信し、サーバーの受信ディレクトリに安全に保存します。WAL ファイルは STDIN を介して提供され、検証用の SHA256SUMS または MD5SUMS ファイルとともに tar ストリームにカプセル化されています。sha256 はデフォルトのハッシュアルゴリズムですが、ユーザーは `barman-wal-archive` を介して `archive-command` を設定するときに `md5` を選択できます。このコマンドは、リモートの `barman-wal-archive` ユーティリティ `barman-cli` パッケージに含まれているから SSH を介して実行することを目的としています。ファイルのコンテンツを完全に管理する場合を除き、このコマンドを直接使用しないでください。

パラメーター

SERVER_NAME

barman ノードのサーバーの名前

-h / --help

ヘルプメッセージを表示して終了します。コマンドの使用法に関する情報を提供します。

-t / --test

WAL 取得のために Barman で指定された Postgres サーバーの接続と構成の両方をテストします。

14.4.19 barman rebuild-xlogdb

概要

```
rebuild-xlogdb
[ { -h | --help } ]
SERVER_NAME [ SERVER_NAME ... ]
```

説明

ディスクの内容に基づいて、サーバーまたは `all` ショートカットを使用するすべてのサーバーの WAL ファイルメタデータを再構築します。WAL アーカイブメタデータは `xlog.db` ファイルに保存され、各 Barman サーバーが独自のコピーを維持します。

パラメーター

SERVER_NAME

barman ノードのサーバーの名前。

-h / --help

ヘルプメッセージを表示して終了します。コマンドの使用法に関する情報を提供します。

14.4.20 barman recover

注釈

このコマンドは非推奨です。代わりに `commands-barman-restore` コマンドを使用します。

14.4.21 barman receive-wal

概要

```
receive-wal
[ --create-slot ]
[ --if-not-exists ]
[ --drop-slot ]
```

(次のページに続く)

```
[ { -h | --help } ]  
[ --reset ]  
[ --stop ]  
SERVER_NAME
```

説明

サーバーのトランザクションログのストリーミングを開始します。このプロセスは、`pg_receivewal` または `pg_receivexlog` を使用して、ストリーミングプロトコルを介して Postgres サーバーから WAL ファイルを受信します。

パラメーター

SERVER_NAME

barman ノードのサーバーの名前。

--create-slot

slot_name 構成パラメーターで構成された物理レプリケーションスロットを作成します。

--if-not-exists

--create-slot が指定され、指定された名前のスロットが既に存在する場合、エラーアウトしません。

--drop-slot

slot_name 構成パラメーターで構成された物理レプリケーションスロットをドロップします。

-h / --help

ヘルプメッセージを表示して終了します。コマンドの使用法に関する情報を提供します。

--reset

receive-wal のステータスをリセットし、サーバーの現在の WAL ファイルからストリーミングをリスタートします。

--stop

サーバーのプロセスを停止します。

警告

barman receive-wal コマンドの --stop オプションは、将来のリリースでは廃止されます。ユーザーは代わりに --stop コマンドの使用を優先する必要があります。これは、この機能进行处理する新しい方法です。

14.4.22 barman restore

概要

```

restore
[ --aws-region AWS_REGION } ]
[ --azure-resource-group AZURE_RESOURCE_GRP ]
[ --bwlimit KBPS ]
[ --exclusive ]
[ --gcp-zone GCP_ZONE ]
[ { --get-wal | --no-get-wal } ]
[ { -h | --help } ]
[ { -j | --jobs } PARALLEL_WORKERS ]
[ --jobs-start-batch-period SECONDS ]
[ --jobs-start-batch-size NUMBER ]
[ --local-staging-path PATH ]
[ { --network-compression | --no-network-compression } ]
[ --no-retry ]
[ --recovery-conf-filename FILENAME ]
[ --recovery-staging-path PATH ]
[ --staging-path STAGING_PATH ]
[ --staging-location STAGING_LOCATION ]
[ --combine-mode COMBINE_MODE ]
[ --remote-ssh-command STRING ]
[ --retry-sleep SECONDS ]
[ --retry-times NUMBER ]
[ --snapshot-recovery-instance INSTANCE_NAME ]
[ --snapshot-recovery-zone GCP_ZONE ]
[ --standby-mode ]
[ --tablespace NAME:LOCATION [ --tablespace NAME:LOCATION ... ] ]
[ --target-action { pause | shutdown | promote } ]
[ --target-immediate ]
[ --target-lsn LSN ]
[ --target-name RESTORE_POINT_NAME ]
[ --target-time TIMESTAMP ]
[ --target-tli TLI ]
[ --target-xid XID ]
[ --staging-wal-directory ]
SERVER_NAME BACKUP_ID DESTINATION_DIR

```

説明

PostgreSQL サーバーの復元操作を実行します。Barman は、宛先ディレクトリのサーバーからバックアップを復元します。復元は、ローカルに barman ノード自身自身でまたはリモートで SSH 経由でアクセス可能な別のマシンで実行できます。場所は、`--remote-ssh-command` オプションを使用するかどうかによって決まります。このコマンドの詳細については、`--remote-ssh-command` セクションを参照してください。`BACKUP_ID` の代わりにショートカットを使用できます。

パラメーター

SERVER_NAME

barman ノードのサーバーの名前

BACKUP_ID

barman カタログのバックアップの ID。 `auto` を使用して、Barman が復元操作に最適なバックアップを自動的に見つけます。

DESTINATION_DIR

バックアップを復元する宛先ディレクトリ。

--aws-region

スナップショットリカバリー用のインスタンスとディスクが存在する AWS リージョンを指定します。このオプションを使用すると、Barman 構成の `aws_region` 値をオーバーライドできます。

--azure-resource-group

スナップショットリカバリーのインスタンスとディスクを含む Azure リソースグループを指定します。このオプションを使用すると、Barman 構成の `azure_resource_group` 値をオーバーライドできます。

--bwlimit

最大転送速度をキロバイト/秒で指定します。0 の値は、制限がないことを示します。この設定は、`bandwidth_limit` 構成オプションをオーバーライドします。

--delta-restore / --no-delta-restore

デルタリカバリーでの `delta-restore` の使用を有効/無効にします。デフォルトは `recovery_options` 設定に基づいています。

--exclusive

ターゲット時間、XID、または LSN を非包括的に設定します。

--gcp-zone

スナップショットリカバリーのインスタンスとディスクが存在する GCP ゾーンを指定します。このオプションを使用すると、Barman 構成の `gcp_zone` 値をオーバーライドできます。

--get-wal / --no-get-wal

リカバリー中の WAL 取得での `get-wal` の使用を有効/無効にします。デフォルトは `recovery_options` 設定に基づいています。

-h / --help

ヘルプメッセージを表示して終了します。コマンドの使用法に関する情報を提供します。

-j / --jobs

バックアップ中にファイルのコピーに使用するパラレルワーカーの数を指定します。この設定は、構成ファイルで指定されている場合、`parallel_jobs` パラメーターをオーバーライドします。

--jobs-start-batch-period

ジョブの単一のバッチを開始する時間を秒単位で指定します。この値は、構成ファイルで設定されている場合、`parallel_jobs_start_batch_period` パラメーターをオーバーライドします。デフォルトは 1 秒です。

--jobs-start-batch-size

単一のバッチで開始するパラレルワーカーの最大数を指定します。この値は、構成ファイルで定義さ

れている場合、`parallel_jobs_start_batch_size` パラメーターをオーバーライドします。デフォルトは 10 ワーカーです。

--local-staging-path

宛先ディレクトリにコピーされる前にバックアップチェーンが結合される Barman ホスト上のパスを指定します。ステージングパス内で作成されたコンテンツは、復元プロセスが完了すると削除されます。このオプションは、ブロックレベルのインクリメンタルバックアップから復元するために必要であり、それ以外の場合は効果がありません。

バージョン 3.15 で非推奨: `--local-staging-path` は非推奨であり、将来のリリースで削除される予定です。代わりに `--staging-path` および `--staging-location` を使用してください。

--network-compression / --no-network-compression

リモート復元中のネットワーク圧縮を有効/無効にします。デフォルトは `network_compression` 構成設定に基づいています。

--no-retry

エラーが発生した場合の再試行はありません。 `--retry-times 0` の設定と同じです。

--recovery-conf-filename

Postgres バージョン 12 以降のバックアップを復旧するときに、Barman が復旧オプションを書き込むファイルの名前を指定します。デフォルトでは、これは `postgresql.auto.conf` に設定されます。ただし、`--recovery-conf-filename` が指定されている場合、リカバリオプションは代わりに指定された値に書き込まれます。デフォルト値はほとんどの Postgres インストールに適していますが、このオプションにより、Postgres が構成メカニズムを変更するツールで管理されている場合たとえば、`postgresql.auto.conf` が `/dev/null` にシンボリックリンクされている場合、代替の場所を指定できます。

--recovery-staging-path

圧縮バックアップのファイルが宛先ディレクトリに解凍される前にステージングされる復旧ホスト上のパスを指定します。バックアップは、命名規則 `barman-staging-SERVER_NAME-BACKUP_ID` に従って、ステージングパス内の独自のディレクトリにステージングされます。このステージングディレクトリは、復元プロセスが完了すると削除されます。このオプションは圧縮バックアップから復元するために必須であり、それ以外の場合は効果がありません。

バージョン 3.15 で非推奨: `--recovery-staging-path` は非推奨であり、将来のリリースで削除される予定です。代わりに `--staging-path` および `--staging-location` を使用してください。

--staging-path

リストア中に中間ファイルがステージングされるパス。圧縮されたバックアップを復元する場合、最終的な宛先にコピーする前に、解凍の一時的な場所として機能します。インクリメンタルバックアップを復元する場合、最終的な宛先にコピーする前にバックアップが結合されます。この場所には、解凍/結合されたバックアップを保存するための十分な領域が必要です。

--staging-location

`--staging-path` がローカルパスかリモートパスであるかを指定します。有効な値は `local` および `remote` です。

--combine-mode

リストア中にインクリメンタルバックアップを組み合わせた場合、`pg_combinebackup` のコピーモードを指定します。

オプションには次のものが含まれます

- **copy** デフォルト インクリメンタル バックアップを組み合わせる場合、標準のファイルコピーを使用します。
- **link** インクリメンタル バックアップを組み合わせる場合、ハードリンクを使用します。合成バックアップの再構成はより速くなりファイルコピーなし、使用するディスク領域は少なくなります。
- **clone** 新しいデータディレクトリにファイルをコピーする代わりに、効率的なファイルクローン作成一部のシステムでは「**reflink**」としても知られますを使用します。これにより、データファイルのほぼ瞬時のコピーが行われます。
- **copy-file-range copy_file_range** システムコールを使用して、効率的なコピーを行います。一部のファイルシステムでは、これにより **clone** と同様の結果が得られ、物理ディスクブロックを共有しますが、他のファイルシステムでは、ブロックをコピーする場合がありますが、最適化されたパスを介して実行します。

各モードの詳細と制限については、[pg_combinebackup documentation](#) を参照してください。

--remote-ssh-command

このオプションは、リモートホストで実行する secure Shell コマンドを指定することにより、リモート復元を有効にします。これは、リモートリストアの構成ファイル 'ssh USER@SERVER' の **ssh_command** サーバーオプションと同様に機能します。

--retry-sleep

コピーが失敗した後、再試行するまでに待機する秒数を指定します。この設定は、バックアップとリストア操作の両方に適用され、構成ファイルで定義されている場合 **basebackup_retry_sleep** パラメーターをオーバーライドします。

--retry-times

エラーが発生した場合に、ベースバックアップコピーを再試行する回数を指定します。これは、バックアップ操作とリストア操作の両方に適用され、構成ファイルに設定されている場合 **basebackup_retry_times** パラメーターをオーバーライドします。

--snapshot-recovery-instance

スナップショットから復元されたディスクが接続されているインスタンスの名前を指定します。このオプションは、**backup_method=snapshot** で作成されたバックアップをリカバリする場合に必要です。

--snapshot-recovery-zone 非推奨

スナップショットリカバリーのインスタンスとディスクを含むゾーン非推奨: **--gcp-zone** に置き換えられます。

--standby-mode

Postgres サーバーをスタンバイとして起動するかどうか。

--tablespace

テーブルスペースの再配置ルールを指定します。**NAME** はテーブルスペース名であり、**LOCATION** はテーブルスペースを復元するための復旧ホスト宛先パスです。

--target-action

回復目標に到達すると、指定されたアクションをトリガーします。このオプションでは、これらのア

クシヨンのいずれかとともにターゲットを定義する必要があります。可能な値は次のとおりです。

- **pause** 復旧目標に到達すると、サーバーは一時停止状態で起動され、ユーザーはインスタンスを検査できます
- **promote** 復旧ターゲットに到達すると、サーバーは復旧オペレーションを終了し、マスターとして昇格します。
- **shutdown** 復旧目標に到達すると、サーバーはシャットダウンされます。

--target-immediate

一貫した状態に到達すると、リカバリは完了しますベースバックアップの終了。

--target-lsn

指定された LSN ログシーケンス番号にリカバリします。Postgres 10 以上が必要です。

--target-name

`pg_create_restore_point(name)` で以前に作成した復元ポイントの指定された名前にリカバリします。

--target-time

指定した時間にリカバリします。YYYY-MM-DD HH:MM:SS.mmm 形式を使用します。

--target-tli

指定されたタイムラインを復元します。数値のタイムライン ID に加えて、特別な値 **current** および **latest** を使用できます。Postgres バージョン 12 以降の場合、デフォルトでは、WAL アーカイブの最新のタイムラインにリカバリします。12 より前の Postgres バージョンの場合、デフォルトでは、バックアップが取得された時点のタイムラインにリカバリします。

--target-xid

指定されたトランザクション ID にリカバリします。

--staging-wal-directory

PITR を実行する際の WAL ファイルの宛先ホスト上のステージングディレクトリ。指定しない場合、宛先ディレクトリ内の `barman_wal` ディレクトリを使用します。

14.4.23 barman replication-status

概要

```
replication-status
[ { -h | --help } ]
[ --minimal ]
[ --source { backup-host | wal-host } ]
[ --target { hot-standby | wal-streamer | all } ]
SERVER_NAME [ SERVER_NAME ... ]
```

説明

指定したサーバーに接続しているストリーミングクライアントのリアルタイム情報とステータスを表示します。構成されているすべてのサーバーの情報を表示するには、`all` ショートカットを指定します。

パラメーター

SERVER_NAME

barman ノードのサーバーの名前

-h / --help

ヘルプメッセージを表示して終了します。コマンドの使用法に関する情報を提供します。

--minimal

機械可読出力。

--source

可能な値は次のとおりです。

- `backup-host` デフォルト サーバーのバックアップ接続情報を使用してクライアントをリストします。
- `wal-host` サーバーの WAL ストリーミング接続情報を使用してクライアントをリストします。

--target

可能な値は次のとおりです。

- `hot-standby` ホットスタンバイサーバーのみをリストします。
- `wal-streamer pg_receivewal` などの WAL ストリーミングクライアントのみをリストします。
- `all` デフォルト すべてのストリーミングクライアントをリストします。

14.4.24 barman show-backup

概要

```
show-backup
[ { -h | --help } ]
SERVER_NAME BACKUP_ID
```

説明

特定のバックアップに関する詳細情報を表示します。このコマンドの詳細については、[Catalog usage](#) を参照してください。BACKUP_ID の代わりにショートカットを使用できます。

パラメーター

SERVER_NAME

barman ノードのサーバーの名前

BACKUP_ID

barman カタログのバックアップの ID。

-h / --help

ヘルプメッセージを表示して終了します。コマンドの使用法に関する情報を提供します。

14.4.25 barman show-servers**概要**

```
show-servers
[ { -h | --help } ]
SERVER_NAME [ SERVER_NAME ... ]
```

説明

conninfo、backup_directory、wals_directory、archive_command などを含むサーバーに関する詳細情報を表示します。構成されているすべてのサーバーに関する情報を表示するには、サーバー名の代わりに all ショートカットを指定します。

パラメーター**SERVER_NAME**

barman ノードのサーバーの名前

-h / --help

ヘルプメッセージを表示して終了します。コマンドの使用法に関する情報を提供します。

14.4.26 barman status**概要**

```
status
[ { -h | --help } ]
SERVER_NAME [ SERVER_NAME ... ]
```

説明

状態、Postgres バージョン、WAL 情報、使用可能なバックアップなどの詳細を含むサーバーのステータスに関する情報を表示します。

パラメーター**SERVER_NAME**

barman ノードのサーバーの名前

-h / --help

ヘルプメッセージを表示して終了します。コマンドの使用法に関する情報を提供します。

14.4.27 barman switch-wal

概要

```
switch-wal
  [ --archive ]
  [ --archive-timeout ]
  [ --force ]
  [ { -h | --help } ]
  SERVER_NAME [ SERVER_NAME ... ]
```

説明

ターゲットサーバー Postgres バージョン 10 以降で `pg_switch_wal()` を実行するか、`pg_switch_xlog()` Postgres バージョン 8.3~9.6 の場合を実行します。

パラメーター

SERVER_NAME

barman ノードのサーバーの名前

--archive

1 つの WAL ファイルがアーカイブされるのを待機します。指定された時間以内に WAL ファイルがアーカイブされない場合デフォルト 30 秒、Barman は失敗終了コードで終了します。このオプションは、スタンバイサーバーでも使用できます。

--archive-timeout

タイムアウトになる前に、新しい WAL ファイルがアーカイブされるのをアーカイバーが待機する時間を秒単位で指定しますデフォルト 30 秒。このオプションは、スタンバイサーバーでも使用できます。

--force

`pg_switch_wal()` の前に CHECKPOINT を実行することにより、スイッチを強制します。

注釈

CHECKPOINT を実行すると、Postgres サーバーの I/O 負荷が増加する可能性があるため、このオプションは注意して使用します。

-h / --help

ヘルプメッセージを表示して終了します。コマンドの使用法に関する情報を提供します。

14.4.28 barman switch-xlog

説明

`switch-wal` コマンドのエイリアス。

14.4.29 barman sync-backup

概要

```
sync-backup
[ { -h | --help } ]
SERVER_NAME BACKUP_ID
```

説明

このコマンドは、サーバーノードに存在するバックアップからすべてのファイルをコピーすることにより、パッシブノードをプライマリと同期します。これはパッシブノードでのみ利用でき、`primary_ssh_command` オプションを使用してプライマリノードとの安全な接続を確立します。BACKUP_ID の代わりにショートカットを使用できます。

パラメーター

SERVER_NAME

barman ノードのサーバーの名前

BACKUP_ID

barman カタログのバックアップの ID。

-h / --help

ヘルプメッセージを表示して終了します。コマンドの使用法に関する情報を提供します。

14.4.30 barman sync-info

概要

```
sync-info
[ { -h | --help } ]
[ --primary ]
SERVER_NAME [ LAST_WAL [ LAST_POS ] ]
```

説明

同期のために、Barman サーバーの現在のステータスに関する情報を収集します。

このコマンドは、正常に完了したすべてのバックアップ、アーカイブされたすべての WAL ファイル、構成、`xlog.db` から読み取られた最後の WAL ファイル、およびファイル内のその位置を含むサーバーの JSON 出力を結果ます。

パラメーター

SERVER_NAME

barman ノードのサーバーの名前

LAST_WAL

指定されたファイルより前の WAL ファイルをスキップするように sync-info に指示しますインクリメンタル同期の場合。

LAST_POS

xlog.db ファイルにすばやく配置するためのヒントインクリメンタル同期用。

-h / --help

ヘルプメッセージを表示して終了します。コマンドの使用法に関する情報を提供します。

--primary

プライマリノードで sync-info を実行します設定されている場合。

14.4.31 barman sync-wals

概要

```
sync-wals
  [ { -h | --help } ]
  SERVER_NAME
```

説明

このコマンドは、サーバーノードからアーカイブされたすべての WAL ファイルをコピーすることにより、パッシブノードをプライマリと同期します。これはパッシブノードでのみ利用でき、`primary_ssh_command` オプションを使用してプライマリノードとの安全な接続を確立します。

パラメーター**SERVER_NAME**

barman ノードのサーバーの名前

-h / --help

ヘルプメッセージを表示して終了します。コマンドの使用法に関する情報を提供します。

14.4.32 barman terminate-process

概要

```
terminate-process SERVER_NAME TASK_NAME
```

説明

`barman terminate-process` コマンドは、指定されたサーバーでアクティブな Barman サブプロセスを終了します。ターゲットプロセスは、タスク名たとえば `backup` または `receive-wal` などで識別されます。サーバーレベルで実行中のプロセスのみを終了できるため、`cron` や `config-update` のようなグローバルプロセスはこのコマンドでは終了できないことに注意してください。

`barman list-processes` の出力を使用して、特定のサーバーのアクティブなプロセスをすべて表示し、終了できるタスクを判断することもできます。このコマンドの詳細については、`barman list-processes` を参照してください。

パラメーター

SERVER_NAME

サブプロセスが実行されているサーバーの名前。

TASK_NAME

終了するサブプロセスを識別するタスク名。

14.4.33 `barman verify-backup`

概要

```
verify-backup
[ { -h | --help } ]
SERVER_NAME BACKUP_ID
```

説明

バックアップマニフェストファイルに対して `pg_verifybackup` を実行します Postgres バージョン 13 以降で利用可能。rsync バックアップの場合、`generate-manifest` コマンドを使用してマニフェストファイルを作成した後に使用できます。バックアップサーバーに `pg_verifybackup` をインストールする必要があります。BACKUP_ID の代わりにショートカットを使用できます。

パラメーター

SERVER_NAME

barman ノードのサーバーの名前

BACKUP_ID

barman カタログのバックアップの ID。

-h / --help

ヘルプメッセージを表示して終了します。コマンドの使用法に関する情報を提供します。

14.4.34 `barman verify`

説明

`verify-backup` コマンドのエイリアス。

14.5 barman-cli コマンド

barman-cli パッケージには、Postgres サーバーとともにインストールする必要がある推奨クライアントユーティリティのコレクションが含まれています。次に、両方のユーティリティのコマンドリファレンスを示します。

14.5.1 barman-wal-archive

概要

```
barman-wal-archive
[ { -h | --help } ]
[ { -V | --version } ]
[ { -U | --user } USER ]
[ --port PORT ]
[ { { -z | --gzip } | { -j | --bzip2 } | --xz | --snappy | --zstd | --lz4 } ]
[ --compression-level COMPRESSION_LEVEL ]
[ { -c | --config } CONFIG ]
[ { -t | --test } ]
[ --md5 ]
BARMAN_HOST SERVER_NAME WAL_PATH
```

説明

このスクリプトは、Postgres サーバーの `archive_command` で利用でき、`put-wal` コマンド Barman 2.6 で紹介を使用して WAL ファイルを Barman ホストに転送できます。Barman ホストへの SSH 接続を確立し、Postgres クラスター内での Barman のシームレスな統合を可能にして、ビジネスの継続性を向上させます。

Exit Statuses は次のとおりです。

- SUCCESS の場合は `0`。
- FAILURE の場合は `non-zero`。

パラメーター

SERVER_NAME

WAL ファイルが取得される Postgres サーバーの Barman で構成されたサーバー名。

BARMAN_HOST

Barman サーバーのホスト。

WAL_PATH

'%p' キーワードの値 `archive_command` によると。

-h / --help

ヘルプメッセージを表示して終了します。

-V / --version

プログラムのバージョン番号を表示して終了します。

-U / --user

Barman サーバーへの SSH 接続のユーザーを指定しますデフォルトは `barman`。

--port

Barman サーバーへの SSH 接続に使用されるポートを定義します。

-z / --gzip

Barman サーバーに送信する前に、WAL ファイルを `gzip` 圧縮します。

-j / --bzip2

Barman サーバーに送信する前に、WAL ファイルを `bzip2` 圧縮します。

--xz

Barman サーバーに送信する前に、WAL ファイルを `xz` 圧縮します。

--snappy

Barman サーバーに送信する前に、WAL ファイルを `snappy` 圧縮します `python-snappy` Python ライブラリをインストールする必要があります。

--zstd

Barman サーバーに送信する前に、WAL ファイルを `zstd` 圧縮します `zstandard` Python ライブラリをインストールする必要があります。

--lz4

Barman サーバーに送信する前に、WAL ファイルを `lz4` 圧縮します `lz4` Python ライブラリをインストールする必要があります。

--compression-level

選択した圧縮アルゴリズムで使用される圧縮レベル。有効な値は、選択したアルゴリズムまたは事前定義されたラベルのいずれか `low`、`medium`、および `high` のサポートされている範囲内の整数です。各アルゴリズムの範囲と、各事前定義されたラベルがマップするレベルは `low` で見つかります。

-c / --config

Barman サーバー上の構成ファイルを指定します。

-t / --test

Barman で指定された Postgres サーバーの接続と構成をテストして、WAL ファイルを受信する準備ができていることを確認します。このオプションは、必須引数 `WAL_PATH` を無視します。

--md5

Barman サーバーに送信するときに WAL ファイルのチェックサムを計算するハッシュアルゴリズムとして `SHA256` の代わりに `MD5` を使用します。Barman サーバーの古いバージョンは `MD5` のみをサポートしていたため、これは古いサーバーバージョンとの互換性を維持するために使用されます。

i 注釈

`barman-wal-archive` で圧縮が有効になっている場合、Barman サーバーで構成された圧縮設定が異なる場合、それが優先されます。

重要

`barman-wal-archive` で圧縮が有効になっている場合、ファイルが Barman に送信される前に、クライアント側で実行されます。圧縮率を高くすると WAL の SHIPPING が遅延し、WAL ファイルが蓄積される可能性があるため、データベースサーバーの負荷と選択した圧縮アルゴリズムとレベルに注意してください。

14.5.2 barman-wal-restore

概要

```
barman-wal-restore
[ { -h | --help } ]
[ { -V | --version } ]
[ { -U | --user } USER ]
[ --port PORT ]
[ { -s | --sleep } SECONDS ]
[ { -p | --parallel } JOBS ]
[ --spool-dir SPOOL_DIR ]
[ { -P | --partial } ]
[ { { -z | --gzip } | { -j | --bzip2 } | --keep-compression } ]
[ { -c | --config } CONFIG ]
[ { -t | --test } ]
BARMAN_HOST SERVER_NAME WAL_NAME WAL_DEST
```

説明

このスクリプトは、Postgres サーバーの `restore_command` として機能し、Barman の `get-wal` 機能を介して WAL ファイルの取得を有効にします。Barman ホストへの SSH 接続を確立し、Postgres クラスタ内での Barman の統合を促進し、ビジネス継続性を強化します。

Exit Statuses は次のとおりです。

- SUCCESS の場合は 0。
- リモート `get-wal` コマンド FAILURE の場合 1、おそらく要求された WAL が見つからないためです。
- 2 ssh 接続の場合 FAILURE。
- FAILURE のその他の non-zero。

パラメーター

SERVER_NAME

WAL ファイルが取得される Postgres サーバーの Barman で構成されたサーバー名。

BARMAN_HOST

Barman サーバーのホスト。

WAL_NAME

'%f' キーワードの値 `restore_command` によると。

WAL_DEST

'%p' キーワードの値 `restore_command` によると。

-h / --help

ヘルプメッセージを表示して終了します。

-V / --version

プログラムのバージョン番号を表示して終了します。

-U / --user

Barman サーバーへの SSH 接続のユーザーを指定しますデフォルトは `barman`。

--port

Barman サーバーへの SSH 接続に使用されるポートを定義します。

-s / --sleep

`get-wal` 要求が失敗した後、`SECONDS` のために一時停止しますデフォルトは `0` - 待機なし。

-p / --parallel

ファイルの数を `peek` に指定し、同時に転送しますデフォルトでは `0` - 無効になっています。

--spool-dir

WAL ファイルのスパールディレクトリを指定しますデフォルトで `/var/tmp/walrestore`。

-P / --partial

取得には部分的な WAL ファイル `.partial` を含めます。

-z / --gzip

`gzip` で圧縮した WAL ファイルを転送します。

-j / --bzip2

`bzip2` で圧縮した WAL ファイルを転送します。

--keep-compression

指定すると、圧縮ファイルはそのまま転送され、クライアントサイドで到着時に解凍されます。

-c / --config

Barman サーバー上の構成ファイルを指定します。

-t / --test

Barman で指定された Postgres サーバーの接続と構成をテストして、WAL ファイルを受信する準備ができていることを確認します。このオプションは、必須引数 `WAL_NAME` および `WAL_DEST` を無視します。

 警告

`-z / --gzip` および `-j / --bzip2` オプションは非推奨であり、将来的に削除される予定です。WAL 圧縮の場合、`compression` 構成オプションを介して Barman サーバーで直接有効にしてください。

第15章 地理的な冗長性

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

この機能を使用すると、ユーザーは PostgreSQL バックアップの地理的に分散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 サーバーではなく別の Barman インストールに接続されている Barman バックアップサーバーは、パッシブノードとして定義されます。パッシブノードは `primary_ssh_command` オプションを介して構成され、グローバルプライマリ Barman インストールの完全なレプリカの場合とサーバーレベル直接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。

15.1 同期情報

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

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

コマンドは、以下を含む JSON オブジェクトを結果ます。

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

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

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

15.2 構成

サーバーをパッシブノードとしての構成は、簡単な操作です。サーバー構成に次のオプションを追加するだけです。

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

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

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

```
forward_config_path = true
```

15.3 ノードの同期

ノードがパッシブ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。

- 標準メンテナンス操作の対象外です。
- barman backup を含む Postgres への直接操作は禁止されています。

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

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

15.4 手動同期

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

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

sync-backup barman は primary_ssh_command を使用してプライマリサーバーに接続し、リモートマシンにバックアップが存在する場合、rsync を使用してすべてのファイルのコピーを開始します。バックアップごとに一度に 1 つの同期プロセスのみが許可されます。

WAL ファイルは、次の方法で同期することもできます。

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

第16章 フックスクリプト

Barman では、`:term:DBA <DBA>` が特定のイベント中にフックスクリプトを実行できます。

- *Before and after creating a backup*。
- *Before and after deleting a backup*。
- *Before and after a WAL file is archived*。
- *Before and after a WAL file is deleted*。
- *Before and after restoring a backup*。

重要

これらのスクリプトは、個々のサーバーに対してオーバーライドできるグローバルオプションを使用して構成できます。削除およびリカバリーフックスクリプトは **version 2.4** で導入されました。

Barman が管理できるフックスクリプトには 2 種類があります。

- **Standard Hook Scripts** これらのスクリプトは 1 回実行され、リターンコードはチェックされません。
- **Retry Hook Scripts** これらのスクリプトは、戻りコードによっては、複数回実行される場合があります。

再試行フックスクリプトを実行すると、Barman は戻りコードを確認し、次のいずれかを返すまでスクリプトの再試行を続けます。

- **SUCCESS** 標準リターンコード 0
- **ABORT_CONTINUE** リターンコード 62
- **ABORT_STOP** リターンコード 63

他のリターンコードは一時的な障害として扱われ、Barman にスクリプトを再試行するように求めます。これにより、ユーザーはより詳細な制御が可能になり、フックスクリプトで障害を一時的なものと思なすかどうかを決定できます。さらに、プリフックスクリプトの場合、**ABORT_STOP** を返すと、ユーザーは Barman が失敗してメインオペレーションを中断するように要求できます。

フックスクリプトは次の順序で実行されます存在しない場合はスキップされます。

1. **Standard pre hook script**。
2. **Retry pre hook script**。
3. **Main event** バックアップオペレーション、リストアオペレーションまたは WAL アーカイブ、再試行プリフックスクリプトが **ABORT_STOP** で中止された場合を除きます。

4. **Retry post hook script**。

5. **Standard post hook script**。

フックスクリプトによって生成されたすべての出力は、Barman のログファイルに記録されます。

注釈

現在、`ABORT_STOP` は、再試行ポストフックスクリプトによって無視されます。このような場合、これにより追加の警告がログに記録され、`ABORT_STOP` は `ABORT_CONTINUE` と同じように動作します。

16.1 バックアップを作成する前と後

- `pre_backup_script` 終了コードをチェックせずに、ベースバックアップの前に 1 回実行されるフックスクリプト。
- `pre_backup_retry_script` ベースバックアップの前に実行される再試行フックスクリプト。成功するか中止されるまで繰り返し実行されます。
- `post_backup_retry_script` ベースバックアップの後に実行される再試行フックスクリプト。成功するか中止されるまで繰り返し実行されます。
- `post_backup_script` ベースバックアップの後、終了コードをチェックせずに 1 回実行されるフックスクリプト。

シェル環境には、**backup scripts** の次の変数が含まれます。

- `BARMAN_BACKUP_DIR` バックアップの宛先ディレクトリ。
- `BARMAN_BACKUP_ID` バックアップの ID。
- `BARMAN_BACKUP_INFO_PATH` ``backup.info`` ファイルのパス。
- `BARMAN_CONFIGURATION` Barman が使用する構成ファイル。
- `BARMAN_ERROR` 任意のエラーメッセージ `post` フェーズにのみ適用されます。
- `BARMAN_PHASE` スクリプトのフェーズを示します `pre` または `post`。
- `BARMAN_PREVIOUS_ID` 以前のバックアップの ID、使用可能な場合。
- `BARMAN_RETRY` 再試行スクリプトの場合は 1 に設定します。それ以外の場合は、0 です。
- `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。

16.2 バックアップを削除する前と後

- `pre_delete_script` バックアップの削除の前に 1 回トリガーされ、終了コードチェックなしのフックスクリプト。
- `pre_delete_retry_script` バックアップの削除前に実行される再試行フックスクリプト。成功または中止されるまで繰り返し実行されます。
- `post_delete_retry_script` バックアップの削除後に実行される再試行フックスクリプト。成功または中止されるまで繰り返し実行されます。
- `post_delete_script` バックアップの削除後に 1 回トリガーされ、終了コードチェックなしのフックスクリプト。

Delete scripts は、**Delete scripts** と同じ環境変数に加えて、

`BARMAN_NEXT_ID` 使用可能な場合、次のバックアップの ID。

16.3 WAL がアーカイブされる前と後

- `pre_archive_script` 終了コードをチェックせずに、メンテナンスによって WAL ファイルがアーカイブされる前に、通常は Barman cron を介して 1 回実行されるフックスクリプト。
- `pre_archive_retry_script` WAL ファイルがメンテナンスによってアーカイブされる前に実行される再試行フックスクリプト。成功または中止されるまで繰り返し実行されます。
- `post_archive_retry_script` WAL ファイルがメンテナンスによってアーカイブされた後に実行される再試行フックスクリプト。成功または中止されるまで繰り返し実行されます。
- `post_archive_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、終了コードをチェックせずに 1 回実行されるフックスクリプト。

WAL archive scripts は、**WAL archive scripts** といくつかの環境変数を共有します。

- `BARMAN_CONFIGURATION` Barman が使用する構成ファイル。
- `BARMAN_ERROR` 生成されたエラーメッセージ事後フェーズのみ。
- `BARMAN_PHASE` スクリプトのフェーズ前または後。
- `BARMAN_SERVER` サーバーの名前。

さらに、次の変数は **WAL archive scripts** 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 ファイルに適用される圧縮のタイプ。

16.4 WAL ファイルが削除される前と後

- `pre_wal_delete_script` WAL ファイルが削除される前に実行されるフックスクリプト。
- `pre_wal_delete_retry_script` WAL ファイルを削除する前に実行される再試行フックスクリプト。成功または中止されるまで繰り返し実行されます。
- `post_wal_delete_retry_script` WAL ファイルが削除された後に実行される再試行フックスクリプト。成功または中止されるまで繰り返し実行されます。
- `post_wal_delete_script` WAL ファイルが削除された後に実行されるフックスクリプト。

WAL delete scripts は、**WAL delete scripts** と同じ環境変数を利用します。

16.5 バックアップを復元する前と後

- `pre_recovery_script` 終了コードをチェックせずに、バックアップ復元の前に 1 回実行されるフックスクリプト。
- `pre_recovery_retry_script` バックアップ復元の前に実行される再試行フックスクリプト。成功または中止されるまで繰り返し実行されます。
- `post_recovery_retry_script` バックアップ復元の後に実行される再試行フックスクリプト。成功するか中止されるまで繰り返し実行されます。
- `post_recovery_script` バックアップ復元後に終了コードをチェックせずに 1 回実行されるフックスクリプト。

Recovery scripts は、**Recovery scripts** と同じ環境変数に加えて、

- `BARMAN_DESTINATION_DIRECTORY` 新しいインスタンスが復元されるディレクトリ。
- `BARMAN_TABLESPACES` 表領域の再配置マップ JSON 形式、該当する場合。
- `BARMAN_REMOTE_COMMAND` 復元中に使用される secure Shell コマンド該当する場合。
- `BARMAN_RECOVER_OPTIONS` 追加のリカバリオプション JSON 形式、該当する場合。

16.6 barman でフックとして barman-cloud-* スクリプトを使用する

installation section のプロセスに従って、*installation section* の使用を開始します。

注釈

構成オプションの詳細については、*configuration* セクションを参照してください。

`barman-cloud-backup` は、次の Barman バックアップタイプの `post-backup script` として使用できます。

- `backup_method = rsync` で作成されたバックアップ。
- `backup_compression` が適用されない場合に `backup_method = postgres` で作成されたバックアップ。

これを構成するには、Barman のサーバー構成に次の行を追加します。

```
post_backup_retry_script = barman-cloud-backup [ OPTIONS ] DESTINATION_URL SERVER_NAME
```

警告

`barman-cloud-backup` は、フックスクリプトとして使用する場合、バックアップステータスが `DONE` である必要があります。バックアップに他のステータスがある場合、バックアップは失敗します。問題を回避するには、`-w / --wait` オプションを使用してバックアップを実行し、バックアップステータスが `WAITING_FOR_WALS` 中にフックスクリプトが実行されないようにすることをお勧めします。

さらに、Postgres サーバーの Barman 構成に次の行を追加することにより、WAL 以前のアーカイブスクリプトとして `barman-cloud-wal-archive` をセットアップします。

```
pre_archive_retry_script = barman-cloud-wal-archive [ OPTIONS ] DESTINATION_URL ↵  
↵ SERVER_NAME
```


第17章 クラウドのBarman

Barman は、Postgres サーバーをクラウドにバックアップする 2 つの主な方法を提供しています。

- **Creating disk volume snapshots as base backups.**

これは、2 つの異なるアプローチで実現できます。

1. バックアップはクラウドにディスクボリュームのスナップショットとして作成されますが、バックアップメタデータと WAL ファイルを保存するように Barman サーバーをセットアップします。これは Barman の統合機能です。このアプローチを選択する場合、詳細については [cloud snapshots backups](#) セクションを参照してください。
2. Barman サーバーを必要とせず、barman クラウドクライアントパッケージが提供するコマンドラインユーティリティと直接対話してバックアップを管理します。バックアップメタデータと WAL ファイルはクラウドオブジェクトストレージに保存されますが、ベースバックアップはクラウドにディスクボリュームのスナップショットとして作成されます。

- **Creating and transferring base backups to a cloud object storage.**

これは、2 つの異なるアプローチを通じて達成することもできます。

1. Barman サーバーを使用せず、Postgres ホストの barman クラウドクライアントパッケージが提供するユーティリティを使用します。ベースバックアップと WAL は両方ともローカルホスト Postgres ホストから読み取られ、バックアップメタデータとともにクラウドオブジェクトストレージに保存されます。
2. ベースバックアップを取得し、バックアップメタデータと WAL ファイルを保存するように Barman サーバーを設定し、barman クラウドクライアントパッケージがフックスクリプトとして提供するユーティリティを使用して、クラウドオブジェクトストレージにコピーします。このアプローチを選択する場合、詳細については [barman でフックとして barman-cloud-* スクリプトを使用する](#) セクションを参照してください。

ドキュメントのこのセクションでは、専用の barman サーバーを必要とせずにバックアップを管理および対話するために使用できる `barman-cloud-*` コマンドに焦点を当てています。それで作業を開始するには、Postgres サーバーと同じマシンに barman クラウドクライアントパッケージをインストールする必要があります。

これらのオプションを理解することは、クラウドのバックアップとリカバリーのニーズに適切なアプローチを選択し、Barman の可能性を最大限に活用するのに役立ちます。

17.1 Barman クラウドクライアントパッケージ

barman クラウドクライアントパッケージは、Barman サーバーを必要とせずに、オブジェクトストレージとディスクボリュームのスナップショットの両方でクラウドバックアップを管理するためのコマンドを提供します。

このユーティリティを使用すると、次のことができます。

- スナップショットバックアップを直接作成および管理します。
- バックアップを作成してクラウドオブジェクトストレージに転送します。

クラウドストレージとディスクボリュームのバックアップを個別に処理するための追加機能を提供しますが、Barman のネイティブ機能を完全に拡張するものではありません。Barman の統合機能と比較して制限があり、一部の操作が異なる場合があります。

注釈

Barman は、AWS S3 および S3 互換オブジェクトストア、Azure Blob Storage、および Google Cloud Storage をサポートしています。

注釈

一部の S3 サードパーティプロバイダーは、`boto3>=1.36` でデフォルトで提供される新しいデータ完全性保護チェックと互換性がありません。プロバイダーがこのグループに該当する場合、AWS S3 との非互換性を克服するのに役立つ回避策があります。

この場合、`barman-cloud` ツールが動作するためには、`when_required` を以下に示す 2 つの環境変数に設定する必要があります。動作する保証はありませんが、これらを設定すると、そのような環境で `barman-cloud` が動作することが確認されています。

```
AWS_REQUEST_CHECKSUM_CALCULATION=when_required
AWS_RESPONSE_CHECKSUM_VALIDATION=when_required
```

重要

AWS boto3 1.36 から、**Data Integrity Protection checks** の動作が変更されました。Barman が使用する一部のメソッドは、`Content-MD5` ヘッダーを必要としません。

これは、新しい boto3 バージョンで使用した場合の **S3-compatible storage providers that have not updated their server-side code may fail** を意味します。たとえば、MinIO は、boto3 1.36 発表の直後にこの変更に対応しました。

MinIO を使用している場合、最新リリースまたは少なくとも `RELEASE.2025-02-03T21-03-04Z` 以降に **must upgrade** して、互換性を保証し、バックアップを削除するときや `keep` アノテーションをリリースする際の障害を回避します。

17.2 インストール

Postgres サーバーをクラウドプロバイダーに直接バックアップするには、これらのサーバーに `barman` クラウドクライアントパッケージをインストールする必要があります。インストールプロセスは、使用しているディストリビューションによって異なることに注意してください。

インストールプロセスについては [installation](#) セクションを参照し、各ディストリビューションの重要な情報を必ずメモしてください。

17.3 オブジェクトロックのサポート

Barman クラウドは、クラウドオブジェクトストレージに保存されているバックアップの Write-Once-Read-Many(WORM) 保護を提供するオブジェクトロックメカニズムと連携できます。オブジェクトロックは、データ保持要件へのコンプライアンスを保証し、バックアップファイルの誤ってまたは悪意のある削除を防止し、重要なバックアップデータをランサムウェア攻撃または不正な削除から保護します。

オブジェクトロックは、一定の期間保持期間または無期限にリーガルホールドでオブジェクトが削除または上書きされるのを防ぎます。これは、規制コンプライアンス要件の場合に特に役立ちます。

注釈

オブジェクトロック検証は、現在 AWS S3 および S3 互換ストレージプロバイダーでのみサポートされています。

重要

Barman は、オブジェクトロックを作成または管理しません。適切な保持ポリシーを使用してクラウドストレージバケットを構成するのはユーザーの責任です。Barman は、削除操作中にこれらのロックを検証および尊重して、構成されたポリシーへのコンプライアンスを保証できます。

17.3.1 S3 オブジェクトロック

Barman は、AWS S3 オブジェクトロックと連携して、バックアップの不変性を維持できます。この機能を使用するには、必要なオブジェクトロック設定で S3 バケットを構成する必要があります。

- **Object versioning enabled** ファンクションへのオブジェクトロックに必要です。
- **Default retention policy** バケット内の新しいオブジェクトのデフォルトの保持ポリシーを構成します。このポリシーは、Barman によってではなく、AWS S3 内で完全に管理されます。

これらの設定が S3 バケットで構成されている場合、AWS は `barman-cloud-backup` および `barman-cloud-wal-archive` を介してアップロードされたオブジェクトに保持ポリシーを自動的に適用します。オブジェクトは、バケットの保持構成に従って保護され、保持期間が終了するまで削除または変更することはできません。

`barman-cloud-backup-delete` でバックアップと WAL を削除する場合、`--check-object-lock` オプションを使用して、削除を試行する前にオブジェクトがオブジェクトロックで保護されているかどうかを確認できます。このフラグは、バックアップファイルのロックステータスを確認し、構成された保持ポリシー

を尊重するように Barman に指示し、削除の失敗を防ぎ、データ保持要件へのコンプライアンスを保証します。

AWS S3 オブジェクトロックの詳細については、[AWS S3 Object Lock documentation](#) を参照してください。

削除プロセスやパフォーマンスの考慮事項に関する情報を含む、`--check-object-lock` オプションの動作の詳細については、`barman-cloud-barman-cloud-backup-delete` コマンドリファレンスを参照してください。

17.4 コマンドリファレンス

このユーティリティを使用して、クラウドでバックアップとリカバリーを管理するために使用できるコマンドがいくつかあります。これらの終了ステータスは、`SUCCESS` (0)、`FAILURE` (1)、`FAILED CONNECTION` (2)、および `INPUT_ERROR` (3) です。その他のゼロ以外の値は `FAILURE` です。

注釈

Barman クラウドコマンドを実行するときに、バックアップ ID の代わりにバックアップ *shortcuts* を指定できます。クラウドコマンドは、次のショートカットをサポートしています。 `first/oldest`、`last/latest` および `last-failed`。

17.4.1 barman-cloud-backup

Synopsis

```
barman-cloud-backup
    [ { -V | --version } ]
    [ --help ]
    [ { { -v | --verbose } | { -q | --quiet } } ]
    [ { -t | --test } ]
    [ --cloud-provider { aws-s3 | azure-blob-storage | google-cloud-
→storage } ]
    [ { { -z | --gzip } | { -j | --bzip2 } | --snappy } ]
    [ { -h | --host } HOST ]
    [ { -p | --port } PORT ]
    [ { -U | --user } USER ]
    [ { -d | --dbname } DBNAME ]
    [ { -n | --name } BACKUP_NAME ]
    [ { -J | --jobs } JOBS ]
    [ { -S | --max-archive-size } MAX_ARCHIVE_SIZE ]
    [ --immediate-checkpoint ]
    [ --min-chunk-size MIN_CHUNK_SIZE ]
    [ --max-bandwidth MAX_BANDWIDTH ]
    [ --snapshot-instance SNAPSHOT_INSTANCE ]
    [ --snapshot-disk NAME [ --snapshot-disk NAME ... ] ]
    [ --snapshot-zone GCP_ZONE ]
    [ -snapshot-gcp-project GCP_PROJECT ]
```

(次のページに続く)

(前のページからの続き)

```

[ --tag KEY,VALUE [ --tag KEY,VALUE ... ] ]
[ --endpoint-url ENDPOINT_URL ]
[ { -P | --aws-profile } AWS_PROFILE ]
[ --profile AWS_PROFILE ]
[ --read-timeout READ_TIMEOUT ]
[ { -e | --encryption } { AES256 | aws:kms } ]
[ --sse-kms-key-id SSE_KMS_KEY_ID ]
[ --aws-region AWS_REGION ]
[ --aws-await-snapshots-timeout AWS_AWAIT_SNAPSHOTS_TIMEOUT ]
[ --aws-snapshot-lock-mode { compliance | governance } ]
[ --aws-snapshot-lock-duration DAYS ]
[ --aws-snapshot-lock-cool-off-period HOURS ]
[ --aws-snapshot-lock-expiration-date DATETIME ]
[ { --azure-credential | --credential } { azure-cli | managed-
→identity | default } ]
[ --encryption-scope ENCRYPTION_SCOPE ]
[ --azure-subscription-id AZURE_SUBSCRIPTION_ID ]
[ --azure-resource-group AZURE_RESOURCE_GROUP ]
[ --gcp-project GCP_PROJECT ]
[ --kms-key-name KMS_KEY_NAME ]
[ --gcp-zone GCP_ZONE ]
DESTINATION_URL SERVER_NAME

```

Description

`barman-cloud-backup` スクリプトは、Postgres サーバーの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転送するために使用されます。また、Barman サーバーからサポートされているクラウドのいずれかに Barman バックアップをコピーするためのフックスクリプト `post_backup_retry_script` として利用することもできます。

このスクリプトには、PGDATA およびテーブルスペースへの読み取りアクセスが必要で、通常は postgres ユーザーとして実行されます。Barman サーバーで使用する場合、Barman バックアップが保存されているディレクトリへの読み取りアクセスが必要です。--snapshot- 引数を使用され、選択したクラウドプロバイダーでスナップショットがサポートされている場合、バックアップは指定されたディスク --snapshot-disk のスナップショットを使用して実行されます。バックアップラベルとメタデータもクラウドにアップロードされます。

❗ 注釈

GCP の場合、`GOOGLE_APPLICATION_CREDENTIALS` env での認証のみがサポートされています。

🔴 重要

設定された --max-archive-size より大きいファイルがデータディレクトリまたはテーブルスペースに存在する場合、クラウドのアップロードは失敗する場合があります。ただし、1GB までの Postgres

ファイルは、`--max-archive-size` 設定に関係なく、常に許可されます。

Parameters

SERVER_NAME

バックアップするサーバーの名前。

DESTINATION_URL

AWS S3 のバケットなどのクラウド宛先の URL。例 `s3://bucket/path/to/folder`。

-V / --version

バージョンを表示して終了します。

--help

このヘルプメッセージを表示して終了します。

-v / --verbose

出力の詳細度を増やしますたとえば、`-vv` は `-v` より大きいです。

-q / --quiet

出力の詳細度を減らしますたとえば、`-qq` は `-q` より小さいです。

-t / --test

クラウドの接続をテストして終了します。

--cloud-provider

ストレージバックエンドとして使用するクラウドプロバイダー。

許可されるオプション

- `aws-s3`。
- `azure-blob-storage`。
- `google-cloud-storage`。

-z / --gzip

クラウドへのアップロード中にバックアップを `gzip` 圧縮します Python < 3.2 では使用しないでください。

-j / --bzip2

クラウドへのアップロード中にバックアップを `bzip2` 圧縮します Python < 3.3 では使用しないでください。

--snappy

クラウドへのアップロード中にバックアップを `snappy`-圧縮しますオプションの `python-snappy` ライブラリが必要です。

-h / --host

Postgres 接続用のホストまたは Unix ソケットデフォルト `libpq` 設定。

-p / --port

Postgres 接続用のポートデフォルト `libpq` 設定。

-U / --user

Postgres 接続のユーザー名デフォルト libpq 設定。

-d / --dbname

Postgres 接続のデータベース名または conninfo 文字列デフォルト "postgres"。

-n / --name

barman-cloud-restore や barman-cloud-backup-delete などのコマンドでこのバックアップを参照するために使用できる名前。

-J / --jobs

クラウドストレージにデータをアップロードするサブプロセスの数デフォルト 2。

-S / --max-archive-size

クラウドストレージにアップロードする際のアーカイブの最大サイズデフォルト 100GB。

--immediate-checkpoint

初期チェックポイントをできるだけ早く実行するように強制します。

--min-chunk-size

クラウドストレージにアップロードする際の個々のチャンクの最小サイズデフォルト aws-s3 の場合は 5MB、azure-blob-storage の場合は 64KB、google-cloud-storage には適用されません。

--max-bandwidth

オブジェクトストレージにバックアップするときに 1 秒ごとにアップロードされるデータの最大量デフォルト 0 -制限なし。

--snapshot-instance

スナップショットとしてバックアップされるディスクが接続されているインスタンス。

--snapshot-disk

スナップショットを取得するディスクの名前。

--tag

クラウドストレージにアップロードされたすべてのファイル、および/またはスナップショットを使用する場合は作成されたスナップショットに追加されるタグ。

--tags

クラウドストレージにアップロードされたすべてのファイル、および/またはスナップショットを使用する場合は作成されたスナップショットに追加されるタグ。

i 注釈

位置引数の前に **--tags** を使用している場合、その後に **--** を挿入してオプション引数の終了を示す必要があります。これは、**--** 以降のすべてを位置引数として扱うようにパーサーに指示します。**--** がないと、Barman は位置引数を最後のオプションの値と誤って解釈する可能性があります。

バージョン 3.15 で非推奨: **--tags** は非推奨です。代わりに **--tag** を使用してください。

Extra options for the AWS cloud provider

--endpoint-url

デフォルトの S3 エンドポイント URL を指定されたものでオーバーライドします。

-P / --aws-profile

プロファイル名 AWS 資格情報ファイルの INI セクションなど。

--profile 非推奨

プロファイル名 AWS 資格情報ファイルの INI セクションなど --aws-profile に置き換えられます。

--read-timeout

接続からの読み取りを待機しているときにタイムアウトが発生するまでの秒単位の時間デフォルト 60 秒。

-e / --encryption

アップロードされたデータを S3 に保存するときに使用される暗号化アルゴリズム。

許可されるオプション

- AES256。
- aws:kms。

--sse-kms-key-id

S3 にアップロードされたデータの暗号化に使用する AWS KMS キー ID。独自のキー ID を使用して、またはキーの完全な ARN を使用して指定できます。-e / --encryption が aws:kms に設定されている場合にのみ許可されます。

--aws-region

EC2 VM と、--snapshot-instance および --snapshot-disk 引数で定義されたストレージボリュームを含む AWS リージョンの名前。

--aws-await-snapshots-timeout

タイムアウトになるまでに AWS でスナップショットが作成されるのを待機する秒単位の時間デフォルト 3600 秒。

--aws-snapshot-lock-mode

スナップショットのロックモード。これは、--snapshot-instance および --snapshot-disk が設定されている場合にのみ有効です。

許可されるオプション

- compliance。
- governance。

--aws-snapshot-lock-duration

ロック期間は、スナップショットがロックされた期間日単位であり、1~36,500 の範囲です。ロック期間または有効期限のいずれかを設定します両方ではありません。

--aws-snapshot-lock-cool-off-period

クーリングオフ期間は、compliance モードでスナップショットをロックするときに指定できるオプションの時間時間単位で、1~72 の範囲で指定します。

--aws-snapshot-lock-expiration-date

ロックの期間は、将来の有効期限によって決まります。YYYY-MM-DDTHH:MM:SS.sssZ 形式を使用し

て、スナップショット作成日時から少なくとも 1 日後である必要があります。ロック期間または有効期限のいずれかを設定します両方ではありません。

Extra options for the Azure cloud provider

--azure-credential / --credential

オプションで、Azure で認証するときに使用する資格情報の種類を指定します。省略すると、Azure Blob Storage の資格情報が環境から取得され、デフォルトの Azure 認証フローが他のすべての Azure サービスでの認証に使用されます。環境で資格情報が見つからない場合、デフォルトの Azure 認証フローは Azure Blob Storage にも使用されます。

許可されるオプション

- `azure-cli`。
- `managed-identity`。
- `default`。

--encryption-scope

Azure のデータの暗号化に使用される Azure Blob Storage サービスで定義された暗号化スコープの名前。

--azure-subscription-id

`--snapshot-instance` および `--snapshot-disk` 引数で定義されたインスタンスとストレージボリュームを所有する Azure サブスクリプションの ID。

--azure-resource-group

`--snapshot-instance` および `--snapshot-disk` 引数で定義されたコンピューティングインスタンスとディスクが属する Azure リソースグループの名前。

Extra options for GCP cloud provider

--gcp-project

ディスクスナップショットを保存する GCP プロジェクト。

--snapshot-gcp-project 非推奨

ディスクスナップショットを保存する GCP プロジェクト `--gcp-project` に置き換えられます。

--kms-key-name

GCS でアップロードされたデータの暗号化に使用する GCP KMS キーの名前。

--gcp-zone

スナップショットを取得するディスクのゾーン。

--snapshot-zone 非推奨

スナップショットを取得するディスクのゾーン- `--gcp-zone` に置き換えられます。

17.4.2 barman-cloud-backup-delete

Synopsis

```

barman-cloud-backup-delete
    [ { -V | --version } ]
    [ --help ]
    [ { { -v | --verbose } | { -q | --quiet } } ]
    [ { -t | --test } ]
    [ --cloud-provider { aws-s3 | azure-blob-storage | google-cloud-
→storage } ]
    [ --endpoint-url ENDPOINT_URL ]
    [ { -r | --retention-policy } RETENTION_POLICY ]
    [ { -m | --minimum-redundancy } MINIMUM_REDUNDANCY ]
    [ { -b | --backup-id } BACKUP_ID ]
    [ --dry-run ]
    [ --check-object-lock ]
    [ { -P | --aws-profile } AWS_PROFILE ]
    [ --profile AWS_PROFILE ]
    [ --read-timeout READ_TIMEOUT ]
    [ { --azure-credential | --credential } { azure-cli | managed-
→identity | default } ]
    [--batch-size DELETE_BATCH_SIZE]
    SOURCE_URL SERVER_NAME

```

Description

`barman-cloud-backup-delete` スクリプトは、`barman-cloud-backup` コマンドで作成した 1 つ以上のバックアップをクラウドストレージから削除し、関連する WAL ファイルを削除するために使用されます。

バックアップは、バックアップ ID `barman-cloud-backup-list` から取得または保持ポリシーによって指定できます。保持ポリシーは Barman サーバーで使用されるものをミラーリングし、指定されたポリシーを満たすために必要でないすべてのバックアップを削除します。バックアップが削除されると、そのバックアップに関連付けられている未使用の WAL ファイルも削除されます。

WAL は、次の場合に未使用とみなされます。

- WAL は、残っている最も古いバックアップの `begin_wal` 値よりも古いです。
- WAL は、クラウドに保存されるアーカイブバックアップには必要ありません。

注釈

GCP の場合、`GOOGLE_APPLICATION_CREDENTIALS` env での認証のみがサポートされています。

重要

各バックアップの削除には、クラウドプロバイダーへの 3 つの個別の要求が含まれます。1 つはバックアップファイル用、もう 1 つは `backup.info` ファイル用、もう 1 つは関連する WAL 用です。大量のバックアップがクラウドストレージに蓄積されている場合、保持ポリシーによって削除すると、大量の削除要求が発生する場合があります。

重要

AWS boto3 1.36 から、**Data Integrity Protection checks** の動作が変更されました。Barman が使用する一部のメソッドは、Content-MD5 ヘッダーを必要としません。

これは、新しい boto3 バージョンで使用した場合の **S3-compatible storage providers that have not updated their server-side code may fail** を意味します。たとえば、MinIO は、boto3 1.36 発表の直後にこの変更に対応しました。

MinIO を使用している場合、最新リリースまたは少なくとも `RELEASE.2025-02-03T21-03-04Z` 以降に **must upgrade** して、互換性を確保し、障害を回避します。

Parameters**SERVER_NAME**

削除するバックアップを保持するサーバーの名前。

SOURCE_URL

AWS S3 のバケットなどのクラウドソースの URL。例 `s3://bucket/path/to/folder`。

-V / --version

バージョンを表示して終了します。

--help

このヘルプメッセージを表示して終了します。

-v / --verbose

出力の詳細度を増やしますたとえば、`-vv` は `-v` より大きいです。

-q / --quiet

出力の詳細度を減らしますたとえば、`-qq` は `-q` より小さいです。

-t / --test

クラウドの接続をテストして終了します。

--cloud-provider

ストレージバックエンドとして使用するクラウドプロバイダー。

許可されるオプションは次のとおりです。

- `aws-s3`。
- `azure-blob-storage`。
- `google-cloud-storage`。

-b / --backup-id

削除するバックアップの ID。バックアップ ID の代わりにショートカットを使用できます。

-m / --minimum-redundancy

常に使用可能なバックアップの最小数。

-r / --retention-policy

指定した場合、提供された保持ポリシーに従って、削除の対象となるすべてのバックアップを削除し

ます。

構文 REDUNDANCY value | RECOVERY WINDOW OF value { DAYS | WEEKS | MONTHS }

--batch-size

クラウドプロバイダーへの 1 回の要求で削除されるオブジェクトの最大数。設定されていない場合、指定されたクラウドプロバイダーの最大許可バッチサイズが使用されます aws-s3 の場合は 1000、azure-blob-storage の場合は 256、および google-cloud-storage の場合は 100。

--dry-run

削除する必要があるオブジェクトを見つけますが、削除しません。

Extra options for the AWS cloud provider

--check-object-lock

削除する前に、バックアップファイルが S3 オブジェクトロックで保護されているかどうかを確認します。S3 バケットにオブジェクトロックが構成されており、削除操作がオブジェクトレベルの保持ポリシーを尊重したい場合に、これを有効にします。ベースバックアップファイルがアクティブな保持期間またはリーガルホールドによってロックされている場合、削除はエラーで中止されます。

パフォーマンス上の理由から、ロックチェックはベースバックアップファイルに対してのみ実行されます。ベースバックアップが削除されると、対応する WAL ファイルは使用できなくなるためです。このチェックでは、ロックステータスを確認するためにファイルごとに 1 つの要求が必要であるため、多くのファイルを含むバックアップの削除時間が増加する可能性があることに注意してください。この機能は AWS S3 互換オブジェクトストアでのみ利用でき、s3:GetObjectRetention および s3:GetObjectLegalHold IAM 権限も必要です。

--endpoint-url

デフォルトの S3 エンドポイント URL を指定されたものでオーバーライドします。

-P / --aws-profile

プロファイル名 AWS 資格情報ファイルの INI セクションなど。

--profile 非推奨

プロファイル名 AWS 資格情報ファイルの INI セクションなど --aws-profile に置き換えられます。

--read-timeout

接続からの読み取りを待機しているときにタイムアウトが発生するまでの秒単位の時間デフォルト 60 秒。

Extra options for the Azure cloud provider

--azure-credential / --credential

オプションで、Azure で認証するときに使用する資格情報の種類を指定します。省略すると、Azure Blob Storage の資格情報が環境から取得され、デフォルトの Azure 認証フローが他のすべての Azure サービスでの認証に使用されます。環境で資格情報が見つからない場合、デフォルトの Azure 認証フローは Azure Blob Storage にも使用されます。

許可されるオプションは次のとおりです。

- azure-cli。
- managed-identity。
- default。

17.4.3 barman-cloud-backup-show

Synopsis

```
barman-cloud-backup-show
    [ { -V | --version } ]
    [ --help ]
    [ { { -v | --verbose } | { -q | --quiet } } ]
    [ { -t | --test } ]
    [ --cloud-provider { aws-s3 | azure-blob-storage | google-cloud-
→storage } ]
    [ --endpoint-url ENDPOINT_URL ]
    [ { -P | --aws-profile } AWS_PROFILE ]
    [ --profile AWS_PROFILE ]
    [ --read-timeout READ_TIMEOUT ]
    [ { --azure-credential | --credential } { azure-cli | managed-
→identity | default } ]
    [ --format FORMAT ]
    SOURCE_URL SERVER_NAME BACKUP_ID
```

Description

このスクリプトは、`barman-cloud-backup` コマンドで作成された特定のバックアップに関する詳細情報を表示します。出力は、`barman-cloud-backup` コマンドリファレンスの `barman show-backup` に似ていますが、情報が少ないです。

注釈

GCP の場合、`GOOGLE_APPLICATION_CREDENTIALS` env での認証のみがサポートされています。

Parameters

BACKUP_ID

バックアップの ID。バックアップ ID の代わりにショートカットを使用できます。

SERVER_NAME

表示するバックアップを保持するサーバーの名前。

SOURCE_URL

AWS S3 のバケットなどのクラウドソースの URL。例 `s3://bucket/path/to/folder`。

-V / --version

バージョンを表示して終了します。

--help

このヘルプメッセージを表示して終了します。

-v / --verbose

出力の詳細度を増やしますたとえば、`-vv` は `-v` より大きいです。

-q / --quiet

出力の詳細度を減らしますたとえば、`-qq` は `-q` より小さいです。

-t / --test

クラウドの接続をテストして終了します。

--cloud-provider

ストレージバックエンドとして使用するクラウドプロバイダー。

許可されるオプションは次のとおりです。

- `aws-s3`。
- `azure-blob-storage`。
- `google-cloud-storage`。

--format

出力フォーマット `console` または `json`。デフォルト `console`。

Extra options for the AWS cloud provider

--endpoint-url

デフォルトの S3 エンドポイント URL を指定されたものでオーバーライドします。

-P / --aws-profile

プロファイル名 AWS 資格情報ファイルの INI セクションなど。

--profile 非推奨

プロファイル名 AWS 資格情報ファイルの INI セクションなど `--aws-profile` に置き換えられます。

--read-timeout

接続からの読み取りを待機しているときにタイムアウトが発生するまでの秒単位の時間デフォルト 60 秒。

Extra options for the Azure cloud provider

--azure-credential / --credential

オプションで、Azure で認証するときに使用する資格情報の種類を指定します。省略すると、Azure Blob Storage の資格情報が環境から取得され、デフォルトの Azure 認証フローが他のすべての Azure サービスでの認証に使用されます。環境で資格情報が見つからない場合、デフォルトの Azure 認証フローは Azure Blob Storage にも使用されます。

許可されるオプションは次のとおりです。

- `azure-cli`。
- `managed-identity`。
- `default`。

17.4.4 barman-cloud-backup-list

Synopsis

```
barman-cloud-backup-list
    [ { -V | --version } ]
    [ --help ]
    [ { { -v | --verbose } | { -q | --quiet } } ]
    [ { -t | --test } ]
    [ --cloud-provider { aws-s3 | azure-blob-storage | google-cloud-
→storage } ]
    [ --endpoint-url ENDPOINT_URL ]
    [ { -P | --aws-profile } AWS_PROFILE ]
    [ --profile AWS_PROFILE ]
    [ --read-timeout READ_TIMEOUT ]
    [ { --azure-credential | --credential } { azure-cli | managed-
→identity | default } ]
    [ --format FORMAT ]
    SOURCE_URL SERVER_NAME
```

Description

このスクリプトは、クラウドに保存されている `barman-cloud-backup` コマンドを使用して作成されたバックアップをリストします。

注釈

GCP の場合、`GOOGLE_APPLICATION_CREDENTIALS` env での認証のみがサポートされています。

Parameters

SERVER_NAME

リストするバックアップを保持するサーバーの名前。

SOURCE_URL

AWS S3 のバケットなどのクラウドソースの URL。例 `s3://bucket/path/to/folder`。

-V / --version

バージョンを表示して終了します。

--help

このヘルプメッセージを表示して終了します。

-v / --verbose

出力の詳細度を増やしますたとえば、`-vv` は `-v` より大きいです。

-q / --quiet

出力の詳細度を減らしますたとえば、`-qq` は `-q` より小さいです。

-t / --test

クラウドの接続をテストして終了します。

--cloud-provider

ストレージバックエンドとして使用するクラウドプロバイダー。

許可されるオプションは次のとおりです。

- `aws-s3`。
- `azure-blob-storage`。
- `google-cloud-storage`。

--format

出力フォーマット `console` または `json`。デフォルト `console`。

Extra options for the AWS cloud provider**--endpoint-url**

デフォルトの S3 エンドポイント URL を指定されたものでオーバーライドします。

-P / --aws-profile

プロファイル名 AWS 資格情報ファイルの INI セクションなど。

--profile 非推奨

プロファイル名 AWS 資格情報ファイルの INI セクションなど `--aws-profile` に置き換えられます。

--read-timeout

接続からの読み取りを待機しているときにタイムアウトが発生するまでの秒単位の時間デフォルト 60 秒。

Extra options for the Azure cloud provider**--azure-credential / --credential**

オプションで、Azure で認証するときに使用する資格情報の種類を指定します。省略すると、Azure Blob Storage の資格情報が環境から取得され、デフォルトの Azure 認証フローが他のすべての Azure サービスでの認証に使用されます。環境で資格情報が見つからない場合、デフォルトの Azure 認証フローは Azure Blob Storage にも使用されます。

許可されるオプションは次のとおりです。

- `azure-cli`。
- `managed-identity`。
- `default`。

17.4.5 barman-cloud-backup-keep

Synopsis

```
barman-cloud-backup-keep
    [ { -V | --version } ]
    [ --help ]
    [ { { -v | --verbose } | { -q | --quiet } } ]
    [ { -t | --test } ]
```

(次のページに続く)

(前のページからの続き)

```

[ --cloud-provider { aws-s3 | azure-blob-storage | google-cloud-
→storage } ]
[ --endpoint-url ENDPOINT_URL ]
[ { -P | --aws-profile } AWS_PROFILE ]
[ --profile AWS_PROFILE ]
[ --read-timeout READ_TIMEOUT ]
[ { --azure-credential | --credential } { azure-cli | managed-
→identity | default } ]
[ { { -r | --release } | { -s | --status } | --target { full |
→standalone } } ]
SOURCE_URL SERVER_NAME BACKUP_ID

```

Description

このスクリプトを使用して、クラウドストレージのバックアップをアーカイブバックアップとして指定し、保持ポリシーに関係なく無期限の保持を保証します。

このスクリプトを使用すると、`barman-cloud-backup` で以前に作成したバックアップをアーカイブバックアップとしてマークできます。アーカイブとしてフラグが設定されると、これらのバックアップは無期限に保存され、標準の保持ポリシーの対象ではありません。

注釈

GCP の場合、`GOOGLE_APPLICATION_CREDENTIALS` env での認証のみがサポートされています。

重要

AWS boto3 1.36 から、**Data Integrity Protection checks** の動作が変更されました。Barman が使用する一部のメソッドは、`Content-MD5` ヘッダーを必要としません。

これは、新しい boto3 バージョンで使用した場合の **S3-compatible storage providers that have not updated their server-side code may fail** を意味します。たとえば、MinIO は、boto3 1.36 発表の直後にこの変更に対応しました。

MinIO を使用している場合、最新のリリースまたは少なくとも `RELEASE.2025-02-03T21-03-04Z` 以降に **must upgrade** して、互換性を確保し、keep アノテーションをリリースする際の障害を回避します。

Parameters

SERVER_NAME

保持するバックアップを保持するサーバーの名前。

SOURCE_URL

AWS S3 のバケットなどのクラウドソースの URL。例 `s3://bucket/path/to/folder`。

BACKUP_ID

保持するバックアップの ID。バックアップ ID の代わりにショートカットを使用できます。

-V / --version

バージョンを表示して終了します。

--help

このヘルプメッセージを表示して終了します。

-v / --verbose

出力の詳細度を増やしますたとえば、`-vv` は `-v` より大きいです。

-q / --quiet

出力の詳細度を減らしますたとえば、`-qq` は `-q` より小さいです。

-t / --test

クラウドの接続をテストして終了します。

--cloud-provider

ストレージバックエンドとして使用するクラウドプロバイダー。

許可されるオプションは次のとおりです。

- `aws-s3`。
- `azure-blob-storage`。
- `google-cloud-storage`。

-r / --release

指定すると、コマンドは keep アノテーションを削除し、バックアップは削除の対象になります。

-s / --status

バックアップの keep ステータスを出力します。

--target

このバックアップのリカバリターゲットを指定します。許可されるオプションは次のとおりです。

- `full`
- `standalone`

Extra options for the AWS cloud provider

--endpoint-url

デフォルトの S3 エンドポイント URL を指定されたものでオーバーライドします。

-P / --aws-profile

プロファイル名 AWS 資格情報ファイルの INI セクションなど。

--profile 非推奨

プロファイル名 AWS 資格情報ファイルの INI セクションなど `--aws-profile` に置き換えられます。

--read-timeout

接続からの読み取りを待機しているときにタイムアウトが発生するまでの秒単位の時間デフォルト 60 秒。

Extra options for the Azure cloud provider

--azure-credential / --credential

オプションで、Azure で認証するときに使用する資格情報の種類を指定します。省略すると、Azure Blob Storage の資格情報が環境から取得され、デフォルトの Azure 認証フローが他のすべての Azure サービスでの認証に使用されます。環境で資格情報が見つからない場合、デフォルトの Azure 認証フローは Azure Blob Storage にも使用されます。

許可されるオプションは次のとおりです。

- `azure-cli`。
- `managed-identity`。
- `default`。

17.4.6 barman-cloud-check-wal-archive**Synopsis**

```
barman-cloud-check-wal-archive
    [ { -V | --version } ]
    [ --help ]
    [ { { -v | --verbose } | { -q | --quiet } } ]
    [ { -t | --test } ]
    [ --cloud-provider { aws-s3 | azure-blob-storage | google-cloud-
→storage } ]
    [ --endpoint-url ENDPOINT_URL ]
    [ { -P | --aws-profile } AWS_PROFILE ]
    [ --profile AWS_PROFILE ]
    [ --read-timeout READ_TIMEOUT ]
    [ { --azure-credential | --credential }
      { azure-cli | managed-identity | default } ]
    [ --timeline TIMELINE ]
    DESTINATION_URL SERVER_NAME
```

Description

サーバーの WAL アーカイブ宛先が新しい Postgres クラスターでの使用に適していることを確認します。デフォルトでは、WAL アーカイブが空であるか、ターゲットバケットが見つからない場合、チェックは成功します。他の条件では失敗します。

注釈

`barman-cloud-check-wal-archive` コマンドは、初期 HeadBucket 呼び出しを実行して、ターゲットバケットが S3 ストレージに既に存在するかどうかを確認します。バケットが存在しない場合、コマンドは自動的に作成しようとします。

これは、これらの操作バケットの存在チェックと自動作成を実行する唯一の Barman コマンドです。

i 注釈

GCP の場合、GOOGLE_APPLICATION_CREDENTIALS env での認証のみがサポートされています。

Parameters**SERVER_NAME**

確認する必要があるサーバーの名前。

DESTINATION_URL

AWS S3 のバケットなどのクラウド宛先の URL。例 `s3://bucket/path/to/folder`。

-V / --version

バージョンを表示して終了します。

--help

このヘルプメッセージを表示して終了します。

-v / --verbose

出力の詳細度を増やしますたとえば、`-vv` は `-v` より大きいです。

-q / --quiet

出力の詳細度を減らしますたとえば、`-qq` は `-q` より小さいです。

-t / --test

クラウドの接続をテストして終了します。

--cloud-provider

ストレージバックエンドとして使用するクラウドプロバイダー。

許可されるオプションは次のとおりです。

- `aws-s3`。
- `azure-blob-storage`。
- `google-cloud-storage`。

--timeline

WAL がチェックを失敗させる最も古いタイムライン。

Extra options for the AWS cloud provider**--endpoint-url**

デフォルトの S3 エンドポイント URL を指定されたものでオーバーライドします。

-P / --aws-profile

プロファイル名 AWS 資格情報ファイルの INI セクションなど。

--profile 非推奨

プロファイル名 AWS 資格情報ファイルの INI セクションなど `--aws-profile` に置き換えられます。

--read-timeout

接続からの読み取りを待機しているときにタイムアウトが発生するまでの秒単位の時間デフォルト 60 秒。

Extra options for the Azure cloud provider

--azure-credential / --credential

オプションで、Azure で認証するときに使用する資格情報の種類を指定します。省略すると、Azure Blob Storage の資格情報が環境から取得され、デフォルトの Azure 認証フローが他のすべての Azure サービスでの認証に使用されます。環境で資格情報が見つからない場合、デフォルトの Azure 認証フローは Azure Blob Storage にも使用されます。

許可されるオプションは次のとおりです。

- `azure-cli`。
- `managed-identity`。
- `default`。

17.4.7 barman-cloud-restore

Synopsis

```
barman-cloud-restore
    [ { -V | --version } ]
    [ --help ]
    [ { { -v | --verbose } | { -q | --quiet } } ]
    [ { -t | --test } ]
    [ --cloud-provider { aws-s3 | azure-blob-storage | google-cloud-
→storage } ]
    [ --endpoint-url ENDPOINT_URL ]
    [ { -P | --aws-profile } AWS_PROFILE ]
    [ --profile AWS_PROFILE ]
    [ --read-timeout READ_TIMEOUT ]
    [ { --azure-credential | --credential } { azure-cli | managed-
→identity | default } ]
    [ --snapshot-recovery-instance SNAPSHOT_RECOVERY_INSTANCE ]
    [ --snapshot-recovery-zone GCP_ZONE ]
    [ --aws-region AWS_REGION ]
    [ --gcp-zone GCP_ZONE ]
    [ --azure-resource-group AZURE_RESOURCE_GROUP ]
    [ --tablespace NAME:LOCATION [ --tablespace NAME:LOCATION ... ] ]
    [ --target-lsn LSN ]
    [ --target-time TIMESTAMP ]
    [ --target-tli TLI ]
    SOURCE_URL SERVER_NAME BACKUP_ID RECOVERY_DESTINATION
```

Description

このスクリプトを使用して、`barman-cloud-backup` コマンドで作成されたクラウドストレージからバックアップを直接復元します。さらに、このスクリプトは、接続されたディスクが正しいスナップショットからクローン化されたことを確認し、オブジェクトストレージからバックアップラベルをダウンロードすることにより、スナップショットバックアップからのリカバリーの準備をできます。

このコマンドは、リカバリーのために Postgres を自動的に準備しません。Postgres を確実に起動するには、手動または外部ツールを使用して、*PITR オプション*、*カスタム `restore\_command` 値*、シグナルファイル、または必要な WAL ファイルを手動で管理する必要があります。

i 注釈

GCP の場合、`GOOGLE_APPLICATION_CREDENTIALS` env での認証のみがサポートされています。

Parameters**SERVER_NAME**

復元するバックアップを保持するサーバーの名前。

SOURCE_URL

AWS S3 のバケットなどのクラウドソースの URL。例 `s3://bucket/path/to/folder`。

BACKUP_ID

復元するバックアップの ID。バックアップ ID の代わりにショートカットを使用できます。それに加えて、`auto` を使用して、Barman に復元操作に最適なバックアップを自動的に見つけさせることができます。

RECOVERY_DESTINATION

リカバリー用のディレクトリへのパス。

-V / --version

バージョンを表示して終了します。

--help

このヘルプメッセージを表示して終了します。

-v / --verbose

出力の詳細度を増やしますたとえば、`-vv` は `-v` より大きいです。

-q / --quiet

出力の詳細度を減らしますたとえば、`-qq` は `-q` より小さいです。

-t / --test

クラウドの接続をテストして終了します。

--cloud-provider

ストレージバックエンドとして使用するクラウドプロバイダー。

許可されるオプションは次のとおりです。

- `aws-s3`。
- `azure-blob-storage`。
- `google-cloud-storage`。

--snapshot-recovery-instance

スナップショットから復元されたディスクが接続されているインスタンス。

--tablespace

テーブルスペースの再配置ルール。

--target-lsn

リカバリターゲット lsn、たとえば 3/64000000。

--target-time

タイムゾーンの有無にかかわらず、%Y-%m-%d %H:%M:%S 形式でのリカバリターゲットのタイムスタンプ。

--target-tli

リカバリターゲットのタイムライン。

Extra options for the AWS cloud provider**--endpoint-url**

デフォルトの S3 エンドポイント URL を指定されたものでオーバーライドします。

-P / --aws-profile

プロファイル名 AWS 資格情報ファイルの INI セクションなど。

--profile 非推奨

プロファイル名 AWS 資格情報ファイルの INI セクションなど --aws-profile に置き換えられます。

--read-timeout

接続からの読み取りを待機しているときにタイムアウトが発生するまでの秒単位の時間デフォルト 60 秒。

--aws-region

EC2 VM と、--snapshot-instance および --snapshot-disk 引数で定義されたストレージボリュームを含む AWS リージョンの名前。

Extra options for the Azure cloud provider**--azure-credential / --credential**

オプションで、Azure で認証するとき使用する資格情報の種類を指定します。省略すると、Azure Blob Storage の資格情報が環境から取得され、デフォルトの Azure 認証フローが他のすべての Azure サービスでの認証に使用されます。環境で資格情報が見つからない場合、デフォルトの Azure 認証フローは Azure Blob Storage にも使用されます。

許可されるオプションは次のとおりです。

- azure-cli。
- managed-identity。
- default。

--azure-resource-group

--snapshot-instance および --snapshot-disk 引数で定義されたコンピューティングインスタンスとディスクが属する Azure リソースグループの名前。

Extra options for GCP cloud provider**--gcp-zone**

スナップショットを取得するディスクのゾーン。

--snapshot-recovery-zone 非推奨

スナップショットリカバリーのインスタンスとディスクを含むゾーン `--gcp-zone` に置き換えられます。

17.4.8 barman-cloud-wal-archive**Synopsis**

```
barman-cloud-wal-archive
    [ { -V | --version } ]
    [ --help ]
    [ { { -v | --verbose } | { -q | --quiet } } ]
    [ { -t | --test } ]
    [ --cloud-provider { aws-s3 | azure-blob-storage | google-cloud-
→storage } ]
    [ { { -z | --gzip } | { -j | --bzip2 } | --xz | --snappy | --zstd | --
→lz4 } ]
    [ --compression-level COMPRESSION_LEVEL ]
    [ --tag KEY,VALUE [ --tag KEY,VALUE ... ] ]
    [ --history-tag KEY,VALUE [ --history-tag KEY,VALUE ... ] ]
    [ --endpoint-url ENDPOINT_URL ]
    [ { -P | --aws-profile } AWS_PROFILE ]
    [ --profile AWS_PROFILE ]
    [ --read-timeout READ_TIMEOUT ]
    [ { -e | --encryption } ENCRYPTION ]
    [ --sse-kms-key-id SSE_KMS_KEY_ID ]
    [ { --azure-credential | --credential } { azure-cli | managed-
→identity |
        default } ]
    [ --encryption-scope ENCRYPTION_SCOPE ]
    [ --max-block-size MAX_BLOCK_SIZE ]
    [ --max-concurrency MAX_CONCURRENCY ]
    [ --max-single-put-size MAX_SINGLE_PUT_SIZE ]
    [ --kms-key-name KMS_KEY_NAME ]
    DESTINATION_URL SERVER_NAME [ WAL_PATH ]
```

Description

`barman-cloud-wal-archive` コマンドは、Postgres サーバーの `archive_command` で使用して、WAL ファイルをクラウドストレージに直接シップするように設計されています。

注釈

Python 2 または Python 3 のサポートされていないバージョンを使用している場合は、圧縮オプション `--gzip` または `--bzip2` の使用を避けてください。このスクリプトは、Python < 3.2 で `gzip` 圧縮された WAL、または Python < 3.3 で `bzip2` 圧縮された WAL を復元できません。

このスクリプトにより、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転送できます。さらに、WAL アーカイブ `pre_archive_retry_script` のフックスクリプトとして利用できます。

注釈

GCP の場合、`GOOGLE_APPLICATION_CREDENTIALS` env での認証のみがサポートされています。

Parameters

SERVER_NAME

WAL をアーカイブするサーバーの名前。

DESTINATION_URL

AWS S3 のバケットなどのクラウド宛先の URL。例 `s3://bucket/path/to/folder`。

WAL_PATH

'%p' キーワードの値 `archive_command` によると。

-V / --version

バージョンを表示して終了します。

--help

このヘルプメッセージを表示して終了します。

-v / --verbose

出力の詳細度を増やしますたとえば、`-vv` は `-v` より大きいです。

-q / --quiet

出力の詳細度を減らしますたとえば、`-qq` は `-q` より小さいです。

-t / --test

クラウドの接続をテストして終了します。

--cloud-provider

ストレージバックエンドとして使用するクラウドプロバイダー。

許可されるオプションは次のとおりです。

- `aws-s3`。
- `azure-blob-storage`。
- `google-cloud-storage`。

-z / --gzip

クラウドにアップロードするときに WAL を gzip 圧縮します。

-j / --bzip2

クラウドへのアップロード中に WAL を bzip2-圧縮します。

--xz

クラウドにアップロードするときに WAL を xz 圧縮します。

--snappy

クラウドへのアップロード中に WAL を Snappy-圧縮します `python-snappy` Python ライブラリをインストールする必要があります。

--zstd

クラウドへのアップロード中に WAL を zstd 圧縮します `zstandard` Python ライブラリをインストールする必要があります。

--lz4

クラウドへのアップロード中に WAL を lz4 圧縮します `lz4` Python ライブラリをインストールする必要があります。

--compression-level

選択した圧縮アルゴリズムで使用される圧縮レベル。有効な値は、選択したアルゴリズムまたは事前定義されたラベルのいずれか `low`、`medium`、および `high` のサポートされている範囲内の整数です。各アルゴリズムの範囲と、各事前定義されたラベルがマップするレベルは `low` で見つかります。

--tag

クラウドストレージにアーカイブされた WAL ファイルに追加されるタグ。

--tags

クラウドストレージにアーカイブされた WAL ファイルに追加されるタグ。

i 注釈

位置引数の前に `--tags` を使用している場合、その後に `--` を挿入してオプショナル引数の終了を示す必要があります。これは、`--` 以降のすべてを位置引数として扱うようにパーサーに指示します。`--` がないと、Barman は位置引数を最後のオプションの値と誤って解釈する可能性があります。

バージョン 3.15 で非推奨: `--tags` は非推奨です。代わりに `--tag` を使用してください。

--history-tag

クラウドストレージにアーカイブされた履歴ファイルに追加されるタグ。

--history-tags

クラウドストレージにアーカイブされた履歴ファイルに追加されるタグ。

i 注釈

位置引数の前に `--history-tags` を使用している場合、その後に `--` を挿入してオプショナル引数の終了を示す必要があります。これは、`--` 以降のすべてを位置引数として扱うようにパーサーに指示します。`--` がないと、Barman は位置引数を最後のオプションの値と誤って解釈する可能性があります。

バージョン 3.15 で非推奨: `--history-tags` は非推奨です。代わりに `--history-tag` を使用してください。

Extra options for the AWS cloud provider**--endpoint-url**

デフォルトの S3 エンドポイント URL を指定されたものでオーバーライドします。

-P / --aws-profile

プロファイル名 AWS 資格情報ファイルの INI セクションなど。

--profile 非推奨

プロファイル名 AWS 資格情報ファイルの INI セクションなど --aws-profile に置き換えられます。

--read-timeout

接続からの読み取りを待機しているときにタイムアウトが発生するまでの秒単位の時間デフォルト 60 秒。

-e / --encryption

アップロードされたデータを S3 に保存するときに使用される暗号化アルゴリズム。

許可されるオプション

- AES256。
- aws:kms。

--sse-kms-key-id

S3 にアップロードされたデータの暗号化に使用する AWS KMS キー ID。独自のキー ID を使用して、またはキーの完全な ARN を使用して指定できます。-e / --encryption が aws:kms に設定されている場合のみ許可されます。

Extra options for the Azure cloud provider**--azure-credential / --credential**

オプションで、Azure で認証するときに使用する資格情報の種類を指定します。省略すると、Azure Blob Storage の資格情報が環境から取得され、デフォルトの Azure 認証フローが他のすべての Azure サービスでの認証に使用されます。環境で資格情報が見つからない場合、デフォルトの Azure 認証フローは Azure Blob Storage にも使用されます。

許可されるオプションは次のとおりです。

- azure-cli。
- managed-identity。
- default。

--encryption-scope

Azure のデータの暗号化に使用される Azure Blob Storage サービスで定義された暗号化スコープの名前。

--max-block-size

コンカレントチャンク方法デフォルト 4MB を介してオブジェクトをアップロードするときに使用されるチャンクサイズ。

--max-concurrency

同時にアップロードされるチャンクの最大数デフォルト 1。

--max-single-put-size

Azure クライアントが単一の要求でオブジェクトをアップロードする最大サイズデフォルト 64MB。これが適用された圧縮後の Postgres WAL セグメントサイズよりも低く設定されている場合、WAL アーカイブのコンカレントチャンクアップロード方法が使用されます。

Extra options for GCP cloud provider

--kms-key-name

GCS でアップロードされたデータの暗号化に使用する GCP KMS キーの名前。

17.4.9 barman-cloud-wal-restore

Synopsis

```
barman-cloud-wal-restore
    [ { -V | --version } ]
    [ --help ]
    [ { { -v | --verbose } | { -q | --quiet } } ]
    [ { -t | --test } ]
    [ --cloud-provider { aws-s3 | azure-blob-storage | google-cloud-
→storage } ]
    [ --endpoint-url ENDPOINT_URL ]
    [ { -P | --aws-profile } AWS_PROFILE ]
    [ --profile AWS_PROFILE ]
    [ --read-timeout READ_TIMEOUT ]
    [ { --azure-credential | --credential } { azure-cli | managed-identity
    | default } ]
    [ --no-partial ]
    SOURCE_URL SERVER_NAME WAL_NAME WAL_DEST
```

Description

barman-cloud-wal-restore スクリプトは、クラウドストレージから WAL ファイルを取得し、Barman サーバーをバイパスして Postgres スタンバイサーバーに直接配置するための `restore_command` として機能します。

このスクリプトは、以前に `barman-cloud-wal-archive` コマンドでアーカイブされた WAL ファイルをダウンロードするために使用されます。 `--no-partial` オプションを呼び出して、`.partial` ファイルの自動ダウンロードを無効にします。

重要

ターゲット Postgres ノードで、`pg_wal` とスプールディレクトリが同じファイルシステム上にある場合、ファイルは名前変更を介して移動されます。これは、コピーや削除よりも高速です。これにより、WAL ファイルの提供が大幅に高速化されます。ディレクトリが別のファイルシステム上にある場合でも、プロセスにはコピーと削除が含まれるため、その場合パフォーマンスの向上はありません。

注釈

GCP の場合、`GOOGLE_APPLICATION_CREDENTIALS` env での認証のみがサポートされています。

Parameters

SERVER_NAME

WAL が復元されるサーバーの名前。

SOURCE_URL

AWS S3 のバケットなどのクラウドソースの URL。例 `s3://bucket/path/to/folder`。

WAL_NAME

'%f' キーワードの値 `restore_command` によると。

WAL_DEST

'%p' キーワードの値 `restore_command` によると。

-V / --version

バージョンを表示して終了します。

--help

このヘルプメッセージを表示して終了します。

-v / --verbose

出力の詳細度を増やしますたとえば、`-vv` は `-v` より大きいです。

-q / --quiet

出力の詳細度を減らしますたとえば、`-qq` は `-q` より小さいです。

-t / --test

クラウドの接続をテストして終了します。

--cloud-provider

ストレージバックエンドとして使用するクラウドプロバイダー。

許可されるオプションは次のとおりです。

- `aws-s3`。
- `azure-blob-storage`。
- `google-cloud-storage`。

--no-partial

部分的な WAL ファイルをダウンロードしないでください

Extra options for the AWS cloud provider**--endpoint-url**

デフォルトの S3 エンドポイント URL を指定されたものでオーバーライドします。

-P / --aws-profile

プロファイル名 AWS 資格情報ファイルの INI セクションなど。

--profile 非推奨

プロファイル名 AWS 資格情報ファイルの INI セクションなど `--aws-profile` に置き換えられます。

--read-timeout

接続からの読み取りを待機しているときにタイムアウトが発生するまでの秒単位の時間デフォルト 60 秒。

Extra options for the Azure cloud provider

--azure-credential / --credential

オプションで、Azure で認証するときに使用する資格情報の種類を指定します。省略すると、Azure Blob Storage の資格情報が環境から取得され、デフォルトの Azure 認証フローが他のすべての Azure サービスでの認証に使用されます。環境で資格情報が見つからない場合、デフォルトの Azure 認証フローは Azure Blob Storage にも使用されます。

許可されるオプションは次のとおりです。

- `azure-cli`。
- `managed-identity`。
- `default`。

第18章 用語集

AWS

アマゾン ウェブ サービス

Barman

バックアップ リカバリーマネージャー。

DBA

データベース管理者。

DEB

Debian パッケージ。

外部構成ファイル

PGDATA の外部に保存されているファイル。たとえば、Ubuntu で `pg_createcluster sudo pg_createcluster 14 main` などを使用して新しい PostgreSQL クラスターを作成すると、新しいデータディレクトリ、通常は `/var/lib/postgresql/` の下に設定されますが、`postgresql.conf` や `pg_hba.conf` のような構成ファイルは、通常は `/etc/postgresql/<version>/main/` の下に保存されて分離されます実際のデータから。

GCP

Google Cloud Platform

GPG

GNU プライバシーガード

IAC

コードとしてのインフラストラクチャ

ICT

情報通信技術。

libpq

Postgres への C アプリケーションプログラマのインターフェイス。libpq は、クライアントプログラムがクエリを Postgres バックエンドサーバーに渡し、これらのクエリの結果を受信できるようにするライブラリ関数セットです。

PGDATA

PostgreSQL データディレクトリ。

PGDG

Postgres グローバル開発グループ。

PITR

ポイント インタイム リカバリー。

RHEL

Red Hat Enterprise Linux。

RPM

Red Hat パッケージマネージャー。

RPO

目標リカバリーポイント。重大なインシデントにより IT サービスからデータが失われる可能性のある最大目標期間。要約すると、これは、損失が許容されるデータの最大量を表します。

SLES

SUSE Linux Enterprise Server

SPOF

単一障害点

VLDB

非常に大規模なデータベース

WORM

一度書いて、たくさん読んでください

第19章 リリースノート

19.1 Barman release notes

© Copyright EnterpriseDB UK Limited 2025 - All rights reserved.

19.1.1 3.17.0 (2026-01-07)

Notable changes

- Enable query, fetch, and selective operations on inactive/disabled servers

Multiple Barman commands now work with inactive and disabled servers based on their safety profile:

Query commands:

- `list-backups`: view backups from inactive/disabled servers
- `status`: check status of inactive/disabled servers
- `show-backup`: display backup details from inactive/disabled servers
- `list-files`: list files from backups on inactive/disabled servers
- `verify-backup`: verify backups on inactive/disabled servers

Fetch commands:

- `restore`: restore backups from inactive servers
- `get-wal`: retrieve WAL files from inactive servers

Modification commands:

- `delete`: delete backups from inactive servers
- `config-switch`: switch configuration models for inactive/disabled servers

This change improves operational flexibility by allowing users to query backup information and perform restore operations even when servers are manually disabled (inactive) or have configuration issues (disabled). The different behavior for disabled servers provides safety against operations on servers with potential configuration conflicts or path issues.

References: BAR-976.

- Deprecate custom compression for WAL files

The custom compression option and its related configuration options (`custom_compression_filter`, `custom_decompression_filter`, and `custom_compression_magic`) are now deprecated and will be removed in a future release.

Users currently using custom compression are advised to migrate to one of the built-in compression algorithms supported by Barman: `gzip`, `bzip2`, `lz4`, `xz`, `zstd`, or `pigz`.

The custom compression feature was designed to provide flexibility for users with specific compression requirements, but maintaining compatibility with arbitrary external compression tools has proven to be complex and error-prone. The built-in compression options provide better performance, reliability, and maintainability.

Existing configurations using custom compression will continue to work in the current release, but users should plan to migrate their configurations before the feature is removed.

References: BAR-993.

Minor changes

- Handle non-UTF8 output when checking for file encryption

In certain scenarios, such as when recovering WALs via `barman-wal-restore` or running `barman rebuild-xlogdb`, Barman executes the `file` command to check whether a file is encrypted.

On macOS systems (which are not officially supported), this command was reported to be raising a `UnicodeDecodeError` exception if its output contained bytes that could not be decoded using UTF-8.

To prevent this, Barman now handles undecodable bytes gracefully, allowing the process to continue without interruption.

Github issue: #1117

References: BAR-926.

- Add `--recovery-option-port` to `barman restore` command

When doing a remote restore with `get-wal` mode, Barman writes a `restore_command` which invokes `barman-wal-restore` to fetch WALs. However, there was currently no way to specify a custom port to the written `barman-wal-restore` command.

This change introduces the `--recovery-option-port` flag to the `barman restore` command, allowing users to specify a value for the `--port` option of `barman-wal-restore` command during remote restores.

References: BAR-943.

- Add S3 Object Lock support for the deletion of backups in `barman-cloud`

`barman-cloud-backup-delete` now supports checking for Object Lock before deleting backups in AWS S3. This feature helps prevent deletion of objects protected by compliance or governance retention policies.

A new `--check-object-lock` option has been added, which enables Object Lock verification before deletion. When enabled, the deletion operation for all objects of a base backup will be aborted if any of those objects is locked. For performance reasons, lock checks are only performed for base backup files, since once the base backup is removed, the corresponding WAL files are no longer useful.

Note that this feature adds overhead to the delete operation, as it requires an additional API call for each object being deleted. This feature is only available for AWS S3-compatible object stores and requires the appropriate permissions to function correctly.

References: BAR-946.

Bugfixes

- Increase coverage of non-compliant S3 object stores for backup deletion

With this fix we increase the coverage of non-compliant S3 object stores for backup deletion.

We've reduced the expression we were matching for the message returned on failure to catch any error code that returns a message that contains "content-MD5".

We believe with this, almost all non-compliant S3 object stores for backup deletion will be covered

References: BAR-974.

- Fix passive node not syncing

In Barman 3.16 the "name" field was removed from the "config" key of the `barman diagnose` command. Turns out the same "config" data is also used when syncing a passive node, and the absence of the "name" field caused the sync to fail.

This change adds back the "name" field to the "config" key, fixing the sync of passive nodes.

Reported in GitHub issue #1125.

References: BAR-971.

- Fix delta restore not working

The delta restore feature was not functioning correctly due to an oversight where Barman was removing the destination directory before performing a restore. The problem was fixed by avoiding the removal of the destination directory on certain operations. Reported in GitHub issue #1139.

References: BAR-1002.

- Fix "rm execution failed" errors during restore

As a consequence of a refactoring implemented during the release of version 3.15, Barman started removing the destination directory of the restore operation before copying backup files. This was to ensure that the files were always being copied to a clean directory.

However, when the destination directory is a mount point, or a subdirectory where the user does not have sufficient permissions on its parent, the removal of the directory fails, leading to errors like "rm execution failed".

The issue is now fixed by avoiding the removal of the destination directory for PGDATA and tablespaces on the restore final destination.

References: BAR-975.

19.1.2 3.16.2 (2025-11-04)

Bugfixes

- Improve barman-cloud delete workaround to inspect error message content

Some S3-compatible storages return an "InvalidRequest" error instead of "MissingContentMD5" during delete operations. Since we cannot rely on the error code alone, this updates the logic to also inspect the error message text to detect MissingContentMD5-like issues and apply the appropriate fallback.

References: BAR-947.

19.1.3 3.16.1 (2025-10-14)

Notable changes

- Automatic bucket creation added to barman-cloud-check-wal-archive

The `barman-cloud-check-wal-archive` command now verifies whether the target bucket exists in the configured cloud storage (e.g., AWS S3) and automatically creates it if it does not.

Previously, automatic bucket creation was handled by `barman-cloud-backup`. This change restores compatibility with CloudNativePG (CNPG) automations that depend on this behavior, which were affected after the release of Barman 3.16.0.

In collaboration with the CloudNativePG team, we have reintroduced this functionality specifically in `barman-cloud-check-wal-archive` to ensure seamless operation of CNPG workflows and internal tests — all while preserving the intended behavior and simplicity for regular Barman users.

References: BAR-931.

19.1.4 3.16.0 (2025-10-02)

Notable changes

- The `restore_command` for local restores no longer includes `sudo`

When using `barman restore --get-wal` for a local restore (i.e. without `--remote-ssh-command`), the generated `restore_command` no longer includes a `sudo -u $USER` prefix. This simplifies the process for the common case where the locally restored Postgres cluster is run by the `barman` user, whom already owns the files by the end of the `barman restore` operation.

Action Required: If you intend to run the restored cluster as a different user (e.g., `postgres`), you must now manually add the appropriate `sudo -u postgres` prefix to the `restore_command` in the recovery configuration and ensure file ownership is correct.

References: BAR-675.

- Deprecate `pygzip` and `pybzip2` compression options

The `pygzip` and `pybzip2` compression options are now deprecated and will be removed in a future release. Users should migrate to the `gzip` and `bzip2` options, which now use Python's internal libraries.

This change was made to improve performance and code consistency. Previously, the `gzip` and `bzip2` options relied on external subprocesses, while their `py` counterparts used native Python libraries. The core compression logic for `gzip` and `bzip2` has been updated to also use Python's built-in libraries, making the `pygzip` and `pybzip2` options redundant.

References: BAR-878.

- Add delta-restore flag and recovery option to barman restore

Introduces the `--delta-restore` flag for barman restore, enabling delta restore mode. This mode restores a backup by reusing identical data already present in the destination directory, which drastically reduces restore time and network traffic.

This feature is also configurable per-server via 'recovery_options' as a new field called delta-restore in the server configuration file.

References: BAR-788.

- S3 Object Deletion Fallback for Object Stores non-compliant with boto3 >=1.36

When using `delete_objects()` with some non-compliant S3-compatible object stores, users could encounter a `MissingContentMD5 ClientError` due to a breaking change in boto3 1.36 that Botocore will no longer automatically compute and populate the `Content-MD5` header.

To address this, Barman now implements a fallback mechanism:

- Bulk deletion via `delete_objects()` is still attempted first.
- If `delete_objects()` fails with `MissingContentMD5`, Barman automatically falls back to deleting objects individually using `delete_object()`, which does not require a `Content-MD5` header.
- Other errors continue to raise exceptions as before.

This change ensures Barman remains compatible with non-compliant object stores without requiring changes on the user's side.

Users with older S3-compatible storage can continue performing backup cleanups without failures, while newer object stores are unaffected.

References: BAR-909.

- Cloud connectivity checks are now only performed with the `-t/--test` flag.

Previously, Barman always tested cloud connectivity and bucket existence for most of the commands, raising errors if connectivity failed or if the bucket did not exist. Both checks are now executed only when the `-t/--test` flag is specified. The only exception was `barman-cloud-check-wal-archive`, which still checks connectivity and the existence of the bucket.

Additionally, buckets are no longer created automatically — Barman assumes the target bucket already exists, making it the user's responsibility to provision the bucket in advance.

References: BAR-895, BAR-908, BAR-918.

Minor changes

- Add combine-mode option to `barman restore` command

Added a `--combine-mode` option to the `barman restore` command, which can be used to specify a copy mode for `pg_combinebackup` when combining incremental backups during a restore. Available modes are `copy`, `link`, `clone`, and `copy-file-range` which are equivalent to the `pg_combinebackup` options `--copy`, `--link`, `--clone`, and `--copy-file-range`, respectively. An equivalent `combine_mode` configuration option can be specified in the configuration file.

This feature allows for greater optimization of the restore process based on the underlying filesystem capabilities.

References: BAR-870.

- Reorganize fields in `barman diagnose` JSON

In the `barman diagnose` JSON, some fields appeared under the server's "config" section even though they were not actual configuration fields.

The fields `"barman_lock_directory"`, `"lock_directory_cleanup"`, and `"config_changes_queue"` were relocated under the "global" key, since they represent global configurations. The "name" field was removed since the server name is already used as the key. Additionally, `"msg_list"` is now stored under the server's name key.

References: BAR-134.

- Add '-p JOBS' to `barman-wal-restore` command when `--get-wal` is `True`

Adds '-p JOBS' to `barman-wal-restore` command when `--get-wal` or `get_wal` configuration option is `True`, JOBS being the number of jobs chosen by the user via `-p` flag or `parallel_jobs` configuration option.

References: BAR-857.

- The ``barman show-backup`` command now displays the backup compression method

The output of the `barman show-backup` command has been enhanced to include a new **Backup Compression** field. If a backup was taken with compression enabled, this field will now display the compression algorithm that was used (e.g., `zstd`). This makes it easier to quickly verify the compression status of a backup.

References: BAR-901.

Bugfixes

- Fix parallelism (`--parallel` option) in `barman-wal-restore`

There has been a regression in `barman-wal-restore` when introducing the `--keep-compression` option in Barman 3.12.0, which made the `--parallel` option not invoke processes as expected. This issue has now been fixed.

References: BAR-865.

- Fix `pg_combinebackup` not being found in `PATH` when `staging_location` is `remote`

When recovering an incremental backup to a remote server with `staging_location` set to `remote`, Barman checks if `pg_combinebackup` is available in the remote PATH. However, a previous issue was making it incorrectly check the local PATH as well, leading to errors if `pg_combinebackup` was not installed locally. The problem has been fixed by ensuring that the check is only performed on the server where `pg_combinebackup` is actually executed.

References: BAR-903.

19.1.5 3.15.0 (2025-08-05)

Notable changes

- Allow incremental backups to be taken with compression and encryption

Previously, Barman prevented incremental backups to be taken if compression was enabled or if the parent backup was already compressed. Since encryption in Barman requires compression, this also meant that incremental backups could not be taken with encryption.

This limitation existed due to the complexity of restoring such backups, which involves multiple staging phases for decryption, decompression, and combination of backups.

After a significant refactoring effort, Barman now fully supports taking incremental backups with both compression and/or encryption. This allows for more flexible backup strategies, enabling users to take advantage of all available features without restrictions.

References: BAR-764, BAR-801, BAR-841.

- Deprecate `local_staging_path` and `recovery_staging_path` configuration options

The `local_staging_path` and `recovery_staging_path` configuration options have been deprecated in favor of the new `staging_path` and `staging_location` options. The old options will continue to work for backwards compatibility, but users are encouraged to migrate to the new configuration options for better clarity and flexibility.

References: BAR-571, BAR-801.

- Improve flexibility and consistency of staging area configuration

Whenever restoring a compressed, encrypted or incremental backup, Barman needs a staging area to handle intermediate files, such as decompressed or decrypted files, or the combined incremental backup data.

Previously, Barman used to have different configuration options for each scenario:

- For incremental backups, the staging area was specified via the `local_staging_path` in the Barman configuration, which could only be a path in the Barman host.
- For compressed backups, the staging area was specified via the `recovery_staging_path`, which could be a path in the Barman host (in case of local recovery) or a path in the target server (in case of remote recovery to a different server).
- For encrypted backups, the staging area was specified via the `local_staging_path` in the Barman configuration, which could only be a path in the Barman host. Note that, since encryption requires compression to be enabled, when restoring an encrypted backup, users needed to specify both `recovery_staging_path` and `local_staging_path`.

Barman now provides a single, unified and more flexible staging area configuration:

- The new `staging_path` configuration option can be used to specify an absolute path for the staging area, where intermediate files will be stored. The default value is `/tmp`.
- The new `staging_location` configuration option can be used to specify whether the staging area is on the Barman host (`local`) or on the remote server (`remote`). The default value is `local`.

This means that, besides having a unified configuration for the staging path, users are now also able to specify with more flexibility where the staging area should be located, either on the Barman host or on the remote Postgres server.

Storage usage has also been optimized to reduce resource consumption whenever possible. This includes:

- Removing intermediate files as soon as they are no longer needed.
- Avoid using the staging area when not necessary, e.g. when restoring a compressed backup locally, no staging is needed and the backup is decompressed directly to its final destination.

This change simplifies configuration and improves consistency and resource usage, making the recovery management more transparent and predictable for end users.

References: BAR-764, BAR-765, BAR-801, BAR-841.

Minor changes

- Add `--staging-path` and `--staging-location` to `barman restore`

Added `--staging-path` and `--staging-location` options to `barman restore` command to allow the user to specify a custom path for the staging area during the restore process. `--staging-path` is the absolute path to be used while `--staging-location` defines its location i.e. `local` or `remote`. They also have its equivalent configuration options in `barman.conf` as `staging_path` and `staging_location` respectively.

References: BAR-570.

- Add option `--if-not-exists` to `barman receive-wal --create-slot`

A new `--if-not-exists` flag has been added to the `barman receive-wal --create-slot` command. This prevents the command from failing when attempting to create a replication slot that has already been created, making it more reliable for use in scripts.

Special thanks to @crazybolillo antonio@zoftko.com a.k.a @antonag32 for creating a PR and contributing to Barman.

References: BAR-527.

- Fail gracefully when an unexpected field is found in the backup metadata.

This update improves how Barman handles backup metadata that includes fields unknown to the current version. This situation can occur when attempting to restore a backup created with a newer version of Barman using an older one. For example, if a newer version adds metadata about encryption, and the older version doesn't recognize that field. That's a case where this change is relevant.

Previously, this would cause an unhelpful exception. Now, Barman will display a clearer message explaining that the backup was created with a newer version and may not be compatible with the current version.

References: BAR-763.

- Improve orphan backup warning for proper cleanup

The warning message for orphan backups has been improved to help users fully clean up incomplete or manually removed backups. Previously, the message did not mention the location of the `backup.info` file, which led to confusion. The updated message now clearly indicates the location of the `backup.info` file to help users properly remove it.

References: BAR-793.

- Improve tag handling in cloud commands to fix parsing errors

The `--tags` command-line option in `barman-cloud-backup` and `barman-cloud-wal-archive` has been replaced to improve parsing reliability and align with standard CLI practices. The previous space-separated format could cause parsing errors, especially when placed before positional arguments.

The commands now accept multiple `--tag` flags, one for each key-value pair.

Before: `barman-cloud-backup ... --tags key1,val1 key2,val2`

After: `barman-cloud-backup ... --tag key1,val1 --tag key2,val2`

The `--tags` option is now deprecated. While existing scripts using the old `--tags` format will continue to work, users are encouraged to update their scripts to the new format. This change is backward compatible and does not introduce a breaking change.

References: BAR-441.

- Clarify WAL Number label in `barman show-backup` output

Rename misleading WAL Number label in `barman show-backup` output to Number of WALs.

References: BAR-791.

- Change "No of files" to "Number of files" in `show-backup` command

In the Wal Information section of the `show-backup` command, the output of "No of files" was changed to "Number of files".

References: BAR-798.

- Validate options passed for servers and models through `config-update`

The `barman config-update` command now performs extra validation on each provided section before enqueueing changes. It verifies, among other things, that all required fields are present, that the provided values conform to the expected types, and that each value also passes its respective parser. In addition, the command will now error out if any of the checks fail. This prevents partial or malformed updates from being applied.

References: BAR-142.

- Add `--check-timeout` argument to `barman check` and `barman backup` commands

A new CLI option has been introduced for the `barman check` and `barman backup` commands to address situations where the check operation duration exceeds the configured `check_timeout`, preventing backups

from being scheduled. This option allows users to disable the timeout or override the default or configured `check_timeout` value.

References: BAR-219.

- Add `--list-empty-directories` to `barman list-files` command

Adds a new `--list-empty-directories` argument to the `list-files` command to add empty directories to the listing.

References: BAR-220.

- Improve archiving check with automatic WAL switching and archiving

To improve the user experience, especially during initial server configuration, the `barman check` command is now more robust.

Previously, when the WAL archive was empty, the check would fail and require manual intervention.

Barman now intelligently handles this by first attempting to archive any existing WAL present in the Barman server. If none exists, it automatically triggers a WAL switch in the Postgres server and tries to archive the resulting file. This enhancement automates a common manual step and makes the initial health check more reliable out-of-the-box.

References: BAR-718.

- Support shortcuts as backup IDs in Barman cloud scripts

It is now possible to specify the shortcuts `first/oldest`, `last/latest` and `last-failed` when referencing backups in the Barman cloud commands. Barman now automatically parses these shortcuts and returns their associated backup ID.

References: BAR-717.

Bugfixes

- Ignore data files of temporary tables during backups

Files that match the pattern of temporary relation files will now be ignored during a `rsync` backup, similar to the exclusion implicitly performed by `pg_basebackup`. This is needed to avoid errors when running Barman in `rsync-concurrent` mode against a Postgres instance that creates and drops many temporary tables, since `rsync` might try to copy a temp file that had already been removed, causing Barman to hang until manually unblocked.

References: BAR-266.

- Improve error message when `pg_combinebackup` is not found

Previously, when restoring an incremental backup and `pg_combinebackup` was not available in the `PATH`, Barman did not handle the exception properly, which made it difficult to understand and fix the error. Now, a proper exception is raised with a clear message indicating that `pg_combinebackup` could not be found in the target server.

References: BAR-840.

- Fail on restore if the target data or tablespace directories are not empty.

Previously, Barman did not verify whether tablespace destination directories were empty before restoring, which could result in their contents being silently overwritten — potentially causing data loss or corruption. Now, the restore process ensures that all destination directories, including those for data and tablespaces, are empty before proceeding. If any are not, Barman will halt the operation and display an appropriate error message, significantly improving safety and data integrity for both local and remote restores.

References: BAR-787.

- Fix errors when deleting orphaned and obsolete backups

Resolved an issue where applying retention policies could fail if a backup was both orphaned and obsolete. Barman now verifies that a base backup directory exists before attempting to delete it, preventing errors when the directory has already been removed.

References: BAR-794.

- Ensure correct WAL segment naming for PG17 boundary LSNs

Addresses a backup hang issue experienced with PostgreSQL 17 and later versions when using barman backup <server> --wait on low-activity instances and modifies the logic within the methods and properties that interact with `pg_walfile_name()` and `pg_walfile_name_offset()`.

As described in barman#1041, PostgreSQL 17 altered the behavior where the end WAL of a backup is now the currently written WAL file, rather than the last completed one. On servers with minimal activity, this current WAL file might not be completed or archived promptly, leading to the backup process stalling indefinitely until a `pg_switch_wal()` occurs.

More details on the PostgreSQL change can be found at <https://www.postgresql.org/docs/release/17.0/>.

References: BAR-519.

- Snapshot Backup Now Supports Non-Root EBS Volumes of Nitro instances on AWS

Resolved an issue where snapshot backups failed to detect non-root EBS volumes on NVMe (Nitro) devices in AWS. Snapshot backups now work reliably with all non-root EBS volumes attached to the instance.

References: BAR-500.

19.1.6 3.14.1 (2025-06-18)

Bugfixes

- Fix WAL archiving performance issues

The Barman WAL archiving process was suffering from a performance degradation due to processing of WAL files that would not be archived in the current run. There was also an oversight in the encryption logic introduced in 3.14, that caused Barman to check for encryption in WAL files streamed to the server, which is unnecessary as streamed WALs are never GPG-encrypted, nor compressed. With both issues fixed, the archiving process will be significantly faster and more resource-efficient.

Thanks to @thealex55 for the detailed analysis in the issue #1087.

References: BAR-775.

19.1.7 3.14.0 (2025-05-15)

Notable changes

- Implementation of GPG encryption for tar backups and WAL files

Implement GPG encryption of tar backups. Encryption starts at the end of the backup, encrypting the backup of PGDATA and tablespaces present in the backup directory. Encrypted backup files will have the .pgp extension added.

Barman supports the decryption and restoration of GPG-encrypted backups using a passphrase obtained through the new `encryption_passphrase_command` configuration option. During the restore process, decrypted files are staged in the `local_staging_path` setting on the Barman host, ensuring a reliable and safe restore process.

New configuration options required for encryption and decryption of backups and WAL files needed to be added. The new options are `encryption`, `encryption_key_id`, and `encryption_passphrase_command`.

WAL files are all encrypted with GPG when `encryption = gpg`. This includes changing the way that xlogdb records are read and written (maintaining backwards compatibility), and a new logic to detect when files are encrypted and the encryption process itself.

Decryption of GPG-encrypted WAL files during the restore process when using the `get-wal` and `no-get-wal` flags of the barman restore command. This extends the functionality added for decrypting backups via the `encryption_passphrase_command` configuration option.

There's a new field in `show-backup` to expose if a backup was encrypted, and specifies the encryption method that was used, if any.

The `barman check` command verifies if the user's encryption settings are correctly configured in the Barman server and functioning as expected.

References: BAR-683, BAR-687, BAR-693, BAR-669, BAR-671, BAR-692, BAR-685, BAR-680, BAR-670, BAR-681, BAR-702.

- Deprecation of Python versions 3.6 and 3.7

As of version 3.14 of Barman, we are deprecating support for Python 3.6 and 3.7, which are not supported by the Python community anymore. Barman still supports Python 3.8 as it's the newest version available on RHEL 8 systems which contains all needed modules.

References: BAR-737.

Minor changes

- Allow compression level to be specified for WAL compression in Barman server

Add a new `compression_level` parameter to the Barman configuration. This option accepts a valid integer value or one of the predefined options: `low`, `medium`, and `high`. Each option corresponds to a different level depending on the compression algorithm chosen.

References: BAR-540.

- Add client-side compression to `barman-wal-archive`

Client-side compression options have been added to `barman-wal-archive`, supporting the same algorithms that are available on a Barman server. When enabled, compression is applied on the client side before sending the WAL to the Barman server. The `--compression-level` parameter allows specifying a desired compression level for the chosen algorithm.

References: BAR-262.

- Add `--compression-level` parameter to `barman-cloud-wal-archive`

A parameter called `compression-level` was added to `barman-cloud-wal-archive`, which allows a level to be specified for the compression algorithm in use.

References: BAR-557.

- Add Snappy compression algorithm to Barman server

The Snappy compression, previously only available in `barman-cloud-wal-archive`, is now also available for standard Barman server. As with all other algorithms, it can be configured by setting `snappy` in the `compression` configuration parameter.

References: BAR-557.

- Introduce the new `list-processes` sub-command for listing the server processes

Add a new `list-processes` command that outputs all active subprocesses for a Barman server. The command displays each process's PID and task.

References: BAR-664.

- Introduce the new `terminate-process` sub-command for terminating Barman subprocesses

Add a new `terminate-process` command that allows users to terminate an active Barman subprocess for a given server by specifying its task name. Barman will terminate the subprocess as long as it belongs to the specified server and it is currently active.

References: BAR-665.

- Remove the pin from boto3 version used in cloud scripts

After thorough investigation of issues with `boto3 >= 1.36`, we've decided to remove the pin that kept the dependency at version 1.35.

Both AWS and MinIO object stores work correctly with the latest version, and using a version of `boto3` that is `>= 1.36` ensures the Barman cloud scripts work in a FIPS-compliant environment.

References: BAR-637.

Bugfixes

- Ensure minimum redundancy check considers only 'non-incremental backups'

An issue was reported where the `minimum_redundancy` rule could be violated due to the inclusion of incremental backups in the redundancy count. As an example: in a scenario where the catalog contained one full backup and two dependent incremental backups, and the user had `minimum_redundancy = 2`, the rule was incorrectly considered satisfied. As a result, deleting the full backup triggered cascading deletion of its incremental dependents, leaving zero backups in the catalog.

This issue has been fixed by updating the `minimum_redundancy` logic to consider only non-incremental backups (i.e. only full, rsync, snapshot). This ensures that full backups cannot be deleted if doing so would violate the configured minimum redundancy level.

References: BAR-707.

- Fix usage of `barman-wal-restore` with `--keep-compression` using `gzip`, `bzip2`, and `pigz` compression algorithms

Fix an issue in `barman-wal-restore` where, when trying to restore WALs compressed with `gzip`, `bzip2` or `pigz` while having `--keep-compression` specified, leading to unexpected errors.

References: BAR-722.

19.1.8 3.13.3 (2025-04-24)

Bugfixes

- Fix local restore of block-level incremental backups

When performing a local restore of block-level incremental backups, Barman was facing errors like the following:

```
ERROR: Destination directory '/home/vagrant/restore/internal_no_get_wal' must be ↵  
↵empty
```

That error was caused by a regression when the option `--staging-wals-directory` was introduced in version 3.13.0. Along with it came a new check to ensure the WAL destination directory was empty before proceeding. However, when restoring block-level incremental backups locally, Barman was setting up the WAL destination directory before performing this check, triggering the error above.

References: BAR-655.

- Fix regression when running `barman-cloud-backup` as a hook

Barman 3.13.2 changed the location of the `backup.info` metadata file as part of the work delivered to fix issues in WORM environments.

However, those changes introduced a regression when using `barman-cloud-backup` as a backup hook in the Barman server: the hook was not aware of the new location of the metadata file.

This update fixes that issue, so `barman-cloud-backup` becomes aware of the new folder structure, and properly locates the `backup.info` file, avoiding runtime failures.

References: BAR-696.

- Avoid decompressing partial WAL files when custom compression is configured

Fixed an issue where Barman attempted to decompress partial WAL files when custom compression was configured. Partial WAL files are never compressed, so any attempt to decompress them is incorrect and caused errors when using the `--partial` flag with `barman-wal-restore` or `barman get-wal`.

References: BAR-697.

- Fixed `barman-cloud-backup` not recycling temporary part files

This fixes a `barman-cloud-backup` problem where temporary part files were not deleted after being uploaded to the cloud, leading to disk space exhaustion. The issue happened only when using Python ≥ 3.12 and it was due to a change in Python that removed the `delete` attribute of named-temporary file objects, which Barman used to rely on when performing internal checks.

References: BAR-674.

- Fixed backup annotations usage in WORM environments

Barman previously stored backup annotation files, used to track operations like `barman keep` and `barman delete`, inside the backup directory itself. These annotations help determine whether a backup should be kept or marked for deletion. However, in WORM environments, files in the backup directory cannot be modified or deleted after a certain period, which caused issues with managing backup states. This fix relocates annotation files to a dedicated metadata directory, as to ensure that such operations function correctly in WORM environments.

References: BAR-663.

19.1.9 3.13.2 (2025-03-27)

Minor changes

- Fix errors when using an immutable storage

Added a new `worm_mode` configuration to enable WORM (Write Once Read Many) handling in Barman, allowing it to support backups on immutable storage.

This fix also provides automatic relocation of the `backup.info` file in a new directory `meta` inside `backup_directory`. This will let Barman update it in future when needed.

Barman will also *not* purge the `wals` directory for WAL files that are not needed when running the first backup. This will add some extra space which will be reclaimed when this first backup is obsolete and removed (by that time, the backups and the WALs will be outside the retention policy window).

Added additional notes to the documentation explaining limitations when running with an immutable storage for backups. In particular the need for a grace period in the immutability of files and the fact that `barman keep` is not supported in these environments.

References: BAR-649, BAR-645, BAR-650, BAR-651, BAR-652.

19.1.10 3.13.1 (2025-03-20)

Minor changes

- Improve behavior of the backup shortcuts `last-full` / `latest-full`

The shortcuts `last-full` / `latest-full` were retrieving not the last full backup of the server, but the last full backup of the server which was eligible as the parent for an incremental backup.

While this was the expected behavior, the feedback from the community has shown that it was confusing for the users.

From now on, the shortcuts `last-full` / `latest-full` will retrieve the last full backup of the Barman server, independently if that backup is eligible as the parent for an incremental backup or not.

The eligibility of the full backup as the parent of an incremental backup will still be validated by Barman in a later step, and a proper message will be displayed in case it doesn't suit as a parent.

References: BAR-555.

Bugfixes

- Fix error message when parsing invalid `--target-time` in `barman restore`

When using the `barman restore` command, the error message when parsing invalid `--target-time` string was:

EXCEPTION: local variable 'parsed_target' referenced before assignment

That exception was replaced with an understandable error message.

References: BAR-627.

- Fix mutual exclusive arguments in the cloud restore command

In the `barman-cloud-restore` command, we were checking that `target_tli` and `target_lsn` were mutually exclusive arguments, where the correct pair to check would be `target_time` and `target_lsn`.

References: BAR-624.

- Fix Barman not honoring `custom_decompression_filter`

Fixed an issue where Barman was not honoring the configured `custom_decompression_filter` if the compression algorithm specified was natively supported by Barman. Custom filters now take priority over native handlers when decompressing WAL files.

References: BAR-584.

- Fix `barman restore` with `--no-get-wal` and `--standby`

Fixed an issue where Barman was removing the `pg_wal` directory during recovery if `--no-get-wal` and `--standby-mode` were specified together. The issue happened due to Barman incorrectly filling the recovery parameters referencing `pg_wal`, including `recovery_end_command`, which led to this issue. Barman will now ignore filling such parameters as they are not required for this specific case.

References: BAR-630.

- Fix argument parsing issue in `barman restore` and `barman-cloud-restore`

In Barman 3.13.0, a regression was introduced causing errors when using `barman restore` and `barman-cloud-restore` commands. Specifically, the `backup_id` positional argument, which was made optional in that version, conflicted with other arguments, causing unrecognized arguments and errors.

For example, running `barman-cloud-restore` like this:

```
barman-cloud-restore source_url server_name backup_id --cloud-provider aws-s3_
↪recovery_dir
```

Would trigger an error like this:

```
barman-cloud-restore: error: unrecognized arguments: recovery_dir
```

This fix resolves the issue by making `backup_id` a required argument again. Additionally, a new "auto" value is now accepted as a `backup_id`, allowing Barman to automatically choose the best backup for restoration without needing a specific `backup_id`. This update fixes argument handling and still allows a smooth and flexible restoration process for the user.

References: BAR-596.

19.1.11 3.13.0 (2025-02-20)

Notable changes

- Add new `xlogdb_directory` configuration

Introduces a new `xlogdb_directory` configuration option. This parameter can be set either globally or per-server, and allows you to specify a custom directory for the `xlog.db` file. This file stores metadata of archived WAL files and is used internally by Barman in various scenarios. If unset, it defaults to the value of `wals_directory`. Additionally, the file was also renamed to contain the server name as a prefix.

References: BAR-483.

- Make "backup_id" optional when restoring a backup

Historically, Barman always required a "backup_id" to restore a backup, and would use that backup as the source for the restore.

This feature removes the need for specifying which backup to use as a source for restore, making it optional.

This change applies to both Barman and the `barman-cloud` scripts.

Now the user is able to restore a backup in the following ways:

1. Provide a "backup_id"
2. Do not provide a "backup_id". It will retrieve the most recent backup
3. Do not provide a "backup_id", but provide a recovery target, such as:
 - "target_time" (mutually exclusive with `target_lsn`) Will get the closest backup prior to the "target_time"
 - "target_lsn" (mutually exclusive with "target_time") Will get the closest backup prior to the "target_lsn"
 - "target_tli" (can be used combined with "target_time" or "target_lsn") Will get the most recent backup that matches

the timeline. If combined with other recovery targets, it will get the most recent backup prior to the `target_time` or `target_lsn` that matches the timeline

The recovery targets `--target-xid`, `--target-name` and `--target-immediate` are not supported, and will error out with a message if used.

This feature will provide flexibility and ease when restoring a postgres cluster.

References: BAR-541, BAR-473.

Minor changes

- Add current active model to `barman show-server` and `barman status`

Previously, after applying a configuration model, the only way to check which model is currently active for a server was via the `barman diagnose` command. With this update, the `barman status` and `barman show-server` commands now also display the current active configuration model for a server, if any.

References: BAR-524, BAR-400.

- Add `--staging-wal-directory` option to `barman restore` command to allow alternative WAL directory on PITR

A new command line option `--staging-wal-directory` was added to the `restore/recover` command to allow an alternative destination directory for WAL files when performing PITR. Previously, WAL files were copied to a `barman_wal` directory within the restore destination directory. This enhancement provides greater flexibility, such as storing WALs on separate partitions during recovery.

References: BAR-224.

- Pin boto3 version to any version $\leq 1.35.99$

Boto3 version 1.36 has changed the way S3 integrity is checked making this version incompatible with the current Barman code, generating the following error:

An error occurred (MissingContentLength) when calling the PutObject operation

As a temporary workaround, the version for boto3 is pinned to any version $\leq 1.35.99$ until support for 1.36 is implemented in Barman.

References: BAR-535.

- Make `barman-wal-archive` smarter when dealing with duplicate WAL files

Under some corner cases, Postgres could attempt to archive the same WAL twice. For example: if `barman-wal-archive` copies the WAL file over to the Barman host, but the script is interrupted before reporting success to Postgres. New executions of `barman-wal-archive` could fail when trying to archive the same file again because the WAL was already copied from Postgres to Barman, but not yet processed by the asynchronous Barman WAL archiver.

This minor change deals with this situation by verifying the checksum of the existing and the incoming file. If the checksums match the incoming file is ignored, otherwise an output info message is sent and the incoming file is moved to the errors directory. The code will exit with 0 in both situations, avoiding WALs piling up in the Postgres host due to a failing `archive_command`.

References: BAR-225.

- Document procedure to clear WAL archive failure check

While redesigning the Barman docs we missed adding a note advising users to run a `switch-wal` command if the server is idle and `barman check` returns a failure on "WAL archiving".

This addresses the gap left from the previous documentation.

References: BAR-521.

- Delete WALs by deleting the entire directory at once, when possible

Previously, when WAL files needed to be deleted (e.g., due to deletion of a backup), Barman would iterate over every WAL file and delete them individually. This could cause performance issues, mainly in systems which use ZFS filesystem. With this change, the entire directory will be deleted whenever noticed that all files in the directory are no longer needed by Barman.

References: BAR-511.

- Add support for `DefaultAzureCredential` option on Azure authentication

Users can now explicitly use Azure's `DefaultAzureCredential` for authentication by using the `default` option for `azure_credential` in the server configuration or the `--azure-credential default` option in the case of `barman-cloud-*`. Previously, that could only be set as a fallback when no credential was provided and no environment variables were set.

References: BAR-539.

- Improve diagnose output for retention policy info

Improves the output of the `barman diagnose` command to display a more user-friendly string representations. Specifically, "REDUNDANCY 2" is shown instead of "redundancy 2 b" for the 'retention_policy' attribute, and "MAIN" is shown instead of "simple-wal 2 b" for the 'wal_retention_policy' attribute.

References: BAR-100.

Bugfixes

- Fix PITR when using `barman restore` with `--target-tli`

Barman was not creating the `recovery.signal` nor filling `recovery_target_timeline` in `postgresql.auto.conf` in these cases:

- The only recovery target passed to `barman restore` was `--target-tli`; or
- `--target-tli` was specified with some other `--target-*` option, but the specified target timeline was the same as the timeline of the chosen backup.

Now, if any `--target-*` option is passed to `barman restore`, that will be correctly treated as PITR.

References: BAR-543.

- Fix bug when AWS 'profile' variable is referenced before assignment

An issue was introduced by BAR-242 as part of the Barman 3.12.0 release. The issue was causing `barman-cloud-backup-delete` (and possibly other commands) to fail with errors like this when `--aws-profile` argument or `aws_profile` configuration were not set:

```
ERROR: Barman cloud backup delete exception: local
variable 'profile' referenced before assignment`
```

References: BAR-518.

- Fix `--zstd` flag on `barman-cloud-wal-archive`

Fixed a bug with the `--zstd` flag on `barman-cloud-wal-archive` where it was essentially being ignored and not really compressing the WAL file before upload.

References: BAR-567.

19.1.12 3.12.1 (2024-12-09)

Bugfixes

- Add isoformat fields for backup start and end times in json output

This patch modifies the json output of the `infofile` object adding two new fields: `begin_time_iso` and `end_time_iso`. The new fields allow the use of a more standard and timezone aware time format, preserving compatibility with previous versions. It is worth noting that in the future the iso format for dates will be the standard used by barman for storing dates and will be used everywhere non human readable output is requested.

As part of the work, this patch reverts BAR-316, which was introduced on Barman 3.12.0.

References: BAR-494.

19.1.13 3.12.0 (2024-11-21)

Minor changes

- Add FIPS support to Barman

The `md5` hash algorithm is not FIPS compliant, so it is going to be replaced by `sha256`. `sha256` is FIPS compliant, vastly used, and is considered secure for most practical purposes. Up until this release, Barman's WAL archive client used `hashlib.md5` to generate checksums for tar files before they were sent to the Barman server. Here, a tar file is a file format used for bundling multiple files together with a MD5SUMS file that lists the checksums and their corresponding paths. In this release, the `md5` hashing algorithm is replaced by `sha256` as the default. As a result, checksums for the tar files will be calculated using `sha256`, and the MD5SUMS file will be named `SHA256SUMS`. Barman still has the ability to use the nondefault `md5` algorithm and the MD5SUMS file from the client if there is a use case for it. The user just needs to add the `--md5` flag to the `barman-wal-archive archive_command`.

References: BAR-155, CP-34954, CP-34391.

- Removed el7, debian10, and ubuntu1804 support; updated Debian and SLES.

Support for el7, debian10, and ubuntu1804 has been removed. Additionally, version 12 and version name "bookworm" has been added for Debian, addressing a previously missing entry. The SLES image version has also been updated from sp4 to sp5.

References: BAR-389.

- Add support for Postgres Extended 17 (PGE) and Postgres Advanced Server 17 (EPAS)

Tests were conducted on Postgres Extended 17 (PGE) and Postgres Advanced Server 17 (EPAS), confirming full compatibility with the latest features in Barman. This validation ensures that users of the latest version of PGE and EPAS can leverage all the new capabilities of Barman with confidence.

References: BAR-331.

- Improve WAL compression with zstd, lz4 and xz algorithms

Introduced support for xz compression on WAL files. It can be enabled by specifying `xz` in the `compression` server parameter. WALs will be compressed when entering the Barman's WAL archive. For the cloud, it can be enabled by specifying `--xz` when running `barman-cloud-wal-archive`.

Introduced support for zstandard compression on WAL files. It can be enabled by specifying `zstd` in the `compression` server parameter. WALs will be compressed when entering the Barman's WAL archive. For the cloud, it can be enabled by specifying `--zstd` when running `barman-cloud-wal-archive`.

Introduced support for lz4 compression on WAL files. It can be enabled by specifying `lz4` in the `compression` server parameter. WALs will be compressed when entering the Barman's WAL archive. For the cloud, it can be enabled by specifying `--lz4` when running `barman-cloud-wal-archive`.

References: BAR-265, BAR-423, BAR-264.

- Improve WAL upload performance on S3 buckets by avoiding multipart uploads

Previously, WAL files were being uploaded to S3 buckets using multipart uploads provided by the `boto3` library via the `upload_fileobj` method. It was noticed that multipart upload is slower when used for small files, such as WAL segments, compared to when uploading it in a single PUT request. This has been improved by avoiding multipart uploads for files smaller than 100MB. The average upload time of each WAL file is expected to be reduced by around 15% with this change.

References: BAR-374.

- Modify behavior when enforcing retention policy for `KEEP:STANDALONE` full backups

When enforcing the retention policy on full backups created with `backup_method = postgres`, Barman was previously marking all dependent (child) incremental backups as `VALID`, regardless of the `KEEP` target used. However, this approach is incorrect:

- For backups labeled `KEEP:STANDALONE`, Barman only retains the WAL files needed to restore the server to the exact state of that backup. Because these backups are self-contained, any dependent child backups are no longer needed once the root backup is outside the retention policy.
- In contrast, backups marked `KEEP:FULL` are intended for point-in-time recovery. To support this, Barman retains all WALs, as well as any child backups, to ensure the backup's consistency and allow recovery to the latest possible point.

This distinction ensures that `KEEP:STANDALONE` backups serve as snapshots of a specific moment, while `KEEP:FULL` backups retain everything needed for full point-in-time recovery.

References: BAR-366.

- Update documentation and user-facing features for Barman's recovery process.

Barman docs and the tool itself used to use the terms "recover"/"recovery" both for referencing:

- The Postgres recovery process;
- The process of restoring a backup and preparing it for recovery.

Both the code and documentation have been revised to accurately reflect the usage of the terms "restore" and "recover"/"recovery".

Also, the `barman recover` command was renamed to `barman restore`. The old name is still kept as an alias for backward compatibility.

References: BAR-337.

- Add `--keep-compression` flag to `barman-wal-restore` and `get-wal`

A new `--keep-compression` option has been added to both `barman-wal-restore` and `get-wal`. This option controls whether compressed WAL files should be decompressed on the Barman server before being fetched. When specified with `get-wal`, default decompression is skipped, and the output is the WAL file content in its original state. When specified with `barman-wal-restore`, the WAL file is fetched as-is and, if compressed, decompressed on the client side.

References: BAR-435.

- Ransomware protection - Add AWS Snapshot Lock Support

Barman now supports AWS EBS Snapshot Lock, a new integrated feature to prevent accidental or malicious deletions of Amazon EBS snapshots. When a snapshot is locked, it can't be deleted by any user but remains fully accessible for use. This feature enables you to store snapshots in WORM (Write-Once-Read-Many) format for a specified duration, helping to meet regulatory requirements by keeping the data secure and tamper-proof until the lock expires.

Special thanks to Rui Marinho, our community contributor who started this feature.

References: BAR-242.

- Prevent orphan files from being left from a crash while deleting a backup

This commit fixes an issue where backups could leave behind files if the system crashed during the deletion of a backup.

Now, when a backup is deleted, it will get a "delete marker" at the start. If a crash happens while the backup is being deleted, the marker will help recognize incomplete backup removals when the server restarts.

The Barman cron job has been updated to look for these deleted markers. If it finds a backup with a "delete marker", it will complete the process.

References: BAR-244.

- Add support for using tags with snapshots

Barman now supports tagging the snapshots when creating backups using the `barman-cloud-backup` script command. A new argument called `--tags` was added.

Special thanks to Rui Marinho, our community contributor who started this feature.

References: BAR-417.

- Use ISO format instead of ctime when producing JSON output of Barman cloud commands

The ctime format has no information about the time zone associated with the timestamp. Besides that, that format is better suited for human consumption. For machine consumption the ISO format is better suited.

References: BAR-316.

Bugfixes

- Fix barman check which returns wrong results for Replication Slot

Previously, when using architectures which backup from a standby node and stream WALs from the primary, Barman would incorrectly use `conninfo` (pointing to a standby server) for replication checks, leading to errors such as:

```
replication slot (WAL streaming): FAILED (replication slot 'barman' doesn't
exist. Please execute 'barman receive-wal --create-slot pg17')
```

This fixes the following issue [#1024](#) by ensuring `wal_conninfo` is used for WAL replication checks if it's set.

`wal_conninfo` takes precedence over `wal_streaming_conninfo`, when both are set. With this change, if only `wal_conninfo` is set, it will be used and will not fall back to `conninfo`.

Also, in the documentation, changes were made so it is explicit that when `conninfo` points to a standby server, `wal_conninfo` must be set and used for accurate replication status checks.

References: BAR-409.

- Fix missing options for `barman keep`

The error message that the Barman CLI emitted when running `barman keep` without any options suggested there were shortcut aliases for status and release. These aliases, `-s` and `-r`, do not exist, so the error message was misleading. This fixes the issue by including these short options in the Barman CLI, aligning it with other tools like `barman-cloud-backup-keep`, where these shortcuts already exist.

References: BAR-356.

- Lighten standby checks related to `conninfo` and `primary_conninfo`

When backing up a standby server, Barman performs some checks to assert that `conninfo` is really pointing to a standby (in recovery mode) and that `primary_conninfo` is pointing to a primary (not in recovery).

The problem, as reported in the issues [#704](#) and [#744](#), is that when a failover occurs, the `conninfo` will now be pointing to a primary instead and the checks will start failing, requiring the user to change Barman configs manually whenever a failover occurs.

This fix solved the issue by making such checks non-critical, which means they will still fail but Barman will keep operating regardless. Essentially, Barman will ignore `primary_conninfo` if `conninfo` does not point to a standby. Warnings about this misconfiguration will also be emitted whenever running any Barman command so the user can be aware.

References: BAR-348.

- Check for `USAGE` instead of `MEMBER` when calling `pg_has_role` in Barman

To work correctly Barman database user needs to be included in some roles. Barman was verifying the conditions was satisfied by calling `pg_has_role` in Postgres. However, it was check for the `MEMBER` privilege instead of `USAGE`. This oversight was fixed.

This change is a contribution from @RealGreenDragon.

References: BAR-489.

19.1.14 3.11.1 (2024-08-22)

Bugfixes

- Fix failures in `barman-cloud-backup-delete`. This command was failing when applying retention policies due to a bug introduced by the previous release.

19.1.15 3.11.0 (2024-08-22)

Notable changes

- Add support for Postgres 17+ incremental backups. This major feature is composed of several small changes:
 - Add `--incremental` command-line option to `barman backup` command. This is used to specify the parent backup when taking an incremental backup. The parent can be either a full backup or another incremental backup.
 - Add `latest-full` shortcut backup ID. Along with `latest`, this can be used as a shortcut to select the parent backup for an incremental backup. While `latest` takes the latest backup independently if it is full or incremental, `latest-full` takes the latest full backup.
 - `barman keep` command can only be applied to full backups when `backup_method = postgres`.
 - Retention policies do not take incremental backups into consideration. As incremental backups cannot be recovered without having the complete chain of backups available up to the full backup, only full backups account for retention policies. If a full backup has dependent incremental backups and the retention policy is applied, the full backup will propagate its status to the associated incremental backups. When the full backup is flagged with any `KEEP` target, Barman will set the status of all related incremental backups to `VALID`.
 - When deleting a backup all the incremental backups depending on it, if any, are also removed.
 - `barman recover` needs to combine the full backup with the chain of incremental backups when recovering. The new CLI option `--local-staging-path`, and the corresponding `local_staging_path` configuration option, are used to specify the path in the Barman host where the backups will be combined when recovering an incremental backup.

- Changes to `barman show-backup` output:
 - Add the “Estimated cluster size” field. It's useful to have an estimation of the data directory size of a cluster when restoring a backup. It's particularly useful when recovering compressed backups or incremental backups, situations where the size of the backup doesn't reflect the size of the data directory in Postgres. In JSON format, this is stored as `cluster_size`.
 - Add the “WAL summarizer” field. This field shows if `summarize_wal` was enabled in Postgres at the time the backup was taken. In JSON format, this is stored as `server_information.summarize_wal`. This field is omitted for Postgres 16 and older.
 - Add “Data checksums” field. This shows if `data_checksums` was enabled in Postgres at the time the backup was taken. In JSON format, this is stored as `server_information.data_checksums`.
 - Add the “Backup method” field. This shows the backup method used for this backup. In JSON format, this is stored as `base_backup_information.backup_method`.
 - Rename the field “Disk Usage” as “Backup Size”. The latter provides a more comprehensive name which represents the size of the backup in the Barman host. The JSON field under `base_backup_information` was also renamed from `disk_usage` to `backup_size`.
 - Add the “WAL size” field. This shows the size of the WALs required by the backup. In JSON format, this is stored as `base_backup_information.wal_size`.
 - Refactor the field “Incremental size”. It is now named “Resources saving” and it now shows an estimation of resources saved when taking incremental backups with `rsync` or `pg_basebackup`. It compares the backup size with the estimated cluster size to estimate the amount of disk and network resources that were saved by taking an incremental backup. In JSON format, the field was renamed from `incremental_size` to `resource_savings` under `base_backup_information`.
 - Add the `system_id` field to the JSON document. This field contains the system identifier of Postgres. It was present in console format, but was missing in JSON format.
 - Add fields related with Postgres incremental backups:
 - * “Backup type” : indicates if the Postgres backup is full or incremental. In JSON format, this is stored as `backup_type` under `base_backup_information`.
 - * “Root backup” : the ID of the full backup that is the root of a chain of one or more incremental backups. In JSON format, this is stored as `catalog_information.root_backup_id`.
 - * “Parent backup” : the ID of the full or incremental backup from which this incremental backup was taken. In JSON format, this is stored as `catalog_information.parent_backup_id`.
 - * “Children Backup(s)” : the IDs of the incremental backups that were taken with this backup as the parent. In JSON format, this is stored as `catalog_information.children_backup_ids`.
 - * “Backup chain size” : the number of backups in the chain from this incremental backup up to the root backup. In JSON format, this is stored as `catalog_information.chain_size`.
- Changes to `barman list-backup` output:
 - It now includes the backup type in the JSON output, which can be either `rsync` for backups taken with `rsync`, `full` or `incremental` for backups taken with `pg_basebackup`, or `snapshot` for cloud

snapshots. When printing to the console the backup type is represented by the corresponding labels R, F, I or S.

- Remove tablespaces information from the output. That was bloating the output. Tablespaces information can still be found in the output of `barman show-backup`.
- Always set a timestamp with a time zone when configuring `recovery_target_time` through `barman recover`. Previously, if no time zone was explicitly set through `--target-time`, Barman would configure `recovery_target_time` without a time zone in Postgres. Without a time zone, Postgres would assume whatever is configured through `timezone` GUC in Postgres. From now on Barman will issue a warning and configure `recovery_target_time` with the time zone of the Barman host if no time zone is set by the user through `--target-time` option.
- When recovering a backup with the “no get wal” approach and `--target-lsn` is set, copy only the WAL files required to reach the configured target. Previously Barman would copy all the WAL files from its archive to Postgres.
- When recovering a backup with the “no get wal” approach and `--target-immediate` is set, copy only the WAL files required to reach the consistent point. Previously Barman would copy all the WAL files from its archive to Postgres.
- `barman-wal-restore` now moves WALs from the spool directory to `pg_wal` instead of copying them. This can improve performance if the spool directory and the `pg_wal` directory are in the same partition.
- `barman check-backup` now shows the reason why a backup was marked as **FAILED** in the output and logs. Previously for a user to know why the backup was marked as **FAILED**, they would need to run `barman show-backup` command.
- Add configuration option `aws_await_snapshots_timeout` and the corresponding `--aws-await-snapshots-timeout` command-line option on `barman-cloud-backup`. This specifies the timeout in seconds to wait for snapshot backups to reach the completed state.
- Add a keep-alive mechanism to rsync-based backups. Previously the Postgres session created by Barman to run `pg_backup_start()` and `pg_backup_stop()` would stay idle for as long as the base backup copy would take. That could lead to a firewall or router dropping the connection because it was idle for a long time. The keep-alive mechanism sends heartbeat queries to Postgres through that connection, thus reducing the likelihood of a connection getting dropped. The interval between heartbeats can be controlled through the new configuration option `keepalive_interval` and the corresponding CLI option `--keepalive-interval` of the `barman backup` command.

Bugfixes

- When recovering a backup with the “no get wal” approach and `--target-time` set, copy all WAL files. Previously Barman would attempt to “guess” the WAL files required by Postgres to reach the configured target time. However, the mechanism was not robust enough as it was based on the stats of the WAL file in the Barman host (more specifically the creation time). For example: if there were archiving or streaming lag between Postgres and Barman, that could be enough for recovery to fail because Barman would miss to copy all the required WAL files due to the weak check based on file stats.
- Pin `python-snappy` to 0.6.1 when running Barman through Python 3.6 or older. Newer versions of `python-snappy` require `cramjam` version 2.7.0 or newer, and these are only available for Python 3.7

or newer.

- `barman receive-wal` now exits with code 1 instead of 0 in the following cases:
 - Being unable to run with `--reset` flag because `pg_receivewal` is running.
 - Being unable to start `pg_receivewal` process because it is already running.
- Fix and improve information about Python in `barman diagnose` output:
 - The command now makes sure to use the same Python interpreter under which Barman is installed when outputting the Python version through `python_ver` JSON key. Previously, if an environment had multiple Python installations and/or virtual environments, the output could eventually be misleading, as it could be fetched from a different Python interpreter.
 - Added a `python_executable` key to the JSON output. That contains the path to the exact Python interpreter being used by Barman.

19.1.16 3.10.1 (2024-06-12)

Bugfixes

- Make `argcomplete` optional to avoid installation issues on some platforms.
- Load `barman.auto.conf` only when the file exists.
- Emit a warning when the `cfg_changes.queue` file is malformed.
- Correct in documentation the postgresql version where `pg_checkpoint` is available.
- Add `--no-partial` option to `barman-cloud-wal-restore`.

19.1.17 3.10.0 (2024-01-24)

Notable changes

- Limit the average bandwidth used by `barman-cloud-backup` when backing up to either AWS S3 or Azure Blob Storage according to the value set by a new CLI option `--max-bandwidth`.
- Add the new configuration option `lock_directory_cleanup` That enables cron to automatically clean up the `barman_lock_directory` from unused lock files.
- Add support for a new type of configuration called `model`. The model acts as a set of overrides for configuration options for a given Barman server.
- Add a new `barman` command `barman config-update` that allows the creation and the update of configurations using JSON

Bugfixes

- Fix a bug that caused `--min-chunk-size` to be ignored when using `barman-cloud-backup` as hook script in Barman.

19.1.18 3.9.0 (2023-10-03)

Notable changes

- Allow `barman switch-wal --force` to be run against PG $\geq$ 14 if the user has the `pg_checkpoint` role (thanks to toydarian for this patch).
- Log the current check at `info` level when a check timeout occurs.
- The minimum size of an upload chunk when using `barman-cloud-backup` with either S3 or Azure Blob Storage can now be specified using the `--min-chunk-size` option.
- `backup_compression = none` is supported when using `pg_basebackup`.
- For PostgreSQL 15 and later: the allowed `backup_compression_level` values for `zstd` and `lz4` have been updated to match those allowed by `pg_basebackup`.
- For PostgreSQL versions earlier than 15: `backup_compression_level = 0` can now be used with `backup_compression = gzip`.

Bugfixes

- Fix `barman recover` on platforms where Multiprocessing uses `spawn` by default when starting new processes.

19.1.19 3.8.0 (2023-08-31)

Notable changes

- Clarify package installation. `barman` is packaged with default python version for each operating system.
- The `minimum-redundancy` option is added to `barman-cloud-backup-delete`. It allows to set the minimum number of backups that should always be available.
- Add a new `primary_checkpoint_timeout` configuration option. Allows define the amount of seconds that Barman will wait at the end of a backup if no new WAL files are produced, before forcing a checkpoint on the primary server.

Bugfixes

- Fix race condition in `barman` retention policies application. Backup deletions will now raise a warning if another deletion is in progress for the requested backup.
- Fix `barman-cloud-backup-show man` page installation.

19.1.20 3.7.0 (2023-07-25)

Notable changes

- Support is added for snapshot backups on AWS using EBS volumes.
- The `--profile` option in the `barman-cloud-*` scripts is renamed `--aws-profile`. The old name is deprecated and will be removed in a future release.

- Backup manifests can now be generated automatically on completion of a backup made with `backup_method = rsync`. This is enabled by setting the `autogenerate_manifest` configuration variable and can be overridden using the `--manifest` and `--no-manifest` CLI options.

Bugfixes

- The `barman-cloud-*` scripts now correctly use continuation tokens to page through objects in AWS S3-compatible object stores. This fixes a bug where `barman-cloud-backup-delete` would only delete the oldest 1000 eligible WALs after backup deletion.
- Minor documentation fixes.

19.1.21 3.6.0 (2023-06-15)

Notable changes

- PostgreSQL version 10 is no longer supported.
- Support is added for snapshot backups on Microsoft Azure using Managed Disks.
- The `--snapshot-recovery-zone` option is renamed `--gcp-zone` for consistency with other provider-specific options. The old name is deprecated and will be removed in a future release.
- The `snapshot_zone` option and `--snapshot-zone` argument are renamed `gcp_zone` and `--gcp-zone` respectively. The old names are deprecated and will be removed in a future release.
- The `snapshot_gcp_project` option and `--snapshot-gcp-project` argument are renamed to `gcp_project` and `--gcp-project`. The old names are deprecated and will be removed in a future release.

Bugfixes

- Barman will no longer attempt to execute the `replication-status` command for a passive node.
- The `backup_label` is deleted from cloud storage when a snapshot backup is deleted with `barman-cloud-backup-delete`.
- Man pages for the `generate-manifest` and `verify-backup` commands are added.
- Minor documentation fixes.

19.1.22 3.5.0 (2023-03-29)

Notable changes

- Python 2.7 is no longer supported. The earliest Python version supported is now 3.6.
- The `barman`, `barman-cli` and `barman-cli-cloud` packages for EL7 now require python 3.6 instead of python 2.7. For other supported platforms, Barman packages already require python versions 3.6 or later so packaging is unaffected.

- Support for PostgreSQL 10 will be discontinued in future Barman releases; 3.5.x is the last version of Barman with support for PostgreSQL 10.
- Backups and WALs uploaded to Google Cloud Storage can now be encrypted using a specific KMS key by using the `--kms-key-name` option with `barman-cloud-backup` or `barman-cloud-wal-archive`.
- Backups and WALs uploaded to AWS S3 can now be encrypted using a specific KMS key by using the `--sse-kms-key-id` option with `barman-cloud-backup` or `barman-cloud-wal-archive` along with `--encryption=aws:kms`.
- Two new configuration options are provided which make it possible to limit the rate at which parallel workers are started during backups with `backup_method = rsync` and recoveries. `parallel_jobs_start_batch_size` can be set to limit the amount of parallel workers which will be started in a single batch, and `parallel_jobs_start_batch_period` can be set to define the time in seconds over which a single batch of workers will be started. These can be overridden using the arguments `--jobs-start-batch-size` and `--jobs-start-batch-period` with the `barman backup` and `barman recover` commands.
- A new option `--recovery-conf-filename` is added to `barman recover`. This can be used to change the file to which Barman should write the PostgreSQL recovery options from the default `postgresql.auto.conf` to an alternative location.

Bugfixes

- Fix a bug which prevented `barman-cloud-backup-show` from displaying the backup metadata for backups made with `barman backup` and uploaded by `barman-cloud-backup` as a post-backup hook script.
- Fix a bug where the PostgreSQL connection used to validate backup compression settings was left open until termination of the Barman command.
- Fix an issue which caused `rsync-concurrent` backups to fail when running for a duration greater than `idle_session_timeout`.
- Fix a bug where the backup name was not saved in the backup metadata if the `--wait` flag was used with `barman backup`.
- Thanks to `mojtabash78`, `mhkarimi1383`, `epolkerman`, `barthisrael` and `hzetters` for their contributions.

19.1.23 3.4.0 (2023-01-26)

Notable changes

- This is the last release of Barman which will support Python 2 and new features will henceforth require Python 3.6 or later.
- A new `backup_method` named `snapshot` is added. This will create backups by taking snapshots of cloud storage volumes. Currently only Google Cloud Platform is supported however support for AWS and Azure will follow in future Barman releases. Note that this feature requires a minimum Python version of 3.7. Please see the Barman manual for more information.
- Support for snapshot backups is also added to `barman-cloud-backup`, with minimal support for restoring a snapshot backup added to `barman-cloud-restore`.

- A new command `barman-cloud-backup-show` is added which displays backup metadata stored in cloud object storage and is analogous to `barman show-backup`. This is provided so that snapshot metadata can be easily retrieved at restore time however it is also a convenient way of inspecting metadata for any backup made with `barman-cloud-backup`.
- The instructions for installing Barman from RPMs in the docs are updated.
- The formatting of NFS requirements in the docs is fixed.
- Supported PostgreSQL versions are updated in the docs (this is a documentation fix only - the minimum supported major version is still 10).

19.1.24 3.3.0 (2022-12-14)

Notable changes

- A backup can now be given a name at backup time using the new `--name` option supported by the `barman backup` and `barman-cloud-backup` commands. The backup name can then be used in place of the backup ID when running commands to interact with backups. Additionally, the commands to list and show backups have been updated to include the backup name in the plain text and JSON output formats.
- Stricter checking of PostgreSQL version to verify that Barman is running against a supported version of PostgreSQL.

Bugfixes

- Fix inconsistencies between the `barman cloud` command docs and the help output for those commands.
- Use a new PostgreSQL connection when switching WALs on the primary during the backup of a standby to avoid undefined behaviour such as `SSL error` messages and failed connections.
- Reduce log volume by changing the default log level of stdout for commands executed in child processes to `DEBUG` (with the exception of `pg_basebackup` which is deliberately logged at `INFO` level due to it being a long-running process where it is frequently useful to see the output during the execution of the command).

19.1.25 3.2.0 (2022-10-20)

Notable changes

- `barman-cloud-backup-delete` now accepts a `--batch-size` option which determines the maximum number of objects deleted in a single request.
- All `barman-cloud-*` commands now accept a `--read-timeout` option which, when used with the `aws-s3` cloud provider, determines the read timeout used by the `boto3` library when making requests to S3.

Bugfixes

- Fix the failure of `barman recover` in cases where `backup_compression` is set in the Barman configuration but the PostgreSQL server is unavailable.

19.1.26 3.1.0 (2022-09-14)

Notable changes

- Backups taken with `backup_method = postgres` can now be compressed using lz4 and zstd compression by setting `backup_compression = lz4` or `backup_compression = zstd` respectively. These options are only supported with PostgreSQL 15 (beta) or later.
- A new option `backup_compression_workers` is available which sets the number of threads used for parallel compression. This is currently only available with `backup_method = postgres` and `backup_compression = zstd`.
- A new option `primary_conninfo` can be set to avoid the need for backups of standbys to wait for a WAL switch to occur on the primary when finalizing the backup. Barman will use the connection string in `primary_conninfo` to perform WAL switches on the primary when stopping the backup.
- Support for certain Rsync versions patched for CVE-2022-29154 which require a trailing newline in the `--files-from` argument.
- Allow `barman receive-wal` maintenance options (`--stop`, `--reset`, `--drop-slot` and `--create-slot`) to run against inactive servers.
- Add `--port` option to `barman-wal-archive` and `barman-wal-restore` commands so that a custom SSH port can be used without requiring any SSH configuration.
- Various documentation improvements.
- Python 3.5 is no longer supported.

Bugfixes

- Ensure PostgreSQL connections are closed cleanly during the execution of `barman cron`.
- `barman generate-manifest` now treats pre-existing `backup_manifest` files as an error condition.
- `backup_manifest` files are renamed by appending the backup ID during recovery operations to prevent future backups including an old `backup_manifest` file.
- Fix epoch timestamps in json output which were not timezone-aware.
- The output of `pg_basebackup` is now written to the Barman log file while the backup is in progress.
- We thank barthisrael, elhananjair, kraynopp, lucianobotti, and mxy for their contributions to this release.

19.1.27 3.0.1 (2022-06-27)

Bugfixes

- Fix package signing issue in PyPI (same sources as 3.0.0)

19.1.28 3.0.0 (2022-06-23)

Breaking changes

- PostgreSQL versions 9.6 and earlier are no longer supported. If you are using one of these versions you will need to use an earlier version of Barman.
- The default backup mode for Rsync backups is now concurrent rather than exclusive. Exclusive backups have been deprecated since PostgreSQL 9.6 and have been removed in PostgreSQL 15. If you are running Barman against PostgreSQL versions earlier than 15 and want to use exclusive backups you will now need to set `exclusive_backup` in `backup_options`.
- The backup metadata stored in the `backup.info` file for each backup has an extra field. This means that earlier versions of Barman will not work in the presence of any backups taken with 3.0.0. Additionally, users of `pg-backup-api` will need to upgrade it to version 0.2.0 so that `pg-backup-api` can work with the updated metadata.

Notable changes

- Backups taken with `backup_method = postgres` can now be compressed by `pg_basebackup` by setting the `backup_compression` config option. Additional options are provided to control the compression level, the backup format and whether the `pg_basebackup` client or the PostgreSQL server applies the compression. NOTE: Recovery of these backups requires Barman to stage the compressed files on the recovery server in a location specified by the `recovery_staging_path` option.
- Add support for PostgreSQL 15. Exclusive backups are not supported by PostgreSQL 15 therefore Barman configurations for PostgreSQL 15 servers are not allowed to specify `exclusive_backup` in `backup_options`.
- Use `custom_compression_magic`, if set, when identifying compressed WAL files. This allows Barman to correctly identify uncompressed WALs (such as `*.partial` files in the `streaming` directory) and return them instead of attempting to decompress them.

Minor changes

- Various documentation improvements.

Bugfixes

- Fix an ordering bug which caused Barman to log the message "Backup failed issuing start backup command." while handling a failure in the stop backup command.
- Fix a bug which prevented recovery using `--target-tli` when timelines greater than 9 were present, due to hexadecimal values from WAL segment names being parsed as base 10 integers.
- Fix an import error which occurs when using barman cloud with certain python2 installations due to issues with the `enum34` dependency.
- Fix a bug where Barman would not read more than three bytes from a compressed WAL when attempting to identify the magic bytes. This means that any custom compressed WALs using magic longer than three bytes are now decompressed correctly.

- Fix a bug which caused the `--immediate-checkpoint` flag to be ignored during backups with `backup_method = rsync`.

19.1.29 2.19 (2022-03-09)

Notable changes

- Change `barman diagnose` output date format to ISO8601.
- Add Google Cloud Storage (GCS) support to `barman cloud`.
- Support `current` and `latest` recovery targets for the `--target-tli` option of `barman recover`.
- Add documentation for installation on SLES.

Bugfixes

- `barman-wal-archive --test` now returns a non-zero exit code when an error occurs.
- Fix `barman-cloud-check-wal-archive` behaviour when `-t` option is used so that it exits after connectivity test.
- `barman recover` now continues when `--no-get-wal` is used and `"get-wal"` is not set in `recovery_options`.
- Fix `barman show-servers --format=json ${server}` output for inactive server.
- Check for presence of `barman_home` in configuration file.
- Passive `barman` servers will no longer store two copies of the tablespace data when syncing backups taken with `backup_method = postgres`.
- We thank richyen for his contributions to this release.

19.1.30 2.18 (2022-01-21)

Notable changes

- Add snappy compression algorithm support in `barman cloud` (requires the optional `python-snappy` dependency).
- Allow Azure client concurrency parameters to be set when uploading WALs with `barman-cloud-wal-archive`.
- Add `--tags` option in `barman cloud` so that backup files and archived WALs can be tagged in cloud storage (aws and azure).
- Update the `barman cloud` exit status codes so that there is a dedicated code (2) for connectivity errors.
- Add the commands `barman verify-backup` and `barman generate-manifest` to check if a backup is valid.
- Add support for Azure Managed Identity auth in `barman cloud` which can be enabled with the `--credential` option.

Bugfixes

- Change `barman-cloud-check-wal-archive` behavior when bucket does not exist.
- Ensure `list-files` output is always sorted regardless of the underlying filesystem.
- Man pages for `barman-cloud-backup-keep`, `barman-cloud-backup-delete` and `barman-cloud-check-wal-archive` added to Python packaging.
- We thank richyen and stratakis for their contributions to this release.

19.1.31 2.17 (2021-12-01)

Notable changes

- Resolves a performance regression introduced in version 2.14 which increased copy times for `barman backup` or `barman recover` commands when using the `--jobs` flag.
- Ignore `rsync` partial transfer errors for `sender` processes so that such errors do not cause the backup to fail (thanks to barthisrael).

19.1.32 2.16 (2021-11-17)

Notable changes

- Add the commands `barman-check-wal-archive` and `barman-cloud-check-wal-archive` to validate if a proposed archive location is safe to use for a new PostgreSQL server.
- Allow Barman to identify WAL that's already compressed using a custom compression scheme to avoid compressing it again.
- Add `last_backup_minimum_size` and `last_wal_maximum_age` options to `barman check`.

Bugfixes

- Use `argparse` for command line parsing instead of the unmaintained `argh` module.
- Make timezones consistent for `begin_time` and `end_time`.
- We thank chtitux, George Hansper, stratakis, Thoro, and vrms for their contributions to this release.

19.1.33 2.15 (2021-10-12)

Notable changes

- Add plural forms for the `list-backup`, `list-server` and `show-server` commands which are now `list-backups`, `list-servers` and `show-servers`. The singular forms are retained for backward compatibility.
- Add the `last-failed` backup shortcut which references the newest failed backup in the catalog so that you can do:
 - `barman delete <SERVER> last-failed`

Bugfixes

- Tablespaces will no longer be omitted from backups of EPAS versions 9.6 and 10 due to an issue detecting the correct version string on older versions of EPAS.

19.1.34 2.14 (2021-09-22)

Notable changes

- Add the `barman-cloud-backup-delete` command which allows backups in cloud storage to be deleted by specifying either a backup ID or a retention policy.
- Allow backups to be retained beyond any retention policies in force by introducing the ability to tag existing backups as archival backups using `barman keep` and `barman-cloud-backup-keep`.
- Allow the use of SAS authentication tokens created at the restricted blob container level (instead of the wider storage account level) for Azure blob storage
- Significantly speed up `barman restore` into an empty directory for backups that contain hundreds of thousands of files.

Bugfixes

- The backup privileges check will no longer fail if the user lacks "userepl" permissions and will return better error messages if any required permissions are missing (#318 and #319).

19.1.35 2.13 (2021-07-26)

Notable changes

- Add Azure blob storage support to `barman-cloud`
- Support tablespace remapping in `barman-cloud-restore` via `--tablespace name:location`
- Allow `barman-cloud-backup` and `barman-cloud-wal-archive` to run as Barman hook scripts, to allow data to be relayed to cloud storage from the Barman server

Bugfixes

- Stop backups failing due to `idle_in_transaction_session_timeout` <https://github.com/EnterpriseDB/barman/issues/333>
- Fix a race condition between backup and archive-wal in updating `xlog.db` entries (#328)
- Handle PGDATA being a symlink in `barman-cloud-backup`, which led to "seeking backwards is not allowed" errors on restore (#351)
- Recreate `pg_wal` on restore if the original was a symlink (#327)
- Recreate `pg_tblspc` symlinks for tablespaces on restore (#343)
- Make `barman-cloud-backup-list` skip backups it cannot read, e.g., because they are in Glacier storage (#332)

- Add `-d database` option to `barman-cloud-backup` to specify which database to connect to initially (#307)
- Fix "Backup failed uploading data" errors from `barman-cloud-backup` on Python 3.8 and above, caused by attempting to pickle the boto3 client (#361)
- Correctly enable server-side encryption in S3 for buckets that do not have encryption enabled by default.

In Barman 2.12, `barman-cloud-backup`'s `--encryption` option did not correctly enable encryption for the contents of the backup if the backup was stored in an S3 bucket that did not have encryption enabled. If this is the case for you, please consider deleting your old backups and taking new backups with Barman 2.13.

If your S3 buckets already have encryption enabled by default (which we recommend), this does not affect you.

19.1.36 2.12.1 (2021-06-30)

Bugfixes

- Allow specifying `target-tli` with other `target-*` recovery options.
- Fix incorrect NAME in `barman-cloud-backup-list` manpage.
- Don't raise an error if SIGALRM is ignored.
- Fetch `wal_keep_size`, not `wal_keep_segments`, from Postgres 13.

19.1.37 2.12 (2020-11-05)

Notable changes

- Introduce a new `backup_method` option called `local-rsync` which targets those cases where Barman is installed on the same server where PostgreSQL is and directly uses `rsync` to take base backups, bypassing the SSH layer.

Bugfixes

- Avoid corrupting boto connection in worker processes.
- Avoid connection attempts to PostgreSQL during tests.

19.1.38 2.11 (2020-07-09)

Notable changes

- Introduction of the `barman-cli-cloud` package that contains all cloud related utilities.
- Add `barman-cloud-wal-restore` to restore a WAL file previously archived with `barman-cloud-wal-archive` from an object store.
- Add `barman-cloud-restore` to restore a backup previously taken with `barman-cloud-backup` from an object store.
- Add `barman-cloud-backup-list` to list backups taken with `barman-cloud-backup` in an object store.

- Add support for arbitrary archive size for barman-cloud-backup.
- Add support for --endpoint-url option to cloud utilities.
- Remove strict superuser requirement for PG 10+ (by Kaarel Moppel).
- Add --log-level runtime option for barman to override default log level for a specific command.
- Support for PostgreSQL 13

Bugfixes

- Suppress messages and warning with SSH connections in barman-cli (GH-257).
- Fix a race condition when retrieving uploaded parts in barman-cloud-backup (GH-259).
- Close the PostgreSQL connection after a backup (GH-258).
- Check for uninitialized replication slots in receive-wal --reset (GH-260).
- Ensure that begin_wal is valorised before acting on it (GH-262).
- Fix bug in XLOG/WAL arithmetic with custom segment size (GH-287).
- Fix rsync compatibility error with recent rsync.
- Fix PostgreSQLClient version parsing.
- Fix PostgreSQL exception handling with non ASCII messages.
- Ensure each postgres connection has an empty search_path.
- Avoid connecting to PostgreSQL while reading a backup.info file.

If you are using already `barman-cloud-wal-archive` or `barman-cloud-backup` installed via RPM/Apt package and you are upgrading your system, you must install the `barman-cli-cloud` package. All cloud related tools are now part of the `barman-cli-cloud` package, including `barman-cloud-wal-archive` and `barman-cloud-backup` that were previously shipped with `barman-cli`. The reason is complex dependency management of the boto3 library, which is a requirement for the cloud utilities.

19.1.39 2.10 (2019-12-05)

Notable changes

- Pull .partial WAL files with `get-wal` and `barman-wal-restore`, allowing `restore_command` in a recovery scenario to fetch a partial WAL file's content from the Barman server. This feature simplifies and enhances RPO=0 recovery operations.
- Store the PostgreSQL system identifier in the server directory and inside the backup information file. Improve `check` command to verify the consistency of the system identifier with active connections (standard and replication) and data on disk.
- A new script called `barman-cloud-wal-archive` has been added to the `barman-cli` package to directly ship WAL files from PostgreSQL (using `archive_command`) to cloud object storage services that are compatible with AWS S3. It supports encryption and compression.

- A new script called `barman-cloud-backup` has been added to the `barman-cli` package to directly ship base backups from a local PostgreSQL server to cloud object storage services that are compatible with AWS S3. It supports encryption, parallel upload, compression.
- Automated creation of replication slots through the server/global option `create_slot`. When set to `auto`, Barman creates the replication slot, in case `streaming_archiver` is enabled and `slot_name` is defined. The default value is `manual` for back-compatibility.
- Add `'-w/--wait'` option to backup command, making Barman wait for all required WAL files to be archived before considering the backup completed. Add also the `--wait-timeout` option (default 0, no timeout).
- Redact passwords from Barman output, in particular from `barman diagnose` (InfoSec)
- Improve robustness of `receive-wal --reset` command, by verifying that the last partial file is aligned with the current location or, if present, with replication slot's.
- Documentation improvements

Bugfixes

- Wrong string matching operation when excluding tablespaces inside PGDATA (GH-245).
- Minor fixes in WAL delete hook scripts (GH-240).
- Fix PostgreSQL connection aliveness check (GH-239).

19.1.40 2.9 (2019-08-01)

Notable changes

- Transparently support PostgreSQL 12, by supporting the new way of managing recovery and standby settings through GUC options and signal files (`recovery.signal` and `standby.signal`)
- Add `--bwlimit` command line option to set bandwidth limitation for backup and recover commands
- Ignore WAL archive failure for check command in case the latest backup is `WAITING_FOR_WALS`
- Add `--target-lsn` option to set recovery target Log Sequence Number for recover command with PostgreSQL 10 or higher
- Add `--spool-dir` option to `barman-wal-restore` so that users can change the spool directory location from the default, avoiding conflicts in case of multiple PostgreSQL instances on the same server (thanks to Drazen Kacar).
- Rename `barman_xlog` directory to `barman_wal`
- JSON output writer to export command output as JSON objects and facilitate integration with external tools and systems (thanks to Marcin Onufry Hlybin). Experimental in this release.

Bugfixes

- replication-status doesn't show streamers with no slot (GH-222)
- When checking that a connection is alive ("SELECT 1" query), preserve the status of the PostgreSQL connection (GH-149). This fixes those cases of connections that were terminated due to idle-in-transaction timeout, causing concurrent backups to fail.

19.1.41 2.8 (2019-05-17)

Notable changes

- Add support for reuse_backup in geo-redundancy for incremental backup copy in passive nodes
- Improve performance of rsync based copy by using strptime instead of the more generic dateutil.parser (#210)
- Add '--test' option to barman-wal-archive and barman-wal-restore to verify the connection with the Barman server
- Complain if backup_options is not explicitly set, as the future default value will change from exclusive_backup to concurrent_backup when PostgreSQL 9.5 will be declared EOL by the PGDG
- Display additional settings in the show-server and diagnose commands: archive_timeout, data_checksums, hot_standby, max_wal_senders, max_replication_slots and wal_compression.
- Merge the barman-cli project in Barman

Minor changes

- Improve messaging of check --nagios for inactive servers.
- Log remote SSH command with recover command.
- Hide logical decoding connections in replication-status command.

This release officially supports Python 3 and deprecates Python 2 (which might be discontinued in future releases).

PostgreSQL 9.3 and older is deprecated from this release of Barman. Support for backup from standby is now limited to PostgreSQL 9.4 or higher and to WAL shipping from the standby (please refer to the documentation for details).

Bugfixes

- Fix encoding error in get-wal on Python 3 (Jeff Janes, #221).
- Fix exclude_and_protect_filter (Jeff Janes, #217).
- Remove spurious message when resetting WAL (Jeff Janes, #215).
- Fix sync-wals error if primary has WALs older than the first backup.
- Support for double quotes in synchronous_standby_names setting.

19.1.42 2.7 (2019-03-12)

Notable changes

- Fix error handling during the parallel backup. Previously an unrecoverable error during the copy could have corrupted the barman internal state, requiring a manual kill of barman process with SIGTERM and a manual cleanup of the running backup in PostgreSQL. (GH#199).
- Fix support of UTF-8 characters in input and output (GH#194 and GH#196).
- Ignore history/backup/partial files for first sync of geo-redundancy (GH#198).
- Fix network failure with geo-redundancy causing cron to break (GH#202).
- Fix backup validation in PostgreSQL older than 9.2.
- Various documentation fixes.

19.1.43 2.6 (2019-02-04)

Notable changes

- Add support for Geographical redundancy, introducing 3 new commands: sync-info, sync-backup and sync-wals. Geo-redundancy allows a Barman server to use another Barman server as data source instead of a PostgreSQL server.
- Add put-wal command that allows Barman to safely receive WAL files via PostgreSQL's archive_command using the barman-wal-archive script included in barman-cli.
- Add ANSI colour support to check command.

Bugfixes

- Fix switch-wal on standby with an empty WAL directory.
- Honour archiver locking in wait_for_wal method.
- Fix WAL compression detection algorithm.
- Fix current_action in concurrent stop backup errors.
- Do not treat lock file busy as an error when validating a backup.

19.1.44 2.5 (2018-10-23)

Notable changes

- Add support for PostgreSQL 11
- Add check-backup command to verify that WAL files required for consistency of a base backup are present in the archive. Barman now adds a new state (WAITING_FOR_WALS) after completing a base backup, and sets it to DONE once it has verified that all WAL files from start to the end of the backup exist. This command is included in the regular cron maintenance job. Barman now notifies users attempting to recover a backup that is in WAITING_FOR_WALS state.

- Allow switch-xlog --archive to work on a standby (just for the archive part)

Bugfixes

- Fix decoding errors reading external commands output (issue #174).
- Fix documentation regarding WAL streaming and backup from standby.

19.1.45 2.4 (2018-05-25)

Notable changes

- Add standard and retry hook scripts for backup deletion (pre/post).
- Add standard and retry hook scripts for recovery (pre/post).
- Add standard and retry hook scripts for WAL deletion (pre/post).
- Add --standby-mode option to barman recover to add standby_mode = on in pre-generated recovery.conf.
- Add --target-action option to barman recover, allowing users to add shutdown, pause or promote to the pre-generated recovery.conf file.
- Improve usability of point-in-time recovery with consistency checks (e.g. recovery time is after end time of backup).
- Minor documentation improvements.
- Drop support for Python 3.3.

Bugfixes

- Fix remote get_file_content method (GitHub #151), preventing incremental recovery from happening.
- Unicode issues with command (GitHub #143 and #150).
- Add --wal-method=none when pg_basebackup >= 10 (GitHub #133).
- Stop process manager module from overwriting lock files content
- Relax the rules for rsync output parsing
- Ignore vanished files in streaming directory
- Case insensitive slot names (GitHub #170)
- Make DataTransferFailure.from_command_error() more resilient (GitHub #86)
- Rename command() to barman_command() (GitHub #118)
- Initialise synchronous standby names list if not set (GitHub #111)
- Correct placeholders ordering (GitHub #138)
- Force datestyle to iso for replication connections
- Returns error if delete command does not remove the backup

- Fix exception when calling `is_power_of_two(None)`
- Downgraded sync standby names messages to debug (GitHub #89)

19.1.46 2.3 (2017-09-05)

Notable changes

- Add support to PostgreSQL 10
- Follow naming changes in PostgreSQL 10:
 - The `switch-xlog` command has been renamed to `switch-wal`.
 - In commands output, the `xlog` word has been changed to `WAL` and `location` has been changed to `LSN` when appropriate.
- Add the `--network-compression/--no-network-compression` options to `barman recover` to enable or disable network compression at run-time
- Add `--target-immediate` option to `recover` command, in order to exit recovery when a consistent state is reached (end of the backup, available from PostgreSQL 9.4)
- Show cluster state (master or standby) with `barman status` command
- Documentation improvements

Bugfixes

- Fix high memory usage with `parallel_jobs > 1` (#116)
- Better handling of errors using parallel copy (#114)
- Make `barman diagnose` more robust with system exceptions
- Let `archive-wal` ignore files with `.tmp` extension

19.1.47 2.2 (2017-07-17)

Notable changes

- Implement parallel copy for backup/recovery through the `parallel_jobs` global/server option to be overridden by the `--jobs` or `-j` runtime option for the backup and recover command. Parallel backup is available only for the `rsync` copy method. By default, it is set to 1 (for behaviour compatibility with previous versions).
- Support custom WAL size for PostgreSQL 8.4 and newer. At backup time, Barman retrieves from PostgreSQL `wal_segment_size` and `wal_block_size` values and computes the necessary calculations.
- Improve check command to ensure that incoming directory is empty when `archiver=off`, and streaming directory is empty when `streaming_archiver=off` (#80).
- Add `external_configuration` to `backup_options` so that users can instruct Barman to ignore backup of configuration files when they are not inside PGDATA (default for Debian/Ubuntu installations). In this case, Barman does not display a warning anymore.

- Add `--get-wal` and `--no-get-wal` options to `barman recover`
- Add `max_incoming_wals_queue` global/server option for the `check` command so that a non blocking error is returned in case incoming WAL directories for both archiver and the `streaming_archiver` contain more files than the specified value.
- Documentation improvements
- File format changes:
 - The format of `backup.info` file has changed. For this reason a backup taken with Barman 2.2 cannot be read by a previous version of Barman. But, backups taken by previous versions can be read by Barman 2.2.

Bugfixes

- Allow replication-status to work against a standby
- Close any PostgreSQL connection before starting `pg_basebackup` (#104, #108)
- Safely handle paths containing special characters
- Archive `.partial` files after promotion of streaming source
- Recursively create directories during recovery (SF#44)
- Improve `xlog.db` locking (#99)
- Remove `tablespace_map` file during `recover` (#95)
- Reconnect to PostgreSQL if connection drops (SF#82)

19.1.48 2.1 (2017-01-05)

Notable changes

- Add `--archive` and `--archive-timeout` options to `switch-xlog` command.
- Preliminary support for PostgreSQL 10 (#73).
- Minor additions:
 - Add last archived WAL info to diagnose output.
 - Add start time and execution time to the output of `delete` command.

Bugfixes

- Return failure for `get-wal` command on inactive server
- Make `streaming_archiver_names` and `streaming_backup_name` options global (#57)
- Fix `rsync` failures due to files truncated during transfer (#64)
- Correctly handle compressed history files (#66)

- Avoid de-referencing symlinks in `pg_tblspc` when preparing recovery (#55)
- Fix comparison of last archiving failure (#40, #58)
- Avoid failing recovery if `postgresql.conf` is not writable (#68)
- Fix output of replication-status command (#56)
- Exclude files from backups like `pg_basebackup` (#65, #72)
- Exclude directories from other Postgres versions while copying tablespaces (#74)
- Make retry hook script options global

19.1.49 2.0 (2016-09-27)

Notable changes

- Support for `pg_basebackup` and base backups over the PostgreSQL streaming replication protocol with `backup_method=postgres` (PostgreSQL 9.1 or higher required)
- Support for physical replication slots through the `slot_name` configuration option as well as the `--create-slot` and `--drop-slot` options for the `receive-wal` command (PostgreSQL 9.4 or higher required). When `slot_name` is specified and `streaming_archiver` is enabled, `receive-wal` transparently integrates with `pg_receivexlog`, and check makes sure that slots exist and are actively used
- Support for the new backup API introduced in PostgreSQL 9.6, which transparently enables concurrent backups and backups from standby servers using the standard `rsync` method of backup. Concurrent backup was only possible for PostgreSQL 9.2 to 9.5 versions through the `pgespresso` extension. The new backup API will make `pgespresso` redundant in the future
- If properly configured, Barman can function as a synchronous standby in terms of WAL streaming. By properly setting the `streaming_archiver_name` in the `synchronous_standby_names` priority list on the master, and enabling replication slot support, the `receive-wal` command can now be part of a PostgreSQL synchronous replication cluster, bringing `RPO=0` (PostgreSQL 9.5.5 or higher required)
- Introduce `barman-wal-restore`, a standard and robust script written in Python that can be used as `restore_command` in `recovery.conf` files of any standby server of a cluster. It supports remote parallel fetching of WAL files by efficiently invoking `get-wal` through SSH. Currently available as a separate project called `barman-cli`. The `barman-cli` package is required for remote recovery when `get-wal` is listed in `recovery_options`
- Control the maximum execution time of the `check` command through the `check_timeout` global/server configuration option (30 seconds by default)
- Limit the number of WAL segments that are processed by an `archive-wal` run, through the `archiver_batch_size` and `streaming_archiver_batch_size` global/server options which control archiving of WAL segments coming from, respectively, the standard archiver and `receive-wal`
- Removed locking of the XLOG database during check operations
- The `show-backup` command is now aware of timelines and properly displays which timelines can be used as recovery targets for a given base backup. Internally, Barman is now capable of parsing `.history` files

- Improved the logic behind the retry mechanism when copy operations experience problems. This involves backup (rsync and postgres) as well as remote recovery (rsync)
- Code refactoring involving remote command and physical copy interfaces

Bugfixes

- Correctly handle .history files from streaming
- Fix replication-status on PostgreSQL 9.1
- Fix replication-status when sent and write locations are not available
- Fix misleading message on pg_receive_xlog termination

19.1.50 1.6.1 (2016-05-23)

Minor changes

- Add --peek option to get-wal command to discover existing WAL files from the Barman's archive
- Add replication-status command for monitoring the status of any streaming replication clients connected to the PostgreSQL server. The --target option allows users to limit the request to only hot standby servers or WAL streaming clients
- Add the switch-xlog command to request a switch of a WAL file to the PostgreSQL server. Through the '--force' it issues a CHECKPOINT beforehand
- Add streaming_archiver_name option, which sets a proper application_name to pg_receive_xlog when streaming_archiver is enabled (only for PostgreSQL 9.3 and above)
- Check for *superuser* privileges with PostgreSQL's standard connections (#30)
- Check the WAL archive is never empty
- Check for 'backup_label' on the master when server is down
- Improve barman-wal-restore contrib script

Bugfixes

- Treat the "failed backups" check as non-fatal
- Rename '-x' option for get-wal as '-z'
- Add archive_mode=always support for PostgreSQL 9.5 (#32)
- Properly close PostgreSQL connections when necessary
- Fix receive-wal for pg_receive_xlog version 9.2

19.1.51 1.6.0 (2016-02-29)

Notable changes

- Support for streaming replication connection through the `streaming_conninfo` server option
- Support for the `streaming_archiver` option that allows Barman to receive WAL files through PostgreSQL's native streaming protocol. When set to 'on', it relies on `pg_receivexlog` to receive WAL data, reducing Recovery Point Objective. Currently, WAL streaming is an additional feature (standard log archiving is still required)
- Implement the `receive-wal` command that, when `streaming_archiver` is on, wraps `pg_receivexlog` for WAL streaming. Add `--stop` option to stop receiving WAL files via streaming protocol. Add `--reset` option to reset the streaming status and restart from the current xlog in Postgres.
- Automatic management (startup and stop) of `receive-wal` command via cron command
- Support for the `path_prefix` configuration option
- Introduction of the `archiver` option (currently fixed to on) which enables continuous WAL archiving for a specific server, through log shipping via PostgreSQL's `archive_command`
- Support for `streaming_wals_directory` and `errors_directory` options
- Management of WAL duplicates in `archive-wal` command and integration with `check` command
- Verify if `pg_receivexlog` is running in `check` command when `streaming_archiver` is enabled
- Verify if failed backups are present in `check` command
- Accept compressed WAL files in incoming directory
- Add support for the `pigz` compressor (thanks to Stefano Zacchiroli zack@upsilon.cc)
- Implement `pygzip` and `pybzip2` compressors (based on an initial idea of Christoph Moench-Tegeder christoph@2ndquadrant.de)
- Creation of an implicit restore point at the end of a backup
- Current size of the PostgreSQL data files in `barman status`
- Permit `archive_mode=always` for PostgreSQL 9.5 servers (thanks to Christoph Moench-Tegeder christoph@2ndquadrant.de)
- Complete refactoring of the code responsible for connecting to PostgreSQL
- Improve messaging of cron command regarding sub-processes
- Native support for Python ≥ 3.3
- Changes of behaviour:
 - Stop trashing WAL files during `archive-wal` (commit:e3a1d16)

Bugfixes

- Atomic WAL file archiving (#9 and #12)
- Propagate "-c" option to any Barman subprocess (#19)
- Fix management of backup ID during backup deletion (#22)
- Improve archive-wal robustness and log messages (#24)
- Improve error handling in case of missing parameters

19.1.52 1.5.1 (2015-11-16)

Minor changes

- Add support for the 'archive-wal' command which performs WAL maintenance operations on a given server
- Add support for "per-server" concurrency of the 'cron' command
- Improved management of xlog.db errors
- Add support for mixed compression types in WAL files (SF.net#61)

Bugfixes

- Avoid retention policy checks during the recovery
- Avoid 'wal_level' check on PostgreSQL version < 9.0 (#3)
- Fix backup size calculation (#5)

19.1.53 1.5.0 (2015-09-28)

Notable changes

- Add support for the get-wal command which allows users to fetch any WAL file from the archive of a specific server
- Add support for retry hook scripts, a special kind of hook scripts that Barman tries to run until they succeed
- Add active configuration option for a server to temporarily disable the server by setting it to False
- Add barman_lock_directory global option to change the location of lock files (by default: 'barman_home')
- Execute the full suite of checks before starting a backup, and skip it in case one or more checks fail
- Forbid to delete a running backup
- Analyse include directives of a PostgreSQL server during backup and recover operations
- Add check for conflicting paths in the configuration of Barman, both intra (by temporarily disabling a server) and inter-server (by refusing any command, to any server).
- Add check for wal_level

- Add barman-wal-restore script to be used as restore_command on a standby server, in conjunction with barman get-wal
- Implement a standard and consistent policy for error management
- Improved cache management of backups
- Improved management of configuration in unit tests
- Tutorial and man page sources have been converted to Markdown format
- Add code documentation through Sphinx
- Complete refactor of the code responsible for managing the backup and the recover commands
- Changed internal directory structure of a backup
- Introduce copy_method option (currently fixed to rsync)

Bugfixes

- Manage options without '=' in PostgreSQL configuration files
- Preserve Timeline history files (Fixes: #70)
- Workaround for rsync on SUSE Linux (Closes: #13 and #26)
- Disables dangerous settings in postgresql.auto.conf (Closes: #68)

19.1.54 1.4.1 (2015-05-05)

Minor changes

- Improved management of xlogdb file, which is now correctly fsynced when updated. Also, the rebuild-xlogdb command now operates on a temporary new file, which overwrites the main one when finished.
- Add unit tests for dateutil module compatibility
- Modified Barman version following PEP 440 rules and added support of tests in Python 3.4

Bugfixes

- Fix for WAL archival stop working if first backup is EMPTY (Closes: #64)
- Fix exception during error handling in Barman recovery (Closes: #65)
- After a backup, limit cron activity to WAL archiving only (Closes: #62)
- Improved robustness and error reporting of the backup delete command (Closes: #63)
- Fix computation of WAL production ratio as reported in the show-backup command

19.1.55 1.4.0 (2015-01-26)

Notable changes

- Incremental base backup implementation through the `reuse_backup` global/server option. Possible values are `off` (disabled, default), `copy` (preventing unmodified files from being transferred) and `link` (allowing for deduplication through hard links).
- Store and show deduplication effects when using `reuse_backup= link`.
- Added transparent support of `pg_stat_archiver` (PostgreSQL 9.4) in `check`, `show-server` and `status` commands.
- Improved administration by invoking WAL maintenance at the end of a successful backup.
- Changed the way unused WAL files are trashed, by differentiating between concurrent and exclusive backup cases.
- Improved performance of WAL statistics calculation.
- Treat a missing `pg_ident.conf` as a `WARNING` rather than an error.
- Refactored output layer by removing remaining `yield` calls.
- Check that `rsync` is in the system path.
- Include history files in WAL management.
- Improved robustness through more unit tests.

Bugfixes

- Fixed bug #55: Ignore `fsync EINVAL` errors on directories.
- Fixed bug #58: retention policies delete.

19.1.56 1.3.3 (2014-08-21)

Notable changes

- Added `"last_backup_max_age"`, a new global/server option that allows administrators to set the max age of the last backup in a catalogue, making it easier to detect any issues with periodical backup execution
- Improved robustness of `"barman backup"` by introducing two global/ server options: `"basebackup_retry_times"` and `"basebackup_retry_sleep"`. These options allow an administrator to specify, respectively, the number of attempts for a copy operation after a failure, and the number of seconds of wait before retrying
- Improved the recovery process via `rsync` on an existing directory (incremental recovery), by splitting the previous `rsync` call into several ones - invoking checksum control only when necessary
- Added support for PostgreSQL 8.3

Minor changes

- Support for comma separated list values configuration options
- Improved backup durability by calling fsync() on backup and WAL files during "barman backup" and "barman cron"
- Improved Nagios output for "barman check --nagios"
- Display compression ratio for WALs in "barman show-backup"
- Correctly handled keyboard interruption (CTRL-C) while performing barman backup
- Improved error messages of failures regarding the stop of a backup
- Wider coverage of unit tests

Bugfixes

- Copies "recovery.conf" on the remote server during "barman recover" (#45)
- Correctly detect pre/post archive hook scripts (#41)

19.1.57 1.3.2 (2014-04-15)

Bugfixes

- Fixed incompatibility with PostgreSQL 8.4 (Closes #40, bug introduced in version 1.3.1)

19.1.58 1.3.1 (2014-04-14)

Minor changes

- Added support for concurrent backup of PostgreSQL 9.2 and 9.3 servers that use the "pgespresso" extension. This feature is controlled by the "backup_options" configuration option (global/ server) and activated when set to "concurrent_backup". Concurrent backup allows DBAs to perform full backup operations from a streaming replicated standby.
- Added the "barman diagnose" command which prints important information about the Barman system (extremely useful for support and problem solving)
- Improved error messages and exception handling interface

Bugfixes

- Fixed bug in recovery of tablespaces that are created inside the PGDATA directory (bug introduced in version 1.3.0)
- Fixed minor bug of unhandled -q option, for quiet mode of commands to be used in cron jobs (bug introduced in version 1.3.0)
- Minor bug fixes and code refactoring

19.1.59 1.3.0 (2014-02-03)

Notable changes

- Refactored BackupInfo class for backup metadata to use the new FieldListFile class (infofile module)
- Refactored output layer to use a dedicated module, in order to facilitate integration with Nagios (NagiosOutputWriter class)
- Refactored subprocess handling in order to isolate stdin/stderr/ stdout channels (command_wrappers module)
- Refactored hook scripts management
- Extracted logging configuration and userid enforcement from the configuration class.
- Support for hook scripts to be executed before and after a WAL file is archived, through the 'pre_archive_script' and 'post_archive_script' configuration options.
- Implemented immediate checkpoint capability with --immediate-checkpoint command option and 'immediate_checkpoint' configuration option
- Implemented network compression for remote backup and recovery through the 'network_compression' configuration option (#19)
- Implemented the 'rebuild-xlogdb' command (Closes #27 and #28)
- Added deduplication of tablespaces located inside the PGDATA directory
- Refactored remote recovery code to work the same way local recovery does, by performing remote directory preparation (assuming the remote user has the right permissions on the remote server)
- 'barman backup' now tries and create server directories before attempting to execute a full backup (#14)

Bugfixes

- Fixed bug #22: improved documentation for tablespaces relocation
- Fixed bug #31: 'barman cron' checks directory permissions for lock file
- Fixed bug #32: xlog.db read access during cron activities

19.1.60 1.2.3 (2013-09-05)

Minor changes

- Added support for PostgreSQL 9.3
- Added support for the "--target-name" recovery option, which allows to restore to a named point previously specified with pg_create_restore_point (only for PostgreSQL 9.1 and above users)
- Introduced Python 3 compatibility

Bugfixes

- Fixed bug #27 about flock() usage with barman.lockfile (many thanks to Damon Snyder damonsnyder@users.sf.net)

19.1.61 1.2.2 (2013-06-24)

Bugfixes

- Fix python 2.6 compatibility

19.1.62 1.2.1 (2013-06-17)

Minor changes

- Added the "bandwidth_limit" global/server option which allows to limit the I/O bandwidth (in KBPS) for backup and recovery operations
- Added the "tablespace_bandwidth_limit" global/server option which allows to limit the I/O bandwidth (in KBPS) for backup and recovery operations on a per tablespace basis
- Added /etc/barman/barman.conf as default location

Bugfixes

- Avoid triggering the minimum_redundancy check on FAILED backups (thanks to Jérôme Vanandruel)

19.1.63 1.2.0 (2013-01-31)

Notable changes

- Added the "retention_policy_mode" global/server option which defines the method for enforcing retention policies (currently only "auto")
- Added the "minimum_redundancy" global/server option which defines the minimum number of backups to be kept for a server
- Added the "retention_policy" global/server option which defines retention policies management based on redundancy (e.g. REDUNDANCY 4) or recovery window (e.g. RECOVERY WINDOW OF 3 MONTHS)
- Added retention policy support to the logging infrastructure, the "check" and the "status" commands
- The "check" command now integrates minimum redundancy control
- Added retention policy states (valid, obsolete and potentially obsolete) to "show-backup" and "list-backup" commands
- The 'all' keyword is now forbidden as server name
- Added basic support for Nagios plugin output to the 'check' command through the --nagios option
- Barman now requires argh => 0.21.2 and argcomplete-
- Minor bug fixes

19.1.64 1.1.2 (2012-11-29)

Minor changes

- Added "configuration_files_directory" option that allows to include multiple server configuration files from a directory
- Support for special backup IDs: latest, last, oldest, first
- Management of multiple servers to the 'list-backup' command. 'barman list-backup all' now list backups for all the configured servers.
- Added "application_name" management for PostgreSQL >= 9.0

Bugfixes

- Fixed bug #18: ignore missing WAL files if not found during delete

19.1.65 1.1.1 (2012-10-16)

Bugfixes

- Fix regressions in recover command.

19.1.66 1.1.0 (2012-10-12)

Notable changes

- Support for hook scripts to be executed before and after a 'backup' command through the 'pre_backup_script' and 'post_backup_script' configuration options.
- Management of multiple servers to the 'backup' command. 'barman backup all' now iteratively backs up all the configured servers.
- Add warning in recovery when file location options have been defined in the postgresql.conf file (issue #10)
- Fail fast on recover command if the destination directory contains the ':' character (Closes: #4) or if an invalid tablespace relocation rule is passed
- Report an informative message when pg_start_backup() invocation fails because an exclusive backup is already running (Closes: #8)

Bugfixes

- Fixed bug #9: "9.2 issue with pg_tablespace_location()"

19.1.67 1.0.0 (2012-07-06)

Notable changes

- Backup of multiple PostgreSQL servers, with different versions. Versions from PostgreSQL 8.4+ are supported.
- Support for secure remote backup (through SSH)
- Management of a catalog of backups for every server, allowing users to easily create new backups, delete old ones or restore them
- Compression of WAL files that can be configured on a per server basis using compression/decompression filters, both predefined (gzip and bzip2) or custom
- Support for INI configuration file with global and per-server directives. Default location for configuration files are /etc/barman.conf or ~/.barman.conf. The '-c' option allows users to specify a different one
- Simple indexing of base backups and WAL segments that does not require a local database
- Maintenance mode (invoked through the 'cron' command) which performs ordinary operations such as WAL archival and compression, catalog updates, etc.
- Added the 'backup' command which takes a full physical base backup of the given PostgreSQL server configured in Barman
- Added the 'recover' command which performs local recovery of a given backup, allowing DBAs to specify a point in time. The 'recover' command supports relocation of both the PGDATA directory and, where applicable, the tablespaces
- Added the '--remote-ssh-command' option to the 'recover' command for remote recovery of a backup. Remote recovery does not currently support relocation of tablespaces
- Added the 'list-server' command that lists all the active servers that have been configured in barman
- Added the 'show-server' command that shows the relevant information for a given server, including all configuration options
- Added the 'status' command which shows information about the current state of a server, including Postgres version, current transaction ID, archive command, etc.
- Added the 'check' command which returns 0 if everything Barman needs is functioning correctly
- Added the 'list-backup' command that lists all the available backups for a given server, including size of the base backup and total size of the related WAL segments
- Added the 'show-backup' command that shows the relevant information for a given backup, including time of start, size, number of related WAL segments and their size, etc.
- Added the 'delete' command which removes a backup from the catalog
- Added the 'list-files' command which lists all the files for a single backup
- RPM Package for RHEL 5/6

第20章 よくある質問

20.1 一般

Can Barman perform physical backups of Postgres instances?

はい。Barman は、ベースバックアップと WAL アーカイブを管理する Postgres サーバーの物理バックアップのためのアプリケーションです。ディザスターリカバリーアプリケーションです。Barman は、論理バックアップ別名ダンプをサポートしていません。

I am already using pg_dump. What is the difference between pg_dump and Barman and why should I use Barman instead?

すでに pg_dump を使用している場合、それは良いことです。ただし、ビジネスがデータベースに基づいている場合、論理バックアップ pg_dump によって実行されるものでは十分ではありません。これらのダンプは、特定の時点でのデータベースのスナップショットです。通常、人々はこれらの活動を夜に行います。日中にクラッシュが発生した場合、ダンプが開始されてからクラッシュまでのすべてのトランザクションは永久に失われます。このコンテキストでは、ポイント インタイム リカバリーを可能にする物理的バックアップに基づいて、ディザスターリカバリーのためのより堅牢なソリューションを導入する必要があります。

I manage several Postgres instances. Can Barman manage multiple database servers?

はい。Barman は、Postgres サーバーのリモートバックアップを可能にするように設計されています。これにより、DBA<DBA>は一元化されたホストから複数のサーバーのバックアップを管理できます。Barman を使用すると、WAL セグメントのベースバックアップと継続的アーカイブのためのバックアップサーバーのカタログを定義できます。

Does Barman manage replication and high availability as well? How does it compare with repmgr, OmniPITR, walmgr and similar tools?

いいえ。Barman は、純粋なディザスターリカバリーソリューションを目指しています。Postgres サーバーのクラスターの唯一のバックアップを担当します。高可用性を探している場合は、Patroni や repmgr などのツールを使用することをお勧めします。Barman は、高可用性をカバーするために必要なデザインよりもシンプルで侵襲の少ないデザインが必要であるため、特にディザスターリカバリーのみをターゲットとしています。これが、Barman が上記のような既存の HA ツールを複製しない理由です。サーバーには何もインストールする必要はありません。唯一の要件は、Barman クライアントのアクセスを構成することです。これは、高可用性に必要なものよりも侵襲的です。

Can I define retention policies for my backups?

はい。Barman 1.2.0 は、使用しているすべてのサーバーのベースバックアップと WAL セグメントの両方のリテンションポリシーのサポートを導入しました。サーバー構成ファイルで保持ポリシーを指定できます。リテンションポリシーは、時間たとえば RECOVERY WINDOW OF 30 DAYS またはベースバックアップの数たとえば REDUNDANCY 3 に基づくことができます。

Can I specify different retention policies for base backups and WAL segments?

はい、確かに。たとえば、過去 12 か月のベースバックアップを保持し、先月のみポイントインタイムリカバリーの WAL を保持したい場合が考えられます。

Does Barman guarantee data protection and security?

Barman は、SSH 接続プロセスとファイル管理用と標準の Postgres 接続データベースの照会用を使用して、リモート/ローカル Postgres サーバーと通信します。SSH 通信が安全な方法で発生することを確認するのはシステム管理者の義務です。同様に、データベースサーバーとの通信が安全な方法で発生することを確認するのはデータベース管理者の義務です。

Is the Barman interface complex or hard to understand?

Barman には、サーバーのリスト、バックアップのリスト、バックアップの詳細情報の表示などのいくつかのコマンドを実行できるシンプルなコンソールインターフェイスがあります。サーバーの新しいベースバックアップを起動すること、およびローカルに復元することも簡単です。またはリモートで。

Can Barman manage backups in the cloud e.g. using AWS S3 or Azure Blob Storage?

はい。Barman は現在、Amazon、Azure、Google クラウドプロバイダーでバックアップを管理できます。このバックアップ方法は、Barman 用語でスナップショットバックアップと呼ばれます。このシナリオでは、データベースサーバーが存在するストレージボリュームのスナップショットを介してバックアップが取得されます。この場合、Barman は、主に WAL ファイル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。詳細については、[Barman Cloud](#) および [Barman Cloud](#) セクションを確認してください。

Do you have packages for RedHat and Debian based distributions?

Barman 公式パッケージは、PGDG によって提供されています。Barman パッケージは代替リポジトリで見つけることができますが、互換性、安定性、最新の更新へのアクセスが保証されるため、`:term:'PGDG'` リポジトリの使用をお勧めします。詳細については、`:ref:'installation'` セクションを参照してください。

Does Barman allow me to limit the bandwidth usage for backup and recovery?

はい。Barman 1.2.1 は、グローバル、サーバー、およびテーブルスペースレベルでの帯域幅制限のサポートを導入しました。

20.2 バックアップ

Does Barman support incremental backups

はい。Barman は、使用しているバックアップ方法とバックアップする Postgres バージョンに応じて、ファイルレベルとブロックレベルのインクリメンタルバックアップをサポートしています。

How many backups can Barman handle for each Postgres instance?

Barman、データベースサーバーごとに複数のベースバックアップを保存できます。バックアップ数の唯一の制限は、Barman ホスト上のディスク領域です。

I have continuous archiving in place, but managing WAL files and understanding which ones “belong” to a particular base backup is not obvious. Can Barman simplify this?

幸いなことに、はい。Barman が動作する方法は、特定のサーバーの WAL セグメントからベースバックアップを分離することです。これらは非常に関連していますが、Barman は、WAL アーカイブを、最初

のベースバックアップから最後に使用可能なデータまでの連続したデータのストリームと考えています。Barman の優れた機能は、すべての WAL ファイルをベースバックアップにリンクすることにより、ベースバックアップと WAL セグメントの 2 つのコンポーネントの観点からバックアップのサイズを決定できることです。

Can Barman compress base backups?

現在、Barman は `pg_basebackup` 圧縮機能のおかげで、`backup_method = postgres` を使用してバックアップを圧縮できます。これは、`backup_compression` 構成オプションを使用して有効にできます。Rsync ベースのバックアップの場合、現時点では圧縮方法はありませんが、実行可能であり、現在のデザインで許可されています。詳細については、`backup_method = postgres` セクションを確認してください。

Can Barman compress WAL segments?

はい。WAL セグメントの圧縮フィルタを指定し、WAL ファイルのサイズを 5/10 倍に大幅に削減できます。これは、構成で `compression` オプションを設定することにより実行されます。詳細については、`compression` セクションを参照してください。

Can Barman back up tablespaces?

はい。テーブルスペースは、Barman によって透過的かつ自動的に処理されます。

Can I backup from a Postgres standby server?

はい、Barman は、`postgres` および `rsync` バックアップ方法の両方でスタンバイサーバーからのバックアップをネイティブにサポートしています。

What's the difference between Full and Incremental backups when using the rsync backup method in Barman?

`rsync` バックアップ方法では、オンディスクバックアップにフルバックアップとインクリメンタルバックアップの明確な区別はありません。実際、`rsync` で作成されるバックアップはすべてフルバックアップですが、ハードリンクを使用して共通ファイルを共有する場合があるため、記憶領域が削減され、バックアップの作成が高速化されます。

`rsync` を `reuse_backup = link` とともに使用する場合、最後のバックアップ以降にまったく同じファイルは再度コピーされません。代わりに、既存のファイルへのハードリンクが作成されます。これにより、すべてのファイルがバックアップフォルダーで完全に利用できるため、バックアップがいっぱいであるように見えますが、変更されていないファイルは複製されるのではなくリンクされているため、ディスク上で使用および転送される実際のサイズは小さくなります。このため、各 `rsync` バックアップは完全な "スナップショット" であり、以前のバックアップとは独立しており、バックアップを削除しても他のバックアップは変更されません。

注釈

ハードリンクは、ディスクに保存されているファイルの実際のデータへのポインタです。ハードリンクが作成されると、ストレージデバイス上の同じ基になるデータブロックを指すため、複数のファイル名を介して同じデータにアクセスできます。

これと比較して、`backup_method = postgres` 方法 Postgres 17 以降では、インクリメンタルバックアップはバックアップのチェーンに依存し、インクリメンタルバックアップを復元するには、フルバックアップ

プおよび中間インクリメンタルバックアップと組み合わせる必要があります。この場合、インクリメンタルバックアップを削除すると、後続のインクリメンタルバックアップが無効になります。

要約すると、rsync バックアップは、変更のないファイルの重複を回避するという点でファイルレベルのインクリメンタルですが、各バックアップは、以前のバックアップから独立した完全な"スナップショット"のままです。

20.3 インストールと構成

Does Barman have to be installed on the same server as Postgres?

いいえ。Barman は、Postgres が実行されているホストと同じホスト上にある必要はありません。ローカルにインストールするか、別のサーバー通常はバックアップ専用にインストールするかは、あなたの選択です。Barman 専用のサーバーを使用することを強くお勧めします。

Can I have multiple configurations for different users in a Barman server?

はい。Barman には構成ファイルが必要です。システム全体の構成 `/etc/barman.conf` またはユーザー構成 `~/.barman.conf` を使用できます。たとえば、Barman を使用するシステムに複数のユーザーを設定し、各ユーザーが管理対象 Postgres サーバーのサブセットで作業できます。この方法により、ユーザーごとにバックアップを保護できます。

20.4 リカバリー

Does Barman manage recovery?

はい。Barman を使用すると、バックアップサーバーまたはリモートノードで Postgres インスタンスを復旧できます。リモートでの復旧は、`recover` コマンドで SSH コマンドを指定するだけです。Barman はこれを使用して、バックアップを復元するために宛先ホストに接続します。

Does Barman manage recovery to a specific transaction or to a specific time?

はい。Barman、タイムスタンプ、トランザクション ID、ログシーケンス番号 LSN、または以前に作成した名前付け復元ポイントを指定することにより、ポイントインタイムリカバリーを実行できます。`recover` コマンドに追加オプションを追加するだけです。詳細については、Recovery セクションの `recover` を参照してください。

Does Barman support timelines?

はい、Barman は、復旧の Postgres タイムラインを処理します。

Does Barman handle tablespaces and their mapping during recovery operations?

はい。デフォルトでは、テーブルスペースはソースサーバーにあったのと同じパスに復元されます。`recover` コマンドで `--tablespace` オプションを指定することにより、必要に応じて再マップできます。

During recovery, does Barman allow me to relocate the PGDATA directory? What about tablespaces?

はい。サーバーを復旧するときに、PGDATA ディレクトリとすべてのテーブルスペースがある場合は、別の場所を指定できます。これにより、一時的なサンドボックスサーバーをセットアップできます。これは、サンドボックスサーバーからテーブルをダンプし、マスターサーバーで再作成することにより、マスターから意図せずドロップしたテーブルを復旧する場合に特に役立ちます。

20.5 要件

Does Barman work on Windows?

Barman は、Windows で Postgres サーバーのバックアップを取得できます。リカバリ部分はサポートされていません。さらに、Barman は UNIX ボックスで実行する必要があります。

第21章 ライセンス

Barman は EnterpriseDB UK Limited が所有しており、GNU General Public License v3 に基づいてライセンスされています。

Barman は、当初、欧州委員会の第7次フレームワークプログラムの支援によりサポートされている研究プロジェクト 4CaaSt から部分的な資金提供を受けています。

Barman への貢献を歓迎します。これらの貢献はリリースノートで確認されます。すべての Barman コードを無料のままにするために、EnterpriseDB UK Limited は、寄稿者に著作権の割り当てを完了し、雇用主からのレンタル作品の請求を否認することを求めています。著作権譲渡フォームについては、barman@enterprisedb.com までご連絡ください。

© Copyright EnterpriseDB UK Limited 2011-2025

索引

A

AWS, [201](#)

B

Barman, [201](#)

D

DBA, [201](#)

DEB, [201](#)

G

GCP, [201](#)

GPG, [201](#)

I

IAC, [201](#)

ICT, [201](#)

L

libpq, [201](#)

P

PGDATA, [201](#)

PGDG, [201](#)

PITR, [201](#)

R

RHEL, [202](#)

RPM, [202](#)

RPO, [202](#)

S

SLES, [202](#)

SPOF, [202](#)

V

VLDB, [202](#)

W

WORM, [202](#)

ン

外部構成ファイル, [201](#)