

Barman User Guide

リリース *3.19.0*

EnterpriseDB UK Limited

2026 年 05 月 26 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 . . . 14
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　データベースユーザー . . . 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 ストリーミングバックアップ . . . 50
7.3 Rsync バックアップ . . . 52
7.4 ローカル Rsync バックアップ . . . 54
7.5 ローカルからクラウドへのバックアップ . . . 54
7.6 一般的なバックアップ設定 . . . 58

第 8 章 クラウド スナップショット バックアップ **67**
8.1 要件と構成 . . . 68
8.2 Google Cloud Platform . . . 68
8.3 Microsoft Azure . . . 69
8.4 アマゾン ウェブ サービス . . . 70
8.5 バックアッププロセス . . . 72

第 9 章 WAL アーカイブ **77**
9.1 WAL 圧縮 . . . 77
9.2 WAL 暗号化 . . . 79
9.3 同期 WAL ストリーミング . . . 79

第 10 章 カタログ情報 **81**
10.1 目的 . . . 81
10.2 使用法 . . . 82

第 11 章 リカバリー **85**
11.1 ローカルリカバリー . . . 86
11.2 リモートリカバリー . . . 86
11.3 デルタリカバリー . . . 86
11.4 テーブルスペースのリマッピング . . . 87
11.5 ポイント インタイム リカバリー . . . 87
11.6 Barman から WAL の取得 . . . 88
11.7 暗号化されたバックアップの復元 . . . 92
11.8 圧縮バックアップの復元 . . . 93
11.9 ブロックレベルのインクリメンタル バックアップのリカバリー . . . 93
11.10 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 . . . 95
11.11 クラウドからのバックアップの復元 . . . 96
11.12 .partial WAL ファイルの制限 . . . 97
11.13 外部構成ファイルの管理 . . . 97
11.14 スナップショットバックアップからの復旧 . . . 98

第 12 章 リテンション ポリシー **101**
12.1 概要 . . . 101
12.2 リテンションポリシーの主要コンポーネント . . . 101
12.3 リテンションポリシーの主な目的 . . . 102
12.4 最小限の冗長性の安全性 . . . 102
12.5 リテンションポリシーの範囲 . . . 103
12.6 ユースケース . . . 104
12.7 保持ポリシーが適用される方法 . . . 104
12.8 構成と構文 . . . 104
12.9 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 . . . 105

12.10 クラウドバックアップのリテンションポリシー . . . 105

第 13 章 バックアップのエクスポートとインポート 107
13.1 概要 . . . 107
13.2 制限事項 . . . 107
13.3 Tarball レイアウト . . . 107
13.4 Tarball ファイル名と完全性 . . . 108
13.5 使用法 . . . 108

第 14 章 診断と障害対応 111
14.1 Barman ステータス . . . 111
14.2 Barman ショーサーバー . . . 112
14.3 Barman 複製ステータス . . . 112
14.4 Barman 診断 . . . 112

第 15 章 構成リファレンス 115
15.1 使用法 . . . 115
15.2 オプション . . . 116
15.3 構成モデル . . . 144

第 16 章 コマンドリファレンス 147
16.1 `barman` . . . 147
16.2 ショートカット . . . 149
16.3 終了ステータス . . . 149
16.4 サブコマンド . . . 149
16.5 `barman-cli` コマンド . . . 180

第 17 章 地理的な冗長性 185
17.1 同期情報 . . . 185
17.2 構成 . . . 185
17.3 ノードの同期 . . . 186
17.4 手動同期 . . . 186

第 18 章 フックスクリプト 187
18.1 バックアップを作成する前と後 . . . 188
18.2 バックアップを削除する前と後 . . . 189
18.3 WAL がアーカイブされる前と後 . . . 189
18.4 WAL ファイルが削除される前と後 . . . 190
18.5 バックアップを復元する前と後 . . . 190
18.6 barman でフックとして `barman-cloud-*` スクリプトを使用する . . . 190

第 19 章 クラウドの Barman 193
19.1 Barman クラウドクライアントパッケージ . . . 194
19.2 インストール . . . 195
19.3 オブジェクトロックのサポート . . . 195
19.4 コマンドリファレンス . . . 196

第 20 章 よくある質問 227
20.1 一般 . . . 227

20.2 バックアップ . . . 228
20.3 インストールと構成 . . . 230
20.4 リカバリー . . . 230
20.5 要件 . . . 231

第 21 章 ライセンス **233**

第 22 章 Barman リリースノート **235**
22.1 3.19.02026-05-20 . . . 235
22.2 3.18.02026-03-12 . . . 238
22.3 3.17.02026-01-07 . . . 242
22.4 3.16.22025-11-04 . . . 245
22.5 3.16.12025-10-14 . . . 245
22.6 3.16.02025-10-02 . . . 246
22.7 3.15.02025-08-05 . . . 248
22.8 3.14.12025-06-18 . . . 253
22.9 3.14.02025-05-15 . . . 254
22.10 3.13.32025-04-24 . . . 256
22.11 3.13.22025-03-27 . . . 257
22.12 3.13.12025-03-20 . . . 258
22.13 3.13.02025-02-20 . . . 259
22.14 3.12.12024-12-09 . . . 262
22.15 3.12.02024-11-21 . . . 263
22.16 3.11.12024-08-22 . . . 267
22.17 3.11.02024-08-22 . . . 267
22.18 3.10.12024-06-12 . . . 270
22.19 3.10.02024-01-24 . . . 270
22.20 3.9.02023-10-03 . . . 271
22.21 3.8.02023-08-31 . . . 271
22.22 3.7.02023-07-25 . . . 272
22.23 3.6.02023-06-15 . . . 272
22.24 3.5.02023-03-29 . . . 273
22.25 3.4.02023-01-26 . . . 274
22.26 3.3.02022-12-14 . . . 274
22.27 3.2.02022-10-20 . . . 275
22.28 3.1.02022-09-14 . . . 275
22.29 3.0.12022-06-27 . . . 276
22.30 3.0.02022-06-23 . . . 276
22.31 2.192022-03-09 . . . 278
22.32 2.182022-01-21 . . . 278
22.33 2.172021-12-01 . . . 279
22.34 2.162021-11-17 . . . 279
22.35 2.152021-10-12 . . . 279
22.36 2.142021-09-22 . . . 280
22.37 2.132021-07-26 . . . 280
22.38 2.12.12021-06-30 . . . 281
22.39 2.122020-11-05 . . . 281

22.40 2.112020-07-09 . . . 282
22.41 2.102019-12-05 . . . 283
22.42 2.92019-08-01 . . . 284
22.43 2.82019-05-17 . . . 284
22.44 2.72019-03-12 . . . 285
22.45 2.62019-02-04 . . . 285
22.46 2.52018-10-23 . . . 286
22.47 2.42018-05-25 . . . 286
22.48 2.32017-09-05 . . . 287
22.49 2.22017-07-17 . . . 288
22.50 2.12017-01-05 . . . 289
22.51 2.02016-09-27 . . . 290
22.52 1.6.12016-05-23 . . . 291
22.53 1.6.02016-02-29 . . . 291
22.54 1.5.12015-11-16 . . . 293
22.55 1.5.02015-09-28 . . . 293
22.56 1.4.12015-05-05 . . . 294
22.57 1.4.02015-01-26 . . . 295
22.58 1.3.32014-08-21 . . . 295
22.59 1.3.22014-04-15 . . . 296
22.60 1.3.12014-04-14 . . . 296
22.61 1.3.02014-02-03 . . . 297
22.62 1.2.32013-09-05 . . . 297
22.63 1.2.22013-06-24 . . . 298
22.64 1.2.12013-06-17 . . . 298
22.65 1.2.02013-01-31 . . . 298
22.66 1.1.22012-11-29 . . . 299
22.67 1.1.12012-10-16 . . . 299
22.68 1.1.02012-10-12 . . . 299
22.69 1.0.02012-07-06 . . . 300

第 23 章 用語集 **303**

索引 **305**

: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

ディザスター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ーリカバリーと呼び、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、ディザスターリカバリーに関連する文化的な側面をより強調することが重要です。人間のいない道具は役に立ちません。 Barman との私たちの使命は、次のようなディザスターリカバリーの文化を促進することです。

- バックアップ手順に焦点を当てます。
- 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ー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、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、その後に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**
>
> 特定の Postgres サーバーのバックアップを管理するには、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切り替える必要がある場合は、 Barman 構成で新しい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 公式コミュニティパッケージは、 :term:`PGDG`によって提供されています。

> **注釈**
>
> EnterpriseDB の顧客は、EDB が提供するパッケージを使用する必要があります。これにより、他の EDB 製品とのより良い互換性が保証されます。

Barman 機能スイートを構成するパッケージは 4 つあります `barman` 、 `python3-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。
- `python3-barman` は、 Barman の Python ライブラリを保持するパッケージです。これは、他のすべての Barman パッケージに必要なパッケージです。
- `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。 EDB サブスクリプションをお持ちでない限り、 PGDG リポジトリを使用することをお勧めします。互換性、安定性、最新の更新へのアクセスが保証されます。

> **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 バージョン 3.8 から 3.14
- 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`
  - `argcomplete` オプショナル
- 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。

```
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。

```
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、 :term: 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の例では、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:term:*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: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。

```
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の `.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ーリカバリー計画は、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。

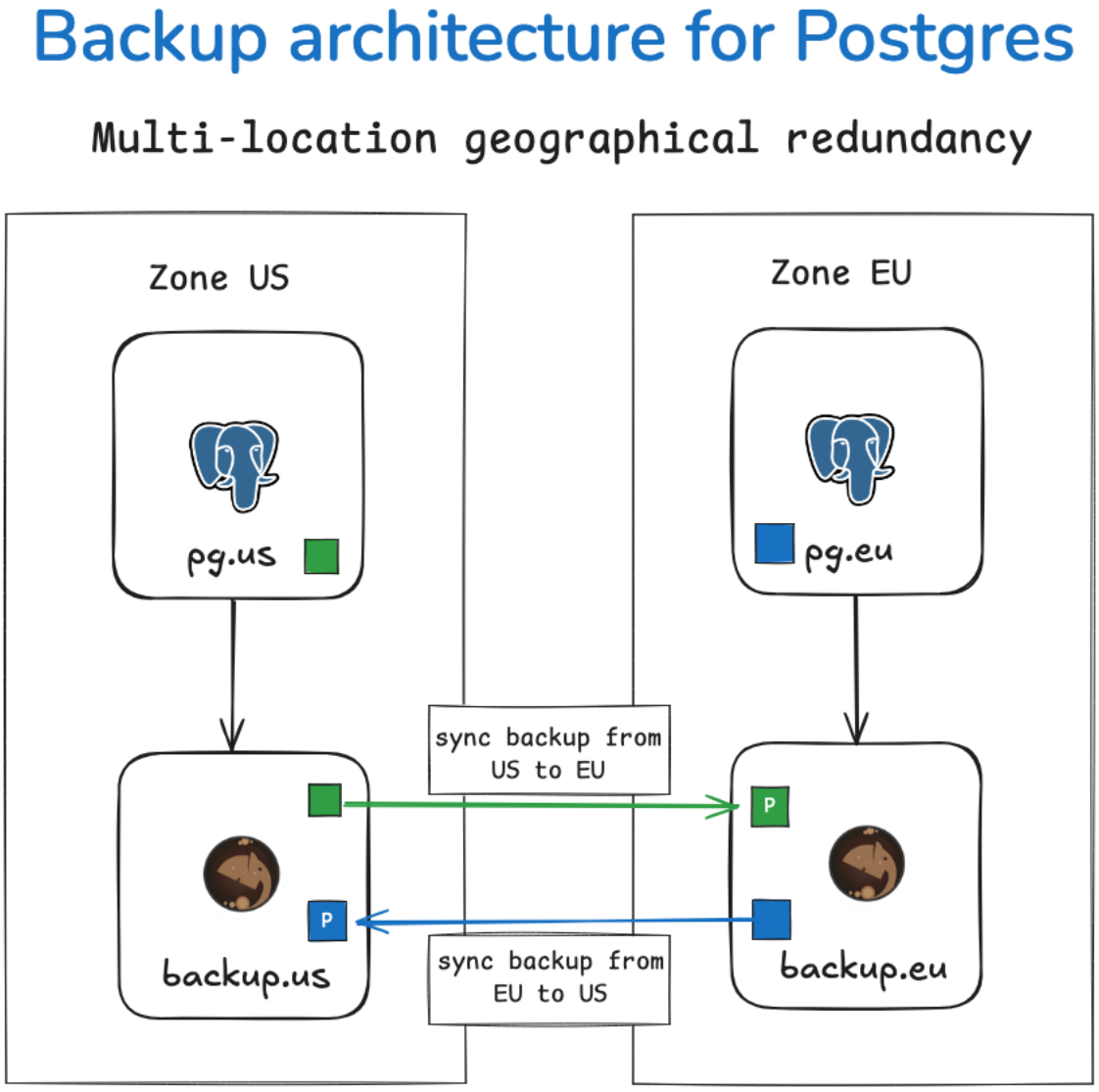
ディザスターリカバリーアーキテクチャのモデリングを開始するための良い方法には、次のものが含まれます。

- 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分自身です。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/アベイラビリティーゾーンにある 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ーゾーン 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。



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、複雑なディザスター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、現在パラ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。

> **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。

> **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、特定の Postgres サーバーの構成に `backup_method = postgres` を含めることにより、この方法を設定できます。

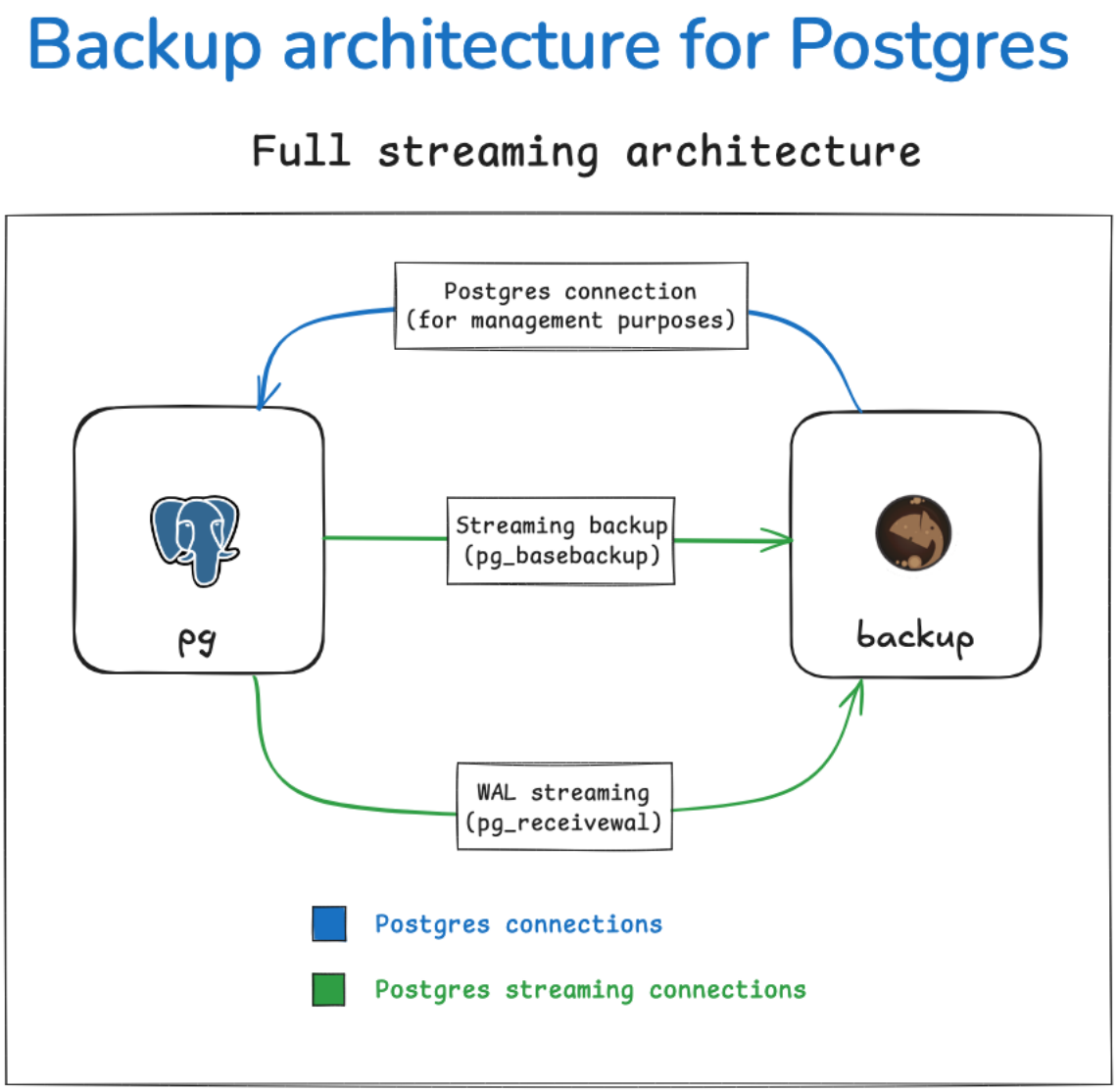
このアプローチでは、 Postgres 17 以降で利用可能な `pg_basebackup` が提供する *block-level incremental backups* サポートを活用できます。ブロックレベルのインクリメンタルバックアップは、重複排除率の点で

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。



構成するには、次のものが必要です。

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).

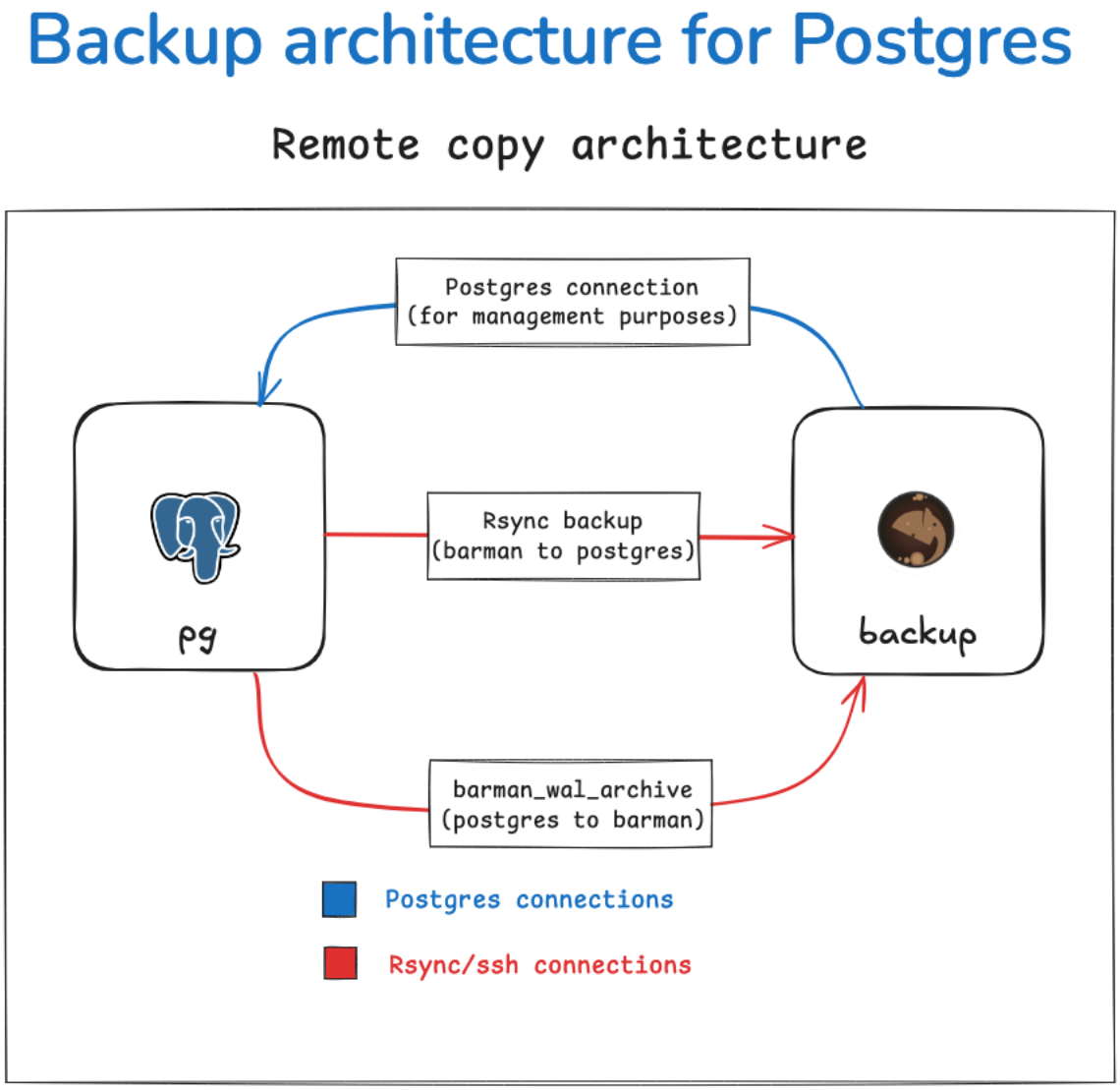
4.5.2 シナリオ 2 rsync / SSH を介したバックアップ

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。



構成するには、次のものが必要です。

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。

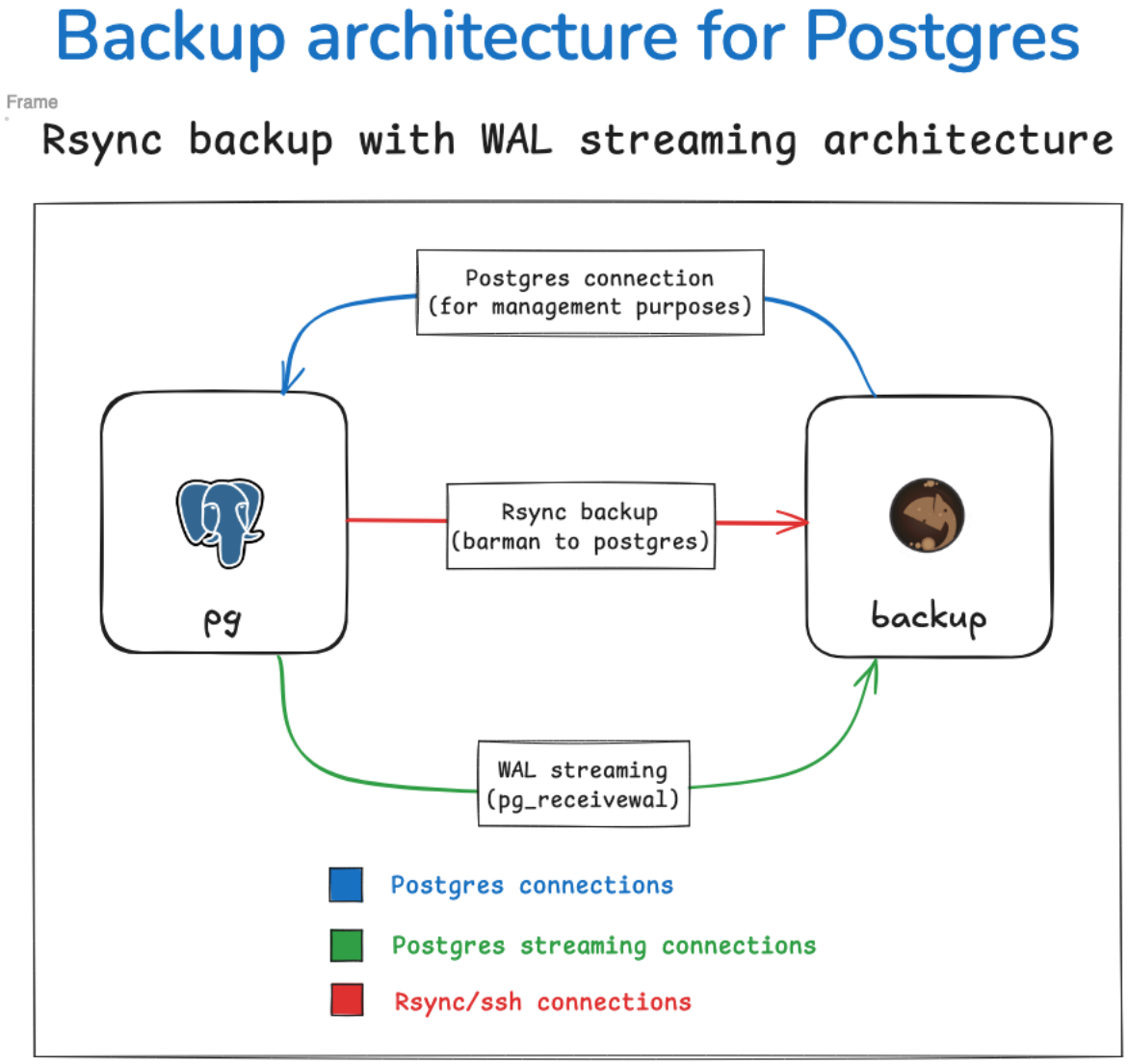
2. An SSH connection to be used by Rsync for base backup operations that allow the **barman** user on the Barman server to connect as the **postgres** user on the Postgres server.

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。



これを行うには、 `backup_method` を `rsync` として構成し、特定の Postgres サーバーの構成で `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。

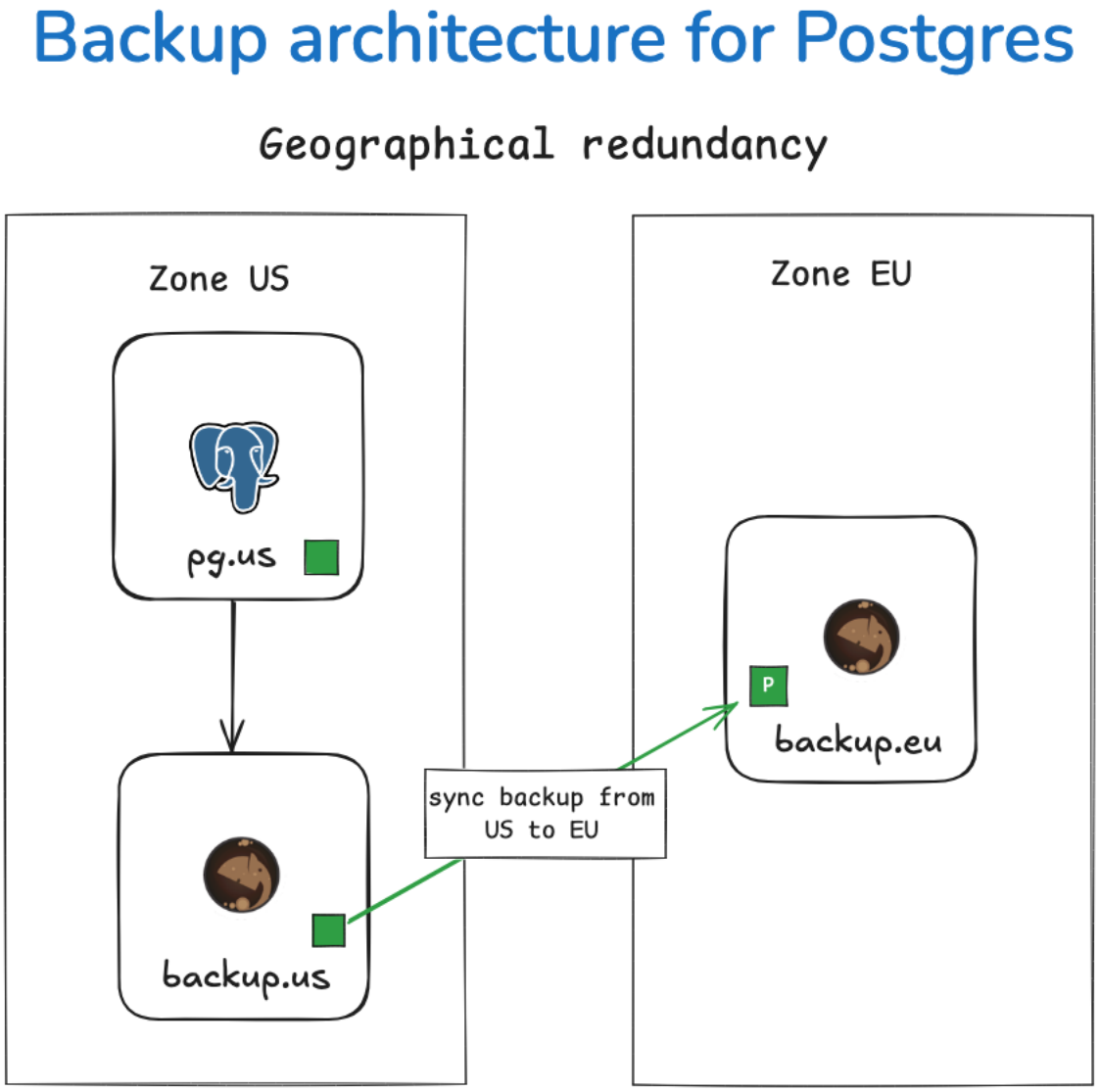
1. 地理的冗長性 Barman 2.6 で導入
2. フックスクリプト

地理的冗長性を使用すると、別のデータセンター/アベイラビリティーゾーンにある別の 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ーゾーン 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。



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 データベースユーザー

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。

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、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。 pghost 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必要があるため、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

pghost 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。

pghost から barmanhost に正常に接続するには、 pghost ユーザ`の公開キーを、 barmanhost の pghost 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、 .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ターにあり、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。

> **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。

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。

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 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
```

7.2.1 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。

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。

7.2.2 クラウドへのストリーミングバックアップ

クラウドストレージ URL に対して `backup_method = postgres` 、 `basebackups_directory` 、 `wals_directory` で構成されている場合、 Barman はクラウドストレージへの直接のストリーミングバックアップをサポートします。このモードでは、バックアップは Postgres サーバーから `pg_basebackup` を介して Barman サーバーにストリーミングされ、完全なバックアップをローカルに保存することなく、チャンクでクラウドに動的にプッシュされます。 WAL ファイルは Postgres サーバーから Barman サーバーにストリーミングされ、アーカイブがトリガーされるたびにクラウドにアップロードされます。これは定期的に、または `barman archive-wal` コマンドを手動で実行したときに発生します。

これは、現在、クラウド環境でブロックレベルのインクリメンタルバックアップを有効にする唯一の方法です。

この機能を構成するには、 `backup_method = postgres` 、 `basebackups_directory` 、および `wals_directory` をクラウドストレージ URL に設定します。例、S3 互換ストレージを使用する場合

```
backup_method = postgres
basebackups_directory = s3://mybucket/barman/
wals_directory = s3://mybucket/barman/
```

構成が完了すると、データをクラウドに安全に保存しながら、インクリメンタルバックアップ、保持ポリシーなどを含む Barman サーバーのすべての機能を活用できます。

注釈

`basebackups_directory` と `wals_directory` に異なる URL を設定することは可能ですが、両方のオプションに同じ URL、場合によっては同じパスを使用することを強くお勧めします。これにより、Barman が各サーバーのデータを専用のサブディレクトリに整理するため、クラウドでの一貫したストレージ構造が保証されます。

警告

アクティブなサーバーで `basebackups_directory` および/または `wals_directory` の値を変更することは非推奨です。サーバーを非アクティブ化し、既存のデータを新しい場所に移動し、再アクティブ化

することにより、注意して行ったとしても、データの損失または構成ミスの潜在的なリスクがあります。

代わりに、安全なオプションは、 `basebackups_directory` または `wals_directory` の新しい値で新しいサーバー構成を作成し、新しいサーバーに新しい完全なバックアップを実行し、新しいサーバーが完全に動作したら、古いサーバーを廃止することです。

この機能を使用する場合、 Barman はクラウドにアップロードする前にチャンクを処理するローカルステージングスペースを必要とします。使用できるステージング領域の量とその場所は、オプション `cloud_staging_max_size` および `cloud_staging_directory` で構成でき、または実行時に `--cloud-staging-max-size` および `--cloud-staging-directory` CLI オプションを使用してオーバーライドできます。

これは実験的な機能です。このため、いくつかの制限が適用されます。

1. 現在、S3 互換ストレージのみが宛先としてサポートされています。
2. バックアップと WAL の暗号化はサポートされていません。
3. バックアップの圧縮はサポートされていません。 WAL 圧縮は、圧縮方法として `pigz` または `custom` を使用する場合を除きサポートされます。
4. `verify-backup` 、 `generate-manifest` 、 `rebuild-xlogdb` 、 `get-wal` などのバックアップまたは WAL ファイルへのアクセスを必要とする Barman サブコマンドはサポートされておらず、実行すると失敗します。
5. Barman **地理的な冗長性** 機能とその関連コマンドはサポートされていません。
6. 今のところ、この機能は Linux ベースのディストリビューションでのみ動作し、BSD 派生システム FreeBSD や OpenBSD などではサポートされていません。

7.3 Rsync 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、特定の Postgres サーバーの Barman 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。

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。

7.3.1 Rsync を使用した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。

7.4 ローカル Rsync 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ライアンスにあります。

これらの場合、 Barman は local-rsync と呼ばれるローカルバックアップ方法をサポートしています。この方法は、リモート rsync 方法と同じ rsync ベースのバックアップアプローチを使用しますが、トランスポートレイヤーの SSH に依存しません。代わりに、Postgres が実行されているのと同じサーバーでローカルにファイルコピー操作を実行します。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。

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
- Barman のホーム構成。

ローカルバックアップ用に Barman をセットアップすることをお勧めします。 Barman のインストール後に次の手順に従うことです。

1. グローバル構成ファイルで barman_user = postgres または別のサーバーを使用している場合は、サーバーを実行しているオペレーティングシステムユーザーを設定して、 Barman がデータファイルを読み取るためのアクセスを保証します。
2. バックアップディレクトリとその他の関連ディレクトリを同じユーザーがアクセスできるように構成します。これは、構成ファイルたとえば backup_directory 、 barman_home 、 log_filename などコミュニティパッケージが提供するデフォルトのパスは、 barman ユーザーが所有しており、これらのディレクトリの権限と所有権はパッケージの更新中に元の状態に復元され、問題が発生する可能性があるため、使用しないでください。

詳細については、 *Configuration options* 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。

7.5 ローカルからクラウドへのバックアップ

Barman は、 local-to-cloud バックアップ方法を使用してクラウドオブジェクトストレージに直接バックアップを実行できます。この方法は、ローカルファイルシステム上の Postgres PGDATA ディレクトリからデータを直接読み取り、バックアップデータ用の中間ローカルストレージを必要とせずに、クラウドストレージに直接アップロードします。

> **注釈**

現在、**Amazon S3** および S3 互換ストレージのみがサポートされています。 Azure Blob Storage と Google Cloud Storage のサポートは、 Barman の将来のバージョンで追加されます。

他のバックアップ方法と同様に、このアプローチはデータベースサーバーとの調整に Postgres の低レベルバックアップ API を使用しますが、バックアップデータはディスクから読み取られるときにクラウドに直接アップロードします。

重要

`local-to-cloud` バックアップ方法には以下が必要です。

- Postgres 9.6 以降同時バックアップサポートの場合。
- Barman は、PGDATA ディレクトリへの直接ファイルシステムアクセスを持っている必要があります。
- Barman を実行するユーザーは、データベースクラスターの所有者である必要があります。
- 適切に構成された AWS 資格情報または S3 互換ストレージの資格情報。
- 適切に構成された S3 または S3 互換オブジェクトストレージバケットへのネットワークアクセス。

ローカルからクラウドへのバックアップを構成するには、 `backup_method` を `local-to-cloud` に設定し、クラウドストレージパス URL を構成します。

```
backup_method = local-to-cloud
basebackups_directory = s3://bucket-name
wals_directory = s3://bucket-name
conninfo = host=localhost dbname=postgres user=postgres
```

`basebackups_directory` は、S3 ストレージ URL たとえば `s3://bucket/path` である必要があります。同様に、 `wals_directory` は、WAL アーカイブの S3 保存場所を指す必要があります。

注釈

Barman は各サーバーのデータを専用のサブディレクトリに整理するため、クラウドで一貫したストレージ構造を維持するために、 `basebackups_directory` と `wals_directory` の両方に同じバケットとパスを使用することをお勧めします。

警告

アクティブなサーバーで `basebackups_directory` および/または `wals_directory` の値を変更することは非推奨です。サーバーを非アクティブ化し、既存のデータを新しい場所に移動し、再アクティブ化することにより、注意して行ったとしても、データの損失または構成ミスの潜在的なリスクがあります。

代わりに、安全なオプションは、 `basebackups_directory` または `wals_directory` の新しい値で新しいサーバー構成を作成し、新しいサーバーに新しい完全なバックアップを実行し、新しいサーバーが

完全に動作したら、古いサーバーを廃止することです。

7.5.1 AWS S3 構成オプション

`local-to-cloud` バックアップ方法は、さまざまな AWS S3 構成オプションをサポートしています。

Bandwidth Control:

- `bandwidth_limit` クラウドアップロードの 1 秒あたりのキロバイト単位の最大転送速度デフォルト `0` 制限なしを意味します。

AWS S3 Options:

- `aws_region` S3 バケットの AWS リージョン。
- `aws_profile` 認証用の AWS プロファイル名。
- `aws_encryption` S3 暗号化方法 `AES256` または `aws:kms` 。
- `aws_sse_kms_key_id` サーバーサイドの暗号化のための KMS キー ID。
- `aws_read_timeout` 秒単位の S3 読み取りタイムアウト。

Cloud Options:

- `cloud_upload_max_archive_size` クラウドアーカイブごとの最大サイズデフォルト `100G` 。
- `cloud_upload_min_chunk_size` マルチパートアップロードの最小チャンクサイズ。
- `cloud_delete_batch_size` 1 回のバッチ操作で削除するファイルの数。

Compression:

- `compression` WAL 圧縮アルゴリズム `gzip` 、 `bzip2` 、 `xz` 、 `snappy` 、 `zstd` 、 `lz4` 、または `none` 。
- `compression_level` 圧縮レベル-選択したアルゴリズムに応じて、 `low` 、 `medium` 、 `high` 、または数値に設定できます。

圧縮設定の詳細については、以下の バックアップ圧縮 セクションを参照してください。

7.5.2 Local-to-Cloud を使用する場合

`local-to-cloud` バックアップ方法は、次の場合に適切です。

- Postgres と Barman は同じマシンで実行されるか、 Barman は PGDATA ディレクトリに直接ローカルファイルシステムにアクセスします。
- バックアップデータ自分自身用の中間ローカルストレージを使用せずに、AWS S3 または S3 互換ストレージに直接バックアップしたい。
- メインの `barman` CLI を介したシンプルで統合されたクラウドバックアップ構成が必要です。

7.5.3 制限事項

`local-to-cloud` バックアップ方法には次の制限があります。

- **does not** は、インクリメンタル バックアップをサポート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をサポートする Postgres 17 以降の `postgres` 方法とは異なります。
- **does not** は、ハードリンクを介したファイルレベルの重複排除をサポートします `reuse_backup` を使用した `rsync` とは異なります。
- **Only AWS S3 and S3-compatible storage are currently supported** - Azure Blob Storage および Google Cloud Storage のサポートは、将来のバージョンで追加されます。
- PGDATA へのローカルファイルシステムのアクセスが必要ですリモートサーバーの `rsync` または `postgres` メソッドとは異なります。

7.5.4 構成例

次に、AWS S3 を使用したローカルからクラウドへのバックアップの完全な構成例を示します。

```
[myserver]
backup_method = local-to-cloud
barman_user = postgres
basebackups_directory = s3://my-backup-bucket
wals_directory = s3://my-backup-bucket
conninfo = host=localhost dbname=postgres user=postgres
bandwidth_limit = 5120  ; 5 MB/s
compression = gzip
aws_region = us-west-2
aws_profile = barman
archiver = on
```

この構成では、 Barman はローカルの PGDATA ディレクトリからデータを直接読み取り、WAL ファイルの帯域幅調整と gzip 圧縮を使用して指定された S3 バケットにアップロードすることによりバックアップを実行します。

7.5.5 ローカルからクラウドへの WAL アーカイブ

`local-to-cloud` バックアップ方法を使用する場合、*barmancloud-wal-archive* **`コマンドを使用して** *WAL* **ファイルをクラウドにアーカイブできます。これは、** `*archive_command*` を介して構成で指定されたクラウドストレージに WAL ファイルをアップロードします。

```
archive_command = 'barman cloud-wal-archive myserver %p'
```

WAL 生成が多い期間に WAL アーカイブバックログを削減するために、 `--parallel` フラグまたは `cloud_wal_archive_parallel` 構成オプションを使用して追加の WAL ファイルを同時にアップロードできます。

```
archive_command = 'barman cloud-wal-archive --parallel 2 myserver %p'
```

`--parallel N` が設定されている場合 N > 1、プライマリ WAL が正常にアーカイブされた後、 `pg_wal/archive_status` で準備ができている最大 `N - 1` 追加の WAL ファイルがバックグラウンドプロセスでアップロードされます。

7.6 一般的なバックアップ設定

7.6.1 インクリメンタル バックアップ

インクリメンタル バックアップには、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 Postgres 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。

7.6.2 スタンバイサーバーからのバックアップ

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。

> **注釈**
>
> Postgres クラスター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。

> **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、 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。

7.6.3 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を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 、 postgres および local-to-cloud 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。

7.6.4 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。

7.6.5 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。

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。

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。

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。

圧縮場所

Postgres 15 以降では、圧縮を発生する場所 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。

backup_compression	backup_compression_	Postgres	Barman
gzip	プレーン	タール	なし
gzip	タール	タール	タール
lz4	プレーン	tar、lz4	なし
lz4	タール	tar、lz4	tar、lz4
zstd	プレーン	tar、zstd	なし
zstd	タール	tar、zstd	tar、zstd
なし	タール	タール	タール

7.6.6 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。

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。

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。

7.6.7 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。

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。

7.6.8 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。

7.6.9 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。

第8章　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する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`

スナップショットバックアップを構成するには、特定の Postgres サーバーの Barman 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
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。

8.1 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。

8.2 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ナルであり、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
```

8.3 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ナルであり、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
```

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
```

8.4 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ナルであり、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_await_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_await_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背後にある概念については、 Amazon EBS snapshot lock concepts を参照してください。

8.5 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。

8.5.1 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

  Base backup information:
    Backup Method        : snapshot-concurrent
    Backup Size          : 1.0 KiB (16.0 MiB with WALs)
    WAL Size             : 16.0 MiB
    Timeline             : 1
```

(次のページに続く)

(前のページからの続き)

```
  Begin WAL            : 000000010000000000000001A
  End WAL              : 000000010000000000000001A
  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       : 000000010000000000000001B

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",
          "snapshot_name": "barman-av-ubuntu20-primary-pgdata-20230123t131430",
          "snapshot_project": "project_id"
        }
      },
      {
        "mount": {
```

(次のページに続く)

(前のページからの続き)

```
          "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9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。

9.1 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

> **注釈**
>
> クラウド WAL ストレージ `wals_directory` がクラウドオブジェクトストアをポイントしている場合は、インメモリ圧縮アルゴリズム `gzip`、`bzip2`、`xz`、`zstd`、`lz4`、`snappy` のみをサポートしています。`pigz` および `custom` は、外部プロセスに依存し、インメモリ圧縮できないため、サポートされていません。

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 のシッピングが遅延し、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。

9.2 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。

9.3 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`ゼロでシナリオを構成するには、 Barman サーバー上の構成でストリーミング接続を介して WAL をアーカイブし、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 /"Syncronous 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: 0/9000098
  Number of streaming clients: 1

  1. #1 Sync WAL streamer
    Application name: barman_receive_wal
    Sync stage      : 3/3 Remote write
    Communication   : TCP/IP
    IP Address      : 139.59.135.32 / Port: 58262 / Host: -
    User name       : streaming_barman
    Current state   : streaming (sync)
    Replication slot: barman
    WAL sender PID  : 2501
    Started at      : 2016-09-16 10:33:01.725883+00:00
    Sent location   : 0/9000098 (diff: 0 B)
    Write location  : 0/9000098 (diff: 0 B)
    Flush location  : 0/9000098 (diff: 0 B)
```

第10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。

10.1 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。

10.2 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。

> **注釈**
>
> commands-barman-import-backup を使用して以前にエクスポートした tar ボールをインポートすることにより、バックアップをカタログに追加することもできます。

10.2.1 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` 。

10.2.2 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
```

(次のページに続く)

(前のページからの続き)

```
PostgreSQL Version     : 170000
PGDATA directory       : /var/lib/pgsql/17/data
Estimated Cluster Size : 244.7 GiB

Server information:
  Checksums            : on
  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            : 0000000100000CFD000000AD
  End WAL              : 0000000100000D0D00000008
  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       : 0000000100000D95000000E7

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11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。

11.1 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。

11.2 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。

11.3 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。

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。

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。

11.4 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。

11.5 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。

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

11.6 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。

11.6.1 リカバリー中の部分的な WAL ファイル

--no-get-wal を使用する場合、 Barman はオプションで --partial-wal フラグを指定して、 .partial WAL ファイルを復旧先にコピーできます。 .partial WAL ファイルは、Postgres によって部分的に書き込まれましたが、まだ完了およびアーカイブされていない WAL セグメントです。

```
barman restore SERVER_NAME BACKUP_ID DESTINATION_PATH --no-get-wal --partial-wal
```

リカバリーに .partial ファイルを含めると、最後に完了した WAL セグメント以降に書き込まれたトランザクションがリカバリー中に利用できるようになるため、RPO 目標リカバリーポイントが削減されます。

> **注釈**
>
> --get-wal を使用する場合、 .partial ファイルはリカバリー中に既に透過的に処理されています。 barman-wal-restore リモートと barman get-wal ローカルはどちらも .partial ファイルをネイティブにサポートし、 --partial-wal フラグを必要としません。

Barman は、2 つのソースからの .partial ファイルを検討します。

- **Streaming WALs directory** streaming_wals_directory pg_receivewal によってアクティブに書き込まれているファイル。最後にアーカイブされた WAL の直後のセグメントのみが考慮されます。ディレクトリ内の他の .partial ファイルは無視されます。これにより、アーカイブされた WAL と部分セグメント間の連続性が保証されます。このソースからのファイルは常にプレーンであり、解凍や復号化は必要ありません。
- **Main WAL archive** wals/ 接尾辞はそのままでアーカイブされた .partial ファイル — 通常、スタンバイが昇格し、PostgreSQL が進行中のセグメントで archive_command を呼び出したとき。これらのファイルは、リカバリに必要な通常の WAL 範囲の一部として生成されます。 barman cron がそれらを圧縮または暗号化している場合、それらは宛先に書き込まれる前に解凍および/または復号化されます。

次の追加ルールが両方のソースに適用されます。

- 予想されるセグメント名がターゲットタイムラインをエンコードするため、別のタイムラインにある .partial ファイルは自動的に除外されます。
- --target-lsn を使用する場合、セグメント名がターゲット LSN を含む WAL セグメント以下の .partial ファイルのみが含まれます。そのポイントを超えるセグメントはスキップされます。
- --target-immediate を使用する場合、リカバリは最初の一貫した状態で停止するため、 .partial ファイルは完全にスキップされます。
- .partial サフィックスは、PostgreSQL がプレーンセグメント名でファイルを見つけられるように、復旧先のすべてのファイル名から削除されます。

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。

11.6.2 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 *
```

11.6.3 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。

11.6.4 クラウドリカバリーに cloud-wal-restore を使用する

クラウドバックアップから復元する場合、 backup_method = local-to-cloud または backup_method = postgres で取得したかどうかにかかわらず、 barman cloud-wal-restore を restore_command として使用して、 Barman サーバーをバイパスして、構成済みのクラウドプロバイダーから直接 WAL ファイルを取得できます。

注釈

--no-get-wal モードデフォルトでのクラウドバックアップの復元はサポートされていません。したがって、クラウドに保存されている WAL を復元する唯一の方法は、cloud-wal-restore を restore_command として使用することです。クラウドバックアップに barman restore で --no-get-wal を使用しようとすると、無視され警告されます。

クラウドリカバリーの restore_command の例は次のとおりです。

```
restore_command = 'barman cloud-wal-restore --parallel 5 SERVER_NAME %f %p'
```

言うまでもなく、 wals_directory は、WAL ファイルが保存されるクラウドの場所、たとえば wals_directory = s3://barman/wals をポイントする必要があります。

11.7 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、パスワードボールト、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 `encrytion_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。

11.8 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。

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。

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。

11.9.1 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。

11.10 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。

11.11 クラウドからのバックアップの復元

クラウドバックアップの復元は、backup_method = local-to-cloud または backup_method = postgres で作成されたかにかかわらず、ローカルバックアップを復元するのと同じプロセス、単純な barman restore コマンドに従います。主な違いは、以下のセクションで説明するように、いくつかの代替復元フローを提供することです。

クラウド WAL 復元の場合、 クラウドリカバリーに *cloud-wal-restore* を使用する セクションを確認します。

11.11.1 新しいサーバーでのクラウドバックアップの復元

クラウドバックアップを復元する最も効率的な方法は、集中化された Barman サーバーをバイパスし、クラスターが実行される場所に直接プルすることです。これを行うには、クラスターが起動される宛先ホストで barman restore を直接実行します。

このアプローチでは、 Barman インスタンスは、クラウドバックアップを復元することのみを目的としてインストールおよび構成され、他のタスクには非アクティブです。この Barman インスタンスの最小構成は次のようになります。

```
[pg]
description = "Minimal configuration for cloud backup restore"
wals_directory = s3://barman/backups
basebackups_directory = s3://barman/backups
active = off
```

注釈

basebackups_directory および wals_directory は、それぞれバックアップと WAL ファイルが保存されている既存のクラウドの場所を指す必要があります。

重要

active = off を設定して、この Barman インスタンスがデータに干渉する可能性のあるメンテナンスタスクを実行しないようにする必要があります。この設定がないと、 Barman はクラウドでのバックアップの検索を拒否します。

11.11.2 同じ Barman サーバーからのクラウドバックアップの復元

クラウドバックアップを復元する最も簡単な方法は、バックアップをトリガーしたのと同じ Barman サーバーから復元をトリガーすることです。このアプローチは、実行するのと同じくらい簡単です。

```
barman restore SERVER_NAME BACKUP_ID /path/to/destination
```

当然、これは、バックアップをトリガーした Barman インスタンスがインストールされているホストでクラスターを起動する場合にのみ役立ちます。

別のホストに復元するには、 --remote-ssh-command オプションを使用し、SSH 経由でリモートコピーを実行できます。

```
barman restore SERVER_NAME BACKUP_ID /path/to/destination --remote-ssh-command "ssh␣
↪postgres@target-server"
```

警告

このアプローチでは、 Barman サーバーを介してバックアップデータをルーティングします。バックアップは、最初にクラウドから Barman ホストにプルされ、次にターゲットホストに転送されます。これにより、 Barman サーバーでの復元時間の増加と帯域幅の使用量が増加する可能性があります。

11.12 .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。 .partial WAL セグメントに記録されたトランザクションもリカバリーするには、 --partial-wal フラグを使用します。詳細については、 --no-get-wal を参照してください。

11.13 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。

11.14 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。

11.14.1 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。

11.14.2 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ー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。

11.14.3 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ーゾーン。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12章　リテンション ポリシー

12.1 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。

12.2 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。

12.2.1 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。

12.2.2 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。

12.2.3 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。

12.3 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。

12.3.1 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。

12.3.2 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。

12.3.3 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。

12.4 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。

12.5 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。

12.5.1 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。

12.5.2 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。

12.5.3 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と、 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.

12.6 ユースケース

12.6.1 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。

12.6.2 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。

12.6.3 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。

12.7 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。

12.8 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。

12.9 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。

12.10 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用の 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13章　バックアップのエクスポートとインポート

13.1 概要

Barman は、既存のバックアップを単一の自己完結型の tar ボールとしてエクスポートし、後で Barman カタログにインポートして戻すことができます。これにより、外部アーカイブシステムに依存するエンタープライズデータ管理ワークフロー、たとえば、祖父/父/息子戦略の一環として、テープに長期コピーを書き込んだり、ストレージ階層を超えてバックアップを移動したりして、 Barman を橋渡しします。

交換フォーマットは次のとおりです。エクスポートはファイルの tarball です。インポートでは、tarball がディスク上でターゲット Barman サーバーに提示されることのみが必要です。 Barman は、特定の外部製品と統合していません。ディスクからファイルをアーカイブし、後で復元できるシステムは、エクスポートされたバックアップをエンドツーエンドで運ぶことができます。

外部の tar ボールは、オプションでサポートされているアルゴリズム `gzip` 、 `bzip2` または `xz` で圧縮できます。圧縮は外部の tar ボールにのみ適用されます。内部のデータファイルと WAL セグメントは再圧縮されず、既に伝送されている圧縮たとえば、ソースサーバーに構成されている WAL またはバックアップ圧縮などはそのまま保存されます。インポート時に、圧縮形式は tarball から自動的に検出され、解凍は透過的に処理されます。

13.2 制限事項

- `DONE` 状態のフルバックアップのみがエクスポートまたはインポートできます。インクリメンタルバックアップ チェーンは、この機能では現在サポートされていません。
- この機能は、ローカルストレージを使用する Barman サーバーでのみサポートされています。
- エクスポートされた tarball は、ソースクラスターの `systemid` に関連付けられます。別のクラスターへのインポートはサポートされていません。
- 元の tarball ファイル名はアーカイブチェーンを介して保存する必要があり、名前を変更すると完全性チェックが中断されます。
- 暗号化は透過的に保持されます。暗号化されたバックアップはそのままエクスポートおよびインポートされます。

13.3 Tarball レイアウト

エクスポートされた tar ボールには、 Barman がバックアップを再登録および復元するために必要なすべてが含まれています。

```
backup-export-SERVER-BACKUP_ID-TIMESTAMP-CHECKSUM.tar[.gz|.bz2|.xz]
├─── identity.json            # server identity
├─── backup.info              # backup metadata
├─── barman.json              # additional Barman metadata
├─── backup/                  # backup data tree (PGDATA + tablespaces)
├─── wals/<hash>/<wal_name>   # WAL segments required for consistency
└─── xlog.db                  # WAL metadata
```

通常 pg_tblspc/ にあるシンボリックリンクは tarball に保存されません。テーブルスペースデータはプレーンなサブディレクトリとして保存され、シンボリックリンクはインポート中に再構成されます。

13.4 Tarball ファイル名と完全性

ファイル名は次のパターンに従います。

```
backup-export-{SERVER_NAME}-{BACKUP_ID}-{TIMESTAMP}-{CHECKSUM}.tar[.gz|.bz2|.xz]
```

TIMESTAMP は UTC でのエクスポート時間 YYYYMMDDTHHMMSS であり、 CHECKSUM は tar ボール全体の SHA-256 ハッシュの最初の 8 文字です。 barman import-backup が実行されると、ファイルの SHA-256 を再計算し、ファイル名に埋め込まれた値と比較し、不一致の場合にインポートを拒否します。

> **警告**
>
> アーカイブシステムを介して転送する場合、エクスポートされた tarball の名前を変更しないでください。名前を変更すると完全性チェックが中断され、 barman import-backup はそれを拒否します。アーカイブシステムが別の名前でファイルを保存している場合、インポートする前に元の名前に復元します。

13.5 使用法

次のウォークスルーでは、ライフサイクル全体をカバーしています。バックアップのエクスポート、外部アーカイブシステムへの受け渡し、ディスクへの復元、リカバリーのための Barman サーバーへのインポート。

1. ソースサーバーでの **Pick a full backup** は DONE 状態

   ```
   barman list-backups main
   barman show-backup main 20260513T101530
   ```

2. **Export the backup** を既存のディレクトリにコピーします。オプションで外部の tar ボールを --compression {gzip|bzip2|xz} で圧縮し、 --compression-level でレベルを調整します。

   ```
   barman export-backup main 20260513T101530 /var/tmp/barman-exports
   ```

   Barman は、 /var/tmp/barman-exports の下に単一のファイルを書き込みます。その名前には、エクスポートタイムスタンプと完全性チェックサムが埋め込まれます。例

```
backup-export-main-20260513T101530-20260513T134207-9f3a1c20.tar
```

3. **Archive the tarball** を外部バックアップ製品で使用します。ファイルを不透明として扱います。唯一の要件は、元のファイル名がアーカイブチェーンを介して保持されることです。

4. 外部製品を使用して、ターゲット Barman サーバー上の **Restore the tarball back to disk** を、選択したディレクトリにコピーします。

5. **Import the backup** をターゲットカタログに追加します。

```
barman import-backup main \
    /var/tmp/barman-imports/backup-export-main-20260513T101530-20260513T134207-
↪9f3a1c20.tar
```

   インポートはアトミックです。データは basebackups_directory の下のステージングディレクトリに抽出され、エンドツーエンドで検証されて、適切な場所に移動されます。ステップが失敗した場合、ステージングディレクトリと部分的な WAL 変更は自動的にクリーンアップされ、カタログはそのまま残されます。

   成功すると、バックアップには KEEP:STANDALONE の注釈が付けられるため、保持ポリシーによって削除されず、これを可視化するために警告が印刷されます。インポートされたバックアップを再度リテンションに参加させる場合は、 barman keep main BACKUP_ID --release を使用します KEEP:STANDALONE を参照してください。

6. **Verify and restore** 、インポートされたバックアップが完全に使用できることを確認します。

```
barman list-backups main
barman show-backup main 20260513T101530
barman restore main 20260513T101530 /var/lib/postgresql/recovered
```

第14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もあります。

14.1 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: 000000010000003B00000010
    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。

14.2 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
    ...
```

14.3 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/10000148
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/10000148 (diff: 0 B)
    Write LSN  : 3B/10000148 (diff: 0 B)
    Flush LSN  : 3B/10000000 (diff: -328 B)
```

14.4 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15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。

15.1 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 configuration overrides that can be applied to backups of Postgres 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。

> **注釈**
>
> **Comment Handling** 行コメントのみがサポートされています # または ; で始まる行 。インラインコメント同行の構成値の後に表示される # または ; はサポートされておらず、構成値の一部として解析されますが、予期しない動作を発生させる可能性があります。

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。

15.2 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。

15.2.1 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、アーカイバープロセスの 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、アーカイバーは 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、アーカイバー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 | local-to-cloud 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。

> **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 Postgres 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。

> **注釈**

マルチホスト接続文字列たとえば host=primary_db,replica_db および target_session_attrs はサポートされていません。各接続文字列は、単一の特定のホストを指す必要があります。

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。
- gpg 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 = gpg 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。

> **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。

> **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。

> **注釈**
>
> マルチホスト接続文字列はサポートされていません。詳細については、`conninfo` を参照してください。

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` 。

> **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 。

> **注釈**
>
> マルチホスト接続文字列はサポートされていません。詳細については、 conninfo を参照してください。

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。

> **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。

15.2.2 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。

> **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。

> **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。

> **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。

warehousepg_dbid

WarehousePG データベースの整数識別子を指定します。設定されると、この値は WarehousePG 互換性の --target-gp-dbid パラメーターを介して pg_basebackup に渡されます。このオプションは、WarehousePG セグメントをバックアップする場合に必要です。

> **重要**
>
> このオプションは WarehousePG データベースにのみ関連し、標準の PostgreSQL データベースには設定しないでください。
>
> さらに、Barman は、WarehousePG クラスターのバックアップとリストアを処理しません。 WarehousePG クラスターのバックアップおよびリストア操作のオーケストレーションは、ユーザーが外部で管理する必要があります。

> **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_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ファイルの宛先。 クラウドへのストリーミングバックアップ で説明しているように、バックアップをローカルに保存する場合はローカルパス、またはクラウドにストリーミングバックアップを保存する場合はクラウドストレージ URL になります。デフォルト <backup_directory>/base 。

> **警告**
>
> アクティブサーバーで basebackups_directory の値を変更することはお勧めできません。サーバーを非アクティブ化し、既存のデータを新しい場所に移動し、再アクティブ化することにより、注意して行ったとしても、データの損失または構成ミスの潜在的なリスクがあります。
>
> 代わりに、安全なオプションは、 basebackups_directory の新しい値で新しいサーバー構成を作成し、新しいサーバーに新しい完全なバックアップを実行し、新しいサーバーが完全に動作したら、古いサーバーを廃止することです。

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

> **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。

15.2.3 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。

> **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。

> **注釈**
>
> backup_method = snapshot および snapshot_provider = aws の場合、または backup_method が postgres または local-to-cloud および basebackups_directory および/または wals_directory が S3 をポイントしている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

backup_method = snapshot を使用する場合、 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。

S3 を指す backup_method = postgres または local-to-cloud を使用する場合、S3 バケットが存在する AWS リージョンを示します。

> **注釈**
>
> backup_method = snapshot および snapshot_provider = aws の場合、または backup_method が postgres または local-to-cloud および basebackups_directory および/または wals_directory が S3 をポイントしている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_check_object_lock

true に設定されている場合、 Barman は S3 ストレージからベースバックアップファイルを削除する前に S3 オブジェクトロックをチェックします。オブジェクトがロックされている場合、削除は中止されます。デフォルト false 。

> **注釈**
>
> backup_method が postgres または local-to-cloud であり、basebackups_directory が S3 をポイントしている場合にのみサポートされます。非 S3 プロバイダーで有効になっている場合、警告がログに記録され、ロックチェックは実行されません。

スコープグローバル/サーバー/モデル。

aws_encryption

S3 にオブジェクトを保存するときに使用するサーバーサイドの暗号化アルゴリズムを指定します。この値は、クラウドオブジェクトストレージへのすべてのアップロード操作で S3 クライアントに渡されます。

許可されるオプション

- `AES256` S3 管理の暗号化キー SSE-S3 を使用します。
- `aws:kms` AWS Key Management Service 暗号化キー SSE-KMS を使用します。

設定しない場合、バケットの S3 のデフォルトの暗号化設定が使用されます。

> **注釈**
>
> `backup_method` が `postgres` または `local-to-cloud` であり、 `basebackups_directory` および/または `wals_directory` が S3 をポイントしている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_sse_kms_key_id

S3 オブジェクトのサーバーサイドの暗号化に使用する AWS Key Management ServiceKMS キー ID を指定します。このオプションは、 `aws_encryption` が `aws:kms` に設定されている場合にのみ有効です。

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。

`aws_encryption = aws:kms` の使用時に設定されていない場合、S3 は S3 のデフォルトの AWS 管理 KMS キーを使用します。

例

```
[myserver]
backup_method = local-to-cloud
basebackups_directory = s3://my-bucket/barman/backups
aws_encryption = aws:kms
aws_sse_kms_key_id = arn:aws:kms:us-east-1:123456789012:key/abcd1234-5678-90ab-cdef-
↪EXAMPLEKEY
```

> **注釈**
>
> `backup_method` が `postgres` または `local-to-cloud` であり、 `basebackups_directory` および/または `wals_directory` が S3 をポイントしている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_read_timeout

クラウドストレージを使用する場合、S3 操作の読み取りタイムアウトを秒単位で指定します。この値は、クラウドオブジェクトストレージ上のバックアップデータと対話するすべてのコマンドの S3 クライアントに渡されます。設定されていない場合、デフォルトの boto3 読み取りタイムアウト動作が適用されます。

このオプションは、デフォルトのタイムアウトでは十分でない可能性がある高度な非標準の S3 ワークフローに役立ちます。

> **注釈**
>
> backup_method が postgres または local-to-cloud であり、 basebackups_directory および/または wals_directory が S3 をポイントしている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 。

> **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。

> **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。

> **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。

> **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。

> **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。

> **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。

> **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。

cloud_delete_batch_size

クラウドストレージで削除操作を実行するときに、一度に削除するファイルの数を指定します。このオプションは、バックアップと WAL ファイルの削除の両方に関連します。デフォルト値は、S3 の場合は 1000

、Azure Blob の場合は 256 、および GCS の場合は 100 です。

> **注釈**
>
> backup_method が postgres または local-to-cloud であり、 basebackups_directory および/または wals_directory がクラウドストレージをポイントしている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。

cloud_upload_max_archive_size

クラウドのアップロード中に、個々のアーカイブファイルの最大サイズをバイト単位で指定します。バックアップをクラウドストレージにアップロードするとき、Barman はバックアップを複数のアーカイブファイルに分割します。このサイズを超える場合。デフォルト値は 100Gi です。

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。

> **注釈**
>
> backup_method が postgres または local-to-cloud であり、 basebackups_directory がクラウドストレージをポイントしている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。

cloud_upload_min_chunk_size

クラウドストレージへのマルチパートアップロード中に、個々のチャンクの最小サイズをバイト単位で指定します。これにより、大きなファイルをクラウドオブジェクトストレージにアップロードする際の各部分のサイズが制御されます。設定しない場合、クラウドプロバイダーのチャンクサイズがデフォルトになります。S3 の場合は 5Mi 、Azure Blob の場合は 64Ki 。 GCS には適用されません。

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。

> **注釈**
>
> backup_method が postgres または local-to-cloud であり、 basebackups_directory がクラウドストレージをポイントしている場合にのみサポートされます。

cloud_staging_directory

backup_method = postgres を使用してクラウドオブジェクトストレージにバックアップを送信する場合のステージングディレクトリ。これは、バックアップのチャンクをクラウドに送信する前に保存するため

の一時的な場所として使用されます。デフォルト /tmp/barman/cloud-staging 。

スコープグローバル/サーバー/モデル。

cloud_staging_max_size

Barman が新しいバックアップチャンクの生成を停止する前に cloud_staging_directory が成長できる最大サイズ。これは、チャンクのアップロード速度が Postgres からストリーミングの速度に追いつけない場合に、ステージングディレクトリの成長を防ぐために使用されます。デフォルト値は 30Gi です。

パフォーマンスの低下につながる可能性があるため、極端に低い値は推奨されません。少なくとも 10Gi をお勧めします。最適な値は、 Barman へのバックアップの転送速度、および Barman からクラウドへの転送速度、およびバックアップのサイズによって異なります。

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。

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。

> **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ーゾーンを示します。

> **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。

> **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。

> **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 。

> **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。

15.2.4 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。リカバリー後のシナリオでは、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。

15.2.5 ログ先行書き込み WAL

これらの構成オプションは、 Barman が PostreSQL 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。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 ファイルで使用さ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。

cloud_wal_archive_parallel

barman cloud-wal-archive が並列してアップロードする WAL ファイルの数を制御します。 N N > 1 に設定している場合、プライマリ WAL が正常にアーカイブされた後、 pg_wal/archive_status で準備ができている最大 N - 1 追加 WAL ファイルがバックグラウンドプロセスで同時にアップロードされます。デフォルト 0 無効。

> **注釈**
>
> このオプションは、 archive_command として barman cloud-wal-archive を使用して local-to-cloud バックアップ方法を使用する場合にのみ適用されます。

スコープ グローバル/サーバー。

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

アーカイブされた WAL ファイルの宛先。 クラウドへのストリーミングバックアップ で説明しているように、WAL ファイルをローカルに保存する場合はローカルパス、またはクラウドに WAL ファイルを保存する場合はクラウドストレージ URL です。デフォルト <backup_directory>/wals 。

警告

アクティブサーバーで wals_directory の値を変更することはお勧めできません。サーバーを非アクティブ化し、既存のデータを新しい場所に移動し、再アクティブ化することにより、注意して行ったとしても、データの損失または構成ミスの潜在的なリスクがあります。

代わりに、安全なオプションは、 wals_directory の新しい値で新しいサーバー構成を作成し、新しいサーバーに新しい完全なバックアップを実行し、新しいサーバーが完全に動作したら、古いサーバーを廃止することです。

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。

15.2.6 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。

15.2.7 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。

15.3 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。

15.3.1 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。

15.3.2 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。

15.3.3 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 Postgres 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。

15.3.4 特典

- 一貫性 クラスター内の複数の Postgres 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16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。

16.1 barman

16.1.1 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。

16.1.2 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オペレーション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。

16.2 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。

16.3 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 **0** は **0** を意味します。

16.4 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。

16.4.1 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。

16.4.2 barman backup

概要

```
backup
    [ --bwlimit KBPS ]
    [ --cloud-staging-directory PATH ]
    [ --cloud-staging-max-size SIZE ]
    [ { -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

PostreSQL 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。

--cloud-staging-directory
backup_method = postgres を使用してクラウドオブジェクトストレージにバックアップを送信する場合のステージングディレクトリ。これは、バックアップのチャンクをクラウドに送信する前に保存するための一時的な場所として使用されます。デフォルト /tmp/barman/cloud-staging 。

--cloud-staging-max-size
Barman が新しいバックアップチャンクの生成を停止する前に --cloud-staging-directory が成長できる最大サイズ。これは、チャンクのアップロード速度が Postgres からストリーミングの速度に追いつけない場合に、ステージングディレクトリの成長を防ぐために使用されます。デフォルト値は 30Gi です。

パフォーマンスの低下につながる可能性があるため、極端に低い値は推奨されません。少なくとも 10Gi をお勧めします。最適な値は、 Barman へのバックアップの転送速度、および Barman からクラウドへの転送速度、およびバックアップのサイズによって異なります。

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。

-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。

> **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。

> **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。

16.4.3 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。

16.4.4 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。

16.4.5 barman cloud-wal-archive

概要

```
cloud-wal-archive
    [ { -h | --help } ]
    [ { -p | --parallel } PARALLEL ]
    SERVER_NAME WAL_PATH
```

説明

WAL ファイルをローカルディスクから構成済みのクラウドオブジェクトストレージにプッシュします。このコマンドは、ベースバックアップを取得するために `local-to-cloud` バックアップ方法を使用する場合に、Postgres サーバーの `archive_command` で使用することを目的としています。

> **注釈**
>
> WAL ファイルに使用される圧縮アルゴリズムとレベルは、 Barman サーバー構成ファイルで設定できます。指定しない場合、WAL ファイルは圧縮なしでアップロードされます。

パラメーター

SERVER_NAME
: WAL ファイルがアーカイブされる Postgres サーバーの名前。

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。コマンドの使用法に関する情報を提供します。

-p / --parallel
並列してアーカイブする WAL の数。 1 より大きい値に設定すると、プライマリ WAL が正常にアーカイブされた後、 pg_wal/archive_status で準備ができている最大 parallel - 1 追加 WAL ファイルがバックグラウンドプロセスで同時にアップロードされます。これにより、WAL 生成が多い期間に WAL アーカイブのバックログを削減できます。デフォルトは cloud_wal_archive_parallel サーバー構成値であり、自分自身はデフォルトで 0 無効になります。

16.4.6 barman cloud-wal-restore

概要

```
cloud-wal-restore
    [ { -h | --help } ]
    [ { -p | --parallel } PARALLEL ]
    [ --spool-dir SPOOL_DIR ]
    SERVER_NAME WAL_NAME WAL_DEST
```

説明

構成済みのクラウドオブジェクトストレージからローカルディスクに WAL ファイルをプルします。このコマンドは、クラウドオブジェクトストレージから WAL を復元するときに Postgres の restore_command として使用することを目的としています。

このコマンドを使用するには、サーバーにクラウド URL s3://barman/wals などを指す wals_directory を介してクラウドストレージを構成する必要があります。

パラメーター

SERVER_NAME
WAL ファイルが復元される Postgres サーバーの名前。

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。

-p / --parallel
並列してピークおよびダウンロードする WAL ファイルの数を指定します。 1 より大きい値に設定すると、要求された WAL ファイルと次の N - 1 ファイルを同時に取得します。追加ファイルはスプールディレクトリにステージングされるため、後続の復元要求はローカルストレージからすぐに処理できます。デフォルトは 0 無効です。

--spool-dir
--parallel の使用時に取得された追加の WAL をステージングするために使用されるディレクトリ。デフォルトは /var/tmp/walrestore です。

> **Tip**
>
> --parallel を使用し、pg_wal ディレクトリとスプールディレクトリの両方が同じファイルシステム上にある場合、WAL 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、より速く提供されます。

16.4.7 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。

16.4.8 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。

> **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” }]'
```

16.4.9 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。

16.4.10 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。

16.4.11 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。

16.4.12 barman export-backup

概要

```
export-backup
    [ { -h | --help } ]
    [ --compression { gzip | bzip2 | xz } ]
    [ --compression-level COMPRESSION_LEVEL ]
    SERVER_NAME BACKUP_ID OUTPUT_DIRECTORY
```

説明

指定されたバックアップを単一の tar ボールとしてエクスポートし、外部アーカイブシステムへのハンドオフに適しています。 tar ボールは自己完結型です。バックアップデータファイル、テーブルスペース、バックアップを一貫した状態にするために必要な WAL セグメント、および後で commands-barman-import-backup にバックアップを再登録するために必要な Barman メタデータがバンドルされています。

生成されたファイルの名前は、パターン `backup-export-SERVER_NAME-BACKUP_ID-TIMESTAMP-CHECKSUM.tar[.gz|.bz2|.xz]` を使用して名前付けられます。ファイル名は完全性契約の一部です。tarball の名前を変更しないでください。変更しない場合、`backup-export-SERVER_NAME-BACKUP_ID-TIMESTAMP-CHECKSUM.tar[.gz|.bz2|.xz]` はそれを拒否します。

ローカルストレージを使用するサーバー上の `DONE` 状態のフルバックアップのみをエクスポートできます。完全な機能の説明については、 `DONE` 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。

BACKUP_ID
: barman カタログのバックアップの ID。

OUTPUT_DIRECTORY
: エクスポートされた tar ボールが書き込まれる既存のディレクトリ。 Barman を実行しているユーザーが書き込み可能および実行可能である必要があります。

--compression {gzip | bzip2 | xz}
: 指定されたアルゴリズムを使用して、外部の tarball を圧縮します。デフォルトでは、tarball は圧縮されていません。圧縮は外部の tar ボールにのみ適用されます。データファイルと内部に保存されている WAL セグメントは再圧縮されず、既存の WAL 圧縮は、関連する `xlog.db` メタデータを介して保持されます。

--compression-level COMPRESSION_LEVEL
: 選択したアルゴリズムに渡す圧縮レベル。有効なレベルは、`gzip` および `bzip2` の場合は `1` 〜 `9`、`xz` の場合は `0` 〜 `9` です。アルゴリズム自分自身のデフォルトレベルのデフォルト。 `--compression` を設定する必要があります。

> **注釈**
>
> 明示的な `--compression-level` を渡すには、 `gzip` / `bzip2` の場合は Python 3.12 以降、 `xz` の場合は Python 3.14 以降が必要です。古いインタープリターでは、オプションが受け入れられ、警告が生成され、代わりにアルゴリズムのデフォルトレベルが使用されます。

16.4.13 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。

16.4.14 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 ファイルが見つかると、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。

16.4.15 barman import-backup

概要

```
import-backup
    [ { -h | --help } ]
    SERVER_NAME INPUT_TARBALL
```

説明

以前にエクスポートしたバックアップ tarball を SERVER_NAME の Barman カタログに登録し直します。 tarball は、 SERVER_NAME の出力またはこのようなエクスポートから外部アーカイブシステムによってディスクに逐語的に復元されたファイルであることが予想されます。インポートが成功すると、バックアップは完全に使用できるようになります。バックアップは SERVER_NAME に表示され、そのメタデータは SERVER_NAME を介して利用でき、 SERVER_NAME で復元できます。

インポートされたバックアップは、 KEEP:STANDALONE アノテーションを適用することにより、保持から自動的に保護されます KEEP:STANDALONE を参照してください。インポートされたバックアップを再度保持に参加させる場合は、注釈を明示的に削除します。

インポートはアトミックに実行されます。tarball は basebackups_directory の下のステージングディレクトリに抽出され、エンドツーエンドで検証されて、適切な場所に移動されます。ステップが失敗した場合、ステージング領域はクリーンアップされ、カタログと WAL アーカイブはそのまま残されます。完全な機能の説明については、このセクション basebackups_directory 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。

INPUT_TARBALL
: インポートするエクスポート tar ボールへのパス。

16.4.16 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と、 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。

16.4.17 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。

16.4.18 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。

16.4.19 barman list-processes

概要

```
list-processes
    [ { -h | --help } ]
    SERVER_NAME
```

説明

list-processes サブコマンドは、Postgres 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 Postgres 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。

16.4.20 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。

16.4.21 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。

16.4.22 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。

16.4.23 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。

16.4.24 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。

16.4.25 barman receive-wal

概要

```
receive-wal
    [ --create-slot ]
    [ --if-not-exists ]
    [ --drop-slot ]
    [ { -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、この機能を処理する新しい方法です。

16.4.26 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 ]
    [ --partial-wal ]
```

(次のページに続く)

(前のページからの続き)

```
    [ --recovery-conf-filename FILENAME ]
    [ --recovery-staging-path PATH ]
    [ --staging-path STAGING_PATH ]
    [ --staging-location STAGING_LOCATION ]
    [ --combine-mode COMBINE_MODE ]
    [ --remote-ssh-command STRING ]
    [ --restore-command COMMAND ]
    [ --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

PostreSQL 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 ノード自分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。

> **注釈**
> wals_directory がクラウドストレージ S3、Azure、GCS などを使用するように構成されている場合、復元には常に WAL 取得に get-wal が必要です。 --no-get-wal を指定するか、recovery_options から get-wal を省略すると、 Barman は復元オペレーションで get-wal を自動的に有効にし、警告を発行します。

--partial-wal
.partial サフィックスを削除して、.partial WAL ファイルを復旧先にコピーします。 Barman は、2 つのソースを検討します。 streaming_wals_directory の .partial ファイル pg_receivewal によって書き込まれ、とメイン WAL アーカイブの .partial ファイルスタンバイプロモーション中にアーカイブされ、圧縮または暗号化されている場合があります。 --no-get-wal を使用する場合にのみ適用されます。 --get-wal を使用する場合、 .partial ファイルは既に透過的に処理され、このオプションは効果ありません。また、部分的な WAL が必要になる前に、最初の一貫した状態でリカバリが停止するため、 --target-immediate を使用する場合、このフラグは効果がありません。詳細については、 .partial を参照してください。

-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。

--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。

--restore-command
: カスタム restore コマンドを指定して、PostgreSQL リカバリ構成での Barman のデフォルトの restore_command をオーバーライドします。 Barman がビルドしたデフォルトの restore_command は、ほとんどの場合に適していますが、このオプションを使用すると、セットアップで必要な場合に代替コマンドを指定できます。 get-wal モードでリストアを実行する場合にのみ使用できます。

--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、これらのアクショ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ーします。

--target-time
指定した時間にリカバリーします。 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ーします。 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ーします。

--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。

16.4.27 barman replication-status

概要

```
replication-status
    [ { -h | --help } ]
    [ --minimal ]
```

(次のページに続く)

(前のページからの続き)

```
    [ --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。

16.4.28 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。

16.4.29 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。

16.4.30 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。

16.4.31 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。

16.4.32 barman switch-xlog

説明

switch-wal コマンドのエイリアス。

16.4.33 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。

16.4.34 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。

16.4.35 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。

16.4.36 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。

16.4.37 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。

16.4.38 barman verify

説明

verify-backup コマンドのエイリアス。

16.5 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。

16.5.1 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 snamppy 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。

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 のシッピングが遅延し、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。

16.5.2 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 クラスター内での 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
並列して取得する WAL ファイルの数を指定します。デフォルトは 0 無効です。 1 より大きい値に設定すると、要求された WAL ファイルと次の N - 1 ファイルを同時に取得します。追加ファイルはローカルスプールディレクトリ --spool-dir を参照してください。したがって、後続のリストア要求はローカルストレージからすぐに対応できます。

--spool-dir
--parallel の使用時に取得された追加の WAL をステージングするために使用されるディレクトリ。デフォルトは /var/tmp/walrestore です。

-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17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。

17.1 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。

17.2 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
```

17.3 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。

17.4 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18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見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。

18.1 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。

18.2 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。

18.3 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。

18.4 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。

18.5 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。

18.6 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。

警告

barman-cloud-backup をフックスクリプトとして使用する場合、バックアップ名を設定することはできません。圧縮を使用したバックアップで使用することもできません。そうしようとすると、エラーが発生します。

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19章　クラウドのBarman

Barman は、Postgres クラスターをクラウドにバックアップするさまざまな方法を提供します。

- **Using a Barman server**

  Barman サーバーがあることは、構成ファイルを介して Barman をインストールおよび構成したことを意味します。この場合、バックアップと WAL に関するすべてのメタデータはローカルに保存され、メンテナンス操作が定期的に実行されます。

  このモードでは、次のことができます。

  1. バックアップと WAL をローカルに保存する標準の Barman サーバーをセットアップします。次に、 Barman **フックスクリプト** を使用して、バックアップと WAL を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 **フックスクリプト** セクションを参照してください。

  2. バックアップはクラウドで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、メタデータと WAL ファイルをローカルに保存するように Barman サーバーをセットアップします。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。

  3. 完全なバックアップをローカルに保存することなく、Postgres サーバーからバックアップをストリーミングし、クラウドオブジェクトストレージに直接送信する Barman サーバーをセットアップします。 WAL ファイルは Postgres サーバーからストリーミングされ、クラウドに正常にアーカイブされるまで Barman サーバーに一時的に保存されます。これは、現在、クラウドでインクリメンタル バックアップ ブロックレベルでできる唯一の方法です。このアプローチを選択する場合、詳細については **クラウドへのストリーミングバックアップ** セクションを参照してください。

  4. Postgres サーバーで Barman をローカルにセットアップし、 `local-to-cloud` バックアップ方法を使用して、ローカルに保存せずにクラウドに直接バックアップを作成します。 `barman cloud-wal-archive` を `archive_command` として使用して、WAL ファイルをオブジェクトストアに直接アーカイブできます。このアプローチを選択する場合、詳細については `local-to-cloud` セクションを参照してください。

- **Using the cloud scripts, without the need of a Barman server**

  Barman クラウドスクリプトを使用すると、Barman サーバーを必要とせず、`barman-cli-cloud` パッケージが提供するコマンドラインユーティリティを介してバックアップを直接管理および対話できます。この場合、データとメタデータは常にクラウドオブジェクトストレージに保存され、 Barman はローカル状態を保持しません。

  このモードでは、次のことができます。

1. Barman サーバーを使用せずに、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、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。このモードで使用可能なコマンドの詳細については、以下の コマンドリファレンス セクションを確認してください。

2. Barman サーバーを使用せずに、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。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。このモードで使用可能なコマンド、特に `--snapshot-*` オプションの詳細については、以下の コマンドリファレンス セクションを確認してください。

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クラウドスクリプト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。

重要

S3 互換性ステートメント

Barman Cloud は、Boto3 SDK を使用して S3 互換オブジェクトストアと統合します。これにより、Barman Cloud が AWS S3 リファレンス実装で正しく機能することが保証され、S3 互換オブジェクトストアとの Barman Cloud の統合がサポートされますが、 Barman Cloud は、サードパーティベンダーの基になるストレージインフラストラクチャを直接サポートしていません。オペレーションの一貫性が AWS S3 で再現できないサードパーティのオブジェクトストアでのみ発生した場合、これは API の非互換性とみなされます。このような問題は、ユーザーが特定のストレージベンダーと協力して解決する必要があります。

19.1 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。

19.2 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。

19.3 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-ManyWORM 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。

19.3.1 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。

19.4 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` 。

19.4.1 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 | --lz4 } ]
                [ { -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 ]
                [ --tag KEY,VALUE [ --tag KEY,VALUE ... ] ]
                [ --endpoint-url ENDPOINT_URL ]
                [ --addressing-style { auto | virtual | path } ]
                [ { -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 ]
```

(次のページに続く)

(前のページからの続き)

```
                [ --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- 引数が使用され、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。代替の GCP ユニバース S3NS/T-Systems を使用するには、 GOOGLE_CLOUD_UNIVERSE_DOMAIN 環境変数を目的のユニバースドメインに設定します。

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。

--lz4
クラウドへのアップロード中にバックアップを lz4 圧縮しますオプションの lz4 Python 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。

> **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。

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。

--addressing-style
S3 要求に使用するアドレススタイル。これは、特定のアドレススタイルを必要とする S3 互換サービスに接続する場合に役立ちます。

許可されるオプションは次のとおりです。

- auto デフォルト 基になるライブラリによって決定されたアドレス指定スタイルを使用します。

- virtual 仮想ホストスタイルのアドレス指定たとえば、bucket-name.s3.amazonaws.com などを使用します。
- path パススタイルのアドレス指定たとえば s3.amazonaws.com/bucket-name を使用します。

-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。

19.4.2 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 ]
                [ --addressing-style { auto | virtual | path } ]
                [ { -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。代替の GCP ユニバース S3NS/T-Systems を使用するには、 GOOGLE_CLOUD_UNIVERSE_DOMAIN 環境変数を目的のユニバースドメインに設定します。

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場合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。

--addressing-style
S3 要求に使用するアドレススタイル。これは、特定のアドレススタイルを必要とする S3 互換サービスに接続する場合に役立ちます。

許可されるオプションは次のとおりです。

- `auto` デフォルト 基になるライブラリによって決定されたアドレス指定スタイルを使用します。
- `virtual` 仮想ホストスタイルのアドレス指定たとえば、bucket-name.s3.amazonaws.com などを使用します。
- `path` パススタイルのアドレス指定たとえば s3.amazonaws.com/bucket-name を使用します。

-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 。

19.4.3 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 ]
                [ --addressing-style { auto | virtual | path } ]
                [ { --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。代替の GCP ユニバース S3NS/T-Systems を使用するには、 GOOGLE_CLOUD_UNIVERSE_DOMAIN 環境変数を目的のユニバースドメインに設定します。

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。

--addressing-style
: S3 互換のクラウドストレージに接続するときに使用する S3 アドレススタイル。このオプションは、特定のアドレス指定スタイルを必要とするカスタム S3 エンドポイント MinIO などを使用する場合に役立ちます。

  許可されるオプション

  - auto デフォルト boto3 ライブラリで決定されたアドレススタイルを使用します。
  - virtual 仮想ホストスタイルのアドレス指定 bucket-name.s3.amazonaws.com を使用します。
  - path パススタイルのアドレス指定 s3.amazonaws.com/bucket-name を使用します。

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 。

19.4.4 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 ]
                [ --addressing-style { auto | virtual | path } ]
```

(次のページに続く)

(前のページからの続き)

```
                [ { --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。

あらゆる状態のバックアップがリストに含まれます。完了したバックアップステータス DONE は、終了時刻と、該当する場合はアーカイブ保持ターゲット KEEP:FULL などを示します。進行中または失敗したバックアップは、 **アーカイブステータス** 列にステータスたとえば STARTED と空の **終了時間** を表示します。

> **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。代替の GCP ユニバース S3NS/T-Systems を使用するには、 GOOGLE_CLOUD_UNIVERSE_DOMAIN 環境変数を目的のユニバースドメインに設定します。

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。

--addressing-style
S3 互換のクラウドストレージに接続するときに使用する S3 アドレススタイル。このオプションは、特定のアドレス指定スタイルを必要とするカスタム S3 エンドポイント MinIO などを使用する場合に役立ちます。

許可されるオプション

- auto デフォルト boto3 ライブラリで決定されたアドレススタイルを使用します。
- virtual 仮想ホストスタイルのアドレス指定 bucket-name.s3.amazonaws.com を使用します。
- path パススタイルのアドレス指定 s3.amazonaws.com/bucket-name を使用します。

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 。

19.4.5 barman-cloud-backup-keep

Synopsis

```
barman-cloud-backup-keep
                [ { -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 ]
                [ --addressing-style { auto | virtual | path } ]
                [ { --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。代替の GCP ユニバース S3NS/T-Systems を使用するには、 `GOOGLE_CLOUD_UNIVERSE_DOMAIN` 環境変数を目的のユニバースドメインに設定します。

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。

--addressing-style
: S3 互換のクラウドストレージに接続するときに使用する S3 アドレススタイル。このオプションは、特定のアドレス指定スタイルを必要とするカスタム S3 エンドポイント MinIO などを使用する場合に役立ちます。

  許可されるオプション

  - auto デフォルト boto3 ライブラリで決定されたアドレススタイルを使用します。
  - virtual 仮想ホストスタイルのアドレス指定 bucket-name.s3.amazonaws.com を使用します。
  - path パススタイルのアドレス指定 s3.amazonaws.com/bucket-name を使用します。

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 。

19.4.6 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 ]
                [ --addressing-style { auto | virtual | path } ]
```

(次のページに続く)

(前のページからの続き)

```
                [ { --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。

> **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。代替の GCP ユニバース S3NS/T-Systems を使用するには、 GOOGLE_CLOUD_UNIVERSE_DOMAIN 環境変数を目的のユニバースドメインに設定します。

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。

--addressing-style
S3 互換のクラウドストレージに接続するときに使用する S3 アドレススタイル。このオプションは、特定のアドレス指定スタイルを必要とするカスタム S3 エンドポイント MinIO などを使用する場合に役立ちます。

許可されるオプション

- auto デフォルト boto3 ライブラリで決定されたアドレススタイルを使用します。
- virtual 仮想ホストスタイルのアドレス指定 bucket-name.s3.amazonaws.com を使用します。
- path パススタイルのアドレス指定 s3.amazonaws.com/bucket-name を使用します。

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 。

19.4.7 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 ]
                [ --addressing-style { auto | virtual | path } ]
                [ { -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。

> **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。代替の GCP ユニバース S3NS/T-Systems を使用するには、 GOOGLE_CLOUD_UNIVERSE_DOMAIN 環境変数を目的のユニバースドメインに設定します。

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。

`--addressing-style`
S3 要求に使用するアドレススタイル。これは、特定のアドレススタイルを必要とする S3 互換サービスに接続する場合に役立ちます。

許可されるオプションは次のとおりです。

- `auto` デフォルト 基になるライブラリによって決定されたアドレス指定スタイルを使用します。
- `virtual` 仮想ホストスタイルのアドレス指定たとえば、bucket-name.s3.amazonaws.com などを使用します。
- `path` パススタイルのアドレス指定たとえば s3.amazonaws.com/bucket-name を使用します。

`-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` 。

`--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。

19.4.8 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 ]
                [ --addressing-style { auto | virtual | path } ]
                [ { -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。代替の GCP ユニバース S3NS/T-Systems を使用するには、 `GOOGLE_CLOUD_UNIVERSE_DOMAIN` 環境変数を目的のユニバースドメインに設定します。

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。

> **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。

> **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。

--addressing-style
S3 要求に使用するアドレススタイル。これは、特定のアドレススタイルを必要とする S3 互換サービスに接続する場合に役立ちます。

許可されるオプションは次のとおりです。

- auto デフォルト 基になるライブラリによって決定されたアドレス指定スタイルを使用します。
- virtual 仮想ホストスタイルのアドレス指定たとえば、bucket-name.s3.amazonaws.com などを使用します。
- path パススタイルのアドレス指定たとえば s3.amazonaws.com/bucket-name を使用します。

-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。

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。

19.4.9 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 ]
                [ --addressing-style { auto | virtual | path } ]
                [ { -P | --aws-profile } AWS_PROFILE ]
                [ --profile AWS_PROFILE ]
                [ --read-timeout READ_TIMEOUT ]
                [ { --azure-credential | --credential } { azure-cli | managed-identity
                  | default } ]
                [ --no-partial ]
                [ { -p, --parallel } PARALLEL ]
                [ --spool-dir SPOOL_DIR ]
                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。

注釈

スタンバイサーバーが WAL セグメントをアーカイブし、プライマリに昇格すると、部分的な WAL ファイル .partial サフィックスがオブジェクトストアに表示される場合があります。プロモーションの時点で、進行中の WAL セグメントはまだ完了していないため、Postgres は新しいタイムラインに切り替える前に .partial ファイルとしてアーカイブします。これらのファイルは不完全なセグメントを表しており、単独でのクラッシュリカバリーには適していません。

Postgres はそれらを正しく処理する方法を知っているため、WAL 復元中に .partial ファイルをダウンロードすることは無害です。ただし、プロモーションが発生したクラスターで barman-cloud-wal-restore を実行している場合、 restore_command は、完全な WAL が予想される .partial サフィックスなしで名前を変更した後ファイルを取得する場合があります。

このシナリオでは --no-partial を使用して .partial ファイルのダウンロードを抑制します。これにより、 restore_command は、WAL が利用できないことを Postgres に通知し、代わりにタイムライン履歴ファイル .history を介して正しいタイムラインに切り替えるように求めます。

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。代替の GCP ユニバース S3NS/T-Systems を使用するには、 GOOGLE_CLOUD_UNIVERSE_DOMAIN 環境変数を目的のユニバースドメインに設定します。

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

-p / --parallel
並列して取得する WAL ファイルの数を指定します。デフォルトは 0 無効です。 1 より大きい値に設定すると、要求された WAL ファイルと次の N - 1 ファイルを同時に取得します。追加ファイルはローカルスプールディレクトリ --spool-dir を参照してください。したがって、後続のリストア要求はローカルストレージからすぐに対応できます。

--spool-dir
--parallel の使用時に取得された追加の WAL をステージングするために使用されるディレクトリ。デフォルトは /var/tmp/walrestore です。

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。

--addressing-style
S3 要求に使用するアドレススタイル。これは、特定のアドレススタイルを必要とする S3 互換サービスに接続する場合に役立ちます。

許可されるオプションは次のとおりです。

- auto デフォルト 基になるライブラリによって決定されたアドレス指定スタイルを使用します。
- virtual 仮想ホストスタイルのアドレス指定たとえば、bucket-name.s3.amazonaws.com などを使用します。
- path パススタイルのアドレス指定たとえば s3.amazonaws.com/bucket-name を使用します。

-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。

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/ローカルの 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

第22章　Barmanリリースノート

© Copyright EnterpriseDB UK Limited 2025 - 無断複写・転載を禁じます。

22.1 3.19.02026-05-20

22.1.1 重要な変更

- クラウドストレージからのバックアップ復元のサポートを追加します

  Barman は、クラウドオブジェクトストレージに保存されている場合、 `backup_method = local-to-cloud` または `backup_method = postgres` で取得されたバックアップを復元できるようになりました。

  標準の `barman restore` コマンドを使用して、このようなバックアップをリストアできます。

  WAL リストアの場合、リカバリ中にクラウドストレージから WAL ファイルを直接取得するための新しい `barman cloud-wal-restore` コマンドが導入されました。このコマンドは、PostgreSQL の `restore_command` として使用でき、パフォーマンスの向上のための並列取得をサポートしています。

  また、サーバーがクラウドストレージを使用するように `wals_directory` が構成されている場合、リストア操作に常に `get-wal` が必要になるようになりました。 `--no-get-wal` が指定されている場合、または `get-wal` が `recovery_options` にない場合、 Barman は自動的に `get-wal` を有効にし、警告を発行して、リストアを中断せずに続行できるようにします。

  これにより、クラウドバックアップのライフサイクルが完了し、完全にクラウドでのシームレスなバックアップとリカバリーのワークフローが可能になります。

  参考文献 BAR-929。

- バックアップを tarball にエクスポートするための新しい barman allow-backup コマンドを追加

  バックアップを tarball 形式にエクスポートする新しい `barman export-backup` コマンドを導入しました。このコマンドは、サーバー名、バックアップ ID、および宛先ディレクトリを引数として受け取り、バックアップデータを含む tar ボールを作成します。これにより、バックアップを共有またはアーカイブする簡単な方法を提供します。

  参考文献 BAR-1180。

- tarball からバックアップをインポートするための `barman import-backup` コマンドを追加します

  tarball 形式からバックアップをインポートする新しい `barman import-backup` コマンドを導入しました。このコマンドは、サーバー名と tarball パスを引数として取り、以前にエクスポートしたバックアップを Barman カタログに登録し戻す方法を提供します。

  参考文献 BAR-1223。

22.1.2 マイナー変更

- GOOGLE_CLOUD_UNIVERSE_DOMAIN 環境変数を介して代替 GCP ユニバースをサポート

  Barman は、 `GOOGLE_CLOUD_UNIVERSE_DOMAIN` 環境変数を読み取ることにより、代替 GCP ユニバース S3NS / T-Systems などをサポートするようになりました。設定すると、値は `client_options` として GCS ストレージクライアントに渡され、 `barman-cloud-*` コマンドが非標準の GCP エンドポイントに対して動作できます。この機能は、コミュニティメンバー@mgarstecki によって寄稿されました。

  参考文献 BAR-1017。

- `barman-cloud-wal-restore` での並列 WAL リストアのサポートを追加

  並列処理のサポートが `barman-cloud-wal-restore` コマンドに追加され、ユーザーは WAL のリカバリー中に並列してダウンロードする WAL の数を指定できます。この機能強化により、複数の WAL ファイルを同時にプリフェッチすることにより、WAL 復元の全体的なパフォーマンスが向上します。詳細については、 `barman-cloud-wal-restore` のドキュメントを確認してください。

  参考文献 BAR-1253。

- 削除前に S3 オブジェクトロックをチェックするための aws_check_object_lock 構成オプションを追加

  新しい `aws_check_object_lock` 構成オプションを導入しました。有効にすると、 Barman は S3 ストレージからベースバックアップファイルを削除する前に S3 オブジェクトロックをチェックします。オブジェクトがロックされている場合、削除は中止されます。これにより、ロックされたオブジェクトの Barman の動作が `barman-cloud-delete` に調整されます。 S3 ストレージにのみ適用されます。非 S3 クラウドプロバイダーでこのオプションが有効になっている場合、警告がログに記録されます。

  参考文献 BAR-1113。

- 進行中のクラウドバックアップをバックアップカタログで可視化する

  進行中のクラウドバックアップが `barman-cloud-backup-list` にすぐに表示され、アーカイブステータス列に `STARTED` が表示され、空の終了時間が表示されます。以前は、バックアップは正常に完了した場合にのみ表示されます。 `barman-cloud-backup-delete` 、 `barman-cloud-restore` 、および `last` / `latest` / `first` / `oldest` ショートカットはすべて、それぞれ冗長数、自動復元選択、およびショートカットの解決から除外することにより、進行中のバックアップを安全に扱います。

  参考文献 BAR-1235。

- 実行時依存関係としての setuptools を削除

  Barman のランタイム依存関係として Setuptools は必要ありません。 Setuptools は、以前に Python 3.12+互換性の非推奨の distutils モジュールを提供するために含まれていましたが、別の実装に置き換えられました。 Setuptools は、パッケージ化の目的でビルド時に利用可能なままです。

  参考文献 BAR-1170。

- オプトイン.partial WAL ファイルコピーのためのバーマンリストアに-partial-wal フラグを追加します

  新しい `--partial-wal` フラグが `barman restore --no-get-wal` に追加されました。設定すると、Barman は `.partial` WAL ファイルをリカバリーに含め、 `.partial` サフィックスを削除して、PostgreSQL が再生中にそれらを見つけられるようにします。 2 つのソースがサポートされています。

  - ストリーミング WAL ディレクトリ 復元時に pg_receivewal によって書き込まれている進行中のセグメント。これを含めると、最後に完了した WAL セグメント以降に書き込まれたトランザクションがリカバリ中に利用可能になるため、RPO が削減されます。
  - WAL アーカイブ wals/ .partial サフィックスでアーカイブされたセグメント。 スタンバイが昇格し、PostgreSQL が進行中のセグメントで archive_command を呼び出したとき。圧縮および暗号化されたファイルは透過的に処理されます。

  --get-wal を使用する場合、 .partial ファイルは既に透過的に処理されているため、このフラグは効果がありません。

  参考文献 BAR-695。

- 並列 WAL アーカイブサポートを barman cloud-wal-archive に追加します

  barman cloud-wal-archive は、 --parallel フラグと、対応する cloud_wal_archive_parallel 構成オプションをサポートするようになりました。これにより、アーカイブ中に追加の WAL ファイルの日和見的プリフェッチが有効になります。 --parallel N が設定されている場合 N > 1、プライマリ WAL が正常にアーカイブされた後、 pg_wal/archive_status で準備ができている最大 N-1 個の追加 WAL ファイルが識別され、バックグラウンドワーカープロセスで同時にアップロードされます。これにより、WAL 生成が多い期間に WAL アーカイブのバックログを大幅に削減できます。 WAL ファイルが以前の呼び出しによって既にアップロードされている場合、その WAL の後続の archive_command 呼び出しはそれを検出し、0 ですぐに終了し、冗長なアップロードを回避します。

  参考文献 BAR-1101。

22.1.3 バグ修正

- クラウドストレージで更新されないバックアップメタデータを修正する

  注この修正は、 Barman サーバー、つまりバックアップ先としてクラウドストレージを使用する backup_method = local-to-cloud または backup_method = postgres で構成されたサーバーのクラウド統合を対象としています。これは、 barman-cloud-* スクリプト barman-cloud-backup などの問題ではありません。

  バックアップメタデータ backup.info ファイルはクラウドストレージで更新されておらず、これにより、たとえば、クラウドのバックアップのステータスがローカルのものと比較して古くなりました。これは、クラウドからのメタデータが主な情報ソースである可能性がある復元フェーズで問題になる可能性があります。この問題は修正されました。

  参考文献 BAR-1227。

- 使用できない.partial WAL ファイルをコピーする barman restore --no-get-wal を修正

  barman restore --no-get-wal 、 .partial を実行すると、WAL アーカイブに存在する WAL ファイルがリカバリ先にコピーされました。ただし、 Barman によって生成された restore_command は、 .partial サフィックスを考慮していない barman_wal ディレクトリの cp を使用します。したがって、これらのファイルは、リカバリ中に PostgreSQL によって使用されることはありません。 Barman は、デフォルトで .partial ファイルをスキップするようになりました。それらを明示的に含めるには、新しい --partial-wal フラグを使用します。

  参考文献 BAR-1265。

- PostgreSQL 12+の新しいタイムラインから WAL をコピーしない --no-get-wal リストアを修正

  PostgreSQL 12 以降、 recovery_target_timeline のデフォルトが current から latest に変更されました。 --no-get-wal モードで PITR オプションたとえば --target-time を使用して復元する場合、 Barman はバックアップの元のタイムラインから WAL ファイルをコピーしているだけでした。これにより、復旧ターゲットが新しいタイムラインにある場合、リストアが失敗する可能性があります。

  Barman は、デフォルトの Postgres リカバリ動作と一致する、PostgreSQL 12+で PITR を実行するときに、利用可能な最新までのすべてのタイムラインから WAL ファイルを正しくコピーするようになりました。

  参考文献 BAR-1054。

- 中間ファイルが削除されないに関する複数の警告を修正する

  WAL 圧縮を使用する場合、 Barman は、圧縮プロセス中にアーカイブディレクトリに中間ファイルを作成します。中間ファイルパスはリストに保存され、ファイルアーカイブの最後に削除されます。

  ただし、複数のファイルをバッチでアーカイブする場合、各ファイルが処理された後に中間ファイルのリストがクリアされなかったため、後のアーカイブで存在しないファイルを削除しようとし、以下のような警告が発生しました。

```
WARNING: Could not remove intermediary file /home/gustavo/pg/streaming/
↪000000010000001300000 0FD: [Errno 2] No such file or directory: '/home/gustavo/
↪pg/streaming/000000010000001300000 0FD'
```

  このバグは、バージョン 3.18、コミット 5db8ca9b で導入されました。現在は修正され、このような警告は発生しないはずです。

  参考文献 BAR-1228、CP-59700。

22.2 3.18.02026-03-12

22.2.1 重要な変更

- クラウドストレージでのインクリメンタル バックアップのサポートを追加します

  クラウドストレージでのブロックレベルのインクリメンタルバックアップのサポートが追加され、ユーザーは変更されたデータをアップロードするだけでバックアップストレージを効率的に管理し、コストを削減できます。

  この新機能は、ネイティブの Postgres バックアップ pg_basebackup で取得した Barman への、および Barman からクラウドストレージへの動的ストリーミングに依存しています。Barman ホストに完全なバックアップを保存することなく、小さなステージング領域のみを使用します。ステージング領域の場所とサイズは、それぞれ新しい cloud_staging_directory および cloud_staging_max_size 構成オプションを介して構成できます。

  この機能を構成するには、 backup_method = postgres と basebackups_directory と wals_directory をクラウドストレージ URL s3://my-bucket/barman などに設定します。

  構成例

```
[myserver]
backup_method = postgres
basebackups_directory = s3://my-bucket/barman
wals_directory = s3://my-bucket/barman
cloud_staging_directory = /tmp/barman/cloud-staging
cloud_staging_max_size = 30Gi
```

クラウドストレージのバックアップ構造は、現在の barman-cloud-/*スクリプトと同じパターンに従います。つまり、バックアップファイルは tar でアーカイブされ、サーバー名とバックアップ ID、たとえば my-bucket/barman/<server_name>/base/<backup_id>/... に基づいて名前付けられたディレクトリの下にあります。一方、WAL ファイルはサーバーディレクトリの下の wals サブディレクトリ、つまり my-bucket/barman/<server_name>/wals/... に保存されます。この意味で、basebackups_directory および wals_directory は、クラウドストレージのバックアップパスと WAL パスのプレフィックスとして機能します。これにより、既存のバックアップ管理実践との互換性が保証されます。

重要な注意事項

- これは実験的な機能であり、注意すべき制限がいくつかあります。詳細については、ドキュメントを参照してください。
- この方法で作成されたバックアップの復元は、現在 Barman では直接サポートされておらず、これを手動でまたはカスタムスクリプト/プロセスを介して実行するのはユーザーの責任です。このようなバックアップの復元は、将来のリリースで提供される予定です。
- これは、バーマンクラウドエクスペリエンスを統合し、クラウドバックアップ操作を簡素化するための最初のステップであり、将来のリリースではより多くの機能と改善が追加されます。

参考文献 BAR-928。

- local-to-cloud バックアップ方法を barman CLI に追加

新しい local-to-cloud バックアップ方法を追加することにより、メインの barman CLI を介した直接クラウドバックアップのサポートを導入しました。以前は、クラウドバックアップは、スタンドアロンの barman-cloud-* スクリプトを介してのみ利用できました。

この機能強化により、 backup_method = local-to-cloud と basebackups_directory s3://bucket/path などのクラウドストレージ URL を使用して、バーマン構成ファイルにクラウドバックアップを直接構成できるようになりました。これは、すべてのバックアップ方法に統一インターフェイスを提供し、スタンドアロンの barman-cloud-* スクリプトを非推奨にするための最初のステップです。

新しい CloudBackupExecutor は、既存のバーマンバックアップインフラストラクチャとシームレスに統合し、低レベルバックアップ API を使用して PostgreSQL と連携し、バックアップをクラウドオブジェクトストレージに直接アップロードします。

構成例

```
[myserver]
backup_method = local-to-cloud
```

(次のページに続く)

(前のページからの続き)

```
basebackups_directory = s3://my-bucket/barman/backups
conninfo = host=postgres-host dbname=postgres
```

これは、バーマンクラウドエクスペリエンスを統合し、クラウドバックアップ操作を簡素化するための最初のステップです。

次の構成オプションは再利用または Barman 構成に追加されて、 barman-cloud スクリプトと機能パリティを提供します。

- cloud_upload_min_chunk_size
- aws_region
- aws_read_timeout
- aws_encryption
- aws_sse_kms_key_id
- bandwidth_limit
- cloud_delete_batch_size
- cloud_upload_max_archive_size

参照 BAR-1095、BAR-1116、BAR-1122、BAR-1109、BAR-1120、BAR-1121、BAR-1117、BAR-1112、BAR-1115。

- ローカルからクラウドへの WAL アーカイブ用に `barmancloud-wal-archive' コマンドを追加

'local-to-cloud' バックアップ方法を使用する場合に、PostgreSQL pg_wal ディレクトリからクラウドオブジェクトストレージへの直接 WAL ファイルアーカイブを有効にする新しい barman cloud-wal-archive コマンドを導入しました。このコマンドは、 Barman サーバーが PostgreSQL インスタンスと同じホストで実行されている場合、PostgreSQL archive_command として使用するように設計されています。

クラウドオブジェクトストレージに送信される WAL ファイルは、既存の compression および compression_level 構成オプションを使用して、アップロードする前に圧縮することもできます。圧縮された WAL ファイルは、barman-cloud-wal-archive で既に使用されている規則に従って、適切なファイル拡張子たとえば .gz 、 .lz4 で保存されます。

次のインメモリ圧縮アルゴリズムがサポートされています。 gzip 、 bzip2 、 xz 、 snappy 、 zstd 、および lz4 。 pigz および custom は、外部プロセスに依存するため、クラウド WAL ストレージではサポートされていません。

構成例

```
[myserver]
backup_method = local-to-cloud
archiver = on
wals_directory = s3://my-bucket/barman/wals
compression = gzip
compression_level = 6
```

参照元 BAR-1098、BAR-1124

22.2.2 マイナー変更

- S3 互換ストレージプロバイダーの `--addressing-style` オプションを追加

  AWS S3 または S3 互換ストレージプロバイダーを使用する場合、barman-cloud コマンドで新しい `--addressing-style` オプションが利用できるようになりました。このオプションを使用すると、ユーザーは S3 アドレススタイルを明示的に `auto` 、 `virtual` 、または `path` に設定できます。 `boto3` は非 S3 エンドポイントのパススタイルのアドレス指定を自動的にデフォルトにするため、仮想ホストスタイルのアドレス指定を必要とする S3 互換ストレージシステムを使用する場合に役立ちます。

  PR #1134 でこの機能を報告および寄稿していただいた@ChandonPierre に感謝します。

  参考文献 BAR-1011

- クラウド操作の Python 3.14 互換性サポートを追加

  クラウドバックアップ操作に影響を与える `functools.partial` の重大な変更を修正することにより、Python 3.14 のサポートを追加しました。このバージョンでは、 `functools.partial` は、クラス属性として使用されたときに動作を変更するメソッド記述子になりました。この更新により、Python 3.14 にアップグレードしても Barman クラウド操作がシームレスに動作し続けます。

  このパッチは、@pvbiesen によるオリジナルのアイデアに基づいています。彼は機能を提案し、PR #1143 に貢献しました。同様の修正は、PR #1109 で@mjlshen によって提案されました。

  参考文献 BAR-1012

- 圧縮を使用したバックアップのフックとして barman-cloud-backup を使用しないようにする

  圧縮して取得されたバックアップのフックとしての `barman-cloud-backup` の使用は、適切なメッセージでエラーアウトすることにより防止されるようになりました。

  この変更により、圧縮を含むバックアップが誤ってクラウドバックアップフックで使用されることがなくなります、そのような使用はこれまでサポートされていなかった。

  この動作は、警告ノートとともに文書化されています。

  参考文献 BAR-1015。

- Barman クラウドの LZ4 圧縮サポートを追加

  `barman-cloud-backup` コマンドでの LZ4 圧縮のサポートを追加しました。 –lz4 オプションを指定して、クラウドストレージにアップロードする前に LZ4 圧縮を使用してバックアップファイルを圧縮できるようになりました。このオプションは他の圧縮オプション-gzip、-bzip2、-snappy と相互排他的であり、 `lz4` Python ライブラリをインストールする必要があります。

  この機能は、PR #1144 で@vietcgi によって提案および実装されました。

  参考文献 BAR-1013。

- WarehousePG データベース ID 構成の実験的なサポート

  WarehousePG の実験的なサポートの追加。これには、ユーザーが PostgreSQL バックアップ方法を使用するときに WarehousePG セグメントのデータベース ID を指定できる新しい構成オプション

warehousepg_dbid が含まれます。この構成は、バックアップおよびリストア操作のために WarehousePG とのより適切な統合を有効にすることを目的としています。

archive_timeout が WarehousePG セグメントの pg_settings で公開されていないため、Postgres パラメーターの収集も向上しています。

参考文献 BAR-1092。

- バーマンリカバリー操作のカスタム復元コマンドのサポートを追加します

  barman recover コマンドに新しい --restore-command オプションを追加しました。これにより、ユーザーは Barman のデフォルトの restore コマンドをカスタムコマンドでオーバーライドできます。Barman がビルドしたデフォルトの restore コマンドは、ほとんどの場合に適していますが、このオプションを使用すると、セットアップで必要な場合に代替コマンドを指定できます。

  以前は、ユーザーは復旧後に PostgreSQL リカバリ構成ファイルを手動で編集して restore コマンドを変更する必要がありましたが、これはエラーが発生しやすく、追加の手動介入が必要でした。この新しいオプションを使用すると、ユーザーはコマンドラインを介してカスタム復元コマンドを直接指定できるようになりました。

  この機能強化は、@PaganMuffin によって#881 で報告された問題のために適用されました。

  参考文献 BAR-201。

22.2.3 バグ修正

- 長時間実行されている WAL アーカイブサービスのファイルハンドルリークを修正

  WAL ファイルをクラウドストレージにアップロードした後、ファイルハンドルが適切に閉じられない、クラウド WAL アーカイブ機能のリソースリークを修正しました。この問題は、主に、複数の WAL ファイルを順番に処理する長期実行 WAL アーカイブサービスに影響を与える可能性があります。開いているファイルハンドルが蓄積すると、最終的に「Too many open files」エラーが発生し、それ以上の WAL アーカイブ操作が妨げられる可能性があります。

  この修正により、圧縮が使用されるかどうかに関係なく、およびアップロード失敗が発生した場合でも、各アップロード操作が完了した直後にファイルハンドルが適切に閉じられるようになりました。これにより、クラウドプロバイダーへの継続的または高頻度の WAL アーカイブを実行するサービスでのハンドルの枯渇を防ぎます。

  PR #1129 でこの機能に貢献した@ethiebautgeorge-nasuni に感謝します。

  参考文献 BAR-973。

22.3 3.17.02026-01-07

22.3.1 重要な変更

- 非アクティブ/無効なサーバーでのクエリー、取得、および選択的な操作を有効にする

  複数の Barman コマンドは、安全プロファイルに基づいて、非アクティブで無効になったサーバーで動作するようになりました。

  Query commands

- `list-backups` 非アクティブ/無効なサーバーからバックアップを表示する
- `status` 非アクティブ/無効になったサーバーのステータスを確認します
- `show-backup` 非アクティブ/無効なサーバーからのバックアップの詳細を表示します
- `list-files` 非アクティブ/無効なサーバー上のバックアップからファイルをリストする
- `verify-backup` 非アクティブ/無効なサーバーでのバックアップを検証します

Fetch commands

- `restore` 非アクティブなサーバーからバックアップを復元する
- `get-wal` 非アクティブなサーバーから WAL ファイルを取得します

Modification commands

- `delete` 非アクティブなサーバーからバックアップを削除します
- `config-switch` 非アクティブ/無効なサーバーのスイッチ構成モデル

この変更により、サーバーが手動で無効になっている非アクティブな場合、または構成に問題がある無効な場合でも、ユーザーはバックアップ情報を照会し、復元操作を実行できるため、運用の柔軟性が向上します。無効になったサーバーの異なる動作により、潜在的な構成の競合またはパスの問題があるサーバーでの操作に対する安全性が提供されます。

参考文献 BAR-976。

- WAL ファイルのカスタム圧縮を非推奨にする

  `custom` 圧縮オプションとその関連構成オプション `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。

  現在カスタム圧縮を使用しているユーザーは、 Barman がサポートするビルトイン圧縮アルゴリズムのいずれかに移行することをお勧めします。

  カスタム圧縮機能は、特定の圧縮要件を持つユーザーに柔軟性を提供するように設計されましたが、任意の外部圧縮ツールとの互換性を維持することは、複雑でエラーが発生しやすいことがわかっています。ビルトイン圧縮オプションは、より良いパフォーマンス、信頼性、および保守性を提供します。

  カスタム圧縮を使用する既存の構成は、現在のリリースでも引き続き動作しますが、ユーザーは機能が削除される前に構成の移行を計画する必要があります。

  参考文献 BAR-993。

22.3.2 マイナー変更

- ファイル暗号化のチェック時に非 UTF8 出力を処理する

  `barman-wal-restore` または `barman rebuild-xlogdb` を実行して WAL を復旧する場合、 Barman は `file` コマンドを実行して、ファイルが暗号化されているかどうかを確認します。

  macOS システム公式にサポートされていない場合、このコマンドは、その出力に UTF-8 を使用してデコードできないバイトが含まれている場合、 `UnicodeDecodeError` 例外を発生させると報告されています。

これを防ぐために、 Barman はデコード不能なバイトを適切に処理し、プロセスを中断せずに続行できるようになりました。

Github issue #1117

参考文献 BAR-926。

- `--recovery-option-port` を `barman restore` コマンドに追加

  `get-wal` モードでリモート復元を実行する場合、Barman は `barman-wal-restore` を呼び出して WAL を取得する `restore_command` を書き込みます。ただし、現在、作成された `barman-wal-restore` コマンドにカスタムポートを指定する方法はありません。

  この変更では、`--recovery-option-port` フラグが `barman restore` コマンドに導入され、ユーザーはリモートリストア中に `barman-wal-restore` コマンドの `--port` オプションの値を指定できます。

  参考文献 BAR-943。

- barman-cloud でのバックアップの削除のための S3 オブジェクトロックのサポートを追加します

  `barman-cloud-backup-delete` は、AWS S3 でバックアップを削除する前にオブジェクトロックのチェックをサポートするようになりました。この機能は、コンプライアンスまたはガバナンス保持ポリシーによって保護されたオブジェクトの削除を防止するのに役立ちます。

  新しい `--check-object-lock` オプションが追加され、削除前にオブジェクトロックの検証が有効になります。有効にすると、ベースバックアップのすべてのオブジェクトの削除操作は、これらのオブジェクトのいずれかがロックされている場合に中止されます。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。

  この機能は、削除するオブジェクトごとに追加の API 呼び出しを必要とするため、削除操作にオーバーヘッドが追加され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、正しく機能するには適切な権限が必要です。

  参考文献 BAR-946。

22.3.3 バグ修正

- バックアップ削除のための非準拠 S3 オブジェクトストアのカバレッジを拡大する

  この修正では、バックアップ削除の非準拠 S3 オブジェクトストアのカバレッジを拡大します。

  「content-MD5」を含むメッセージを返すエラーコードをキャッチできなかった場合に、返されるメッセージと一致していた式を削減しました。

  これにより、バックアップ削除の非準拠の S3 オブジェクトストアのほぼすべてがカバーされると考えています

  参考文献 BAR-974。

- 同期していないパッシブノードを修正する

  Barman 3.16 では、`barman diagnose` コマンドの“config”キーから“name”フィールドが削除されました。パッシブノードを同期するときにも同じ「config」データが使用され、「name」フィールドの欠如が同期に失敗することがわかりました。

この変更により、“name”フィールドが“config”キーに追加され、パッシブノードの同期が修正されます。

GitHub issue #1125 で報告されました。

参考文献 BAR-971。

- デルタリストアが機能しないを修正

  デルタリストア機能は、 Barman が復元を実行する前に宛先ディレクトリを削除している見落としにより、正しく機能していませんでした。この問題は、特定の操作で宛先ディレクトリの削除を回避することにより修正されました。 GitHub issue #1139 で報告されました。

  参考文献 BAR-1002。

- リストア中の「rmexecution failed」エラーを修正する

  バージョン 3.15 のリリース中に実装されたリファクタリングの結果、 Barman は、バックアップファイルをコピーする前に復元操作の宛先ディレクトリの削除を開始しました。これは、ファイルが常にクリーンなディレクトリにコピーされるようにするためです。

  ただし、宛先ディレクトリがマウントポイント、またはユーザーが親に対する十分な権限を持っていないサブディレクトリである場合、ディレクトリの削除は失敗し、「rm 実行に失敗しました」のようなエラーが発生します。

  この問題は、リストアの最終的な宛先での PGDATA の宛先ディレクトリとテーブルスペースの削除を回避することにより修正されました。

  参考文献 BAR-975。

22.4 3.16.22025-11-04

22.4.1 バグ修正

- barman-cloud の削除回避策を改善して、エラーメッセージのコンテンツを検査する

  一部の S3 互換ストレージは、削除操作中に「MissingContentMD5」の代わりに「InvalidRequest」エラーを結果ます。エラーコードのみに依存することはできないため、これにより、エラーメッセージテキストも検査して MissingContentMD5 のような問題を検出し、適切なフォールバックを適用するようにロジックが更新されます。

  参考文献 BAR-947。

22.5 3.16.12025-10-14

22.5.1 重要な変更

- 自動バケット作成が barman-cloud-check-wal-archive に追加されました

  `barman-cloud-check-wal-archive` コマンドは、ターゲットバケットが構成済みのクラウドストレージ AWS S3 に存在するかどうかを確認し、存在しない場合は自動的に作成します。

以前は、自動バケットの作成は barman-cloud-backup によって処理されていました。この変更により、この動作に依存する CloudNativePG CNPG オートメーションとの互換性が復元され、 Barman 3.16.0 のリリース後に影響を受けました。

CloudNativePG チームと協力して、私たちはこの機能を barman-cloud-check-wal-archive に再導入して、CNPG ワークフローと内部テストのシームレスなオペレーションを保証しながら、通常の Barman ユーザーが意図した動作と簡単さを維持します。

参考文献 BAR-931。

22.6 3.16.02025-10-02

22.6.1 重要な変更

- ローカルリストアの restore_command には、 sudo が含まれなくなりました

  ローカル復元に barman restore --get-wal を使用する場合つまり --remote-ssh-command なしで、生成された restore_command には sudo -u $USER プレフィックスが含まれません。これにより、ローカルに復元された Postgres クラスターが barman restore 操作の終了までに既にファイルを所有している barman ユーザーによって実行される一般的な場合のプロセスが簡素化されます。

  Action Required: 復元されたクラスターを別のユーザーたとえば postgres として実行する予定がある場合は、適切な sudo -u postgres プレフィックスをリカバリ構成の restore_command に手動で追加し、ファイルの所有権が正しいことを確認する必要があります。

  参考文献 BAR-675。

- pygzip および pybzip2 圧縮オプションを非推奨にします

  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。ユーザーは gzip および bzip2 オプションに移行する必要があります。これらは Python の内部ライブラリを使用するようになりました。

  この変更は、パフォーマンスとコードの一貫性を向上させるために行われました。以前は、 gzip および bzip2 オプションは外部サブプロセスに依存していましたが、対応する py はネイティブ Python ライブラリを使用していました。 gzip および bzip2 のコア圧縮ロジックは、Python のビルトインライブラリを使用するように更新され、 pygzip および pybzip2 オプションが冗長になっています。

  参考文献 BAR-878。

- delta-restore フラグとリカバリオプションをバーマンリストアに追加

  barman リストアの-delta-restore フラグを導入し、デルタリストアモードを有効にします。このモードでは、宛先ディレクトリに既に存在する同一のデータを再利用してバックアップを復元します。これにより、復元時間とネットワークトラフィックが大幅に削減されます。

  この機能は、サーバー構成ファイルの delta-restore と呼ばれる新しいフィールドとしての `recovery_options' を介してサーバーごとに構成することもできます。

  参考文献 BAR-788。

- boto3 >=1.36 に準拠していないオブジェクトストアの S3 オブジェクト削除フォールバック

一部の非準拠の S3 互換オブジェクトストアで delete_objects() を使用する場合、ユーザーは MissingContentMD5 ClientError が発生する可能性があります。boto3 1.36 の破壊的変更により、Botocore が Content-MD5 ヘッダーを自動的に計算および設定しなくなりました。

これに対処するために、 Barman はフォールバックメカニズムを実装するようになりました。

- delete_objects() を介した一括削除は、引き続き最初に試行されます。
- delete_objects() が MissingContentMD5 で失敗した場合、Barman は自動的にフォールバックし、 Content-MD5 ヘッダーを必要としない delete_object() を使用してオブジェクトを個別に削除します。
- 他のエラーは、以前と同様に例外を発生させます。

この変更により、 Barman は、ユーザー側で変更を必要とせずに非準拠オブジェクトストアとの互換性を維持できます。

古い S3 互換ストレージを使用するユーザーは、障害なしでバックアップクリーンアップの実行を継続できますが、新しいオブジェクトストアは影響を受けません。

参考文献 BAR-909。

- クラウド接続チェックは、 -t / --test フラグを使用してのみ実行されます。

  以前は、 Barman はほとんどのコマンドでクラウド接続とバケットの存在を常にテストし、接続に失敗した場合、またはバケットが存在しない場合にエラーを生成しました。 -t / --test フラグが指定された場合にのみ、両方のチェックが実行されるようになりました。唯一の例外は barman-cloud-check-wal-archive でしたが、これは引き続き接続とバケットの存在をチェックします。

  さらに、バケットは自動的に作成されなくなりました。Barman は、ターゲットバケットが既に存在することを想定しているため、ユーザーの責任で事前にバケットをプロビジョニングします。

  リファレンス BAR-895、BAR-908、BAR-918。

22.6.2 マイナー変更

- barman restore コマンドに combine-mode オプションを追加

  --combine-mode オプションを barman restore コマンドに追加しました。これは、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ときに pg_combinebackup のコピーモードを指定するために使用できます。使用可能なモードは copy 、 link 、 clone 、および copy-file-range であり、それぞれ pg_combinebackup オプション --copy 、 --link 、 --clone 、および --copy-file-range と同等です。同等の combine_mode 構成オプションは、構成ファイルで指定できます。

  この機能により、基になるファイルシステム機能に基づいて、復元プロセスのより大きな最適化が可能になります。

  参考文献 BAR-870。

- バーマン診断 JSON のフィールドを再編成する

  バーマン診断 JSON では、実際の構成フィールドではない場合でも、サーバーの「config」セクションの下にいくつかのフィールドが表示されます。

フィールド“barman_lock_directory”、“lock_directory_cleanup”、および“config_changes_queue”は、グローバル構成を表すため、“global”キーの下に再配置されました。サーバー名が既にキーとして使用されているため、“name”フィールドが削除されました。さらに、“msg_list”がサーバーの名前キーの下に保存されるようになりました。

参考文献 BAR-134。

- `--get-wal` が `True` の場合、barman-wal-restore コマンドに `-p JOBS' を追加します

  `--get-wal` または `get_wal` 構成オプションが `True` であり、`JOBS` が `-p` フラグまたは `parallel_jobs` 構成オプションを介してユーザーが選択したジョブの数である場合、barman-wal-restore コマンドに `-p JOBS' を追加します。

  参考文献 BAR-857。

- */barman show-backup/* コマンドは、バックアップの圧縮方法を表示するようになりました

  `barman show-backup` コマンドの出力が強化され、新しい `Backup Compression` フィールドが含まれます。圧縮を有効にしてバックアップが取得された場合、このフィールドには、使用された圧縮アルゴリズム `zstd` などが表示されます。これにより、バックアップの圧縮ステータスをすばやく確認することが簡単になります。

  参考文献 BAR-901。

22.6.3 バグ修正

- barman-wal-restore の並列処理を修正します –Parallel オプション

  Barman 3.12.0 で `--keep-compression` オプションを導入したときに `barman-wal-restore` にリグレッションがあり、 `--parallel` オプションが予想通りにプロセスを呼び出しません。この問題は修正されました。

  参考文献 BAR-865。

- `staging_location` が `remote` の場合、 `PATH` で `pg_combinebackup` が見つからない問題を修正

  `staging_location` が `remote` に設定されているリモートサーバーにインクリメンタルバックアップをリカバリーする場合、Barman はリモート `PATH` で `pg_combinebackup` が利用できるかどうかを確認します。ただし、以前の問題により、ローカルの `PATH` も誤ってチェックされ、`pg_combinebackup` がローカルにインストールされていない場合にエラーが発生しました。この問題は、`pg_combinebackup` が実際に実行されるサーバーでのみチェックが実行されるようにすることにより修正されました。

  参考文献 BAR-903。

22.7 3.15.02025-08-05

22.7.1 重要な変更

- 圧縮と暗号化を使用してインクリメンタル バックアップを取得できます

  以前は、 Barman は、圧縮が有効になっている場合、または親バックアップが既に圧縮されている場合、インクリメンタルバックアップが実行されないようにしていました。 Barman での暗号化には圧縮が必要であるため、これは、暗号化では増分バックアップを取得できないことを意味しました。

この制限は、このようなバックアップの復元の複雑さのために存在し、復号化、解凍、およびバックアップの組み合わせのための複数のステージングフェーズが関係します。

重要なリファクタリングの努力の後、 Barman は、圧縮および/または暗号化の両方を使用したインクリメンタルバックアップの取得を完全にサポートするようになりました。これにより、より柔軟なバックアップ戦略が可能になり、ユーザーは使用可能なすべての機能を制限なく利用できます。

リファレンス BAR-764、BAR-801、BAR-841。

- `local_staging_path` および `recovery_staging_path` 構成オプションの非推奨

  `local_staging_path` および `recovery_staging_path` 構成オプションは非推奨であり、新しい `staging_path` および `staging_location` オプションを優先します。古いオプションは下位互換性のために引き続き機能しますが、ユーザーはより良い明確さと柔軟性を得るために新しい構成オプションに移行することをお勧めします。

  リファレンス BAR-571、BAR-801。

- ステージング領域構成の柔軟性と一貫性の向上

  圧縮、暗号化、またはインクリメンタルバックアップを復元するときは、 Barman は、解凍または復号化されたファイルなどの中間ファイル、または結合されたインクリメンタルバックアップデータを処理するためのステージング領域が必要です。

  以前は、 Barman には、シナリオごとに異なる構成オプションがありました。

  - インクリメンタル バックアップの場合、ステージング領域は Barman 構成の `local_staging_path` を介して指定されました。これは、 Barman ホスト内のパスのみです。
  - 圧縮バックアップの場合、ステージング領域は `recovery_staging_path` を介して指定されました。これは、 Barman ホスト内のパスローカルリカバリの場合またはターゲットサーバー内のパス別のサーバーへのリモートリカバリの場合。
  - 暗号化されたバックアップの場合、ステージング領域は Barman 構成の `local_staging_path` を介して指定されました。これは、Barman ホスト内のパスのみです。暗号化には圧縮を有効にする必要があるため、暗号化されたバックアップを復元するとき、ユーザーは `recovery_staging_path` と `local_staging_path` の両方を指定する必要があることに注意してください。

  Barman は、単一の統合されたより柔軟なステージングエリア構成を提供するようになりました。

  - 新しい `staging_path` 構成オプションを使用して、中間ファイルが保存されるステージング領域の絶対パスを指定できます。デフォルト値は `/tmp` です。
  - 新しい `staging_location` 構成オプションを使用して、ステージング領域が Barman ホスト `local` 上にあるか、リモートサーバー `remote` 上にあるかを指定できます。デフォルト値は `local` です。

  これは、ステージングパスの統合された構成を持つことに加えて、ユーザーが Barman ホストまたはリモート Postgres サーバーのいずれかにステージング領域を配置する場所をより柔軟に指定できることを意味します。

  ストレージの使用量も最適化されて、可能な限りリソースの消費量が削減されます。これには以下が含まれます

  - 中間ファイルが必要でなくなったらすぐに削除します。

- 必要以外の場合、ステージング領域の使用を避けてください。 圧縮されたバックアップをローカルに復元する場合、ステージングは必要なく、バックアップは最終的な宛先に直接解凍されます。

この変更により、構成が簡素化され、一貫性とリソースの使用率が向上し、エンドユーザーにとって復旧管理がより透明で予測可能になります。

リファレンス BAR-764、BAR-765、BAR-801、BAR-841。

22.7.2 マイナー変更

- `--staging-path` および `--staging-location` を `barman restore` に追加

  `--staging-path` および `--staging-location` オプションを `barman restore` コマンドに追加して、ユーザーがリストアプロセス中にステージング領域のカスタムパスを指定できるようにします。`--staging-path` は、`--staging-location` がその場所、つまり `local` または `remote` を定義するときに使用される絶対パスです。 `barman.conf` には、それぞれ `staging_path` および `staging_location` として同等の構成オプションがあります。

  参考文献 BAR-570。

- オプション `--if-not-exists` を `barman receive-wal --create-slot` に追加

  新しい `--if-not-exists` フラグが `barman receive-wal --create-slot` コマンドに追加されました。これにより、既に作成されたレプリケーションスロットを作成しようとしたときにコマンドが失敗するのが防止され、スクリプトでの使用の信頼性が向上します。

  PR を作成し、 Barman に貢献してくださった@crazybolillo antonio@zoftko.com 別名@antonag32 に感謝します。

  参考文献 BAR-527。

- バックアップメタデータで予期しないフィールドが見つかった場合、正常に失敗します。

  この更新では、 Barman が現在のバージョンで不明なフィールドを含むバックアップメタデータを処理する方法を改善します。この状況は、 Barman の新しいバージョンで作成されたバックアップを古いバージョンを使用して復元しようとしているときに発生する可能性があります。たとえば、新しいバージョンが暗号化に関するメタデータを追加し、古いバージョンがそのフィールドを認識しない場合。これは、この変更が関連する場合です。

  以前は、これにより役に立たない例外が発生していました。ここで、 Barman は、バックアップが新しいバージョンで作成され、現在のバージョンと互換性がない可能性を説明する明確なメッセージを表示します。

  参考文献 BAR-763。

- 適切なクリーンアップのための孤立したバックアップの警告を改善しました

  孤立したバックアップの警告メッセージが改善され、ユーザーが不完全または手動で削除されたバックアップを完全にクリーンアップしやすくなりました。以前は、メッセージに `backup.info` ファイルの場所が記載されておらず、混乱を招いていました。更新されたメッセージは、`backup.info` ファイルの場所を明確に示し、ユーザーが適切に削除できるようになりました。

  参考文献 BAR-793。

- クラウドコマンドのタグ処理を改善して解析エラーを修正

  barman-cloud-backup および barman-cloud-wal-archive の --tags コマンドラインオプションは、解析の信頼性を向上させ、標準の CLI 実践と調整するために置き換えられました。以前のスペース区切り形式では、特に位置引数の前に配置された場合、解析エラーが発生する可能性がありました。

  コマンドは、複数の --tag フラグを、キーと値のペアごとに 1 つずつ受け入れるようになりました。

  Before: barman-cloud-backup ... --tags key1,val1 key2,val2

  After: barman-cloud-backup ... --tag key1,val1 --tag key2,val2

  --tags オプションは非推奨になりました。古い --tags 形式を使用する既存のスクリプトは引き続き動作しますが、ユーザーはスクリプトを新しい形式に更新することをお勧めします。この変更には下位互換性があり、重大な変更は導入されません。

  参考文献 BAR-441。

- barman show-backup 出力の WAL Number ラベルを明確に

  barman show-backup 出力の誤解を招く WAL Number ラベルの名前を Number of WALs に変更します。

  参考文献 BAR-791。

- show-backup コマンドの「No of files」を「Number of files」に変更します

  show-backup コマンドの Wal Information セクションでは、“No of files”の出力が“Number of files”に変更されました。

  参考文献 BAR-798。

- config-update を介してサーバーとモデルに渡されたオプションを検証する

  barman config-update コマンドは、変更をエンキューする前に、提供された各セクションで追加の検証を実行するようになりました。とりわけ、すべての必須フィールドが存在すること、提供された値が予想される型に準拠していること、および各値がそれぞれのパーサーも渡すことを検証します。さらに、チェックが失敗した場合にコマンドはエラーになるようになりました。これにより、部分的または不正な形式の更新が適用されるのを防ぎます。

  参考文献 BAR-142。

- barman check および barman backup コマンドに --check-timeout 引数を追加

  barman check および barman backup コマンドに新しい CLI オプションが導入され、チェックオペレーションの時間が設定された check_timeout を超え、バックアップがスケジュールされない状況に対処します。このオプションを使用すると、ユーザーはタイムアウトを無効にするか、デフォルトまたは構成済みの check_timeout 値をオーバーライドできます。

  参考文献 BAR-219。

- --list-empty-directories を barman list-files コマンドに追加

  新しい --list-empty-directories 引数を list-files コマンドに追加して、空のディレクトリをリストに追加します。

  参考文献 BAR-220。

- 自動 WAL スイッチとアーカイブによるアーカイブチェックの向上

  ユーザーエクスペリエンス、特にサーバーの初期構成中のユーザーエクスペリエンスを向上させるために、 `barman check` コマンドがより堅牢になりました。

  以前は、WAL アーカイブが空の場合、チェックは失敗し、手動介入が必要でした。

  Barman は、最初に Barman サーバーに存在する既存の WAL をアーカイブしようとすることにより、これをインテリジェントに処理するようになりました。存在しない場合、Postgres サーバーの WAL スイッチを自動的にトリガーし、結果のファイルをアーカイブしようとします。この機能強化により、一般的な手動手順が自動化され、すぐに使える初期ヘルスチェックの信頼性が向上します。

  参考文献 BAR-718。

- Barman クラウドスクリプトでバックアップ ID としてショートカットをサポート

  Barman クラウドコマンドでバックアップを参照するときに、ショートカット `first` / `oldest` 、 `last` / `latest` および `last-failed` を指定できるようになりました。 Barman は、これらのショートカットを自動的に解析し、関連するバックアップ ID を返すようになりました。

  参考文献 BAR-717。

22.7.3 バグ修正

- バックアップ中に一時テーブルのデータファイルを無視する

  一時関係ファイルのパターンと一致するファイルは、 `pg_basebackup` によって暗黙的に実行される除外と同様に、 `rsync` バックアップ中に無視されるようになりました。これは、rsync が既に削除された一時ファイルをコピーしようとする場合があり、手動でブロックが解除されるまで Barman をハングアップする可能性があるため、多くの一時テーブルを作成およびドロップする Postgres インスタンスに対して rsync コンカレントモードで Barman を実行する際のエラーを回避するために必要です。

  参考文献 BAR-266。

- `pg_combinebackup` が見つからない場合のエラーメッセージを改善

  以前は、インクリメンタルバックアップを復元し、PATH で `pg_combinebackup` が利用できない場合、 Barman は例外を適切に処理せず、エラーを理解して修正することが困難でした。現在、適切な例外が発生し、ターゲットサーバーで `pg_combinebackup` が見つからなかったことを示す明確なメッセージが表示されます。

  参考文献 BAR-840。

- ターゲットデータまたはテーブルスペースディレクトリが空でない場合、リストアは失敗します。

  以前、 Barman は、復元する前にテーブルスペースの宛先ディレクトリが空かどうかを検証しなかったため、コンテンツがサイレントに上書きされ、データの損失または破損が発生する可能性がありました。ここで、復元プロセスは、続行する前に、データとテーブルスペースを含むすべての宛先ディレクトリが空であることを確認します。そうでないものがある場合、 Barman はオペレーションを停止し、適切なエラーメッセージを表示し、ローカルとリモートの両方の復元の安全性とデータの完全性を大幅に向上させます。

  参考文献 BAR-787。

- 孤立した古いバックアップを削除する際のエラーを修正する

  バックアップが孤立していて古い場合、保持ポリシーの適用が失敗する場合がある問題を解決しました。 Barman は、ベースバックアップディレクトリを削除しようとする前に、ベースバックアップディレクトリが存在することを確認し、ディレクトリが既に削除されている場合のエラーを防止するようになりました。

  参考文献 BAR-794。

- PG17 境界 LSN の正しい WAL セグメントの名前付けを確認します

  barman バックアップ <server> を使用する場合に PostgreSQL 17 以降のバージョンで発生したバックアップのハング問題を修正し、低アクティビティのインスタンスで待機し、 pg_walfile_name() および pg_walfile_name_offset() と対話するメソッドとプロパティ内のロジックを変更します。

  barman#1041 で説明しているように、PostgreSQL 17 では、バックアップの終了 WAL が、最後に完了した WAL ファイルではなく、現在書き込まれている WAL ファイルになりました。アクティビティが最小限のサーバーでは、この現在の WAL ファイルはすぐに完了またはアーカイブされない場合があり、 pg_switch_wal() が発生するまでバックアッププロセスが無期限に停止する場合があります。

  PostgreSQL の変更の詳細については、 https://www.postgresql.org/docs/release/17.0/ を参照してください。

  参考文献 BAR-519。

- スナップショットバックアップが AWS の Nitro インスタンスの非ルート EBS ボリュームをサポートするようになりました

  スナップショットバックアップが AWS の NVMeNitro デバイスの非ルート EBS ボリュームを検出できなかった問題を解決しました。スナップショットバックアップは、インスタンスに接続されているすべての非ルート EBS ボリュームで確実に動作するようになりました。

  リファレンス BAR-500

22.8 3.14.12025-06-18

22.8.1 バグ修正

- WAL アーカイブのパフォーマンス問題を修正する

  Barman WAL アーカイブプロセスは、現在の実行でアーカイブされない WAL ファイルの処理が原因でパフォーマンスの低下が発生していました。また、3.14 で導入された暗号化ロジックにも誤りがあり、 Barman はサーバーにストリーミングされる WAL ファイルの暗号化をチェックしました。ストリーミングされた WAL は GPG 暗号化も圧縮もされないため、これは不要です。両方の問題が修正されると、アーカイブプロセスが大幅に高速化され、リソース効率が向上します。

  問題 1087 の詳細な分析をしていただいた@thealex55 に感謝します。

  参考文献 BAR-775。

22.9 3.14.02025-05-15

22.9.1 重要な変更

- tar バックアップと WAL ファイルの GPG 暗号化の実装

  tar バックアップの GPG 暗号化を実装します。暗号化はバックアップの最後に開始され、PGDATA のバックアップとバックアップディレクトリに存在するテーブルスペースが暗号化されます。暗号化されたバックアップファイルには、 `.gpg` 拡張子が追加されます。

  Barman は、新しい `encryption_passphrase_command` 構成オプションで取得したパスフレーズを使用して、GPG 暗号化されたバックアップの復号化と復元をサポートしています。復元プロセス中に、復号化されたファイルは Barman ホストの `local_staging_path` 設定でステージングされ、信頼性の高い安全な復元プロセスが保証されます。

  バックアップと WAL ファイルの暗号化と復号化に必要な新しい構成オプションを追加する必要があります。新しいオプションは `encryption` 、 `encryption_key_id` 、および `encryption_passphrase_command` です。

  `encryption = gpg` の場合、WAL ファイルはすべて GPG で暗号化されます。これには、xlogdb レコードの読み取りおよび書き込み方法の変更下位互換性の維持、およびファイルが暗号化されるタイミングと暗号化プロセス自分自身を検出する新しいロジックが含まれます。

  barman list コマンドの get-wal および no-get-wal フラグを使用する場合、復元プロセス中の GPG 暗号化 WAL ファイルの復号化。これにより、 `encryption_passphrase_command` 構成オプションを介してバックアップを復号するために追加された機能が拡張されます。

  `show-backup` には、バックアップが暗号化されたかどうかを公開する新しいフィールドがあり、使用された暗号化方法がある場合は指定します。

  `barman check` コマンドは、ユーザーの暗号化設定が Barman サーバーで正しく構成され、予想通りに機能しているかどうかを確認します。

  参照 BAR-683、BAR-687、BAR-693、BAR-669、BAR-671、BAR-692、BAR-685、BAR-680、BAR-670、BAR-681、BAR-702。

- Python バージョン 3.6 および 3.7 の非推奨

  Barman のバージョン 3.14 の時点で、Python 3.6 および 3.7 のサポートを非推奨にします。これらは Python コミュニティでサポートされていません。 Barman は、必要なすべてのモジュールが含まれている RHEL 8 システムで利用可能な最新バージョンである Python 3.8 を引き続きサポートしています。

  参考文献 BAR-737。

22.9.2 マイナー変更

- Barman サーバーで WAL 圧縮の圧縮レベルを指定できるようにします

  新しい `compression_level` パラメーターを Barman 構成に追加します。このオプションは、有効な整数値または事前定義されたオプションのいずれか `low` 、 `medium` 、および `high` を受け入れます。各オプションは、選択した圧縮アルゴリズムに応じて異なるレベルに対応します。

参考文献 BAR-540。

- クライアントサイドの圧縮を barman-wal-archive に追加する

  クライアントサイドの圧縮オプションが barman-wal-archive に追加され、Barman サーバーで使用可能なものと同じアルゴリズムをサポートしています。有効にすると、圧縮は WAL を Barman サーバーに送信する前にクライアント側で適用されます。 --compression-level パラメーターを使用すると、選択したアルゴリズムに目的の圧縮レベルを指定できます。

  参考文献 BAR-262。

- –compression-level パラメーターを barman-cloud-wal-archive に追加します

  compression-level と呼ばれるパラメーターが barman-cloud-wal-archive に追加されました。これにより、使用中の圧縮アルゴリズムのレベルを指定できます。

  参考文献 BAR-557。

- Snappy 圧縮アルゴリズムを Barman サーバーに追加する

  Snappy 圧縮は、以前は barman-cloud-wal-archive でのみ使用できましたが、標準の Barman サーバーでも使用できるようになりました。他のすべてのアルゴリズムと同様に、 compression 構成パラメーターで snappy を設定することにより構成できます。

  参考文献 BAR-557。

- サーバープロセスをリストするための新しい list-processes サブコマンドの紹介

  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する新しい list-processes コマンドを追加します。このコマンドは、各プロセスの PID とタスクを表示します。

  参考文献 BAR-664。

- Barman サブプロセスを終了するための新しい terminate-process サブコマンドの紹介

  新しい terminate-process コマンドを追加して、ユーザーはタスク名を指定して、特定のサーバーのアクティブな Barman サブプロセスを終了できます。 Barman は、指定されたサーバーに属し、現在アクティブである限り、サブプロセスを終了します。

  参考文献 BAR-665。

- クラウドスクリプトで使用される boto3 バージョンからピンを削除します

  boto3 > = 1.36 の問題を徹底的に調査した後、バージョン 1.35 で依存関係を維持したピンを削除することにしました。

  AWS と MinIO オブジェクトストアは両方とも最新バージョンで正常に動作し、boto3 のバージョン > = 1.36 を使用すると、 Barman クラウドスクリプトが FIPS 準拠の環境で動作します。

  参考文献 BAR-637。

22.9.3 バグ修正

- 最小限の冗長性チェックでは「非インクリメンタル バックアップ」のみを考慮するようにしてください

  冗長数にインクリメンタル バックアップが含まれるため、 minimum_redundancy ルールに違反する可能性がある問題が報告されました。例として、カタログに 1 つの完全バックアップと 2 つの依存増分バックアップが含まれ、ユーザーが minimum_redundancy = 2 であったシナリオでは、ルールは誤って満たされたと考えられました。その結果、完全なバックアップを削除すると、そのインクリメンタルな依存関係のカスケード削除がトリガーされ、カタログにはバックアップがありません。

  この問題は、非インクリメンタルバックアップのみつまりフル、rsync、スナップショットのみを考慮するように minimum_redundancy ロジックを更新することにより修正されました。これにより、構成された最小の冗長レベルに違反する場合、完全なバックアップを削除できません。

  参考文献 BAR-707。

- gzip 、 bzip2 、および pigz 圧縮アルゴリズムを使用して、 --keep-compression での barman-wal-restore の使用を修正します

  --keep-compression を指定しているときに gzip 、 bzip2 または pigz で圧縮された WAL を復元しようとすると、予期しないエラーが発生する barman-wal-restore の問題を修正します。

  参考文献 BAR-722。

22.10 3.13.32025-04-24

22.10.1 バグ修正

- ブロックレベルのインクリメンタルバックアップのローカルリストアを修正する

  ブロックレベルのインクリメンタルバックアップのローカル復元を実行すると、 Barman は次のようなエラーに直面していました。

```
ERROR: Destination directory '/home/vagrant/restore/internal_no_get_wal' must be␣
↪empty
```

  そのエラーは、オプション --staging-wals-directory がバージョン 3.13.0 で導入されたときの回帰によって発生しました。それとともに、続行する前に WAL 宛先ディレクトリが空であることを確認する新しいチェックが付属しました。ただし、ブロックレベルのインクリメンタルバックアップをローカルに復元する場合、 Barman はこのチェックを実行する前に WAL 宛先ディレクトリを設定していたため、上記のエラーがトリガーされました。

  参考文献 BAR-655。

- barman-cloud-backup をフックとして実行する場合の回帰を修正

  Barman 3.13.2 は、WORM 環境の問題を修正するために提供された作業の一部として、 backup.info メタデータファイルの場所を変更しました。

  ただし、これらの変更は、 Barman サーバーでバックアップフックとして barman-cloud-backup を使用する場合に回帰を導入しました。フックは、メタデータファイルの新しい場所を認識しませんでした。

この更新ではその問題が修正されるため、 barman-cloud-backup は新しいフォルダー構造を認識し、 backup.info ファイルを適切に見つけて、実行時障害を回避します。

参考文献 BAR-696。

- カスタム圧縮が構成されている場合、部分的な WAL ファイルの解凍を回避する

  カスタム圧縮が構成されているときに Barman が部分的な WAL ファイルを解凍しようとした問題を修正しました。部分的な WAL ファイルは圧縮されないため、 barman-wal-restore または barman get-wal で --partial フラグを使用すると、それらを解凍しようとするとエラーが発生します。

  参考文献 BAR-697。

- barman-cloud-backup が一時パーツファイルをリサイクルしないを修正した

  これにより、一時パーツファイルがクラウドにアップロードされた後に削除されず、ディスク領域の枯渇につながる barman-cloud-backup 問題が修正されます。この問題は、Python > = 3.12 を使用している場合にのみ発生し、名前付け一時ファイルオブジェクトの delete 属性を削除する Python の変更によるものでした。内部チェックを実行するときに Barman が依存していました。

  参考文献 BAR-674。

- WORM 環境でのバックアップアノテーションの使用を修正

  Barman は、以前に barman keep や barman delete のような操作を追跡するために使用されるバックアップアノテーションファイルをバックアップディレクトリ自分自身内に保存していました。これらの注釈は、バックアップを保持するか、削除用にマークを付けるかを決定するのに役立ちます。ただし、WORM 環境では、バックアップディレクトリ内のファイルは、特定の期間が経過すると変更または削除できないため、バックアップ状態の管理に問題が発生しました。この修正では、アノテーションファイルを専用のメタデータディレクトリに再配置して、このような操作が WORM 環境で正しく機能することを保証します。

  参考文献 BAR-663。

22.11 3.13.22025-03-27

22.11.1 マイナー変更

- 不変ストレージを使用する際のエラーを修正する

  新しい worm_mode 構成を追加して、 Barman での Write Once ReadMany の処理を有効にし、不変ストレージへのバックアップをサポートできるようにします。

  この修正では、backup.info ファイルの自動再配置も提供します backup_directory 内の新しいディレクトリ meta 。これにより、 Barman は将来必要に応じて更新できます。

  Barman は、最初のバックアップを実行するときに必要とされない WAL ファイルの wals ディレクトリをパージしません。これにより、この最初のバックアップが時代遅れで削除されたときに再利用される追加の領域が追加されますその時までに、バックアップと WAL は保持ポリシーウィンドウの範囲外になります。

  バックアップ用の不変ストレージで実行する場合の制限を説明する追加ノートをドキュメントに追加しました。特に、ファイルの不変性における猶予期間の必要性と、 barman keep がこれらの環境ではサポートされていないという事実。

参照 BAR-649、BAR-645、BAR-650、BAR-651、BAR-652。

22.12 3.13.12025-03-20

22.12.1 マイナー変更

- バックアップショートカットの動作を改善します `last-full` / `latest-full`

  ショートカット `last-full` / `latest-full` は、サーバーの最後の完全なバックアップではなく、インクリメンタルバックアップの親としての資格のあるサーバーの最後の完全なバックアップを取得していました。

  これは予想された動作ですが、コミュニティからのフィードバックでは、ユーザーが混乱していることが示されています。

  今後、ショートカット `last-full` / `latest-full` は、そのバックアップがインクリメンタルバックアップの親として適格であるかどうかにかかわらず、 Barman サーバーの最後の完全なバックアップを取得します。

  インクリメンタルバックアップの親としてのフルバックアップの適格性は、後の手順で Barman によって検証され、親として適合しない場合には、適切なメッセージが表示されます。

  参考文献 BAR-555。

22.12.2 バグ修正

- `barman restore` の無効な `--target-time` を解析する際のエラーメッセージを修正

  `barman restore` コマンドを使用する場合、無効な `--target-time` 文字列を解析したときのエラーメッセージは次のとおりです。

  ```
  EXCEPTION: local variable 'parsed_target' referenced before assignment
  ```

  その例外は、理解可能なエラーメッセージに置き換えられました。

  参考文献 BAR-627。

- クラウド復元コマンドの相互排他的な引数を修正する

  `barman-cloud-restore` コマンドでは、 `target_tli` および `target_lsn` が相互排他的な引数であることを確認していました。ここで、チェックする正しいペアは `target_time` および `target_lsn` になります。

  参考文献 BAR-624。

- `custom_decompression_filter` を尊重しない Barman を修正

  指定された圧縮アルゴリズムが Barman によってネイティブにサポートされている場合、 Barman が設定された `custom_decompression_filter` を尊重しない問題を修正しました。 WAL ファイルを解凍するときに、カスタムフィルタがネイティブハンドラーより優先されるようになりました。

  参考文献 BAR-584。

- –no-get-wal および-standby を使用してバーマンの復元を修正する

  --no-get-wal と --standby-mode が一緒に指定されている場合、リカバリー中に Barman が pg_wal ディレクトリを削除していた問題を修正しました。この問題は、 Barman が recovery_end_command を含む pg_wal を参照するリカバリパラメーターを誤って入力したために発生し、この問題が発生しました。 Barman は、この特定のケースに必要ないため、このようなパラメーターの入力を無視します。

  リファレンス BAR-630。

- barman restore および barman-cloud-restore の引数解析の問題を修正

  Barman 3.13.0 では、回帰が導入され、 barman restore および barman-cloud-restore コマンドを使用するとエラーが発生しました。具体的には、そのバージョンでオプショナルになった backup_id 位置引数は他の引数と競合し、認識されない引数とエラーを発生させました。

  例、次のように barman-cloud-restore を実行します。

  ```
  barman-cloud-restore source_url server_name backup_id --cloud-provider aws-s3␣
  ↪recovery_dir
  ```

  次のようなエラーがトリガーされます。

  ```
  barman-cloud-restore: error: unrecognized arguments: recovery_dir
  ```

  この修正では、 backup_id を再度必須引数にすることにより問題を解決します。さらに、新しい“auto”値が backup_id として受け入れられるようになったため、 Barman は特定の backup_id を必要とせずに、復元に最適なバックアップを自動的に選択できます。この更新では、引数処理を修正しましたが、ユーザーはスムーズで柔軟な復元プロセスを可能にします。

  参考文献 BAR-596。

22.13 3.13.02025-02-20

22.13.1 重要な変更

- 新しい xlogdb_directory 構成を追加します

  新しい xlogdb_directory 構成オプションを導入します。このパラメーターは、グローバルまたはサーバーごとに設定でき、 xlog.db ファイルのカスタムディレクトリを指定できます。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、さまざまなシナリオで 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。さらに、ファイルの名前も変更されて、サーバー名が接頭辞として含まれます。

  参考文献 BAR-483。

- バックアップを復元するときに“backup_id”をオプショナルにする

  歴史的に、 Barman はバックアップを復元するには常に「backup_id」を必要とし、そのバックアップを復元のソースとして使用していました。

  この機能により、リストアのソースとして使用するバックアップを指定する必要がなくなり、オプショナルになります。

この変更は、 Barman と barman-cloud スクリプトの両方に適用されます。

これで、ユーザーは次の方法でバックアップを復元できるようになりました。

1. “backup_id”を提供します
2. “backup_id”は指定しないでください。最新のバックアップを取得します
3. “backup_id”は提供しませんが、次のようなリカバリターゲットを提供します。-“target_time” target_lsn と相互排他的“target_time”の前に最も近いバックアップを取得します。“target_lsn” -“target_tli”“target_time”または“target_lsn”と組み合わせて使用する前に、最も近いバックアップを取得します。タイムラインと一致する最新のバックアップを取得します。他の復旧ターゲットと組み合わせる場合、タイムラインと一致する target_time または target_lsn の前に最新のバックアップを取得します

リカバリターゲット `--target-xid` 、 `--target-name` および `--target-immediate` はサポートされておらず、使用するとメッセージでエラーアウトします。

この機能は、postgres クラスターを復元する際の柔軟性と簡単さを提供します。

リファレンス BAR-541、BAR-473。

22.13.2 マイナー変更

- 現在アクティブなモデルを `barman show-server` および `barman status` に追加します

  以前は、構成モデルを適用した後、サーバーで現在アクティブなモデルを確認する唯一の方法は、`barman diagnose` コマンドを使用することでした。この更新により、`barman status` および `barman show-server` コマンドは、サーバーの現在アクティブな構成モデルがある場合はそれも表示するようになりました。

  リファレンス BAR-524、BAR-400。

- `--staging-wal-directory` オプションを `barman restore` コマンドに追加して、PITR の代替 WAL ディレクトリを許可します

  新しいコマンドラインオプション `--staging-wal-directory` が `restore` / `recover` コマンドに追加され、PITR の実行時に WAL ファイルの代替の宛先ディレクトリを使用できます。以前は、WAL ファイルは復元先ディレクトリ内の `barman_wal` ディレクトリにコピーされていました。この機能強化により、リカバリ中に WAL を別のパーティションに保存するなど、柔軟性が向上します。

  参考文献 BAR-224。

- boto3 バージョンを任意のバージョン<= 1.35.99 に固定します

  Boto3 バージョン 1.36 は、S3 の完全性をチェックする方法を変更したため、このバージョンは現在の Barman コードと互換性がなく、次のエラーが生成されます。

  PutObject オペレーションを呼び出すときにエラー MissingContentLength が発生しました

  一時的な回避策として、 boto3 のバージョンは、 Barman で 1.36 のサポートが実装されるまで、バージョン<= 1.35.99 に固定されます。

  参考文献 BAR-535。

- 重複した WAL ファイルを処理する際に barman-wal-archive をよりスマートに

  一部の特殊なケースでは、Postgres は同じ WAL を 2 回アーカイブしようとする場合があります。例えば barman-wal-archive は WAL ファイルを Barman ホストにコピーしますが、スクリプトは Postgres に成功を報告する前に中断された場合。 WAL は既に Postgres から Barman にコピーされていますが、非同期 Barman WAL アーカイバーによってまだ処理されていないため、同じファイルを再度アーカイブしようとすると、 barman-wal-archive の新しい実行が失敗する場合があります。

  このマイナーな変更は、既存のファイルと着信ファイルのチェックサムを確認することにより、この状況に対処します。チェックサムが一致する場合、着信ファイルは無視され、そうでない場合、出力情報メッセージが送信され、着信ファイルは errors ディレクトリに移動されます。コードは、どちらの状況も 0 で終了し、 archive_command の障害が原因で Postgres ホストに WAL が蓄積するのを回避します。

  参考文献 BAR-225。

- WAL アーカイブ障害チェックをクリアするための文書手順

  Barman ドキュメントの再デザイン中に、サーバーがアイドル状態で barman check が「WAL アーカイブ」で失敗を返した場合、 switch-wal コマンドを実行するようにユーザーにアドバイスするノートの追加を忘れていました。

  これにより、以前のドキュメントから残ったギャップが解消されます。

  参考文献 BAR-521。

- 可能であれば、ディレクトリ全体を一度に削除して WAL を削除します

  以前は、WAL ファイルを削除する必要がある場合、バックアップの削除など、 Barman はすべての WAL ファイルを反復し、個別に削除していました。これにより、主に ZFS ファイルシステムを使用するシステムでパフォーマンスの問題が発生する可能性があります。この変更により、ディレクトリ内のすべてのファイルが Barman によって必要とされなくなったことに気づくたびに、ディレクトリ全体が削除されます。

  参考文献 BAR-511。

- Azure 認証での DefaultAzureCredential オプションのサポートを追加します

  ユーザーは、サーバー構成の azure_credential の default オプションまたは barman-cloud-* の場合は --azure-credential default オプションを使用することにより、認証に Azure の DefaultAzureCredential を明示的に使用できるようになりました。以前は、資格情報が提供されず、環境変数が設定されていない場合に、フォールバックとしてのみ設定できました。

  参考文献 BAR-539。

- リテンションポリシー情報の診断出力を改善します

  barman promote コマンドの出力を改善して、より使いやすい文字列表現を表示します。具体的には、`retention_policy' 属性の「redundancy 2 b」の代わりに「REDUNDANCY 2」が表示され、`wal_retention_policy' 属性は「simple-wal 2 b」の代わりに「MAIN」が表示されます。

  参考文献 BAR-100

22.13.3 バグ修正

- barman restore を --target-tli とともに使用する場合の PITR を修正

  Barman は、次の場合に recovery.signal を作成せず、postgresql.auto.conf に recovery_target_timeline を埋めていません。

  - barman restore に渡された唯一のリカバリターゲットは --target-tli でした。または
  - --target-tli は他の --target-* オプションで指定されましたが、指定されたターゲットタイムラインは、選択したバックアップのタイムラインと同じでした。

  --target-* オプションが barman restore に渡される場合、それは PITR として正しく扱われるようになりました。

  参考文献 BAR-543。

- 割り当て前に AWS `profile`変数が参照される場合のバグを修正

  問題は、Barman 3.12.0 リリースの一部として BAR-242 によって導入されました。この問題により、--aws-profile 引数または aws_profile 構成が設定されていない場合、barman-cloud-backup-delete および場合によっては他のコマンドが次のようなエラーで失敗します。

```
ERROR: Barman cloud backup delete exception: local
variable 'profile' referenced before assignment`
```

  参考文献 BAR-518。

- barman-cloud-wal-archive の-zstd フラグを修正する

  barman-cloud-wal-archive の --zstd フラグのバグを修正しました。基本的に無視され、アップロード前に WAL ファイルを実際に圧縮していません。

  参考文献 BAR-567。

22.14 3.12.12024-12-09

22.14.1 バグ修正

- json 出力にバックアップの開始時間と終了時間の isoformat フィールドを追加します

  このパッチは、infofile オブジェクトの json 出力を変更し、2 つの新しいフィールド begin_time_iso および end_time_iso を追加します。新しいフィールドにより、より標準のタイムゾーンを認識する時間形式を使用でき、以前のバージョンとの互換性を維持できます。将来的には、日付の iso 形式がバーマンが日付を保存するために使用する標準になり、人間が判読できない出力が要求される場所はどこでも使用されることに注意してください。

  作業の一部として、このパッチは、Barman 3.12.0 で導入された BAR-316 を元に戻します。

  参考文献 BAR-494。

22.15 3.12.02024-11-21

22.15.1 マイナー変更

- Barman に FIPS サポートを追加する

  md5 ハッシュアルゴリズムは FIPS に準拠していないため、 sha256 に置き換えられます。 sha256 は FIPS に準拠しており、広く使用されており、ほとんどの実用的な目的において安全であると考えられています。このリリースまでは、Barman の WAL アーカイブクライアントは hashlib.md5 を使用して、Barman サーバーに送信される前に tar ファイルのチェックサムを生成していました。ここで、tar ファイルは、複数のファイルをバンドルするために使用されるファイル形式であり、チェックサムとそれに対応するパスをリストする MD5SUMS ファイルです。このリリースでは、 md5 ハッシュアルゴリズムは、デフォルトとして sha256 に置き換えられました。その結果、tar ファイルのチェックサムは sha256 を使用して計算され、 MD5SUMS ファイルの名前は SHA256SUMS になります。 Barman は、ユースケースがある場合、クライアントからデフォルト以外の md5 アルゴリズムと MD5SUMS ファイルを使用する機能があります。ユーザーは、 --md5 フラグを barman-wal-archive archive_command に追加するだけです。

  参考文献 BAR-155、CP-34954、CP-34391

- el7、debian10、および ubuntu1804 サポートを削除しました。 Debian と SLES を更新しました。

  el7、debian10、および ubuntu1804 のサポートが削除されました。さらに、バージョン 12 とバージョン名「bookworm」が Debian に追加され、以前に欠落していたエントリに対処しました。 SLES イメージバージョンも sp4 から sp5 に更新されました。

  参考文献 BAR-389。

- Postgres Extended 17PGE および Postgres Advanced Server 17EPAS のサポートを追加します

  テストは Postgres Extended 17PGE および Postgres Advanced Server 17EPAS で実施され、 Barman の最新機能との完全な互換性が確認されました。この検証により、PGE および EPAS の最新バージョンのユーザーは、 Barman のすべての新機能を自信を持って活用できます。

  参考文献 BAR-331。

- zstd 、 lz4 、および xz アルゴリズムで WAL 圧縮を向上させる

  WAL ファイルの xz 圧縮のサポートを導入しました。 compression サーバーパラメーターで xz を指定することにより、有効にできます。 WAL は、Barman の WAL アーカイブに入るときに圧縮されます。クラウドの場合、 barman-cloud-wal-archive の実行時に --xz を指定することにより有効にできます。

  WAL ファイルでの zstandard 圧縮のサポートを導入しました。 compression サーバーパラメーターで zstd を指定することにより、有効にできます。 WAL は、Barman の WAL アーカイブに入るときに圧縮されます。クラウドの場合、 barman-cloud-wal-archive の実行時に --zstd を指定することにより有効にできます。

  WAL ファイルの lz4 圧縮のサポートを導入しました。 compression サーバーパラメーターで lz4 を指定することにより、有効にできます。 WAL は、Barman の WAL アーカイブに入るときに圧縮されます。クラウドの場合、 barman-cloud-wal-archive の実行時に --lz4 を指定することにより有効にできます。

リファレンス BAR-265、BAR-423、BAR-264。

- マルチパートアップロードを回避することにより、S3 バケットでの WAL アップロードパフォーマンスを向上させます

  以前は、WAL ファイルは、 `upload_fileobj` メソッドを介して boto3 ライブラリが提供するマルチパートアップロードを使用して S3 バケットにアップロードされていました。マルチパートアップロードは、WAL セグメントなどの小さなファイルに使用すると、単一の PUT 要求でアップロードする場合と比較して、時間がかかることに注意しました。これは、100MB 未満のファイルのマルチパートアップロードを回避することにより改善されました。この変更により、各 WAL ファイルの平均アップロード時間は約 15%削減される予定です。

  参考文献 BAR-374。

- `KEEP:STANDALONE` フルバックアップのリテンションポリシーを適用する際の動作を変更します

  `backup_method = postgres` で作成されたフルバックアップにリテンションポリシーを適用する場合、 Barman は以前に、使用される KEEP ターゲットに関係なく、すべての依存子インクリメンタルバックアップを `VALID` としてマーキングしていました。ただし、このアプローチは間違っています。

  - `KEEP:STANDALONE` というラベルのバックアップの場合、Barman は、サーバーをそのバックアップの正確な状態に復元するために必要な WAL ファイルのみを保持します。これらのバックアップは自己完結型であるため、ルートバックアップが保持ポリシーの範囲外になると、依存子のバックアップは必要ありません。
  - 対照的に、 `KEEP:FULL` とマークされたバックアップは、ポイントインタイムリカバリーを目的としています。これをサポートするために、 Barman はすべての WAL と子バックアップを保持して、バックアップの一貫性を保証し、可能な最新のポイントへのリカバリーを可能にします。

  この違いにより、 `KEEP:STANDALONE` バックアップは特定の瞬間のスナップショットとして機能しますが、 `KEEP:FULL` バックアップは完全なポイントインタイムリカバリーに必要なすべてを保持します。

  参考文献 BAR-366。

- Barman の復旧プロセスのドキュメントとユーザー向け機能を更新します。

  Barman ドキュメントとツール自分自身は、両方とも以下を参照するために「recover」/「recovery」という用語を使用していました。

  - Postgres の復旧プロセス。
  - バックアップを復元し、リカバリーの準備をするプロセス。

  コードとドキュメントの両方が変更され、「restore」および「recover」/「recovery」という用語の使用を正確に反映しています。

  また、 `barman recover` コマンドは `barman restore` に名前が変更されました。古い名前は、下位互換性のためのエイリアスとして残されています。

  参考文献 BAR-337。

- –keep-compression フラグを barman-wal-restore および get-wal に追加します

  新しい `--keep-compression` オプションが `barman-wal-restore` と `get-wal` の両方に追加されました。このオプションは、圧縮された WAL ファイルを取得する前に Barman サーバーで解凍するか

どうかを制御します。 get-wal で指定される場合、デフォルトの解凍はスキップされ、出力は元の状態の WAL ファイルコンテンツです。 barman-wal-restore で指定すると、WAL ファイルはそのまま取得され、圧縮されている場合はクライアントサイドで解凍されます。

参考文献 BAR-435。

- ランサムウェア保護-AWS スナップショットロックのサポートを追加

  Barman は、Amazon EBS スナップショットの誤ってまたは悪意のある削除を防ぐための新しい統合機能である AWS EBS Snapshot Lock をサポートするようになりました。スナップショットがロックされている場合、ユーザーは削除できませんが、完全にアクセスして使用できます。この機能を使用すると、スナップショットを指定した期間 Write-Once-Read-Many 形式で保存できます。ロックの有効期限が切れるまでデータを安全に改ざん防止に保つことにより、規制要件を満たすのに役立ちます。

  この機能を開始したコミュニティ貢献者 Rui Marinho に心より感謝いたします。

  参考文献 BAR-242。

- バックアップの削除中のクラッシュから孤立したファイルが残されないようにします

  このコミットは、バックアップの削除中にシステムがクラッシュした場合、バックアップにファイルが残る問題を修正します。

  バックアップが削除されると、最初に「削除マーカー」を取得します。バックアップの削除中にクラッシュが発生した場合、マーカーは、サーバー再起動時に不完全なバックアップ削除を認識するのに役立ちます。

  Barman cron ジョブは、これらの削除されたマーカーを検索するように更新されました。「削除マーカー」を含むバックアップが見つかった場合、プロセスを完了します。

  参考文献 BAR-244。

- スナップショットでタグを使用するサポートを追加します

  Barman は、 barman-cloud-backup script コマンドを使用してバックアップを作成するときに、スナップショットのタグ付けをサポートするようになりました。 –tags と呼ばれる新しい引数が追加されました。

  この機能を開始したコミュニティ貢献者 Rui Marinho に心より感謝いたします。

  参考文献 BAR-417。

- Barman クラウドコマンドの JSON 出力を生成するときは、ctime の代わりに ISO 形式を使用します

  ctime 形式には、タイムスタンプに関連付けられたタイムゾーンに関する情報がありません。それに加えて、そのフォーマットは人間が使用するのにより適しています。マシンでの使用には、ISO 形式の方が適しています。

  参考文献 BAR-316。

22.15.2 バグ修正

- レプリケーションスロットに対して誤った結果を返すバーマンチェックを修正

  以前は、スタンバイノードからバックアップし、プライマリから WAL をストリーミングするアーキテクチャを使用している場合、 Barman はレプリケーションチェックに conninfo スタンバイサーバーをポイントして誤って使用し、次のようなエラーを引き起こしていました。

  ```
  replication slot (WAL streaming):  FAILED (replication slot 'barman' doesn't
  exist. Please execute 'barman receive-wal --create-slot pg17')
  ```

  これにより、 wal_conninfo が設定されている場合は WAL レプリケーションチェックに使用されるようにすることにより、次の問題 #1024 _が修正されます。

  両方が設定されている場合、 wal_conninfo は wal_streaming_conninfo より優先されます。この変更により、 wal_conninfo のみが設定されている場合、それが使用され、 conninfo にフォールバックしません。

  また、ドキュメントでは、 conninfo がスタンバイサーバーをポイントする場合、正確なレプリケーションステータスチェックには wal_conninfo を設定および使用する必要があることを明示的に変更しました。

  参考文献 BAR-409。

- barman keep の欠落しているオプションを修正

  オプションなしで barman keep を実行したときに Barman CLI が出力したエラーメッセージは、status と release のショートカットエイリアスが存在することを示唆していました。これらのエイリアス-s および-r は存在しないため、エラーメッセージは誤解を招くものでした。これは、これらの短いオプションを Barman CLI に含め、これらのショートカットが既に存在する barman-cloud-backup-keep のような他のツールと調整することにより、問題を修正します。

  参考文献 BAR-356。

- conninfo および primary_conninfo に関連するスタンバイチェックを軽減する

  スタンバイサーバーをバックアップするときに、Barman はいくつかのチェックを実行して、conninfo が実際にスタンバイリカバリモードを指していること、および primary_conninfo がプライマリ非リカバリーを指していることをアサートします。

  問題#704 および#744 で報告されたように、フェールオーバーが発生すると、 conninfo が代わりにプライマリを指すようになり、チェックが失敗し始めるため、ユーザーはフェールオーバーが発生するたびに Barman 構成を手動で変更する必要があることです。

  この修正では、このようなチェックを非クリティカルにすることにより問題を解決しました。つまり、チェックはまだ失敗しますが、 Barman は関係なく動作し続けます。基本的に、 Barman は conninfo がスタンバイをポイントしていない場合、 primary_conninfo を無視します。この構成の誤りに関する警告は、 Barman コマンドを実行するたびに出力されるため、ユーザーが気づくことができます。

  参考文献 BAR-348。

- Barman で pg_has_role を呼び出すときに MEMBER ではなく USAGE を確認する

  正しく動作するには、Barman データベースユーザーを一部のロールに含める必要があります。 Barman は、Postgres で pg_has_role を呼び出すことにより、条件が満たされることを確認していました。た

だし、 USAGE ではなく MEMBER 特権のチェックでした。この見落としは修正されました。

この変更は、@RealGreenDragon からの貢献です。

参考文献 BAR-489。

22.16 3.11.12024-08-22

22.16.1 バグ修正

- barman-cloud-backup-delete の障害を修正します。このコマンドは、以前のリリースで導入されたバグのため、保持ポリシーを適用するときに失敗していました。

22.17 3.11.02024-08-22

22.17.1 重要な変更

- Postgres 17+のインクリメンタルバックアップのサポートを追加します。この主要機能は、いくつかの小さな変更で構成されています。
  - --incremental コマンドラインオプションを barman backup コマンドに追加します。これは、インクリメンタルバックアップを作成するときに親バックアップを指定するために使用されます。親は、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。
  - latest-full ショートカットバックアップ ID を追加します。 latest と同様に、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を選択するショートカットとして使用できます。 latest は、フルまたはインクリメンタルの場合を独立して取得しますが、 latest-full は最新のフルバックアップを取得します。
  - barman keep コマンドは、 backup_method = postgres の場合のフルバックアップにのみ適用できます。
  - 保持ポリシーでは、インクリメンタル バックアップは考慮されません。インクリメンタル バックアップは、完全なバックアップまでで使用可能なバックアップの完全なチェーンがないと復旧できないため、完全なバックアップのみが保持ポリシーを考慮します。フルバックアップに依存するインクリメンタルバックアップがあり、リテンションポリシーが適用される場合、フルバックアップはそのステータスを、関連するインクリメンタルバックアップに伝播します。フルバックアップに KEEP ターゲットでフラグが設定されている場合、 Barman は関連するすべてのインクリメンタルバックアップのステータスを VALID に設定します。
  - バックアップを削除すると、それに依存するすべての増分バックアップがある場合は、それも削除されます。
  - barman recover は、リカバリ時に完全バックアップとインクリメンタルバックアップのチェーンを組み合わせる必要があります。新しい CLI オプション --local-staging-path 、および対応する local_staging_path 構成オプションは、インクリメンタルバックアップをリカバリーするときにバックアップが結合される Barman ホスト内のパスを指定するために使用されます。
- barman show-backup 出力の変更

- 「推定クラスターサイズ」フィールドを追加します。バックアップを復元するときに、クラスターのデータディレクトリのサイズを推定すると役立ちます。これは、圧縮バックアップまたはインクリメンタルバックアップを復旧する場合、バックアップのサイズが Postgres のデータディレクトリのサイズを反映していない場合に特に役立ちます。 JSON 形式では、これは `cluster_size` として保存されます。
- 「WAL resumer」フィールドを追加します。このフィールドは、バックアップが作成されたときに Postgres で `summarize_wal` が有効になっていたかどうかを示します。 JSON 形式では、これは `server_information.summarize_wal` として保存されます。このフィールドは、Postgres 16 以前では省略されます。
- 「Data checksums」フィールドを追加します。これは、バックアップが取得されたときに Postgres で `data_checkums` が有効になっていたかどうかを示します。 JSON 形式では、これは `server_information.data_checksums` として保存されます。
- 「バックアップ方法」フィールドを追加します。これは、このバックアップに使用されたバックアップ方法を示します。 JSON 形式では、これは `base_backup_information.backup_method` として保存されます。
- フィールド「Disk Capacity」の名前を「Backup Size」に変更します。後者は、 Barman ホストのバックアップのサイズを表すより包括的な名前を提供します。 `base_backup_information` の下の JSON フィールドの名前も `disk_usage` から `backup_size` に変更されました。
- 「WAL サイズ」フィールドを追加します。これは、バックアップに必要な WAL のサイズを示します。 JSON 形式では、これは `base_backup_information.wal_size` として保存されます。
- フィールド「インクリメンタルサイズ」をリファクタリングします。これは名前が“Resources Saving”になり、 `rsync` または `pg_basebackup` でインクリメンタルバックアップを取得したときに節約されるリソースの推定値を表示するようになりました。バックアップサイズと推定クラスターサイズを比較して、インクリメンタルバックアップを行うことにより保存されたディスクリソースとネットワークリソースの量を推定します。 JSON 形式では、フィールドの名前が `incremental_size` から `base_backup_information` の下の `resource_savings` に変更されました。
- `system_id` フィールドを JSON ドキュメントに追加します。このフィールドには、Postgres のシステム識別子が含まれています。コンソール形式では存在しましたが、JSON 形式ではありませんでした。
- Postgres 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を追加します。
  - “Backup type” Postgres バックアップがフルかインクリメンタルかを示します。 JSON 形式では、これは `base_backup_information` の下の `backup_type` として保存されます。
  - 「ルートバックアップ」1 つ以上のインクリメンタルバックアップのチェーンのルートである完全バックアップの ID。 JSON 形式では、これは `catalog_information.root_backup_id` として保存されます。
  - 「親バックアップ」 この増分バックアップが取得された完全または増分バックアップの ID。 JSON 形式では、これは `catalog_information.parent_backup_id` として保存されます。
  - “Children Backup(s)” このバックアップを親として取得したインクリメンタルバックアップの ID。 JSON 形式では、これは `catalog_information.children_backup_ids` として保存されます。

    * 「バックアップチェーンサイズ」このインクリメンタルバックアップからルートバックアップまでのチェーン内のバックアップの数。 JSON 形式では、これは catalog_information.chain_size として保存されます。

- barman list-backup 出力の変更
  - JSON 出力にバックアップタイプが含まれるようになりました。これは、 rsync で取得したバックアップの場合は rsync 、 pg_basebackup で取得したバックアップの場合は full または incremental 、またはクラウドスナップショットの場合は snapshot です。コンソールに出力する場合、バックアップタイプは、対応するラベル R 、 F 、 I または S で表されます。
  - 出力からテーブルスペース情報を削除します。それは出力が肥大化していました。表領域情報は、 barman show-backup の出力で引き続き見つかります。
- recovery_target_time 〜 barman recover を構成するときは、常にタイムゾーンを使用してタイムスタンプを設定します。以前は、 --target-time を介してタイムゾーンが明示的に設定されていない場合、 Barman は Postgres でタイムゾーンなしで recovery_target_time を構成していました。タイムゾーンがないと、Postgres は Postgres の timezone GUC を介して構成されているものを想定しています。今後、 Barman は --target-time オプションを介してユーザーがタイムゾーンを設定していない場合、 Barman ホストのタイムゾーンで recovery_target_time を構成します。
- --target-lsn が設定されている場合、「no get wal」アプローチでバックアップをリカバリーする場合、構成したターゲットに到達するために必要な WAL ファイルのみをコピーします。以前は、 Barman がアーカイブから Postgres にすべての WAL ファイルをコピーしていました。
- --target-immediate が設定されている場合、「no get wal」アプローチでバックアップをリカバリーする場合、一貫性のあるポイントに到達するために必要な WAL ファイルのみをコピーします。以前は、 Barman がアーカイブから Postgres にすべての WAL ファイルをコピーしていました。
- barman-wal-restore は、WAL をコピーする代わりに、スプールディレクトリから pg_wal に移動するようになりました。これにより、spool ディレクトリと pg_wal ディレクトリが同じパーティションにある場合、パフォーマンスが向上します。
- barman check-backup は、バックアップが出力とログで FAILED としてマークされた理由を表示するようになりました。以前は、バックアップが FAILED としてマークされた理由を知るには、 barman show-backup コマンドを実行する必要がありました。
- 構成オプション aws_await_snapshots_timeout と、対応する --aws-await-snapshots-timeout コマンドラインオプションを barman-cloud-backup に追加します。これは、スナップショットバックアップが完了状態に到達するまで待機するタイムアウトを秒単位で指定します。
- rsync ベースのバックアップにキープアライブメカニズムを追加します。以前は、pg_backup_start() および pg_backup_stop() を実行するために Barman によって作成された Postgres セッションは、ベースバックアップコピーがかかる間アイドル状態のままでした。これにより、ファイアウォールまたはルーターが長時間アイドル状態であったために接続がドロップされる可能性があります。キープアライブメカニズムは、その接続を介してハートビートクエリを Postgres に送信するため、接続がドロップされる可能性が減少します。ハートビート間の間隔は、新しい構成オプション keepalive_interval と、 barman backup コマンドの対応する CLI オプション --keepalive-interval を介して制御できます。

22.17.2 バグ修正

- 「no get wal」アプローチと `--target-time` 設定を使用してバックアップをリカバリーする場合、すべての WAL ファイルをコピーします。以前は、 Barman は、Postgres が構成したターゲット時間に到達するために必要な WAL ファイルを「推測」しようとしました。ただし、このメカニズムは、 Barman ホストの WAL ファイルの統計、具体的には作成時間に基づいていたため、十分に堅牢ではありませんでした。例えば Postgres と Barman の間でアーカイブまたはストリーミングの遅延があった場合、ファイル統計に基づいたチェックが弱いために Barman は必要なすべての WAL ファイルのコピーを逃すため、リカバリーが失敗するのに十分である可能性があります。
- Python 3.6 以前で Barman を実行する場合、`python-snappy` を `0.6.1` に固定します。 `python-snappy` の新しいバージョンには、 `cramjam` バージョン `2.7.0` 以降が必要です。これらは Python 3.7 以降でのみ使用できます。
- `barman receive-wal` は、次の場合に `0` ではなくコード `1` で終了するようになりました。
  - `pg_receivewal` が実行されているため、 `--reset` フラグを使用して実行できません。
  - `pg_receivewal` プロセスは既に実行されているため、開始できません。
- `barman diagnose` 出力の Python に関する情報を修正および改善します。
  - このコマンドは、`python_ver` JSON キーを介して Python バージョンを出力するときに、Barman がインストールされているのと同じ Python インタープリターを使用するようになりました。以前は、環境に複数の Python インストールおよび/または仮想環境が含まれる場合、別の Python インタープリターから取得される可能性があるため、最終的に出力が誤解を招く可能性がありました。
  - `python_executable` キーを JSON 出力に追加しました。これには、 Barman が使用する正確な Python インタープリターへのパスが含まれています。

22.18 3.10.12024-06-12

22.18.1 バグ修正

- 一部のプラットフォームでのインストールの問題を回避するには、 `argcomplete` をオプショナルにします。
- ファイルが存在する場合にのみ `barman.auto.conf` をロードします。
- `cfg_changes.queue` ファイルの形式が不正な場合に警告を発します。
- `pg_checkpoint` が利用可能な postgresql バージョンのドキュメントを修正します。
- `--no-partial` オプションを `barman-cloud-wal-restore` に追加します。

22.19 3.10.02024-01-24

22.19.1 重要な変更

- 新しい CLI オプション `--max-bandwidth` で設定された値に従って、AWS S3 または Azure Blob Storage にバックアップするときに `barman-cloud-backup` が使用する平均帯域幅を制限します。

- 新しい構成オプション lock_directory_cleanup を追加します。これにより、cron は、未使用のロックファイルから barman_lock_directory を自動的にクリーンアップします。
- model と呼ばれる新しいタイプの構成のサポートを追加します。モデルは、特定の Barman サーバーの構成オプションのオーバーライドのセットとして機能します。
- JSON を使用して構成の作成と更新を可能にする新しい barman コマンド barman config-update を追加します

22.19.2 バグ修正

- Barman のフックスクリプトとして barman-cloud-backup を使用する場合、 --min-chunk-size が無視される原因になったバグを修正します。

22.20 3.9.02023-10-03

22.20.1 重要な変更

- ユーザーが pg_checkpoint ロールを持っている場合、PG>=14 に対して barman switch-wal --force を実行できます。このパッチは toydarian に感謝します。
- チェックタイムアウトが発生すると、現在のチェックを info レベルでログに記録します。
- S3 または Azure Blob Storage で barman-cloud-backup を使用する場合のアップロードチャンクの最小サイズを、 --min-chunk-size オプションを使用して指定できるようになりました。
- backup_compression = none は、 pg_basebackup を使用する場合にサポートされます。
- PostgreSQL 15 以降の場合 zstd および lz4 の許可される backup_compression_level 値が、pg_basebackup で許可される値と一致するように更新されました。
- 15 より前の PostgreSQL バージョンの場合 backup_compression_level = 0 を backup_compression = gzip とともに使用できるようになりました。

22.20.2 バグ修正

- 新しいプロセスを開始するときにマルチプロセッシングがデフォルトでスポーンを使用するプラットフォームで barman recover を修正します。

22.21 3.8.02023-08-31

22.21.1 重要な変更

- パッケージのインストールを明確にする。 barman は、各オペレーティングシステムのデフォルトの Python バージョンがパッケージされています。
- minimum-redundancy オプションが barman-cloud-backup-delete に追加されます。常に使用可能なバックアップの最小数を設定できます。

- 新しい primary_checkpoint_timeout 構成オプションを追加します。新しい WAL ファイルが生成されない場合、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、バックアップの最後に Barman が待機する秒数を定義できます。

22.21.2 バグ修正

- バーマンリテンションポリシーアプリケーションの競合状態を修正します。要求されたバックアップで別の削除が進行中の場合、バックアップの削除で警告が発生するようになりました。
- barman-cloud-backup-show のマニュアルページのインストールを修正します。

22.22 3.7.02023-07-25

22.22.1 重要な変更

- EBS ボリュームを使用した AWS でのスナップショットバックアップのサポートが追加されました。
- barman-cloud-* スクリプトの --profile オプションの名前が --aws-profile に変更されます。古い名前は非推奨であり、将来のリリースでは削除される予定です。
- backup_method = rsync で作成されたバックアップの完了時に、バックアップマニフェストを自動的に生成できるようになりました。これは、 autogenerate_manifest 構成変数を設定することにより有効になり、 --manifest および --no-manifest CLI オプションを使用してオーバーライドできます。

22.22.2 バグ修正

- barman-cloud-* スクリプトは、継続トークンを正しく使用して、AWS S3 互換オブジェクトストアのオブジェクトをページングするようになりました。これにより、 barman-cloud-backup-delete がバックアップの削除後に最も古い 1000 の対象 WAL のみを削除するバグが修正されます。
- マイナーなドキュメントの修正。

22.23 3.6.02023-06-15

22.23.1 重要な変更

- PostgreSQL バージョン 10 はサポートされなくなりました。
- 管理対象ディスクを使用した Microsoft Azure でのスナップショットバックアップのサポートが追加されました。
- --snapshot-recovery-zone オプションは、他のプロバイダー固有のオプションとの一貫性を保つために --gcp-zone に名前変更されました。古い名前は非推奨であり、将来のリリースでは削除される予定です。
- snapshot_zone オプションと --snapshot-zone 引数の名前は、それぞれ gcp_zone と --gcp-zone に変更されます。古い名前は非推奨であり、将来のリリースでは削除される予定です。

- snapshot_gcp_project オプションと --snapshot-gcp-project 引数の名前が gcp_project および --gcp-project に変更されます。古い名前は非推奨であり、将来のリリースでは削除される予定です。

22.23.2 バグ修正

- Barman は、パッシブノードに対して replication-status コマンドを実行しようとしなくなりました。
- backup_label は、 barman-cloud-backup-delete でスナップショットバックアップが削除されるときに、クラウドストレージから削除されます。
- generate-manifest および verify-backup コマンドのマニュアルページが追加されました。
- マイナーなドキュメントの修正。

22.24 3.5.02023-03-29

22.24.1 重要な変更

- Python 2.7 はサポートされなくなりました。サポートされている最も古い Python バージョンは 3.6 になりました。
- EL7 の barman 、barman-cli および barman-cli-cloud パッケージでは、python 2.7 ではなく python 3.6 が必要になりました。サポートされている他のプラットフォームの場合、 Barman パッケージには既に Python バージョン 3.6 以降が必要であるため、パッケージ化は影響を受けません。
- PostgreSQL 10 のサポートは、将来の Barman リリースで廃止されます。 3.5.x は、PostgreSQL 10 をサポートする Barman の最後のバージョンです。
- barman-cloud-backup または barman-cloud-wal-archive で --kms-key-name オプションを使用することにより、Google Cloud Storage にアップロードされたバックアップと WAL を特定の KMS キーを使用して暗号化できるようになりました。
- AWS S3 にアップロードされたバックアップと WAL は、 barman-cloud-backup または barman-cloud-wal-archive とともに --sse-kms-key-id オプションを使用することにより、特定の KMS キーを使用して暗号化できるようになりました --encryption=aws:kms 。
- 2 つの新しい構成オプションが提供され、 backup_method = rsync を使用したバックアップおよびリカバリー中にパラレルワーカーが起動されるレートを制限できます。parallel_jobs_start_batch_size は、単一のバッチで起動されるパラレルワーカーの量を制限するように設定でき、 parallel_jobs_start_batch_period は、ワーカーの単一バッチが開始される秒単位の時間を定義できます。これらは、 barman backup および barman recover コマンドで引数 --jobs-start-batch-size および --jobs-start-batch-period を使用してオーバーライドできます。
- 新しいオプション --recovery-conf-filename が barman recover に追加されます。これは、Barman が PostgreSQL リカバリオプションを書き込むファイルをデフォルトの postgresql.auto.conf から別の場所に変更するために使用できます。

22.24.2 バグ修正

- barman-cloud-backup-show が、 barman backup で作成され、 barman-cloud-backup によってバックアップ後フックスクリプトとしてアップロードされたバックアップのバックアップメタデータを表示できないバグを修正しました。
- バックアップ圧縮設定の検証に使用される PostgreSQL 接続が、 Barman コマンドが終了するまで開いたままになるバグを修正しました。
- idle_session_timeout を超える期間実行すると、rsync-concurrent バックアップが失敗する原因となった問題を修正します。
- --wait フラグが barman backup で使用された場合、バックアップ名がバックアップメタデータに保存されないバグを修正しました。
- mojtabash78、mhkarimi1383、epolkerman、barthisrael、および hzetters の貢献に感謝します。

22.25 3.4.02023-01-26

22.25.1 重要な変更

- これは、Python 2 をサポートする Barman の最後のリリースであり、新機能には Python 3.6 以降が必要です。
- snapshot という名前の新しい backup_method が追加されます。これにより、クラウドストレージボリュームのスナップショットを取得することにより、バックアップが作成されます。現在、Google Cloud Platform のみがサポートされていますが、将来の Barman リリースでは、AWS および Azure のサポートが続く予定です。この機能には、Python 最小バージョン 3.7 が必要であることに注意してください。詳細については、 Barman のマニュアルを参照してください。
- スナップショットバックアップのサポートも barman-cloud-backup に追加され、スナップショットバックアップのリストアの最小限のサポートが barman-cloud-restore に追加されます。
- 新しいコマンド barman-cloud-backup-show が追加されました。これは、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メタデータを表示し、 barman show-backup に似ています。これは、復元時にスナップショットのメタデータを簡単に取得できるように提供されていますが、barman-cloud-backup で作成されたバックアップのメタデータを検査する便利な方法でもあります。
- ドキュメントの RPM から Barman をインストールする手順が更新されました。
- ドキュメントの NFS 要件のフォーマットが修正されました。
- サポートされている PostgreSQL バージョンは、ドキュメントで更新されています。これはドキュメントの修正のみです。サポートされている最小メジャーバージョンはまだ 10 です。

22.26 3.3.02022-12-14

22.26.1 重要な変更

- barman backup および barman-cloud-backup コマンドでサポートされた新しい --name オプションを使用して、バックアップ時にバックアップに名前を付けることができるようになりました。バックアップと対話する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名を使用

できます。さらに、バックアップをリストおよび表示するコマンドが更新され、プレーンテキストおよび JSON 出力形式でバックアップ名が含まれるようになりました。

- PostgreSQL バージョンの厳密なチェックにより、 Barman がサポートされている PostgreSQL バージョンに対して実行されていることを確認します。

22.26.2 バグ修正

- barman Cloud コマンドドキュメントとこれらのコマンドのヘルプ出力間の不一致を修正します。
- スタンバイのバックアップ中にプライマリで WAL を切り替えるときは、新しい PostgreSQL 接続を使用して、 `SSL error` メッセージや接続の失敗などの未定義の動作を回避します。
- 子プロセスで実行されるコマンドの stdout のデフォルトログレベルを `DEBUG` に変更することにより、ログボリュームを削減します。ただし、 `pg_basebackup` は除きます。これは、出力を確認することが頻繁に役立つ長時間実行プロセスであるため、 `INFO` レベルで意図的にログに記録されますコマンドの実行中)。

22.27 3.2.02022-10-20

22.27.1 重要な変更

- `barman-cloud-backup-delete` は、単一の要求で削除されるオブジェクトの最大数を決定する `--batch-size` オプションを受け入れるようになりました。
- すべての `barman-cloud-*` コマンドが `--read-timeout` オプションを受け入れるようになりました。これは `aws-s3` クラウドプロバイダーで使用される場合、S3 への要求を行うときに boto3 ライブラリが使用する読み取りタイムアウトを決定します。

22.27.2 バグ修正

- Barman 構成で `backup_compression` が設定されているが、PostgreSQL サーバーが利用できない場合の `barman recover` の障害を修正します。

22.28 3.1.02022-09-14

22.28.1 重要な変更

- `backup_method = postgres` で取得したバックアップを、それぞれ `backup_compression = lz4` または `backup_compression = zstd` を設定することにより、lz4 および zstd 圧縮を使用して圧縮できるようになりました。これらのオプションは、PostgreSQL 15 ベータ以降でのみサポートされています。
- 並列圧縮に使用されるスレッドの数を設定する新しいオプション `backup_compression_workers` が利用できます。これは、現在 `backup_method = postgres` および `backup_compression = zstd` でのみ使用できます。

- 新しいオプション primary_conninfo を設定して、バックアップのファイナライズ時にプライマリで WAL スイッチが発生するのをスタンバイのバックアップが待機する必要を回避できます。 Barman は、primary_conninfo の接続文字列を使用して、バックアップを停止するときにプライマリで WAL スイッチを実行します。
- --files-from 引数の末尾の改行を必要とする CVE-2022-29154 にパッチされた特定の Rsync バージョンのサポート。
- barman receive-wal メンテナンスオプション --stop 、 --reset 、 --drop-slot および --create-slot を非アクティブなサーバーに対して実行できます。
- --port オプションを barman-wal-archive および barman-wal-restore コマンドに追加して、SSH 構成を必要とせずにカスタム SSH ポートを使用できるようにします。
- さまざまなドキュメントの改善。
- Python 3.5 はサポートされなくなりました。

22.28.2 バグ修正

- barman cron の実行中に PostgreSQL 接続が正常に閉じられることを確認します。
- barman generate-manifest は、既存の backup_manifest ファイルをエラー条件として扱うようになりました。
- backup_manifest ファイルは、リカバリ操作中にバックアップ ID を追加して名前が変更され、古い backup_manifest ファイルを含む今後のバックアップが防止されます。
- タイムゾーンを認識しない json 出力のエポックタイムスタンプを修正しました。
- pg_basebackup の出力は、バックアップの進行中に Barman ログファイルに書き込まれるようになりました。
- このリリースに貢献して頂いた barthisrael、elhananjir、kraynopp、lucianobotti、および mxey に感謝します。

22.29 3.0.12022-06-27

22.29.1 バグ修正

- PyPI のパッケージ署名問題を修正しました 3.0.0 と同じソース

22.30 3.0.02022-06-23

22.30.1 重大な変更

- PostgreSQL バージョン 9.6 以前はサポートされなくなりました。これらのバージョンのいずれかを使用している場合、 Barman の以前のバージョンを使用する必要があります。
- Rsync バックアップのデフォルトのバックアップモードは、排他的ではなくコンカレントになりました。排他的バックアップは PostgreSQL 9.6 以降非推奨であり、PostgreSQL では削除されました

15. 15 より前の PostgreSQL バージョンに対して Barman を実行しており、排他的バックアップを使用したい場合は、 `backup_options` で `exclusive_backup` を設定する必要があります。

- 各バックアップの `backup.info` ファイルに保存されるバックアップメタデータには、追加のフィールドがあります。これは、 Barman の以前のバージョンが 3.0.0 で取得されたバックアップが存在すると動作しないことを意味します。さらに、 pg-backup-api のユーザーはバージョン 0.2.0 にアップグレードして、pg-backup-api が更新されたメタデータで動作できるようにする必要があります。

22.30.2 重要な変更

- `backup_compression` 構成オプションを設定することにより、 `backup_method = postgres` で取得したバックアップを pg_basebackup で圧縮できるようになりました。圧縮レベル、バックアップ形式、および pg_basebackup クライアントまたは PostgreSQL サーバーが圧縮を適用するかどうかを制御するための追加オプションが提供されています。注これらのバックアップのリカバリーには、 Barman が `recovery_staging_path` オプションで指定された場所のリカバリサーバー上の圧縮ファイルをステージングする必要があります。

- PostgreSQL 15 のサポートを追加します。排他的バックアップは PostgreSQL 15 ではサポートされていないため、 PostgreSQL 15 サーバーの Barman 構成は `backup_options` で `exclusive_backup` を指定することはできません。

- 圧縮された WAL ファイルを識別するときは、custom_compression_magic が設定されている場合、使用します。これにより、 Barman は非圧縮の WAL `streaming` ディレクトリの `*.partial` ファイルなどを正しく識別し、解凍しようとする代わりにそれらを返すことができます。

22.30.3 マイナー変更

- さまざまなドキュメントの改善。

22.30.4 バグ修正

- Barman がメッセージ「バックアップは開始バックアップコマンドの発行に失敗しました。」ログに記録する原因となった順序バグを修正します。 stopbackup コマンドの障害を処理している間。

- WAL セグメント名の 16 進値が 10 進数の整数として解析されるため、9 より大きいタイムラインが存在する場合に `--target-tli` を使用したリカバリーを妨げたバグを修正します。

- enum34 依存関係の問題により、特定の python2 インストールで barman クラウドを使用するときに発生するインポートエラーを修正します。

- マジックバイトを識別しようとしたときに、 Barman が圧縮 WAL から 3 バイトを超えて読み取らないバグを修正します。これは、3 バイトを超えるマジックを使用したカスタム圧縮 WAL が正しく解凍されることを意味します。

- `backup_method = rsync` を使用したバックアップ中に `--immediate-checkpoint` フラグが無視される原因となったバグを修正します。

22.31 2.192022-03-09

22.31.1 重要な変更

- barman diagnose 出力日付形式を ISO8601 に変更します。
- Barman Cloud に Google Cloud StorageGCS サポートを追加します。
- barman recover の --target-tli オプションの current および latest リカバリターゲットをサポートします。
- SLES へのインストールに関するドキュメントを追加します。

22.31.2 バグ修正

- barman-wal-archive --test は、エラーが発生したときにゼロ以外の終了コードを返すようになりました。
- -t オプションが使用される場合の barman-cloud-check-wal-archive の動作を修正して、接続テスト後に終了するようにしました。
- --no-get-wal が使用され、recovery_options で /"get-wal/" が設定されていない場合、barman recover は継続されるようになりました。
- 非アクティブなサーバーの barman show-servers --format=json ${server} 出力を修正しました。
- 構成ファイルに barman_home の存在を確認します。
- パッシブ barman サーバーは、 backup_method = postgres で取得したバックアップを同期するときに、テーブルスペースデータの 2 つのコピーを保存しなくなりました。
- このリリースへの richyen の貢献に感謝します。

22.32 2.182022-01-21

22.32.1 重要な変更

- barman クラウドにスナッピー圧縮アルゴリズムのサポートを追加しますオプションの python-snappy 依存関係が必要です。
- barman-cloud-wal-archive で WAL をアップロードするときに、Azure クライアントの同時実行パラメーターを設定できるようにします。
- barman クラウドに --tags オプションを追加して、バックアップファイルとアーカイブされた WAL をクラウドストレージ aws および azure でタグ付けできるようにします。
- 接続エラーの専用コード 2 が存在するように、バーマンクラウド終了ステータスコードを更新します。
- コマンド barman verify-backup および barman generate-manifest を追加して、バックアップが有効かどうかを確認します。
- barman クラウドでの Azure マネージド ID 認証のサポートを追加します。これは、 --credential オプションで有効にできます。

22.32.2 バグ修正

- バケットが存在しない場合の barman-cloud-check-wal-archive の動作を変更します。
- list-files 出力が、基になるファイルシステムに関係なく、常にソートされることを確認します。
- barman-cloud-backup-keep、barman-cloud-backup-delete、および barman-cloud-check-wal-archive のマニュアルページが Python パッケージに追加されました。
- このリリースに貢献して頂いた richyen と strakis に感謝いたします。

22.33 2.172021-12-01

22.33.1 重要な変更

- バージョン 2.14 で導入されたパフォーマンスの低下は、 --jobs フラグを使用するときに barman backup または barman recover コマンドのコピー時間を増加させました。
- sender プロセスの rsync 部分転送エラーを無視して、このようなエラーがバックアップの失敗を引き起こさないようにします。barthisrael に感謝します。

22.34 2.162021-11-17

22.34.1 重要な変更

- コマンド barman-check-wal-archive および barman-cloud-check-wal-archive を追加して、提案されたアーカイブの場所が新しい PostgreSQL サーバーに安全に使用できるかどうかを検証します。
- Barman がカスタム圧縮スキームを使用して既に圧縮された WAL を識別して、再圧縮を回避できるようにします。
- last_backup_minimum_size および last_wal_maximum_age オプションを barman check に追加します。

22.34.2 バグ修正

- コマンドライン解析には、メンテナンスされていない argh モジュールの代わりに argparse を使用します。
- begin_time および end_time のタイムゾーンを一貫させます。
- このリリースへの貢献に感謝します。

22.35 2.152021-10-12

22.35.1 重要な変更

- list-backup 、 list-server および show-server コマンドの複数形を追加しました。これらは list-backups 、 list-servers および show-servers になりました。単数形は、下位互換性のために保持されています。

- カタログ内の最新の失敗したバックアップを参照する last-failed バックアップショートカットを追加して、次のことを行うことができます。
  - barman delete <SERVER> last-failed

22.35.2 バグ修正

- EPAS の古いバージョンで正しいバージョン文字列を検出する問題のため、EPAS バージョン 9.6 および 10 のバックアップからテーブルスペースは省略されません。

22.36 2.142021-09-22

22.36.1 重要な変更

- barman-cloud-backup-delete コマンドを追加します。これは、バックアップ ID またはリテンションポリシーを指定して、クラウドストレージのバックアップを削除できます。
- barman keep および barman-cloud-backup-keep を使用して既存のバックアップをアーカイブバックアップとしてタグ付けする機能を導入することにより、有効な保持ポリシーを超えてバックアップを保持できるようにします。
- Azure BLOB ストレージのより広いストレージアカウントレベルではなく、制限された BLOB コンテナレベルで作成された SAS 認証トークンの使用を許可します
- 数十万のファイルを含むバックアップの空のディレクトリへの barman restore を大幅に高速化します。

22.36.2 バグ修正

- ユーザーに「userepl」権限がない場合でもバックアップ権限チェックは失敗しなくなり、必要な権限が欠落している場合には、より良いエラーメッセージを結果ことができます#318 および#319。

22.37 2.132021-07-26

22.37.1 重要な変更

- Azure Blob Storage のサポートを barman-cloud に追加します
- --tablespace name:location を介して barman-cloud-restore でのテーブルスペースのリマッピングをサポート
- barman-cloud-backup および barman-cloud-wal-archive を Barman フックスクリプトとして実行できるようにして、 Barman サーバーからクラウドストレージにデータを中継できるようにします

22.37.2 バグ修正

- idle_in_transaction_session_timeout が原因で Stop バックアップが失敗する https://github.com/EnterpriseDB/barman/issues/333

- xlog.db エントリの更新における backup と archive-wal 間の競合状態を修正します #328
- barman-cloud-backup のシンボリックリンクである PGDATA を処理すると、リストア時に「seeking attempts is not allowed」エラーが発生しました #351
- 元がシンボリックリンクの場合、リストア時に pg_wal を再作成します #327
- リストア時にテーブルスペースの pg_tblspc シンボリックリンクを再作成します #343
- barman-cloud-backup-list で、たとえば、Glacier ストレージにあるため、読み取ることができないバックアップをスキップさせます #332
- `-d database` オプションを barman-cloud-backup に追加して、最初に接続するデータベースを指定します #307
- boto3 クライアントをピクルしようとすると発生する、Python 3.8 以降の barman-cloud-backup からの“Backup failed upgradeing data”エラーを修正します #361
- デフォルトで暗号化が有効になっていないバケットの S3 でサーバーサイドの暗号化を正しく有効にします。

  Barman 2.12 では、barman-cloud-backup の `--encryption` オプションは、暗号化が有効になっていない S3 バケットにバックアップが保存されている場合、バックアップのコンテンツの暗号化を正しく有効にしませんでした。これに該当する場合は、古いバックアップを削除し、 Barman 2.13 で新しいバックアップを取ることを検討してください。

  S3 バケットの暗号化がデフォルトで有効になっている場合これをお勧めしますが、これは影響しません。

22.38 2.12.12021-06-30

22.38.1 バグ修正

- 他の `target-*` リカバリオプションで target-tli を指定できるようにします。
- barman-cloud-backup-list のマンページの誤った NAME を修正します。
- SIGALRM が無視される場合でもエラーを発生させません。
- Postgres 13 から wal_keep_segments ではなく、wal_keep_size を取得します。

22.39 2.122020-11-05

22.39.1 重要な変更

- local-rsync と呼ばれる新しい backup_method オプションを導入します。これは、PostgreSQL と同じサーバーに Barman がインストールされ、rsync を直接使用してベースバックアップを取得し、SSH レイヤーをバイパスする場合を対象としています。

22.39.2 バグ修正

- ワーカープロセスでの boto 接続の破損を回避します。
- テスト中に PostgreSQL への接続試行を回避します。

22.40 2.112020-07-09

22.40.1 重要な変更

- クラウド関連のすべてのユーティリティを含む barman-cli-cloud パッケージの紹介。
- barman-cloud-wal-restore を追加して、オブジェクトストアから barman-cloud-wal-archive で以前にアーカイブされた WAL ファイルを復元します。
- barman-cloud-restore を追加して、以前に barman-cloud-backup で取得したバックアップをオブジェクトストアから復元します。
- barman-cloud-backup-list に、オブジェクトストアで barman-cloud-backup を使用して取得されたバックアップをリストします。
- barman-cloud-backup の任意のアーカイブサイズのサポートを追加します。
- –endpoint-url オプションのサポートをクラウドユーティリティに追加します。
- PG 10+の厳密なスーパーユーザー要件を削除します Kaarel Moppel による。
- barman の-log-level ランタイムオプションを追加して、特定のコマンドのデフォルトのログレベルをオーバーライドします。
- PostgreSQL 13 のサポート

22.40.2 バグ修正

- barman-cli GH-257 の SSH 接続でのメッセージと警告を抑制します。
- barman-cloud-backup GH-259 でアップロードされたパーツを取得する際の競合状態を修正します。
- バックアップ後に PostgreSQL 接続を閉じます GH-258
- accept-wal –reset GH-260 で初期化されていないレプリケーションスロットがないかを確認します。
- begin_wal に基づいて行動する前に、begin_wal が有効になっていることを確認します GH-262
- カスタムセグメントサイズ GH-287 を使用した XLOG / WAL 演算のバグを修正します。
- 最近の rsync との rsync 互換性エラーを修正しました。
- PostgreSQLClient バージョンの解析を修正しました。
- 非 ASCII メッセージを使用した PostgreSQL 例外処理を修正します。
- 各 postgres 接続に空の search_path があることを確認します。
- backup.info ファイルの読み取り中は PostgreSQL への接続を避けてください。

RPM / Apt パッケージを介してインストールされた既に barman-cloud-wal-archive または barman-cloud-backup を使用しており、システムをアップグレードする場合、barman-cli-cloud パッケージをインストールする必要があります。すべてのクラウド関連ツールは、以前に barman-cli に同梱されていた barman-cloud-wal-archive および barman-cloud-backup を含む barman-cli-cloud パッケージの一部になりました。その理由は、クラウドユーティリティの要件である boto3 ライブラリの複雑な依存関係管理です。

22.41 2.102019-12-05

22.41.1 重要な変更

- get-wal および barman-wal-restore を使用して.partial WAL ファイルをプルし、リカバリシナリオの restore_command で Barman サーバーから部分的な WAL ファイルのコンテンツを取得できます。この機能は、RPO=0 リカバリ操作を簡素化および強化します。
- PostgreSQL システム識別子をサーバーディレクトリとバックアップ情報ファイル内に保存します。check コマンドを改善して、システム識別子とアクティブな接続標準およびレプリケーションおよびディスク上のデータとの一貫性を検証します。
- barman-cloud-wal-archive と呼ばれる新しいスクリプトが barman-cli パッケージに追加され、PostgreSQL archive_command を使用して AWS S3 と互換性のあるクラウドオブジェクトストレージサービスに WAL ファイルを直接シッピングします。暗号化と圧縮をサポートしています。
- barman-cloud-backup と呼ばれる新しいスクリプトが barman-cli パッケージに追加され、ローカルの PostgreSQL サーバーから AWS S3 と互換性のあるクラウドオブジェクトストレージサービスにベースバックアップを直接出荷します。暗号化、並列アップロード、圧縮をサポートしています。
- server/global オプション create_slot を介したレプリケーションスロットの自動作成。 auto に設定されている場合、streaming_archive が有効であり、slot_name が定義されている場合、 Barman はレプリケーションスロットを作成します。下位互換性のため、デフォルト値は manual です。
- '-w/–wait' オプションを backup コマンドに追加し、バックアップが完了したと判断する前に、必要なすべての WAL ファイルがアーカイブされるまで Barman を待機させます。 –wait-timeout オプションデフォルト 0、タイムアウトなしも追加します。
- Barman 出力、特に barman promote InfoSec からのパスワードを秘匿化する
- 最後の部分ファイルが現在の場所または存在する場合はレプリケーションスロットとアラインメントされていることを確認することにより、receive-wal –reset コマンドの堅牢性を向上させます。
- ドキュメントの改善

22.41.2 バグ修正

- PGDATA GH-245 内のテーブルスペースを除外する場合の間違った文字列マッチング操作。
- WAL 削除フックスクリプト GH-240 のマイナー修正。
- PostgreSQL 接続生存性チェック GH-239 を修正します。

22.42 2.92019-08-01

22.42.1 重要な変更

- GUC オプションとシグナルファイル recovery.signal および standby.signal を介してリカバリとスタンバイ設定を管理する新しい方法をサポートすることにより、PostgreSQL 12 を透過的にサポートします
- –bwlimit コマンドラインオプションを追加して、バックアップおよびリカバリーコマンドの帯域幅制限を設定します
- 最新のバックアップが WAITING_FOR_WALS の場合、check コマンドの WAL アーカイブ障害を無視する
- PostgreSQL 10 以降で recover コマンドのリカバリターゲットログシーケンス番号を設定するために-target-lsn オプションを追加
- –spool-dir オプションを barman-wal-restore に追加して、ユーザーがスプールディレクトリの場所をデフォルトから変更できるように、同じサーバー上に複数の PostgreSQL インスタンスがある場合の競合を回避できます Drazen Kacar に感謝します。
- barman_xlog ディレクトリの名前を barman_wal に変更します
- コマンド出力を JSON オブジェクトとしてエクスポートし、外部ツールやシステムとの統合を促進するための JSON 出力ライタ Marcin Onufry Hlybin に感謝します。このリリースでは実験的。

22.42.2 バグ修正

- `replication-status` は、スロットのないストリーマーを表示しない GH-222
- 接続が生きていることを確認するとき「SELECT 1」クエリー、PostgreSQL 接続 GH-149 のステータスを保存します。これにより、トランザクション内アイドルタイムアウトにより接続が終了し、同時バックアップが失敗する場合が修正されます。

22.43 2.82019-05-17

22.43.1 重要な変更

- パッシブノードのインクリメンタルバックアップコピーの geo 冗長性での reuse_backup のサポートを追加します
- より汎用的な dateutil.parser の代わりに strptime を使用することにより、rsync ベースのコピーのパフォーマンスを向上させます #210
- '-test' オプションを barman-wal-archive および barman-wal-restore に追加して、 Barman サーバーとの接続を確認します
- PostgreSQL 9.5 が PGDG によって EOL 宣言されると、将来のデフォルト値が clusive_backup から concurrent_backup に変更されるため、backup_options が明示的に設定されていない場合に文句を言う
- show-server および診断コマンドの追加設定を表示します archive_timeout、data_checksums、hot_standby、max_wal_senders、max_replication_slots、および wal_compression。
- Barman に barman-cli プロジェクトをマージします

22.43.2 マイナー変更

- 非アクティブなサーバーの check –nagios のメッセージングを改善します。
- recovery コマンドを使用してリモート SSH コマンドをログに記録します。
- replication-status コマンドでの論理デコード接続を非表示にします。

このリリースは、Python 3 を公式にサポートし、Python 2 将来のリリースで廃止される可能性があります非推奨にします。

PostgreSQL 9.3 以前は、 Barman のこのリリースから非推奨になりました。スタンバイからのバックアップのサポートは、PostgreSQL 9.4 以降、およびスタンバイからの WAL 出荷に制限されています。詳細については、ドキュメントを参照してください。

22.43.3 バグ修正

- Python 3 の get-wal のエンコードエラーを修正しました。Jeff Janes、#221
- exclude_and_protect_filter を修正しました Jeff Janes、#217
- WAL をリセットするときに偽のメッセージを削除します Jeff Janes、#215
- プライマリに最初のバックアップより古い WAL がある場合の sync-wals エラーを修正します。
- synchronous_standby_names 設定での二重引用符のサポート。

22.44 2.72019-03-12

22.44.1 重要な変更

- パラレルバックアップ中のエラー処理を修正しました。以前は、コピー中の回復不能なエラーが barman の内部状態を破損する可能性があり、SIGTERM での barman プロセスの手動 kill と PostgreSQL で実行中のバックアップの手動クリーンアップが必要でした。 GH#199
- 入力および出力 GH#194 および GH#196 での UTF-8 文字のサポートを修正しました。
- 地理的冗長性の最初の同期で履歴/バックアップ/部分ファイルを無視する GH#198 。
- cron がブレークする原因となった地理的冗長性によるネットワーク障害を修正します GH#202 。
- 9.2 より前の PostgreSQL のバックアップ検証を修正しました。
- さまざまなドキュメントの修正。

22.45 2.62019-02-04

22.45.1 重要な変更

- 地理的冗長性のサポートを追加し、3 つの新しいコマンド sync-info、sync-backup、sync-wals を導入します。地理的冗長性により、 Barman サーバーは PostgreSQL サーバーの代わりに別の Barman サーバーをデータソースとして使用できます。

- barman-cli に含まれる barman-wal-archive スクリプトを使用して、 Barman が PostgreSQL の archive_command を介して WAL ファイルを安全に受信できる put-wal コマンドを追加します。
- check コマンドに ANSI カラーサポートを追加します。

22.45.2 バグ修正

- 空の WAL ディレクトリを使用してスタンバイで switch-wal を修正します。
- wait_for_wal メソッドでのアーカイバーロックを尊重します。
- WAL 圧縮検出アルゴリズムを修正します。
- コンカレント停止バックアップエラーの current_action を修正します。
- バックアップを検証するときに、ロックファイルビジーをエラーとして扱わないでください。

22.46 2.52018-10-23

22.46.1 重要な変更

- PostgreSQL 11 のサポートを追加
- check-backup コマンドを追加して、ベースバックアップの一貫性に必要な WAL ファイルがアーカイブに存在することを確認します。 Barman は、ベースバックアップの完了後に新しい状態 WAITING_FOR_WALS を追加し、バックアップの先頭から最後までのすべての WAL ファイルが存在することを確認すると DONE に設定します。このコマンドは、通常の cron メンテナンスジョブに含まれます。 Barman は、 WAITING_FOR_WALS 状態にあるバックアップを復旧しようとしているユーザーに通知するようになりました。
- switch-xlog –archive がスタンバイで動作できるようにしますアーカイブ部分のみ

22.46.2 バグ修正

- 外部コマンド出力の読み取りデコードエラーを修正しました 問題 174
- WAL ストリーミングとスタンバイからのバックアップに関するドキュメントを修正します。

22.47 2.42018-05-25

22.47.1 重要な変更

- バックアップ削除前/後のための標準および再試行フックスクリプトを追加します。
- リカバリー前/後用の標準および再試行フックスクリプトを追加します。
- WAL 削除前/後のための標準のフックスクリプトと再試行フックスクリプトを追加します。
- –standby-mode オプションを barman recovery に追加して、事前に生成された recovery.conf に standby_mode = on を追加します。

- –target-action オプションを barman recovery に追加し、ユーザーが事前に生成された recovery.conf ファイルにシャットダウン、一時停止、またはプロモートを追加できるようにします。
- 一貫性チェックによりポイントインタイムリカバリーの使いやすさを向上させますたとえば、復旧時間はバックアップの終了時間後です。
- マイナーなドキュメントの改善。
- Python 3.3 のサポートを終了します。

22.47.2 バグ修正

- リモートの get_file_content メソッド GitHub #151 を修正し、インクリメンタルリカバリーが発生しないようにします。
- コマンドの Unicode の問題 GitHub #143 および#150。
- pg_basebackup >= 10 の場合、 –wal-method=none を追加します GitHub #133 。
- プロセスマネージャーモジュールがロックファイルのコンテンツを上書きするのを停止する
- rsync 出力の解析ルールを緩和する
- ストリーミングディレクトリ内の消えたファイルを無視する
- 大文字と小文字を区別しないスロット名 GitHub #170
- DataTransferFailure.from_command_error() の復元力を高める GitHub #86
- command() の名前を barman_command() に変更します GitHub #118
- 同期スタンバイ名リストが設定されていない場合は初期化する GitHub #111
- 正しいプレースホルダーの順序 GitHub #138
- レプリケーション接続の日付スタイルを iso に強制する
- delete コマンドでバックアップが削除されない場合、エラーを返します
- is_power_of_two(None) を呼び出す際の例外を修正
- ダウングレードした同期スタンバイの名前付けデバッグするメッセージ GitHub #89

22.48 2.32017-09-05

22.48.1 重要な変更

- PostgreSQL 10 にサポートを追加します
- PostgreSQL 10 の名前の変更に従ってください。
  - switch-xlog コマンドの名前が switch-wal に変更されました。
  - コマンド出力では、xlog ワードが WAL に変更され、ロケーションが LSN に変更されました。
- –network-compression / –no-network-compression オプションを barman recovery に追加して、実行時のネットワーク圧縮を有効または無効にします

- 一貫した状態に到達したときにリカバリを終了するように、recover コマンドに-target-immediate オプションを追加しますバックアップの終了、PostgreSQL 9.4 から利用できます
- barman status コマンドを使用してクラスター状態マスターまたはスタンバイを表示します
- ドキュメントの改善

22.48.2 バグ修正

- Parallel_jobs > 1 での高いメモリ使用量を修正する #116
- パラレルコピーを使用したエラーの処理の向上#114
- システム例外を使用してバーマン診断をより堅牢にする
- archive-wal に.tmp 拡張子を持つファイルを無視させます

22.49 2.22017-07-17

22.49.1 重要な変更

- repair_jobs グローバル/サーバーオプションを使用してバックアップ/リカバリーのパラレルコピーを実装しますが、backup および recover コマンドの-jobs または-j ランタイムオプションでオーバーライドされます。パラレルバックアップは、rsync コピー方法でのみ使用できます。デフォルトでは、1 に設定されています以前のバージョンとの動作の互換性のため。
- PostgreSQL 8.4 以降のカスタム WAL サイズをサポートします。バックアップ時に、 Barman は PostgreSQL から wal_segment_size および wal_block_size 値を取得し、必要な計算を計算します。
- check コマンドを改善して、アーカイバー=オフの場合に受信ディレクトリが空であり、ストリーミング_アーカイバー=オフの場合にストリーミングディレクトリが空であることを確認します#80
- external_configuration を backup_options に追加して、ユーザーが PGDATADebian/Ubuntu インストールのデフォルト内にない場合、構成ファイルのバックアップを無視するように Barman に指示できるようにします。この場合、 Barman は警告を表示しません。
- バーマンリカバリーに-get-wal および-no-get-wal オプションを追加します
- check コマンドの max_incoming_wals_queue グローバル/サーバーオプションを追加して、アーカイバーとストリーミング_アーカイバーの両方の着信 WAL ディレクトリに指定された値より多くのファイルが含まれている場合に、ノンブロッキングエラーが返されるようにします。
- ドキュメントの改善
- ファイル形式の変更
  - backup.info ファイルの形式が変更されました。このため、 Barman 2.2 で取得したバックアップは、以前のバージョンの Barman で読み取ることができません。ただし、以前のバージョンで取得したバックアップは、 Barman 2.2 で読み取ることができます。

22.49.2 バグ修正

- レプリケーションステータスがスタンバイに対して機能することを許可します
- pg_basebackup を開始する前に PostgreSQL 接続を閉じます #104、#108
- 特殊文字を含むパスを安全に処理する
- ストリーミングソースのプロモーション後に.partial ファイルをアーカイブする
- リカバリー中にディレクトリを再帰的に作成する SF#44
- xlog.db ロックの改善 #99
- リカバリー中に tablespace_map ファイルを削除する #95
- 接続がドロップした場合は PostgreSQL に再接続します SF#82

22.50 2.12017-01-05

22.50.1 重要な変更

- –archive および-archive-timeout オプションを switch-xlog コマンドに追加します。
- PostgreSQL 10#73 の暫定サポート。
- マイナーな追加
  - 最後にアーカイブされた WAL 情報を診断出力に追加します。
  - 開始時間と実行時間を delete コマンドの出力に追加します。

22.50.2 バグ修正

- 非アクティブなサーバーでの get-wal コマンドの結果失敗
- stable_archive_names および streaming_backup_name オプションをグローバルにする #57
- 転送中に切り詰められたファイルによる rsync 障害を修正する #64
- 圧縮履歴ファイルを正しく処理する #66
- リカバリーの準備時に pg_tblspc のシンボリックリンクの逆参照を回避する #55
- 最後のアーカイブ失敗の比較を修正しました #40、#58
- postgresql.conf が書き込み可能でない場合のリカバリの失敗を回避する #68
- replication-status コマンドの出力を修正しました #56
- pg_basebackup のようなバックアップからファイルを除外する #65、#72
- テーブルスペースのコピー中に他の Postgres バージョンからディレクトリを除外する #74
- 再試行フックスクリプトオプションをグローバルにする

22.51 2.02016-09-27

22.51.1 重要な変更

- backup_method=postgres を使用した PostgreSQL ストリーミングレプリケーションプロトコルを介した pg_basebackup および base バックアップのサポート PostgreSQL 9.1 以降が必要
- slot_name 構成オプションと、receive-wal コマンドの-create-slot および-drop-slot オプションを介した物理レプリケーションスロットのサポート PostgreSQL 9.4 以降が必要。 slot_name が指定され、streaming_archiver が有効になっている場合、receive-wal は pg_receivexlog と透過的に統合し、スロットが存在し、アクティブに使用されていることを確認します
- PostgreSQL 9.6 で導入された新しいバックアップ API のサポート。これは、同時バックアップと、標準の rsync バックアップ方法を使用してスタンバイサーバーからのバックアップを透過的に有効にします。同時バックアップは、pgespresso 拡張機能を介して PostgreSQL 9.2〜9.5 バージョンでのみ可能でした。新しいバックアップ API は、将来的に pgespresso を冗長化します
- 適切に構成されている場合、 Barman は、WAL ストリーミングの観点から同期スタンバイとして機能できます。マスター上の synchronous_standby_names 優先リストの streaming_archiver_name を適切に設定し、レプリケーションスロットのサポートを有効にすることにより、receive-wal コマンドを PostgreSQL 同期レプリケーションクラスターの一部にすることができ、RPO=0 PostgreSQL 9.5.5 以降が必要です。
- barman-wal-restore は、Python で書かれた標準の堅牢なスクリプトで、クラスターのスタンバイサーバーの recovery.conf ファイルで restore_command として使用できます。 SSH を介して get-wal を効率的に呼び出すことにより、WAL ファイルのリモート並列取得をサポートします。現在、barman-cli と呼ばれる別のプロジェクトとして利用できます。 get-wal が recovery_options にリストされている場合、リモートリカバリには barman-cli パッケージが必要です
- check_timeout グローバル/サーバー構成オプションを使用して check コマンドの最大実行時間を制御しますデフォルトで 30 秒
- archive-wal 実行によって処理される WAL セグメントの数を、それぞれ標準のアーカイバーと receive-wal からの WAL セグメントのアーカイブを制御する archiver_batch_size および streaming_archiver_batch_size グローバル/サーバーオプションを介して制限します
- チェック操作中の XLOG データベースのロックを削除しました
- show-backup コマンドはタイムラインを認識するようになり、特定のベースバックアップのリカバリターゲットとして使用できるタイムラインを適切に表示します。内部的に、 Barman は.history ファイルを解析できるようになりました
- コピー操作で問題が発生した場合の再試行メカニズムの背後にあるロジックを改善しました。これには、バックアップ rsync および postgres およびリモートリカバリ rsync が含まれます
- リモートコマンドおよび物理コピーインターフェイスを含むコードのリファクタリング

22.51.2 バグ修正

- ストリーミングからの.history ファイルを正しく処理する
- PostgreSQL 9.1 のレプリケーションステータスを修正する
- 送信および書き込み場所が利用できない場合のレプリケーションステータスを修正する
- pg_receivexlog 終了時の誤解を招くメッセージを修正

22.52 1.6.12016-05-23

22.52.1 マイナー変更

- get-wal コマンドに-peek オプションを追加して、Barman のアーカイブから既存の WAL ファイルを検出します
- PostgreSQL サーバーに接続しているストリーミングレプリケーションクライアントのステータスを監視するための replication-status コマンドを追加します。 –target オプションを使用すると、ユーザーは要求をホットスタンバイサーバーまたは WAL ストリーミングクライアントのみに制限できます
- switch-xlog コマンドを追加して、WAL ファイルの PostgreSQL サーバーへのスイッチを要求します。 `-force」を介して、事前にチェックポイントを発行します
- repairing_archiver が有効になっているときに適切な application_name を pg_receivexlog に設定する streaming_archiver_name オプションを追加します PostgreSQL 9.3 以降のみ
- PostgreSQL の標準接続を使用して **スーパーユーザー** 特権を確認する #30
- WAL アーカイブが空でないことを確認します
- サーバーがダウンしているときにマスターの `backup_label`を確認する
- barman-wal-restore contrib スクリプトを改善

22.52.2 バグ修正

- “failed tasks”チェックを致命的でないものとして扱う
- get-wal の `-x`オプションの名前を `-z`に変更します
- archive_mode=always の PostgreSQL 9.5 のサポートを追加#32
- 必要に応じて PostgreSQL 接続を適切に閉じる
- pg_receive_xlog バージョン 9.2 の receive-wal を修正

22.53 1.6.02016-02-29

22.53.1 重要な変更

- streaming_conninfo サーバーオプションを介したストリーミングレプリケーション接続のサポート

- Barman が PostgreSQL のネイティブストリーミングプロトコルを介して WAL ファイルを受信できるようにする streaming_archiver オプションのサポート。 'on' に設定すると、pg_receivexlog に依存して WAL データを受信し、目標リカバリポイントを削減します。現在、WAL ストリーミングは追加機能です標準のログアーカイブが引き続き必要です。
- streaming_archiver がオンのときに、WAL ストリーミングの pg_receivexlog をラップする receive-wal コマンドを実装します。ストリーミングプロトコルを介して WAL ファイルの受信を停止するには、–stop オプションを追加します。 –reset オプションを追加して、ストリーミングステータスをリセットし、Postgres の現在の xlog からリスタートします。
- cron コマンドを介した receive-wal コマンドの自動管理スタートアップと停止
- path_prefix 構成オプションのサポート
- PostgreSQL の archive_command を介したログ配布を介して、特定のサーバーの継続的な WAL アーカイブを有効にするアーカイバーオプション現在オンに固定されています
- streaming_wals_directory および errors_directory オプションのサポート
- archive-wal コマンドでの WAL 重複の管理と check コマンドとの統合
- Streaming_archive が有効になっているときに pg_receivexlog が check コマンドで実行されているかどうかを確認する
- 失敗したバックアップが check コマンドに存在するかどうかを確認する
- 受信ディレクトリで圧縮された WAL ファイルを受け入れます
- pigz コンプレッサーのサポートを追加します Stefano Zacchiroli zack@upsilon.cc に感謝します
- pygzip および pybzip2 コンプレッサーを実装します Christoph Moench-Tegeder christoph@2ndquadrant.de の初期アイデアに基づいています
- バックアップの最後に暗黙的な復元ポイントを作成する
- バーマンステータスの PostgreSQL データファイルの現在のサイズ
- PostgreSQL 9.5 サーバーの archive_mode=always を許可します Christoph Moench-Tegeder christoph@2ndquadrant.de に感謝します
- PostgreSQL への接続を担当するコードの完全なリファクタリング
- サブプロセスに関する cron コマンドのメッセージを改善します
- Python > = 3.3 のネイティブサポート
- 行動の変化
  - archive-wal 中に WAL ファイルのトラッシュを停止する commit:e3a1d16

22.53.2 バグ修正

- アトミック WAL ファイルのアーカイブ#9 および#12
- “-c”オプションを Barman サブプロセスに伝播する#19
- バックアップ削除中のバックアップ ID の管理を修正する #22

- archive-wal の堅牢性とログメッセージを向上させる #24
- パラメーターが欠落している場合のエラー処理を改善する

22.54 1.5.12015-11-16

22.54.1 マイナー変更

- 特定のサーバーで WAL メンテナンス操作を実行する `archive-wal' コマンドのサポートを追加します
- `cron`コマンドの「サーバーごとの」同時実行性のサポートを追加します
- xlog.db エラーの管理の向上
- WAL ファイルの混合圧縮タイプのサポートを追加します SF.net#61

22.54.2 バグ修正

- リカバリ中のリテンションポリシーチェックを回避する
- PostgreSQL バージョン<9.0 での `wal_level`チェックを回避する #3
- バックアップサイズの計算を修正する #5

22.55 1.5.02015-09-28

22.55.1 重要な変更

- ユーザーが特定のサーバーのアーカイブから WAL ファイルを取得できる get-wal コマンドのサポートを追加します
- 再試行フックスクリプト、 Barman が成功するまで実行しようとする特別な種類のフックスクリプトのサポートを追加します
- サーバーのアクティブな構成オプションを追加して、False に設定してサーバーを一時的に無効にします
- ロックファイルの場所を変更するための barman_lock_directory グローバルオプションを追加しますデフォルトでは `barman_home`
- バックアップを開始する前にチェックの完全なスイートを実行し、1 つ以上のチェックが失敗した場合にスキップします
- 実行中のバックアップの削除を禁止する
- バックアップおよびリカバリー操作中に PostgreSQL サーバーの include 指示子を分析する
- Barman の構成に競合するパスのチェックを追加します。サーバー内を一時的に無効にすることによると、サーバー間で任意のサーバーへのコマンドを拒否することにより。
- wal_level のチェックを追加
- barman get-wal と組み合わせて、スタンバイサーバーで restore_command として使用する barman-wal-restore スクリプトを追加します

- エラー管理のための標準の一貫したポリシーを実装する
- バックアップのキャッシュ管理の向上
- 単体テストでの構成の管理の向上
- チュートリアルとマニュアルページのソースが Markdown 形式に変換されました
- Sphinx を介してコードドキュメントを追加
- バックアップおよび recover コマンドの管理を担当するコードの完全なリファクタリング
- バックアップの内部ディレクトリ構造の変更
- copy_method オプションを導入します現在 rsync に修正されています

22.55.2 バグ修正

- PostgreSQL 構成ファイルで `=`を使用せずにオプションを管理する
- タイムライン履歴ファイルの保存修正#70
- SUSE Linux での rsync の回避策クローズ#13 および#26
- postgresql.auto.conf の危険な設定を無効にするクローズ#68

22.56 1.4.12015-05-05

22.56.1 マイナー変更

- xlogdb ファイルの管理が改善され、更新時に正しく fsync されるようになりました。また、rebuild-xlogdb コマンドは一時的な新しいファイルを操作するようになり、完了するとメインファイルを上書きします。
- dateutil モジュールの互換性のための単体テストを追加
- PEP 440 ルールに従って Barman バージョンを変更し、Python 3.4 でのテストのサポートを追加しました

22.56.2 バグ修正

- 最初のバックアップが空の場合に WAL アーカイブが動作を停止する問題を修正したクローズ#64
- Barman リカバリーのエラー処理中の例外を修正するクローズ#65
- バックアップ後、cron アクティビティを WAL アーカイブのみに制限しますクローズ#62
- backup delete コマンドの堅牢性とエラー報告の向上クローズ#63
- show-backup コマンドで報告される WAL 生産率の計算を修正します

22.57 1.4.02015-01-26

22.57.1 重要な変更

- resume_backup グローバル/サーバーオプションを介したインクリメンタル ベースバックアップの実装。使用可能な値は off 無効、デフォルト、copy 未変更のファイルが転送されないようにする、および link ハードリンクを介した重複排除を許可します。
- 再利用_backup=リンクを使用する場合、重複排除効果を保存および表示します。
- check、show-server、および status コマンドに pg_stat_archiver PostgreSQL 9.4 の透過的なサポートを追加しました。
- 成功したバックアップの最後に WAL メンテナンスを呼び出すことにより、管理が向上しました。
- 同時バックアップケースと排他的バックアップケースを区別することにより、未使用の WAL ファイルをゴミ箱に入れる方法を変更しました。
- WAL 統計計算のパフォーマンスの向上。
- 欠落している pg_ident.conf は、エラーではなく警告として扱います。
- 残りの yield 呼び出しを削除することにより、出力レイヤーをリファクタリングしました。
- rsync がシステムパスにあることを確認します。
- 履歴ファイルを WAL 管理に含めます。
- より多くの単体テストを介したロバスト性の向上。

22.57.2 バグ修正

- バグ#55 を修正しました ディレクトリでの fsync EINVAL エラーを無視します。
- 修正されたバグ#58 保持ポリシーの削除。

22.58 1.3.32014-08-21

22.58.1 重要な変更

- 管理者がカタログ内の最後のバックアップの最大保存期間を設定できる新しいグローバル/サーバーオプションである last_backup_max_age を追加しました。定期的なバックアップ実行に関する問題を検出しやすくなります
- 2 つのグローバル/サーバーオプション“basebackup_retry_times”と“basebackup_retry_sleep”を導入することにより、“barmanbackup”の堅牢性が向上しました。これらのオプションを使用すると、管理者は、失敗後のコピー操作の試行回数と、再試行までの待機秒数をそれぞれ指定できます。
- 以前の rsync 呼び出しをいくつかのことに分割することにより、既存のディレクトリの rsync を介したリカバリプロセスインクリメンタルリカバリーを改善しました-必要な場合にのみチェックサム制御を呼び出す
- PostgreSQL 8.3 のサポートを追加しました

22.58.2 マイナー変更

- コンマ区切りリスト値構成オプションのサポート
- 「barmanbackup」および「barman cron」中にバックアップファイルと WAL ファイルで fsync() を呼び出すことにより、バックアップの耐久性が向上しました
- “barman check –nagios”の Nagios 出力の向上
- 「barman show-backup」での WAL の表示圧縮率
- バーマンバックアップの実行中にキーボードの中断 CTRL-C が正しく処理される
- バックアップの停止に関する障害のエラーメッセージの改善
- 単体テストの幅広いカバレッジ

22.58.3 バグ修正

- “barman recovery”中にリモートサーバーに“recovery.conf”をコピーします#45
- プレ/ポストアーカイブフックスクリプトを正しく検出する #41

22.59 1.3.22014-04-15

22.59.1 バグ修正

- PostgreSQL 8.4 との非互換性を修正しましたクローズ#40、バージョン 1.3.1 で導入されたバグ

22.60 1.3.12014-04-14

22.60.1 マイナー変更

- “pgespresso”拡張機能を使用する PostgreSQL 9.2 および 9.3 サーバーの同時バックアップのサポートを追加しました。この機能は、“backup_options”構成オプション global/server によって制御され、“concurrent_backup”に設定されている場合にアクティブになります。コンカレントバックアップにより、DBA はストリーミングレプリケートスタンバイから完全なバックアップ操作を実行できます。
- Barman システムに関する重要な情報を出力する「barman promote」コマンドを追加しましたサポートと問題解決に非常に役立ちます
- エラーメッセージと例外処理インターフェイスの改善

22.60.2 バグ修正

- PGDATA ディレクトリ内で作成されたテーブルスペースのリカバリーのバグを修正しましたバージョン 1.3.0 で導入されたバグ
- cron ジョブで使用されるコマンドのクワイエットモード用、未処理の-q オプションのマイナーバグを修正しましたバージョン 1.3.0 で導入されたバグ
- 軽微なバグ修正とコードのリファクタリング

22.61 1.3.02014-02-03

22.61.1 重要な変更

- 新しい FieldListFile クラス infofile モジュールを使用するようにバックアップメタデータの BackupInfo クラスをリファクタリングしました
- Nagios NagiosOutputWriter クラスとの統合を促進するために、専用モジュールを使用するように出力レイヤーをリファクタリングしました
- stdin / stderr / stdout チャネルを分離するためにリファクタリングされたサブプロセス処理 command_wrappers モジュール
- リファクタリングされたフックスクリプト管理
- 構成クラスからロギング構成とユーザー ID の適用を抽出しました。
- `pre_archive_script`および`post_archive_script`構成オプションを介して、WAL ファイルがアーカイブされる前後に実行されるフックスクリプトのサポート。
- –immediate-checkpoint コマンドオプションと`immediate_checkpoint`構成オプションを使用して即時チェックポイント機能を実装
- `network_compression`構成オプションを介してリモートバックアップとリカバリーのネットワーク圧縮を実装しました#19
- `rebuild-xlogdb`コマンドを実装しましたクローズ#27 および#28
- PGDATA ディレクトリ内にあるテーブルスペースの重複排除を追加しました
- リモートディレクトリの準備を実行することにより、ローカルリカバリと同じように動作するようにリモートリカバリコードをリファクタリングしましたリモートユーザーがリモートサーバーに対する適切な権限を持っていることを前提としています。
- `barmanbackup' は、完全なバックアップを実行する前にサーバーディレクトリの作成を試行するようになりました#14

22.61.2 バグ修正

- バグ 22 の修正 テーブルスペースの再配置に関するドキュメントの改善
- 修正したバグ 31'barman cron' は、ロックファイルのディレクトリのアクセス許可をチェックします
- 修正したバグ 32 cron アクティビティ中の xlog.db 読み取りアクセス

22.62 1.2.32013-09-05

22.62.1 マイナー変更

- PostgreSQL 9.3 のサポートを追加しました
- “-target-name”復旧オプションのサポートを追加しました。これにより、以前に pg_create_restore_point で指定した名前付けポイントに復元できます PostgreSQL 9.1 以降のユーザーのみ

- Python 3 互換性の導入

22.62.2 バグ修正

- barman.lockfile での flock() の使用に関するバグ 27 を修正しました Damon Snyder damonsnyder@users.sf.net に感謝します。

22.63 1.2.22013-06-24

22.63.1 バグ修正

- Python 2.6 の互換性を修正

22.64 1.2.12013-06-17

22.64.1 マイナー変更

- バックアップおよびリカバリー操作の I/O 帯域幅 KBPS 単位を制限できる「bandwidth_limit」グローバル/サーバーオプションを追加しました
- テーブルスペースごとにバックアップおよびリカバリー操作の I/O 帯域幅 KBPS 単位を制限できる「tablespace_bandwidth_limit」グローバル/サーバーオプションを追加しました
- デフォルトの場所として/etc/barman/barman.conf を追加しました

22.64.2 バグ修正

- FAILED バックアップでの minimum_redundancy チェックのトリガーを回避します Jérôme Vanandruel に感謝します

22.65 1.2.02013-01-31

22.65.1 重要な変更

- 保持ポリシーを適用する方法を定義する「retention_policy_mode」グローバル/サーバーオプションを追加しました現在は「auto」のみ
- サーバーに保持するバックアップの最小数を定義する“minimum_redundancy”グローバル/サーバーオプションを追加しました
- 冗長性たとえば リダンダンシー 4 またはリカバリウィンドウ例 3 か月のリカバリウィンドウに基づいてリテンションポリシー管理を定義する「retention_policy」グローバル/サーバーオプションを追加しました。
- ロギングインフラストラクチャ、“check”および“status”コマンドにリテンションポリシーのサポートを追加しました
- “check”コマンドに最小冗長制御が統合されました

- 保持ポリシーの状態有効、廃止、および廃止の可能性を「show-backup」および「list-backup」コマンドに追加しました
- `all`キーワードはサーバー名として禁止されました
- –nagios オプションを介して'check' コマンドに Nagios プラグイン出力の基本サポートを追加しました
- Barman には、argh => 0.21.2 および argcomplete-が必要になりました
- マイナーなバグ修正

22.66 1.1.22012-11-29

22.66.1 マイナー変更

- ディレクトリから複数のサーバー構成ファイルを含めることができる「configuration_files_directory」オプションを追加しました
- 特別なバックアップ ID のサポート最新、最後、古い、最初
- `list-backup`コマンドへの複数のサーバーの管理。 `barman list-backup all' は、構成されているすべてのサーバーのバックアップをリストするようになりました。
- PostgreSQL > = 9.0 の“application_name”管理を追加

22.66.2 バグ修正

- 修正したバグ 18. 削除中に見つからない場合は、不足している WAL ファイルを無視します

22.67 1.1.12012-10-16

22.67.1 バグ修正

- recovery コマンドの回帰を修正します。

22.68 1.1.02012-10-12

22.68.1 重要な変更

- `pre_backup_script`および `post_backup_script`構成オプションを介して `backup`コマンドの前後に実行されるフックスクリプトのサポート。
- `backup`コマンドへの複数のサーバーの管理。 `barmanbackup all' は、構成されているすべてのサーバーを反復的にバックアップするようになりました。
- postgresql.conf ファイルでファイルの場所オプションが定義されている場合、リカバリーに警告を追加します 問題 10
- 宛先ディレクトリに `:`文字が含まれる場合クローズ#4 または無効なテーブルスペース再配置ルールが渡された場合、recover コマンドでフェイルファースト

- 排他的バックアップが既に実行されているために pg_start_backup() 呼び出しが失敗した場合、情報メッセージをレポートしますクローズ#8

22.68.2 バグ修正

- バグ 9.2 問題を修正しました pg_tablespace_location()

22.69 1.0.02012-07-06

22.69.1 重要な変更

- 異なるバージョンの複数の PostgreSQL サーバーのバックアップ。 PostgreSQL 8.4 以降のバージョンがサポートされています。
- 安全なリモートバックアップのサポート SSH を介して
- すべてのサーバーのバックアップのカタログの管理。ユーザーは新しいバックアップを簡単に作成、古いバックアップを削除、または復元できます。
- 事前定義された gzip および bzip2 またはカスタムの両方の圧縮/解凍フィルタを使用して、サーバーごとに構成できる WAL ファイルの圧縮
- グローバルおよびサーバーごとの指示子を使用した INI 構成ファイルのサポート。構成ファイルのデフォルトの場所は/etc/barman.conf または~/.barman.conf です。 `-c`オプションを使用すると、ユーザーは別のオプションを指定できます
- ローカルデータベースを必要としない、ベースバックアップと WAL セグメントの単純なインデックス作成
- メンテナンスモード `cron`コマンドから呼び出されます。WAL のアーカイブと圧縮、カタログの更新などの通常の操作を実行します。
- Barman で構成された特定の PostgreSQL サーバーの完全な物理ベースバックアップを取得する'backup' コマンドを追加しました
- 特定のバックアップのローカルリカバリを実行する `recover' コマンドを追加し、DBA がポイント・インタイムを指定できるようにします。 `recover`コマンドは、PGDATA ディレクトリと、該当する場合はテーブルスペースの両方の再配置をサポートしています
- バックアップのリモートリカバリーのための `-remote-ssh-command' オプションを `recover' コマンドに追加しました。リモートリカバリーは現在テーブルスペースの再配置をサポートしていません
- barman で構成されたすべてのアクティブサーバーをリストする `list-server' コマンドを追加しました
- 特定のサーバーの関連情報を表示する `show-server`コマンドを追加しました すべての構成オプションを含む
- Postgres バージョン、現在のトランザクション ID、archive コマンドなどを含むサーバーの現在の状態に関する情報を表示する `status' コマンドを追加しました。
- Barman が必要とするすべてが正常に機能している場合に 0 を返す 'check' コマンドを追加しました

- ベースバックアップのサイズ、関連する WAL セグメントの合計サイズを含む、特定のサーバーで使用可能なすべてのバックアップをリストする `list-backup' コマンドを追加しました
- 開始時刻、サイズ、関連する WAL セグメントの数とそのサイズなど、特定のバックアップの関連情報を表示する `show-backup' コマンドを追加しました。
- カタログからバックアップを削除する `delete`コマンドを追加しました
- 単一のバックアップのすべてのファイルをリストする `list-files' コマンドを追加しました
- RHEL 5/6 の RPM パッケージ

第23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

索引

A

AWS, **303**

B

Barman, **303**

D

DBA, **303**
DEB, **303**

G

GCP, **303**
GPG, **303**

I

IAC, **303**
ICT, **303**

L

libpq, **303**

P

PGDATA, **303**
PGDG, **303**
PITR, **303**

R

RHEL, **304**
RPM, **304**
RPO, **304**

S

SLES, **304**
SPOF, **304**

V

VLDB, **304**

W

WORM, **304**

ン

外部構成ファイル, **303**