



Introduction to pgBackRest

リリース *2.40.0*

EnterpriseDB Corporation

2023 年 01 月 05 日

目次

第 1 章	‘Installation’	3
1.1	利用可能なプラットフォーム	3
1.2	Debian/Ubuntu	3
1.3	RHEL/CentOS	3
1.4	EDB Postgres Advanced Server	4
第 2 章	Quick Start	5
2.1	構成	5
2.2	バックアップ	6
2.3	バックアップ情報	7
2.4	復元	7
2.5	回復対象	7
2.6	用語集	8
2.7	pgBackRest	8
2.7.1	復元コマンド	8
2.8	PostgreSQL	8
第 3 章	‘Recommended Settings’	9
3.1	暗号化	9
3.2	最大プロセス数	9
3.3	バックアップのファストスタート	9
3.4	デルタ	10
3.5	ログレベル	10
3.6	用語集	10
3.7	pgBackRest	10
第 4 章	Retention Policy	11
4.1	完全バックアップの保持	11
4.2	差分バックアップ保持	12
4.3	アーカイブの保持	12
4.4	repo1-retention-archive-type	12
4.5	repo1-retention-archive	13
4.6	用語集	13
4.7	pgBackRest	13
第 5 章	Running pgBackRest Locally on the Database Host’	15
5.1	説明	15
5.2	構成	15
5.3	グローバルセクション	15
5.3.1	マウントポイント	15
5.3.2	S3 互換のオブジェクトストア	16

5.3.3	Azure 互換のオブジェクトストア	16
5.3.4	GCS 互換のオブジェクトストア	17
5.3.5	推奨設定	17
5.3.6	保持ポリシー	17
5.4	スタンザセクション	18
5.5	アーカイブの設定	18
5.6	バックアップと復元	19
5.7	用語集	19
5.8	pgBackRest	19
5.9	PostgreSQL	19
第 6 章	Running pgBackRest from a Dedicated Repository Host'	21
6.1	説明	21
6.2	リポジトリホストで専用ユーザーを作成する	21
6.3	パスワードレス SSH 接続のセットアップ	21
6.4	構成	22
6.5	バックアップサーバー	22
6.6	データベースサーバー	23
6.7	アーカイブの設定	24
6.8	バックアップと復元	24
6.9	用語集	24
6.10	pgBackRest	24
6.11	PostgreSQL	25
第 7 章	'Multiple Repositories'	27
7.1	構成	27
7.2	初期化	28
7.3	スタンザ作成コマンド	28
7.4	チェックコマンド	28
7.5	アーカイブプッシュコマンド	29
7.6	バックアップ	31
7.7	バックアップコマンド	31
7.8	情報コマンド	32
7.9	復元	35
7.10	復元コマンド	37
7.11	アーカイブ取得コマンド	39
第 8 章	'TLS connections'	43
8.1	説明	43
8.2	リポジトリホストで専用ユーザーを作成する	43
8.3	証明書の生成	44
8.4	構成	45
8.5	バックアップを実行する	48
8.6	ストリーミングレプリケーション用にサーバーを準備する	49
8.7	スタンバイサーバーをセットアップする	50
8.8	スタンバイサーバーからバックアップを取る	51
8.9	結論	52

第 9 章	‘Appendix’	53
9.1	概念	53
9.2	バックアップ	53
9.3	復元する	54
9.4	ログ先行書き込み (WAL)	54
9.5	暗号化	54
9.6	構成リファレンス	54
9.7	グローバル	55
9.8	スタンザ	55
9.9	コマンドライン引数と環境変数	55
9.10	オプションの優先順位	56
9.11	スタンザでの複数のデータベースホストの構成	56
第 10 章	‘Frequently Asked Questions’	57

pgBackRest は、組織が PostgreSQL データベースのディザスターリカバリーソリューションを実装できるようにすることを目的とした使いやすいバックアップと復元ツールです。

注釈: pgBackRest のドキュメントをお探しですか? <http://www.pgbackrest.org> に向かいます

pgBackRest

v2.40 は現在の安定版リリースです。リリースノートは [Releases](#) ページにあります。

EDB は、このオープンソースソフトウェアでコミュニティと協力しています。パッケージは **PostgreSQL** コミュニティによって提供されています。このドキュメントに示されているユースケースはすべて、**EDB-PostgresAdvancedServer** によって完全にテストおよびサポートされています。

第1章 ‘Installation’

1.1 利用可能なプラットフォーム

pgBackRest パッケージは、次のプラットフォームで利用できます。

1.2 Debian/Ubuntu

pgBackRest をインストールするには、[PostgreSQL apt](#) リポジトリを構成してから実行します。

```
$ sudo apt-get install pgbackrest
```

次の追加パッケージがインストールされます。

- postgresql-client-common
- postgresql-common

1.3 RHEL/CentOS

pgBackRest は [PostgreSQL yum](#) リポジトリからインストールでき、Enterprise Linux 用の Extra Packages ([EPEL](#)) が必要です。

リポジトリを構成したら、次のコマンドを実行して pgBackRest をインストールします。

```
$ sudo yum install pgbackrest
```

依存関係を満たすために、次の追加パッケージをインストールできます。

- libpq5
- libzstd

1.4 EDB Postgres Advanced Server

PGDG パッケージは **PostgreSQL** ライブラリを取得し、`postgres` システムユーザーを作成します。

EDBPostgresAdvancedServer がシステムに既にインストールされている場合、そのときにインストールされたライブラリが使用されます。

デフォルトでは、パッケージは `postgres` が所有する次のディレクトリを作成します。

- `/var/lib/pgbackrest`
- `/var/log/pgbackrest`
- `/var/spool/pgbackrest`

`pg_hba.conf` 設定では、`postgres` ユーザーが `pgbackrest` コマンドを実行できる必要があります。

EDBPostgresAdvancedServer を使用している場合、`enterprisedb` システムユーザーは `pgbackrest` コマンドを実行します。次のコマンドは、`pgBackRest` ディレクトリの所有権を変更します。

```
$ sudo chown -R enterprisedb: /var/lib/pgbackrest
$ sudo chown -R enterprisedb: /var/log/pgbackrest
$ sudo chown -R enterprisedb: /var/spool/pgbackrest
```

第2章 Quick Start

このセクションでは、基本的な pgBackRest 構成プロセスと、ローカルで実行されているデータベースクラスターの最初のバックアップを作成する方法について説明します。また、`backup`、`restore`、および `info` コマンドについても簡単に紹介します。

このセクションに進む前に、'Installation' 手順に従う必要があります。

2.1 構成

デフォルトの pgBackRest 構成ファイルの場所は `/etc/pgbackrest/pgbackrest.conf` です。存在しない場合、次に `/etc/pgbackrest.conf` が使用されます。

構成パラメーターの詳細については、`appendix section` を参照してください。

次の例では、ローカルホストで実行されているデータベースクラスターのバックアップを作成するように `demo` スタンザを構成します。

EDBPostgresAdvancedServer の場合：

```
[global]
repo1-path=/var/lib/edb/as13/backups

[demo]
pg1-path=/var/lib/edb/as13/data
pg1-user=enterprisedb
pg1-port=5444
```

PostgreSQL の場合：

```
[global]
repo1-path=/var/lib/pgsql/13/backups

[demo]
pg1-path=/var/lib/pgsql/13/data
pg1-user=postgres
pg1-port=5432
```

pgBackRest を構成したら、データベースアーカイバプロセスを設定します。

```
# postgresql.conf

archive_mode = on
archive_command = pgbackrest --stanza=demo archive-push %p
```

`archive_mode` パラメーターを変更するにはサービスの再起動が必要であり、`archive_command` を変更するには構成のリロードのみが必要であるため、新しいデータベースクラスターを開始するときに空の `archive_command` で `archive_mode` を有効にすることをお勧めします。

システムユーザー (`postgres` または `enterprisedb`) の ID を想定し、pgBackRest リポジトリを開始します。

```
$ pgbackrest --stanza=demo --log-level-console=info stanza-create
```

構成とアーカイブプロセスを確認します。

```
$ pgbackrest --stanza=demo --log-level-console=info check
```

2.2 バックアップ

pgBackRest を呼び出すときは、次のいずれかのバックアップタイプを指定できます。

- フルバックアップ - `full` - すべてのデータベースクラスターファイルをコピーします。 - 以前のバックアップへの依存関係はありません。

例：

```
$ pgbackrest --stanza=demo --type=full backup
```

- インクリメンタルバックアップ - `incr` - 最後のバックアップ以降に変更されたデータベースクラスターファイルのみをコピーします。 - 他のバックアップに依存して増分バックアップを復元します。

例：

```
$ pgbackrest --stanza=demo --type=incr backup
```

- 差分バックアップ - `diff` - 最後の完全バックアップ以降に変更されたデータベースクラスターファイルのみをコピーします。 - 増分バックアップに似ていますが、常に最後の完全バックアップに依存します。

例：

```
$ pgbackrest --stanza=demo --type=diff backup
```

増分または差分バックアップの基になる完全バックアップがない場合、pgBackRest は代わりに完全バックアップモードで実行されます。

2.3 バックアップ情報

`info` コマンドを使用して、バックアップに関する情報を取得します。

```
$ pgbackrest --stanza=demo info
stanza: demo
  status: ok
  cipher: none

  db (current)
    wal archive min/max (13-1): 0000000010000000000000000001/0000000010000000000000000007

    full backup: 20201019-162148F
      timestamp start/stop: 2020-10-19 16:21:48 / 2020-10-19 16:21:58
      wal start/stop: 0000000010000000000000000003 / 0000000010000000000000000003
      database size: 23.9MB, backup size: 23.9MB
      repository size: 2.9MB, repository backup size: 2.9MB
  ...
```

デフォルトでは、`info` コマンドは、要求されたスタンザのバックアップの要約を提供します。--stanza を指定しないと、`info` コマンドはすべてのスタンザで動作します。

2.4 復元

`restore` コマンドを使用して、バックアップを復元します。--delta オプションを使用すると、既存のデータベースクラスタファイルを保持し、必要な場合にのみ回復できます。--delta オプションを指定しない場合、すべてのファイルが復元されます。`restore` コマンドを実行する前に、データベースクラスタを停止し、(--delta オプションが使用されている場合を除く) データディレクトリおよびテーブルスペースからすべてのファイルを削除します。

```
$ pgbackrest --stanza=demo restore
```

現在、バックアップの一貫性を確認する唯一の方法は、バックアップを復元することです。その後、定期的にバックアップの復元を試すことをお勧めします。

2.5 回復対象

デフォルトでは、`restore` コマンドは最新のバックアップを復元し、リポジトリで見つかったアーカイブされたすべての WAL ファイルを適用します。

復元するリカバリターゲットのタイプを選択するには、次のいずれかのキーワードを使用して--type コマンドラインオプションを含めます。

- `default` - アーカイブストリームの最後まで回復します。
- `immediate` - データベースが一貫するまでのみ回復します。

- **lsn** - LSN（ログシーケンス番号）に回復します。
- **name** - 復元ポイントまで回復します。
- **xid** - トランザクション ID に回復します。
- **time** - 特定のタイムスタンプに回復します。
- **preserve** - 既存のリカバリ設定を保持します。
- **standby** - `standby_mode=on` または `standby.signal` を追加します。
- **none** - リカバリ設定が提供されていません。

`--type` が **lsn**、**name**、トランザクション ID (**xid**)、または **time** に設定されている場合、`--target` オプションを使用して復元ポイントを指定します。

pgBackRest は、復旧ターゲット時間が指定されている場合、復元する正しいバックアップセットを自動的に選択します。それ以外の場合は、最新のバックアップセットが使用されます。特定のバックアップセットを復元するには、`--set` オプションを使用します。

`recovery_target_action` PostgreSQL 設定を変更するには、`restore --target-action` オプションを使用します。

- **pause** - リカバリターゲットに達したら一時停止します（デフォルト）。
- **promote** - リカバリターゲットに達したときにタイムラインをプロモートして切り替えます。
- **shutdown** - リカバリターゲットに達したらサーバーをシャットダウンします。

2.6 用語集

2.7 pgBackRest

デルタ log-level-console pg-path pg-port pg-user repo-path

2.7.1 復元コマンド

`restore --set restore --target restore --target-action restore --type`

2.8 PostgreSQL

`archive_command archive_mode recovery_target_action standby_mode`

第3章 ‘Recommended Settings’

このセクションでは、pgBackRest を使用する際の推奨設定について説明します。

3.1 暗号化

次の設定を使用して、リポジトリ `repo1` 内のファイルを暗号化します。

```
repo1-cipher-pass=FIXME  
repo1-cipher-type=aes-256-cbc
```

`cipher-pass` は **base64** でエンコードされている必要があります。たとえば、ランダムなパスフレーズを生成するには、次の設定を使用します。

```
$ openssl rand -base64 48
```

3.2 最大プロセス数

次の設定を使用して、圧縮の使用とファイル転送の最大プロセス数を設定します。

```
process-max=FIXME
```

各プロセスは圧縮と転送を実行してコマンドを高速に実行しますが、サーバーの負荷とパフォーマンスに影響を与えるほどに `process-max` を設定しないでください。

3.3 バックアップのファストスタート

`start-fast=y` 設定を使用して、チェックポイントでバックアップをすばやく開始します。これは、pgBackRest 内で **true** を `pg_start_backup()` データベース関数の `fast` パラメーターに渡すことにより実現されます。バックアップは、データベースサーバ自身自身によってトリガーされる次の定期チェックポイントを待たずに、すぐに開始されます。

3.4 デルタ

チェックサムを使用して復元またはバックアップするには、`delta=y` 設定を使用します。

復元中、`data` ディレクトリと `tablespaces` ディレクトリは存在するはずですが、空です。このオプションは、チェックサムを使用してデルタ復元を実行します。

バックアップ中に、このオプションはタイムスタンプの代わりにチェックサムを使用して、ファイルをコピーするかどうかを決定します。

3.5 ログレベル

各コマンドの出力は、コンソールとログファイルに出力されます。

次の `log-level-console` および `log-level-file` 設定を使用して、ログレベルを適用します。

- オフ；
- エラー；
- 警告；
- 情報；
- 詳細；
- デバッグ；
- トレース。

注意：トレースレベルのログ記録により、キーやパスワードなどのシークレットが公開される場合があります。

実行したコマンドでエラーが発生した場合にログファイルに出力を追加し、詳細についてコマンドの再実行を回避するには、次の設定を使用します。

```
log-level-console=info
log-level-file=debug
```

3.6 用語集

3.7 pgBackRest

デルタ `log-level-console` `log-level-file` `process-max` `repo-cipher-pass` `repo-cipher-type`

第4章 Retention Policy

保持ポリシーオプションを設定すると、古いバックアップを削除できます。これらの設定は、領域の節約とデータ保持のトレードオフになります。

4.1 完全バックアップの保持

完全バックアップの保持は、`repo1-retention-full-type` および `repo1-retention-full` オプションによって決定されます。

`repo1-retention-full-type` オプションを `count` (デフォルト) に設定して、特定の数の完全バックアップを保持できます。例として、`repo1-retention-full-type=count` および `repo1-retention-full=2` の場合、pgBackRest は最後に成功した 2 つの完全バックアップを保持します。

または、`repo1-retention-full-type` オプションを `time` に設定して、特定の期間完全バックアップを保持できます。この構成では、保持要件を満たすために、保持期間よりも古い完全バックアップが少なくとも 1 つ残ります。日単位で測定された `repo1-retention-full` 値より古い他のすべての完全バックアップは、リポジトリから削除されます。例として、`repo1-retention-full-type=time` および `repo1-retention-full=30` (日) の場合、30 日より古い完全バックアップが 1 つ保持されます。この例で、リポジトリが 5 つの完全バックアップ (1 つは 25 日、1 つは 35 日、残りは古い) を保持する場合、30~25 日前の期間に復元するには 35 日前の完全バックアップが引き続き必要ですであり、構成された保持期間よりも古い場合でも保持されます。他のすべての古いバックアップは、保持メンテナンス中に削除されます。

完全バックアップが期限切れになると、それに関連付けられているすべての差分バックアップと増分バックアップも期限切れになります。`repo1-retention-full` オプションが定義されていない場合、警告が発行されます。

残りの最も古い完全バックアップよりも古いアーカイブされた WAL ファイルは自動的に削除されますが、この動作はアーカイブ保持設定 (`repo1-retention-archive-type` および `repo1-retention-archive`) で変更できます。

`repo1-retention-full` オプションを最大値 (9999999) に設定して、すべてのバックアップを無期限に保持します。

4.2 差分バックアップ保持

差分バックアップ保持は、`repo1-retention-diff` オプションによって決定されます。

差分バックアップが期限切れになると、差分バックアップに関連付けられているすべての増分バックアップも期限切れになります。

定義されていない場合、すべての差分バックアップは、依存する完全バックアップの有効期限が切れるまで保持されます。

4.3 アーカイブの保持

pgBackRest では、アーカイブの特定の保持ポリシーを定義できます。

警告: pgBackRest は、バックアップの開始場所に基づいて、クリーニングするアーカイブを定義します。場合によっては、リポジトリ内の最も古いアーカイブされた WAL ファイルが最初のバックアップの開始場所よりも古い場合があります。その場合、最初のバックアップの有効期限が切れるまで削除されません。

4.4 `repo1-retention-archive-type`

`repo1-retention-archive-type` オプションを設定して、WAL 保持の対象となるバックアップタイプを定義できます。

`repo1-retention-archive-type` オプションが `full`（デフォルト）に設定されている場合、pgBackRest は `repo1-retention-archive` で定義された数の完全バックアップについてアーカイブされた WAL ファイルを保持します。

`repo1-retention-archive-type` オプションが `diff`（差分）に設定されている場合、pgBackRest は `repo1-retention-archive` で定義された数の完全バックアップと差分バックアップを保持します。例として、最後に作成されたバックアップが完全バックアップの場合、リポジトリの保持では差分としてカウントされます。

`repo1-retention-archive-type` オプションが `incr`（インクリメンタル）に設定されている場合、pgBackRest は `repo1-retention-archive` で定義された数の完全、差分、およびインクリメンタルバックアップのアーカイブを保持します。

警告: `repo1-retention-archive-type` オプションをデフォルトのままにすることをお勧めします。その場合、完全バックアップとともにアーカイブが削除されます。

4.5 repo1-retention-archive

`repo1-retention-archive` オプションを設定して、継続的な WAL 保持のために保持されるバックアップの数を定義できます。

警告: バックアップの一貫性を保つために必要な（完全および差分バックアップの保持ポリシーに基づいて）WAL セグメントは、`repo1-retention-archive` オプションの構成に関係なく、バックアップが期限切れになるまで常に保持されます。

`repo1-retention-archive` は他のオプションに大きく依存しています。

- `repo1-retention-full-type` が `time` に設定されている場合、`repo1-retention-archive` 値はデフォルトで、`repo1-retention-full` 設定を満たした後に保持された最も古い完全バックアップよりも古いアーカイブされた WAL ファイルを削除します。
- `repo1-retention-archive-type` を `incr` に設定する場合、`repo1-retention-archive` を設定する必要があります。
- `repo1-retention-archive` 値が設定されておらず、`repo1-retention-full-type` が `count`（デフォルト）に設定されている場合、`repo1-retention-archive-type` 値はアーカイブの有効期限を定義します。それぞれ `repo1-retention-diff`。この構成により、アーカイブされた WAL ファイルは、対応する期限切れのバックアップに対してのみ削除されます。

`repo1-retention-archive` 設定を `repo1-retention-archive-type` と組み合わせて使用すると、余分な WAL セグメントを削除してディスク領域を再利用できます。この構成は、有効期限が切れた WAL を含むバックアップからポイントインタイムリカバリを実行する機能を無効にするため、**お勧めしません**。

4.6 用語集

4.7 pgBackRest

`repo-retention-archive-type` `repo-retention-archive` `repo-retention-diff` `repo-retention-full-type` `repo-retention-full`

第5章 Running pgBackRest Locally on the Database Host'

5.1 説明

- pgBackRest はデータベースサーバーでローカルに実行され、リポジトリをリモートストレージ（NFS、S3、Azure または GCS 互換のオブジェクトストア）に保存します。

*バックアップを取るノードでアクティブな cron タスク。

- 別のサーバーからバックアップを取得するように cron タスクを手動で再構成します。

! Use case diagram

5.2 構成

5.3 グローバルセクション

[global] セクションには次が含まれます。

*リポジトリの場所;

*推奨設定;

*保持ポリシー。

このセクションの設定は、すべてのノード（プライマリとスタンバイ）で同じである必要があります。

5.3.1 マウントポイント

repo1-type を **posix** または **cifs** に設定し、共有スペースにバックアップとアーカイブを保存するための repo1-path の場所を指定します。

```
repo1-type=posix
repo1-path=/backup_space
```

5.3.2 S3 互換のオブジェクトストア

```
repo1-type=s3
repo1-path=/repo1
repo1-s3-bucket=bucket-name
repo1-s3-endpoint=...
repo1-s3-key=accessKey1
repo1-s3-key-secret=***
repo1-s3-region=...
```

バックアップとアーカイブは `bucket-name/repo1` に保存されます。

AWS インスタンスが IAM ロールに関連付けられている場合、一時的な認証情報を自動的に取得できます。資格情報は、有効期限が切れてから 5 分以内に自動的に更新されます。

これを行うには、`repo1-s3-key` と `repo1-s3-key-secret` を設定する代わりに、次を使用します。

```
repo1-s3-key-type=auto
repo1-s3-role=authrole
```

Web ID 資格情報を自動的に取得するには、`repo1-s3-key-type=web-id` オプションを設定し、`AWS_ROLE_ARN` および `AWS_WEB_IDENTITY_TOKEN_FILE` 環境変数を定義します。

`repo1-s3-kms-key-id` オプションを設定して、指定した AWS キーマネージメントサービスキーを使用したサーバー側の暗号化を有効にします。

5.3.3 Azure 互換のオブジェクトストア

```
repo1-type=azure
repo1-azure-key-type=shared
repo1-path=/repo1
repo1-azure-container=container-name
repo1-azure-account=pgbackrest
repo1-azure-key=***
```

Shared Access Signature を使用するには、`repo1-azure-key-type` オプションを `sas` に設定し、`repo1-azure-key` オプションを Shared アクセス署名トークンに設定します。

5.3.4 GCS 互換のオブジェクトストア

```
repo1-type=gcs
repo1-gcs-key-type=service
repo1-path=/repo1
repo1-gcs-bucket=bucket-name
repo1-gcs-key=/etc/pgbackrest/gcs-key.json
```

`repo1-gcs-key` は、`repo1-gcs-key-type` オプションに応じてトークン または サービス キーファイルです。

5.3.5 推奨設定

これらの設定を使用して、暗号化と並列操作を有効にし、コンソールとログファイルに十分な情報を表示します。

```
repo1-cipher-type=aes-256-cbc
repo1-cipher-pass=FIXME
process-max=FIXME
log-level-console=info
log-level-file=debug
start-fast=y
delta=y
```

暗号化パスフレーズと、圧縮の使用とファイル転送に使用するプロセスの最大数を調整します。

詳細については、'Recommended Settings' ページを参照してください。

5.3.6 保持ポリシー

古いバックアップとアーカイブを削除できるようにするには、保持ポリシーを定義する必要があります。次の設定を使用して、最新の完全バックアップのみを保持します。

```
repo1-retention-full=1
```

その他のオプションと詳細については、[Retention Policy](#) ページを参照してください。

5.4 スタンザセクション

[demo] スタンザセクションはデータベースサーバーごとに異なり、次のものが含まれます。

- **PostgreSQL** 構成ファイルの場所

```
pg1-path=/var/lib/pgsql/13/data
```

または **EDBPostgresAdvancedServer**

```
pg1-path=/var/lib/edb/as13/data
pg1-user=enterprisedb
pg1-port=5444
```

- 他のすべての **PostgreSQL** ノード

```
pg2-host=other_node_hostname_or_ip
pg2-path=/var/lib/pgsql/13/data
```

または **EDBPostgresAdvancedServer**

```
pg2-host=other_node_hostname_or_ip
pg2-host-user=enterprisedb
pg2-path=/var/lib/edb/as13/data
```

- **restore** コマンドの実行後にレプリケーション **primary_conninfo** を設定する特定のリカバリオプション：

```
recovery-option=primary_conninfo=host=primary_vip user=replication_user ...
```

5.5 アーカイブの設定

pgBackRest を構成したら、各ノードでデータベースアーカイバプロセスを設定します。

```
# postgresql.conf

archive_mode = on
archive_command = pgbackrest --stanza=demo archive-push %p
```

archive_mode パラメーターを変更するにはサービスの再起動が必要であり、**archive_command** を変更するには構成のリロードのみが必要であるため、新しいデータベースクラスターを開始するときに空の **archive_command** で **archive_mode** を有効にすることをお勧めします。

単純な **デモ** スタンザ構成をセットアップするには、

Quick Start ページを参照してください。

システムユーザー (**postgres** または **enterprisedb**) の ID を想定し、プライマリサーバーから pgBackRest リポジトリを開始します。

```
$ pgbackrest --stanza=demo stanza-create
```

構成とアーカイブプロセスを確認します。

```
$ pgbackrest --stanza=demo check
```

5.6 バックアップと復元

これら2つのコマンドの詳細については、クイックスタートの [バックアップ](#) および [復元](#) セクションを参照してください。

5.7 用語集

5.8 pgBackRest

デルタ log-level-console log-level-file pg-host-user pg-host pg-path pg-port pg-user process-max recovery-option repo-azure-account repo-azure-container repo-azure-key-type repo-azure-key repo-cipher-pass repo-cipher-type repo-path repo-retention-full repo-s3-bucket repo-s3-endpoint repo-s3-key-secret repo-s3-key-type repo-s3-key repo-s3-kms-key-id repo-s3-region repo-s3-role repo-type start-fast

5.9 PostgreSQL

archive_command archive_mode primary_conninfo

第6章 Running pgBackRest from a Dedicated Repository Host'

6.1 説明

- pgBackRest は専用のバックアップサーバーで実行され、ローカルまたはリモートのストレージ（NFS、S3、Azure または GCS 互換のオブジェクトストア）にリポジトリを保存します。

*プライマリデータベースクラスターからバックアップを取得する cron タスクがアクティブです。

- 別のサーバーからバックアップを取得するように cron タスクを手動で再構成します。

! Use case diagram

6.2 リポジトリホストで専用ユーザーを作成する

pgbackrest ユーザーは、バックアップとアーカイブのリポジトリを所有するために作成されます。すべてのユーザーがリポジトリを所有できますが、混乱を避けるために **postgres** または **enterprisedb** を使用しないことをお勧めします。

```
$ sudo groupadd pgbackrest
$ sudo adduser -g pgbackrest -n pgbackrest
$ sudo chown pgbackrest: /var/log/pgbackrest
```

6.3 パスワードレス SSH 接続のセットアップ

pgBackRest では、ホスト間の通信を有効にするためにパスワードなしの SSH 接続が必要です。

次の SSH コマンドの例は、パスワードの入力を求めずに成功します。

```
# From the PostgreSQL nodes

$ sudo -u postgres ssh pgbackrest@backup-server

# From the EDB Postgres Advanced Server nodes

$ sudo -u enterprisedb ssh pgbackrest@backup-server
```

(次のページに続く)

```
# From the repository host

$ sudo -u pgbackrest ssh postgres@postgres-node-1
$ sudo -u pgbackrest ssh postgres@postgres-node-2
$ ...
```

6.4 構成

6.5 バックアップサーバー

リポジトリは、バックアップサーバーから到達可能である必要があります。

first use case で説明されているさまざまなサポートされているタイプにあります。

たとえば、バックアップとアーカイブをローカルに保存する `/backup_space` を作成しましょう。

```
$ sudo mkdir /backup_space
$ sudo chown pgbackrest: /backup_space
```

構成ファイルの `[global]` セクションは次のようになります。

```
[global]
repo1-path=/backup_space
repo1-retention-full=2
repo1-cipher-type=aes-256-cbc
repo1-cipher-pass=FIXME
process-max=FIXME
log-level-console=info
log-level-file=debug
start-fast=y
delta=y
```

暗号化パスフレーズと、圧縮の使用とファイル転送に使用するプロセスの最大数を調整します。

詳細については、'Recommended Settings' のページを参照してください。

[demo] スタンザセクションには、データベースノードごとに特定の `pg-` オプションが含まれます。

- PostgreSQL のプライマリデータベースサーバー構成

```
pg1-host=postgres-node-1
pg1-path=/var/lib/pgsql/13/data
```

または **EDBPostgresAdvancedServer**

```
pg1-host=postgres-node-1
pg1-host-user=enterprisedb
pg1-path=/var/lib/edb/as13/data
```

- 他のすべての **PostgreSQL** ノード

```
pg2-host=other_node_hostname_or_ip
pg2-path=/var/lib/pgsql/13/data
```

または **EDBPostgresAdvancedServer**

```
pg2-host=other_node_hostname_or_ip
pg2-host-user=enterprisedb
pg2-path=/var/lib/edb/as13/data
```

pg2、pg3、…を使用して複数のノードを定義して、それらのサーバーが使用可能な場合にのみスタンバイサーバーに到達します。

6.6 データベースサーバー

データベースサーバー上の pgBackRest 構成ファイルを更新します。

```
# pgbackrest.conf

[global]
repo1-host=backup-server
repo1-host-user=pgbackrest
log-level-console=info
log-level-file=debug
delta=y

[demo]
pg1-path=/var/lib/pgsql/13/data
recovery-option=primary_conninfo=host=primary_vip user=replication_user ...
```

詳細については、`demo` ページ。

6.7 アーカイブの設定

pgBackRest を構成したら、各ノードでデータベースアーカイブプロセスを設定します。

```
# postgresql.conf

archive_mode = on
archive_command = pgbackrest --stanza=demo archive-push %p
```

`archive_mode` パラメーターを変更するにはサービスの再起動が必要であり、`archive_command` を変更するには構成のリロードのみが必要であるため、新しいデータベースクラスターを開始するときに空の `archive_command` で `archive_mode` を有効にすることをお勧めします。

バックアップサーバー (リポジトリホストとも呼ばれます) から、前に作成したシステムユーザーで pgBackRest リポジトリを開始します。

```
$ sudo -u pgbackrest pgbackrest --stanza=demo stanza-create
```

構成とアーカイブプロセスを確認します。

```
$ sudo -u pgbackrest pgbackrest --stanza=demo check
```

6.8 バックアップと復元

これら 2 つのコマンドの詳細については、クイックスタートの [バックアップ](#) および [復元](#) セクションを参照してください。

`backup` コマンドはリポジトリホスト (別名 `_backup-server_`) で実行されますが、`restore` コマンドはデータベースノード (別名 `_postgres-node-1_`、…) で実行する必要があります。

6.9 用語集

6.10 pgBackRest

デルタ log-level-console log-level-file pg-host pg-path pg-port pg-user process-max recovery-option
repo-cipher-pass repo-cipher-type repo-host repo-host-user repo-path repo-retention-full start-fast

6.11 PostgreSQL

archive_command archive_mode primary_conninfo

第7章 ‘Multiple Repositories’

v2.33

pgBackrest リリースの主な機能は、マルチプルのリポジトリのサポートです。これにより、さまざまなバックアップリポジトリにさまざまな動作を定義できます。例として、（スペースを節約するための）保持期間が短い高速復元用のローカルリポジトリと、コールドストレージ用に大きいリモートリポジトリを使用できます。

この機能では、コマンドを適用するリポジトリを設定するために使用できる新しい `--repo` オプションが導入されました。複数のリポジトリを構成できるように、すべての `repo*-` オプションにはインデックスが付けられています。

このセクションでは、**デモ** クラスタを（CentOS 7で **EDBPostgresAdvancedServer** バージョン 13 を使用して）セットアップして、この新機能が各 pgBackRest コマンドにどのように影響するかを示します。

7.1 構成

このデモでは、データベースホストに2つのローカルリポジトリをセットアップすることに注意してください。実稼働目的では、各リポジトリを個別のストレージに保存し、少なくとも1つのリモートリポジトリを構成することをお勧めします。

```
[global]
# repository details

repo1-path=/var/lib/edb/as13/backups/repo1
repo1-retention-full=1
repo2-path=/var/lib/edb/as13/backups/repo2
repo2-retention-full=1

# general options

process-max=2
log-level-console=info
log-level-file=debug
start-fast=y
delta=y

[demo]
pg1-path=/var/lib/edb/as13/data
```

(次のページに続く)

```
pg1-user=enterprisedb
pg1-port=5444
```

安全上の理由から、`--repo` オプションは構成ファイルで定義できません。定義すると、警告がトリガーされます：`WARN: configuration file contains command-line only option 'repo'`。複数のリポジトリが構成されており、コマンドに`--repo` オプションが指定されていない場合、優先順位が最も高いリポジトリ（たとえば、`repo1` 次に `repo2`）が選択されます。

`repo1` のみが下位互換性を維持するように構成されている場合、`--repo` オプションは必要ありません。ただし、単一のリポジトリが構成されている場合、構成で `repo1` を使用することをお勧めします。

7.2 初期化

7.3 スタンザ作成コマンド

`stanza-create` コマンドは、構成されているすべてのリポジトリで自動的に動作します。

```
$ pgbackrest --stanza=demo stanza-create
P00 INFO: stanza-create for stanza demo on repo1
P00 INFO: stanza-create for stanza demo on repo2
```

7.4 チェックコマンド

`check` コマンドは、新しい WAL セグメントのアーカイブをトリガーし、定義されたすべてのリポジトリにプッシュしようとします。

```
$ pgbackrest --stanza=demo check
P00 INFO: check repo1 configuration (primary)
P00 INFO: check repo2 configuration (primary)
P00 INFO: check repo1 archive for WAL (primary)
P00 INFO: WAL segment ... successfully archived to ... on repo1
P00 INFO: check repo2 archive for WAL (primary)
P00 INFO: WAL segment ... successfully archived to ... on repo2
```

7.5 アーカイブプッシュコマンド

archive-push コマンドは、構成されたすべてのリポジトリに WAL を常にアーカイブします。バックアップは、リポジトリごとに個別にスケジュールする必要があります。バックアップの種類と保持はリポジトリごとに異なる可能性があるため、ほとんどの場合、これが望ましいです。

archive_command は引き続き通常どおりに定義できます。

```
$ psql -d postgres -c "show archive_command;"
      archive_command
-----
pgbackrest --stanza=demo archive-push %p
(1 row)
```

これは、archive-push アクティビティを示す PostgreSQL ログの DEBUG 抽出です。

```
...
P000 DEBUG:      storage/storage::storageNewWrite: => {
    type: posix, name: {"/var/lib/edb/as13/backups/repo1/archive/demo/13-1/
→000000001000000000/
                                0000000010000000000000000005-
→be9a61a800934879842fa647b0d50385f44e0228.gz"},
    modeFile: 0640, modePath: 0750, createPath: true, syncFile: true, syncPath: true,
→atomic: true}
...
P000 DEBUG:      storage/storage::storageNewWrite: => {
    type: posix, name: {"/var/lib/edb/as13/backups/repo2/archive/demo/13-1/
→000000001000000000/
                                0000000010000000000000000005-
→be9a61a800934879842fa647b0d50385f44e0228.gz"},
    modeFile: 0640, modePath: 0750, createPath: true, syncFile: true, syncPath: true,
→atomic: true}
...
P000 INFO: pushed WAL file 0000000010000000000000000005 to the archive
```

archive-push コマンドは、WAL アーカイブを到達可能なすべてのリポジトリにプッシュしようとします。考え方は、WAL ファイルが削除されないように PostgreSQL にエラーをスローする必要がある場合でも、できるだけ多くのリポジトリにアーカイブすることです。

```
P000 INFO: archive-push command begin: [pg_wal/0000000010000000000000000006] ...
...
P000 INFO: archive-push command begin: [pg_wal/0000000010000000000000000007] ...
ERROR: [104]: archive-push command encountered error(s):
      repo2: [PathOpenError] unable to list file info for path ...: [13] Permission
→denied
P000 INFO: archive-push command end: aborted with exception [104]
DETAIL:  The failed archive command was: pgbackrest --stanza=demo archive-push pg_wal/
→0000000010000000000000000007
```

(次のページに続く)

(前のページからの続き)

```
WARNING: archiving write-ahead log file "000000010000000000000007" failed too many
→times, will try again later
```

PostgreSQL archiver プロセスはエラーを報告するはずですが。

```
$ ps -o pid,cmd fx |grep archiver
4440 \_ postgres: archiver failed on 000000010000000000000007
```

次の WAL セグメントはアーカイブしないでください。

```
$ psql -d postgres -c "SELECT pg_create_restore_point(generate some activity); SELECT
→pg_switch_wal();"
pg_switch_wal
-----
0/80001C8
(1 row)

$ ls as13/data/pg_wal/archive_status/ |grep ".ready"
000000010000000000000007.ready
000000010000000000000008.ready
```

pgBackRest 自分自身内で非同期アーカイブプロセスを使用するために `archive-async=y` を構成に追加すると、これは非常に便利になります。PostgreSQL archiver プロセスがまだスタックしている場合でも、アーカイブは作業リポジトリに到達します。

```
$ $ ls /var/lib/edb/as13/backups/repo1/archive/demo/13-1/0000000100000000/
000000010000000000000005-be9a61a800934879842fa647b0d50385f44e0228.gz
000000010000000000000006-7108296955e1208c93447dcfc5712ce59af97907.gz
000000010000000000000007-f1b6fcdd23e96a19a7a89dc92613674c2316aa49.gz
000000010000000000000008-791c44d39ef772d58e51d64c8e3589231d3916a4.gz
000000010000000000000009-47241214eb3fd5e693eed778caa7b304a778ecc2.gz

$ ls /var/lib/edb/as13/backups/repo2/archive/demo/13-1/0000000100000000/
000000010000000000000005-be9a61a800934879842fa647b0d50385f44e0228.gz
000000010000000000000006-7108296955e1208c93447dcfc5712ce59af97907.gz
```

さらに進む前に、archiver プロセスのブロックを解除（および非同期アーカイブを削除）しましょう。

```
$ $ ps -o pid,cmd fx |grep archiver
4440 \_ postgres: archiver last was 000000010000000000000000
```

7.6 バックアップ

7.7 バックアップコマンド

いくつかのバックアップを取りましょう。

```
$ pgbackrest backup --stanza=demo --type=full
P00 INFO: backup command begin ...
P00 INFO: repo option not specified, defaulting to repo1
P00 INFO: execute non-exclusive pg_start_backup(): backup begins after the
↳requested immediate checkpoint completes
P00 INFO: backup start archive = 00000000100000000000000000C, lsn = 0/C000060
P00 INFO: full backup size = 50.4MB
P00 INFO: execute non-exclusive pg_stop_backup() and wait for all WAL segments to
↳archive
P00 INFO: backup stop archive = 00000000100000000000000000C, lsn = 0/C000138
P00 INFO: check archive for segment(s)
↳00000000100000000000000000C:00000000100000000000000000C
P00 INFO: new backup label = 20210419-142212F
P00 INFO: backup command end: completed successfully

$ pgbackrest backup --stanza=demo --type=full --repo=2
P00 INFO: backup command begin ...
P00 INFO: execute non-exclusive pg_start_backup(): backup begins after the
↳requested immediate checkpoint completes
P00 INFO: backup start archive = 00000000100000000000000000E, lsn = 0/E000028
P00 INFO: full backup size = 50.5MB
P00 INFO: execute non-exclusive pg_stop_backup() and wait for all WAL segments to
↳archive
P00 INFO: backup stop archive = 00000000100000000000000000E, lsn = 0/E000138
P00 INFO: check archive for segment(s)
↳00000000100000000000000000E:00000000100000000000000000E
P00 INFO: new backup label = 20210419-142414F
P00 INFO: backup command end: completed successfully

$ pgbackrest backup --stanza=demo --type=incr --repo=1
P00 INFO: backup command begin ...
P00 INFO: last backup label = 20210419-142212F, version = ...
P00 INFO: execute non-exclusive pg_start_backup(): backup begins after the
↳requested immediate checkpoint completes
P00 INFO: backup start archive = 000000001000000000000000010, lsn = 0/10000028
P00 INFO: incr backup size = 50.5MB
P00 INFO: execute non-exclusive pg_stop_backup() and wait for all WAL segments to
↳archive
P00 INFO: backup stop archive = 000000001000000000000000010, lsn = 0/10000100
```

(次のページに続く)

(前のページからの続き)

```

P00 INFO: check archive for segment(s)
→0000000100000000000000010:0000000100000000000000010
P00 INFO: new backup label = 20210419-142212F_20210419-142502I
P00 INFO: backup command end: completed successfully

$ pgbackrest backup --stanza=demo --type=incr --repo=2
P00 INFO: backup command begin ...
P00 INFO: last backup label = 20210419-142414F, version = ...
P00 INFO: execute non-exclusive pg_start_backup(): backup begins after the
→requested immediate checkpoint completes
P00 INFO: backup start archive = 0000000100000000000000012, lsn = 0/12000028
P00 INFO: incr backup size = 50.6MB
P00 INFO: execute non-exclusive pg_stop_backup() and wait for all WAL segments to
→archive
P00 INFO: backup stop archive = 0000000100000000000000012, lsn = 0/12000100
P00 INFO: check archive for segment(s)
→0000000100000000000000012:0000000100000000000000012
P00 INFO: new backup label = 20210419-142414F_20210419-142556I
P00 INFO: backup command end: completed successfully

```

ここでは、各リポジトリで full と incr のバックアップを交互に使用しました。

7.8 情報コマンド

```

$ pgbackrest info --stanza=demo
stanza: demo
  status: ok
  cipher: none

db (current)
  wal archive min/max (13): 0000000100000000000000002/0000000100000000000000012

  full backup: 20210419-142212F
    timestamp start/stop: 2021-04-19 14:22:12 / 2021-04-19 14:22:26
    wal start/stop: 000000010000000000000000C / 000000010000000000000000C
    database size: 50.4MB, database backup size: 50.4MB
    repo1: backup set size: 8MB, backup size: 8MB

  full backup: 20210419-142414F
    timestamp start/stop: 2021-04-19 14:24:14 / 2021-04-19 14:24:26
    wal start/stop: 000000010000000000000000E / 000000010000000000000000E
    database size: 50.5MB, database backup size: 50.5MB
    repo2: backup set size: 8MB, backup size: 8MB

```

(次のページに続く)

(前のページからの続き)

```

incr backup: 20210419-142212F_20210419-142502I
timestamp start/stop: 2021-04-19 14:25:02 / 2021-04-19 14:25:04
wal start/stop: 00000000100000000000000010 / 00000000100000000000000010
database size: 50.5MB, database backup size: 613.4KB
repo1: backup set size: 8MB, backup size: 26.3KB
backup reference list: 20210419-142212F

incr backup: 20210419-142414F_20210419-142556I
timestamp start/stop: 2021-04-19 14:25:56 / 2021-04-19 14:25:57
wal start/stop: 00000000100000000000000012 / 00000000100000000000000012
database size: 50.6MB, database backup size: 661.5KB
repo2: backup set size: 8MB, backup size: 28KB
backup reference list: 20210419-142414F

```

デフォルトの順序では、リポジトリが混在する日付でバックアップがソートされます。相互に依存するバックアップを見つけるのは難しい場合があります。

次に、このビューをリポジトリごとに分割できます。

```

$ pgbackrest info --stanza=demo --repo=1
stanza: demo
  status: ok
  cipher: none

db (current)
wal archive min/max (13): 000000001000000000000000C/00000000100000000000000012

full backup: 20210419-142212F
timestamp start/stop: 2021-04-19 14:22:12 / 2021-04-19 14:22:26
wal start/stop: 000000001000000000000000C / 000000001000000000000000C
database size: 50.4MB, database backup size: 50.4MB
repo1: backup set size: 8MB, backup size: 8MB

incr backup: 20210419-142212F_20210419-142502I
timestamp start/stop: 2021-04-19 14:25:02 / 2021-04-19 14:25:04
wal start/stop: 00000000100000000000000010 / 00000000100000000000000010
database size: 50.5MB, database backup size: 613.4KB
repo1: backup set size: 8MB, backup size: 26.3KB
backup reference list: 20210419-142212F

$ pgbackrest info --stanza=demo --repo=2
stanza: demo
  status: ok
  cipher: none

```

(次のページに続く)

(前のページからの続き)

```

db (current)
wal archive min/max (13): 0000000010000000000000000000E/0000000010000000000000000012

full backup: 20210419-142414F
timestamp start/stop: 2021-04-19 14:24:14 / 2021-04-19 14:24:26
wal start/stop: 0000000010000000000000000000E / 0000000010000000000000000000E
database size: 50.5MB, database backup size: 50.5MB
repo2: backup set size: 8MB, backup size: 8MB

incr backup: 20210419-142414F_20210419-142556I
timestamp start/stop: 2021-04-19 14:25:56 / 2021-04-19 14:25:57
wal start/stop: 00000000100000000000000000012 / 00000000100000000000000000012
database size: 50.6MB, database backup size: 661.5KB
repo2: backup set size: 8MB, backup size: 28KB
backup reference list: 20210419-142414F

```

‘walarchivemin/max’ は、現在アーカイブに保存されている WAL の最小値と最大値を示し、マルチプルのリポジトリの場合、--repo オプションが設定されていない限り、すべてのリポジトリで報告されます。アーカイブの保持ポリシーまたはその他の理由により、ギャップが生じる場合があります。

そのコンテンツを削除して、最初のリポジトリを壊しましょう。

```

$ pgbackrest --stanza=demo stanza-delete --repo=1 --force
P00 INFO: stanza-delete command end: completed successfully

$ pgbackrest --stanza=demo stanza-create
P00 INFO: stanza-create for stanza demo on repo1
P00 INFO: stanza-create for stanza demo on repo2
P00 INFO: stanza demo already exists on repo2 and is valid
P00 INFO: stanza-create command end: completed successfully

```

複数のリポジトリが構成されている場合、ステータス **mixed** は、1 つ以上のリポジトリでスタンザが正常な状態でないことを示します。この場合、スタンザの状態はリポジトリごとに追加の行で詳細に説明されます。

```

$ pgbackrest info --stanza=demo
stanza: demo
  status: mixed
    repo1: error (no valid backups)
    repo2: ok
  cipher: none

db (current)
wal archive min/max (13): 0000000010000000000000000000E/0000000010000000000000000012

```

(次のページに続く)

(前のページからの続き)

```

full backup: 20210419-142414F
    timestamp start/stop: 2021-04-19 14:24:14 / 2021-04-19 14:24:26
    wal start/stop: 0000000010000000000000000000000E / 0000000010000000000000000000000E
    database size: 50.5MB, database backup size: 50.5MB
    repo2: backup set size: 8MB, backup size: 8MB

incr backup: 20210419-142414F_20210419-142556I
    timestamp start/stop: 2021-04-19 14:25:56 / 2021-04-19 14:25:57
    wal start/stop: 00000000100000000000000000000012 / 00000000100000000000000000000012
    database size: 50.6MB, database backup size: 661.5KB
    repo2: backup set size: 8MB, backup size: 28KB
    backup reference list: 20210419-142414F

```

この状態は、新しいバックアップを作成することで解決できます。

```

$ pgbackrest backup --stanza=demo --type=full --repo=1
P00  INFO: backup command begin ...
P00  INFO: execute non-exclusive pg_start_backup(): backup begins after the
→requested immediate checkpoint completes
P00  INFO: backup start archive = 000000001000000000000000000014, lsn = 0/14000028
P00  INFO: full backup size = 50.6MB
P00  INFO: execute non-exclusive pg_stop_backup() and wait for all WAL segments to
→archive
P00  INFO: backup stop archive = 000000001000000000000000000014, lsn = 0/14000138
P00  INFO: check archive for segment(s)
→000000001000000000000000000014:000000001000000000000000000014
P00  INFO: new backup label = 20210419-143400F
P00  INFO: backup command end: completed successfully

```

7.9 復元

いくつかのデータ、バックアップ、および復元ポイントで状況を開始しましょう。

```

$ createdb test
$ psql -d test -c "CREATE TABLE t1(id int);"
CREATE TABLE
$ psql -d test -c "INSERT INTO t1 VALUES (1);"
INSERT 0 1

$ pgbackrest backup --stanza=demo --type=full --repo=1
P00  INFO: backup command begin ...
P00  INFO: execute non-exclusive pg_start_backup(): backup begins after the
→requested immediate checkpoint completes

```

(次のページに続く)

(前のページからの続き)

```

P00 INFO: backup start archive = 0000000100000000000000016, lsn = 0/16000028
P00 INFO: full backup size = 63.0MB
P00 INFO: execute non-exclusive pg_stop_backup() and wait for all WAL segments to
→archive
P00 INFO: backup stop archive = 0000000100000000000000016, lsn = 0/16000138
P00 INFO: check archive for segment(s)
→0000000100000000000000016:0000000100000000000000016
P00 INFO: new backup label = 20210419-143643F
P00 INFO: backup command end: completed successfully
P00 INFO: expire command begin ...
P00 INFO: repo1: expire full backup 20210419-143400F
P00 INFO: repo1: remove expired backup 20210419-143400F
P00 INFO: expire command end: completed successfully

$ psql -d test -c "INSERT INTO t1 VALUES (2);"
INSERT 0 1
$ pgbackrest backup --stanza=demo --type=full --repo=2
P00 INFO: backup command begin ...
P00 INFO: execute non-exclusive pg_start_backup(): backup begins after the
→requested immediate checkpoint completes
P00 INFO: backup start archive = 0000000100000000000000018, lsn = 0/18000028
P00 INFO: full backup size = 63MB
P00 INFO: execute non-exclusive pg_stop_backup() and wait for all WAL segments to
→archive
P00 INFO: backup stop archive = 0000000100000000000000018, lsn = 0/18000138
P00 INFO: check archive for segment(s)
→0000000100000000000000018:0000000100000000000000018
P00 INFO: new backup label = 20210419-143749F
P00 INFO: backup command end: completed successfully
P00 INFO: expire command begin ...
P00 INFO: repo2: expire full backup set 20210419-142414F, 20210419-142414F_20210419-
→142556I
P00 INFO: repo2: remove expired backup 20210419-142414F_20210419-142556I
P00 INFO: repo2: remove expired backup 20210419-142414F
P00 INFO: expire command end: completed successfully

$ psql -d postgres -c "select pg_create_restore_point(RP1);"
pg_create_restore_point
- - - - -
0/190000C8
(1 row)

$ psql -d postgres -Atc "select current_timestamp,current_setting(datestyle),txid_
→current();"
19-APR-21 14:38:55.313716 +00:00|Redwood, SHOW_TIME|1183

```

(次のページに続く)

(前のページからの続き)

```
$ psql -d test -c "INSERT INTO t1 VALUES (3); SELECT pg_switch_wal();"
pg_switch_wal
-----
0/19000238
(1 row)

$ pgbackrest info --stanza=demo
stanza: demo
  status: ok
  cipher: none

db (current)
  wal archive min/max (13): 0000000100000000000000016/0000000100000000000000019

  full backup: 20210419-143643F
    timestamp start/stop: 2021-04-19 14:36:43 / 2021-04-19 14:37:01
    wal start/stop: 0000000100000000000000016 / 0000000100000000000000016
    database size: 63.0MB, database backup size: 63.0MB
    repo1: backup set size: 10MB, backup size: 10MB

  full backup: 20210419-143749F
    timestamp start/stop: 2021-04-19 14:37:49 / 2021-04-19 14:38:03
    wal start/stop: 0000000100000000000000018 / 0000000100000000000000018
    database size: 63MB, database backup size: 63MB
    repo2: backup set size: 10MB, backup size: 10MB
```

上記の例でわかるように、backup コマンドは、バックアップを作成したときにリポジトリで **expire** コマンドを自動的にトリガーしました。このコマンドは1つのリポジトリでのみ動作するため、そのリポジトリに対して backup または expire コマンドが実行されるまで、他のリポジトリの有効期限はありません。

デフォルトの保持ポリシーは多数のバックアップに基づいているため、新しいバックアップのみがバックアップとアーカイブに影響を与えることが予想されます。

7.10 復元コマンド

restore コマンドは、デフォルトで、バックアップが存在する最初のリポジトリから最新のバックアップを選択します。リポジトリがチェックされる順序は、pgbackrest.conf で構成されたりポジトリの順序によって指示されます（たとえば、repo1 は repo2 の前にチェックされます）。特定のリポジトリから復元するには、--repo オプションを使用できます。

--type=time を指定し、--target でターゲットタイムを指定することで PITR を行うことができます。--set オプションでバックアップが指定されていない場合、構成されたりポジトリで、要求された時間が含まれているかどうかチェックされます。バックアップが見つからない場合、バックアップを含む最初のリポジトリからの最新のバックアップが使用されます。

```

$ mkdir /tmp/restored_data
$ pgbackrest restore --stanza=demo --target="RP1" --type=name --no-delta --pg1-path=/
→tmp/restored_data
P00  INFO: repo1: restore backup set 20210419-143643F

$ rm -rf /tmp/restored_data/*
$ pgbackrest restore --stanza=demo --target="2021-04-19 14:38:55.313716+00:00" --
→type=time --no-delta --pg1-path=/tmp/restored_data
HINT: time format must be YYYY-MM-DD HH:MM:SS with optional msec and optional
→timezone (+/- HH or HHMM or HH:MM)
      - if timezone is omitted, local time is assumed (for UTC use +00)
P00  INFO: repo1: restore backup set 20210419-143643F

$ rm -rf /tmp/restored_data/*
$ pgbackrest restore --stanza=demo --target="RP1" --type=name --no-delta --pg1-path=/
→tmp/restored_data --repo=2
P00  INFO: repo2: restore backup set 20210419-143749F

```

repo2 のバックアップの方が新しい場合でも、最初に見つかった一致が保持されます。**repo1** で新しいバックアップを作成して、pgBackRest が **repo2** のバックアップを自動選択するかどうかを確認しましょう。

```

$ pgbackrest backup --stanza=demo --type=full --repo=1
P00  INFO: backup command begin ...
P00  INFO: execute non-exclusive pg_start_backup(): backup begins after the
→requested immediate checkpoint completes
P00  INFO: backup start archive = 0000000100000000000000001B, lsn = 0/1B000028
P00  INFO: full backup size = 63MB
P00  INFO: execute non-exclusive pg_stop_backup() and wait for all WAL segments to
→archive
P00  INFO: backup stop archive = 0000000100000000000000001B, lsn = 0/1B000138
P00  INFO: check archive for segment(s)
→0000000100000000000000001B:0000000100000000000000001B
P00  INFO: new backup label = 20210419-144656F
P00  INFO: backup command end: completed successfully
P00  INFO: expire command begin ...
P00  INFO: repo1: expire full backup 20210419-143643F
P00  INFO: repo1: remove expired backup 20210419-143643F
P00  INFO: expire command end: completed successfully

$ pgbackrest info --stanza=demo
stanza: demo
    status: ok
    cipher: none

    db (current)
        wal archive min/max (13): 00000001000000000000000018/0000000100000000000000001B

```

(次のページに続く)

(前のページからの続き)

```

full backup: 20210419-143749F
    timestamp start/stop: 2021-04-19 14:37:49 / 2021-04-19 14:38:03
    wal start/stop: 00000000100000000000000018 / 00000000100000000000000018
    database size: 63MB, database backup size: 63MB
    repo2: backup set size: 10MB, backup size: 10MB

full backup: 20210419-144656F
    timestamp start/stop: 2021-04-19 14:46:56 / 2021-04-19 14:47:10
    wal start/stop: 0000000010000000000000001B / 0000000010000000000000001B
    database size: 63MB, database backup size: 63MB
    repo1: backup set size: 10MB, backup size: 10MB

$ rm -rf /tmp/restored_data/*
$ pgbackrest restore --stanza=demo --target="2021-04-19 14:38:55.313716+00:00" --
→type=time --no-delta --pg1-path=/tmp/restored_data
P00  INFO: repo2: restore backup set 20210419-143749F

```

リカバリープロセスはターゲットによっては複雑で注意が必要な場合があるため、info コマンドは、復元するリポジトリを決定するのに役立ちます。

7.11 アーカイブ取得コマンド

複数のリポジトリが設定されている場合、WAL はリポジトリから優先順位に従ってフェッチされます (例、repo1、repo2 など)。一般に、より高速なストレージの優先度が高い方が良いでしょう。このコマンドは、--repo オプションで指定することにより、単一のリポジトリを操作できます。

```

root# systemctl stop edb-as-13.service
$ mv /var/lib/edb/as13/data /var/lib/edb/as13/data.orig
$ mkdir -m 700 /var/lib/edb/as13/data

$ pgbackrest restore --stanza=demo --target="2021-04-19 14:38:55.313716+00:00" --
→type=time --no-delta --target-action=promote
P00  INFO: restore command begin ...
P00  INFO: write updated /var/lib/edb/as13/data/postgresql.auto.conf
P00  INFO: restore global/pg_control (performed last to ensure aborted restores
→cannot be started)
P00  INFO: restore command end: completed successfully

$ cat /var/lib/edb/as13/data/postgresql.auto.conf|grep restore_command
restore_command = pgbackrest --stanza=demo archive-get %f "%p"

root# systemctl start edb-as-13.service
$ cat /var/lib/edb/as13/data/log/edb-2021-04-19_145147.log

```

(次のページに続く)

(前のページからの続き)

```
...
LOG: starting point-in-time recovery to 2021-04-19 14:38:55.313716+00
P00 INFO: found 00000001000000000000000018 in the repo2: 13-1 archive
P00 INFO: found 00000001000000000000000019 in the repo2: 13-1 archive
LOG: recovery stopping before commit of transaction 1183, time 2021-04-19 14:38:55.
    ↪314118+00
LOG: redo done at 0/19000100
P00 INFO: unable to find 00000002.history in the archive
LOG: selected new timeline ID: 2
LOG: archive recovery complete
P00 INFO: pushed WAL file 00000002.history to the archive

$ ls /var/lib/edb/as13/backups/repo*/archive/demo/13-1/00000002.history
/var/lib/edb/as13/backups/repo1/archive/demo/13-1/00000002.history
/var/lib/edb/as13/backups/repo2/archive/demo/13-1/00000002.history
```

PostgreSQL は、回復に必要な WAL を **repo2** で検出し、新しいタイムラインを選択して、履歴ファイルをリポジトリにプッシュしました。

新しいタイムラインが作成されるたびに、PostgreSQL は、いつどのタイムラインから分岐したかを示すタイムライン履歴ファイルを作成します。これらの履歴ファイルは、複数のタイムラインを含むバックアップから回復するときに、システムが適切な WAL セグメントファイルを選択できるようにするために必要です。

```
root# systemctl stop edb-as-13.service
$ rm -rf /var/lib/edb/as13/data/*
$ rm -rf /var/lib/edb/as13/backups/repo1/archive/demo/13-1/00000002.history
$ pgbackrest restore --stanza=demo --target="2021-04-19 14:38:55.313716+00:00" --
    ↪type=time --target-timeline=current --no-delta --target-action=promote
P00 INFO: restore command begin ...
P00 INFO: repo2: restore backup set 20210419-143749F
P00 INFO: write updated /var/lib/edb/as13/data/postgresql.auto.conf
P00 INFO: restore global/pg_control (performed last to ensure aborted restores
    ↪cannot be started)
P00 INFO: restore command end: completed successfully

root# systemctl start edb-as-13.service
$ cat /var/lib/edb/as13/data/log/edb-2021-04-19_145721.log
P00 INFO: found 00000002.history in the repo2: 13-1 archive
P00 INFO: unable to find 00000003.history in the archive
LOG: selected new timeline ID: 3
P00 INFO: pushed WAL file 00000003.history to the archive
```

上記のように、履歴ファイルは復元されたバックアップセットと同じリポジトリ内に保存されるため、PostgreSQL は新しい正確なタイムラインを選択できます。

履歴ファイルを **repo1** に移動してから、リトライしてみましょう。

```
root# systemctl stop edb-as-13.service
$ rm -rf /var/lib/edb/as13/data/*
$ mv /var/lib/edb/as13/backups/repo2/archive/demo/13-1/00000002.history /var/lib/edb/
→as13/backups/repo1/archive/demo/13-1/00000002.history
$ pgbackrest restore --stanza=demo --target="2021-04-19 14:38:55.313716+00:00" --
→type=time --target-timeline=current --no-delta --target-action=promote
root# systemctl start edb-as-13.service
$ cat /var/lib/edb/as13/data/log/edb-2021-04-19_150037.log
P00 INFO: found 00000002.history in the repo1: 13-1 archive
P00 INFO: found 00000003.history in the repo1: 13-1 archive
P00 INFO: unable to find 00000004.history in the archive
LOG: selected new timeline ID: 4
P00 INFO: pushed WAL file 00000004.history to the archive
```

ご覧のとおり、00000002.history と 00000003.history の両方が **repo1** で見つかったため、復元されたバックアップセットが **repo2** からのものであっても、PostgreSQL は次のタイムラインを正しく選択できました。

複数のリポジトリを使用する場合、**archive-get** コマンドは、主に他のリポジトリで不足しているファイルを見つけることができるため、1つのリポジトリでのギャップ（ディスク領域の不足など）を許容します。

第8章 ‘TLS connections’

8.1 説明

TLS サーバーは、リモートホストへのプロトコル接続に SSH を使用する代替手段です。

2.37 リリースからサポートされています。

このセクションでは、Use Case 2 で説明されているような専用リポジトリホストから pgBackRest を実行するために使用される構成を調べますが、SSH 接続ではなく TLS を使用します。

リポジトリホストが **backup-srv** と呼ばれる デモ クラスター (Rocky Linux 8 で EDBPostgresAdvancedServer バージョン 14 を使用) をセットアップし、_Streaming Replication_ の 3 つの Postgres ノード: **pg1-srv**、**pg2-srv**、**pg3-srv**。

8.2 リポジトリホストで専用ユーザーを作成する

pgbackrest ユーザーは、バックアップとアーカイブのリポジトリを所有するために作成されます。すべてのユーザーがリポジトリを所有できますが、混乱を避けるために postgres または enterprisedb を使用しないことをお勧めします。

backup-srv にユーザーとリポジトリの場所を作成します。

```
$ sudo groupadd pgbackrest
$ sudo adduser -g pgbackrest -n pgbackrest
$ sudo chown pgbackrest: /var/log/pgbackrest
```

リポジトリは、バックアップサーバーから到達可能である必要があります。

first use case で説明されているさまざまなサポートされているタイプにあります。

たとえば、バックアップとアーカイブをローカルに保存する **/backup_space** を作成しましょう。

```
$ sudo mkdir /backup_space
$ sudo chown pgbackrest: /backup_space
```

8.3 証明書の生成

TLS サーバーの実装は証明書に依存しています。

この例では、* 独自のルート認証局 (CA) を作成します。* Postgres ノードとバックアップサーバーのキーと証明書を生成します。

```
$ mkdir certs && cd certs
$ openssl req -new -x509 -days 365 -nodes -out ca.crt -keyout ca.key -subj "/CN=root-
→ca"
$ openssl req -new -nodes -out backup-srv.csr -keyout backup-srv.key -subj "/"
→CN=backup-srv"
$ openssl req -new -nodes -out pg1-srv.csr -keyout pg1-srv.key -subj "/CN=pg1-srv"
$ openssl req -new -nodes -out pg2-srv.csr -keyout pg2-srv.key -subj "/CN=pg2-srv"
$ openssl req -new -nodes -out pg3-srv.csr -keyout pg3-srv.key -subj "/CN=pg3-srv"
$ openssl x509 -req -in backup-srv.csr -days 365 -CA ca.crt -CAkey ca.key -
→CAcreateserial -out backup-srv.crt
$ openssl x509 -req -in pg1-srv.csr -days 365 -CA ca.crt -CAkey ca.key -
→CAcreateserial -out pg1-srv.crt
$ openssl x509 -req -in pg2-srv.csr -days 365 -CA ca.crt -CAkey ca.key -
→CAcreateserial -out pg2-srv.crt
$ openssl x509 -req -in pg3-srv.csr -days 365 -CA ca.crt -CAkey ca.key -
→CAcreateserial -out pg3-srv.crt
$ rm *.csr
```

次に、各サーバーに展開する必要があります (ca.crt + server_name.crt + server_name.key)。

これは、ホスト間の共有フォルダーを使用した例です。

```
# on backup-srv

$ sudo -u pgbackrest mkdir ~pgbackrest/certs
$ sudo cp `hostname`. * ~pgbackrest/certs
$ sudo cp ca.crt ~pgbackrest/certs
$ sudo chown -R pgbackrest: ~pgbackrest/certs
$ sudo chmod -R og-rwx ~pgbackrest/certs

# on pg1-srv, pg2-srv and pg3-srv

$ sudo -u enterprisedb mkdir ~enterprisedb/certs
$ sudo cp `hostname`. * ~enterprisedb/certs
$ sudo cp ca.crt ~enterprisedb/certs
$ sudo chown -R enterprisedb: ~enterprisedb/certs
$ sudo chmod -R og-rwx ~enterprisedb/certs
```

8.4 構成

demo というスタンザの構成を準備します。

backup-srv pgBackRest 構成ファイルを更新します。

```
# /etc/pgbackrest.conf

[global]
# repo details

repo1-path=/backup_space
repo1-retention-full=2

# general options

process-max=2
log-level-console=info
log-level-file=debug
start-fast=y
delta=y

# tls server options

tls-server-address=*
tls-server-cert-file=/home/pgbackrest/certs/backup-srv.crt
tls-server-key-file=/home/pgbackrest/certs/backup-srv.key
tls-server-ca-file=/home/pgbackrest/certs/ca.crt
tls-server-auth=pg1-srv=demo
tls-server-auth=pg2-srv=demo
tls-server-auth=pg3-srv=demo

[demo]
pg1-host=pg1-srv
pg1-host-user=enterprisedb
pg1-path=/var/lib/edb/as14/data
pg1-host-type=tls
pg1-host-cert-file=certs/backup-srv.crt
pg1-host-key-file=certs/backup-srv.key
pg1-host-ca-file=certs/ca.crt
```

pg1-srv、**pg2-srv**、および **pg3-srv** pgBackRest 構成ファイルを更新します。

```
# /etc/pgbackrest.conf

[global]
repo1-host=backup-srv
```

(次のページに続く)

(前のページからの続き)

```

repo1-host-user=pgbackrest
repo1-host-type=tls
repo1-host-cert-file=/var/lib/edb/certs/pg1-srv.crt
repo1-host-key-file=/var/lib/edb/certs/pg1-srv.key
repo1-host-ca-file=/var/lib/edb/certs/ca.crt

# general options

process-max=2
log-level-console=info
log-level-file=debug

# tls server options

tls-server-address=*
tls-server-cert-file=/var/lib/edb/certs/pg1-srv.crt
tls-server-key-file=/var/lib/edb/certs/pg1-srv.key
tls-server-ca-file=/var/lib/edb/certs/ca.crt
tls-server-auth=backup-srv=demo

[demo]
pg1-path=/var/lib/edb/as14/data
pg1-user=enterprisedb
pg1-port=5444

```

(注：明らかに、 **pg2-srv** の `_pg2-srv_` と **pg3-srv** の `_pg3-srv_` を調整します。)

クライアントは、証明書を検証し、証明書の CN (*Common Name*) を `tls-server-auth` で構成されたサーバー上のリストと照合することにより、サーバーで承認されます。`tls-server-auth` オプションを繰り返すことで、必要な数のスタンザに対して、または `tls-server-auth=*` を指定することにより、すべてのスタンザに対して CN を承認できます。

アイデアは、各ホストで TLS サーバーを実行して、他のホストからの要求を処理することです。例では、リポジトリホストで実行されている `backup` コマンドは、Postgres ノードで実行されている TLS サーバーのクライアントとして機能します。Postgres ノードで実行されている `archive-push` コマンドは、リポジトリホストで実行されている TLS サーバーのクライアントとして機能します。

EDB Postgres Advanced Server コマンドを使用して TLS サーバーを起動でき、SIGINT シグナル (Control+C) で終了するまで実行されます。

PGDG パッケージでまだ行われていない場合は、**backup-srv** サーバーでサービスファイルを作成します。

```

# /etc/systemd/system/pgbackrest.service

[Unit]
Description=pgBackRest Server
After=network.target

```

(次のページに続く)

(前のページからの続き)

```

StartLimitIntervalSec=0

[Service]
Type=simple
User=pgbackrest
Restart=always
RestartSec=1
ExecStart=/usr/bin/pgbackrest server
ExecStartPost=/bin/sleep 3
ExecStartPost=/bin/bash -c "[ ! -z $MAINPID ]"
ExecReload=/bin/kill -HUP $MAINPID

[Install]
WantedBy=multi-user.target

```

サーバーを起動して、実行されていることを確認します。

```

$ sudo systemctl daemon-reload
$ sudo systemctl enable pgbackrest
$ sudo systemctl start pgbackrest
$ pgbackrest server-ping

```

server-ping コマンドは、認証が試行されないため、生存確認としてのみ機能します。

tls-* の構成を変更した場合は、プロセスを終了/再起動することを忘れないでください。

次に、この構成を使用して、**pg1-srv**、**pg2-srv**、および **pg3-srv** で同じ手順（サービスの構成とスタート）を適用します。

```

# /etc/systemd/system/pgbackrest.service

[Unit]
Description=pgBackRest Server
After=network.target
StartLimitIntervalSec=0

[Service]
Type=simple
User=enterisedb
Restart=always
RestartSec=1
ExecStart=/usr/bin/pgbackrest server
ExecStartPost=/bin/sleep 3
ExecStartPost=/bin/bash -c "[ ! -z $MAINPID ]"
ExecReload=/bin/kill -HUP $MAINPID

```

(次のページに続く)

[Install]`WantedBy=multi-user.target`

次に、`postgresql.conf` ファイル (**pg1-srv**) で Postgres アーカイブを構成しましょう。

```
listen_addresses = *
archive_mode = on
archive_command = pgbackrest --stanza=demo archive-push %p
```

これらの変更を加えた後、バックアップを実行する前に、Postgres インスタンスを再起動する必要があります。

最後にスタンザを作成し、**backup-srv** で構成を確認しましょう。

```
$ sudo -iu pgbackrest pgbackrest --stanza=demo stanza-create
...
P00 INFO: stanza-create command end: completed successfully

$ sudo -iu pgbackrest pgbackrest --stanza=demo check
...
P00 INFO: check command end: completed successfully
```

8.5 バックアップを実行する

すべてが正常に動作している場合、最初のバックアップを作成できます。

```
$ sudo -iu pgbackrest pgbackrest --stanza=demo --type=full backup
P00 INFO: backup command begin 2.38: ..
P00 INFO: execute non-exclusive pg_start_backup(): backup begins after the requested
        immediate checkpoint.
→ completes
P00 INFO: backup start archive = 00000001000000000000000004, lsn = 0/4000060
P00 INFO: check archive for prior segment 00000001000000000000000003
P00 INFO: execute non-exclusive pg_stop_backup() and wait for all WAL segments to
→ archive
P00 INFO: backup stop archive = 00000001000000000000000004, lsn = 0/4000138
P00 INFO: check archive for segment(s)
→ 00000001000000000000000004:00000001000000000000000004
P00 INFO: new backup label = 20220308-143110F
P00 INFO: full backup size = 56.5MB, file total = 1691
P00 INFO: backup command end: completed successfully
```

`info` コマンドは、pgBackRest が正しく構成されている任意のサーバーから実行できます。

```
# From the Postgres node

$ sudo -iu enterprisedb pgbackrest --stanza=demo info
stanza: demo
  status: ok
  cipher: none

  db (current)
    wal archive min/max (14): 0000000100000000000000002/0000000100000000000000004

    full backup: 20220308-143110F
      timestamp start/stop: 2022-03-08 14:31:10 / 2022-03-08 14:31:20
      wal start/stop: 0000000100000000000000004 / 0000000100000000000000004
      database size: 56.5MB, database backup size: 56.5MB
      rep01: backup set size: 9.1MB, backup size: 9.1MB

# From the backup server

$ sudo -iu pgbackrest pgbackrest --stanza=demo info
stanza: demo
  status: ok
  cipher: none

  db (current)
    wal archive min/max (14): 0000000100000000000000002/0000000100000000000000004

    full backup: 20220308-143110F
      timestamp start/stop: 2022-03-08 14:31:10 / 2022-03-08 14:31:20
      wal start/stop: 0000000100000000000000004 / 0000000100000000000000004
      database size: 56.5MB, database backup size: 56.5MB
      rep01: backup set size: 9.1MB, backup size: 9.1MB
```

8.6 ストリーミングレプリケーション用にサーバーを準備する

pg1-srv で、レプリケーション用の特定のユーザーを作成します。

```
$ sudo -iu enterprisedb psql -d postgres
postgres=# CREATE ROLE replic_user WITH LOGIN REPLICATION PASSWORD mypwd;
```

pg_hba.conf を構成します。

```
host    replication    replic_user    pg2-srv    scram-sha-256
host    replication    replic_user    pg3-srv    scram-sha-256
```

構成をリロードします。

```
$ sudo systemctl reload edb-as-14.service
```

pg2-srv および **pg3-srv** で `~enterprisedb/.pgpass` を構成します。

```
$ sudo -iu enterprisedb
$ echo "*:~:replication:replic_user:mypwd" >> ~enterprisedb/.pgpass
$ chown enterprisedb: ~enterprisedb/.pgpass
$ chmod 0600 ~enterprisedb/.pgpass
```

8.7 スタンバイサーバーをセットアップする

pg2-srv と **pg3-srv** で `/etc/pgbackrest.conf` を調整して、
`recovery-option` を追加します。アイデアは、`restore` コマンドで `_Streaming`

`Replication_` 接続文字列を自動的に構成することです。

```
recovery-option=primary_conninfo=host=pg1-srv user=replic_user
```

次に、`info` コマンドを実行して、構成が正しいことを確認します。上記と同じ出力が出力されます。

pg2-srv および **pg3-srv** で **pg1-srv** サーバーから取得したバックアップを復元します。

```
$ sudo -iu enterprisedb pgbackrest --stanza=demo --type=standby restore
P00 INFO: restore command begin 2.38: ...
P00 INFO: repo1: restore backup set 20220308-143110F, recovery will start at ...
P00 INFO: write updated /var/lib/edb/as14/data/postgresql.auto.conf
P00 INFO: restore global/pg_control
(performed last to ensure aborted restores cannot be started)
P00 INFO: restore size = 56.5MB, file total = 1691
P00 INFO: restore command end: completed successfully
```

復元すると、`postgresql.auto.conf` ファイルに情報が追加されます。

```
# Recovery settings generated by pgBackRest restore on ...

primary_conninfo = host=pg1-srv user=replic_user
restore_command = pgbackrest --stanza=demo archive-get %f "%p"
```

`--type=standby` オプションは、Postgres がスタンバイモードで起動するために必要な `standby.signal` を作成します。あとは Postgres インスタンスを起動するだけです。

```
$ sudo systemctl enable edb-as-14.service
$ sudo systemctl start edb-as-14.service
```

レプリケーション設定が正しい場合、**pg1-srv** サーバーでこれらのプロセスが表示されます。

```
$ sudo -u enterprisedb ps -o pid,cmd fx
  PID CMD
30594 /usr/edb/as14/bin/edb-postmaster -D /var/lib/edb/as14/data
...
31436 \_ postgres: walsender replic_user ... streaming 0/5001FB0
31439 \_ postgres: walsender replic_user ... streaming 0/5001FB0
30222 /usr/bin/pgbackrest server
```

これで、セーフティネットとして_ストリーミングレプリケーション_とアーカイブリカバリを使用する 3 ノードクラスターができました。

8.8 スタンバイサーバーからバックアップを取る

backup-srv の pgBackRest 構成ファイルの [demo] スタンザセクションに、次の設定を追加します。

```
pg2-host=pg2-srv
pg2-host-user=enterprisedb
pg2-path=/var/lib/edb/as14/data
pg2-host-type=tls
pg2-host-cert-file=certs/backup-srv.crt
pg2-host-key-file=certs/backup-srv.key
pg2-host-ca-file=certs/ca.crt
pg3-host=pg3-srv
pg3-host-user=enterprisedb
pg3-path=/var/lib/edb/as14/data
pg3-host-type=tls
pg3-host-cert-file=certs/backup-srv.crt
pg3-host-key-file=certs/backup-srv.key
pg3-host-ca-file=certs/ca.crt
backup-standby=y
```

次に、構成で見つかった最初のスタンバイサーバーからデータを取得してバックアップを実行します。

```
$ sudo -iu pgbackrest pgbackrest --stanza=demo --type=full backup
P00 INFO: backup command begin 2.38: ...
P00 INFO: execute non-exclusive pg_start_backup(): backup begins after the requested
      immediate checkpoint.
→ completes
P00 INFO: backup start archive = 00000001000000000000000006, lsn = 0/6000028
P00 INFO: wait for replay on the standby to reach 0/6000028
P00 INFO: replay on the standby reached 0/6000028
P00 INFO: check archive for prior segment 00000001000000000000000005
P00 INFO: execute non-exclusive pg_stop_backup() and wait for all WAL segments to
→ archive
P00 INFO: backup stop archive = 00000001000000000000000006, lsn = 0/6000138
```

(次のページに続く)

(前のページからの続き)

```
P00    INFO: check archive for segment(s) ↳  
↳00000000100000000000000006:0000000010000000000000000006  
P00    INFO: new backup label = 20220308-145104F  
P00    INFO: full backup size = 56.5MB, file total = 1691  
P00    INFO: backup command end: completed successfully
```

8.9 結論

TLS サーバーは、リモート操作のための SSH の代替を提供します。この新機能は、パフォーマンスとユーザーエクスペリエンスを大幅に向上させることができるコンテナ化された世界に完全に適合します。

第9章 ‘Appendix’

9.1 概念

次の概念は、pgBackRest、PostgreSQL、およびこのユーザーガイドに関連するものとして定義されています。

9.2 バックアップ

バックアップは、ハードウェア障害からの回復、ポイントインタイムリカバリの実行、または新しいスタンバイの起動のために復元できるデータベースクラスターの一貫したコピーです。

完全バックアップ：pgBackRest はデータベースクラスターのコンテンツ全体をバックアップにコピーします。データベースクラスターの最初のバックアップは常に完全バックアップです。pgBackRest は、完全バックアップを直接復元します。完全バックアップは、一貫性のために完全バックアップの外部のファイルに依存しません。

差分バックアップ：pgBackRest は、最後の完全バックアップ以降に変更されたデータベースクラスターファイルのみをコピーします。差分バックアップを復元するために、pgBackRest は選択された差分バックアップ内のすべてのファイルと、以前の完全バックアップから変更されていない適切なファイルをコピーします。差分バックアップの利点は、完全バックアップよりも必要なディスク領域が少ないことです。差分バックアップを復元するには、差分バックアップと完全バックアップが有効である必要があります。

インクリメンタルバックアップ：pgBackRest は、最後のバックアップ（インクリメンタルバックアップ、差分バックアップ、または完全バックアップ）以降に変更されたデータベースクラスターファイルのみをコピーします。増分バックアップには、前回のバックアップ以降に変更されたファイルのみが含まれるため、一般的には完全バックアップまたは差分バックアップよりもはるかに小さくなります。増分バックアップでは、増分バックアップを復元する前に他のバックアップが有効である必要があります。増分バックアップには最後のバックアップ以降に変更または追加されたファイルのみが含まれるため、増分バックアップを復元するには、以前のすべての増分バックアップ（以前の差分バックアップおよび以前の完全バックアップ）が有効である必要があります。差分バックアップが存在しない場合、増分バックアップを復元するには、以前のすべての増分バックアップ（以前の完全バックアップと完全バックアップ自分自身）が有効である必要があります。

9.3 復元する

復元とは、ライブデータベースクラスターとして起動されるシステムにバックアップをコピーする行為です。復元が正しく動作するには、バックアップファイルと 1 つ以上の WAL セグメントが必要です。

9.4 ログ先行書き込み (WAL)

WAL ロギングは、コミットされた変更が失われないようにするために PostgreSQL が使用するメカニズムです。トランザクションは WAL にシーケンシャルに書き込まれ、これらの書き込みがディスクにフラッシュされると、トランザクションはコミットされたと見なされます。その後、バックグラウンドのライタプロセスがメインデータベースクラスターファイル (ヒープとも呼ばれます) に変更を書き込みます。クラッシュが発生した場合、データベースの一貫性を保つために WAL が再生されます。

WAL は概念的には無限ですが、実際にはセグメントと呼ばれる個々の 16MB のファイルに分割されます。WAL セグメントは命名規則 0000000100000A1E000000FE に従います。最初の 8 桁の 16 進数はタイムラインを表し、次の 16 桁はログシーケンス番号 (LSN) です。

9.5 暗号化

暗号化とは、適切なパスワード (パスフレーズとも呼ばれます) を入力しない限り認識できない形式にデータを変換するプロセスです。リポジトリ内に保存されているデータへの不正アクセスを防ぐために、pgBackRest はユーザーが指定したパスワードで認証した後にリポジトリを暗号化します。

9.6 構成リファレンス

プロジェクトのドキュメントには、コマンドと構成キーの完全なリストがあります。

- コマンドライン操作の場合は [Command reference](#)。
- pgBackRest 構成を作成するための [構成リファレンス](#)。

pgBackRest 全体をコマンドラインパラメーターで使用できますが、複数のオプションを即座に構成する場合、複雑なインストールでは構成ファイルの方が実用的です。構成ファイルのデフォルトの場所は `/etc/pgbackrest/pgbackrest.conf` です。構成ファイルがその場所に存在しない場合、代わりに古い `/etc/pgbackrest.conf` ファイルがロードされます (存在する場合)。

コマンドラインオプションでシークレットを指定しないように、構成ファイルを使用することもお勧めします。

9.7 グローバル

構成ファイルの global セクションを使用して、デフォルト値を設定します。

グローバルセクションでは、特定のコマンドを呼び出すときに使用される値を定義することもできます。たとえば、次の構成設定は `archive-push` コマンドの圧縮レベルを指定します。

```
[global:archive-push]
compress-level=3
```

9.8 スタンザ

スタンザは、データベースクラスタの場所、データベースクラスタのバックアップの方法、使用可能なアーカイブオプションなどを定義するデータベースクラスタ構成です。ほとんどのデータベースサーバーには1つのデータベースクラスタのみがあるため、スタンザ; 一方、バックアップサーバーには、バックアップが必要なデータベースクラスタごとにスタンザがあります。

スタンザ名はプライマリとすべてのレプリカに使用されるため、`main` や `prod` などのローカルインスタンス名ではなく、`app` や `dw` などの実際のファンクションを説明する名前を選択することをお勧めします。

スタンザレベルの設定は、グローバル構成設定をオーバーライドします。

9.9 コマンドライン引数と環境変数

オプションをコマンドライン引数または環境変数として構成することもできます。

- コマンドラインで非ブール引数を使用するには、オプションの前に2つのダッシュを付けます。
- コマンドラインでブールオプションを `false` に設定するには、`--no-` プレフィックスを使用します。
- 環境変数を設定するには、オプション名を大文字にし、先頭に `PGBACKREST_` を付け、ダッシュをアンダースコアに置き換えます。

次の表は、`pgbackrest.conf` ファイル、コマンドライン引数、および環境変数を使用してオプションを指定する方法を示しています。

Configuration File	Command Line Argument	Environment Variable
<code>pg1-host=1.2.3.4</code>	<code>--pg1-host=1.2.3.4</code>	<code>PGBACK- REST_PG1_HOST=1.2.3.4</code>
<code>compress=y</code>	<code>--compress</code>	<code>PGBACK- REST_COMPRESS=y</code>
<code>delta=n</code>	<code>--no-delta</code>	<code>PGBACKREST_DELTA=n</code>

9.10 オプションの優先順位

すべてのオプションの優先順位（高いものから低いもの）は次のとおりです。

- コマンドライン引数
- 環境変数
- [stanza:command]
- [stanza]
- [global:command]
- [global]
- デフォルト（内部）

9.11 スタンザでの複数のデータベースホストの構成

各スタンザは複数のデータベースホストを構成できます。そのため、ホスト 2 のすべての `pg1-*` 構成オプションを `pg2-*` で、ホスト 3 の場合で `pg3-*` などを構成することもできます。これは、これらのデータベースクラスターが互いにリンクされていることを前提としています（

ストリーミングレプリケーション用にサーバーを準備する など）。

第10章 ‘Frequently Asked Questions’

- 構成ファイルで1つのスタンザのみが定義されているのに、なぜ`--stanza` オプションを指定する必要がありますか？

リポジトリ固有のコマンド (`info`、`start`、`stop`) を除き、スタンザを追加してもスタンザが指定されていないために使用するスタンザがわからないために突然エラーがスローされないように、`--stanza` オプションが必要です。

- データベースクラスター全体のバックアップから特定のデータベースのみを復元することはできますか？

はい。

`restore --db-include <https://pgbackrest.org/command.html#command-restore/category-command/option-db-include>`_` オプションを使用して、特定のデータベースを復元できます。回復後、`pg_dump` コマンドを使用して選択したデータベースを抽出できます。

このオプションで明示的に含まれていないデータベースは、スペースを節約するためにスペースでゼロ化されたファイルとして復元されます。ただし、PostgreSQL はより少ないディスク領域を使用してリカバリを実行できます。これらのデータベースにはアクセスできなくなるため、`DROP DATABASE` コマンドを使用して削除できます。

- SSH プロトコルを使用せずにリモートホストにアクセスするように pgBackRest をセットアップすることはできますか？

はい。2.37 リリース以降、pgBackRest はクライアント証明書で TLS を使用して、ホスト間の通信を有効にできます。次に、各ホストで TLS サーバーを構成して起動する必要があります。

詳細については、[TLS のセットアップ] (<https://pgbackrest.org/user-guide-rhel.html#repo-host/config>) を参照してください。

- 専用のリモートバックアップサーバーにはいくつの CPU コアが必要ですか？

圧縮/解凍は Postgres ホストで実行されるため、リポジトリホストは SSH または TLS 接続を暗号化/復号化するだけで済みます。通常、4 つの Postgres コアごとに 1 つのリポジトリコアが出発点として適しています。