バックアップ関連

オンライン論理バックアップ

①pg_dumpall を用いても、設定ファイル等は自動的に取得できない。

②その代わりに物理バックアップ（オンライン、オフライン共）で別途バックアップが必要であるテーブル空間等は、pg_dump や pg_dumpall では対応している。

オンライン物理バックアップ

●pg_basebackup の方が pg_start_backup/pg_stop_backup よりメリットが大きい気がするが、前者によるバックアップ取得時間は DB サイズに影響を受けるため、スナップショット機能を使用できる後者の使用を検討すべき。

●pg_basebackup はスーパーユーザ及びレプリケーション権限者が実行、pg_start_backup/pg_stop_backup はスーパーユーザのみが実行できる。

●pg_basebackup はオプションでテーブル空間もバックアップできる。**（←その他のオプションも含めて調べる。）**

●pg_start_backup/pg_stop_backup は、排他又は非排他モードを選択できるが、排他モードは非推奨。排他モードの場合、pg_start_backup の処理冒頭で排他モード開始、pg_stop_backup 処理冒頭で終了となる。

オフライン物理バックアップ

●ポスグレを停止させる必要があるため、サービスが停止する。また、リストアの際には DB の状態がバックアップの時点に戻る。詳細は省略するが、簡単な手順は以下のとおり。

①pg_ctl stop でポスグレを停止して、pg_ctl status で停止を確認

②バックアップは、スナップショット、cp、tar、rcync 等の Linux 等のコマンドやツールを使用する。

③pg_ctl start でポスグレ起動

バックアップファイルのチェックサム

●DB クラスタにおいて、チェックサムを有効（initdb 時での設定等）にすることで、以下の機能が働く。

①pg_basebackup 実行時にデフォルトでチェックサムの確認を行い、データの破損を検知できる。

②バックアップマニュフェストファイルを取得し、pg_verifybackup コマンドを使用することで、取得済みのバックアップファイルのデータの破損を検知できる。

オンライン物理バックアップ（pg_start_backup/pg_stop_backup）を利用したリカバリ

●ベースバックアップの取得

◎pg_start_backup を実行すると、①：実行開始直後の WAL の LSN 取得、②：WAL ファイルの切り替え（スイッチ）、③：チェックポイントの発行とその LSN、④：前記①と③を元にスイッチ後の WAL ファイル名の特定、⑤：前記①から④を backup_label ファイルに記録する。

◎チェックポイント処理が終われば、pg_start_backup 処理が完了するので、Linux の cp コマンド等でクラスタ全体のバックアップを取得する。

◎その後、pg_stop_backup を実行すると、再び WAL スイッチが入り、pg_start_backup 時にスイッチされた WAL ファイルへの書き込みは終了しアーカイブ化される。そして、再スイッチされた WAL ファイルがアーカイブ化されるのを待って pg_start_backup 処理が完了

●リカバリ準備として、①：ベースバックアップ、②：稼働中のクラスタの WAL ファイル、③：pg_start_backup 時にスイッチされた WAL ファイルのアーカイブ以降の全てのアーカイブ（backup_label ファイルに記載された WAL ファイル名に相当）を用意する。①には②及び③の内容はデータディスクに反映されていないためである。

●再起動

◎準備完了後、次項の recovery.signal ファイルがクラスタデータディレクトリに存在する状態で再起動すれば、アーカイブ（未適用の WAL も含む）が適用されリカバリが完了する。その際、同ディレクトリにある backup_label ファイルを参照することで、適用開始位置がわかる。

◎再起動の際には、上記 2 つのファイル以外に pg_control ファイル（global ディレクトリ内）と postgresql.conf ファイル（クラスタ配下）も読み込まれる。pg_control ファイルは、リカバリ等を問わず、ポスグレ起動時に最初に読み込まれる。バイナリ形式なので pg_controldata コマンドによって text 形式で表示される。

◎backup_label が読み込めない等の際には、pg_control の情報をもとにリカバリされる。いずれにせよ、リカバリ完了後は役目を終えた backup_label の内容は pg_control に反映された後、削除される。

recovery.signal ファイル（アーカイブ適用ファイル）

●archive_command パラメータに記載するシェルコマンドによるアーカイブの保存先は、restore_command パラメータに記載する保存元と同じである。

●アーカイブ（未適用の WAL も含む）を適用するためには、recovery.signal ファイル（名前だけの空ファイル）をクラスタデータディレクトリ（クラスタ本体のディレクトリ）に作成する。

●ストリーミングレプリケーションにおいて、同一サーバであれば、プライマリで archive_command、スタンバイで restore_command を設定する。なお、ロジカルレプリケーションではアーカイブは使用しない。

●recovery.signal ファイルと standby.signal ファイルの混同注意。

①ストリーミングレプリケーションにおいて、スタンバイ側の recovery.conf ファイル内で standby_mode=on としていたが、現在はクラスタデータディレクトリに standby.signal ファイルが存在すれば OK。

②今は、recovery.conf は postgresql.conf に統合されて存在しないので、もし、recovery.conf が存在するとポスグレは起動しない。

参考：tar コマンドについて

●長らく、tar から離れていたため、上記のオフライン物理バックアップを機にまとめた事項を以下に示す。

①アーカイブ及び展開処理：**「tar cvf test.tar test*（オプションに-は不要）」**の解釈は、カレントディレクトリにある f（test が付くファイル）を c（test.tar という名のアーカイブファイル）にする。そして v（処理状況を表示、ここでは test*の一覧）を行う。以降 v は省略する。

「tar xf test.tar test*」の解釈は、 f（test.tar というファイル）を x（展開）する。

②圧縮及び解凍処理：**「gzip test.tar」**によって test.tar.gz が生成し、**「gunzip test.tar.gz」**で解凍

③アーカイブと圧縮をまとめて処理：**「tar czf test.tar.gz test*」**の解釈は、カレントディレクトリにある f（test が付くファイル）を c z（test.tar.gz という名のアーカイブ後に圧縮したファイル）にする。

④解凍及び展開をまとめて処理：**「tar zxf test.tar.gz」**の解釈は、 f（test.tar.gz がというファイル）を z x（解凍後に展開）にする。圧縮も解凍も z である。圧縮を展開するためには解凍することが当然であるため、解凍時の z は省略できる。