



Red Hat Enterprise Linux 9

データベースサーバーの設定および使用

Red Hat Enterprise Linux 9 でデータベースサーバーを設定および使用するためのガイド

Red Hat Enterprise Linux 9 データベースサーバーの設定および使用

Red Hat Enterprise Linux 9 でデータベースサーバーを設定および使用するためのガイド

Enter your first name here. Enter your surname here.

Enter your organisation's name here. Enter your organisational division here.

Enter your email address here.

法律上の通知

Copyright © 2022 | You need to change the HOLDER entity in the en-US/Configuring_and_using_database_servers.ent file |.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

本書は、Red Hat Enterprise Linux 9 で MariaDB および PostgreSQL を使用してデータベースサーバーを設定する基本について説明します。

目次

多様性を受け入れるオープンソースの強化	4
RED HAT ドキュメントへのフィードバックの提供	5
第1章 データベースサーバーの概要	6
第2章 MARIADB の使用	7
2.1. MARIADB の概要	7
2.2. MARIADB のインストール	7
2.3. MARIADB の設定	8
2.4. MARIADB サーバーでの TLS 暗号化の設定	8
2.4.1. MariaDB サーバーに CA 証明書、サーバー証明書、および秘密鍵を配置する	9
2.4.2. MariaDB サーバーでの TLS の設定	10
2.4.3. 特定のユーザーアカウントに TLS で暗号化された接続を要求する	11
2.5. MARIADB クライアントでの TLS 暗号化のグローバルな有効化	12
2.5.1. デフォルトで TLS 暗号化を使用するように MariaDB クライアントを設定する	12
2.6. MARIADB データのバックアップ	13
2.6.1. mariadb-dump を使用した論理バックアップの実行	14
2.6.2. Mariabackup ユーティリティーを使用した物理的なオンラインバックアップの実行	15
2.6.3. Mariabackup ユーティリティーを使用したデータの復元	17
2.6.4. ファイルシステムのバックアップの実行	18
2.6.5. バックアップソリューションとしてレプリケーションを使用	19
2.7. MARIADB 10.5 への移行	19
2.7.1. MariaDB 10.3 と MariaDB 10.5 の主な相違点	19
2.7.2. RHEL 8 バージョンの MariaDB 10.3 から、RHEL 9 バージョンの MariaDB 10.5 への移行	21
2.8. GALERA で MARIADB を複製する	22
2.8.1. MariaDB Galera クラスターの概要	22
2.8.2. MariaDB Galera クラスターを構築するためのコンポーネント	23
2.8.3. MariaDB Galera クラスターのデプロイメント	24
2.8.4. 新規ノードの MariaDB Galera クラスターへの追加	25
2.8.5. MariaDB Galera クラスターの再起動	26
第3章 MYSQL の使用	27
3.1. MYSQL を使い始める	27
3.2. MYSQL のインストール	27
3.3. MYSQL の設定	28
3.4. MYSQL データのバックアップ	28
3.4.1. mysqldump を使用した論理バックアップの実行	29
3.4.2. ファイルシステムのバックアップの実行	30
3.4.3. バックアップソリューションとしてレプリケーションを使用	31
3.5. MYSQL8.0 の RHEL9 バージョンへの移行	31
3.6. MYSQL の複製	32
3.6.1. MySQL ソースサーバーの設定	33
3.6.2. MySQL レプリカサーバーの設定	34
3.6.3. MySQL ソースサーバーでのレプリケーションユーザーの作成	35
3.6.4. レプリカサーバーをソースサーバーに接続する	35
3.6.5. 検証手順	36
3.6.6. 関連情報	37
第4章 POSTGRESQL の使用	38
4.1. POSTGRESQL の概要	38
4.2. POSTGRESQL のインストール	38

4.3. POSTGRESQL ユーザー	39
4.4. POSTGRESQL の設定	39
4.5. POSTGRESQL データのバックアップ	40
4.5.1. SQL ダンプを使用した PostgreSQL データのバックアップ	41
4.5.1.1. SQL ダンプの長所と短所	41
4.5.1.2. pg_dump を使用した SQL ダンプの実行	41
4.5.1.3. pg_dumpall を使用した SQL ダンプの実行	42
4.5.1.4. pg_dump を使用したダンプされたデータベースの復元	42
4.5.1.5. pg_dumpall を使用したダンプされたデータベースの復元	43
4.5.1.6. 別のサーバーでのデータベースの SQL ダンプの実行	43
4.5.1.7. 復元中の SQL エラーの処理	44
4.5.2. ファイルシステムレベルのバックアップを使用した PostgreSQL データのバックアップ	44
4.5.2.1. ファイルシステムレベルのバックアップの長所と短所	45
4.5.2.2. ファイルシステムレベルのバックアップの実行	45
4.5.3. 継続的にアーカイブして PostgreSQL データのバックアップを作成	45
4.5.3.1. 継続的なアーカイブの概要	45
4.5.3.2. 継続的なアーカイブの長所と短所	46
4.5.3.3. WAL アーカイブの設定	47
4.5.3.4. ベースバックアップの作成	48
4.5.3.5. 継続的なアーカイブバックアップを使用したデータベースの復元	50
4.6. POSTGRESQL の RHEL9 バージョンへの移行	51
4.6.1. pg_upgrade ユーティリティを使用した高速アップグレード	51
4.6.2. ダンプおよび復元のアップグレード	53

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、[弊社の CTO、Chris Wright のメッセージ](#) を参照してください。

RED HAT ドキュメントへのフィードバックの提供

ご意見ご要望をお聞かせください。ドキュメントの改善点はございますか。

- 特定の部分についての簡単なコメントをお寄せいただく場合は、以下をご確認ください。
 1. ドキュメントの表示が **Multi-page HTML** 形式になっていて、ドキュメントの右上隅に **Feedback** ボタンがあることを確認してください。
 2. マウスカーソルで、コメントを追加する部分を強調表示します。
 3. そのテキストの下に表示される **Add Feedback** ポップアップをクリックします。
 4. 表示される手順に従ってください。
- Bugzilla を介してフィードバックを送信するには、新しいチケットを作成します。
 1. [Bugzilla](#) の Web サイトに移動します。
 2. Component で **Documentation** を選択します。
 3. **Description** フィールドに、ドキュメントの改善に関するご意見を記入してください。ドキュメントの該当部分へのリンクも記入してください。
 4. **Submit Bug** をクリックします。

第1章 データベースサーバーの概要

データベースサーバーは、データベース管理システム (DBMS) の機能を提供するサービスです。DBMS は、データベース管理のためのユーティリティーを提供し、エンドユーザー、アプリケーション、およびデータベースと対話します。

Red Hat Enterprise Linux 9 は、以下のデータベース管理システムを提供します。

- MariaDB 10.5
- MySQL 8.0
- PostgreSQL 13
- Redis 6

第2章 MARIADB の使用

MariaDB サーバーは、MySQL テクノロジーに基づくオープンソースの高速で堅牢なデータベースサーバーです。ここでは、RHEL システムに MariaDB をインストールして設定する方法、MariaDB データをバックアップする方法、MariaDB の以前のバージョンから移行する方法、および MariaDB Galera クラスタを使用してデータベースを複製する方法について説明します。

2.1. MARIADB の概要

MariaDB は、データを構造化情報に変換して、データにアクセスする SQL インターフェースを提供するリレーショナルデータベースです。これには、複数のストレージエンジンとプラグイン、および地理的な情報システム (GIS) および JSON (JavaScript Object Notation) 機能が含まれます。

Red Hat Enterprise Linux 9 では、以下について説明します。

- 「[MariaDB のインストール](#)」で、MariaDB サーバーをインストールする方法について。
- 「[MariaDB の設定](#)」で、MariaDB 設定を調整する方法について。
- 「[MariaDB サーバーでの TLS 暗号化の設定](#)」で、TLS 暗号化を MariaDB で設定する方法について。
- 「[MariaDB クライアントでの TLS 暗号化のグローバルな有効化](#)」で、TLS 暗号化を MariaDB クライアントでグローバルに有効にする方法について。
- 「[MariaDB データのバックアップ](#)」で、MariaDB データのバックアップ方法について。
- 「[MariaDB 10.5 への移行](#)」で、RHEL 8 バージョンの MariaDB 10.3 から RHEL 9 バージョンの MariaDB 10.5 に移行する方法について。
- 「[Galera で MariaDB を複製する](#)」で、MariaDB Galera クラスタを使用したデータベースの複製方法について。

2.2. MARIADB のインストール

RHEL 9.0 は、この Application Stream の初期バージョンとして MariaDB 10.5 を提供します。これは、RPM パッケージとして簡単にインストールできます。追加の MariaDB バージョンは、RHEL 9 の今後のマイナーリリースで、ライフサイクルが短いモジュールとして提供されます。

MariaDB をインストールするには、以下の手順を行います。

手順

1. MariaDB サーバーパッケージをインストールします。

```
# dnf install mariadb-server
```

2. **mariadb** サービスを起動します。

```
# systemctl start mariadb.service
```

3. **mariadb** サービスが、システムの起動時に起動するようにします。

```
# systemctl enable mariadb.service
```

4. **推奨:** MariaDB のインストール時にセキュリティを強化する場合は、次のコマンドを実行します。

```
$ mariadb-secure-installation
```

このコマンドは、完全にインタラクティブなスクリプトを起動して、プロセスの各ステップのプロンプトを表示します。このスクリプトを使用すると、次の方法でセキュリティを改善できます。

- root アカウントのパスワードの設定
- 匿名ユーザーの削除
- リモート root ログインの拒否 (ローカルホスト外)



注記

RPM パッケージが競合しているため、RHEL 9 では MariaDB および MySQL データベースサーバーを同時にインストールすることはできません。RHEL 9 では、コンテナ内で異なるバージョンのデータベースサーバーを使用できます。

2.3. MARIADB の設定

MariaDB サーバーをネットワーク用に設定するには、以下の手順に従います。

手順

1. `/etc/my.cnf.d/mariadb-server.cnf` ファイルの `[mysqld]` セクションを編集します。以下の設定ディレクティブを設定できます。
 - **bind-address:** サーバーがリッスンするアドレスです。設定可能なオプションは以下のとおりです。
 - ホスト名
 - IPv4 アドレス
 - IPv6 アドレス
 - **skip-networking:** サーバーが TCP/IP 接続をリッスンするかどうかを制御します。以下の値が使用できます。
 - 0 - すべてのクライアントをリッスンする
 - 1 - ローカルクライアントのみをリッスンする
 - **port:** MariaDB が TCP/IP 接続をリッスンするポート。
2. `mariadb` サービスを再起動します。

```
# systemctl restart mariadb.service
```

2.4. MARIADB サーバーでの TLS 暗号化の設定

デフォルトでは、MariaDB は暗号化されていない接続を使用します。安全な接続のために、MariaDB サーバーで TLS サポートを有効にし、暗号化された接続を確立するようにクライアントを設定します。

2.4.1. MariaDB サーバーに CA 証明書、サーバー証明書、および秘密鍵を配置する

MariaDB サーバーで TLS 暗号化を有効にする前に、認証局 (CA) 証明書、サーバー証明書、および秘密鍵を MariaDB サーバーに保存します。

前提条件

- Privacy Enhanced Mail (PEM) 形式の以下のファイルがサーバーにコピーされています。
 - サーバーの秘密鍵: **server.example.com.key.pem**
 - サーバー証明書: **server.example.com.crt.pem**
 - 認証局 (CA) 証明書: **ca.crt.pem**

秘密鍵および証明書署名要求 (CSR) の作成や、CA からの証明書要求に関する詳細は、CA のドキュメントを参照してください。

手順

1. CA およびサーバー証明書を `/etc/pki/tls/certs/` ディレクトリーに保存します。

```
# mv <path>/server.example.com.crt.pem /etc/pki/tls/certs/  
# mv <path>/ca.crt.pem /etc/pki/tls/certs/
```

2. MariaDB サーバーがファイルを読み込めるように、CA およびサーバー証明書にパーミッションを設定します。

```
# chmod 644 /etc/pki/tls/certs/server.example.com.crt.pem /etc/pki/tls/certs/ca.crt.pem
```

証明書は、セキュアな接続が確立される前は通信の一部であるため、任意のクライアントは認証なしで証明書を取得できます。そのため、CA およびサーバーの証明書ファイルに厳密なパーミッションを設定する必要はありません。

3. サーバーの秘密鍵を `/etc/pki/tls/private/` ディレクトリーに保存します。

```
# mv <path>/server.example.com.key.pem /etc/pki/tls/private/
```

4. サーバーの秘密鍵にセキュアなパーミッションを設定します。

```
# chmod 640 /etc/pki/tls/private/server.example.com.key.pem  
# chgrp mysql /etc/pki/tls/private/server.example.com.key.pem
```

承認されていないユーザーが秘密鍵にアクセスできる場合は、MariaDB サーバーへの接続は安全ではなくなります。

5. SELinux コンテキストを復元します。

```
# restorecon -Rv /etc/pki/tls/
```

2.4.2. MariaDB サーバーでの TLS の設定

セキュリティを向上させるには、MariaDB サーバーで TLS サポートを有効にします。その結果、クライアントは TLS 暗号化を使用してサーバーでデータを送信できます。

前提条件

- MariaDB サーバーをインストールしている。
- **mariadb** サービスが実行している。
- Privacy Enhanced Mail (PEM) 形式の以下のファイルがサーバー上にあり、**mysql** ユーザーが読み取りできます。
 - サーバーの秘密鍵: **/etc/pki/tls/private/server.example.com.key.pem**
 - サーバー証明書: **/etc/pki/tls/certs/server.example.com.crt.pem**
 - 認証局 (CA) 証明書 **/etc/pki/tls/certs/ca.crt.pem**
- サーバー証明書のサブジェクト識別名 (DN) またはサブジェクトの別名 (SAN) フィールドは、サーバーのホスト名と一致します。

手順

1. **/etc/my.cnf.d/mariadb-server-tls.cnf** ファイルを作成します。

- a. 以下の内容を追加して、秘密鍵、サーバー、および CA 証明書へのパスを設定します。

```
[mariadb]
ssl_key = /etc/pki/tls/private/server.example.com.key.pem
ssl_cert = /etc/pki/tls/certs/server.example.com.crt.pem
ssl_ca = /etc/pki/tls/certs/ca.crt.pem
```

- b. 証明書失効リスト (CRL) がある場合は、それを使用するように MariaDB サーバーを設定します。

```
ssl_crl = /etc/pki/tls/certs/example.crl.pem
```

- c. オプション:暗号化のない接続試行を拒否します。この機能を有効にするには、以下を追加します。

```
require_secure_transport = on
```

- d. オプション:サーバーがサポートする TLS バージョンを設定します。たとえば、TLS 1.2 および TLS 1.3 をサポートするには、以下を追加します。

```
tls_version = TLSv1.2,TLSv1.3
```

デフォルトでは、サーバーは TLS 1.1、TLS 1.2、および TLS 1.3 をサポートします。

2. **mariadb** サービスを再起動します。

```
# systemctl restart mariadb
```

検証

トラブルシューティングを簡素化するには、ローカルクライアントが TLS 暗号化を使用するように設定する前に、MariaDB サーバーで以下の手順を実行します。

1. MariaDB で TLS 暗号化が有効になっていることを確認します。

```
# mysql -u root -p -e "SHOW GLOBAL VARIABLES LIKE 'have_ssl';"
+-----+-----+
| Variable_name | Value      |
+-----+-----+
| have_ssl      | YES       |
+-----+-----+
```

`have_ssl` 変数が `yes` に設定されている場合、TLS 暗号化が有効になります。

2. MariaDB サービスが特定の TLS バージョンのみをサポートするように設定している場合は、`tls_version` 変数を表示します。

```
# mysql -u root -p -e "SHOW GLOBAL VARIABLES LIKE 'tls_version';"
+-----+-----+
| Variable_name | Value      |
+-----+-----+
| tls_version   | TLSv1.2,TLSv1.3 |
+-----+-----+
```

関連情報

- [MariaDB サーバーに CA 証明書、サーバー証明書、および秘密鍵を配置する](#)

2.4.3. 特定のユーザーアカウントに TLS で暗号化された接続を要求する

機密データにアクセスできるユーザーは、ネットワーク上で暗号化されていないデータ送信を回避するために、常に TLS で暗号化された接続を使用する必要があります。

すべての接続にセキュアなトランスポートが必要なサーバーで設定できない場合は (`require_secure_transport = on`)、TLS 暗号化を必要とするように個別のユーザーアカウントを設定します。

前提条件

- MariaDB サーバーで TLS サポートが有効になっている。
- セキュアなトランスポートを必要とするように設定するユーザーが存在する。

手順

1. 管理ユーザーとして MariaDB サーバーに接続します。

```
# mysql -u root -p -h server.example.com
```

管理ユーザーがリモートでサーバーにアクセスする権限を持たない場合は、MariaDB サーバーでコマンドを実行し、`localhost` に接続します。

2. **REQUIRE SSL** 句を使用して、ユーザーが TLS 暗号化接続を使用して接続する必要があるよう強制します。

```
MariaDB [(none)]> ALTER USER 'example'@'%' REQUIRE SSL;
```

検証

1. TLS 暗号化を使用して、**example** ユーザーとしてサーバーに接続します。

```
# mysql -u example -p -h server.example.com --ssl
...
MariaDB [(none)]>
```

エラーが表示されず、インタラクティブな MariaDB コンソールにアクセスできる場合は、TLS との接続は成功します。

2. TLS を無効にして、**example** ユーザーとして接続を試みます。

```
# mysql -u example -p -h server.example.com --skip-ssl
ERROR 1045 (28000): Access denied for user 'example'@'server.example.com' (using
password: YES)
```

このユーザーに TLS が必要だが無効になっているため、サーバーはログインの試行を拒否しました (**--skip-ssl**)。

関連情報

- [MariaDB サーバーでの TLS の設定](#)

2.5. MARIADB クライアントでの TLS 暗号化のグローバルな有効化

MariaDB サーバーが TLS 暗号化に対応している場合は、安全な接続のみを確立し、サーバー証明書を検証するようにクライアントを設定します。この手順では、サーバー上のすべてのユーザーで TLS サポートを有効にする方法を説明します。

2.5.1. デフォルトで TLS 暗号化を使用するように MariaDB クライアントを設定する

RHEL では、MariaDB クライアントが TLS 暗号化を使用するようにグローバルに設定でき、サーバー証明書の Common Name (CN) が、ユーザーが接続するホスト名と一致することを検証します。これにより、man-in-the-middle 攻撃 (中間者攻撃) を防ぎます。

前提条件

- MariaDB サーバーで TLS サポートが有効になっている。
- サーバー証明書を発行した認証局(CA)が RHEL で信頼されていない場合は、CA 証明書がクライアントにコピーされています。

手順

1. RHEL が、サーバー証明書を発行した CA を信頼しない場合は、以下を行います。
 - a. CA 証明書を `/etc/pki/ca-trust/source/anchors/` ディレクトリーにコピーします。


```
# cp <path>/ca.crt.pem /etc/pki/ca-trust/source/anchors/
```

- b. すべてのユーザーが CA 証明書ファイルを読み取りできるようにするパーミッションを設定します。

```
# chmod 644 /etc/pki/ca-trust/source/anchors/ca.crt.pem
```

- c. CA 信頼データベースを再構築します。

```
# update-ca-trust
```

2. 以下の内容で `/etc/my.cnf.d/mariadb-client-tls.cnf` ファイルを作成します。

```
[client-mariadb]
ssl
ssl-verify-server-cert
```

これらの設定は、MariaDB クライアントが TLS 暗号化 (**ssl**) を使用し、クライアントがホスト名をサーバー証明書 (**ssl-verify-server-cert**) の CN と比較することを定義します。

検証

- ホスト名を使用してサーバーに接続し、サーバーの状態を表示します。

```
# mysql -u root -p -h server.example.com -e status
...
SSL:      Cipher in use is TLS_AES_256_GCM_SHA384
```

SSL エントリーに **Cipher in use is...** が含まれている場合、接続は暗号化されています。

このコマンドで使用するユーザーには、リモートで認証するパーミッションがあることに注意してください。

接続するホスト名がサーバーの TLS 証明書のホスト名と一致しない場合、**ssl-verify-server-cert** パラメーターにより接続が失敗します。たとえば、**localhost** に接続する場合は、以下のようになります。

```
# mysql -u root -p -h localhost -e status
ERROR 2026 (HY000): SSL connection error: Validation of SSL server certificate failed
```

関連情報

- **mysql(1)** man ページの **--ssl*** パラメーターの説明。

2.6. MARIADB データのバックアップ

Red Hat EnterpriseLinux 9 で MariaDB データベースからデータをバックアップする主な方法は 2 つあります。

- 論理バックアップ
- 物理バックアップ

論理バックアップは、データの復元に必要な SQL ステートメントで構成されます。この種類のバックアップは、情報およびレコードをプレーンテキストファイルにエクスポートします。

物理バックアップに対する論理バックアップの主な利点は、移植性と柔軟性です。データは、物理バックアップではできない他のハードウェア設定である MariaDB バージョンまたはデータベース管理システム (DBMS) で復元できます。

mariadb.service が稼働している場合は、論理バックアップを実行できることに注意してください。論理バックアップには、ログと設定ファイルが含まれません。

物理バックアップは、コンテンツを格納するファイルおよびディレクトリーのコピーで構成されます。

物理バックアップは、論理バックアップと比較して、以下の利点があります。

- 出力が少なくなる。
- バックアップのサイズが小さくなる。
- バックアップおよび復元が速くなる。
- バックアップには、ログファイルと設定ファイルが含まれる。

mariadb.service が実行していない場合や、データベースのすべてのテーブルがロックされていて、バックアップ中に変更しないようにする場合は、物理バックアップを実行する必要があります。

以下のいずれかの MariaDB バックアップ方法で、MariaDB データベースのデータのバックアップを使用できます。

- **mariadb-dump** を使用した論理バックアップ
- **Mariabackup** ユーティリティーを使用した物理的なオンラインバックアップ
- ファイルシステムのバックアップ
- バックアップソリューションとしてレプリケーションを使用

2.6.1. mariadb-dump を使用した論理バックアップの実行

mariadb-dump クライアントはバックアップユーティリティーで、バックアップ目的でデータベースまたはデータベースの集合をダンプしたり、別のデータベースサーバーに転送したりできます。**mariadb-dump** の出力は、通常、サーバーテーブル構造を再作成し、データを設定する SQL ステートメントで構成されます。**mariadb-dump** は、CSV などの XML や区切られたテキスト形式など、他の形式のファイルを生成することもできます。

mariadb-dump バックアップを実行するには、以下のいずれかのオプションを使用できます。

- 選択したデータベースを1つまたは複数バックアップ
- すべてのデータベースをバックアップする。
- あるデータベースのテーブルのサブセットのバックアップを作成する。

手順

- 単一のデータベースをダンプするには、以下を実行します。

```
# mariadb-dump [options] --databases db_name > backup-file.sql
```

- 複数のデータベースを一度にダンプするには、次のコマンドを実行します。

```
# mariadb-dump [options] --databases db_name1 [db_name2 ...] > backup-file.sql
```

- すべてのデータベースをダンプするには、以下を実行します。

```
# mariadb-dump [options] --all-databases > backup-file.sql
```

- 1つ以上のダンプされたフルデータベースをサーバーにロードし直すには、以下を実行します。

```
# mariadb < backup-file.sql
```

- データベースをリモート MariaDB サーバーにロードするには、以下を実行します。

```
# mariadb --host=remote_host < backup-file.sql
```

- あるデータベースでテーブルのサブセットをダンプするには、**mariadb-dump** コマンドの末尾に、選択したテーブルの一覧を追加します。

```
# mariadb-dump [options] db_name [tbl_name ...] > backup-file.sql
```

- 1つのデータベースからダンプされたテーブルのサブセットをロードするには、以下を実行します。

```
# mariadb db_name < backup-file.sql
```



注記

この時点で、**db_name** データベースが存在している必要があります。

- **mariadb-dump** がサポートするオプションの一覧を表示するには、以下のコマンドを実行します。

```
$ mariadb-dump --help
```

関連情報

- [MariaDB ドキュメント - mariadb-dump](#).

2.6.2. Mariabackup ユーティリティーを使用した物理的なオンラインバックアップの実行

Mariabackup は、Percona XtraBackup テクノロジーをベースとしたユーティリティーです。これにより、InnoDB、Aria、および MyISAM テーブルの物理的なオンラインバックアップを実行できます。このユーティリティーは、AppStream リポジトリから **mariadb-backup** パッケージで提供されます。

Mariabackup は、**MariaDB** サーバーの完全バックアップ機能に対応します。これには、暗号化されたデータおよび圧縮データが含まれます。

前提条件

- **mariadb-backup** パッケージがシステムにインストールされている。

```
# dnf install mariadb-backup
```

- **Mariabackup** には、バックアップを実行するユーザーの認証情報を指定する必要があります。認証情報はコマンドラインまたは設定ファイルで指定できます。
- **Mariabackup** のユーザーは、**RELOAD**、**LOCK TABLES**、および **REPLICATION CLIENT** の権限が必要です。

Mariabackup を使用してデータベースのバックアップを作成するには、以下の手順を行います。

手順

- コマンドラインで認証情報を提供する間にバックアップを作成するには、以下を実行します。

```
$ mariabackup --backup --target-dir <backup_directory> --user <backup_user> --password <backup_passwd>
```

target-dir オプションは、バックアップファイルを格納するディレクトリーを定義します。完全バックアップを実行する場合は、ターゲットディレクトリーが空であるか、存在しない必要があります。

ユーザー オプションおよび **パスワード** オプションにより、ユーザー名とパスワードを設定できます。

- 設定ファイルに認証情報を設定してバックアップを作成するには、次のコマンドを実行します。
 1. **/etc/my.cnf.d/** ディレクトリーに設定ファイルを作成します (例: **/etc/my.cnf.d/mariabackup.cnf**)。
 2. 以下の行を新規ファイルの **[xtrabackup]** セクションまたは **[mysqld]** セクションに追加します。

```
[xtrabackup]
user=myuser
password=mypassword
```

3. バックアップを実行します。

```
$ mariabackup --backup --target-dir <backup_directory>
```

重要

Mariabackup は、設定ファイルの **[mariadb]** セクションのオプションは読み込みません。MariaDB サーバーにデフォルト以外のデータディレクトリーが指定されている場合は、Mariabackup がデータディレクトリーを見つけることができるように、設定ファイルの **[xtrabackup]** セクションまたは **[mysqld]** セクションでこのディレクトリーを指定する必要があります。

デフォルト以外のデータディレクトリーを指定するには、MariaDB 設定ファイルの **[xtrabackup]** セクションまたは **[mysqld]** セクションに以下の行を追加します。

```
datadir=/var/mycustomdatadir
```

関連情報

- [Full Backup and Restore with Mariabackup](#) .

2.6.3. Mariabackup ユーティリティーを使用したデータの復元

バックアップが完了したら、**mariabackup** コマンドに以下のいずれかのオプションを使用して、バックアップからデータを復元できます。

- **--copy-back** を使用すると、元のバックアップファイルを保持できます。
- **--move-back** は、バックアップファイルをデータディレクトリーに移動し、元のバックアップファイルを削除します。

Mariabackup ユーティリティーを使用してデータを復元するには、以下の手順に従います。

前提条件

- **mariadb** サービスが稼働していないことを確認します。

```
# systemctl stop mariadb.service
```

- データディレクトリーが空であることを確認します。
- Mariabackup のユーザーは、**RELOAD**、**LOCK TABLES**、および **REPLICATION CLIENT** の権限が必要です。

手順

1. **mariabackup** コマンドを実行します。

- データを復元し、元のバックアップファイルを保持するには、**--copy-back** オプションを使用します。

```
$ mariabackup --copy-back --target-dir=/var/mariadb/backup/
```

- データを復元し、元のバックアップファイルを削除するには、**--move-back** オプションを使用します。

```
$ mariabackup --move-back --target-dir=/var/mariadb/backup/
```

2. ファイルの権限を修正します。

データベースを復元するとき、**Mariabackup** は、バックアップのファイルおよびディレクトリーの権限を保持します。ただし、**Mariabackup** は、ユーザーおよびグループがデータベースを復元する際にファイルをディスクに書き込みます。バックアップの復元後、**MariaDB** サーバーのユーザーおよびグループ (通常は共に **mysql**) が一致するように、データディレクトリーの所有者の調整が必要になる場合があります。

たとえば、ファイルの所有権を **mysql** ユーザーおよびグループに再帰的に変更するには、次のコマンドを実行します。

```
# chown -R mysql:mysql /var/lib/mysql/
```

3. **mariadb** サービスを起動します。

```
# systemctl start mariadb.service
```

関連情報

- [Full Backup and Restore with Mariabackup](#) .

2.6.4. ファイルシステムのバックアップの実行

MariaDB データファイルのファイルシステムバックアップを作成するには、**MariaDB** データディレクトリーの内容をバックアップ場所にコピーします。

現在の設定またはログファイルのバックアップも作成するには、以下の手順の中から任意の手順を選択します。

手順

1. **mariadb** サービスを停止します。

```
# systemctl stop mariadb.service
```

2. データファイルを必要な場所にコピーします。

```
# cp -r /var/lib/mysql /backup-location
```

3. 必要に応じて、設定ファイルを必要な場所にコピーします。

```
# cp -r /etc/my.cnf /etc/my.cnf.d /backup-location/configuration
```

4. 必要に応じて、ログファイルを必要な場所にコピーします。

```
# cp /var/log/mariadb/* /backup-location/logs
```

5. **mariadb** サービスを起動します。

```
# systemctl start mariadb.service
```

6. バックアップされたデータをバックアップ場所から `/var/lib/mysql` ディレクトリーに読み込む際は、**mysql:mysql** が `/var/lib/mysql` 内のすべてのデータの所有者であることを確認してください。

```
# chown -R mysql:mysql /var/lib/mysql
```

2.6.5. バックアップソリューションとしてレプリケーションを使用

レプリケーションは、ソースサーバー用の代替バックアップソリューションです。ソースサーバーの複製となるレプリカサーバーを作成すると、ソースに影響を与えずにレプリカでバックアップを実行できます。ソースは、レプリカをシャットダウンする間に依然として実行でき、レプリカからデータのバックアップを作成できます。



警告

レプリケーション自体は、バックアップソリューションとしては十分ではありません。レプリケーションは、ハードウェア障害からソースサーバーを保護しますが、データ損失に対する保護は保証していません。この方法とともに、レプリカでその他のバックアップソリューションを使用することが推奨されます。

関連情報

- MariaDB Galera クラスターを使用して MariaDB データベースを複製する方法の詳細は、「[Galera で MariaDB を複製する](#)」を参照してください。
- レプリケーションをバックアップソリューションとして使用する場合は、[MariaDB ドキュメンテーション](#) を参照してください。

2.7. MARIADB 10.5 への移行

RHEL 8 では、MariaDB サーバーはバージョン 10.3 および 10.5 で利用できます。各サーバーは、個別のモジュールストリームで提供されます。RHEL 9 は、**MariaDB 10.5** および **MySQL 8.0** を提供します。ここでは、RHEL 8 バージョンの **MariaDB 10.3** から RHEL 9 バージョンの **MariaDB 10.5** への移行を説明します。

2.7.1. MariaDB 10.3 と MariaDB 10.5 の主な相違点

MariaDB 10.3 と MariaDB 10.5 の間の重要な変更点には以下が含まれます。

- MariaDB はデフォルトで **unix_socket** 認証プラグインを使用するようになりました。ユーザーはこのプラグインを使用すると、ローカルの Unix ソケットファイルを介して MariaDB に接続する際に、オペレーティングシステムの認証情報を使用できます。
- MariaDB は、**mariadb-*** という名前のバイナリーと、**mariadb-*** バイナリーを指す **mysql*** シンボリックリンクを追加します。たとえば、**mysqladmin**、**mysqlaccess**、および **mysqlshow** のシンボリックリンクは、**mariadb-admin**、**mariadb-access**、および **mariadb-show** のバイナリーをそれぞれポイントします。
- 各ユーザーロールに合わせて、**SUPER** 特権が複数の特権に分割されました。その結果、一部のステートメントに必要な特権が変更されました。

- 並列のレプリケーションでは、**slave_parallel_mode** はデフォルトで **optimistic** に設定されるようになりました。
- InnoDB ストレージエンジンで、変数のデフォルトが変更されました (**innodb_adaptive_hash_index** は **OFF** へ、**innodb_checksum_algorithm** は **full_crc32** へ変更)。
- MariaDB は、以前使用された **readline** ライブラリーではなく、MariaDB コマンド履歴 (**.mysql_history** ファイル) を管理する基盤となるソフトウェアの **libedit** 実装を使用するようになりました。この変更は、**.mysql_history** ファイルを直接使用しているユーザーに影響します。**.mysql_history** は MariaDB または MySQL アプリケーションによって管理されるファイルであるため、ユーザーは直接ファイルでは機能しないことに注意してください。人間が判読可能な外観は偶然です。



注記

セキュリティを強化するために、履歴ファイルの維持を考慮することができます。コマンド履歴の記録を無効にするには、以下を実行します。

1. 存在する場合は、**.mysql_history** ファイルを削除します。
2. 以下のいずれかの方法を使用します。
 - **MYSQL_HISTFILE** 変数を **/dev/null** に設定し、これをシェルの起動ファイルに追加します。
 - **.mysql_history** ファイルを **/dev/null** へのシンボリックリンクに変更します。

```
$ ln -s /dev/null $HOME/.mysql_history
```

MariaDB Galera クラスタがバージョン 4 にアップグレードされ、以下の主な変更点が加えられました。

- Galera は、サイズ制限なしのトランザクションの複製をサポートする、新しいストリーミングレプリケーション機能を追加します。ストリーミングレプリケーションの実行時に、クラスタは小さなフラグメントでトランザクションを複製します。
- Galera が Global Transaction ID (GTID) に完全に対応するようになりました。
- **/etc/my.cnf.d/galera.cnf** ファイルの **wsrep_on** オプションのデフォルト値が **1** から **0** に変更され、エンドユーザーが必要な追加オプションを設定せずに **wsrep** レプリケーションを開始できないようにします。

MariaDB 10.5 の PAM プラグインへの変更には、以下が含まれます。

- MariaDB 10.5 は、PAM (Pluggable Authentication Modules) プラグインの新バージョンを追加します。PAM プラグインバージョン 2.0 は、個別の **setuid root** ヘルパーバイナリーを使用して PAM 認証を実行します。これにより、MariaDB が追加の PAM モジュールを使用できるようになります。
- ヘルパーバイナリーは、**mysql** グループのユーザーによってのみ実行できます。デフォルトでは、グループには **mysql** ユーザーのみが含まれます。Red Hat では、このヘルパーユーティリティを介してスロットルまたはログの記録をせずにパスワード推測攻撃を防ぐために、管理者が **mysql** グループにユーザーをさらに追加しないことを推奨しています。

- MariaDB 10.5 では、プラグ可能な認証モジュール (PAM) プラグインとその関連ファイルが新しいパッケージ **mariadb-pam** に移動しました。したがって、**MariaDB** に PAM 認証を使用しないシステムに、新しい **setuid root** バイナリーが導入されることはありません。
- **mariadb-pam** パッケージには、両方の PAM プラグインバージョンが含まれています。バージョン 2.0 はデフォルトで、バージョン 1.0 は **auth_pam_v1** 共有オブジェクトライブラリーとして利用できます。
- MariaDB サーバーでは、**mariadb-pam** パッケージはデフォルトでインストールされません。MariaDB 10.5 で PAM 認証プラグインを利用できるようにするには、**mariadb-pam** パッケージを手動でインストールします。

2.7.2. RHEL 8 バージョンの MariaDB 10.3 から、RHEL 9 バージョンの MariaDB 10.5 への移行

この手順では、**mariadb-upgrade** ユーティリティーを使用して、MariaDB 10.3 から MariaDB 10.5 に移行する方法を説明します。

mariadb-upgrade ユーティリティーは、**mariadb-server-utils** サブパッケージにより提供され、**mariadb-server** パッケージの依存関係としてインストールされます。

前提条件

- アップグレードを実行する前に、MariaDB データベースに保存されている全データのバックアップを作成します。

手順

1. **mariadb-server** パッケージが RHEL9 システムにインストールされていることを確認します。

```
# dnf install mariadb-server
```

2. データのコピー時に、**mariadb** サービスがソースおよびターゲットのシステムで稼働していないことを確認します。

```
# systemctl stop mariadb.service
```

3. ソースの場所から RHEL 9 ターゲットシステムの **/var/lib/mysql/** ディレクトリーにデータをコピーします。
4. ターゲットシステムでコピーされたファイルに適切なパーミッションと SELinux コンテキストを設定します。

```
# restorecon -vr /var/lib/mysql
```

5. **mysql:mysql** が **/var/lib/mysql** ディレクトリー内のすべてのデータの所有者であることを確認してください。

```
# chown -R mysql:mysql /var/lib/mysql
```

6. **/etc/my.cnf.d/** にあるオプションファイルに MariaDB 10.5 に対して有効なオプションのみが含まれるように、設定を調整します。詳細は、[MariaDB 10.4](#) および [MariaDB 10.5](#) のアップストリームドキュメントを参照してください。

7. ターゲットシステムで、**MariaDB** サーバーを起動します。

- スタンドアロンを実行しているデータベースをアップグレードする場合:

```
# systemctl start mariadb.service
```

- **Galera** クラスターノードをアップグレードする場合:

```
# galera_new_cluster
```

mariadb サービスが自動的に起動します。

8. **mariadb-upgrade** ユーティリティを実行して、内部テーブルをチェックし、修復します。

- スタンドアロンを実行しているデータベースをアップグレードする場合:

```
$ mariadb-upgrade
```

- **Galera** クラスターノードをアップグレードする場合:

```
$ mariadb-upgrade --skip-write-binlog
```



重要

インプレースアップグレードに関連する特定のリスクと既知の問題があります。たとえば、一部のクエリーが動作しなかったり、アップグレード前とは異なる順序で実行される場合があります。これらのリスクと問題、およびインプレースアップグレードに関する全般的な情報は、[MariaDB 10.5 Release Notes](#) を参照してください。

2.8. GALERA で MARIADB を複製する

本セクションでは、Red Hat Enterprise Linux 9 の Galera ソリューションを使用して、**MariaDB** データベースを複製する方法を説明します。

2.8.1. MariaDB Galera クラスターの概要

Galera レプリケーションは、複数の **MariaDB** サーバーで構成される同期マルチソース **MariaDB Galera クラスター** の作成に基づいています。レプリカが通常読み取り専用である従来のプライマリー/レプリカ設定とは異なり、**MariaDB Galera クラスター** のノードはすべて書き込み可能にすることができます。

Galera レプリケーションと **MariaDB** データベースとの間のインターフェースは、書き込みセットレプリケーション API (**wsrep API**) で定義されます。

MariaDB Galera クラスター の主な機能は以下のとおりです。

- 同期のレプリケーション
- アクティブ/アクティブのマルチソーストポロジー
- クラスターノードへの読み取りおよび書き込み
- 自動メンバーシップ制御、失敗したノードのクラスターからの削除

- 自動ノードの参加
- 行レベルの並列レプリケーション
- ダイレクトクライアント接続: ユーザーはクラスターノードにログインし、レプリケーションの実行中にノードを直接操作できます。

同期レプリケーションとは、サーバーがトランザクションに関連付けられた書き込みセットをクラスター内のすべてのノードにブロードキャストすることで、コミット時にトランザクションをレプリケートすることを意味します。クライアント(ユーザーアプリケーション)はデータベース管理システム(DBMS)に直接接続し、ネイティブの MariaDB と同様の動作が発生します。

同期レプリケーションは、クラスター内の1つのノードで発生した変更が、クラスター内の他のノードで同時に発生することを保証します。

そのため、同期レプリケーションには、非同期のレプリケーションと比べて次のような利点があります。

- 特定のクラスターノード間の変更の伝播に遅延がない
- すべてのクラスターノードには常に一貫性がある
- いずれかのクラスターノードがクラッシュしても、最新の変更は失われない
- すべてのクラスターノードのトランザクションが並列に実行する
- クラスター全体にわたる因果関係

関連情報

- [About Galera replication](#)
- [What is MariaDB Galera Cluster](#)
- [Getting started with MariaDB Galera Cluster](#)

2.8.2. MariaDB Galera クラスターを構築するためのコンポーネント

MariaDB Galera クラスターを構築するには、システムに以下のパッケージをインストールする必要があります。

- **mariadb-server-galera**: MariaDB Galera クラスターのサポートファイルとスクリプトが含まれます。
- **mariadb-server**: MariaDB アップストリームがパッチを適用し、書き込みセットレプリケーション API (wsrep API) を組み込みます。この API は、Galera レプリケーションと MariaDB との間のインターフェースを提供します。
- **galera**: MariaDB アップストリームがパッチを適用し、MariaDB の完全サポートを追加します。galera パッケージには、以下の内容が含まれます。
 - Galera Replication Library は、レプリケーション機能全体を提供します。
 - Galera Arbitrator ユーティリティは、スプリットブレインのシナリオで投票に参加するクラスターメンバーとして使用できます。ただし、Galera Arbitrator は実際のレプリケーションには参加できません。

- Galera Arbitrator ユーティリティーのデプロイに使用される **Galera Systemd service** および **Galera wrapper script**。RHEL 9 は、`/usr/lib/systemd/system/garbd.service` および `/usr/sbin/garb-systemd` にあるこれらのファイルのアップストリームバージョンを提供します。

関連情報

- [Galera Replication Library](#)
- [Galera Arbitrator](#)
- [mysql-wsrep プロジェクト](#)

2.8.3. MariaDB Galera クラスターのデプロイメント

前提条件

- MariaDB Galera Cluster パッケージをインストールします。以下に例を示します。

```
# dnf install galera
```

これにより、以下のパッケージがインストールされます。

- mariadb-server-galera**
- mariadb-server**
- galera**
mariadb-server-galera パッケージが **mariadb-server** パッケージおよび **galera** パッケージを依存関係としてプルします。

MariaDB Galera クラスターを構築するコンポーネントの詳細は、「[MariaDB Galera クラスターを構築するためのコンポーネント](#)」を参照してください。

- MariaDB サーバーのレプリケーション設定は、システムを初めてクラスターに追加する前に更新する必要があります。
デフォルト設定は、`/etc/my.cnf.d/galera.cnf` ファイルで配布されます。

MariaDB Galera クラスターをデプロイする前に、以下の文字列で開始するように、すべてのノードの `/etc/my.cnf.d/galera.cnf` ファイルに **wsrep_cluster_address** オプションを設定します。

```
gcomm://
```

- 初期ノードでは、**wsrep_cluster_address** を空のリストとして設定できます。

```
wsrep_cluster_address="gcomm://"
```

- その他のすべてのノードに **wsrep_cluster_address** を設定して、実行中のクラスターに属するノードへのアドレスを追加します。以下に例を示します。

```
wsrep_cluster_address="gcomm://10.0.0.10"
```

Galera Cluster アドレスの設定方法は、「[Galera Cluster Address](#)」を参照してください。

手順

1. ノードで以下のラッパーを実行して、新規クラスターの最初のノードをブートストラップします。

```
# galera_new_cluster
```

このラッパーにより、MariaDB サーバーデーモン (**mariadb**) に **--wsrep-new-cluster** オプションが指定されて実行されるようになります。このオプションは、接続する既存クラスターがないという情報を提供します。したがって、ノードは新規 UUID を作成し、新しいクラスターを特定します。



注記

mariadb サービスは、複数の MariaDB サーバープロセスと対話する systemd メソッドをサポートします。したがって、複数の MariaDB サーバーを実行している場合は、インスタンス名を接尾辞として指定して、特定のインスタンスをブートストラップできます。

```
# galera_new_cluster mariadb@node1
```

2. 各ノードで次のコマンドを実行して、その他のノードをクラスターに接続します。

```
# systemctl start mariadb
```

その結果、ノードはクラスターに接続し、それ自体をクラスターの状態と同期します。

関連情報

- [Getting started with MariaDB Galera Cluster](#) .

2.8.4. 新規ノードの MariaDB Galera クラスターへの追加

新規ノードを MariaDB Galera クラスターに追加するには、以下の手順に従います。

この手順に従って、既存のノードを再接続することもできます。

手順

- 特定のノードで、**/etc/my.cnf.d/galera.cnf** 設定ファイルの **[mariadb]** セクション内にある **wsrep_cluster_address** オプションで、1つ以上の既存クラスターメンバーにアドレスを指定します。

```
[mariadb]
wsrep_cluster_address="gcomm://192.168.0.1"
```

新規ノードを既存クラスターノードのいずれかに接続すると、クラスター内のすべてのノードを表示できるようになります。

ただし、**wsrep_cluster_address** のクラスターの全ノードを表示することが推奨されます。

したがって、1つ以上のクラスターノードがダウンしても、その他のクラスターノードに接続することでノードがクラスターに参加できます。すべてのメンバーがメンバーシップに同意する

と、クラスターの状態が変更します。新規ノードの状態がクラスターの状態と異なる場合、新しいノードは Incremental State Transfer (IST) または State Snapshot Transfer (SST) のいずれかを要求し、他のノードとの一貫性を確保します。

関連情報

- [Getting started with MariaDB Galera Cluster](#) .
- [Introduction to State Snapshot Transfers](#) .

2.8.5. MariaDB Galera クラスターの再起動

すべてのノードを同時にシャットダウンすると、クラスターが終了し、実行中のクラスターは存在しなくなります。ただし、クラスターのデータは引き続き存在します。

クラスターを再起動するには、「[MariaDB Galera クラスターのデプロイメント](#)」の説明に従って、最初のノードをブートストラップします。



警告

クラスターがブートストラップされず、最初のノードの `mariadb` が `systemctl start mariadb` コマンドでのみ起動した場合、ノードは `/etc/my.cnf.d/galera.cnf` ファイルの `wsrep_cluster_address` オプションに記載されている少なくとも1つのノードに接続しようとします。ノードが現在実行していない場合は、再起動に失敗します。

関連情報

- [Getting started with MariaDB Galera Cluster](#) .

第3章 MySQL の使用

MySQL サーバーは、オープンソースの高速で堅牢なデータベースサーバーです。ここでは、RHEL システムに MySQL をインストールして設定する方法、MySQL データをバックアップする方法、MySQL の以前のバージョンから移行する方法、および MySQL を複製する方法について説明します。

3.1. MySQL を使い始める

MySQL は、データを構造化情報に変換して、データにアクセスする SQL インターフェースを提供するリレーショナルデータベースです。これには、複数のストレージエンジンとプラグイン、および地理的な情報システム (GIS) および JSON (JavaScript Object Notation) 機能が含まれます。

ここでは、以下について説明します。

- [MySQL のインストール](#) で、MySQL サーバーをインストールする方法について。
- [MySQL の設定](#) で、MySQL 設定を調整する方法について。
- [MySQL データのバックアップ](#) で、MySQL データをバックアップする方法について。
- [MySQL 8.0 の RHEL9 バージョンへの移行](#) で、MySQL 8.0 の RHEL 8 バージョンから MySQL 8.0 の RHEL 9 バージョンへの移行方法について。
- [MySQL の複製](#) で、MySQL データベースを複製する方法について。

3.2. MySQL のインストール

RHEL 9.0 は、この Application Stream の初期バージョンとして MySQL 8.0 を提供します。これは、RPM パッケージとして簡単にインストールできます。

MySQL をインストールするには、以下の手順に従います。

手順

1. MySQL サーバーパッケージをインストールします。

```
# dnf install mysql-server
```

2. **mysqld** サービスを開始します。

```
# systemctl start mysqld.service
```

3. **mysqld** サービスを有効にして、起動時に起動するようにします。

```
# systemctl enable mysqld.service
```

4. **推奨**:MySQL をインストールする際のセキュリティーを向上させるには、以下のコマンドを実行します。

```
$ mysql_secure_installation
```

このコマンドは、完全にインタラクティブなスクリプトを起動して、プロセスの各ステップのプロンプトを表示します。このスクリプトを使用すると、次の方法でセキュリティーを改善できます。

- root アカウントのパスワードの設定
- 匿名ユーザーの削除
- リモート root ログインの拒否 (ローカルホスト外)



注記

RPM パッケージが競合しているため、RHEL 9 では **MySQL** および **MariaDB** データベースサーバーを同時にインストールすることはできません。RHEL 9 では、コンテナ内で異なるバージョンのデータベースサーバーを使用できます。

3.3. MYSQL の設定

MySQL サーバーをネットワーク用に設定するには、以下の手順に従います。

手順

1. `/etc/my.cnf.d/mysql-server.cnf` ファイルの **[mysqld]** セクションを編集します。以下の設定ディレクティブを設定できます。
 - **bind-address:** サーバーがリッスンするアドレスです。設定可能なオプションは以下のとおりです。
 - ホスト名
 - IPv4 アドレス
 - IPv6 アドレス
 - **skip-networking:** サーバーが TCP/IP 接続をリッスンするかどうかを制御します。以下の値が使用できます。
 - 0 - すべてのクライアントをリッスンする
 - 1 - ローカルクライアントのみをリッスンする
 - **port:** MySQL が TCP/IP 接続をリッスンするポート。
2. **mysqld** サービスを再起動します。

```
# systemctl restart mysqld.service
```

3.4. MYSQL データのバックアップ

Red Hat Enterprise Linux 9 で **MySQL** データベースからデータをバックアップする主な方法は 2 つあります。

- 論理バックアップ
- 物理バックアップ

論理バックアップは、データの復元に必要な SQL ステートメントで構成されます。この種類のバックアップは、情報およびレコードをプレーンテキストファイルにエクスポートします。

物理バックアップに対する論理バックアップの主な利点は、移植性と柔軟性です。データは、物理バックアップではできない他のハードウェア設定である **MySQL** バージョンまたはデータベース管理システム (DBMS) で復元できます。

mysqld.service が実行されている場合は、論理バックアップを実行できることに注意してください。論理バックアップには、ログと設定ファイルが含まれません。

物理バックアップは、コンテンツを格納するファイルおよびディレクトリーのコピーで構成されます。

物理バックアップは、論理バックアップと比較して、以下の利点があります。

- 出力が少なくなる。
- バックアップのサイズが小さくなる。
- バックアップおよび復元が速くなる。
- バックアップには、ログファイルと設定ファイルが含まれる。

mysqld.service が実行されていない場合、またはバックアップ中の変更を防ぐためにデータベース内のすべてのテーブルがロックされている場合は、物理バックアップを実行する必要があることに注意してください。

以下の **MySQL** バックアップアプローチのいずれかを使用して、**MySQL** データベースからデータをバックアップできます。

- **mysqldump** を使用した論理バックアップ
- ファイルシステムのバックアップ
- バックアップソリューションとしてレプリケーションを使用

3.4.1. **mysqldump** を使用した論理バックアップの実行

mysqldump クライアントはバックアップユーティリティーで、バックアップ目的でデータベースまたはデータベースの集合をダンプしたり、別のデータベースサーバーに転送したりできます。**mysqldump** の出力は通常、サーバーテーブル構造を再作成するか、データを入力するか、またはその両方を行う SQL ステートメントで構成されます。**mysqldump** は、XML や CSV などの区切りテキスト形式を含む他の形式のファイルを生成することもできます。

mysqldump バックアップを実行するには、以下のいずれかのオプションを使用できます。

- 選択したデータベースを1つまたは複数バックアップ
- すべてのデータベースをバックアップする。
- あるデータベースのテーブルのサブセットのバックアップを作成する。

手順

- 単一のデータベースをダンプするには、以下を実行します。

```
# mysqldump [options] --databases db_name > backup-file.sql
```

- 複数のデータベースを一度にダンプするには、次のコマンドを実行します。

```
# mysqldump [options] --databases db_name1 [db_name2 ...] > backup-file.sql
```

- すべてのデータベースをダンプするには、以下を実行します。

```
# mysqldump [options] --all-databases > backup-file.sql
```

- 1つ以上のダンプされたフルデータベースをサーバーにロードし直すには、以下を実行します。

```
# mysql < backup-file.sql
```

- データベースをリモート MySQL サーバーにロードするには、以下を実行します。

```
# mysql --host=remote_host < backup-file.sql
```

- あるデータベースでテーブルのサブセットをダンプするには、**mysqldump** コマンドの末尾に、選択したテーブルの一覧を追加します。

```
# mysqldump [options] db_name [tbl_name ...] > backup-file.sql
```

- 1つのデータベースからダンプされたテーブルのサブセットをロードするには、以下を実行します。

```
# mysql db_name < backup-file.sql
```



注記

この時点で、**db_name** データベースが存在している必要があります。

- **mysqldump** がサポートするオプションの一覧を表示するには、以下を実行します。

```
$ mysqldump --help
```

関連情報

- [mysqldump を使用した論理バックアップ](#)

3.4.2. ファイルシステムのバックアップの実行

MySQL データファイルのファイルシステムバックアップを作成するには、MySQL データディレクトリの内容をバックアップ場所にコピーします。

現在の設定またはログファイルのバックアップも作成するには、以下の手順の中から任意の手順を選択します。

手順

1. **mysqld** サービスを停止します。

```
# systemctl stop mysqld.service
```

2. データファイルを必要な場所にコピーします。

```
# cp -r /var/lib/mysql /backup-location
```

3. 必要に応じて、設定ファイルを必要な場所にコピーします。

```
# cp -r /etc/my.cnf /etc/my.cnf.d /backup-location/configuration
```

4. 必要に応じて、ログファイルを必要な場所にコピーします。

```
# cp /var/log/mysql/* /backup-location/logs
```

5. **mysqld** サービスを開始します。

```
# systemctl start mysqld.service
```

6. バックアップされたデータをバックアップ場所から **/var/lib/mysql** ディレクトリーに読み込む際は、**mysql:mysql** が **/var/lib/mysql** 内のすべてのデータの所有者であることを確認してください。

```
# chown -R mysql:mysql /var/lib/mysql
```

3.4.3. バックアップソリューションとしてレプリケーションを使用

レプリケーションは、ソースサーバー用の代替バックアップソリューションです。ソースサーバーの複製となるレプリカサーバーを作成すると、ソースに影響を与えずにレプリカでバックアップを実行できます。ソースは、レプリカをシャットダウンする間に依然として実行でき、レプリカからデータのバックアップを作成できます。

MySQLデータベースを複製する方法の手順については、[MySQL の複製](#) を参照してください。



警告

レプリケーション自体は、バックアップソリューションとしては十分ではありません。レプリケーションは、ハードウェア障害からソースサーバーを保護しますが、データ損失に対する保護は保証していません。この方法とともに、レプリカでその他のバックアップソリューションを使用することが推奨されます。

関連情報

- [MySQL replication documentation](#)

3.5. MYSQL8.0 の RHEL9 バージョンへの移行

RHEL 8 には、MySQL データベースファミリーのサーバーの MySQL 8.0、MariaDB 10.3、および MariaDB 10.5 の実装が含まれています。RHEL 9 は、MySQL 8.0 および MariaDB 10.5 を提供します。

この手順では、`mysql_upgrade`ユーティリティーを使用して、RHEL 8バージョンのMySQL 8.0からRHEL 9バージョンのMySQL 8.0への移行について説明します。`mysql_upgrade`ユーティリティーは、`mysql-server`パッケージによって提供されます。

前提条件

- アップグレードを実行する前に、MySQL データベースに保存されているすべてのデータをバックアップすること。[MySQL データのバックアップ](#)を参照してください。

手順

1. `mysql-server` パッケージが RHEL9 システムにインストールされていることを確認します。

```
# dnf install mysql-server
```

2. データのコピー時に、`mysqld` サービスがソースシステムとターゲットシステムのどちらでも実行されていないことを確認してください。

```
# systemctl stop mysqld.service
```

3. ソースの場所から RHEL 9 ターゲットシステムの `/var/lib/mysql/` ディレクトリーにデータをコピーします。
4. ターゲットシステムでコピーされたファイルに適切なパーミッションと SELinux コンテキストを設定します。

```
# restorecon -vr /var/lib/mysql
```

5. `mysql:mysql` が、`/var/lib/mysql` ディレクトリー内のすべてのデータの所有者であることを確認してください。

```
# chown -R mysql:mysql /var/lib/mysql
```

6. ターゲットシステムで MySQL サーバーを起動します。

```
# systemctl start mysqld.service
```

注記:MySQL の以前のバージョンでは、内部テーブルをチェックおよび修復するために `mysql_upgrade` コマンドが必要でした。これは、サーバーの起動時に自動的に実行されるようになりました。

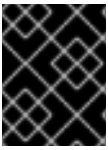
3.6. MYSQL の複製

MySQL には、基本的なものから高度なものまで、レプリケーション用のさまざまな設定オプションが用意されています。このセクションでは、グローバルトランザクション識別子 (GTID) を使用して、新しくインストールした MySQL サーバーに MySQL でレプリケートするトランザクションベースの方法について説明します。GTID を使用すると、トランザクションの識別と整合性の検証が簡素化されます。

MySQL でレプリケーションを設定するには、以下を行う必要があります。

- [ソースサーバーを設定する](#)

- レプリカサーバーを設定する
- ソースサーバーにレプリケーションユーザーを作成する
- レプリカサーバーをソースサーバーに接続する



重要

レプリケーションに既存の MySQL サーバーを使用する場合は、最初にデータを同期する必要があります。詳細は、[アップストリームのドキュメント](#)を参照してください。

3.6.1. MySQL ソースサーバーの設定

このセクションでは、MySQL ソースサーバーがデータベースサーバーで行われたすべての変更を適切に実行および複製するために必要な設定オプションについて説明します。

前提条件

- ソースサーバーがインストールされている。

手順

1. **[mysqld]** セクションの `/etc/my.cnf.d/mysql-server.cnf` ファイルに以下のオプションを含めません。
 - **bind-address=source_ip_address**
このオプションは、レプリカからソースへの接続に必要です。
 - **server-id=id**
id は一意である必要があります。
 - **log_bin=path_to_source_server_log**
このオプションは、MySQL ソースサーバーのバイナリーログファイルへのパスを定義します。例: `log_bin=/var/log/mysql/mysql-bin.log`
 - **gtid_mode=ON**
このオプションは、サーバー上でグローバルトランザクション識別子 (GTID) を有効にします。
 - **enforce-gtid-consistency=ON**
サーバーは、GTID を使用して安全にログに記録できるステートメントのみの実行を許可することにより、GTID の整合性を強化します。
 - オプション: **binlog_do_db=db_name**
選択したデータベースのみを複製する場合は、このオプションを使用します。選択した複数のデータベースを複製するには、各データベースを個別に指定します。


```
binlog_do_db=db_name1
binlog_do_db=db_name2
binlog_do_db=db_name3
```
 - オプション: **binlog_ignore_db=db_name**
このオプションを使用して、特定のデータベースをレプリケーションから除外します。
2. **mysqld** サービスを再起動します。

```
# systemctl restart mysqld.service
```

3.6.2. MySQL レプリカサーバーの設定

このセクションでは、レプリケーションを成功させるために MySQL レプリカサーバーに必要な設定オプションについて説明します。

前提条件

- レプリカサーバーがインストールされている。

手順

- [mysqld]** セクションの `/etc/my.cnf.d/mysql-server.cnf` ファイルに以下のオプションを含めます。

- **server-id=id**
id は一意である必要があります。
- **relay-log=path_to_replica_server_log**
リレーログは、レプリケーション中に MySQL レプリカサーバーによって作成されたログファイルのセットです。
- **log_bin=path_to_replica_sever_log**
このオプションは、MySQL レプリカサーバーのバイナリーログファイルへのパスを定義します。例: `log_bin=/var/log/mysql/mysql-bin.log`

このオプションはレプリカでは必須ではありませんが、強く推奨します。

- **gtid_mode=ON**
このオプションは、サーバー上でグローバルトランザクション識別子 (GTID) を有効にします。
- **enforce-gtid-consistency=ON**
サーバーは、GTID を使用して安全にログに記録できるステートメントのみの実行を許可することにより、GTID の整合性を強化します。
- **log-replica-updates=ON**
このオプションにより、ソースサーバーから受信した更新がレプリカのバイナリーログに記録されます。
- **skip-replica-start=ON**
このオプションは、レプリカサーバーの起動時に、レプリカサーバーがレプリケーションスレッドを開始しないようにします。
- **オプション: binlog_do_db=db_name**
特定のデータベースのみを複製する場合は、このオプションを使用します。複数のデータベースを複製するには、各データベースを個別に指定します。

```
binlog_do_db=db_name1
binlog_do_db=db_name2
binlog_do_db=db_name3
```

- **オプション: binlog_ignore_db=db_name**

このオプションを使用して、特定のデータベースをレプリケーションから除外します。

2. **mysqld** サービスを再起動します。

```
# systemctl restart mysqld.service
```

3.6.3. MySQL ソースサーバーでのレプリケーションユーザーの作成

レプリケーションユーザーを作成し、このユーザーにレプリケーショントラフィックに必要なパーミッションを付与する必要があります。この手順は、適切なパーミッションを持つレプリケーションユーザーを作成する方法を示しています。これらの手順は、ソースサーバーでのみ実行してください。

前提条件

- ソースサーバーは、[MySQL ソースサーバーの設定](#) で説明されているように、インストールおよび設定されている。

手順

1. レプリケーションユーザーを作成します。

```
mysql> CREATE USER 'replication_user'@'replica_server_ip' IDENTIFIED WITH mysql_native_password BY 'password';
```

2. ユーザーにレプリケーション権限を付与します。

```
mysql> GRANT REPLICATION SLAVE ON *.* TO 'replication_user'@'replica_server_ip';
```

3. MySQL データベースの付与テーブルを再読み込みします。

```
mysql> FLUSH PRIVILEGES;
```

4. ソースサーバーを読み取り専用状態に設定します。

```
mysql> SET @@GLOBAL.read_only = ON;
```

3.6.4. レプリカサーバーをソースサーバーに接続する

MySQL レプリカサーバーでは、認証情報とソースサーバーのアドレスを設定する必要があります。次の手順を使用して、レプリカサーバーを実装します。

前提条件

- ソースサーバーは、[MySQL ソースサーバーの設定](#) で説明されているように、インストールおよび設定されている。
- レプリカサーバーは、[MySQL レプリカサーバーの設定](#) で説明されているように、インストールおよび設定されている。
- レプリケーションユーザーを作成している。[MySQL ソースサーバーでのレプリケーションユーザーの作成](#) を参照してください。

手順

1. レプリカサーバーを読み取り専用状態に設定します。

```
mysql> SET @@GLOBAL.read_only = ON;
```

2. レプリケーションソースを設定します。

```
mysql> CHANGE REPLICATION SOURCE TO
-> SOURCE_HOST='source_ip_address',
-> SOURCE_USER='replication_user',
-> SOURCE_PASSWORD='password',
-> SOURCE_AUTO_POSITION=1;
```

3. MySQL レプリカサーバーでレプリカスレッドを開始します。

```
mysql> START REPLICA;
```

4. ソースサーバーとレプリカサーバーの両方で、読み取り専用状態の設定を解除します。

```
mysql> SET @@GLOBAL.read_only = OFF;
```

5. オプション:デバッグの目的で、レプリカサーバーのステータスを検査します。

```
mysql> SHOW REPLICA STATUS\G;
```



注記

レプリカサーバーの起動または接続に失敗した場合は、**SHOW MASTER STATUS** コマンドの出力に表示されるバイナリーログファイルの位置に続く特定の数のイベントをスキップできます。たとえば、定義された位置から最初のイベントをスキップします。

```
mysql> SET GLOBAL SQL_SLAVE_SKIP_COUNTER=1;
```

レプリカサーバーを再起動してみてください。

6. オプション:レプリカサーバーでレプリカスレッドを停止します。

```
mysql> STOP REPLICA;
```

3.6.5. 検証手順

1. ソースサーバーにサンプルデータベースを作成します。

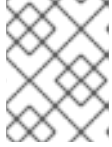
```
mysql> CREATE DATABASE test_db_name;
```

2. **test_db_name** データベースが、レプリカサーバーで複製されていることを確認します。

3. ソースサーバーまたはレプリカサーバーのいずれかで以下のコマンドを実行して、MySQL サーバーのバイナリーログファイルに関するステータス情報を表示します。


```
mysql> SHOW MASTER STATUS;
```

ソースで実行されたトランザクションの GTID のセットを示す **Executed_Gtid_Set** 列は、空であってはなりません。



注記

レプリカサーバーで **SHOW SLAVE STATUS** を使用すると、同じ GTID のセットが **Executed_Gtid_Set** 行に表示されます。

3.6.6. 関連情報

- [MySQL Replication documentation](#)
- [How To Set Up Replication in MySQL](#)
- [Replication with Global Transaction Identifiers](#)

第4章 POSTGRESQL の使用

PostgreSQL サーバーは、SQL 言語をベースにした、オープンソースの堅牢かつ拡張性に優れたデータベースサーバーです。ここでは、RHEL システムに PostgreSQL をインストールして設定する方法、PostgreSQL データをバックアップする方法、および PostgreSQL の以前のバージョンから移行する方法について説明します。

4.1. POSTGRESQL の概要

PostgreSQL サーバーは、オブジェクトリレーショナルデータベースシステムを提供します。これにより、広範なデータセットと多数の同時ユーザーを管理できます。このような理由から、PostgreSQL サーバーは、大量のデータを管理するためにクラスターで使用できます。

PostgreSQL サーバーには、データの整合性の確保、耐障害性のある環境の構築、アプリケーションの構築を行うための機能が含まれます。ユーザーは、データベースを再コンパイルすることなく、ユーザー独自のデータ型、カスタム関数、またはさまざまなプログラミング言語のコードでデータベースを拡張できます。

ここでは、以下について説明します。

- 「[PostgreSQL のインストール](#)」で、PostgreSQL のインストール方法について。
- 「[PostgreSQL ユーザー](#)」で、ユーザー、ロール、および権限について。
- 「[PostgreSQL の設定](#)」で、PostgreSQL 設定の調整方法について。
- 「[PostgreSQL データのバックアップ](#)」で、データベースのバックアップ方法について。
- 「[PostgreSQL の RHEL9 バージョンへの移行](#)」で、PostgreSQL 13 の RHEL 9 バージョンへの移行方法について。移行の前提条件に、データバックアップの実行があります。

4.2. POSTGRESQL のインストール

RHEL 9.0 は、この Application Stream の初期バージョンとして PostgreSQL 13 を提供します。これは、RPM パッケージとして簡単にインストールできます。追加の PostgreSQL バージョンは、RHEL 9 の今後のマイナーリリースで、ライフサイクルが短いモジュールとして提供されます。

PostgreSQL をインストールするには、以下の手順に従います。

手順

1. PostgreSQL サーバーパッケージをインストールします。

```
# dnf install postgresql-server
```

postgres のスーパーユーザーが自動的に作成されます。

2. データベースクラスターを初期化します。

```
# postgresql-setup --initdb
```

Red Hat は、デフォルトの `/var/lib/pgsql/data` ディレクトリーにデータを保存することを推奨します。

3. **postgresql** サービスを開始します。

```
# systemctl start postgresql.service
```

4. **postgresql** サービスが、システムの起動時に起動するようにします。

```
# systemctl enable postgresql.service
```

4.3. POSTGRESQL ユーザー

PostgreSQL ユーザーは以下のタイプのもので。

- **postgres** UNIX システムユーザー: PostgreSQL サーバーおよびクライアントアプリケーション (**pg_dump** など) を実行する場合にのみ使用してください。データベース作成およびユーザー管理などの、PostgreSQL 管理上の対話的な作業には、**postgres** システムユーザーを使用しないでください。
- データベースのスーパーユーザー: デフォルトの **postgres** PostgreSQL スーパーユーザーは、**postgres** システムユーザーとは関係ありません。 **pg_hba.conf** ファイルの **postgres** のスーパーユーザーのアクセスを制限することができます。制限しない場合には、その他のパーミッションの制限はありません。他のデータベースのスーパーユーザーを作成することもできます。
- 特定のデータベースアクセスパーミッションを持つロール:
 - データベースユーザー: デフォルトでログインするパーミッションがある。
 - ユーザーのグループ: グループ全体のパーミッションを管理できるようにします。

ロールはデータベースオブジェクト (テーブルや関数など) を所有することができ、SQL コマンドを使用してオブジェクト権限を他のロールに割り当てることができます。

標準のデータベース管理権限には

SELECT、**INSERT**、**UPDATE**、**DELETE**、**TRUNCATE**、**REFERENCES**、**TRIGGER**、**CREATE**、**CONNECT**、**TEMPORARY**、**EXECUTE**、および **USAGE** が含まれます。

ロール属性は、**LOGIN**、**SUPERUSER**、**CREATEDB**、および **CREATEROLE** などの特別な権限です。



重要

Red Hat は、スーパーユーザーではないロールとしてほとんどのタスクを実行することを推奨します。一般的な方法として、**CREATEDB** および **CREATEROLE** の権限を持つロールを作成し、このロールをデータベースおよびロールのすべてのルーチン管理に使用します。

関連情報

- [PostgreSQL database roles.](#)
- [PostgreSQL privileges.](#)

4.4. POSTGRESQL の設定

PostgreSQL データベースでは、データおよび設定ファイルはすべて、データベースクラスターと呼ばれる1つのディレクトリに保存されます。Red Hat は、デフォルトの `/var/lib/pgsql/data/` ディレクトリにすべてのデータを保存することを推奨します。

PostgreSQL 設定は、以下のファイルで構成されます。

- **postgresql.conf**: データベースのクラスターパラメーターの設定に使用されます。
- **postgresql.auto.conf**: **postgresql.conf** と同様の基本的な PostgreSQL 設定を保持します。ただし、このファイルはサーバーの制御下にあります。これは、**ALTER SYSTEM** クエリーにより編集され、手動で編集することはできません。
- **pg_ident.conf**: 外部認証メカニズムから PostgreSQL ユーザー ID へのユーザー ID のマッピングに使用されます。
- **pg_hba.conf**: PostgreSQL データベースのクライアント認証の設定に使用されます。

PostgreSQL 設定を変更するには、以下の手順に従います。

手順

1. 各設定ファイル (例: `/var/lib/pgsql/data/postgresql.conf`) を編集します。
2. **postgresql** サービスを再起動して、変更を有効にします。

```
# systemctl restart postgresql.service
```

例4.1 PostgreSQL データベースクラスターパラメーターの設定

以下の例では、`/var/lib/pgsql/data/postgresql.conf` ファイルのデータベースクラスターパラメーターの基本設定を示しています。

```
# This is a comment
log_connections = yes
log_destination = 'syslog'
search_path = '$user', public'
shared_buffers = 128MB
```

例4.2 PostgreSQL でのクライアント認証の設定

以下の例では、`/var/lib/pgsql/data/pg_hba.conf` ファイルでクライアント認証を設定する方法を説明します。

```
# TYPE DATABASE USER ADDRESS METHOD
local all all trust
host postgres all 192.168.93.0/24 ident
host all all .example.com scram-sha-256
```

4.5. POSTGRESQL データのバックアップ

PostgreSQL データをバックアップするには、以下のいずれかの方法を使用します。

- SQL dump: 「[SQL ダンプを使用した PostgreSQL データのバックアップ](#)」を参照してください。
- ファイルシステムレベルのバックアップ: 「[ファイルシステムレベルのバックアップを使用した PostgreSQL データのバックアップ](#)」を参照してください。
- 継続的アーカイブ: 「[継続的にアーカイブして PostgreSQL データのバックアップを作成](#)」を参照してください。

4.5.1. SQL ダンプを使用した PostgreSQL データのバックアップ

SQL ダンプのメソッドは、SQL コマンドを使用したダンプファイルの生成に基づいています。ダンプがデータベースサーバーにアップロードされると、ダンプ時と同じ状態でデータベースが再作成されます。

SQL ダンプは、以下の PostgreSQL クライアントアプリケーションによって保証されます。

- `pg_dump` は、ロールまたはテーブル空間に関するクラスター全体の情報なしに単一のデータベースをダンプします。
- `pg_dumpall` は、指定のクラスターに各データベースをダンプし、ロールやテーブル空間定義などのクラスター全体のデータを保持します。

デフォルトでは、`pg_dump` コマンドおよび `pg_dumpall` コマンドは、結果を標準出力に書き込みます。ダンプをファイルに保存するには、出力を SQL ファイルにリダイレクトします。作成される SQL ファイルは、テキスト形式またはその他の形式のいずれかになります。これにより並列処理が可能になり、オブジェクトの復元をより詳細に制御できます。

データベースにアクセスできる任意のリモートホストから、SQL ダンプを実行できます。

4.5.1.1. SQL ダンプの長所と短所

SQL ダンプには、他の PostgreSQL バックアップ方法と比較して、以下の長所があります。

- SQL ダンプは、サーバーのバージョン固有ではない唯一の PostgreSQL バックアップメソッドです。`pg_dump` ユーティリティの出力は、PostgreSQL の後続のバージョンに再読み込みできます。これは、ファイルシステムレベルのバックアップ、または継続的なアーカイブにはできません。
- SQL ダンプは、32 ビットサーバーから 64 ビットサーバーへなど、異なるアーキテクチャーにデータベースを転送する際に有効な唯一の方法です。
- SQL ダンプは、内部的に一貫性のあるダンプを提供します。ダンプは、`pg_dump` の実行開始時のデータベースのスナップショットを表します。
- `pg_dump` ユーティリティは、実行中のデータベースの他の操作をブロックしません。

SQL ダンプの短所は、ファイルシステムレベルのバックアップと比較して時間がかかることです。

4.5.1.2. pg_dump を使用した SQL ダンプの実行

クラスター全体の情報なしに単一のデータベースをダンプするには、`pg_dump` ユーティリティを使用します。

前提条件

- ダンプするすべてのテーブルへの読み取りアクセスが必要です。データベース全体をダンプするには、**postgres** のスーパーユーザーまたはデータベースの管理者権限を持つユーザーとして、コマンドを実行する必要があります。

手順

- クラスタ全体の情報なしでデータベースをダンプします。

```
$ pg_dump dbname > dumpfile
```

pg_dump が接続するデータベースサーバーを指定するには、以下のコマンドラインオプションを使用します。

- ホストを定義する **-h** オプション
デフォルトのホストは、ローカルホストか、**PGHOST** 環境変数で指定されているものです。
- ポートを定義する **-p** オプション
デフォルトのポートは、**PGPORT** 環境変数またはコンパイル済みデフォルトで示されます。

4.5.1.3. pg_dumpall を使用した SQL ダンプの実行

特定のデータベースクラスタで各データベースをダンプし、クラスタ全体のデータを保持するには、**pg_dumpall** ユーティリティを使用します。

前提条件

- **postgres** スーパーユーザーまたはデータベースの管理者権限を持つユーザーとして、コマンドを実行する必要があります。

手順

- データベースクラスタのすべてのデータベースをダンプし、クラスタ全体のデータを保存します。

```
$ pg_dumpall > dumpfile
```

pg_dumpall が接続するデータベースサーバーを指定するには、以下のコマンドラインオプションを使用します。

- ホストを定義する **-h** オプション
デフォルトのホストは、ローカルホストか、**PGHOST** 環境変数で指定されているものです。
- ポートを定義する **-p** オプション
デフォルトのポートは、**PGPORT** 環境変数またはコンパイル済みデフォルトで示されます。
- デフォルトのデータベースを定義する **-l** オプション
このオプションにより、初期化時に自動的に作成された **postgres** データベースとは異なるデフォルトのデータベースを選択できます。

4.5.1.4. pg_dump を使用したダンプされたデータベースの復元

pg_dump ユーティリティを使用してダンプした SQL ダンプからデータベースを復元するには、以下の手順に従います。

並列実行

前提条件

- **postgres** スーパーユーザーまたはデータベースの管理者権限を持つユーザーとして、コマンドを実行する必要があります。

手順

1. 新しいデータベースを作成します。

```
$ createdb dbname
```

2. ダンプされたデータベースのオブジェクトを所有するか、オブジェクトに対する権限が許可されたユーザーがすべて存在していることを確認してください。このようなユーザーが存在しない場合、復元は元の所有権と権限でオブジェクトの再作成に失敗します。
3. **psql** ユーティリティを実行して、**pg_dump** ユーティリティが作成したテキストファイルのダンプを復元します。

```
$ psql dbname < dumpfile
```

ここでの **dumpfile** は、**pg_dump** コマンドの出力になります。非テキストファイルのダンプを復元するには、代わりに **pg_restore** ユーティリティを使用します。

```
$ pg_restore non-plain-text-file
```

4.5.1.5. pg_dumpall を使用したダンプされたデータベースの復元

pg_dumpall ユーティリティを使用してダンプしたデータベースクラスターからデータを復元するには、以下の手順に従います。

前提条件

- **postgres** スーパーユーザーまたはデータベースの管理者権限を持つユーザーとして、コマンドを実行する必要があります。

手順

1. ダンプされたデータベースのオブジェクトを所有するか、オブジェクトに対する権限が許可されたユーザーがすべて、すでに存在していることを確認してください。このようなユーザーが存在しない場合、復元は元の所有権と権限でオブジェクトの再作成に失敗します。
2. **psql** ユーティリティを実行して、**pg_dumpall** ユーティリティにより作成されたテキストファイルのダンプを復元します。

```
$ psql < dumpfile
```

ここでの **dumpfile** は、**pg_dumpall** コマンドの出力になります。

4.5.1.6. 別のサーバーでのデータベースの SQL ダンプの実行

pg_dump および **psql** はパイプに対する読み書きが可能であるため、あるサーバーから別のサーバーにデータベースを直接ダンプできます。

手順

- データベースを、サーバーから別のサーバーにダンプするには、以下のコマンドを実行します。

```
$ pg_dump -h host1 dbname | psql -h host2 dbname
```

4.5.1.7. 復元中の SQL エラーの処理

デフォルトでは、SQL エラーが発生した場合、**psql** は実行を継続します。これにより、データベースの復元は一部のみとなります。

デフォルトの動作を変更するには、ダンプを復元する際に以下のいずれかの方法を使用します。

前提条件

- **postgres** スーパーユーザーまたはデータベースの管理者権限を持つユーザーとして、コマンドを実行する必要があります。

手順

- **ON_ERROR_STOP** 変数を設定して SQL エラーが発生した場合は、終了ステータスが 3 で **psql** を終了します。

```
$ psql --set ON_ERROR_STOP=on dbname < dumpfile
```

- ダンプ全体が単一のトランザクションとして復元されるように指定して、復元が完全に完了するかキャンセルされるようにします。
 - **psql** ユーティリティーを使用してテキストファイルのダンプを復元する場合:

```
$ psql -1
```

- **pg_restore** ユーティリティーを使用してテキストファイル以外のダンプを復元する場合:

```
$ pg_restore -e
```

この方法を使用する場合は、多少のエラーでも、すでに何時間も実行している復元操作をキャンセルできます。

1. 関連情報

- [PostgreSQL Documentation - SQL dump](#).

4.5.2. ファイルシステムレベルのバックアップを使用した PostgreSQL データのバックアップ

ファイルシステムレベルのバックアップを実行するには、PostgreSQL データベースファイルを別の場所にコピーします。たとえば、以下のいずれかの方法を使用できます。

- **tar** ユーティリティーを使用してアーカイブファイルを作成します。
- **rsync** ユーティリティーを使用して、ファイルを別の場所にコピーします。
- データディレクトリの一貫したスナップショットを作成します。

4.5.2.1. ファイルシステムレベルのバックアップの長所と短所

ファイルシステムレベルのバックアップは、他の PostgreSQL バックアップ方法と比較して、以下の長所があります。

- ファイルシステムレベルのバックアップは通常、SQL ダンプよりも高速です。

ファイルシステムレベルのバックアップは、他の PostgreSQL バックアップ方法と比較して、以下の短所があります。

- このバックアップメソッドは、RHEL 8 から RHEL 9 にアップグレードし、アップグレードしたシステムにデータを移行する場合には適していません。ファイルシステムレベルのバックアップは、アーキテクチャー固有および RHEL のメジャーバージョン固有のもので、アップグレードに成功しなかった場合は、RHEL 8 システムでデータを復元できますが、RHEL 9 システムではデータを復元できません。
- データバックアップまたはデータ復元を行う前に、データベースサーバーをシャットダウンする必要があります。
- 特定の個別ファイルまたはテーブルのバックアップと復元はできません。ファイルシステムのバックアップは、データベースクラスター全体の完全バックアップと復元でのみ機能します。

4.5.2.2. ファイルシステムレベルのバックアップの実行

ファイルシステムレベルのバックアップを実行するには、以下の手順を行います。

手順

1. データベースクラスターの場所を選択し、このクラスターを初期化します。

```
# postgresql-setup --initdb
```

2. postgresql サービスを停止します。

```
# systemctl stop postgresql.service
```

3. 任意のメソッドを使用してファイルシステムのバックアップを作成します (例: **tar** アーカイブ)。

```
$ tar -cf backup.tar /var/lib/pgsql/data
```

4. postgresql サービスを起動します。

```
# systemctl start postgresql.service
```

関連情報

- [PostgreSQL Documentation - file system level backup](#) .

4.5.3. 継続的にアーカイブして PostgreSQL データのバックアップを作成

4.5.3.1. 継続的なアーカイブの概要

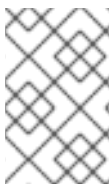
PostgreSQL は、データベースのデータファイルに対するすべての変更を、クラスタのデータディレクトリーの `pg_wal/` サブディレクトリーで利用可能なログ先行書き込み (WAL) ファイルに記録します。このログは、主にクラッシュからの復元を目的としています。クラッシュ後、最後のチェックポイント以降に作成されたログエントリーを使用して、データベースの整合性まで復元できます。

オンラインバックアップとも呼ばれる継続的なアーカイブメソッドは、WAL ファイルを、稼働中のサーバーまたはファイルシステムレベルのバックアップで実行されるベースバックアップの形式でデータベースクラスタのコピーと組み合わせます。

データベース復元が必要な場合は、データベースクラスタのコピーからデータベースを復元してから、バックアップを作成した WAL ファイルからログを再生して、システムを現在の状態にすることができます。

継続的なアーカイブメソッドでは、少なくとも最後のベースバックアップの開始時間までさかのぼって、アーカイブされたすべての WAL ファイルの連続したシーケンスを保持する必要があります。したがって、ベースバックアップの理想的な頻度は、以下に依存します。

- アーカイブされた WAL ファイルで利用可能なストレージボリューム。
- 復元が必要な場合の、データ復元の最大許容期間。最後のバックアップからの期間が長い場合、システムはより多くの WAL セグメントを再生するため、回復に時間がかかります。



注記

`pg_dump` および `pg_dumpall` SQL ダンプは、継続的にアーカイブするバックアップソリューションの一部として使用することができません。SQL ダンプは論理バックアップを生成しますが、WAL 再生で使用する上で十分な情報は含まれていません。

継続的なアーカイブメソッドを使用してデータベースのバックアップと復元を実行するには、以下の手順に従います。

1. WAL ファイルのアーカイブ手順の設定およびテスト: 「[WAL アーカイブの設定](#)」を参照してください。
2. ベースバックアップの実行: 「[ベースバックアップの作成](#)」を参照してください。

データを復元するには、「[継続的なアーカイブバックアップを使用したデータベースの復元](#)」の手順に従います。

4.5.3.2. 継続的なアーカイブの長所と短所

継続的なアーカイブは、PostgreSQL のその他のバックアップ方法と比較して、以下の利点があります。

- 継続的なバックアップメソッドでは、バックアップ内の内部不整合がログ再生により修正されるため、整合性が完全に取れないベースバックアップを使用することができます。したがって、実行中の PostgreSQL サーバーでベースバックアップを実行できます。
- ファイルシステムのスナップショットは必要ありません。tar または同様のアーカイブユーティリティーで十分です。
- 継続的にバックアップを行うには、継続的に WAL ファイルをアーカイブします。これは、ログ再生用の WAL ファイルの順序が無限に長くなる可能性があるためです。これは、特に大規模なデータベースで有用です。

- 継続的バックアップは、特定の時点への復旧 (ポイントインタイムリカバリー) をサポートします。WAL エントリーを最後まで再生する必要はありません。再生はいつでも停止でき、ベースバックアップを作成してから、データベースをいつでもその状態に復元できます。
- 一連の WAL ファイルが同じベースのバックアップファイルで読み込まれた別のマシンが継続的に利用可能である場合は、任意の時点で、データベースで現在が一番近いコピーで、他のマシンを復元できます。

継続的なアーカイブには、その他の PostgreSQL バックアップ方法と比較して、以下の短所があります。

- 継続バックアップ方法は、サブセットではなく、データベースクラスター全体の復元のみをサポートします。
- 継続的にバックアップするには、大きなアーカイブストレージが必要です。

4.5.3.3. WAL アーカイブの設定

稼働中の PostgreSQL サーバーでは、ログ先行書き込み (WAL) レコードのシーケンスを生成します。サーバーは、このシーケンスを WAL セグメントファイルに物理的に分割します。このファイルには、WAL シーケンスの位置を反映する数値名が与えられます。WAL のアーカイブを使用しない場合は、セグメントファイルは再利用され、より大きなセグメント番号に名前が変更されます。

WAL データをアーカイブする場合、各セグメントファイルの内容がキャプチャーされ、新しい場所に保存されてから、セグメントファイルが再利用されます。別のマシン上の NFS マウントディレクトリ、テープドライブ、または CD など、コンテンツの保存場所には複数のオプションがあります。

WAL レコードには、設定ファイルへの変更が含まれていないことに注意してください。

WAL のアーカイブを有効にするには、以下の手順に従います。

手順

1. `/var/lib/pgsql/data/postgresql.conf` ファイルで以下を行います。
 - a. `wal_level` 設定パラメーターを `replica` 以降に設定します。
 - b. `archive_mode` パラメーターを `on` に設定します。
 - c. `archive_command` 設定パラメーターでシェルコマンドを指定します。 `cp` コマンド、別のコマンド、またはシェルスクリプトを使用できます。
2. `postgresql` サービスを再起動して、変更を適用します。

```
# systemctl restart postgresql.service
```

3. アーカイブコマンドをテストして、既存のファイルを上書きせず、失敗するとゼロ以外の終了ステータスを返すことを確認します。
4. データを保護するには、セグメントファイルがグループまたはワールド読み取りアクセスを持たないディレクトリにアーカイブされていることを確認してください。



注記

archive コマンドは、完了した WAL セグメントでのみ実行されます。WAL トラフィックをほとんど生成しないサーバーでは、トランザクションの完了とアーカイブストレージへの安全な記録の間にかかなりの遅延が生じる可能性があります。アーカイブされていないデータの古さを制限するには、以下を行います。

- **archive_timeout** パラメーターを設定して、サーバーが特定の頻度で新しい WAL セグメントファイルに切り替えるように強制します。
- **pg_switch_wal** パラメーターを使用して、セグメント切り替えを強制し、トランザクションが終了後すぐにアーカイブされるようにします。

例4.3 WAL セグメントをアーカイブするためのシェルコマンド

この例は、**archive_command** 設定パラメーターに設定できる簡単なシェルコマンドを示しています。

以下のコマンドは、完了したセグメントファイルを必要な場所にコピーします。

```
archive_command = 'test ! -f /mnt/server/archivedir/%f && cp %p /mnt/server/archivedir/%f'
```

%p パラメーターは、アーカイブするファイルの相対パスに置き換えられ、**%f** パラメーターはファイル名に置き換えられます。

このコマンドは、アーカイブ可能な WAL セグメントを **/mnt/server/archivedir/** ディレクトリーにコピーします。**%p** パラメーターおよび **%f** パラメーターを置き換えると、実行されたコマンドは以下ようになります。

```
test ! -f /mnt/server/archivedir/00000001000000A9000000065 && cp
pg_wal/00000001000000A9000000065 /mnt/server/archivedir/00000001000000A9000000065
```

アーカイブされる新規ファイルごとに同様のコマンドが生成されます。

関連情報

- WAL アーカイブの設定に関する詳細は、[PostgreSQL 13 ドキュメント](#) を参照してください。

4.5.3.4. ベースバックアップの作成

ベースバックアップは、複数の方法で作成できます。本セクションでは、実行中の PostgreSQL サーバーで **pg_basebackup** ユーティリティーを使用してベースバックアップを実行する最も簡単な方法を説明します。

ベースバックアッププロセスは、WAL アーカイブ領域に保存され、ベースバックアップに必要な最初の WAL セグメントファイルにちなんで名付けられたバックアップ履歴ファイルを作成します。

バックアップ履歴ファイルは、開始時間および終了時間、およびバックアップの WAL セグメントが含まれる小さなテキストファイルです。ラベル文字列を使用して関連するダンプファイルを特定した場合は、バックアップ履歴ファイルを使用して復元するダンプファイルを判断できます。



注記

データを確実に復元できるようにするために、複数のバックアップセットを維持することを検討してください。

ベースバックアップを実行するには、以下の手順を行います。

前提条件

- **postgres** スーパーユーザー、データベースの管理者権限のあるユーザー、または少なくとも **REPLICATION** パーミッションを持つ別のユーザーとして、コマンドを実行する必要があります。
- ベースバックアップ中およびベースバックアップ後に生成されたすべての WAL セグメントファイルを保持する必要があります。

手順

1. **pg_basebackup** ユーティリティを使用してベースバックアップを実行します。

- 個々のファイル (プレーン形式) としてベースバックアップを作成するには、以下を実行します。

```
$ pg_basebackup -D backup_directory -Fp
```

backup_directory を希望のバックアップの場所に置き換えます。

テーブル空間を使用し、サーバーと同じホストでベースバックアップを実行する場合は、**--tablespace-mapping** オプションも使用する必要があります。そうしないと、バックアップを同じ場所に書き込もうとすると、バックアップが失敗します。

- **tar** アーカイブ (**tar** および圧縮形式) としてベースバックアップを作成するには、以下を実行します。

```
$ pg_basebackup -D backup_directory -Ft -z
```

backup_directory を希望のバックアップの場所に置き換えます。

このようなデータを復元するには、ファイルを正しい場所に手動で抽出する必要があります。

2. ベースバックアッププロセスが完了すると、バックアップ履歴ファイルで指定されている、データベースクラスタのコピーとバックアップ中に使用された WAL セグメントファイルを安全にアーカイブします。
3. ベースバックアップで使用されている WAL セグメントファイルよりも数値が小さい WAL セグメントを削除します。これらはベースバックアップよりも古く、復元には必要ないためです。

pg_basebackup が接続するデータベースサーバーを指定するには、以下のコマンドラインオプションを使用します。

- ホストを定義する **-h** オプション
デフォルトのホストは、ローカルホストまたは **PGHOST** 環境変数により指定されたホストです。
- ポートを定義する **-p** オプション

デフォルトのポートは、**PGPORT** 環境変数またはコンパイル済みデフォルトで示されます。

関連情報

- [PostgreSQL ドキュメント - base backup](#)
- [PostgreSQL ドキュメント - pg_basebackup ユーティリティー](#)

4.5.3.5. 継続的なアーカイブバックアップを使用したデータベースの復元

継続バックアップを使用してデータベースを復元するには、以下の手順を行います。

手順

1. サーバーを停止します。

```
# systemctl stop postgresql.service
```

2. 必要なデータを一時的な場所にコピーします。

必要に応じて、クラスターデータのディレクトリー全体と、すべてのテーブル空間をコピーします。既存データベースのコピーを2つ保持するには、システムに十分な空き領域が必要になることに注意してください。

十分な容量がない場合は、クラスターの **pg_wal** ディレクトリーの内容を保存します。これには、システムがダウンする前にアーカイブされなかったログが含まれます。

3. クラスターデータディレクトリー、および使用しているテーブル空間のルートディレクトリー下の既存ファイルおよびサブディレクトリーをすべて削除します。
4. ベースバックアップからデータベースファイルを復元します。
以下の点を確認してください。

- ファイルは、正しい所有権 (**root** ではなくデータベースシステムのユーザー) で復元されます。
- ファイルは、正しい権限で復元されます。
- **pg_tblspc/** サブディレクトリーのシンボリックリンクが正しく復元されます。

5. **pg_wal/** サブディレクトリーにあるファイルをすべて削除します。
このファイルは、ベースバックアップから作成されるため、非推奨になりました。**pg_wal/** をアーカイブしていない場合は、適切な権限で再作成します。
6. 手順2で保存したアーカイブされていない WAL セグメントファイルを **pg_wal/** にコピーします。
7. クラスターデータディレクトリーの **recovery.conf** リカバリーコマンドファイルを作成し、**restore_command** 設定パラメーターにシェルコマンドを指定します。**cp** コマンド、別のコマンド、またはシェルスクリプトを使用できます。以下に例を示します。

```
restore_command = 'cp /mnt/server/archivedir/%f "%p"'
```

8. サーバーを起動します。

```
# systemctl start postgresql.service
```

サーバーは復元モードに入り、引き続き必要なアーカイブファイル (WAL) を読み込みます。

外部エラーにより復元が終了した場合は、サーバーを再起動して復元を続行します。復元プロセスが完了すると、サーバーは **recovery.conf** の名前を **recovery.done** に変更します。これにより、サーバーが通常のデータベース操作を開始した後に、誤ってリカバリモードに戻るのを防ぐことができます。

9. データベースのコンテンツを確認して、データベースが必要な状態に復元されたことを確認します。
データベースが必要な状態に復元されていない場合は、手順1に戻ります。データベースが必要な状態に復元された場合は、**pg_hba.conf** ファイルでクライアント認証設定を復元して接続できるようにします。

継続バックアップを使用した復元の詳細は、[PostgreSQL ドキュメンテーション](#) を参照してください。

関連情報

- [PostgreSQL ドキュメント - continuous archiving method](#)

4.6. PostgreSQL の RHEL9 バージョンへの移行

Red Hat Enterprise Linux 8 は、複数のモジュールストリームに PostgreSQL を提供します。PostgreSQL 10 (デフォルトの postgresql ストリーム)、PostgreSQL 9.6、PostgreSQL 12、および PostgreSQL 13RHEL 9 では、PostgreSQL 13 が利用できます。

Red Hat Enterprise Linux 上の PostgreSQL のユーザーは、データベースファイルの移行パスを使用できます。

- [pg_upgrade ユーティリティーを使用した高速アップグレード](#)
- [ダンプおよび復元のアップグレード](#)

高速アップグレードメソッドは、ダンプおよび復元のプロセスよりも速くなります。ただし、場合によっては高速アップグレードが機能せず、たとえばアーキテクチャー間のアップグレードなど、ダンプおよび復元プロセスのみを使用できます。

新しいバージョンの PostgreSQL に移行するための前提条件として、すべての PostgreSQL データベースをバックアップします。

データベースをダンプし、SQL ファイルのバックアップを実行することは、ダンプおよび復元プロセスで必要であり、高速アップグレードメソッドとして推奨されます。

新しいバージョンの PostgreSQL に移行する前に、移行する PostgreSQL バージョンと、移行元と移行先のバージョンの間にあるすべて PostgreSQL バージョンの [アップストリームの互換性ノート](#) を参照してください。

4.6.1. pg_upgrade ユーティリティーを使用した高速アップグレード

高速アップグレードでは、バイナリーデータファイルを `/var/lib/pgsql/data/` ディレクトリーにコピーし、**pg_upgrade** ユーティリティーを使用する必要があります。

この方法を使用して、RHEL 8 バージョンの PostgreSQL 12 から、RHEL 9 バージョンの PostgreSQL 13 にデータを移行できます。

以下の手順では、高速アップグレードメソッドを使用して、RHEL 8 バージョンの PostgreSQL 12 から、RHEL 9 バージョンの PostgreSQL 13 への移行方法を説明します。12 以外の `postgresql` ストリームからの移行には、以下のいずれかの方法を使用します。

- RHEL 8 で PostgreSQL サーバーをバージョン 12 に更新し、`pg_upgrade` ユーティリティーを使用して RHEL 9 バージョンの PostgreSQL 13 への高速アップグレードを実行します。詳細は、[Migrating to a RHEL 9 version of PostgreSQL](#) を参照してください。
- RHEL 8 バージョンの PostgreSQL と RHEL 9 の PGPostgreSQL 13 との間で直接、ダンプおよび復元のアップグレードを使用します。

前提条件

- アップグレードを実行する前に、PostgreSQL データベースに保存されているすべてのデータのバックアップを作成します。デフォルトでは、すべてのデータは、RHEL 8 および RHEL 9 システムの両方の `/var/lib/pgsql/data/` ディレクトリーに保存されます。

手順

1. RHEL 9 システムで、`postgresql-server` パッケージおよび `postgresql-upgrade` パッケージをインストールします。

```
# dnf install postgresql-server postgresql-upgrade
```

必要に応じて、RHEL 8 で PostgreSQL サーバーモジュールを使用していた場合は、それらを 2 つのバージョンで RHEL 9 システムにもインストールし、PostgreSQL 12 (`postgresql-upgrade` パッケージでインストール) および対象バージョンの PostgreSQL 13 (`postgresql-server` パッケージでインストール) の両方に対してコンパイルします。サードパーティーの PostgreSQL サーバーモジュールをコンパイルする必要がある場合は、`postgresql-devel` パッケージと `postgresql-upgrade-devel` パッケージの両方に対してビルドしてください。

2. 以下の項目を確認します。
 - 基本設定:RHEL 9 システムで、サーバーがデフォルトの `/var/lib/pgsql/data` ディレクトリーを使用し、データベースが正しく初期化され、有効になっているかどうかを確認します。さらに、データファイルは、`/usr/lib/systemd/system/postgresql.service` ファイルに記載されているパスと同じパスに保存する必要があります。
 - PostgreSQL サーバー:システムは複数の PostgreSQL サーバーを実行できます。これらのすべてのサーバーのデータディレクトリーが独立して処理されていることを確認してください。
 - PostgreSQL サーバーモジュール:RHEL 8 で使用されていた PostgreSQL サーバーモジュールも、RHEL 9 システムにインストールされていることを確認してください。プラグインは `/usr/lib64/pgsql/` ディレクトリーにインストールされていることに注意してください。
3. データのコピー時に、`postgresql` サービスがソースおよびターゲットのシステムで稼働していないことを確認します。

```
# systemctl stop postgresql.service
```

4. データベースファイルをソースの場所から RHEL 9 システムの `/var/lib/pgsql/data/` ディレクトリーにコピーします。
5. PostgreSQL ユーザーで以下のコマンドを実行して、アップグレードプロセスを実行します。


```
# postgresql-setup --upgrade
```

これでバックグラウンドで **pg_upgrade** プロセスが開始します。

障害が発生すると、**postgresql-setup** は通知のエラーメッセージを提供します。

6. **/var/lib/pgsql/data-old** から新規クラスターに、以前の設定をコピーします。
高速アップグレードは、新しいデータスタックで以前の設定を再利用せず、設定がゼロから生成されることに注意してください。古い設定と新しい設定を手動で組み合わせたい場合は、データディレクトリーの *.conf ファイルを使用します。
7. 新しい PostgreSQL サーバーを起動します。

```
# systemctl start postgresql.service
```

8. PostgreSQL ホームディレクトリーにある **analyze_new_cluster.sh** スクリプトを実行します。

```
su postgres -c '~/analyze_new_cluster.sh'
```

9. システムの起動時に、新しい PostgreSQL サーバーを自動的に起動させる場合は、次のコマンドを実行します。

```
# systemctl enable postgresql.service
```

4.6.2. ダンプおよび復元のアップグレード

ダンプおよび復元のアップグレードを使用する場合は、すべてのデータベースのコンテンツを SQL ファイルのダンプファイルにダンプする必要があります。ダンプおよび復元のアップグレードは高速なアップグレード方法よりも低速であり、生成された SQL ファイルで手動修正が必要になる場合があります。

この方法を使用して、任意の RHEL 8 バージョンの PostgreSQL から、RHEL 9 バージョンの PostgreSQL 13 にデータを移行できます。

RHEL 8 および RHEL 9 システムでは、PostgreSQL データは、デフォルトで **/var/lib/pgsql/data/** ディレクトリーに保存されます。

ダンプおよび復元のアップグレードを実行するには、ユーザーを **root** に変更します。

以下の手順では、RHEL 8 デフォルトバージョンの PostgreSQL 10 から、RHEL 9 バージョンの PostgreSQL 13 への移行を説明します。

手順

1. RHEL 8 システムで PostgreSQL 10 サーバーを起動します。

```
# systemctl start postgresql.service
```

2. RHEL 8 システムで、すべてのデータベースのコンテンツを **pgdump_file.sql** ファイルにダンプします。

```
su - postgres -c "pg_dumpall > ~/pgdump_file.sql"
```

3. データベースが正しくダンプされたことを確認します。

```
su - postgres -c 'less "$HOME/pgdump_file.sql"'
```

これにより、ダンプされた sql ファイルのパスが `/var/lib/pgsql/pgdump_file.sql` に表示されます。

4. RHEL 9 システムで、**postgresql-server** パッケージをインストールします。

```
# dnf install postgresql-server
```

必要に応じて、RHEL 8 で PostgreSQL サーバーモジュールを使用していた場合は、RHEL 9 システムにもインストールしてください。サードパーティーの PostgreSQL サーバーモジュールをコンパイルする必要がある場合は、**postgresql-devel** パッケージに対してビルドします。

5. RHEL 9 システムで、新しい PostgreSQL サーバーのデータディレクトリーを初期化します。

```
# postgresql-setup --initdb
```

6. RHEL 9 システムで、**pgdump_file.sql** を PostgreSQL ホームディレクトリーにコピーし、ファイルが正しくコピーされたことを確認します。

```
su - postgres -c 'test -e "$HOME/pgdump_file.sql" && echo exists'
```

7. RHEL 8 システムから設定ファイルをコピーします。

```
su - postgres -c 'ls -l $PGDATA/*.conf'
```

コピーされる設定ファイルは、以下のとおりです。

- `/var/lib/pgsql/data/pg_hba.conf`
- `/var/lib/pgsql/data/pg_ident.conf`
- `/var/lib/pgsql/data/postgresql.conf`

8. RHEL 9 システムで、新しい PostgreSQL サーバーを起動します。

```
# systemctl start postgresql.service
```

9. RHEL 9 システムで、ダンプされた sql ファイルからデータをインポートします。

```
su - postgres -c 'psql -f ~/pgdump_file.sql postgres'
```