Red Hat Enterprise Linux 9

配置和使用数据库服务器

在数据库服务器上安装、配置、备份和迁移数据

Last Updated: 2024-06-26

Red Hat Enterprise Linux 9 配置和使用数据库服务器

在数据库服务器上安装、配置、备份和迁移数据

法律通告

摘要

在 Red Hat Enterprise Linux 9 上安装 MariaDB、MySQL 或 PostgreSQL 数据库服务器。配置所选的数据库服务器，备份您的数据，并将数据迁移到数据库服务器的后续版本。

目录

对红帽文档提供反馈

我们感谢您对我们文档的反馈。让我们了解如何改进它。

通过 Jira 提交反馈（需要帐户）

1. 登录到 Jira 网站。
2. 在顶部导航栏中点 **Create**
3. 在 **Summary** 字段中输入描述性标题。
4. 在 **Description** 字段中输入您的改进建议。包括到文档相关部分的链接。
5. 点对话框底部的 **Create**。

第 1 章 介绍

数据库服务器是一种提供数据库管理系统(DBMS)功能的服务。DBMS 为数据库管理提供工具，并与最终用户、应用程序和数据库进行交互。

Red Hat Enterprise Linux 9 提供以下数据库管理系统：

- **MariaDB 10.5**
- **MariaDB 10.11** - 从 RHEL 9.4 开始可用
- **MySQL 8.0**
- **PostgreSQL 13**
- **PostgreSQL 15** - 从 RHEL 9.2 开始提供
- **PostgreSQL 16** - 从 RHEL 9.4 开始提供
- **Redis 6**

第 2 章 使用 MARIADB

MariaDB 服务器是一个基于 MySQL 技术的开源、快速、强大的数据库服务器。MariaDB 是一个关系型数据库，它将数据转换为结构化信息，并为访问数据提供 SQL 接口。它包含多个存储引擎和插件，以及地理信息系统(GIS)和 JavaScript 对象表示法(JSON)功能。

了解如何在 RHEL 系统上安装和配置 MariaDB，如何备份 MariaDB 数据、如何从早期的 MariaDB 版本迁移，以及如何使用 MariaDB Galera 集群复制数据库。

2.1. 安装 MARIADB

RHEL 9.0 提供 MariaDB 10.5 作为此 Application Stream 的初始版本，可作为 RPM 软件包轻松安装。在 RHEL 9 的次版本中，其他 MariaDB 版本作为模块提供较短的生命周期。

RHEL 9.4 引入了 MariaDB 10.11 作为 **mariadb:10.11** 模块流。

注意

按照设计，无法并行安装同一模块的多个版本(stream)。因此，您必须从 **mariadb** 模块中只选择一个可用流。您可以在容器中使用不同版本的 MariaDB 数据库服务器，请参阅 在容器中运行多个 MariaDB 版本。

由于 RPM 软件包有冲突，所以在 RHEL 9 中无法并行安装 MariaDB 和 MySQL 数据库服务器。您可以在容器中并行使用 MariaDB 和 MySQL 数据库服务器，请参阅 在容器中运行多个 MySQL 和 MariaDB 版本。

要安装 MariaDB，请使用以下流程：

流程

1. 安装 MariaDB 服务器软件包：
   a. 对于 RPM 软件包中的 MariaDB 10.5 ：

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

   b. 对于 MariaDB 10.11，从 **mariadb** 模块中选择流（版本） **11** 并指定服务器配置文件，例如：

```
# dnf module install mariadb:10.11/server
```

2. 启动 **mariadb** 服务：

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

3. 启用 **mariadb** 服务，使其在引导时启动：

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

2.1.1. 在容器中运行多个 MariaDB 版本

要在同一主机上运行不同版本的 MariaDB，请在容器中运行它们，因为您无法并行安装同一模块的多个版本(streams)。

先决条件

- **container-tools** 元数据包已安装。

流程

1. 使用您的红帽客户门户网站帐户向 **registry.redhat.io** registry 进行身份验证：

```
# podman login registry.redhat.io
```

如果您已登录到容器 registry，请跳过这一步。

2. 在容器中运行 MariaDB 10.5 ：

```
$ podman run -d --name <container_name> -e
MYSQL_ROOT_PASSWORD=<mariadb_root_password> -p <host_port_1>:3306
rhel9/mariadb-105
```

有关使用此容器镜像的更多信息，请参阅 红帽生态系统目录。

3. 在容器中运行 MariaDB 10.11 ：

```
$ podman run -d --name <container_name> -e
MYSQL_ROOT_PASSWORD=<mariadb_root_password> -p <host_port_2>:3306
rhel9/mariadb-1011
```

有关使用此容器镜像的更多信息，请参阅 红帽生态系统目录。

注意

两个数据库服务器的容器名称和主机端口必须有所不同。

4. 要确保客户端可以访问网络中的数据库服务器，请在防火墙中打开主机端口：

```
# firewall-cmd --permanent --add-port={<host_port_1>/tcp,<host_port_2>/tcp,...}
# firewall-cmd --reload
```

验证步骤

1. 显示正在运行的容器信息：

```
$ podman ps
```

2. 连接到数据库服务器，并以 root 用户身份登录：

```
# mysql -u root -p -h localhost -P <host_port> --protocol tcp
```

其他资源

- 构建、运行和管理容器
- 浏览红帽生态系统目录中的容器

2.2. 配置 MARIADB

要为联网配置 MariaDB 服务器，请使用以下流程：

流程

1. 编辑**/etc/my.cnf.d/mariadb-server.cnf**文件的**[mysqld]**部分。您可以设置以下配置指令：
   - **bind-address** - 是服务器监听的地址。可能的选项有：
     - 主机名
     - IPv4 地址
     - IPv6 地址
   - **skip-networking** - 控制服务器是否监听 TCP/IP 连接。可能的值有：
     - 0 - 监听所有客户端
     - 1 - 只监听本地客户端
   - **port** - MariaDB 监听 TCP/IP 连接的端口。
2. 重启 **mariadb** 服务：

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

2.3. 在 MARIADB 服务器上设置 TLS 加密

默认情况下，MariaDB 使用未加密的连接。对于安全连接，在 MariaDB 服务器上启用 TLS 支持，并将您的客户端配置为建立加密连接。

2.3.1. 将 CA 证书、服务器证书和私钥放在 MariaDB 服务器上

在 MariaDB 服务器中启用 TLS 加密前，先在 MariaDB 服务器上存储证书颁发机构(CA)证书、服务器证书和私钥。

先决条件

- 以下 Privacy Enhanced Mail(PEM)格式的文件已复制到服务器：
  - 服务器的私钥：**server.example.com.key.pem**
  - 服务器证书：**server.example.com.crt.pem**
  - 证书颁发机构(CA)证书：**ca.crt.pem**

  有关创建私钥和证书签名请求(CSR)，以及从 CA 请求证书的详情，请查看您的 CA 文档。

流程

1. 将 CA 和服务器证书存储在 **/etc/pki/tls/certs/** 目录中：

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

2. 设置 CA 和服务器证书的权限，使 MariaDB 服务器能够读取文件：

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

由于证书是建立安全连接前通信的一部分，因此任何客户端都可以在不需要身份验证的情况下检索它们。因此，您不需要对 CA 和服务器证书文件设置严格的权限。

3. 将服务器的私钥存储在 **/etc/pki/tls/private/** 目录中：

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

4. 对服务器的私钥设置安全权限：

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

如果未授权的用户可以访问私钥，因此到 MariaDB 服务器的连接不再是安全的。

5. 恢复 SELinux 上下文：

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

2.3.2. 在 MariaDB 服务器上配置 TLS

要提高安全性，请在 MariaDB 服务器上启用 TLS 支持。因此，客户端可以使用 TLS 加密与服务器传输数据。

先决条件

- MariaDB 服务器已安装。
- **mariadb** 服务正在运行。
- 服务器上存在 Privacy Enhanced Mail(PEM)格式的以下文件，并可由 **mysql** 用户读取：
  - 服务器的私钥：**/etc/pki/tls/private/server.example.com.key.pem**
  - 服务器证书：**/etc/pki/tls/certs/server.example.com.crt.pem**
  - 证书颁发机构(CA)证书 **/etc/pki/tls/certs/ca.crt.pem**
- 主题可识别名称(DN)或服务器证书中的主题备用名称(SAN)字段与服务器的主机名相匹配。

流程

1. 创建 **/etc/my.cnf.d/mariadb-server-tls.cnf** 文件：
   a. 添加以下内容来配置到私钥、服务器和 CA 证书的路径：

```
[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. 如果您有一个证书撤销列表(CRL)，则将 MariaDB 服务器配置为使用它：

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

c. 可选：拒绝未加密的连接尝试。要启用此功能，请附加：

```
require_secure_transport = on
```

d. 可选：设置服务器应支持的 TLS 版本。例如，要支持 TLS 1.2 和 TLS 1.3，请附加：

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

默认情况下，服务器支持 TLS 1.1、TLS 1.2 和 TLS 1.3。

2. 重启 **mariadb** 服务：

```
# systemctl restart mariadb
```

验证

要简化故障排除，请在将本地客户端配置为使用 TLS 加密之前在 MariaDB 服务器上执行以下步骤：

1. 验证 MariaDB 现在是否启用了 TLS 加密：

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

如果 **have_ssl** 变量设置为 **yes**，则启用 TLS 加密。

2. 如果您将 MariaDB 服务配置为只支持特定的 TLS 版本，则显示 **tls_version** 变量：

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

其他资源

- 将 CA 证书、服务器证书和私钥放在 MariaDB 服务器上

2.3.3. 对特定的用户帐户需要 TLS 加密连接

可以访问敏感数据的用户应始终使用 TLS 加密连接，以避免通过网络发送未加密的数据。

如果您无法在服务器上配置所有连接都需要安全传输(**require_secure_transport = on**)，请将单个用户帐户配置为需要 TLS 加密。

先决条件

- MariaDB 服务器启用了 TLS 支持。
- 您配置为需要安全传输的用户已存在。

流程

1. 以管理员用户身份连接到 MariaDB 服务器：

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

   如果您的管理用户没有远程访问服务器的权限，请在 MariaDB 服务器上执行命令，并连接到 **localhost**。

2. 使用 **REQUIRE SSL** 子句强制用户必须使用 TLS 加密连接进行连接：

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

验证

1. 使用 TLS 加密，以 **example** 用户身份连接到服务器：

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

   如果没有显示错误，且您可以访问交互式 MariaDB 控制台，则与 TLS 的连接成功。

2. 尝试以禁用 TLS 的 **example** 用户身份进行连接：

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

   服务器拒绝登录尝试，因为此用户需要 TLS，但被禁用了(**--skip-ssl**)。

其他资源

- 在 MariaDB 服务器上设置 TLS 加密

2.4. 在 MARIADB 客户端中全局启用 TLS 加密

如果您的 MariaDB 服务器支持 TLS 加密，请将客户端配置为仅建立安全连接，并验证服务器证书。这个流程描述了如何为服务器上的所有用户启用 TLS 支持。

2.4.1. 将 MariaDB 客户端配置为默认使用 TLS 加密

在 RHEL 上，您可以全局配置 MariaDB 客户端使用 TLS 加密，并验证服务器证书中的通用名称(CN)与用户连接的主机名匹配。这可防止中间人攻击。

先决条件

- MariaDB 服务器启用了 TLS 支持。
- 如果 RHEL 不信任发布服务器证书的证书颁发机构(CA)，则 CA 证书已被复制到客户端。
- 如果 MariaDB 服务器运行 RHEL 9.2 或更高版本，并且启用了 FIPS 模式，则这个客户端支持 Extended Master Secret(EMS)扩展或使用 TLS 1.3。没有 EMS 的 TLS 1.2 连接会失败。如需更多信息，请参阅 强制 TLS 扩展"Extended Master Secret" 知识库文章。

流程

1. 如果 RHEL 不信任发布服务器证书的 CA：
   a. 将 CA 证书复制到 **/etc/pki/ca-trust/source/anchors/** 目录中：

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

   b. 设置允许所有用户读取 CA 证书文件的权限：

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

   c. 重建 CA 信任数据库：

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

2. 使用以下内容创建 **/etc/my.cnf.d/mariadb-client-tls.cnf** 文件：

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

   这些设置定义 MariaDB 客户端使用 TLS 加密(**ssl**)，并且客户端将主机名与服务器证书中的 CN(**ssl-verify-server-cert**)进行比较。

验证

- 使用主机名连接到服务器，并显示服务器的状态：

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

  如果 **SSL** 条目包含 **Cipher in use is...**，代表连接已加密。

  请注意，您在这个命令中使用的用户具有远程身份验证的权限。

  如果您连接的主机名与服务器的 TLS 证书中的主机名不匹配，则 **ssl-verify-server-cert** 参数会导致连接失败。例如，如果您连接到 **localhost** ：

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

其他资源

- **mysql(1)** 手册页中的 **--ssl*** 参数描述。

2.5. 备份 MARIADB 数据

在 Red Hat Enterprise Linux 9 中从 **MariaDB** 数据库备份数据主要有两种方法：

- 逻辑备份
- 物理备份

逻辑备份 由恢复数据所需的 SQL 语句组成。这种类型的备份以纯文本文件的形式导出信息和记录。

与物理备份相比，逻辑备份的主要优势在于可移植性和灵活性。数据可以在其他硬件配置上恢复，**MariaDB** 版本或数据库管理系统(DBMS)上恢复，这些系统无法进行物理备份。

请注意，如果 **mariadb.service** 正在运行，则可以执行逻辑备份。逻辑备份不包括日志和配置文件。

物理备份由保存内容的文件和目录副本组成。

与逻辑备份相比，物理备份具有以下优点：

- 输出更为紧凑。
- 备份的大小会较小。
- 备份和恢复速度更快。
- 备份包括日志和配置文件。

请注意，当 **mariadb.service** 没有运行或者数据库中的所有表都被锁定以防止备份期间更改时，必须执行物理备份。

您可以使用以下一种 MariaDB 备份方法，来从 MariaDB 数据库备份数据：

- 使用 **mariadb-dump** 的逻辑备份
- 使用 **Mariabackup** 工具的物理在线备份
- 文件系统备份
- 作为备份解决方案复制

2.5.1. 使用 mariadb-dump 执行逻辑备份

mariadb-dump 客户端是一种备份实用程序，可用于转储数据库或数据库集合，用于备份或传输到其他数据库服务器。**mariadb-dump** 的输出通常由 SQL 语句组成，用于重新创建服务器表结构、生成表的数据。**mariadb-dump** 也可以以其他格式生成文件，包括 XML 和分隔的文本格式，如 CSV。

要执行 **mariadb-dump** 备份，您可以使用以下选项之一：

- 备份一个或多个所选的数据库
- 备份所有数据库
- 从一个数据库备份表子集

流程

- 要转储单个数据库，请运行：

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

- 要一次转储多个数据库，请运行：

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

- 要转储所有数据库，请运行：

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

- 要将一个或多个转储的完整数据库加载回服务器，请运行：

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

- 要将数据库加载到远程 MariaDB 服务器，请运行：

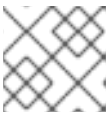
```
# mariadb --host=remote_host < backup-file.sql
```

- 要从一个数据库转储表子集，请在 **mariadb-dump** 命令末尾添加所选表的列表：

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

- 要载入从一个数据库转储的表的子集，请运行：

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



注意

此时，*db_name* 数据库必须存在。

- 要查看 mariadb-dump 支持的选项列表，请运行：

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

其他资源

- MariaDB 文档 - mariadb-dump

2.5.2. 使用 Mariabackup 工具执行物理在线备份

mariabackup 是一个基于 Percona XtraBackup 技术的工具，能够执行 InnoDB、Aria 和 MyISAM 表的物理在线备份。这个工具是由 AppStream 存储库中的 **mariadb-backup** 软件包提供的。

mariabackup 支持对 MariaDB 服务器的全备份功能，其中包括加密和压缩的数据。

先决条件

- **mariadb-backup** 软件包已在系统中安装：

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

- 您必须为 Mariabackup 提供要在其下运行备份的用户的凭证。您可以在命令行中或通过配置文件来提供凭证。
- Mariabackup 的用户必须具有 **RELOAD**、**LOCK TABLES** 和 **REPLICATION CLIENT** 特权。

要使用 Mariabackup 创建数据库备份，请使用以下流程：

流程

- 要在在命令行上提供凭证的同时创建备份，请运行：

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

target-dir 选项定义存储备份文件的目录。如果要执行全备份，目标目录必是空或者不存在。

user 和 **password** 选项允许您配置用户名和密码。

- 要使用配置文件中设置的凭证创建备份：
  1. 在 **/etc/my.cnf.d/** 目录中创建配置文件，例如 **/etc/my.cnf.d/mariabackup.cnf**。
  2. 将以下行添加到新文件的 **[xtrabackup]** 或 **[mysqld]** 部分中：

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

  3. 执行备份：

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

其他资源

- 使用 Mariabackup 进行全备份和恢复

2.5.3. 使用 Mariabackup 工具恢复数据

备份完成后，您可以使用 **mariabackup** 命令及以下一个选项来从备份中恢复数据：

- **--copy-back** 允许您保存原始的备份文件。
- **--move-back** 将备份文件移到数据目录中，并删除原始的备份文件。

要使用 Mariabackup 工具来恢复数据，请使用以下流程：

先决条件

- 验证 **mariadb** 服务是否没有运行：

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

- 验证数据目录是否为空。
- Mariabackup 的用户必须具有 **RELOAD**、**LOCK TABLES** 和 **REPLICATION CLIENT** 特权。

流程

1. 运行 **mariabackup** 命令：
   - 要恢复数据并保留原始备份文件，请使用 **--copy-back** 选项：

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

   - 要恢复数据并删除原始备份文件，请使用 **--move-back** 选项：

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

2. 修复文件权限。
   恢复数据库时，Mariabackup 会保留备份的文件和目录特权。但是，Mariabackup 以恢复数据库的用户和组的身份将文件写入磁盘。恢复备份后，您可能需要调整数据目录的所有者，以匹配 MariaDB 服务器的用户和组，通常两者都为 **mysql**。

   例如，要递归地将文件的所有权改为 **mysql** 用户和组：

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

3. 启动 **mariadb** 服务：

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

其他资源

- 使用 Mariabackup 进行全备份和恢复

2.5.4. 执行文件系统备份

要创建 MariaDB 数据文件的文件系统备份，请将 MariaDB 数据目录的内容复制到您的备份位置。

要同时备份当前的配置或日志文件，请使用以下流程的可选步骤：

流程

1. 停止 **mariadb** 服务：

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

2. 将数据文件复制到所需位置：

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

3. （可选）将配置文件复制到所需位置：

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

4. （可选）将日志文件复制到所需位置：

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

5. 启动 **mariadb** 服务：

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

6. 将备份位置的备份数据加载到 **/var/lib/mysql** 目录时，请确保 **mysql:mysql** 是 **/var/lib/mysql** 中所有数据的所有者：

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

2.5.5. 作为备份解决方案复制

复制是源服务器的一个替代的备份解决方案。如果源服务器复制到副本服务器，备份可以在副本上运行，而不会对源造成任何影响。当您关闭副本，并从副本备份数据时，源仍然可以运行。

警告

复制本身并不是一个足够的备份解决方案。复制可以防止源服务器出现硬件故障，但它不能确保防止数据的丢失。建议您将对副本的任何其他备份解决方案与此方法一起使用。

其他资源

- 使用 Galera 复制 MariaDB
- 作为备份解决方案复制

2.6. 迁移到 MARIADB 10.5

在 RHEL 8 中，**MariaDB** 服务器在 10.3、10.5 和 10.11 版本中可用，每个版本都由单独的模块流提供。RHEL 9 提供 **MariaDB 10.5**、**MariaDB 10.11** 和 **MySQL 8.0**。

这部分论述了从 RHEL 8 中的 **MariaDB 10.3** 版本迁移到 RHEL 9 中的 **MariaDB 10.5** 版本。

2.6.1. MariaDB 10.3 和 MariaDB 10.5 之间的显著区别

MariaDB 10.3 和 **MariaDB 10.5** 之间的显著变化包括：

- **MariaDB** 现在默认使用 **unix_socket** 身份验证插件。该插件允许用户在通过本地 UNIX 套接字文件连接到 **MariaDB** 时使用操作系统凭证。
- **MariaDB** 添加了名为 binary、**mysql*** 符号链接，指向 **mariadb114** 二进制文件。**例如，mysqladmin、mysqlaccess 和 mysqlshow 分别指向 mariadb-admin、mariadb-access**

和 mariadb-show 二进制代码。

- SUPER 特权已被分成几个特权，以更好地与每个用户角色保持一致。因此，某些语句已更改了所需的权限。

- 在并行 复制中，slave_parallel_mode 现在被默认设置为 静态。

- 在 InnoDB 存储引擎 中，以下变量的默认值已发生变化：innodb_adaptive_hash_index 变为 OFF，innodb_checksum_algorithm 变为 full_crc32。

- MariaDB 现在使用用于管理 MariaDB 命令历史记录（the .mysql_history 文件）的底层软件的 libedit 实施，而不是之前使用的 readline 库。此更改会影响直接使用 .mysql_history 文件的用户。注意 .mysql_history 是一个由 MariaDB 或 MySQL 应用管理的文件，用户不应直接使用该文件。人类可读的外表是巧合。

注意

要提高安全性，您可以考虑不维护历史记录文件。禁用记录命令历史记录：

1. 删除 .mysql_history 文件（如果存在的话）。

2. 使用以下任一方法：

   - 将 MYSQL_HISTFILE 变量设置为 /dev/null，并将此设置包含在您的任何 shell 启动文件中。

   - 将 .mysql_history 文件更改为指向 /dev/null 的符号链接：

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

MariaDB Galera 集群 已升级到版本 4，有以下显著变化：

- Galera 添加了一个新的流复制特性，其支持复制无限大小的事务。在执行流复制的过程中，集群以小片段复制事务。

- Galera 现在完全支持全球交易 ID(GTID)。

- /etc/my.cnf.d/galera.cnf 文件中的 wsrep_on 选项的默认值已从 1 改为 0，以防止最终用户在没有配置所需的附加选项的情况下启动 wsrep 复制。

在 MariaDB 10.5 中对 PAM 插件的更改包括：

- MariaDB 10.5 添加了可插拔验证模块(PAM)插件的新版本。PAM 插件版本 2.0 使用单独的 setuid root 帮助程序二进制文件来执行 PAM 身份验证，这使得 MariaDB 可以使用额外的 PAM 模块。

- 帮助程序二进制文件只能由 mysql 组中的用户执行。默认情况下，组只包含 mysql 用户。红帽建议管理员不要向 mysql 组添加更多用户，以防止无需通过这个助手工具进行节流或记录的情况下的密码猜测攻击。

- 在 MariaDB 10.5 中，可插拔验证模块(PAM)插件及其相关文件已移至新软件包 mariadb-pam。因此，在不使用对 MariaDB 进行PAM 验证的系统中不会引入新的 setuid root 二进制文件。

- mariadb-pam 软件包同时包含 PAM 插件版本： 版本 2.0 是默认值，版本 1.0 作为 auth_pam_v1 共享对象库提供。

- 默认情况下，mariadb-pam 软件包不与 MariaDB 服务器一起安装 。要使 PAM 身份验证插件在 MariaDB 10.5 中提供，请手动安装 mariadb-pam 软件包。

2.6.2. 从 RHEL 8 的 MariaDB 10.3 迁移到 RHEL 9 版本的 MariaDB 10.5

这个步骤描述了使用 mariadb-upgrade 程序从 MariaDB 10.3 迁移到 MariaDB 10.5。

mariadb-upgrade 实用程序由 mariadb-server-utils 子软件包提供，该子软件包作为 mariadb-server 软件包的依赖项安装。

先决条件

- 在执行升级前，备份存储在 MariaDB 数据库中的所有数据。

流程

1. 确定在 RHEL 9 系统中安装了 mariadb-server 软件包：

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

2. 确保 mariadb 服务在复制数据时没有在源和目标系统上运行：

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

3. 将源位置的数据复制到 RHEL 9 目标系统的 /var/lib/mysql/ 目录中。

4. 对目标系统上复制的文件设置适当的权限和 SELinux 上下文：

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

5. 确保 mysql:mysql 是 /var/lib/mysql 目录中所有数据的所有者：

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

6. 调整配置，以便位于 /etc/my.cnf.d/ 中的选项文件只包含对 MariaDB 10.5 有效的选项。详情请参阅 MariaDB 10.4 和 MariaDB 10.5 的上游文档。

7. 在目标系统中启动 MariaDB 服务器。

   - 在升级独立运行的数据库时：

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

   - 在升级 Galera 集群节点时：

   ```
   # galera_new_cluster
   ```

mariadb 服务将自动启动。

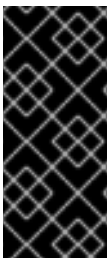
8. 执行 mariadb-upgrade 工具来检查和修复内部表。

- 在升级独立运行的数据库时：

```
$ mariadb-upgrade
```

- 在升级 Galera 集群节点时：

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



重要

有一些与原位升级相关的风险和已知问题。例如，一些查询可能无法正常工作，或者它们以与升级前不同的顺序运行。有关这些风险和问题的更多信息，以及有关原位升级的常规信息，请参阅 MariaDB 10.5 发行注记。

2.7. 从 MARIADB 10.5 升级到 MARIADB 10.11

这部分论述了在 RHEL 9 中从 MariaDB 10.5 迁移到 MariaDB 10.11。

2.7.1. MariaDB 10.5 和 MariaDB 10.11 之间的显著区别

MariaDB 10.5 和 MariaDB 10.11 之间的显著变化包括：

- 新的 sys_schema 功能是视图、功能和程序的集合，用于提供有关数据库使用情况的信息。
- CREATE TABLE,ALTER TABLE,RENAME TABLE,DROP TABLE,DROP DATABASE, 和相关的数据定义语言(DDL)语句现在是原子的。语句必须完全完成，否则更改会被恢复。请注意，当使用 DROP TABLE 删除多个表时，只有每个单个丢弃是原子的，而不是表的完整列表。
- 新的 GRANT … TO PUBLIC 权限可用。

- SUPER 和 READ ONLY ADMIN 权限现在分开。
- 现在，您可以在新的 UUID 数据库数据类型中存储通用唯一标识符。
- MariaDB 现在支持安全套接字层(SSL)协议版本 3。
- MariaDB 服务器现在需要正确配置 SSL 才能启动。在以前的版本中， MariaDB 静默禁用 SSL，并在 SSL 配置错误时使用不安全连接。
- MariaDB 现在支持通过 natural_sort_key （） 函数的自然排序顺序。
- 新的 SFORMAT 功能现在可用于任意文本格式。
- utf8 字符集（和相关 collations）现在默认是 utf8mb3 的别名。
- MariaDB 支持 Unicode Collation Algorithm (UCA) 14 联合。
- MariaDB 的 systemd 套接字激活文件现在包括在 /usr/share/ 目录中。请注意，它们不是 RHEL 中默认配置的一部分，而不是上游。
- 错误消息现在包含 MariaDB 字符串而不是 MySQL。
- 错误消息现在包括在中文语言中。
- 默认的 logrotate 文件已更改。在迁移到 MariaDB 10.11 前检查您的配置。
- 对于 MariaDB 和 MySQL 客户端，命令行中指定的连接属性（如 --port=3306）现在强制客户端和服务器间的协议通信类型，如 tcp、套接字、管道 或内存。在以前的版本中，如果通过 UNIX 套接字连接的 MariaDB 客户端，则指定的端口会被忽略。

2.7.2. 从 RHEL 9 的 MariaDB 10.5 版本升级到 MariaDB 10.11

此流程描述了使用 dnf 和 mariadb-upgrade 工具从 mariadb-server RPM 软件包提供的 MariaDB 10.5 升级到 mariadb:10.11 模块流。

mariadb-upgrade 实用程序由 mariadb-server-utils 子软件包提供，该子软件包作为 mariadb-server 软件包的依赖项安装。

先决条件

- 在执行升级前，备份存储在 MariaDB 数据库中的所有数据。

流程

1. 停止 MariaDB 服务器：

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

2. 从非模块化 MariaDB 10.5 切换到模块化 MariaDB 10.11 ：

```
# dnf module switch-to mariadb:10.11
```

3. 调整配置，以便位于 /etc/my.cnf.d/ 中的选项文件只包含对 MariaDB 10.11 有效的选项。详情请查看 MariaDB 10.6 和 MariaDB 10.11 的上游文档。

4. 启动 MariaDB 服务器。

   - 在升级独立运行的数据库时：

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

   - 在升级 Galera 集群节点时：

   ```
   # galera_new_cluster
   ```

mariadb 服务将自动启动。

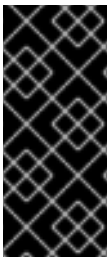
5. 执行 mariadb-upgrade 工具来检查和修复内部表。

- 在升级独立运行的数据库时：

```
# mariadb-upgrade
```

- 在升级 Galera 集群节点时：

```
# mariadb-upgrade --skip-write-binlog
```



重要

有一些与原位升级相关的风险和已知问题。例如，一些查询可能无法正常工作，或者它们以与升级前不同的顺序运行。有关这些风险和问题的更多信息，以及有关原位升级的常规信息，请参阅 MariaDB 10.11 发行注记。

2.8. 使用 GALERA 复制 MARIADB

您可以通过在 Red Hat Enterprise Linux 9 上使用 Galera 解决方案来复制 MariaDB 数据库。

2.8.1. MariaDB Galera 集群介绍

Galera 复制是基于同步多源 MariaDB Galera集群 的创建，该群集由多个 MariaDB 服务器组成。与传统的主/副本设置(副本通常是只读的)不同，MariaDB Galera 群集中的节点都是可写的。

Galera 复制和 MariaDB 数据库之间的接口由写集复制 API(wsrep API) 定义的。

MariaDB Galera 集群 的主要特性是 ：

- 同步复制

- 主动-主动多源拓扑
- 对任何集群节点的读和写
- 自动成员资格控制，故障节点从集群中删除
- 自动节点加入
- 行一级的并行复制
- 直接客户端连接：用户可以登录到集群节点，并在复制运行时直接使用这些节点

同步复制意味着服务器在提交时复制事务，方法是将与事务关联的写入集合广播到集群中的每个节点。客户端（用户应用程序）直接连接到数据库管理系统(DBMS)，可以体验类似于原生 MariaDB 的行为。

同步复制保证集群中一个节点上的更改会同时在集群中的其他节点上发生。

因此，与异步复制相比，同步复制具有以下优势：

- 在特定集群节点间传播更改没有延迟
- 所有集群节点始终一致
- 如果其中一个集群节点崩溃，则不会丢失最新的更改
- 所有集群节点上的事务都会并行执行
- 整个集群的因果关系

其他资源

- 关于 Galera 复制
- 什么是 MariaDB Galera 集群
- MariaDB Galera 集群入门

2.8.2. 构建 MariaDB Galera 集群的组件

要构建 MariaDB Galera 集群，您必须在您的系统上安装以下软件包：

- mariadb-server-galera - 包含 MariaDB Galera 集群 的支持文件和脚本。
- MariaDB-server - 由 MariaDB 上游打补丁，以包含写入集复制 API(wsrep API)。此 API 提供 Galera 复制和 MariaDB 之间的接口。
- Galera - 由 MariaDB 上游打补丁，以添加对 MariaDB 的完全支持。galera 软件包包含以下内容：
  - Galera Replication 程序库 提供整个复制功能。
  - Galera Arbitrator 工具可用作参与脑裂场景的集群成员。但是，Galera Arbitrator 无法参与实际的复制。
  - Galera Systemd 服务 和 Galera 打包程序脚本，它们用于部署 Galera Arbitrator 工具。RHEL 9 提供这些文件的上游版本，位于 /usr/lib/systemd/system/garbd.service 和 /usr/sbin/garb-systemd。

其他资源

- Galera 复制程序

- Galera Arbitrator
- MySQL-wsrep 项目

2.8.3. 部署 MariaDB Galera 集群

您可以部署 MariaDB Galera Cluster 软件包，并更新配置。要组成一个新集群，您必须引导集群的第一个节点。

先决条件

- 安装 MariaDB Galera Cluster 软件包：

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

因此，以下软件包会与依赖项一起安装：

  - mariadb-server-galera
  - mariadb-server
  - galera

有关构建 MariaDB Galera Cluster 的软件包要求的更多信息，请参阅 构建 MariaDB 集群的组件。

- 在系统首次添加到集群前，必须更新 MariaDB 服务器复制配置。

默认配置在 /etc/my.cnf.d/galera.cnf 文件中。

在部署 MariaDB Galera 集群 之前，请将所有节点上的 /etc/my.cnf.d/galera.cnf 文件中的 wsrep_cluster_address 选项设置为以以下字符串开头：

```
gcomm://
```

- 对于初始节点，可以将 wsrep_cluster_address 设置为空列表：

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

- 对于所有其他节点，将 wsrep_cluster_address 设置为包含已属于正在运行的集群的一部分的任何节点的地址。例如：

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

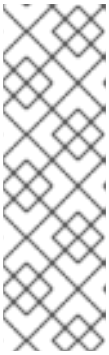
有关如何设置 Galera 集群地址的更多信息，请参阅 Galera 集群地址。

流程

1. 通过在该节点上运行以下 wrapper 来引导新集群的第一个节点：

```
# galera_new_cluster
```

这个打包程序可确保 MariaDB 服务器守护进程(mariadbd)通过 --wsrep-new-cluster 选项运行。此选项提供了没有要连接的现有群集的信息。因此，节点会创建一个新的 UUID 来识别新集群。



注意

mariadb 服务支持 systemd 方法来与多个 MariaDB 服务器进程进行交互。因此，在有多个 MariaDB 服务器运行的情况下，您可以通过将实例名称指定为后缀来引导特定的实例：

```
# galera_new_cluster mariadb@node1
```

2. 在每个节点上运行以下命令将其他节点连接到集群：

```
# systemctl start mariadb
```

因此，节点连接到集群，并将自己与集群的状态同步。

其他资源

- MariaDB Galera 集群入门

2.8.4. 在 MariaDB Galera 集群中添加新节点

要在 MariaDB Galera 集群 中添加新节点，请使用以下步骤。

请注意，您也可以使用此流程重新连接已存在的节点。

流程

- 在特定节点上，在 /etc/my.cnf.d/galera.cnf 配置文件的 [mariadb] 部分的 wsrep_cluster_address 选项中为一个或多个现有群集成员提供一个地址：

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

当新节点连接到现有群集节点中的一个时，就可以看到集群中的所有节点。

但是，最好在 wsrep_cluster_address 中列出集群的所有节点。

因此，任何节点都可以通过连接到任何其他群集节点来加入群集，即使一个或多个群集节点停机了也没关系。当所有成员就成员资格达成一致时，集群的状态将会改变。如果新节点的状态与集群状态不同，新节点需要请求增加状态转移(IST)或状态快照传输(SST)，来确保与其他节点保持一致。

其他资源

- MariaDB Galera 集群入门
- State Snapshot Transfers 简介

2.8.5. 重启 MariaDB Galera 集群

如果您同时关闭所有节点，您将停止集群，正在运行的集群不再存在。但是，集群的数据仍然存在。

要重启集群，请引导第一个节点，如 配置 MariaDB Galera 集群 中所述。

警告

如果集群没有启动，并且第一个节点上的 mariadbd 只是通过 systemctl start mariadb 命令来启动的，那么节点会尝试连接到 /etc/my.cnf.d/galera.cnf 文件 wsrep_cluster_address 选项中列出的至少一个节点。如果当前没有节点运行，那么重启失败。

其他资源

- MariaDB Galera 集群入门。

2.9. 开发 MARIADB 客户端应用程序

红帽建议针对 MariaDB 客户端库开发 MariaDB 客户端应用程序。

针对 MariaDB 客户端库构建应用程序所需的开发文件和程序由 mariadb-connector-c-devel 软件包提供。

不使用直接库名称，而是使用 mariadb_config 程序，该程序在 mariadb-connector-c-devel 软件包中分发。此程序确保返回正确的构建标志。

第 3 章 使用 MYSQL

MySQL 服务器是一个开源、快速且强大的数据库服务器。MySQL 是一个关系型数据库，其将数据转换为结构化的信息，并提供 SQL 接口来访问数据。它包含多个存储引擎和插件，以及地理信息系统(GIS)和 JavaScript 对象表示法(JSON)功能。

了解如何在 RHEL 系统上安装和配置 MySQL，如何备份 MySQL 数据、如何从较早的 MySQL 版本迁移，以及如何复制 MySQL。

3.1. 安装 MYSQL

RHEL 9.0 提供 MySQL 8.0，作为此 Application Stream 的初始版本，您可以作为 RPM 软件包轻松安装。

注意

由于 RPM 软件包有冲突，因此 MySQL 和 MariaDB 数据库服务器无法在 RHEL 9 中并行安装。您可以在容器中并行使用 MySQL 和 MariaDB 数据库服务器，请参阅 在容器中运行多个 MySQL 和 MariaDB 版本。

要安装 MySQL，请使用以下流程。

流程

1. 安装 MySQL 服务器软件包：

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

2. 启动 mysqld 服务：

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

3. 在引导时启用 mysqld 服务：

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

4. *建议：*要在安装 MySQL 时提高安全性，请运行以下命令：

```
$ mysql_secure_installation
```

此命令启动一个完全交互的脚本，该脚本会提示过程中的每一步。该脚本可让您通过以下方法提高安全性：

- 为 root 帐户设置密码
- 删除匿名用户
- 禁止远程 root 登录（在本地主机之外）

3.1.1. 在容器中运行多个 MySQL 和 MariaDB 版本

要在同一个主机上运行 MySQL 和 MariaDB，请在容器中运行它们，因为您无法因为 RPM 软件包冲突而并行安装这些数据库服务器。

此流程包括 MySQL 8.0 和 MariaDB 10.5 作为示例，但您可以使用红帽生态系统目录中提供的任何 MySQL 或 MariaDB 容器版本。

先决条件

- container-tools 元数据包已安装。

流程

1. 使用您的红帽客户门户网站帐户向 registry.redhat.io registry 进行身份验证：

```
# podman login registry.redhat.io
```

如果您已登录到容器 registry，请跳过这一步。

2. 在容器中运行 MySQL 8.0 ：

```
$ podman run -d --name <container_name> -e
MYSQL_ROOT_PASSWORD=<mysql_root_password> -p <host_port_1>:3306
rhel9/mysql-80
```

有关使用此容器镜像的更多信息，请参阅 红帽生态系统目录。

3. 在容器中运行 MariaDB 10.5 ：

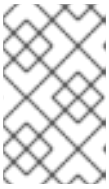
```
$ podman run -d --name <container_name> -e
MYSQL_ROOT_PASSWORD=<mariadb_root_password> -p <host_port_2>:3306
rhel9/mariadb-105
```

有关使用此容器镜像的更多信息，请参阅 红帽生态系统目录。

4. 在容器中运行 MariaDB 10.11 ：

```
$ podman run -d --name <container_name> -e
MYSQL_ROOT_PASSWORD=<mariadb_root_password> -p <host_port_3>:3306
rhel9/mariadb-1011
```

有关使用此容器镜像的更多信息，请参阅 红帽生态系统目录。



注意

两个数据库服务器的容器名称和主机端口必须有所不同。

5. 要确保客户端可以访问网络中的数据库服务器，请在防火墙中打开主机端口：

```
# firewall-cmd --permanent --add-port=
{<host_port_1>/tcp,<host_port_2>/tcp,<host_port_3>/tcp,...}
# firewall-cmd --reload
```

验证步骤

1. 显示正在运行的容器信息：

```
$ podman ps
```

2. 连接到数据库服务器，并以 root 用户身份登录：

```
# mysql -u root -p -h localhost -P <host_port> --protocol tcp
```

其他资源

- 构建、运行和管理容器
- 浏览红帽生态系统目录中的容器

3.2. 配置 MYSQL

要为网络配置 MySQL 服务器，请使用以下流程。

流程

1. 编辑 /etc/my.cnf.d/mysql-server.cnf 文件的 [mysqld] 部分。您可以设置以下配置指令：
   - bind-address - 是服务器监听的地址。可能的选项有：
     - 主机名
     - IPv4 地址
     - IPv6 地址
   - skip-networking - 控制服务器是否监听 TCP/IP 连接。可能的值有：

- 0 - 监听所有客户端
- 1 - 只监听本地客户端

- 端口 - MySQL 侦听 TCP/IP 连接的端口。

2. 重启 mysqld 服务：

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

3.3. 在 MYSQL 服务器上建立 TLS 加密

默认情况下，MySQL 使用未加密的连接。对于安全连接，请在 MySQL 服务器上启用 TLS 支持，并将您的客户端配置为建立加密连接。

3.3.1. 将 CA 证书、服务器证书和私钥放在 MySQL 服务器上

在 MySQL 服务器上启用 TLS 加密前，请将证书颁发机构(CA)证书、服务器证书和私钥存储在 MySQL 服务器上。

先决条件

- 以下 Privacy Enhanced Mail(PEM)格式的文件已复制到服务器：
  - 服务器的私钥：server.example.com.key.pem
  - 服务器证书：server.example.com.crt.pem
  - 证书颁发机构(CA)证书：ca.crt.pem

  有关创建私钥和证书签名请求(CSR)，以及从 CA 请求证书的详情，请查看您的 CA 文档。

流程

1. 将 CA 和服务器证书存储在 /etc/pki/tls/certs/ 目录中：

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

2. 在 CA 和服务器证书上设置权限，以使 MySQL 服务器能够读取文件：

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

由于证书是建立安全连接前通信的一部分，因此任何客户端都可以在不需要身份验证的情况下检索它们。因此，您不需要对 CA 和服务器证书文件设置严格的权限。

3. 将服务器的私钥存储在 /etc/pki/tls/private/ 目录中：

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

4. 对服务器的私钥设置安全权限：

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

如果未授权的用户可以访问私钥，则到 MySQL 服务器的连接不再是安全的。

5. 恢复 SELinux 上下文：

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

3.3.2. 在 MySQL 服务器上配置 TLS

要提高安全性，请在 MySQL 服务器上启用 TLS 支持。因此，客户端可以使用 TLS 加密与服务器传输数据。

先决条件

- 您已安装了 MySQL 服务器。

- mysqld 服务正在运行。
- 服务器上存在 Privacy Enhanced Mail(PEM)格式的以下文件，并可由 mysql 用户读取：
  - 服务器的私钥：/etc/pki/tls/private/server.example.com.key.pem
  - 服务器证书：/etc/pki/tls/certs/server.example.com.crt.pem
  - 证书颁发机构(CA)证书 /etc/pki/tls/certs/ca.crt.pem
- 主题可识别名称(DN)或服务器证书中的主题备用名称(SAN)字段与服务器的主机名相匹配。

流程

1. 创建 /etc/my.cnf.d/mysql-server-tls.cnf 文件：
   a. 添加以下内容来配置到私钥、服务器和 CA 证书的路径：

```
[mysqld]
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. 如果您有证书撤销列表(CRL)，请将 MySQL 服务器配置为使用它：

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

   c. 可选：拒绝未加密的连接尝试。要启用此功能，请附加：

```
require_secure_transport = on
```

   d. 可选：设置服务器应支持的 TLS 版本。例如，要支持 TLS 1.2 和 TLS 1.3，请附加：

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

默认情况下，服务器支持 TLS 1.1、TLS 1.2 和 TLS 1.3。

2. 重启 mysqld 服务：

```
# systemctl restart mysqld
```

验证

要简化故障排除，请在将本地客户端配置为使用 TLS 加密前在 MySQL 服务器上执行以下步骤：

1. 验证 MySQL 现在是否启用了 TLS 加密：

```
# mysql -u root -p -h <MySQL_server_hostname> -e "SHOW session status LIKE
'Ssl_cipher';"
+---------------+-----------------------+
| Variable_name | Value                 |
+---------------+-----------------------+
| Ssl_cipher    | TLS_AES_256_GCM_SHA384 |
+---------------+-----------------------+
```

2. 如果您将 MySQL 服务器配置为只支持特定的 TLS 版本，请显示 tls_version 变量：

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

3. 验证服务器是否使用正确的 CA 证书、服务器证书和私钥文件：

```
# mysql -u root -e "SHOW GLOBAL VARIABLES WHERE Variable_name REGEXP
'^ssl_ca|^ssl_cert|^ssl_key';"
+----------------+-------------------------------------------------+
| Variable_name  | Value                                           |
+----------------+-------------------------------------------------+
| ssl_ca         | /etc/pki/tls/certs/ca.crt.pem                   |
| ssl_capath     |                                                 |
| ssl_cert       | /etc/pki/tls/certs/server.example.com.crt.pem   |
| ssl_key        | /etc/pki/tls/private/server.example.com.key.pem |
+----------------+-------------------------------------------------+
```

-

其他资源

- 将 CA 证书、服务器证书和私钥放在 MySQL 服务器上

3.3.3. 对特定的用户帐户需要 TLS 加密连接

可以访问敏感数据的用户应始终使用 TLS 加密连接，以避免通过网络发送未加密的数据。

如果您无法在服务器上配置所有连接都需要安全传输(require_secure_transport = on)，请将单个用户帐户配置为需要 TLS 加密。

先决条件

- MySQL 服务器启用了 TLS 支持。
- 您配置为需要安全传输的用户已存在。
- CA 证书存储在客户端上。

流程

1. 以管理用户身份连接到 MySQL 服务器：

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

如果您的管理用户没有远程访问服务器的权限，请在 MySQL 服务器上执行命令并连接到 localhost。

2. 使用 REQUIRE SSL 子句强制用户必须使用 TLS 加密连接进行连接：

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

验证

1

1. 使用 TLS 加密，以 example 用户身份连接到服务器：

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

如果没有显示错误，且您可以访问交互式 MySQL 控制台，则与 TLS 的连接成功。

默认情况下，如果服务器提供，客户端会自动使用 TLS 加密。因此，--ssl-ca=ca.crt.pem 和 --ssl-mode=VERIFY_IDENTITY 选项不是必需的，但提高了安全性，因为使用这些选项，客户端可以验证服务器的身份。

2. 尝试以禁用 TLS 的 example 用户身份进行连接：

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

服务器拒绝登录尝试，因为此用户需要 TLS，但禁用了(--ssl-mode=DISABLED)。

其他资源

- 在 MySQL 服务器上配置 TLS

3.4. 在 MYSQL 客户端中使用 CA 证书验证全局启用 TLS 加密

如果您的 MySQL 服务器支持 TLS 加密，请将您的客户端配置为仅建立安全连接，并验证服务器证书。这个流程描述了如何为服务器上的所有用户启用 TLS 支持。

3.4.1. 将 MySQL 客户端配置为默认使用 TLS 加密

在 RHEL 上，您可以全局配置 MySQL 客户端使用 TLS 加密，并验证服务器证书中的通用名称(CN)是否与用户连接的主机名匹配。这可防止中间人攻击。

先决条件

- MySQL 服务器启用了 TLS 支持。
- CA 证书存储在客户端的 /etc/pki/tls/certs/ca.crt.pem 文件中。

流程

- 使用以下内容创建 /etc/my.cnf.d/mysql-client-tls.cnf 文件：

```
[client]
ssl-mode=VERIFY_IDENTITY
ssl-ca=/etc/pki/tls/certs/ca.crt.pem
```

这些设置定义 MySQL 客户端使用 TLS 加密，并且客户端将主机名与服务器证书中的 CN 进行比较(ssl-mode=VERIFY_IDENTITY)。另外，它还指定到 CA 证书的路径(ssl-ca)。

验证

- 使用主机名连接到服务器，并显示服务器的状态：

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

如果 SSL 条目包含 Cipher in use is...，代表连接已加密。

请注意，您在这个命令中使用的用户具有远程身份验证的权限。

如果您连接的主机名与服务器的 TLS 证书中的主机名不匹配，则 ssl-mode=VERIFY_IDENTITY 参数导致连接失败。例如，如果您连接到 localhost ：

```
# mysql -u root -p -h localhost -e status
ERROR 2026 (HY000): SSL connection error: error:0A000086:SSL routines::certificate
verify failed
```

其他资源

- mysql(1) 手册页中的 --ssl* 参数描述。

3.5. 备份 MYSQL 数据

在 Red Hat Enterprise Linux 9 中，备份 MySQL 数据库数据有两个主要方法：

- 逻辑备份
- 物理备份

逻辑备份 由恢复数据所需的 SQL 语句组成。这种类型的备份以纯文本文件的形式导出信息和记录。

与物理备份相比，逻辑备份的主要优势在于可移植性和灵活性。数据可以在其他硬件配置、MySQL 版本或数据库管理系统(DBMS)上恢复，而这些数据无法进行物理备份。

请注意，如果 mysqld.service 正在运行，也可以执行逻辑备份。逻辑备份不包括日志和配置文件。

物理备份由保存内容的文件和目录副本组成。

与逻辑备份相比，物理备份具有以下优点：

- 输出更为紧凑。
- 备份的大小会较小。
- 备份和恢复速度更快。
- 备份包括日志和配置文件。

请注意，当 mysqld.service 没有运行或数据库中的所有表被锁住时，才能执行物理备份，以防在备份过程中数据有更改。

您可以使用以下 MySQL 备份方法之一从 MySQL 数据库备份数据：

- 使用 mysqldump 的逻辑备份
- 文件系统备份
- 作为备份解决方案复制

3.5.1. 使用 mysqldump 执行逻辑备份

mysqldump 客户端是一种备份实用程序，可用于转储数据库或数据库集合，用于备份或传输到其他数据库服务器。mysqldump 的输出通常由 SQL 语句组成，用于重新创建服务器表结构，生成表的数据。mysqldump 也可以以其他格式生成文件，包括 XML 和分隔的文本格式，如 CSV。

要执行 mysqldump 备份，您可以使用以下一种选项：

- 备份一个或多个所选的数据库
- 备份所有数据库
- 从一个数据库备份表子集

流程

- 要转储单个数据库，请运行：

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

- 要一次转储多个数据库，请运行：

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

- 要转储所有数据库，请运行：

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

- 要将一个或多个转储的完整数据库加载回服务器，请运行：

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

- 要将数据库加载到远程 MySQL 服务器，请运行：

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

- 要从一个数据库转储表的字面子集，请在 mysqldump 命令的末尾添加一个所选表的列表：

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

- 要加载从一个数据库转储的表的字面子集，请运行：

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

注意

此时，*db_name* 数据库必须存在。

- 要查看 mysqldump 支持的选项列表，请运行：

```
$ mysqldump --help
```

其他资源

- 使用 mysqldump 的逻辑备份

3.5.2. 执行文件系统备份

要创建 MySQL 数据文件的文件系统备份，请将 MySQL 数据目录的内容复制到您的备份位置。

要同时备份当前的配置或日志文件，请使用以下流程的可选步骤：

流程

1. 停止 mysqld 服务：

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

2. 将数据文件复制到所需位置：

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

3. （可选）将配置文件复制到所需位置：

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

4. （可选）将日志文件复制到所需位置：

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

5. 启动 mysqld 服务：

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

6. 将备份位置的备份数据加载到 /var/lib/mysql 目录时，请确保 mysql:mysql 是 /var/lib/mysql 中所有数据的所有者：

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

3.5.3. 作为备份解决方案复制

复制是源服务器的一个替代的备份解决方案。如果源服务器复制到副本服务器，备份可以在副本上运行，而不会对源造成任何影响。当您关闭副本，并从副本备份数据时，源仍然可以运行。

有关如何复制 MySQL 数据库的说明，请参阅 复制 MySQL。

警告

复制本身并不是一个足够的备份解决方案。复制可以防止源服务器出现硬件故障，但它不能确保防止数据的丢失。建议您将对副本的任何其他备份解决方案与此方法一起使用。

其他资源

- MySQL 复制文档

3.6. 迁移到 RHEL 9 版本的 MYSQL 8.0

RHEL 8 包含 MySQL 8.0、MariaDB 10.3，以及来自 MySQL 数据库系列服务器的 MariaDB 10.5 实施。RHEL 9 提供 MySQL 8.0 和 MariaDB 10.5。

此流程描述了使用 mysql_upgrade 程序从 RHEL 8 的 MySQL 8.0 版本迁移到 MySQL 8.0 的 RHEL 9 版本。mysql_upgrade 工具由 mysql-server 软件包提供。

先决条件

- 在进行升级前，请备份存储在 MySQL 数据库中的所有数据。请参阅备份 MySQL 数据。

流程

1. 确定在 RHEL 9 系统中安装了 mysql-server 软件包：

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

2. 确保在复制数据时 mysqld 服务不在源或目标系统上运行：

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

3. 将源位置的数据复制到 RHEL 9 目标系统的 /var/lib/mysql/ 目录中。

4. 对目标系统上复制的文件设置适当的权限和 SELinux 上下文：

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

5. 确保 mysql:mysql 是 /var/lib/mysql 目录中所有数据的所有者：

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

6. 在目标系统上启动 MySQL 服务器：

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

备注：在较早版本的 MySQL 中，需要 mysql_upgrade 命令来检查和修复内部表。现在，当您启动服务器时会自动完成此操作。

3.7. 复制 MYSQL

MySQL 为复制提供各种配置选项，范围从基本到高级。这部分论述了使用全局事务标识符(GTID)在新安装的 MySQL 上复制 MySQL 的事务方式。使用 GTID 简化了事务识别和一致性验证。

要在 MySQL 中设置复制，您必须：

- 配置源服务器
- 配置副本服务器
- 在源服务器上创建复制用户
- 将副本服务器连接到源服务器

重要

如果要使用现有的 MySQL 服务器进行复制，您必须首先同步数据。如需更多信息，请参阅 上游文档。

3.7.1. 配置 MySQL 源服务器

您可以设置 MySQL 源服务器正确运行所需的配置选项，并复制数据库服务器上所做的所有更改。

先决条件

- 源服务器已安装。

流程

1. 包括 /etc/my.cnf.d/mysql-server.cnf 文件中 [mysqld] 部分下的以下选项：

   - bind-address=*source_ip_adress*

     从副本到源的连接需要这个选项。

   - server-id=*id*

     id 必须是唯一的。

   - log_bin=*path_to_source_server_log*

     此选项定义 MySQL 源服务器的二进制日志文件的路径。例如：log_bin=/var/log/mysql/mysql-bin.log。

   - gtid_mode=ON

     此选项在服务器上启用全局事务标识符(GTID)。

- **enforce-gtid-consistency=ON**

  服务器通过仅允许执行可使用 GTID 进行安全记录的语句来强制实施 GTID 一致性。

- *可选:* **binlog_do_db=*db_name***

  如果您只想复制所选的数据库，则使用这个选项。要复制多个所选的数据库，请分别指定每个数据库：

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

- *可选:* **binlog_ignore_db=*db_name***

  使用此选项从复制中排除特定的数据库。

2. **重启 mysqld 服务：**

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

3.7.2. 配置 MySQL 副本服务器

您可以设置 MySQL 副本服务器所需的配置选项，以确保成功复制。

先决条件

- **副本服务器已安装。**

流程

1. **包括 /etc/my.cnf.d/mysql-server.cnf 文件中 [mysqld] 部分下的以下选项：**

   - **server-id=*id***

id 必须是唯一的。

- relay-log=*path_to_replica_server_log*

  中继日志是在复制过程中由 MySQL 副本服务器创建的一组日志文件。

- log_bin=*path_to_replica_sever_log*

  此选项定义了 MySQL 副本服务器的二进制日志文件的路径。例如：log_bin=/var/log/mysql/mysql-bin.log。

  副本中不需要这个选项，但强烈建议使用。

- gtid_mode=ON

  此选项在服务器上启用全局事务标识符(GTID)。

- enforce-gtid-consistency=ON

  服务器通过仅允许执行可使用 GTID 进行安全记录的语句来强制实施 GTID 一致性。

- log-replica-updates=ON

  这个选项可确保从源服务器接收的更新记录在副本的二进制日志中。

- skip-replica-start=ON

  此选项可确保在副本服务器启动时不启动复制线程。

- *可选:* binlog_do_db=*db_name*

  如果您只想复制某些数据库，则使用这个选项。要复制多个数据库，请分别指定每个数据库：

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

- *可选:* **binlog_ignore_db=*db_name***

  使用此选项从复制中排除特定的数据库。

2. 重启 **mysqld** 服务：

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

3.7.3. 在 MySQL 源服务器上创建复制用户

您必须创建一个复制用户，并授予这个用户所需的复制流量的权限。此流程演示了如何创建具有适当权限的复制用户。仅在源服务器上执行这些步骤。

先决条件

- 源服务器已安装并配置，如 配置 MySQL 源服务器 中所述。

流程

1. 创建复制用户：

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

2. 授予用户复制权限：

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

3. 重新载入 MySQL 数据库中的授权表：

```
mysql> FLUSH PRIVILEGES;
```

4. 将源服务器设置为只读状态：

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

3.7.4. 将副本服务器连接到源服务器

在 MySQL 副本服务器上，您必须配置凭证和源服务器的地址。使用以下流程实现副本服务器。

先决条件

- 源服务器已安装并配置，如 配置 MySQL 源服务器 中所述。
- 副本服务器已安装并配置，如 配置 MySQL 副本服务器 中所述。
- 您已创建了复制用户。请参阅 在 MySQL 源服务器上创建复制用户。

流程

1. 将副本服务器设置为只读状态：

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

2. 配置复制源：

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

3. 在 MySQL 副本服务器中启动副本线程：

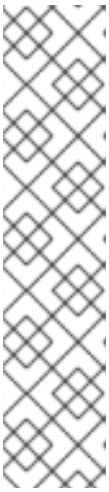
```
mysql> START REPLICA;
```

4. 在源和目标服务器上取消只读状态的设置：

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

5. *可选：*检查副本服务器的状态以进行调试：

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



注意

如果复制服务器启动或连接失败，您可以跳过 SHOW MASTER STATUS 命令的输出中显示的二进制日志文件位置后的某些事件。例如，从定义的位置跳过第一个事件：

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

尝试再次启动副本服务器。

6. *可选：*停止副本服务器中的副本线程：

```
mysql> STOP REPLICA;
```

3.7.5. 验证步骤

1. 在源服务器上创建一个示例数据库：

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

2. 验证 ***test_db_name*** 数据库是否在副本服务器上进行复制。

3. 在源或副本服务器上执行以下命令，显示 MySQL 服务器的二进制日志文件的状态信息：

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

Executed_Gtid_Set 列，针对在源上执行的事务显示一组 GTID，它不能为空。

注意

当在副本服务器上使用 SHOW SLAVE STATUS 时，Executed_Gtid_Set 行中会显示相同的 GTID。

3.7.6. 其他资源

- MySQL 复制文档
- 如何在 MySQL 中设置复制
- 带有全局事务标识符的复制

3.8. 开发 MYSQL 客户端应用程序

红帽建议针对 MariaDB 客户端库开发您的 MySQL 客户端应用程序。客户端和服务器之间的通信协议在 MariaDB 和 MySQL 之间兼容。MariaDB 客户端库适用于最常见的 MySQL 场景，除了限制特定于 MySQL 的功能的数量。

针对 MariaDB 客户端库构建应用程序所需的开发文件和程序由 mariadb-connector-c-devel 软件包提供。

不使用直接库名称，而是使用 mariadb_config 程序，该程序在 mariadb-connector-c-devel 软件包中分发。此程序确保返回正确的构建标志。

第 4 章 使用 POSTGRESQL

PostgreSQL 服务器是一个基于 SQL 语言的开源、健壮且高度可扩展的数据库服务器。PostgreSQL 服务器提供了一个对象关系型数据库系统，其可以管理大量的数据集和大量的并发用户。因此，PostgreSQL 服务器可用在集群中，来管理大量的数据。

PostgreSQL 服务器包含确保数据完整性、构建容错环境和应用程序的功能。使用 PostgreSQL 服务器，您可以使用您自己的数据类型、自定义函数或来自不同编程语言的代码来扩展数据库，而无需重新编译数据库。

了解如何在 RHEL 系统上安装和配置 PostgreSQL，如何备份 PostgreSQL 数据，以及如何从早期的 PostgreSQL 版本迁移。

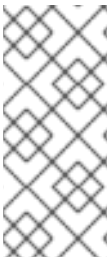
4.1. 安装 POSTGRESQL

RHEL 9 提供 PostgreSQL 13 作为此应用程序流的初始版本，您可以轻松地作为 RPM 软件包安装。

在 RHEL 9 的次版本中，其他 PostgreSQL 版本作为模块提供较短的生命周期：

- RHEL 9.2 引入了 PostgreSQL 15 作为 postgresql:15 模块流
- RHEL 9.4 引入了 PostgreSQL 16 作为 postgresql:16 模块流

要安装 PostgreSQL，请使用以下流程：



注意

按照设计，无法并行安装同一模块的多个版本(stream)。因此，您必须从 postgresql 模块中只选择一个可用流。您可以在容器中使用不同版本的 PostgreSQL 数据库服务器，请参阅 在容器中运行多个 PostgreSQL 版本。

流程

1. 安装 PostgreSQL 服务器软件包：

- 对于 RPM 软件包中的 PostgreSQL 13 ：

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

- 对于 PostgreSQL 15 或 PostgreSQL 16，请从 postgresql 模块中选择流（版本）15 或 16，并指定 服务器配置文件，例如：

```
# dnf module install postgresql:16/server
```

postgres 超级用户会自动创建。

2. 初始化数据库集群：

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

红帽建议将数据存储在默认的 /var/lib/pgsql/data 目录中。

3. 启动 postgresql 服务：

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

4. 启用 postgresql 服务，以便在引导时启动：

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

重要

如果要从 RHEL 9 中的早期 postgresql 流升级，请按照 切换到更新的流 和 迁移到 RHEL 9 版本的 PostgreSQL 中描述的两个步骤进行。

4.1.1. 在容器中运行多个 PostgreSQL 版本

要在同一主机上运行不同版本的 PostgreSQL，请在容器中运行它们，因为您无法并行安装同一模块的多个版本(streams)。

此流程包括 PostgreSQL 13 和 PostgreSQL 15 作为示例，但您可以使用 Red Hat Ecosystem Catalog 中提供的任何 PostgreSQL 容器版本。

先决条件

- container-tools 元数据包已安装。

流程

1. 使用您的红帽客户门户网站帐户向 registry.redhat.io registry 进行身份验证：

```
# podman login registry.redhat.io
```

如果您已登录到容器 registry，请跳过这一步。

2. 在容器中运行 PostgreSQL 13 ：

```
$ podman run -d --name <container_name> -e POSTGRESQL_USER=<user_name> -e
POSTGRESQL_PASSWORD=<password> -e
POSTGRESQL_DATABASE=<database_name> -p <host_port_1>:5432
rhel9/postgresql-13
```

有关使用此容器镜像的更多信息，请参阅 红帽生态系统目录。

3. 在容器中运行 PostgreSQL 15 ：

```
$ podman run -d --name <container_name> -e POSTGRESQL_USER=<user_name> -e
POSTGRESQL_PASSWORD=<password> -e
POSTGRESQL_DATABASE=<database_name> -p <host_port_2>:5432
rhel9/postgresql-15
```

有关使用此容器镜像的更多信息，请参阅 红帽生态系统目录。

4. 在容器中运行 PostgreSQL 16 ：

```
$ podman run -d --name <container_name> -e POSTGRESQL_USER=<user_name> -e
POSTGRESQL_PASSWORD=<password> -e
```

```
POSTGRESQL_DATABASE=<database_name> -p <host_port_3>:5432
rhel9/postgresql-16
```

有关使用此容器镜像的更多信息，请参阅 红帽生态系统目录。

注意

两个数据库服务器的容器名称和主机端口必须有所不同。

5. 要确保客户端可以访问网络中的数据库服务器，请在防火墙中打开主机端口：

```
# firewall-cmd --permanent --add-port=
{<host_port_1>/tcp,<host_port_2>/tcp,<host_port_3>/tcp,...}
# firewall-cmd --reload
```

验证步骤

1. 显示正在运行的容器信息：

```
$ podman ps
```

2. 连接到数据库服务器，并以 root 用户身份登录：

```
# psql -u postgres -p -h localhost -P <host_port> --protocol tcp
```

其他资源

- 构建、运行和管理容器
- 浏览红帽生态系统目录中的容器

4.2. 创建 POSTGRESQL 用户

PostgreSQL 用户为以下类型：

- postgres UNIX 系统用户 - 应该仅用于运行 PostgreSQL 服务器和客户端应用程序，如

pg_dump。不要将 postgres 系统用户用于 PostgreSQL 管理的任何交互式工作，如数据库创建和用户管理。

- 数据库超级用户 - 默认的 postgres PostgreSQL 超级用户与 postgres 系统用户无关。您可以在 pg_hba.conf 文件中限制 postgres 超级用户的权限，否则没有其他权限限制。您也可以创建其他数据库超级用户。

- 具有特定数据库访问权限的角色：

  - 数据库用户 - 默认具有登录权限

  - 一组用户 - 启用整个组的管理权限

角色可以拥有数据库对象（如表和函数），并且可以使用 SQL 命令将对象特权分配给其他角色。

标准数据库管理特权包括 SELECT、INSERT、UPDATE、DELETE、TRUNCATE、REFERENCES、TRIGGER、CREATE、CONNECT、TEMPORARY、EXECUTE 和 USAGE。

角色属性是特殊的特权，如 LOGIN、SUPERUSER、CREATEDB 和 CREATEROLE。

重要

红帽建议以不是超级用户的角色身份执行大部分任务。常见的做法是创建一个具有 CREATEDB 和 CREATEROLE 特权的角色，并将此角色用于所有数据库和角色的日常管理。

先决条件

- 已安装 PostgreSQL 服务器。

- 初始化数据库集群。

流程

- 要创建用户，请为用户设置密码，并为该用户分配 CREATEROLE 和 CREATEDB 权限：

```
postgres=# CREATE USER mydbuser WITH PASSWORD 'mypasswd' CREATEROLE
CREATEDB;
```

将 *mydbuser* 替换为用户名，*mypasswd* 替换为用户的密码。

其他资源

- PostgreSQL 数据库角色
- PostgreSQL 特权
- 配置 PostgreSQL

例 4.1. 初始化、创建和连接到 PostgreSQL 数据库

本例演示如何初始化 PostgreSQL 数据库，创建具有例行数据库管理特权的数据库用户，以及如何创建可通过具有管理特权的任何系统帐户访问的数据库帐户。

1. 安装 PosgreSQL 服务器：

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

2. 初始化数据库集群：

```
# postgresql-setup --initdb
* Initializing database in '/var/lib/pgsql/data'
* Initialized, logs are in /var/lib/pgsql/initdb_postgresql.log
```

3. 将密码哈希算法设置为 scram-sha-256。

   a. 在 /var/lib/pgsql/data/postgresql.conf 文件中，更改以下行：

```
#password_encryption = md5              # md5 or scram-sha-256
```

改为：

```
password_encryption = scram-sha-256
```

b. 在 /var/lib/pgsql/data/pg_hba.conf 文件中，更改 IPv4 本地连接的以下行：

```
host    all             all             127.0.0.1/32            ident
```

改为：

```
host    all             all             127.0.0.1/32            scram-sha-256
```

4. 启动 postgresql 服务：

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

5. 以名为 postgres 的系统用户登录：

```
# su - postgres
```

6. 启动 PostgreSQL 互动终端：

```
$ psql
psql (13.7)
Type "help" for help.

postgres=#
```

7. 可选：获取有关当前数据库连接的信息：

```
postgres=# \conninfo
You are connected to database "postgres" as user "postgres" via socket in
"/var/run/postgresql" at port "5432".
```

8. 创建名为 mydbuser 的用户，为 mydbuser 设置密码，并为 mydbuser 分配 CREATEROLE 和 CREATEDB 权限：

```
postgres=# CREATE USER mydbuser WITH PASSWORD 'mypasswd'
CREATEROLE CREATEDB;
CREATE ROLE
```

mydbuser 用户现在可以执行日常数据库管理操作：创建数据库并管理用户索引。

9. 使用 \q meta 命令从互动终端注销：

```
postgres=# \q
```

10. 注销 postgres 用户会话：

```
$ logout
```

11. 以 mydbuser 用户身份登录 PostgreSQL 终端，指定主机名并连接到默认的 postgres 数据库，该数据库在初始化过程中创建：

```
# psql -U mydbuser -h 127.0.0.1 -d postgres
Password for user mydbuser:
Type the password.
psql (13.7)
Type "help" for help.

postgres=>
```

12. 创建名为 mydatabase 的数据库：

```
postgres=> CREATE DATABASE mydatabase;
CREATE DATABASE
postgres=>
```

13. 从会话注销：

```
postgres=# \q
```

14. 以 mydbuser 用户身份连接到 mydatabase：

```
# psql -U mydbuser -h 127.0.0.1 -d mydatabase
Password for user mydbuser:
```

```
psql (13.7)
Type "help" for help.
mydatabase=>
```

15. 可选：获取有关当前数据库连接的信息：

```
mydatabase=> \conninfo
You are connected to database "mydatabase" as user "mydbuser" on host
"127.0.0.1" at port "5432".
```

4.3. 配置 POSTGRESQL

在 PostgreSQL 数据库中，所有数据和配置文件都存储在一个名为database cluster的目录中。红帽建议将所有数据（包括配置文件）存储在默认的 /var/lib/pgsql/data/ 目录中。

PostgreSQL 配置由以下文件组成：

- PostgreSQL.conf - 用于设置数据库集群参数。
- PostgreSQL.auto.conf - 包含与 postgresql.conf 类似的基本 PostgreSQL 设置。但是这个文件由服务器控制。它由 ALTER SYSTEM 查询来编辑，无法手动编辑。
- pg_ident.conf - 用于将来自外部身份验证机制的用户身份映射到 PostgreSQL 用户身份。
- pg_hba.conf - 用于为 PostgreSQL 数据库配置客户端身份验证。

要修改 PostgreSQL 配置，请使用以下流程：

流程

1. 编辑相应的配置文件，如 /var/lib/pgsql/data/postgresql.conf。
2. 重启 postgresql 服务，以使修改生效：

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

例 4.2. 配置 PostgreSQL 数据库集群参数

本例展示了 /var/lib/pgsql/data/postgresql.conf 文件中数据库集群参数的基本设置。

```
# This is a comment
log_connections = yes
log_destination = 'syslog'
search_path = '"$user", public'
shared_buffers = 128MB
password_encryption = scram-sha-256
```

例 4.3. 在 PostgreSQL 中设置客户端身份验证

本例演示了如何在 /var/lib/pgsql/data/pg_hba.conf 文件中设置客户端身份验证。

```
# 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
```

4.4. 在 POSTGRESQL 服务器中配置 TLS 加密

默认情况下，PostgreSQL 使用未加密的连接。如需更多安全连接，您可以在 PostgreSQL 服务器中启用传输层安全(TLS)支持，并将客户端配置为建立加密连接。

先决条件

- 已安装 PostgreSQL 服务器。
- 初始化数据库集群。
- 如果服务器运行 RHEL 9.2 或更高版本，并且启用了 FIPS 模式，则客户端必须支持扩展 Master Secret (EMS)扩展或使用 TLS 1.3。没有 EMS 的 TLS 1.2 连接会失败。如需更多信息，请参阅 强制 TLS 扩展"Extended Master Secret" 知识库文章。

流程

1. 安装 OpenSSL 库：

```
# dnf install openssl
```

2. 生成 TLS 证书和密钥：

```
# openssl req -new -x509 -days 365 -nodes -text -out server.crt \
  -keyout server.key -subj "/CN=dbhost.yourdomain.com"
```

将 *dbhost.yourdomain.com* 替换为您的数据库主机和域名。

3. 将签名的证书和私钥复制到数据库服务器的所需位置：

```
# cp server.{key,crt} /var/lib/pgsql/data/.
```

4. 将签名证书的所有者和组所有权改为 postgres 用户：

```
# chown postgres:postgres /var/lib/pgsql/data/server.{key,crt}
```

5. 限制私钥的权限，使其只可由所有者读取：

```
# chmod 0400 /var/lib/pgsql/data/server.key
```

6. 通过在 /var/lib/pgsql/data/postgresql.conf 文件中更改以下行，将密码哈希算法设置为 scram-sha-256 ：

```
#password_encryption = md5              # md5 or scram-sha-256
```

改为：

```
password_encryption = scram-sha-256
```

7. 通过在 /var/lib/pgsql/data/postgresql.conf 文件中更改以下行，将 PostgreSQL 配置为使用 SSL/TLS：

```
#ssl = off
```

改为：

```
ssl=on
```

8. 通过更改 /var/lib/pgsql/data/pg_hba.conf 文件中的 IPv4 本地连接，将所有数据库的访问限制为只使用 TLS 的客户端的连接：

```
host  all  all  127.0.0.1/32  ident
```

改为：

```
hostssl  all  all  127.0.0.1/32  scram-sha-256
```

另外，您可以通过添加以下新行来限制单个数据库和用户的访问：

```
hostssl mydatabase mydbuser 127.0.0.1/32  scram-sha-256
```

将 *mydatabase* 替换为数据库名称，并将 *mydbuser* 替换为用户名。

9. 通过重启 postgresql 服务来有效地进行更改：

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

验证

- 手动验证连接是否已加密：

1. 以 *mydbuser* 用户身份连接到 PostgreSQL 数据库，指定主机名和数据库名称：

```
$ psql -U mydbuser -h 127.0.0.1 -d mydatabase
Password for user mydbuser:
```

将 *mydatabase* 替换为数据库名称，并将 *mydbuser* 替换为用户名。

2. 获取有关当前数据库连接的信息：

```
mydbuser=> \conninfo
You are connected to database "mydatabase" as user "mydbuser" on host
"127.0.0.1" at port "5432".
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits:
256, compression: off)
```

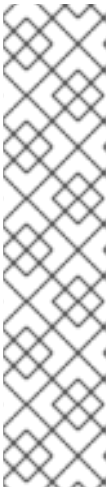
- 您可以编写一个简单的应用程序，验证与 PostgreSQL 的连接是否已加密。本例演示了使用 C 编写的那些使用 libpq 客户端库（由 libpq-devel 软件包提供）的应用程序：

```
#include <stdio.h>
#include <stdlib.h>
#include <libpq-fe.h>

int main(int argc, char* argv[])
{
//Create connection
PGconn* connection = PQconnectdb("hostaddr=127.0.0.1 password=mypassword
port=5432 dbname=mydatabase user=mydbuser");

if (PQstatus(connection) ==CONNECTION_BAD)
   {
   printf("Connection error\n");
   PQfinish(connection);
   return -1; //Execution of the program will stop here
   }
   printf("Connection ok\n");
   //Verify TLS
   if (PQsslInUse(connection)){
    printf("TLS in use\n");
    printf("%s\n", PQsslAttribute(connection,"protocol"));
   }
   //End connection
   PQfinish(connection);
   printf("Disconnected\n");
   return 0;
}
```

将 *mypassword* 替换为密码，将 *mydatabase* 替换为数据库名称，并将 *mydbuser* 替换为用户名。



注意

您必须使用 **-lpq** 选项为编译加载 **pq** 库。例如，使用 GCC 编译器编译应用程序：

```
$ gcc source_file.c -lpq -o myapplication
```

其中 *source_file.c* 包含上面的示例代码，*myapplication* 是应用程序的名称，用于验证安全的 PostgreSQL 连接。

例 4.4. 使用 TLS 加密初始化、创建和连接到 PostgreSQL 数据库

本例演示如何初始化 PostgreSQL 数据库，创建数据库用户和数据库，以及如何使用安全连接连接到数据库。

1. 安装 PosgreSQL 服务器：

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

2. 初始化数据库集群：

```
# postgresql-setup --initdb
* Initializing database in '/var/lib/pgsql/data'
* Initialized, logs are in /var/lib/pgsql/initdb_postgresql.log
```

3. 安装 OpenSSL 库：

```
# dnf install openssl
```

4. 生成 TLS 证书和密钥：

```
# openssl req -new -x509 -days 365 -nodes -text -out server.crt \
  -keyout server.key -subj "/CN=dbhost.yourdomain.com"
```

将 *dbhost.yourdomain.com* 替换为您的数据库主机和域名。

5. 将签名的证书和私钥复制到数据库服务器的所需位置：

```
# cp server.{key,crt} /var/lib/pgsql/data/.
```

6. 将签名证书的所有者和组所有权改为 **postgres** 用户：

```
# chown postgres:postgres /var/lib/pgsql/data/server.{key,crt}
```

7. 限制私钥的权限，使其只可由所有者读取：

```
# chmod 0400 /var/lib/pgsql/data/server.key
```

8. 将密码哈希算法设置为 **scram-sha-256**。在 **/var/lib/pgsql/data/postgresql.conf** 文件中，更改以下行：

```
#password_encryption = md5            # md5 or scram-sha-256
```

改为：

```
password_encryption = scram-sha-256
```

9. 将 **PostgreSQL** 配置为使用 **SSL/TLS**。在 **/var/lib/pgsql/data/postgresql.conf** 文件中，更改以下行：

```
#ssl = off
```

改为：

```
ssl=on
```

10. 启动 **postgresql** 服务：

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

11. 以名为 **postgres** 的系统用户登录：

```
# su - postgres
```

12. 以 postgres 用户身份启动 PostgreSQL 互动终端：

```
$ psql -U postgres
psql (13.7)
Type "help" for help.

postgres=#
```

13. 创建名为 mydbuser 的用户，再为 mydbuser 设置一个密码：

```
postgres=# CREATE USER mydbuser WITH PASSWORD 'mypasswd';
CREATE ROLE
postgres=#
```

14. 创建名为 mydatabase 的数据库：

```
postgres=# CREATE DATABASE mydatabase;
CREATE DATABASE
postgres=#
```

15. 为 mydbuser 用户授予所有权限：

```
postgres=# GRANT ALL PRIVILEGES ON DATABASE mydatabase TO mydbuser;
GRANT
postgres=#
```

16. 从互动终端注销：

```
postgres=# \q
```

17. 注销 postgres 用户会话：

```
$ logout
```

18. 通过更改 /var/lib/pgsql/data/pg_hba.conf 文件中的 IPv4 本地连接，将所有数据库的访问限制为只使用 TLS 的客户端的连接：

```
host  all  all  127.0.0.1/32  ident
```

改为：

```
hostssl  all  all  127.0.0.1/32  scram-sha-256
```

19. 通过重启 postgresql 服务来有效地进行更改：

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

20. 以 mydbuser 用户身份连接到 PostgreSQL 数据库，指定主机名和数据库名称：

```
$ psql -U mydbuser -h 127.0.0.1 -d mydatabase
Password for user mydbuser:
psql (13.7)
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256,
compression: off)
Type "help" for help.

mydatabase=>
```

4.5. 备份 POSTGRESQL 数据

要备份 PostgreSQL 数据，请使用以下方法之一：

SQL 转储

请参阅 使用 SQL 转储进行备份。

文件系统级别备份

请参阅 文件系统级备份。

持续归档

请参阅 持续归档。

4.5.1. 使用 SQL 转储备份 PostgreSQL 数据

SQL 转储方法基于使用 SQL 命令生成转储文件。当转储上传回数据库服务器时，它会按与转储时相

同的状态重新创建数据库。

以下 PostgreSQL 客户端应用程序为 SQL 转储提供了保证：

- pg_dump 转储单个数据库，而无需有关角色或表空间的集群范围的信息
- pg_dumpall 转储给定集群中的每个数据库，并保留集群范围的数据，如角色和表空间定义。

默认情况下，pg_dump 和 pg_dumpall 命令将它的们结果写入标准输出。要将转储保存到文件中，请将输出重定向到 SQL 文件。生成的 SQL 文件可以是文本格式，也可以是允许并行且可以更详细地控制对象恢复的其他格式。

您可以在任何可访问数据库的远程主机中执行 SQL 转储。

4.5.1.1. SQL 转储的优点和缺陷

与其它 PostgreSQL 备份方法相比，SQL 转储具有以下优点：

- SQL 转储是唯一的、不针对特定服务器版本的 PostgreSQL 备份方法。pg_dump 工具的输出可以重新加载到 PostgreSQL 的后续版本中，这不适用于文件系统级备份或持续归档。
- SQL 转储是将数据库传输到不同计算机架构（比如从 32 位服务器传输到 64 位服务器）的唯一方法。
- SQL 转储提供内部一致的转储。转储表示在pg_dump 开始运行时的数据库快照。
- pg_dump 程序不会阻止数据库中的其他操作。

SQL 转储的一个缺点是，与文件系统级备份相比，它需要更长的时间。

4.5.1.2. 使用 pg_dump 执行 SQL 转储

要转储一个没有集群范围信息的单个数据库，请使用 pg_dump 工具。

先决条件

- 您必须对要转储的所有表具有读的权限。若要转储整个数据库，您必须以 postgres 超级用户或具有数据库管理员特权的用户身份运行命令。

流程

- 转储没有集群范围信息的数据库：

```
$ pg_dump dbname > dumpfile
```

要指定 pg_dump 会联系哪个数据库服务器，请使用以下命令行选项：

- -h 选项用来定义主机 。

  默认主机要么是本地主机，要么是 PGHOST 环境变量所指定的主机。

- -p 选项用来定义端口 。

  默认端口是由 PGPORT 环境变量或编译后的默认值指明的。

4.5.1.3. 使用 pg_dumpall 执行 SQL 转储

要转储给定数据库集群中的每个数据库，并保留集群范围的数据，请使用 pg_dumpall 工具。

先决条件

- 您必须以 postgres 超级用户或具有数据库管理员特权的用户身份运行命令。

流程

- 转储数据库集群中的所有数据库，并保留集群范围的数据：

```
$ pg_dumpall > dumpfile
```

要指定pg_dumpall与哪个数据库服务器联系，请使用以下命令行选项：

- -h 选项用来定义主机 。

  默认主机要么是本地主机，要么是 PGHOST 环境变量所指定的主机。

- -p 选项用来定义端口 。

  默认端口是由 PGPORT 环境变量或编译后的默认值指明的。

- -l 选项用来定义默认数据库。

  这个选项使您能够选择一个与初始化过程中自动创建的 postgres 数据库不同的默认数据库。

4.5.1.4. 恢复使用 pg_dump 转储的数据库

要从使用 pg_dump 工具转储的 SQL 转储恢复数据库，请按照以下流程。

先决条件

- 您必须以 postgres 超级用户或具有数据库管理员特权的用户身份运行命令。

流程

1. 创建新数据库：

   ```
   $ createdb dbname
   ```

2. 确保所有拥有对象的用户或对转储数据库中的对象赋予了权限的用户都已存在。如果这样的用户不存在，恢复将无法重新创建具有原始所有权和权限的对象。

3. 运行 psql 工具来恢复 pg_dump 程序创建的文本文件转储：

```
$ psql dbname < dumpfile
```

其中 dumpfile 是 pg_dump 命令的输出。要恢复非文本文件转储，请使用 pg_restore 工具：

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

4.5.1.5. 恢复使用 pg_dumpall 转储的数据库

要从使用 pg_dumpall 工具转储的数据库集群中恢复数据，请按照以下步骤。

先决条件

- 您必须以 postgres 超级用户或具有数据库管理员特权的用户身份运行命令。

流程

1. 确保所有拥有对象的用户或对转储数据库中的对象赋予了权限的用户都已存在。如果这样的用户不存在，恢复将无法重新创建具有原始所有权和权限的对象。

2. 运行 psql 工具来恢复由 pg_dumpall 工具创建的文本文件转储：

```
$ psql < dumpfile
```

其中 dumpfile 是 pg_dumpall 命令的输出。

4.5.1.6. 在另一服务器上执行数据库的 SQL 转储

将数据库从一台服务器直接转储到另一台服务器是可能的，因为 pg_dump 和 psql 可以写入管道并从管道读取。

流程

- 要从一个服务器到另一个服务器转储数据库，请运行：

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

4.5.1.7. 在恢复过程中处理 SQL 错误

默认情况下，如果出现 SQL 错误，psql 会继续执行，从而导致数据库只部分恢复。

要修改默认行为，在恢复转储时使用以下任一方法：

先决条件

- 您必须以 postgres 超级用户或具有数据库管理员特权的用户身份运行命令。

流程

- 请设置 ON_ERROR_STOP 变量，使 psql 在发生 SQL 错误时退出，且有一个为 3 的退出状态码：

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

- 指定整个转储作为一个事务来恢复，以便要么全部完成，要么全部取消。
  - 使用 psql 工具恢复文本文件转储时：

```
$ psql -1
```

  - 使用 pg_restore 工具恢复非文本文件转储时：

```
$ pg_restore -e
```

请注意，在使用这个方法时，即使一个小的错误也可以取消已经运行了很长时间的恢复操作。

4.5.1.8. 其他资源

- PostgreSQL 文档 - SQL 转储

4.5.2. 使用文件系统级别备份来备份 PostgreSQL 数据

要创建文件系统级别备份，请将 PostgreSQL 数据库文件复制到另一个位置。例如，您可以使用以下任一方法：

- 使用 tar 工具创建归档文件。
- 使用 rsync 工具将文件复制到其它位置。
- 创建数据目录的一致快照。

4.5.2.1. 文件系统备份的优点和限制

文件系统级别备份与其他 PostgreSQL 备份方法相比有以下优点：

- 文件系统级的备份通常比 SQL 转储要快。

与其它 PostgreSQL 备份方法相比，文件系统级备份有以下限制：

- 当您要从 RHEL 8 升级到 RHEL 9 时，这个备份方法不合适，并将您的数据迁移到升级的系统。文件系统级别备份特定于架构和 RHEL 主版本。如果升级不成功，但无法在 RHEL 9 系统中恢复数据，则可以在 RHEL 8 系统中恢复数据。
- 在备份和恢复数据前，必须关闭数据库服务器。
- 无法备份和恢复某些独立文件或表。备份文件系统只适用于完全备份和恢复整个数据库集群。

4.5.2.2. 执行文件系统级别备份

要执行文件系统级别备份，请使用以下步骤。

流程

1. 选择数据库集群的位置，并初始化该集群：

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

2. 停止 postgresql 服务：

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

3. 使用任意方法创建文件系统备份，如 tar 归档：

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

4. 启动 postgresql 服务：

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

其他资源

- PostgreSQL 文档 - 文件系统级备份

4.5.3. 通过持续存档来备份 PostgreSQL 数据

4.5.3.1. 持续归档介绍

PostgreSQL 将对数据库的数据文件所做的每项修改记录到预写日志(WAL)文件中，该文件位于集群数据目录的 pg_wal/ 子目录中。此日志主要用于崩溃恢复。崩溃后，可用上次检查点以后所记录的日志条目将数据库恢复到一致。

持续归档方法也称为在线备份，以在运行的服务器上执行的基础备份或文件系统级备份的形式，将 WAL 文件与数据库集群的副本结合起来。

如果需要进行数据库恢复，您可以从数据库集群的副本恢复数据库，然后从备份的 WAL 文件中重新执行日志，使系统恢复到当前状态。

使用持续归档方法时，您必须保持所有归档的 WAL 文件的连续顺序，这些文件至少可扩展到上一次基础备份的开始时间。因此，基本备份的理想频率取决于：

- 归档 WAL 文件的存储卷。

- 需要恢复时数据恢复的最可能持续时间。如果自上次备份起已有较长时间，系统会重新执行更多的 WAL 段，因此恢复需要更长的时间。

注意

您不能将 pg_dump 和 pg_dumpall SQL 转储用作持续归档备份解决方案的一部分。SQL 转储生成逻辑备份，但所包含的信息不足以供WAL重新执行。

要使用持续归档方法执行数据库备份和恢复，请按照以下说明：

1. 设置并测试归档 WAL 文件的流程 - 请参阅 WAL 归档。

2. 执行基础备份 - 请参阅 基础备份。

要恢复您的数据，请遵循 使用持续归档恢复数据库 中的说明。

4.5.3.2. 持续归档的优点和缺陷

与其它 PostgreSQL 备份方法相比，持续归档具有以下优势：

- 使用持续备份方法时，可以使用不完全一致的基础备份，因为备份中的任何内部不一致都可以被重新执行日志所修正。因此，您可以在正在运行的 PostgreSQL 服务器上执行基础备份。

- 不需要文件系统快照； tar 或类似的归档工具就足够了。

- 持续备份可以通过继续归档 WAL 文件来实现，因为日志重播的 WAL 文件序列可能会无限期地延长。这对大型数据库尤其重要。

- 持续备份支持点恢复。不需要将 WAL 条目重新显示到结尾。可在任何时间点停止重新执

行，并且数据库可以恢复到执行基础备份以后的任何状态。

- 如果已经加载了相同的基础备份文件的另一台机器可以连续使用WAL文件系列，那么可以在任何时候用数据库几乎当前的副本来恢复其它机器。

与其他 PostgreSQL 备份方法相比，持续归档有以下缺点：

- 持续备份方法只支持恢复整个数据库集群，而不是子集。
- 持续备份需要广泛的归档存储。

4.5.3.3. 设置 WAL 归档

运行的 PostgreSQL 服务器会生成一系列预写日志(WAL)记录。服务器物理上将该序列分成 WAL 段文件，这些文件被指定了数字名称，以反映它们在 WAL 序列中的位置。如果不进行 WAL 归档，段文件将被重新使用，并被重命名为更高的段号。

在归档 WAL 数据时，在重用段文件之前，都会捕获每一个段文件的内容，并将其保存在一个新的位置。您有多个保存内容的选项，例如其他机器上的 NFS 挂载目录、磁带驱动器或 CD。

请注意，WAL 记录不包括对配置文件的修改。

要启用 WAL 归档，请使用以下流程：

流程

1. 在 /var/lib/pgsql/data/postgresql.conf 文件中：
   a. 将 wal_level 配置参数设置为 replica 或更高的值。
   b. 将 archive_mode 参数设置为 on。
   c.

在 archive_command 配置参数中指定 shell 命令。您可以使用 cp 命令、其它命令或 shell 脚本。

2. 重启 postgresql 服务以使修改生效：

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

3. 测试您的归档命令，并确保它不会覆盖现有的文件，如果失败，它将返回一个非零退出状态。

4. 要保护您的数据，请确保将段文件归档到不具有组或全局读权限的目录中。

注意

归档命令只对已完成的 WAL 段执行。生成小 WAL 流量的服务器在交易完成和其归档存储中的安全记录之间可能会有很长时间的延迟。要限制未归档数据可保留多久，您可以：

- 设置 archive_timeout 参数，来强制服务器以给定频率切换到新的 WAL 段文件。
- 使用 pg_switch_wal 参数强制段切换，以确保交易在完成后立即归档。

例 4.5. 用于归档 WAL 段的 shell 命令

本例显示了您可以在 archive_command 配置参数中设置的简单 shell 命令。

以下命令将完成的段文件复制到所需位置：

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

其中 %p 参数替换为归档文件的相对路径，%f 参数替换为文件名。

此命令将可归档的 WAL 段复制到 /mnt/server/archivedir/ 目录中。替换 %p 和 %f 参数后，执行的命令如下所示：

```
test ! -f /mnt/server/archivedir/00000001000000A900000065 && cp
pg_wal/00000001000000A900000065 /mnt/server/archivedir/00000001000000A900000065
```

对每个归档的新文件都会生成类似的命令。

其他资源

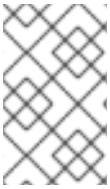
- PostgreSQL 16 文档

4.5.3.4. 进行基础备份

您可以通过多种方法创建基础备份：执行基础备份的最简单方法是在运行的 PostgreSQL 服务器上使用 pg_basebackup 工具。

基础备份进程会创建一个备份历史记录文件，该文件存储在 WAL 归档区，并以基础备份所需的第一个 WAL 段文件来命名。

备份历史记录文件是一个小文本文件，其包含开始和结束时间，以及备份的 WAL 段。如果您使用标签字符串来标识关联的转储文件，那么您可以使用备份历史记录文件来确定要恢复哪个转储文件。



注意

请考虑保留多个备份集，以确保您可以恢复数据。

要执行基础备份，请使用以下流程：

先决条件

- 您必须以 postgres 超级用户身份、具有数据库管理员特权的用户身份或至少具有 REPLICATION 权限的其他用户身份来运行命令。
- 您必须保留在基础备份期间和之后生成的所有 WAL 段文件。

流程

1. 使用 pg_basebackup 工具执行基础备份。

   - 将基础备份创建为单个的文件（纯格式）：

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

     使用您选择的备份位置替换 *backup_directory*。

     如果您在与服务器相同的主机上使用表空间并执行基础备份，那么也必须使用 --tablespace-mapping 选项，否则当试图将备份写入到同一位置时，备份将失败。

   - 将基础备份创建为一个 tar 归档（tar 和压缩格式）：

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

     使用您选择的备份位置替换 *backup_directory*。

     要恢复此数据，您必须手动提取正确位置中的文件。

2. 基础备份进程完成后，将备份历史记录文件中指定的数据库集群副本和备份过程中使用的 WAL 段文件进行安全归档。

3. 删除比基础备份中使用的 WAL 段文件数值更低的WAL段，因为这些比基础备份旧，并且不再需要进行恢复。

要指定serverpg_basebackup将与哪个数据库联系，请使用以下命令行选项：

- -h 选项用来定义主机的。

  默认主机要么是本地主机，要么是 PGHOST 环境变量所指定的主机。

- -p 选项用来定义端口。

  默认端口是由 PGPORT 环境变量或编译后的默认值指明的。

其他资源

- PostgreSQL 文档 - 基础备份
- PostgreSQL 文档 - pg_basebackup 工具

4.5.3.5. 使用持续归档备份来恢复数据库

要使用持续备份来恢复数据库，请使用以下流程：

流程

1. 停止服务器：

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

2. 将必要的数据复制到临时位置。

   最好复制整个集群数据目录和任何表空间。请注意，这需要系统上有足够的可用空间来保存现有数据库的两个副本。

   如果您没有足够的空间，就保存集群的pg_wal 目录的内容，其中可能包含系统关闭前没有归档的日志。

3. 删除集群数据目录下的所有现有文件和子目录，并在您要使用的任何表空间的根目录下删除。

4. 从您的基础备份恢复数据库文件。

   确保：

- 恢复的文件具有正确的所有权（数据库系统用户，而不是 root）。
- 恢复的文件具有正确的权限。
- pg_tblspc/ 子目录中的符号链接被正确恢复。

5. 删除 pg_wal/ 子目录中的任何文件。

这些文件源自基础备份，因此已过时。如果您没有归档 pg_wal/，请重新创建它，并使其具有正确的权限。

6. 将你在步骤 2 中保存的任何未归档的 WAL 段文件复制到 pg_wal/ 中。

7. 在集群数据目录中创建 restore.conf 恢复命令文件，并在 restore_command 配置参数中指定 shell 命令。您可以使用 cp 命令、其它命令或 shell 脚本。例如：

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

8. 启动服务器：

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

服务器将进入恢复模式，并继续读取所需的存档 WAL 文件。

如果恢复因为外部错误而终止，那么可以重启服务器，它将继续进行恢复。恢复过程完成后，服务器将 restore.conf 重命名为 restore.done。这可以防止服务器在启动正常的数据库操作后意外重新进入恢复模式。

9. 检查数据库的内容，确保数据库已恢复到所需的状态。

如果数据库尚未恢复到所需状态，请返回到第 1 步。如果数据库已恢复到所需的状态，那么通过恢复 pg_hba.conf 文件中的客户端身份验证配置来允许用户进行连接。

有关使用持续备份恢复的更多信息，请参阅 PostgreSQL 文档。

4.5.3.6. 其他资源

- 持续归档方法

4.6. 迁移到 RHEL 9 的 POSTGRESQL 版本

Red Hat Enterprise Linux 8 在多个模块流中提供 PostgreSQL ：PostgreSQL 10 （默认的 postgresql 流）、PostgreSQL 9.6、PostgreSQL 12、PostgreSQL 13、PostgreSQL 15 和 PostgreSQL 16。

在 RHEL 9 中，PostgreSQL 13、PostgreSQL 15 和 PostgreSQL 16 可用。

在 RHEL 中，您可以为数据库文件使用两个 PostgreSQL 迁移路径：

- 使用 pg_upgrade 工具快速升级
- 转储和恢复升级

快速升级方法比转储和恢复过程要快。然而，在某些情况下，快速升级无法正常工作，例如，当跨架构升级时，只能使用转储和恢复过程。

迁移到更新版本的 PostgreSQL 的先决条件是备份所有 PostgreSQL 数据库。

转储和恢复过程需要转储数据库并执行SQL文件备份，建议使用快速升级方法。

在迁移到 PostgreSQL 的后续版本前，请参阅您要迁移的 PostgreSQL 版本 的上游兼容性备注，以及您要迁移的版本和目标版本之间的所有跳过的 PostgreSQL 版本。

4.6.1. PostgreSQL 15 和 PostgreSQL 16 之间的显著区别

PostgreSQL 16 引入了以下显著变化。

postmasters 二进制文件不再可用

PostgreSQL 不再与 postmaster 二进制文件一起发布。使用提供的 systemd 单元文件(systemctl start postgres 命令)启动 postgresql 服务器的用户不受这个更改的影响。如果您之前通过 postmaster 二进制文件直接启动 postgresql 服务器，则现在必须使用 postgres 二进制文件。

文档不再被打包

PostgreSQL 不再提供软件包中的 PDF 格式文档。改为使用 在线文档。

4.6.2. PostgreSQL 13 和 PostgreSQL 15 之间的显著区别

PostgreSQL 15 包括以下向后不兼容更改：

公共模式的默认权限

在 PostgreSQL 15 中 public 模式的默认权限已被修改。新创建的用户需要使用 GRANT ALL ON SCHEMA public TO myuser; 命令明确授予权限。

以下示例在 PostgreSQL 13 及早期版本中正常工作：

```
postgres=# CREATE USER mydbuser;
postgres=# \c postgres mydbuser
postgres=$ CREATE TABLE mytable (id int);
```

以下示例在 PostgreSQL 15 及更高版本中工作：

```
postgres=# CREATE USER mydbuser;
postgres=# GRANT ALL ON SCHEMA public TO mydbuser;
postgres=# \c postgres mydbuser
postgres=$ CREATE TABLE mytable (id int);
```

注意

确保在 pg_hba.conf 文件中正确配置了 mydbuser 访问。如需更多信息，请参阅 创建 PostgreSQL 用户。

pipeline 模式下不再支持 PQsendQuery （）

从 PostgreSQL 15 开始，pipeline 模式中不再支持 libpq PQsendQuery（） 函数。修改受影响的应用程序，以使用 PQsendQueryParams（） 函数。

4.6.3. 使用 pg_upgrade 工具快速升级

作为系统管理员，您可以使用快速升级方法升级到 PostgreSQL 的最新版本。要执行快速升级，将二进制数据文件复制到 /var/lib/pgsql/data/ 目录中，并使用 pg_upgrade 工具。

您可以使用此方法迁移数据：

- 从 RHEL 8 的 PostgreSQL 12 版本到 PostgreSQL 13的 RHEL 版本
- 从 RHEL 8 或 9 版本的 PostgreSQL 13 到 RHEL 的 PostgreSQL 15
- 从 RHEL 8 或 9 的 PostgreSQL 15 版本到 PostgreSQL 16的 RHEL 版本

以下流程描述了使用快速升级方法从 RHEL 8 版本的 PostgreSQL 12 迁移到 RHEL 9 版本的 PostgreSQL 13。对于从 12 以外的 postgresql 流进行迁移，请使用以下方法之一：

- 将 RHEL 8 上的 PostgreSQL 服务器更新至版本 12，然后使用 pg_upgrade 程序执行一个到 RHEL 9 版本的 PostgreSQL 13 的快速升级。
- 使用 dump 和 restore 在 PostgreSQL 的任何 RHEL 8 版本和 RHEL 9 中相同或更新的 PostgreSQL 版本之间进行升级。

先决条件

- 在执行升级前，请备份存储在 PostgreSQL 数据库中的所有数据。默认情况下，所有数据都存储在 RHEL 8 和 RHEL 9 系统的 /var/lib/pgsql/data/ 目录中。

流程

1. 在 RHEL 9 系统中，安装 postgresql-server 和 postgresql-upgrade 软件包：

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

另外，如果您在 RHEL 8 上使用了任何 PostgreSQL 服务器模块，那么也可以在 RHEL 9 系统上安装该模块的两个版本，分别针对 PostgreSQL 12 （作为 postgresql-upgrade 软件包安装）和 PostgreSQL 13 的目标版本（作为 postgresql-server 软件包安装）进行编译。如果您需要编译第三方PostgreSQL服务器模块，请根据postgresql-devel和postgresql-upgrade-devel软件包来构建它。

2. 检查以下项：

- 基本配置：在 RHEL 9 系统中，检查您的服务器是否使用默认 /var/lib/pgsql/data 目录，且数据库已正确初始化并启用。此外，数据文件必须存储在 /usr/lib/systemd/system/postgresql.service 文件中提及的相同路径。
- PostgreSQL 服务器：您的系统可以运行多个 PostgreSQL 服务器。确保所有这些服务器的数据目录都是独立处理的。
- PostgreSQL 服务器模块：确保在 RHEL 8 中使用的 PostgreSQL 服务器模块也安装在 RHEL 9 系统中。请注意，插件安装在 /usr/lib64/pgsql/ 目录中。

3. 确保 postgresql 服务在复制数据时未在源和目标系统上运行。

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

4. 将源位置中的数据库文件复制到 RHEL 9 系统上的 /var/lib/pgsql/data/ 目录中。

5. 以 PostgreSQL 用户身份运行以下命令来执行升级过程：

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

这会在后台启动 pg_upgrade 进程。

在出现故障时，postgresql-setup 会提供一条说明性的错误消息。

6. 将之前的配置从 /var/lib/pgsql/data-old 复制到新集群。

请注意，快速升级不会在较新的数据栈中重用之前的配置，配置是从零开始生成的。如果要手动组合旧配置和新配置，请使用数据目录中的 *.conf 文件。

7. 启动新的 PostgreSQL 服务器：

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

8. 分析新的数据库集群。

- 对于 PostgreSQL 13 ：

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

- 对于 PostgreSQL 15 或更高版本：

```
su postgres -c 'vacuumdb --all --analyze-in-stages'
```

注意

您可能需要使用 ALTER COLLATION 名称 REFRESH VERSION，请查看 上游文档 了解详细信息。

9. 如果您希望新 PostgreSQL 服务器在引导时自动启动，请运行：

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

4.6.4. 转储和恢复升级

使用转储和恢复升级时，您必须将所有的数据库内容转储到 SQL 文件转储文件中。请注意，转储和恢复升级比快速升级方法慢，可能需要在生成的 SQL 文件中进行一些手动修复。

您可以使用此方法将数据从 PostgreSQL 的任何 RHEL 8 版本迁移到 RHEL 9 中 PostgreSQL 的任何相等或更新版本。

在 RHEL 8 和 RHEL 9 系统中，PostgreSQL 数据默认存储在 /var/lib/pgsql/data/ 目录中。

要执行转储和恢复升级，请将用户改为 root。

以下流程描述了从 PostgreSQL 10 的 RHEL 8 的默认版本迁移到 PostgreSQL 13 的 RHEL 9 版本。

流程

1. 在 RHEL 8 系统中，启动 PostgreSQL 10 服务器：

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

2. 在 RHEL 8 系统中，将所有数据库内容转储到 pgdump_file.sql 文件中：

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

3. 确保正确转储数据库：

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

结果显示的转储的 sql 文件的路径为：/var/lib/pgsql/pgdump_file.sql。

4. 在 RHEL 9 系统中，安装 postgresql-server 软件包：

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

另外，如果您在 RHEL 8 中使用了任何 PostgreSQL 服务器模块，也需要在 RHEL 9 系统中安装它们。如果您需要编译第三方 PostgreSQL 服务器模块，请根据 postgresql-devel 软件包进行构建。

-

5. 在 RHEL 9 系统中，初始化新 PostgreSQL 服务器的数据目录：

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

6. 在 RHEL 9 系统中，将 pgdump_file.sql 复制到 PostgreSQL 主目录中，并检查是否已正确复制该文件：

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

7. 复制 RHEL 8 系统中的配置文件：

```
su - postgres -c 'ls -1 $PGDATA/*.conf'
```

要复制的配置文件包括：

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

8. 在 RHEL 9 系统中，启动新的 PostgreSQL 服务器：

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

9. 在 RHEL 9 系统中，从转储的 sql 文件中导入数据：

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