

How to Manage Password Policies in MySQL 8

Learn how to effectively manage password policies in MySQL 8, including setting password expiration, complexity requirements, and security best practices for your database.

Contents

01	Introduction	3
02	Prerequisites	3
03	1. Create a Sample User	3
04	2. Manage the Password Reuse Policy	4
05	3. Manage the Password Expiry Policy	7
06	4. Change the Current Password Verification Policy	11
07	5. Use Random Password Generator	14
08	6. Enable Failed Login Tracking and Temporarily Lock Accounts	15
09	Conclusion	16

Introduction

MySQL is the world's most popular open-source database management system that offers on-demand scalability, high performance, complete workflow control, round-the-clock uptime guarantee, and reduced cost of ownership.

MySQL 8 ships with fine-grained data security features, including robust authentication and authorization features, to protect your most crucial data, especially when working in a multi-user environment.

In this guide, you'll explore the different users' password management capabilities that MySQL offers, including password reuse policy, current password verification policy, random password generator, failed login tracking, and temporary account locking.

Prerequisites

To test this MySQL password policies guide, make sure you've the following:

- An [Ubuntu 20.04 server](#).
- A non-root user with `sudo` privileges.
- A [MySQL 8 Server](#).

1. Create a Sample User

Log in to your MySQL server as root.

```
$ sudo mysql -u root -p
```

Key in the root password of your MySQL server and press Enter to proceed. Then, issue the command below to create a sample user.

```
mysql> CREATE USER 'john'@'localhost'  
IDENTIFIED BY 'EXAMPLE_PASSWORD';
```

Output:

```
Query OK, 0 rows affected (0.01 sec)
```

Exit from the MySQL command-line interface.

```
mysql> QUIT;
```

Output:

```
Bye
```

With the sample user in place, you'll now restrict the use of previous passwords in the next step.

2. Manage the Password Reuse Policy

Starting from MySQL 8.0.13, you can restrict users from reusing old passwords to tighten the security if the previous passwords have been compromised.

You can control the default password reuse policy by explicitly setting the `password_history` and `password_reuse_interval` global variables in your MySQL configuration file.

The variable `password_history` determines the number of unique passwords you must set after each password before reusing it. Then, the `password_reuse_interval` defines the time it takes in days before you can reuse an old password.

To set these two variables, open your MySQL configuration files using nano.

```
$ sudo nano /etc/mysql/my.cnf
```

Then, enter the information below into the file to allow users to reuse an old password only after setting three new passwords. Also, restrict users from reusing old passwords that have been used for the last 365 days.

```
...  
[mysqld]  
password_history=3  
password_reuse_interval=365
```

Save and close the file. Then, restart the MySQL server to load the new configurations.

```
$ sudo systemctl restart mysql
```

Now, log in to the MySQL server as `john` to test the new settings.

```
$ sudo mysql -u john -p
```

Enter your password and press Enter to continue. Once you get the `mysql>` prompt, try to set your new password to your current password by executing the statement below twice.

```
mysql> ALTER USER 'john'@'localhost' IDENTIFIED BY 'EXAMPLE_PASSWORD';
```

Output.

```
ERROR 3638 (HY000): Cannot use these credentials for 'john@localhost' because they  
contradict the password history policy
```

Log out from the MySQL server.

```
mysql> QUIT;
```

Please note changing the configuration file establishes the password-reuse policy globally. If you want to set up the policy for each user account independently or you would probably want to override the values you've set in the global variables, define the `PASSWORD HISTORY` and the `PASSWORD REUSE INTERVAL` when executing the `CREATE USER` and `ALTER USER` statements.

To do this, log in to MySQL as root.

```
$ sudo mysql -u root -p
```

Enter your password and press Enter to proceed. Then, create a new user named `mary` and set the `PASSWORD HISTORY` to `5` and `PASSWORD REUSE INTERVAL` to `180 DAY`.

```
mysql> CREATE USER 'mary'@'localhost'  
        PASSWORD HISTORY 5  
        PASSWORD REUSE INTERVAL 180 DAY;
```

Output.

```
Query OK, 0 rows affected (0.01 sec)
```

To update the password-reuse policy of an already existing user, for example, `john`, use the syntax below.

```
mysql> ALTER USER 'john'@'localhost'  
        PASSWORD HISTORY 5  
        PASSWORD REUSE INTERVAL 180 DAY;
```

Output.

```
Query OK, 0 rows affected (0.01 sec)
```

The above commands should override any password reuse policy that you've set globally. You can confirm the current settings for each user by executing the command below against the `mysql.user` table.

```
mysql> SELECT  
        User,  
        Password_reuse_history,  
        Password_reuse_time  
        FROM mysql.user;
```

Output.

```
+-----+-----+-----+
| User          | Password_reuse_history | Password_reuse_time |
+-----+-----+-----+
...
| john          |          5 |          180 |
| mary         |          5 |          180 |
...
+-----+-----+-----+
9 rows in set (0.00 sec)
```

Exit from the MySQL server.

```
mysql> QUIT;
```

After restricting the reuse of the most previous passwords, you'll now configure a policy that requires passwords to expire after a given time.

3. Manage the Password Expiry Policy

You can set the MySQL server to require users' passwords to be changed periodically. Every user will be required to change their passwords after a set period, for instance, 90 days.

To track expired passwords, MySQL maintains a timestamp that indicates when a user last changed their password. Therefore, when a user connects to the MySQL server with a password older than its permitted lifetime, they must update it.

To set the password-expiration policy globally, define a `default_password_lifetime` variable and set its value to the number of days that passwords should last in the MySQL configuration file.

Open the MySQL configuration file using `nano`.

```
$ sudo nano /etc/mysql/my.cnf
```

Then, enter the information below into the file to allow passwords to expire after 90 days.

```
...
[mysqld]
default_password_lifetime=90
```

If you want to establish a policy that allows passwords never to expire, set the `default_password_lifetime` to 0.

```
...
[mysqld]
default_password_lifetime=0
```

Save and close the file. Then restart MySQL to load the new changes.

```
$ sudo systemctl restart mysql
```

You may override the global password-expiration policy for individual accounts by defining a `PASSWORD EXPIRE` option when executing the `CREATE USER` and `ALTER USER` statements.

To make the settings for each user, log in to MySQL as root.

```
$ sudo mysql -u root -p
```

Key in your root password and press Enter to continue. Then, to create a user named `smith` with a password expiration policy of 90 days, execute the statement below.

```
mysql> CREATE USER 'smith'@'localhost' PASSWORD EXPIRE INTERVAL 90 DAY;
```

Output.

```
Query OK, 0 rows affected (0.01 sec)
```

To create the same user with a never-expiring password, execute the command below.

```
mysql> CREATE USER 'smith'@'localhost' PASSWORD EXPIRE NEVER;
```

Output.

```
Query OK, 0 rows affected (0.01 sec)
```

To change the password expiration policy for an already existing user on the MySQL server, use the `ALTER USER` statement. For instance, change the `PASSWORD EXPIRE INTERVAL` for user `john` to `365` days by executing the command below.

```
mysql> ALTER USER 'john'@'localhost' PASSWORD EXPIRE INTERVAL 365 DAY;
```

To disable the password-expiration policy, use the command below.

```
mysql> ALTER USER 'john'@'localhost' PASSWORD EXPIRE NEVER;
```

Output.

```
Query OK, 0 rows affected (0.00 sec)
```

You can also manually expire a user account password by using the `PASSWORD EXPIRE` option in an `ALTER USER` statement. For example, execute the command below to mark `john's` password as expired.

```
mysql> ALTER USER 'john'@'localhost' PASSWORD EXPIRE;
```

Once a password has expired either manually or automatically, you must reset it the next time you log in to the MySQL server. In case you execute any command before resetting it, you'll get the error below.

```
ERROR 1820 (HY000): You must reset your password using ALTER USER statement before executing this statement.
```

To reset your password as a non-privileged user, log in to your MySQL server with the target user account and execute the statement below. Replace `john'@'localhost` with the correct username and `EXAMPLE_PASSWORD_2` with a strong value.

```
mysql> ALTER USER 'john'@'localhost' IDENTIFIED BY 'EXAMPLE_PASSWORD_2';
```

As a root user, you can confirm each user's current password expiry status by executing the statement below against the `mysql.user` table.

```
mysql> SELECT
      User,
      password_expired
FROM mysql.user;
```

Output.

```
+-----+-----+
| User          | password_expired |
+-----+-----+
...
| john          | N                |
| mary          | N                |
| smith         | N                |
| smith2        | N                |
+-----+-----+
11 rows in set (0.00 sec)
```

Exit from the MySQL server.

```
mysql> QUIT;
```

You've now defined MySQL passwords to expire as required. Next, you'll explore a MySQL policy that requires users to confirm their current passwords before changing them.

4. Change the Current Password Verification Policy

Starting from MySQL 8.0.13, you can force non-privileged users to provide their current password when changing their account passwords.

This option restricts anyone who might hijack a MySQL user's session from altering their password unless they know the correct credentials. By default, this feature is disabled, and you must turn it on either globally or individually.

To enable the password verification feature globally, open the MySQL configuration file.

```
$ sudo nano /etc/mysql/my.cnf
```

Then, enter the information below into the file.

```
...  
[mysqld]  
password_require_current=ON
```

Save and close the file. Then restart the MySQL server.

```
$ sudo systemctl restart mysql
```

You've now enabled the `password_require_current` globally. To allow password verification for individual users, use the `PASSWORD REQUIRE CURRENT` option when executing the `CREATE USER` and `ALTER USER` statements. Settings for individual accounts override the global settings. To set this, first, log in to the MySQL server as root.

```
$ sudo mysql -u root -p
```

Enter your password and press Enter to proceed. Then, define a user named `james` with the `password verification-required` policy enabled using the `PASSWORD REQUIRE CURRENT` option.

```
mysql> CREATE USER 'james'@'localhost'  
        IDENTIFIED WITH mysql_native_password BY 'EXAMPLE_PASSWORD'  
        PASSWORD REQUIRE CURRENT;
```

Output.

```
Query OK, 0 rows affected (0.00 sec)
```

To disable the `password verification-required` policy when creating a new user, for example, `roe`, use the `PASSWORD REQUIRE OPTIONAL` option.

```
mysql> CREATE USER 'roe'@'localhost'  
        IDENTIFIED WITH mysql_native_password BY 'EXAMPLE_PASSWORD'  
        PASSWORD REQUIRE CURRENT OPTIONAL;
```

Output.

```
Query OK, 0 rows affected (0.00 sec)
```

For existing users, use the commands below to enable or disable the `require-current-password` policy.

- Turn the `password verification-required` policy to `OFF` for user `james`.

```
mysql> ALTER USER 'james'@'localhost' PASSWORD REQUIRE CURRENT OPTIONAL;
```

Output.

```
Query OK, 0 rows affected (0.01 sec)
```

- Turn the `password verification-required` policy to `ON` for user `james`.

```
mysql> ALTER USER 'james'@'localhost' PASSWORD REQUIRE CURRENT;
```

Output.

```
Query OK, 0 rows affected (0.01 sec)
```

Exit from the MySQL server.

```
mysql> QUIT;
```

Now, log in to the MySQL server as user `james`.

```
$ sudo mysql -u james -p
```

Enter the password for user `james` and press Enter to proceed. Now, try changing the current password for user `james` by executing the statement below.

```
mysql> ALTER USER 'james'@'localhost' IDENTIFIED BY 'NEW_PASSWORD';
```

Since you're required to enter your current password, you should get the error below.

```
ERROR 3892 (HY000): Current password needs to be specified in the REPLACE clause to change it.
```

Try defining a wrong password using the `REPLACE` option.

```
mysql> ALTER USER 'james'@'localhost' IDENTIFIED BY 'NEW_PASSWORD' REPLACE  
'WRONG_OLD_PASSWORD';
```

Again, the command will fail with an `incorrect current password` error.

```
ERROR 3891 (HY000): Incorrect current password. Specify the correct password which has  
to be replaced.
```

Execute the statement this time around with the correct, current password.

```
mysql> ALTER USER 'james'@'localhost' IDENTIFIED BY 'NEW_PASSWORD' REPLACE
'EXAMPLE_PASSWORD';
```

When your password change is successful, you should get the following output.

```
Query OK, 0 rows affected (0.01 sec)
```

Exit from the MySQL server.

```
mysql> QUIT;
```

Password verification is now working as expected. Next, you'll use MySQL to generate strong passwords for new and existing accounts.

5. Use Random Password Generator

Starting from MySQL 8.0.18, you can generate strong passwords for users using an in-built random password generator. By default, the generated password is 20 characters long and is controlled by the `generated_random_password_length` system variable with a range of 5 to 255 characters.

To test the password generator, log in to MySQL server as root.

```
$ sudo mysql -u root -p
```

Then, enter your password and hit Enter to proceed. Next, execute the command below to create a user named `peter` with a random password.

```
mysql> CREATE USER 'peter'@'localhost'
IDENTIFIED BY RANDOM PASSWORD;
```

MySQL creates the user and displays the generated password as follows.

```
+-----+-----+-----+
| user   | host       | generated password |
+-----+-----+-----+
```

```
| peter | localhost | 8<_[:DU9@IMmGVll;/y |
+-----+-----+-----+
1 row in set (0.01 sec)
```

To alter the password for the user, use the syntax below.

```
mysql> ALTER USER 'peter'@'localhost' IDENTIFIED BY RANDOM PASSWORD;
```

The login credentials for user `peter` should be updated and displayed as shown below.

```
+-----+-----+-----+
| user  | host      | generated password  |
+-----+-----+-----+
| peter | localhost | 5L+DT;W2Fq!YQAQDku,* |
+-----+-----+-----+
1 row in set (0.01 sec)
```

Next, you'll set MySQL to track failed login attempts and temporarily or permanently ban user accounts connecting to the server with the wrong credentials.

6. Enable Failed Login Tracking and Temporarily Lock Accounts

Starting from MySQL 8.0.19, you can enable failed login tracking and temporarily lock accounts that violate the rule. For example, when a client fails to provide the correct password in a connection attempt, you can either lock their account for a few days or permanently disable it until you unlock it.

To test this, create a user named `bob` and define the `FAILED_LOGIN_ATTEMPTS` to `3`. Also, set the `PASSWORD_LOCK_TIME` to `3`. In simple terms, the statement below locks `bob's` account for `3` days after `3` failed logins.

```
mysql> CREATE USER 'bob'@'localhost'
        IDENTIFIED BY 'EXAMPLE_PASSWORD'
        FAILED_LOGIN_ATTEMPTS 3 PASSWORD_LOCK_TIME 3;
```

Output.

```
Query OK, 0 rows affected (0.00 sec)
```

To change the settings for an already existing account, use the syntax below.

```
mysql> ALTER USER 'bob'@'localhost'  
        FAILED_LOGIN_ATTEMPTS 5 PASSWORD_LOCK_TIME 10;
```

To completely lock an account when the set failed login attempts have been violated, set the `PASSWORD_LOCK_TIME` to `UNBOUNDED`.

```
mysql> ALTER USER 'bob'@'localhost'  
        FAILED_LOGIN_ATTEMPTS 5 PASSWORD_LOCK_TIME UNBOUNDED;
```

Next, in a new session, try logging in with `bob` credentials.

```
$ mysql -u bob -p
```

Repeat the login process with some wrong passwords until you get the error below.

```
Access denied for user 'bob'@'localhost' (using password: YES)  
...  
ERROR 3955 (HY000): Access denied for user 'bob'@'localhost'. Account is blocked for  
unlimited day(s) (unlimited day(s) remaining) due to 3 consecutive failed logins.
```

The error shows that the account for the user has been locked.

Conclusion

In this tutorial, you've tightened the security of your MySQL users' accounts by setting strict rules on the reuse of previous passwords.

You've also enabled password expiry, password-verification policy and used the MySQL inbuilt random password generator to set up strong passwords.

In the end, you're able to track and lock accounts violating the defined failed logins. Use the knowledge in this guide to safeguard the most critical data in your MySQL projects.



VULTR

