# Upgrading MySQL 5.7 to 8.0 on CentOS/RHEL

[Databases](https://www.webhi.com/how-to/tutorial/database/), [Linux system administration](https://www.webhi.com/how-to/tutorial/linux-sysadmin/)



Upgrading major versions of MySQL can be a complex process due to potential compatibility issues and configuration changes between versions. In this tutorial, we will walk through the full process of upgrading from MySQL 5.7 to 8.0 on CentOS/RHEL 7/8 servers.

The key steps we will cover are:

* Preparing for the upgrade by backing up data and validating config
* Installing the new MySQL 8.0 repositories
* Stopping and removing the old MySQL 5.7 packages
* Installing the new MySQL 8.0 packages
* Starting MySQL 8.0 and running post-installation steps
* Validating the upgrade and making any needed configuration changes

Following this process will help ensure a smooth and successful upgrade to MySQL 8.0.

## Prerequisites

Before starting the upgrade process, make sure your system meets the following prerequisites:

* CentOS/RHEL 7 or 8 operating system
* Existing installation of MySQL 5.7 from default CentOS/RHEL repos
* All important MySQL data has been backed up
* You have a maintenance window to take MySQL offline during the upgrade
* You have root or sudo privileges on the server to install software

## Step 1 – Backup and Prepare

To avoid any data loss, it is crucial to make backups before starting the upgrade process. Here are the key backup steps:

### Backup MySQL Data

Use mysqldump to create logical backups of all databases:

$ mysqldump --all-databases --routines --events > mysql\_backup.sql

Store this backup file in a safe location off the server.

You can also take binary backups like file system snapshots or LVM snapshots as additional protection.

### Review and Backup Configuration

The main MySQL config files that may need updates after the upgrade are:

* /etc/my.cnf – The main MySQL config file
* /etc/my.cnf.d/ – Contains config snippets that override my.cnf
* /var/lib/mysql/ – Contains database files

Review these files and back them up to a safe location. Pay particular attention to any custom settings like log file paths, memory allocation, etc.

After the upgrade, we will need to merge any needed custom configs back into the new files.

### Record All MySQL Users

The upgrade could potential affect user accounts and permissions. Capture a list of all current users by running:

mysql> SELECT User, Host FROM mysql.user;

This will let you recreate or fix any users if necessary after the upgrade.

### Test and Validate Replication (if used)

If using MySQL replication, make sure replication is healthy and all servers are fully in sync before starting the upgrade.

Perform some test writes and reads on the master and slaves to validate. Resolve any replication issues before proceeding.

## Step 2 – Install MySQL 8.0 Repos

Now we are ready to setup the new MySQL 8.0 repositories. This will let us install the latest MySQL 8.0 releases using yum.

First install the MySQL GPG key:

$ rpm --import <https://repo.mysql.com/RPM-GPG-KEY-mysql-2023>

This allows RPM packages from MySQL to be authenticated.

Next install the MySQL community repo package:

$ yum install <https://repo.mysql.com/mysql80-community-release-el7-3.noarch.rpm>

This will setup the MySQL 8.0 yum repo configuration.

Refresh your repo data:

$ yum makecache

Your CentOS/RHEL system can now install MySQL 8.0.

## Step 3 – Stop MySQL 5.7 and Remove Packages

Now we need to properly stop MySQL 5.7 and remove the old package files.

### Stop MySQL Service

Stop the MySQL service:

$ systemctl stop mysqld.service

Verify it shut down properly:

$ systemctl status mysqld.service

### Remove Old MySQL Packages

Next use yum to remove just the MySQL 5.7 packages:

$ yum remove mysql-community-client-5.7\* mysql-community-common-5.7\* mysql-community-libs-5.7\* mysql-community-server-5.7\*

This will leave config files in place, which we will clean up later.

## Step 4 – Install MySQL 8.0

With the old version removed, we can now install MySQL 8.0:

$ yum install mysql-community-server

This will pull in the latest MySQL 8.0 server, client, common libraries, and tools.

When the install finishes, MySQL 8.0 is now in place but not fully configured yet.

## Step 5 – Start MySQL 8.0 and Run mysql\_upgrade

The next steps are to start MySQL 8.0 and run mysql\_upgrade to make any needed changes for the new version.

### Start MySQL

Start the new MySQL 8.0 service:

$ systemctl start mysqld.service

Verify it started successfully:

$ systemctl status mysqld.service

The service should be running and you can now connect to it locally.

### Run mysql\_upgrade

The mysql\_upgrade tool will check your databases and make any needed adjustments like changing the data dictionary format. This is a key part of the upgrade process.

Connect to MySQL and run it:

$ mysql\_upgrade -u root -p

Enter your root password when prompted. This will run through any updates needed to your data for 8.0 compatibility.

When it finishes run this to verify the data dictionary is updated:

mysql> SHOW VARIABLES LIKE 'dd\_version';

It should show a value like ‘2’ for MySQL 8.0.

### Review Error Log

Check over the last MySQL error log at /var/log/mysqld.log for any issues reported during startup or mysql\_upgrade.

Resolve any errors before continuing.

At this point MySQL 8.0 should be running successfully.

## Step 6 – Update Configuration Files

Next we need to update any configuration files and settings for MySQL 8.0.

### Update my.cnf

The main my.cnf file is likely outdated and missing new 8.0 specific options.

Create a new default version:

$ mysqld --initialize --user=mysql --basedir=/usr --datadir=/var/lib/mysql

This will generate a new my.cnf under /var/lib/mysql with 8.0 defaults.

Copy it to /etc replacing the old file:

$ cp /var/lib/mysql/my.cnf /etc/my.cnf

Open up this new /etc/my.cnf file and re-apply any custom settings you previously had, such as:

* Custom memory and engine settings
* Replication settings
* Any other customizations

Save the updated file when finished.

### Update App Armor Settings

If using AppArmor security, it needs updated settings for MySQL 8.0:

$ aa-disable /etc/apparmor.d/usr.sbin.mysqld

Then update it:

$ echo "alias /etc/apparmor.d/usr.sbin.mysqld -> /etc/apparmor.d/disable/" | tee -a /etc/apparmor.d/usr.sbin.mysqld

This disables the old AppArmor rules and will use the defaults.

### Restart MySQL and Validate Config

Restart MySQL to load the new my.cnf settings:

$ systemctl restart mysqld.service

Check that custom settings have applied correctly:

mysql> SHOW VARIABLES;

Verify your memory, storage engine, and other settings are as expected.

Also check that:

mysql> SELECT @@global.innodb\_default\_row\_format;

Shows ‘dynamic’ which is required for MySQL 8.0.

Adjust my.cnf further if any settings did not apply correctly.

## Step 7 – Create New User Accounts

If you had any custom user accounts in MySQL 5.7, they will need recreated in 8.0.

Check the list of users you captured earlier, and recreate each account using new CREATE USER syntax:

# Create app user  
mysql> CREATE USER 'myapp'@'localhost' IDENTIFIED BY 'password';   
# Grant permissions   
mysql> GRANT SELECT,INSERT,UPDATE,DELETE ON myapp.\* TO 'myapp'@'localhost';

Test connecting with each recreated user to validate permissions are working.

You may also want to force password expiration with:

mysql> ALTER USER 'user'@'host' PASSWORD EXPIRE;

This will require users to reset their password on first login.

## Step 8 – Cleanup Old Installation

Once the upgrade is complete, we can remove any remaining artifacts from the old MySQL 5.7 installation:

### Remove Log Files

Clear out old log files from 5.7:

$ rm /var/log/mysqld.log\*

### Remove Config Files

Remove any old unused 5.7 config files:

$ rm /etc/my.cnf.d/\*mysql57\*

### Remove Data Directories

If not reusing existing data dirs, remove the 5.7 data directory:

$ rm -rf /var/lib/mysql57

This helps eliminate any confusion over multiple data directories.

### Restart and Monitor MySQL

Restart MySQL one last time:

$ systemctl restart mysqld.service

Monitor logs and performance for a while to ensure stable behavior.

The upgrade to 8.0 is now fully complete!

## Conclusion

Upgrading from MySQL 5.7 to 8.0 involves careful planning, preparation, and testing. The key steps are:

* Backup all data and validate the existing install
* Install the new MySQL 8.0 repositories
* Stop and remove older 5.7 packages
* Install MySQL 8.0 releases
* Run mysql\_upgrade to update system databases
* Update configuration files like my.cnf
* Recreate user accounts and rerun tests
* Cleanup old install artifacts and logs

Following this process methodically helps avoid issues and ensure a smooth transition to the latest MySQL version. Testing thoroughly validates that applications have compatibility with new behaviors.

Your MySQL infrastructure is now upgraded with the latest features and improvements in MySQL 8.0!