**MY SQL Installation in UBUNTU 22.04**

**Reference Website Links**

https://phoenixnap.com/kb/how-to-install-mysql-on-ubuntu-18-04

https://www.digitalocean.com/community/tutorials/how-to-install-mysql-on-ubuntu-20-04

https://www.cyberciti.biz/faq/installing-mysql-server-on-ubuntu-22-04-lts-linux/

1. **Login to CLI Mode**
2. **sudo apt-get update**
3. **sudo apt-get upgrade**
4. **sudo apt install mysql-server**
5. **Do you want to continue : Y**
6. **sudo mysql -u root –p**
7. **or (sudo mysql)**
8. **ALTER** **USER** 'root'@'localhost' **IDENTIFIED** **WITH** mysql\_native\_password **BY** '**Synaptic@123'**;
9. **Exit**
10. sudo mysql\_secure\_installation
11. **Enter password for user root:xxxxx**
12. Do you want to continue with the password provided ? **Y**
13. **Please enter 0 = LOW, 1 = MEDIUM and 2 = STRONG: 2**
14. **Change the password for root ? ((Press y|Y for Yes, any other key for No) : Y**
15. **Enter the password: Synaptic@123**
16. **Reenter the password: Synaptic@123**
17. **Do you wish to continue with the password provided?(Press y|Y for Yes, any other key for No) : Y**
18. Remove anonymous users? **Y**
19. Remove test database and access to it? **N**
20. Disallow root login remotely? **N**
21. **Reload privilege tables now? (Press y|Y for Yes, any other key for No) : Y**
22. sudo systemctl is-enabled mysql.service
23. sudo systemctl status mysql.service
24. sudo systemctl stop mysql.service
25. sudo systemctl start mysql.service
26. sudo mysql -u root -p
27. sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf
28. Exit
29. Change the bind-address ip from 127.0.0.1 to 0.0.0.0
30. Ctrl+X and hit Y to save the file
31. sudo systemctl restart mysql
32. sudo mysql -u root –p
33. CREATE USER '**nabard**'@'%' IDENTIFIED BY '**Synaptic@123'**;
34. GRANT ALL PRIVILEGES ON \* . \* TO '**nabard**'@'%';
35. FLUSH PRIVILEGES;
36. exit
37. sudo ufw allow mysql
38. **sudo systemctl restart mysql**

**MY SQL UnInstallation in UBUNTU 22.04**

1. **sudo systemctl status mysql**
2. **sudo systemctl stop mysql**
3. **sudo systemctl status mysql**
4. **sudo apt purge mysql-server\***

**Or**

**You may use this command to ensure that any kind of MySQL installed on your system is uninstalled**

1. **sudo apt purge mysql-server mysql-client mysql-common mysql-server-core-\* mysql-client-core-\***
2. **ls /etc/mysql**
3. **sudo ls /var/lib/mysql**
4. **sudo rm -r /etc/mysql /var/lib/mysql**
5. **sudo rm -r /var/log/mysql**
6. **sudo apt autoremove**

[**https://www.digitalocean.com/community/tutorials/how-to-move-a-mysql-data-directory-to-a-new-location-on-ubuntu-20-04**](https://www.digitalocean.com/community/tutorials/how-to-move-a-mysql-data-directory-to-a-new-location-on-ubuntu-20-04)

## [Step 1 — Moving the MySQL Data Directory](https://www.digitalocean.com/community/tutorials/how-to-move-a-mysql-data-directory-to-a-new-location-on-ubuntu-20-04#step-1-moving-the-mysql-data-directory)

To prepare for moving MySQL’s data directory, let’s verify the current location by starting an interactive MySQL session using the administrative credentials. Run the following command to open the MySQL server prompt:

1. sudo mysql

Copy

**Note**: If you configured your **root** MySQL user to authenticate using a password, you can connect to MySQL as this user with the following command:

1. mysql -u root -p

Copy

When prompted, supply the MySQL user password. Then from the MySQL prompt, run the following SELECT statement. This will return this MySQL instance’s active data directory, which is always recorded in MySQL’s datadir variable::

1. SELECT @@datadir;

Copy

Output

+-----------------+

| @@datadir |

+-----------------+

| /var/lib/mysql/ |

+-----------------+

1 row in set (0.00 sec)

This output confirms that MySQL is configured to use the default data directory, /var/lib/mysql/, so that’s the directory you need to move. Once you’ve confirmed this, write exit to leave the monitor and return to your command prompt:

1. exit

Copy

Output

Bye

To ensure the integrity of the data, shut down MySQL before making changes to the data directory:

1. sudo systemctl stop mysql

Copy

Note that systemctl doesn’t display the outcome of all service management commands, so if you want to check if you’ve succeeded, use the following command:

1. sudo systemctl status mysql

Copy

You can confirm it’s shut down if the Active line in the output states it’s inactive (dead) as highlighted in the following example:

Output

● mysql.service - MySQL Community Server

Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset:>

Active: inactive (dead) since Wed 2022-03-23 19:03:49 UTC; 5s ago

Process: 3415 ExecStart=/usr/sbin/mysqld (code=exited, status=0/SUCCESS)

Main PID: 3415 (code=exited, status=0/SUCCESS)

Status: "Server shutdown complete"

Now that the server is shut down, you can copy the existing database directory, /var/lib/mysql, to the new location, /mnt/volume-nyc1-01, with rsync. Using the -a flag preserves the permissions and other directory properties, while-v provides verbose output so you can follow the progress:

**Note:** Be sure there is no trailing slash on the directory, which may be added if you use tab completion. When there’s a trailing slash, rsync will dump the contents of the directory into the mount point instead of transferring it into a containing mysql directory.

1. sudo rsync -av /var/lib/mysql /mnt/volume-nyc1-01

We need to create folders in /data/erpdb/

sudo rsync -av /var/lib/mysql /data/erpdb/

CopyOnce the rsync command is complete, rename the current folder with a .bak extension and keep it until you’ve confirmed the move was successful. By renaming it, you’ll avoid confusion that could arise from files in both the new and the old location:

1. sudo mv /var/lib/mysql /var/lib/mysql.bak

Copy

Now you’re ready to proceed with the next step and begin configuration.

## [Step 2 — Pointing to the New Data Location](https://www.digitalocean.com/community/tutorials/how-to-move-a-mysql-data-directory-to-a-new-location-on-ubuntu-20-04#step-2-pointing-to-the-new-data-location)

MySQL has several ways to override configuration values. By default, the datadir is set to /var/lib/mysql in the /etc/mysql/mysql.conf.d/mysqld.cnf file. Edit this file in your preferred text editor to reflect the new data directory. Here we’ll use nano:

1. sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf

Copy

Find the line that begins with datadir=. Uncomment the line by deleting the pound sign (#) and change the path to reflect the new location. In this case, the updated file contents will be follows:

/etc/mysql/mysql.conf.d/mysqld.cnf

. . .

datadir=/mnt/volume-nyc1-01/mysql

. . .

Once you’ve made this update, save and exit the file. If you’re using nano, you can do this by pressing CTRL + X, then Y, and ENTER. Now it’s almost time to bring up MySQL again, but before that, there’s one more thing to configure in order to be successful.

## [Step 3 — Configuring AppArmor Access Control Rules](https://www.digitalocean.com/community/tutorials/how-to-move-a-mysql-data-directory-to-a-new-location-on-ubuntu-20-04#step-3-configuring-apparmor-access-control-rules)

In this step, you need to tell AppArmor to let MySQL write to the new directory by creating an alias between the default directory and the new location. AppArmor is a security module in the Linux kernel that allows system administrators to restrict program capabilities through program profiles, rather than users themselves. Start by opening up and editing the AppArmor alias file:

1. sudo nano /etc/apparmor.d/tunables/alias

Copy

At the bottom of the file, uncomment the following line and add the alias rule:

/etc/apparmor.d/tunables/alias

. . .

alias /var/lib/mysql/ -> /mnt/volume-nyc1-01/mysql/,

. . .

When you’re finished, save and exit the file.

For the changes to take effect, restart AppArmor:

1. sudo systemctl restart apparmor

Copy

**Note:** If you skipped the AppArmor configuration step, you will receive the following error message:

Output

Job for mysql.service failed because the control process

exited with error code. See "systemctl status mysql.service"

and "journalctl -xe" for details.

Since this message doesn’t make an explicit connection between AppArmor and the data directory, this error can take some time to figure out.

Once you’ve properly configured AppArmor, you can move on to the next step.

## [Step 4 — Restarting MySQL](https://www.digitalocean.com/community/tutorials/how-to-move-a-mysql-data-directory-to-a-new-location-on-ubuntu-20-04#step-4-restarting-mysql)

Now it’s time to start MySQL. If you do, however, you’ll run into another error. Instead of an AppArmor issue, this error is caused by mysql-systemd-start, a script that supports managing MySQL through systemd. You can inspect this script with the following command:

1. nano /usr/share/mysql/mysql-systemd-start

Copy

This script checks for the existence of either a directory, -d, or a symbolic link, -L, that matches the default data directory path. If it doesn’t find either of these, the script will trigger an error and prevent MySQL from starting:

/usr/share/mysql/mysql-systemd-start

. . .

if [ ! -d /var/lib/mysql ] && [ ! -L /var/lib/mysql ]; then

echo "MySQL data dir not found at /var/lib/mysql. Please create one."

exit 1

fi

if [ ! -d /var/lib/mysql/mysql ] && [ ! -L /var/lib/mysql/mysql ]; then

echo "MySQL system database not found. Please run mysql\_install\_db tool."

exit 1

fi

. . .

After you’ve inspected this file, close it without making any changes.

Since you need either an appropriate directory or symbolic link to start the server, you must create the minimal directory structure to pass the script’s environment check:

1. sudo mkdir /var/lib/mysql/mysql -p

Copy

Now you’re ready to start MySQL:

1. sudo systemctl start mysql

Copy

Confirm MySQL is running by checking the status:

1. sudo systemctl status mysql

Copy

Output

● mysql.service - MySQL Community Server

Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset:>

Active: active (running) since Wed 2022-03-23 20:51:18 UTC; 4s ago

Process: 17145 ExecStartPre=/usr/share/mysql/mysql-systemd-start pre (code=>

Main PID: 17162 (mysqld)

Status: "Server is operational"

Tasks: 38 (limit: 1132)

Memory: 376.7M

CGroup: /system.slice/mysql.service

└─17162 /usr/sbin/mysqld

To ensure that the new data directory is indeed in use, start the MySQL monitor:

1. mysql -u sammy -p

Copy

Now query for the value of the data directory again:

1. SELECT @@datadir;

Copy

Output

+----------------------------+

| @@datadir |

+----------------------------+

| /mnt/volume-nyc1-01/mysql/ |

+----------------------------+

1 row in set (0.01 sec)

After you’ve restarted MySQL and confirmed that it’s using the new location, take the opportunity to ensure that your database is fully functional. Once you’ve finished, exit the database as in the following and return to the command prompt:

1. exit

Copy

Output

Bye

Now that you’ve verified the integrity of any existing data, you can remove the backup data directory:

1. sudo rm -Rf /var/lib/mysql.bak

Copy

Then restart MySQL one final time:

1. sudo systemctl restart mysql

Copy

And finally, confirm it’s working as expected by checking the status:

1. sudo systemctl status mysql

Copy

Output

● mysql.service - MySQL Community Server

Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset:>

Active: active (running) since Wed 2022-03-23 20:53:03 UTC; 4s ago

Process: 17215 ExecStartPre=/usr/share/mysql/mysql-systemd-start pre (code=>

Main PID: 17234 (mysqld)

Status: "Server is operational"

Tasks: 38 (limit: 1132)

Memory: 368.9M

CGroup: /system.slice/mysql.service

└─17234 /usr/sbin/mysqld

/var/lib/mysql/ =====Old path

/data/erpdb/mysql ======== new path

sudo systemctl stop mysql

sudo systemctl status mysql

sudo rsync -av /var/lib/mysql /data/erpdb/

sudo mv /var/lib/mysql /var/lib/mysql.bak

sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf

datadir=/data/erpdb/mysql

sudo nano /etc/apparmor.d/tunables/alias

/data/erpdb/mysql/,

sudo systemctl restart apparmor

nano /usr/share/mysql/mysql-systemd-start

sudo mkdir /var/lib/mysql/mysql -p

sudo systemctl start mysql

sudo systemctl status mysql

**SSL Certificate Installation for Tomcat**

Create a separate directory like SSL and go to SSL directory and generate CSR

#cd SSL

First generate CSR to get cert

you will get CSR and key file in SSL directory now. Where is private key.

**Convert the certificate and private key to a PKCS12 file**

You cannot directly import private key information to a keystore using keytool. You must convert the certificate and private key into a PKCS12 (.p12) file, and then you can import the PKCS12 file to your keystore.

Run the following command:

openssl pkcs12 -export -in [filename-certificate] -inkey [private key] -name [host] -out [filename-new-PKCS-12.p12]

Ex: openssl pkcs12 -export -in 9ee972c0la34057.crt -inkey unitetools.key -name unitetools -out unitetools-12.p12

Import the certificate to your keystore

1. Import the PKCS12 certificate by executing the following command:
2. keytool -importkeystore -deststorepass coopsindia@123 -destkeystore coopsindia.jks -srckeystore coopsindia.p12 -srcstoretype PKCS12
3. where the [password] is the password you specified when you created the private key.
4. Run one of the following commands:
   * If you have a CA bundle file, import it by running the following command:

Ex: keytool -importkeystore -deststorepass Synaptic@123 -destkeystore unitetools.jks -srckeystore unitetools-12.p12 -srcstoretype PKCS12

**Edit server.xml file**

Comment existing string and add below line

<Connector port="443" protocol="HTTP/1.1"

connectionTimeout="20000"

redirectPort="443"

SSLEnabled="true"

scheme="https"

secure="true"

sslProtocol="TLS"

keystoreFile="/home/ubuntu/SSL/coopsindia.com.jks"

keystorePass="coopsindia@123" />

**Automatic DB Backup Scripts**

00 23 \* \* \* mysqldump -u [username] -p [database] | gzip > /home/backup\_$(date +%F.%H%M%S).sql.gz

Sudo crontab -e

0 13 \* \* \* /dbdump.sh

find /home/damien/Pictures/\* -mtime +1 -exec rm -f {} \;

find /tmp/farman -maxdepth 1 -mtime 13 -type f -exec mv "{}" /tmp/new/ \;

to copy other server

scp -r /tmp/new/ root@10.67.55.213:/tmp/farman/

delete more the one day

find /tmp/new/ -type f mtime +1 -ls -exec rm -f -- {} \;

to auto authentication need to ssh key gen

1 genrate rsa key on server where from you connect destonation server

ssh-keygen -t rsa -b 2048

2 copy public key to destination server

ssh-copy-id -i /root/.ssh/id\_rsa.pub root@10.67.55.250 (destination)

3 enter first time password

4 next login can without password

**UFW redirection rule for 443 to 8443**

1 Open the UFW configuration file for editing. Typically, the file is located at **/etc/default/ufw**. Use your preferred text editor to open the file with administrative privileges. For example:

# Nano /etc/default/ufw

2 Locate the line that says **DEFAULT\_FORWARD\_POLICY** and ensure that it is set to **ACCEPT**. It should look like this

DEFAULT\_FORWARD\_POLICY="ACCEPT"

3 Save this file

4 Now, open the UFW rules file for editing. The file is usually located at **/etc/ufw/before.rules**. Use your text editor with administrative privileges to open the file. For instance:

sudo nano /etc/ufw/before.rules

5 Add the following lines at the beginning of the file, before the **\*filter** line:

\*nat

:PREROUTING ACCEPT [0:0]

-A PREROUTING -p tcp --dport 443 -j REDIRECT --to-port 88

COMMIT

\*nat

:PREROUTING ACCEPT [0:0]

-A PREROUTING -p tcp –dport 80 -j REDIRECT --to-port 88

COMMIT

6 save this file

7 enable and disable ufw

#ufw disable

#ufw enable