Lab 4: Installing MySQL Server on Ubuntu Server

First update your repository index to make sure your system is up to date.

```
sudo apt update
```

And upgrade software on your system:

```
Sudo apt upgrade
```

Install MySQL server:

```
sudo apt install mysql-server
```

Check to see if MySQL server service is running:

```
sudo service mysql status
```

MySQL server is now running on your system, unless there are some errors and issues. However, at this point MySQL will not start automatically on reboot. You can set MySQL service to start on start up by using this command:

```
sudo systemctl enable mysql
```

Additionally, you can control the MySQL service using these commands:

```
sudo service mysql stop
sudo service mysql start
sudo service mysql restart
```

MySQL server default settings have a root account that does not have password (insecure, and dangerous). However, by default only the local host can access MySQL server. The MySQL server configurations are in this directory:

```
/etc/mysql/mysql.conf.d
```

We will revisit configurations in the directory later. Before editing any configurations at this point (and allowing remote login to the server) it is recommended that you configure some security first. We can go into MySQL database itself to set passwords and policies, and edit a bunch of configurations in files manually. Instead of changing all the MySQL settings manually to implement many of the recommended settings you can install this package to help with implementing many of those best practices automatically:

```
sudo mysql_secure_installation
```

NOTE: Read all the questions in each step carefully. Do not forget your MySQL root password.

NOTE: MySQL usernames and passwords ARE NOT the same as your Linux Username and password. The users are stored in the MySQL Database itself and are independent from Linux UID/Password.

I recommend setting the root password to a secure password and not allowing root to access the server remotely.

At this point you can visit the MySQL server configuration files to allow remote connections.

Before we move on run this command:

```
sudo ss -tap | grep mysql
```

This should tell you what port (socket) MySQL server is listening on and from what IP addresses (i.e. what IP address is allowed to connect to the server). You can also look at the MySQL error logs for troubleshooting if you are having any troubles:

```
sudo tail /var/log/mysql/error.log
```

Now let's finally set up MySQL Server basic configuration so we can connect to it remotely:

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

NOTE: Since we are editing the configuration files, we will need to restart the MySQL service in order for the changes to take effect. Do not forget to restart the MySQL service to apply new settings (sudo service mysql restart).

In this configuration files find the line that says "port" (is most likely commented in favour of defaults). Uncomment the line and set your port number to anything that you prefer.

Then find the line with "bind-address". This is the IP address that the MySQL service will accept connections from. By default, it only accepts incoming connection from 127.0.0.1 (local host only). You can add other bind-address lines to allow additional IP addresses. In this case since this is a VM and hopefully not exposed to internet I just do this to simplify our configuration:

```
bind-address            = 127.0.0.1
bind-address            = 0.0.0.0
```

Save the file and exit. In order to apply the settings that we just changed, restart the MySQL service:

```
sudo service mysql restart
sudo service mysql status
sudo ss -tap | grep mysql
```

You can get to the MySQL console from your CLI terminal using "mysql" command:

```
sudo mysql -u root
```

NOTE: You may have noticed that when configuring mysql_secure_installation that we disabled root access from anywhere but the local machine. Therefore, at this point only Linux root may be able to log into the MySQL root account locally (two root accounts are NOT the same). In case it gives you "access denied" you need to supply the MySQL root password by using option -p:

```
sudo mysql -uroot -p
```

If everything goes well, you should be getting a "mysql>" prompt.

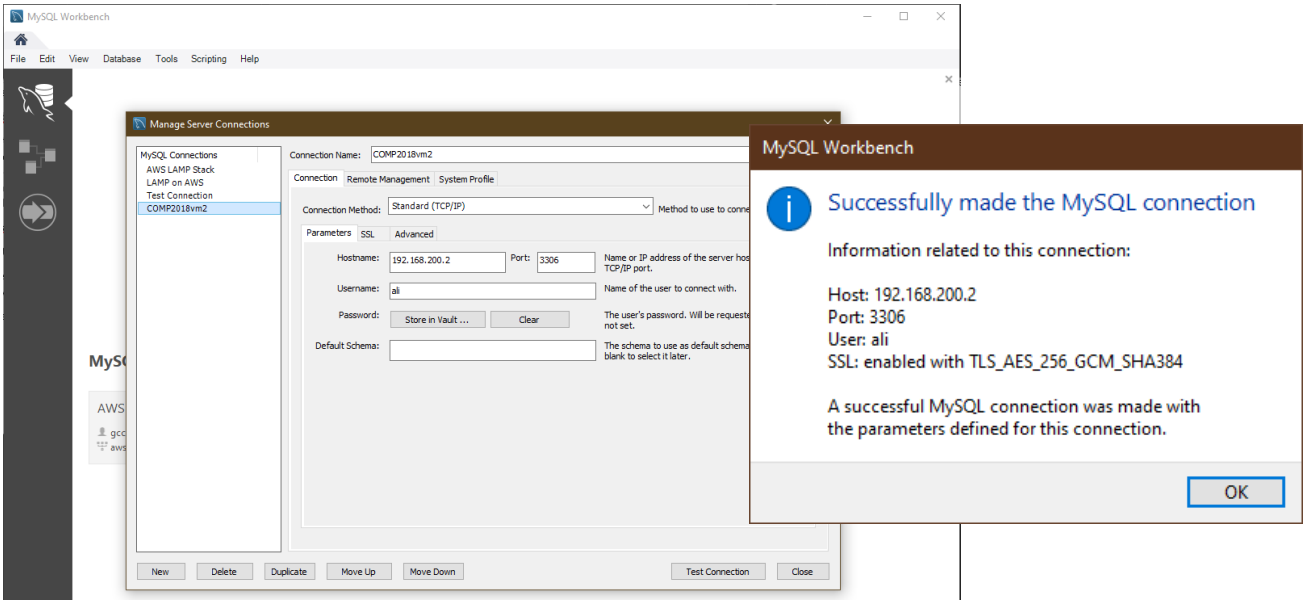
Let's add a MySQL user and database so we can use a client application such as MySQL Workbench to access and use the server. Here are the MySQL command (fill <value> with your own values):

```
CREATE USER '<Insert Username>'@'%' IDENTIFIED WITH mysql_native_password BY
'<insert password>';
CREATE DATABASE <name a database> /*\!40100 DEFAULT CHARACTER SET utf8 */;
GRANT ALL PRIVILEGES ON <name a database>.* TO '<insert username>'@'%';
FLUSH PRIVILEGES;
```

You can view the current users and what authentication method they are using by issuing the following query. Sample output is also provided:

```
mysql> SELECT user,plugin,host FROM mysql.user;
+------------------+-----------------------+-----------+
| user             | plugin                | host      |
+------------------+-----------------------+-----------+
| ali              | mysql_native_password | %         |
| debian-sys-maint | caching_sha2_password | localhost |
| mysql.infoschema | caching_sha2_password | localhost |
| mysql.session    | caching_sha2_password | localhost |
| mysql.sys        | caching_sha2_password | localhost |
| root             | auth_socket           | localhost |
+------------------+-----------------------+-----------+
6 rows in set (0.00 sec)
```

Go to your Host PC and install a MySQL Client (MySQL Workbench) and try connecting to your server:



Assignment 4: MySQL database backup and restore

1. Create a screenshot of you MySQL Workbench or another client that can successfully connect to your MySQL Server running on your VM via your host operating system (see example above).
2. Create a command that backs up your entire MySQL Databases to a file that can later be imported into other systems' MySQL server:
    a. Use mysqldump command
    b. Use the help to figure out what option you can use to do this as root on your Linux machine to dump everything, including all the MySQL user accounts and their passwords
    c. You can also study this: https://dev.mysql.com/doc/refman/8.0/en/backup-methods.html
3. Figure out the command that can restore this backup to your current MySQL server (hint: use mysql command)!
4. Use mysqldump to only backup the specific database that was created for your remote user in the lab.