Instructions: This week's lab is focused on remote management tools and becoming familiar with MySQL, a database management system (DBMS). We've learned that Linux servers operate a good portion of the Internet. A majority of these servers operate a database to support the backend of a web interface, or application. Therefore, as Linux administrators, it is important that we familiarize ourselves with popular DBMSs.

Remote management tools are invaluable. With each technology advancement, we move further away from physical servers with specific purposes to virtualized stacks running most, if not all, core networking services. Therefore, we may only have the option of remote management if no other tools are available or there is no physical server to connect user interface hardware.

As you work through the lab I will ask for screenshots of your output or answers that you'll receive based on your output. Paste your screenshots into a Word document with a brief explanation of each screenshot. Make sure you have a cover page with your full name.

1) Cleanup
   a. Before we dig into the main content of this lab we will issue a couple of commands to help keep your system clean. As time goes on, packages are no longer needed and the repository needs to be cleaned on your Linux system. Some distributions are good at automatically performing these steps, others are not. Combined with your package manager, the following options will help with cleanup.
   b. Before starting your Ubuntu VM, go into its Settings via the VirtualBox Manager and ensure Network Adapter 1 is **Enabled** and set to **NAT**. Next, configure Adapter 2 as **Enabled** and set to **Host-only Adapter**. Click ok and start your Ubuntu VM.
   c. After you're logged in, use Ubuntu's package manager to make sure you have the latest package updates. Then, use it again with the `autoremove` option, then again with the `autoclean` options.
      i. Note: We're nearing the end of this course so instead of me giving you step-by-step commands, I'll expect you to find the correct commands through the tools at your fingertips.

*****Screenshot #1, show your output after using autoremove and autoclean**

2) Remote Management Setup
   a. Next, let's setup our Ubuntu workstation to accept SSH connections. This is a common method used in the field to remotely manage Linux servers and workstations. It requires very little network bandwidth and is the most acceptable form of Linux remote management.
   b. On your Ubuntu VM, use your package manager to install the `openssh-server` package.
      i. Reboot your VM.
   c. After Ubuntu reboots, login, open a terminal, and find the IP address of your second networking adapter.
      i. Hint: it should be a Class C private address of something like: 192.168.xxx.xxx.
   d. On your host computer (your physical Windows workstation), open a web browser and navigate to https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html
      i. Scroll down and find the putty.exe download link for 64-bit. Click the link to download PuTTY.
      ii. After the download is complete, create a PuTTY shortcut on your desktop
   e. Double-click the PuTTY shortcut to launch the program
      i. Under the Host Name (or IP address) box, enter the IP address of your Ubuntu VM
      ii. Leave the Port number set to 22 (SSH) and click Open
   f. Login with your standard user account on your Ubuntu VM.
      i. It's a bad security habit to logon with the root account, so don't start that habit

*****Screenshot #2, show that you're logged in remotely to your Ubuntu VM using PuTTY from your desktop**

3) <u>SSH with public/private keys</u>

   a. SSH is more secure than other protocols, like Telnet, but it is becoming standard practice to use Public Key Infrastructure (PKI) to increase security. PKI takes advantage of current encryption algorithms to create a public and private key pair to unlock, or authenticate to our devices. The public key resides on the remote workstation. The private key is held by the individual (administrator) and is presented to the remote device to authenticate using the public/private combination. The private key must be kept secret, meaning do not let someone get a hold of the private key file. Only the private/public key combination can unlock a device. You cannot gain access with a public/public combination or private/private combination.
   b. Again, through your own research, setup your Ubuntu VM to force authentication using PKI.
      i. Hint: you'll still use PuTTY like you did on Step 2. However, this time you will present your Ubuntu VM with a private key. Your Ubuntu VM will hold the public key. There are plenty of resources on the Internet. Also, there is more than one way to setup PKI. I will not be grading on how you got there, only your end result of logging in remotely via PuTTY using a public/private key combination.

*****Screenshot #3, PuTTY screen showing a logged in user after authenticating with the public key**

4) MySQL Database setup

   a. We're going to install MySQL on our Ubuntu VM. MySQL is managed by Oracle, an industry leader in big data and database services. MySQL is the free, non-commercial version.
   b. Use your package manager to install `mysql-server`
      i. During installation, MySQL will ask for a root password
         1. This root account is *not* the same account as your Ubuntu root, so, the passwords do not need to be the same. However, I highly encourage the use of the same password (so you don't forget it) for our labs. In the real-world, you would never, <u>*never*</u> reuse root passwords.
   c. Confirm `mysql` is running after installation by checking its status

*****Screenshot #4, show the running status of MySQL**

   d. Let's login to our MySQL database
      i. enter `mysql –u root –p`
      ii. enter your root password, the root password for MySQL captured during installation
      iii. Notice you're now at a `mysql` prompt. From here, you can manipulate any database on the local instance. You cannot enter standard Linux commands at this point.
   e. Create a new user with a name and password of your choice
      i. `CREATE USER '<username>'@'localhost' IDENTIFIED BY '<password>';`
      ii. Note: at this step and through the rest of this lab, you'll replace <username> and <password> with the actual username and password you create.
      iii. Example: `CREATE USER 'tracey'@'localhost' IDENTIFIED BY 'P@ssw0rd123';`
         1. Note #2: Notice the semi-colon at the end of the CREATE USER command. Every MySQL command must end with a semi-colon; otherwise you'll get nowhere!
   f. Grant all privileges to your new user, this basically creates a new admin account that is not root
      i. `GRANT ALL PRIVILEGES ON *.* TO '<username>'@'localhost' WITH GRANT OPTION;`

g. Create a new user with the __same__ username and password you used above, but notice we replace 'localhost' with '%'. This allows that user to login to the database from any remote system. This is good for remote database management, but bad security practice. Don't forget to add privileges.
   i. `CREATE USER '<username>'@'%' IDENTIFIED BY '<password>';`
   ii. `GRANT ALL PRIVILEGES ON *.* TO '<username>'@'%' WITH GRANT OPTION;`

h. Verify your new users have the privileges you set with the previous commands
   i. `SHOW GRANTS FOR '<username>'@'localhost';`
   ii. `SHOW GRANTS FOR '<username>'@'%';`

i. Type `exit` to close your MySQL prompt and return to your shell

*****Screenshot #5, PuTTY output showing grant options for both user accounts**

5) <u>Creating a database and table</u>

a. Login to MySQL with the user you created above
   i. `mysql –u <username> –p`
   ii. Enter your password
      1. If you ever forget who you're logged in as, run the following command:
         a. `SELECT CURRENT_USER();`
         b. you should see `<username>@localhost`

b. With a clean install, MySQL builds generic databases
   i. enter `SHOW DATABASES;` to view all databases on the instance

c. Create a new database called MIS_Students
   i. `CREATE DATABASE MIS_Students;`
   ii. Use the `SHOW DATABASES;` command to confirm your new database was successfully created

d. To interact with your new database, you have to tell MySQL which database you want to use
   i. `USE MIS_Students;`

e. Let's look to see if there are any tables built in your database
   i. `SHOW TABLES;`
      1. You should not see any tables here

f. Create a new table called students with three fields, stud_id, firstname, and lastname. We'll set the stud_id as a primary key and to auto increment each time we enter a record.
   i. `CREATE TABLE students (stud_id INT AUTO_INCREMENT PRIMARY KEY, firstname VARCHAR(30), lastname VARCHAR(30));`

g. `SHOW TABLES;`
   i. you should see your new table, students

h. Use a standard SQL query to view the data in the students table
   i. `SELECT * FROM students;`
   ii. You should see an empty set, meaning you have no data in the table

*****Screenshot #6, output showing your students table with an empty set**

6) <u>Inserting data into MySQL</u>

a. Make sure you're still set to use the MIS_Students database. To quickly check, use the following command
   i. `SELECT DATABASE();`
      1. You should see one database, MIS_Students
      2. If not, perform a USE command to select the proper database

b. Let's enter some students into our new table, students

        i. `INSERT INTO students (firstname,lastname) VALUES ('bob','barker'), ('david','letterman'), ('alice','cooper');`

   c. Verify your entries with a SELECT statement
        i. `SELECT * FROM students;`
        ii. You should see 3 students with stud_id of 1,2,3 (notice you didn't need to enter the stud_id, we set this to auto increment)
   d. Enter `exit` to logout of MySQL

*****Screenshot #7, output showing the three rows entered via the INSERT INTO command**

7) <u>Scripting database entries</u>
   a. Make sure you're in your `/home/<username>/Documents` directory, if not, get there
   b. Use your favorite text editor to create a new file called `mysqlscript`
   c. Enter the contents of the next bullet into your script. Here, you're passing a command to log into MySQL, then, you're going to insert 10 new students. You should have noticed entering records via MySQL commands is very tedious with zero error tolerance. Scripting is faster, plus it allows you to save and review at your own pace.
   d.
```
#!/bin/bash

mysql --user=<username> --password=<password> MIS_Students << EOF
INSERT INTO students (firstname,lastname) VALUES('fname4','lname4'),
('fname5','lname5'), ('fname6','lname6'), ('fname7','lname7'),
('fname8','lname8'), ('fname9','lname9'), ('fname10','lname10'),
('fname11','lname11'), ('fname12','lname12'), ('fname13','lname13');
EOF
```
   e. Save and exit your file
   f. Copy your new file with the .sh extension. This helps highlight your working file versus your script
        i. `cp mysqlscript mysqlcript.sh`
   g. Now use the `chmod` command to add execution permission to your `mysqlscript.sh` file
   h. Verify your execution permission (x) is set with a long list command
   i. Run the new script
        i. `./mysqlscript.sh`
        ii. You'll get a warning about using a password in the command line. This is a good error, you shouldn't set passwords in scripts, but we did just to move along in the lab.

8) <u>Verifying your work</u>
   a. Log back into MySQL with the user you created
   b. Change your database to MIS_Students
   c. Use a SELECT query to view the 10 new students, plus the original 3 we created
        i. `SELECT * FROM students;`
        ii. You should see an output of all students with their automatically created stud_id
   d. Exit MySQL

*****Screenshot #8, output showing 13 rows in the students table**

9) Using MySQL Workbench

a. Gaining fluency in the Linux/MySQL command line interface (CLI) in invaluable. However, there are times when a GUI is quicker, more efficient, and allows you to find errors easily. Let's use MySQL's native GUI, Workbench. You can install this from PuTTY, but after the install, you'll need to login to your VM directly to use the GUI.

b. Use your package manager to install `mysql-workbench`
   i. This package has a lot of dependencies, so be patient while it installs

c. Close PuTTY and head over to your Ubuntu VM desktop

d. Once you're logged in, find Workbench under your applications menu (top left corner of the desktop) and launch the workbench application
   i. Click on Local instance 3306
   ii. Notice it asks for your root password, again this is the MySQL root password

e. Click on Server Status
   i. Here you see metrics like connections, traffic, diskspace, encryption levels

f. Click on Client Connections
   i. This display shows all connected users. You can kill query(s) and connection(s)
   ii. If you start a new PuTTY session and login to your MySQL server, you'll see a new client connection

g. Click on Users and Privileges
   i. From here you can add/delete accounts, change passwords, change permissions, etc…
   ii. Create a new user in the GUI. Notice, you're performing the same steps as with the CLI
      1. Click on the Administrative Roles tab, click DBA
      2. Notice DBA automatically enabled all other roles, DBA is powerful, this is what your root account has.
   iii. Now, click Apply, this saves your changes

h. Under Instance, click Startup / Shutdown
   i. This allows you to start or stop the server instance. If you try to stop it, you should receive an error. You cannot stop an instance while logged in as a user. The best way is to stop the MySQL service via CLI.

i. Under Performance, click Dashboard
   i. This screen displays live performance statistics, this is good for troubleshooting and determining required hardware for future builds

j. Under Schemas, you should see your MIS_Students database.
   i. Here, you can view your database setup, properties, tables, etc.
   ii. Notice, you can't really do a whole lot, but that's okay

k. In the main body of Workbench, you should still see the Performance Dashboard. At the top click the X to close all windows except for Query 1

l. In Query 1, enter a MySQL command to create a new database MIS_Courses
   i. Hint: it's the same command you used in the CLI
   ii. Once your command is set, click the first lightning bolt, next to the save icon
   iii. Click the refresh icon next to SCHEMAS to see your new database

m. Expand the MIS_Courses database by clicking the right arrow next to the name
   i. Right-click on Tables and select Create Table…
   ii. Give the table a name: courses
   iii. Name the first column, course_id, check PK, NN, and AI
      1. PK = Primary Key
      2. NN = Not Null, does not allow null (or empty) entries

3. AI = Auto Increment

iv. Create a second column, course_name

v. Create a third column, course_code

vi. Click Apply to save all changes

n. Refresh your SCHEMA again, you should see your new database with your new table. This should have seemed more intuitive and quicker than the CLI. This doesn't mean GUI is better!

*****Screenshot #9, your Workbench window showing your new database and table with columns**

10) <u>Quick Copy</u>

a. A quick copy allows you to grab a file from one machine and save it to another via a network connection. Be careful, this can open up your machine to unwanted guests if a port scan is being performed.

b. Open a shell on your Ubuntu VM, or, via PuTTY

   i. PuTTY is easier and quicker, less overhead on your Host and VM

c. Make sure you're in your home directory and type the following python command

   i. `python -m SimpleHTTPServer`

d. Now, on your Host, open a web browser and navigate to 192.168.xxx.xxx:8000

   i. Replace the x's with your actual IP address of the 2nd Adapter on your VM

   ii. It's the same IP address you used to PuTTY in

e. When your VM responds, notice you have clickable links in your home directory. From here, you can download any files from your home directory, but only your home directory

*****Screenshot #10, show your host web browser with the contents of your home directory**

11) <u>Upload your work.</u>