



Integrated Cloud Applications & Platform Services

MySQL for Database Administrators

Activity Guide

D61762GC50

Edition 5.0 | June 2019 | D106726

Learn more from Oracle University at education.oracle.com



Authors

KimSeong Loh
Mark Lewin

Technical Contributors and Reviewers

Frederic Descamps
Hananto Wicaksono
Igor Ilyin
Mirko Ortensi

Editors

Adrita Biswas
Aju Kumar
Raj Kumar
Moushmi Mukherjee

Graphic Editors

Kavya Bellur
Pushparaj Kundar

Publishers

Pavithran Adka
Veena Narasimhan

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Disclaimer

This document contains proprietary information and is protected by copyright and other intellectual property laws. You may copy and print this document solely for your own use in an Oracle training course. The document may not be modified or altered in any way. Except where your use constitutes "fair use" under copyright law, you may not use, share, download, upload, copy, print, display, perform, reproduce, publish, license, post, transmit, or distribute this document in whole or in part without the express authorization of Oracle.

The information contained in this document is subject to change without notice. If you find any problems in the document, please report them in writing to: Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. This document is not warranted to be error-free.

Restricted Rights Notice

If this documentation is delivered to the United States Government or anyone using the documentation on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

Trademark Notice

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Table of Contents

Practices for Lesson 1: Introduction to MySQL.....	7
Practices for Lesson 1	8
Practices for Lesson 2: Installing and Upgrading MySQL	9
Course Practice Environment: Security Credentials.....	10
Practices for Lesson 2: Overview	11
Practice 2-1: Installing MySQL	12
Solution 2-1: Installing MySQL	13
Practice 2-2: Connecting to MySQL	17
Solution 2-2: Connecting to MySQL	18
Practice 2-3: Configuring the MySQL Service.....	20
Solution 2-3: Configuring the MySQL Service.....	21
Practice 2-4: Upgrading MySQL.....	24
Solution 2-4: Upgrading MySQL.....	25
Practices for Lesson 3: Understanding MySQL Architecture.....	33
Practices for Lesson 3: Overview	34
Practice 3-1: Configuring Tablespaces	35
Solution 3-1: Configuring Tablespaces	37
Practice 3-2: Configuring the Buffer Pool.....	45
Solution 3-2: Configuring the Buffer Pool.....	46
Practices for Lesson 4: Configuring MySQL.....	49
Practices for Lesson 4: Overview	50
Practice 4-1: Modifying a Setting by Using Command-Line Arguments.....	51
Solution 4-1: Modifying a Setting by Using Command-Line Arguments.....	52
Practice 4-2: Modifying the Configuration File.....	57
Solution 4-2: Modifying the Configuration File.....	58
Practice 4-3: Changing Dynamic Settings	61
Solution 4-3: Changing Dynamic Settings	62
Practice 4-4: Persisting Global Variables.....	64
Solution 4-4: Persisting Global Variables.....	65
Practice 4-5: Configuring the Client.....	69
Solution 4-5: Configuring the Client.....	70
Practice 4-6: Running Multiple MySQL Servers on the Same Host with <code>systemd</code>	73
Solution 4-6: Running Multiple MySQL Servers on the Same Host with <code>systemd</code>	75
Practices for Lesson 5: Monitoring MySQL	81
Practices for Lesson 5: Overview	82

Practice 5-1: Configuring the Slow Query Log	83
Solution 5-1: Configuring the Slow Query Log	84
Practice 5-2: Using Performance Schema	88
Solution 5-2: Using Performance Schema	90
Practice 5-3: Installing MySQL Enterprise Monitor.....	98
Solution 5-3: Installing MySQL Enterprise Monitor.....	100
Practice 5-4: Monitoring Server Activity	101
Solution 5-4: Monitoring Server Activity	104
Practices for Lesson 6: Managing MySQL Users	109
Practices for Lesson 6: Overview	110
Practice 6-1: Quiz—User Management	111
Solution 6-1: Quiz—User Management	113
Practice 6-2: Creating Users and Roles.....	115
Solution 6-2: Creating Users and Roles.....	116
Practice 6-3: Granting Permissions	120
Solution 6-3: Granting Permissions	121
Practices for Lesson 7: Securing MySQL	127
Practices for Lesson 7: Overview	128
Practice 7-1: Quiz – Securing MySQL	129
Solution 7-1: Quiz—Securing MySQL	131
Practice 7-2: Enabling SSL for Secure Connections	132
Solution 7-2: Enabling SSL for Secure Connections	134
Practice 7-3: Configuring MySQL Enterprise Firewall	140
Solution 7-3: Configuring MySQL Enterprise Firewall	142
Practices for Lesson 8: Maintaining a Stable System	149
Practices for Lesson 8: Overview	150
Practice 8-1: Quiz—Maintaining a Stable System.....	151
Solution 8-1: Quiz—Maintaining a Stable System.....	153
Practice 8-2: Identifying the Cause of Slow Performance.....	154
Solution 8-2: Identifying the Cause of Slow Performance.....	156
Practices for Lesson 9: Optimizing Query Performance	165
Practices for Lesson 9: Overview	166
Practice 9-1: Improving Query Performance with Indexes.....	167
Solution 9-1: Improving Query Performance with Indexes.....	169
Practice 9-2: Using MySQL Enterprise Monitor Query Analyzer.....	180
Solution 9-2: Using MySQL Enterprise Monitor Query Analyzer.....	189
Practices for Lesson 10: Choosing a Backup Strategy	191
Practices for Lesson 10: Overview	192
Practice 10-1: Quiz – Backup Strategies	193

Solution 10-1: Quiz – Backup Strategies	195
Practices for Lesson 11: Performing Backups	197
Practices for Lesson 11: Overview	198
Practice 11-1: Backing Up with MySQL Enterprise Backup.....	199
Solution 11-1: Backing Up with MySQL Enterprise Backup.....	201
Practice 11-2: Backing Up with mysqldump and mysqlpump	210
Solution 11-2: Backing Up with mysqldump and mysqlpump	212
Practice 11-3: Backing Up by Using the Binary Log.....	220
Solution 11-3: Backing Up by Using the Binary Log.....	222
Practices for Lesson 12: Configuring a Replication Topology	231
Practices for Lesson 12: Overview	232
Practice 12-1: Quiz–Configuring Replication	233
Solution 12-1: Quiz–Configuring Replication	235
Practice 12-2: Configuring Replication.....	236
Solution 12-2: Configuring Replication.....	238
Practice 12-3: Adding a New Slave	247
Solution 12-3: Adding a New Slave	248
Practice 12-4: Enabling GTID and Configuring Circular Replication	252
Solution 12-4: Enabling GTID and Configuring Circular Replication	253
Practices for Lesson 13: Administering a Replication Topology.....	261
Practices for Lesson 13: Overview	262
Practice 13-1: Monitoring Replication with MySQL Enterprise Monitor	263
Solution 13-1: Monitoring Replication with MySQL Enterprise Monitor	266
Practice 13-2: Troubleshooting Replication Errors	271
Solution 13-2: Troubleshooting Replication Errors	272
Practice 13-3: Performing a Replication Failover	276
Solution 13-3: Performing a Replication Failover	277
Practice Scripts for Lesson 13.....	283
Practices for Lesson 14: Achieving High Availability with MySQL InnoDB Cluster.....	285
Practices for Lesson 14: Overview	286
Practice 14-1: Creating Member Instances.....	287
Solution 14-1: Creating Member Instances.....	289
Practice 14-2: Creating MySQL InnoDB Cluster	296
Solution 14-2: Creating MySQL InnoDB Cluster	298
Practice 14-3: Deploying MySQL Router and Testing the Cluster	308
Solution 14-3: Deploying MySQL Router and Testing the Cluster	309
Practice 14-4: Testing High Availability.....	313
Solution 14-4: Testing High Availability.....	314
Practices for Lesson 15: Conclusion	319

Practices for Lesson 15.....	320
------------------------------	-----

Practices for Lesson 1: Introduction to MySQL

Practices for Lesson 1

There are no practices for this lesson.

Practices for Lesson 2: Installing and Upgrading MySQL

Course Practice Environment: Security Credentials

For operating system (OS) usernames and passwords, see the following:

- If you are attending a classroom-based or live virtual class, ask your instructor or LVC producer for OS credential information.
- If you are using a self-study format, refer to the communication that you received from Oracle University for this course.

When you first log in to the OS using these credentials, you are required to change your password. The new password must meet the following requirements:

- At least eight characters
- At least one uppercase and one lowercase letter
- At least one digit
- At least one “special character” (for example: *, [, !, %, \$)
- Not based on a dictionary word

The following example demonstrates this process:

```
You are required to change your password immediately (root enforced)
Last login: <date> from <IP address>
Changing password for root.
(current) UNIX password: <initial password>
New password: MySQL8.0!
Retype new password: MySQL8.0!
```

Note: You should record your OS password somewhere and keep it safe.

Practices for Lesson 2: Overview

Overview

In these practices, you install the MySQL server and command-line client, perform the basic initial configuration that is required to use them, and then upgrade to a later version of the software.

Notes

- Many of the steps require that you type commands in a terminal window. You might find it convenient to launch a terminal window and keep it open for the duration of the practices.
- Some steps require that you view or edit a text file. Course computers come preinstalled with commonly used editors, such as `nano`, `gedit` (Gnome Text Editor), and `vim`. Use the editor with which you are most comfortable.
 - Note that if you launch a graphical editor, such as `gedit` from the command line, it produces debug information that might appear confusing but which you can safely ignore.
- The MySQL version numbers referred to in the practices titled “Installing MySQL” and “Upgrading MySQL” are those installed in Oracle classroom environments. The versions of the software might be different on your machine. In that case, use the earlier version for the “Installing MySQL” practice and the later version for the “Upgrading MySQL” practice.

Assumptions

- You are logged in as the `root` user in a terminal window on the course image.
 - The `#` prompt in your Linux terminal window indicates that you are logged in as `root`.
 - Sample commands in the solutions indicate that you are logged in as `root` by showing a `#` prompt.
- The `mysql-advanced-version.tar.gz` and `mysql-commercial-version.tar.gz` files are in the `/stage/MySQL-Server` directory.
 - The `mysql-advanced-version.tar.gz` file should be an older commercial edition of MySQL. The `mysql-commercial-version.tar.gz` file should be a newer commercial edition of MySQL. In Oracle classroom environments, the version numbers are 5.7.25 and 8.0.16, respectively. They may be different in non-Oracle classroom environments.

Practice 2-1: Installing MySQL

Overview

In this practice, you install MySQL 5.7 from a binary archive distribution and perform the initial configuration before you start the server daemon. You upgrade to MySQL 8.0 in the practice titled “Upgrading MySQL.”

Duration

This practice should take you approximately 15 minutes to complete.

Tasks

1. Extract the MySQL 5.7 binary archive from `/stage/MySQL-Server/mysql-advanced-5.7.*.tar.gz` to `/opt`.
2. Create a symbolic link from the newly extracted MySQL directory to `/usr/local/mysql`.
When you install a new version of MySQL in the practice titled “Upgrading MySQL,” you can replace this symbolic link with a link to the new version’s directory without overwriting the existing version’s directory.
3. View the contents of the `/usr/local/mysql/bin` directory.
4. Add `/usr/local/mysql/bin` to the `root` user’s executable search path by adding the line `export PATH=$PATH:/usr/local/mysql/bin` to the `~/.bashrc` file.
5. Apply your changes to the `~/.bashrc` file, by executing the `Linux source` command.
6. Copy the `/labs/my.cnf` file to `/etc/my.cnf`.
7. View the contents of the `/etc/my.cnf` file.
8. The `mysql` user does not as yet exist on the host machine. Create it as a system user by executing the `adduser -r` command.
9. Initialize the data directory by executing `mysqld --initialize` and note the temporary password that appears.
10. The `pid-file` option refers to the `/var/run/mysqld` directory, which does not as yet exist. Create that directory and grant its ownership to the `mysql` user and group.
11. Launch MySQL by executing the `mysqld_safe` script.

Solution 2-1: Installing MySQL

Solution Steps

1. Extract the MySQL 5.7 binary archive from `/stage/MySQL-Server/mysql-advanced-5.7.*.tar.gz` to `/opt`.

Enter the following commands at the Linux terminal prompt and receive the results shown:

```
# cd /opt  
# tar xf /stage/MySQL-Server/mysql-advanced-5.7*.tar.gz
```

- The wildcard symbol expands to the complete file name, which includes the version and architecture of the archive on your host.

2. Create a symbolic link from the newly extracted MySQL directory to `/usr/local/mysql`.

Enter the following command at the Linux terminal prompt:

```
# ln -s /opt/mysql* /usr/local/mysql
```

When you install a new version of MySQL in the practice titled “Upgrading MySQL,” you replace this symbolic link with a link to the new version’s directory without overwriting the existing version’s directory.

3. View the contents of the `/usr/local/mysql/bin` directory.

Enter the following commands at the Linux terminal prompt and receive the results shown:

```
# ls /usr/local/mysql/bin  
innoschecksum mysql_embedded  
lz4_decompress mysqlimport  
myisamchk mysql_install_db  
myisam_ftdump mysql_plugin  
myisamlog mysqlpump  
myisampack mysql_secure_installation  
my_print_defaults mysqlshow  
mysql mysqlslap  
mysqladmin mysql_ssl_rsa_setup  
mysqlbinlog mysqltest_embedded  
mysqlcheck mysql_tzinfo_to_sql  
mysql_client_test_embedded mysql_upgrade  
mysql_config perror  
mysql_config_editor replace  
mysqld resolveip  
mysqld-debug resolve_stack_dump  
mysqld_multi zlib_decompress  
mysqld_safe  
mysqldump  
mysqldumpslow
```

4. Add /usr/local/mysql/bin to the root user's executable search path by adding the line `export PATH=$PATH:/usr/local/mysql/bin` to the `~/.bashrc` file.

Edit the `~/.bashrc` file with a text editor, such as `vim` or `gedit`, and add the following highlighted line at the bottom of the file.

```
...
# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi
export PATH=$PATH:/usr/local/mysql/bin
```

The `~` symbol refers to the current Linux user's home directory.

5. Apply your changes to the `~/.bashrc` file by executing the Linux `source` command.

Enter the following command at the Linux terminal prompt:

```
# source ~/.bashrc
```

6. Copy the `/labs/my.cnf` file to `/etc/my.cnf`.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# cp /labs/my.cnf /etc/my.cnf
```

- Some Linux distributions include a `/etc/my.cnf`. The Oracle Linux version used in classroom environments does not.

7. View the contents of the /etc/my.cnf file.

Open the /etc/my.cnf file with a text editor, such as gedit, or a viewer, such as less or cat.

```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Disabling symbolic-links is recommended to prevent assorted
# security risks
symbolic-links=0

[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
```

The options in the /etc/my.cnf file specify the following:

- The data directory is in /var/lib/mysql.
- The socket file is in the data directory.
- The process runs as the mysql user.
- When you launch MySQL via mysqld_safe, the server logs errors in the /var/log/mysqld.log file, and the PID file is in the /var/run/mysqld directory.
- The default base directory (not amended in /etc/my.cnf) is /usr/local/mysql.

If you opened the /etc/my.cnf file with a text editor, close the text editor before you proceed to the next step.

8. The mysql user does not as yet exist on the host machine. Create it as a system user by executing the adduser -r command.

Enter the following command at the Linux terminal prompt:

```
# adduser -r mysql
```

- The -r switch creates a system user that has no password, that does not expire, and that has no home directory. It also creates a group with the same name.

9. Initialize the data directory by executing mysqld --initialize and note the temporary password that appears.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# mysqld --initialize
<date-and-time> 0 [Warning] TIMESTAMP with implicit DEFAULT value is deprecated. Please use --
explicit_defaults_for_timestamp server option (see documentation for more details).
<date-and-time> 0 [Warning] InnoDB: New log files created,
LSN=45790
```

```

<date-and-time> 0 [Warning] InnoDB: Creating foreign key
constraint system tables.

<date-and-time> 0 [Warning] No existing UUID has been found, so
we assume that this is the first time that this server has been
started. Generating a new UUID: 72204628-bd2b-11e6-b002-
3417eb99981d.

<date-and-time> 0 [Warning] Gtid table is not ready to be used.
Table 'mysql.gtid_executed' cannot be opened.

<date-and-time> 0 [Warning] CA certificate ca.pem is self
signed.

<date-and-time> 1 [Note] A temporary password is generated for
root@localhost: 3C_Va/o*88DP

```

The --initialize command-line option creates a new data directory in /var/lib/mysql owned by the mysql user and group, using the settings in the /etc/my.cnf file. The new data directory contains the initial InnoDB system tablespace, system databases, and SSL encryption keys and certificates. The command also creates an initial temporary password for the local MySQL root account. The password that appears on your screen is different from the password that is shown in the preceding output. Make a note of the password that appears, paying particular attention to any special characters that it might contain.

10. The pid-file option refers to the /var/run/mysqld directory, which does not as yet exist. Create that directory and grant its ownership to the mysql user and group.

Enter the following commands at the Linux terminal prompt:

```

# mkdir /var/run/mysqld
# chown mysql:mysql /var/run/mysqld/

```

11. Launch MySQL by executing the mysqld_safe script.

Enter the following commands at the Linux terminal prompt and receive the results shown:

```

# mysqld_safe &
[1] 16192
<date-and-time> mysqld_safe Logging to '/var/log/mysqld.log'.
<date-and-time> mysqld_safe Starting mysqld daemon with
databases from /var/lib/mysql

```

- MySQL is now running.
- Press the return key to return to the terminal prompt.

Practice 2-2: Connecting to MySQL

Overview

In this practice, you connect to the newly configured MySQL server and change the initial random password.

Duration

This practice should take you approximately five minutes to complete.

Tasks

1. The `/etc/my.cnf` file does not as yet contain an option group that configures the client. Add a `[client]` option group that contains the same `socket` value as the value contained in the `[mysqld]` option group.
2. Launch the `mysql` command-line client, logging in as the `root` user with the temporary password that you noted in the preceding practice.
3. Attempt to change the current database to `mysql` and note the error message that appears.
4. Change the MySQL `root` password to `oracle`.
5. Log out of the `mysql` client.
6. Log in again, this time providing the new password.
7. Attempt to change the current database to `mysql` and note any differences in behavior from step 3.
8. Log out of the `mysql` client.

Solution 2-2: Connecting to MySQL

Solution Steps

1. The `/etc/my.cnf` file does not as yet contain an option group that configures the client. Add a `[client]` option group that contains the same `socket` value as the value contained in the `[mysqld]` option group.

Edit the `/etc/my.cnf` file and add the following lines at the bottom of the file:

```
[client]
socket=/var/lib/mysql/mysql.sock
```

2. Launch the `mysql` command-line client, logging in as the `root` user with the temporary password that you noted in the preceding practice.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# mysql -uroot -p
Enter password: 3C_Va/o*88DP
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.25-enterprise-commercial-advanced

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All
rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or
its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current
input statement.

mysql>
```

- The password that you enter is the same password that you derived in the preceding activity and is different from that shown in the preceding output.
- The `mysql>` prompt appears, showing that you are logged in to MySQL.

3. Attempt to change the current database to `mysql` and note the error message that appears.

Enter the following statement at the `mysql` prompt and receive the results shown:

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

- The password that you used to log in is a temporary password that is expired. You must change it before you can execute any other statements.

4. Change the MySQL root password to oracle.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> ALTER USER USER() IDENTIFIED BY 'oracle';
Query OK, 0 rows affected (#.## sec)
```

- The USER() function returns the current user's full account name (in this case, root@localhost) and provides a useful way to identify the current user without typing out the full account name.

5. Log out of the mysql client.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> EXIT
Bye
```

6. Log in again, this time providing the new password.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# mysql -uroot -p
Enter password: oracle
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 3
...
mysql>
```

- MySQL accepts the new password.

7. Attempt to change the current database to mysql and note any differences in behavior from step 3.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> USE mysql
Reading table information for completion of table and column
names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql>
```

- The command changes the default database successfully. When you change the initial temporary password, it re-enables the account.

8. Log out of the mysql client.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> EXIT
Bye
```

Practice 2-3: Configuring the MySQL Service

Overview

In this practice, you configure the MySQL installation to work with systemd and enable automatic restart when the host reboots.

Note: If you install MySQL using an RPM or Debian package on a Linux distribution that uses systemd, the installer performs the steps in this practice for you. Because you installed the MySQL server from generic binaries, you must configure systemd support yourself.

Duration

This practice should take you approximately ten minutes to complete.

Tasks

1. Stop MySQL by executing an appropriate `mysqladmin` command.
2. Inspect the contents of the `/labs/service/mysql-5.7/mysqld.service` file. This is the service unit configuration file.
3. Copy the `mysqld.service` file to the `/usr/lib/systemd/system` directory.
4. Enter a `systemctl` command that enables the `mysqld` service to start automatically when the host machine reboots.
5. Enter a `systemctl` command that starts the `mysqld` service.
6. Enter a `systemctl` command that reports the status of the `mysqld` service.

Solution 2-3: Configuring the MySQL Service

Solution Steps

1. Stop MySQL by executing an appropriate mysqladmin command.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# mysqladmin -uroot -p shutdown
Enter password: oracle
<date-and-time> mysqld_safe mysqld from pid file
/var/run/mysqld/mysqld.pid ended
[1]+  Done                      mysqld_safe
...
```

2. Inspect the contents of the /labs/service/mysql-5.7/mysqld.service file. This is the service unit configuration file.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# cat /labs/service/mysql-5.7/mysqld.service
[Unit]
Description=MySQL Server
Documentation=man:mysqld(7)
Documentation=http://dev.mysql.com/doc/refman/en/using-
systemd.html
After=network.target
After=syslog.target

[Install]
WantedBy=multi-user.target

[Service]
User=mysql
Group=mysql

Type=forking

PIDFile=/usr/local/mysql/data/mysqld.pid

# Disable service start and stop timeout logic of systemd for
mysqld service.
TimeoutSec=0

# Start main service
ExecStart=/usr/local/mysql/bin/mysqld --defaults-
file=/etc/my.cnf --daemonize
--pid-file=/usr/local/mysql/data/mysqld.pid $MYSQLD_OPTS
```

```
# Use this to switch malloc implementation
EnvironmentFile=/etc/sysconfig/mysql

# Sets open_files_limit
LimitNOFILE = 5000

Restart=on-failure

RestartPreventExitStatus=1

PrivateTmp=false
```

- Copy the mysqld.service file to the /usr/lib/systemd/system directory.

Enter the following command at the Linux terminal prompt:

```
# cp /labs/service/mysql-5.7/mysqld.service \
> /usr/lib/systemd/system
```

- Enter a systemctl command that enables the mysqld service to start automatically when the host machine reboots.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# systemctl enable mysqld.service
Created symlink from /etc/systemd/system/multi-
user.target.wants/mysqld.service to
/usr/lib/systemd/system/mysqld.service.
```

- Enter a systemctl command that starts the mysqld service.

Enter the following command at the Linux terminal prompt:

```
# systemctl start mysqld
```

- Enter a systemctl command that reports the status of the mysqld service.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# systemctl status mysqld
● mysqld.service - MySQL Server
   Loaded: loaded (/usr/lib/systemd/system/mysqld.service;
             enabled; vendor preset: disabled)
     Active: active (running) since ...
           Docs: man:mysqld(7)
                     http://dev.mysql.com/doc/refman/en/using-systemd.html
      Process: 4621 ExecStart=/usr/local/mysql/bin/mysqld --
                 defaults-file=/etc/my.cnf --daemonize --pid-
                 file=/var/run/mysqld/mysqld.pid $MYSQLD_OPTS (code=exited,
                 status=0/SUCCESS)
        Main PID: 4624 (mysqld)
       CGroup: /system.slice/mysqld.service
```

```
[4624 /usr/local/mysql/bin/mysqld --defaults-
file=/etc/my.cnf --daemonize --pid-
file=/var/run/mysqld/mysqld.pid

...
<date-and-time> edvmr1p0 systemd[1]: Started MySQL Server.
Hint: Some lines were ellipsized, use -l to show in full.
```

- The mysqld service is running.

Practice 2-4: Upgrading MySQL

Overview

In this practice, you will install MySQL Shell and run the `checkForServerUpgrade` utility. Then, you will perform an in-place upgrade of the existing MySQL 5.7 server to MySQL 8.0.

Duration

This practice should take you approximately 15 minutes to complete.

Tasks

1. Install MySQL Shell from the RPM file in the `/stage/MySQL-Shell` directory.
2. Run `mysqlsh` to perform a check for upgrade using the following command:

```
mysqlsh root@localhost -- util checkForServerUpgrade
```
3. Enter a `systemctl` command that stops the running `mysqld` service.
4. Verify that the `mysqld` service is not running.
5. Extract the MySQL Server 8.0 commercial edition binary archive from `/stage/MySQL-Server` to `/opt`.
6. List the contents of the `/opt` directory to verify that both versions of MySQL server are installed.
7. Remove the symbolic link to `/usr/local/mysql`, which you created in the practice titled “Installing MySQL.” It links to the directory that contains the older version of the MySQL server and is no longer required.
8. Create a symbolic link from the newly extracted MySQL directory to `/usr/local/mysql`.
9. Replace the existing `mysqld.service` file in the `/usr/lib/systemd/system` directory with the `/labs/service/mysql-8.0/mysqld.service` file.
10. Execute `systemctl daemon-reload` to make `systemd` aware of the new service unit configuration file.
11. Start the `mysqld` service to launch the new version of the server.
12. Verify that the `mysqld` service is running.
13. Run `mysql_upgrade` to complete the upgrade process and ensure that there are no incompatibilities between the versions. Is this step required?
14. Use `systemctl` to view the full logs during the startup of the MySQL server. Can you find the logs related to the upgrade process?
15. Verify that the new version of MySQL server is now running on your system.
16. Close all open Linux terminal windows.

Solution 2-4: Upgrading MySQL

Solution Steps

1. Install MySQL Shell from the RPM file in the /stage/MySQL-Shell directory.

Enter the following commands at the Linux terminal prompt and receive the results shown:

```
# cd /stage/MySQL-Shell
# yum -y install mysql-shell-commercial-8*.rpm
Loaded plugins: langpacks, ulninfo
Repository is over 2 weeks old. Install yum-cron? Or run: yum makecache fast
Examining mysql-shell-commercial-version.rpm: mysql-shell-commercial-version
Marking mysql-shell-commercial-version.rpm to be installed
Resolving Dependencies
--> Running transaction check
--> Package mysql-shell-commercial.x86_64 0:version will be installed
--> Finished Dependency Resolution
...
=====
Package           Arch   Version      Repository
Size
=====
Installing:
mysql-shell-commercial x86_64  version      /mysql-shell-commercial-
version 18 M

Transaction Summary
=====
Install 1 Package

Total size: 18 M
Installed size: 18 M
Downloading packages:
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : mysql-shell-commercial-version          1/1
  Verifying   : mysql-shell-commercial-version          1/1

Installed:
  mysql-shell-commercial.x86_64 0: version

Complete!
```

2. Run mysqlsh to perform a check for upgrade using the following command:

```
mysqlsh root@localhost -- util checkForServerUpgrade
```

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# mysqlsh root@localhost -- util checkForServerUpgrade
Please provide the password for 'root@localhost': oracle
Save password for 'root@localhost'? [Y]es/[N]o/[E]r[er] (default
No):
The MySQL server at localhost, version 5.7.25-enterprise-
commercial-advanced - MySQL Enterprise Server - Advanced Edition
(Commercial), will now be checked for compatibility issues for
upgrade to MySQL 8.0.16...
1) Usage of old temporal type
   No issues found

2) Usage of db objects with names conflicting with reserved
   keywords in 8.0
   No issues found

3) Usage of utf8mb3 charset
   No issues found

4) Table names in the mysql schema conflicting with new tables
   in 8.0
   No issues found

5) Foreign key constraint names longer than 64 characters
   No issues found

6) Usage of obsolete MAXDB sql_mode flag
   No issues found

7) Usage of obsolete sql_mode flags
   No issues found

8) ENUM/SET column definitions containing elements longer than
   255 characters
   No issues found

9) Usage of partitioned tables in shared tablespaces
   No issues found

10) Usage of removed functions
    No issues found

11) Usage of removed GROUP BY ASC/DESC syntax
    No issues found

12) Removed system variables for error logging to the system log
    configuration
```

To run this check requires full path to MySQL server configuration file to be specified at 'configPath' key of options dictionary

More information:
<https://dev.mysql.com/doc/relnotes/mysql/8.0/en/news-8-0-13.html#mysqld-8-0-13-logging>

13) Removed system variables
To run this check requires full path to MySQL server configuration file to be specified at 'configPath' key of options dictionary

More information:
<https://dev.mysql.com/doc/refman/8.0/en/added-deprecated-removed.html#optvars-removed>

14) System variables with new default values
To run this check requires full path to MySQL server configuration file to be specified at 'configPath' key of options dictionary

More information:
<https://mysqlserverteam.com/new-defaults-in-mysql-8-0/>

15) Schema inconsistencies resulting from file removal or corruption
No issues found

16) Issues reported by 'check table x for upgrade' command
No issues found

17) New default authentication plugin considerations
Warning: The new default authentication plugin 'caching_sha2_password' offers more secure password hashing than previously used 'mysql_native_password' (and consequent improved client connection authentication). However, it also has compatibility implications that may affect existing MySQL installations.
If your MySQL installation must serve pre-8.0 clients and you encounter compatibility issues after upgrading, the simplest way to address those issues is to reconfigure the server to revert to the previous default authentication plugin (mysql_native_password). For example, use these lines in the server option file:

```
[mysqld]
default_authentication_plugin=mysql_native_password
```

However, the setting should be viewed as temporary, not as a long term or permanent solution, because it causes new accounts created with the setting in effect to forego the improved authentication security.

If you are using replication please take time to understand how the authentication plugin changes may impact you.

More information:

<https://dev.mysql.com/doc/refman/8.0/en/upgrading-from-previous-series.html#upgrade-caching-sha2-password-compatibility-issues>

<https://dev.mysql.com/doc/refman/8.0/en/upgrading-from-previous-series.html#upgrade-caching-sha2-password-replication>

Errors: 0
Warnings: 1
Notices: 0

No fatal errors were found that would prevent an upgrade, but some potential issues were detected. Please ensure that the reported issues are not significant before upgrading.

- Take note of all the tests that has been performed.

3. Enter a `systemctl` command that stops the running `mysqld` service.

Enter the following command at the Linux terminal prompt:

```
# systemctl stop mysqld
```

4. Verify that the `mysqld` service is not running.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# systemctl status mysqld
● mysqld.service - MySQL Server
   Loaded: loaded (/usr/lib/systemd/system/mysqld.service;
             enabled; vendor preset: disabled)
     Active: inactive (dead) since ...
       Docs: man:mysqld(7)
                 http://dev.mysql.com/doc/refman/en/using-systemd.html
      Process: 4621 ExecStart=/usr/local/mysql/bin/mysqld --
                 defaults-file=/etc/my.cnf --daemonize --pid-
                 file=/var/run/mysqld/mysqld.pid $MYSQLD_OPTS (code=exited,
                 status=0/SUCCESS)
        Main PID: 4624 (code=exited, status=0/SUCCESS)
          ...
<date-and-time> edvmr1p0 systemd[1]: Stopped MySQL Server.
Hint: Some lines were ellipsized, use -l to show in full.
```

5. Extract the MySQL Server 8.0 commercial edition binary archive from `/stage/MySQL-Server` to `/opt`.

Enter the following commands at the Linux terminal prompt:

```
# cd /opt
# tar xf /stage/MySQL-Server/mysql-commercial-8*.tar.gz
```

- Your terminal might already be in the /opt directory at the start of this step.
 - The wildcard symbol expands to the complete file name, which includes the version and architecture of the archive on your host.
6. List the contents of the /opt directory to verify that both versions of MySQL server are installed.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# ls /opt
...
mysql-advanced-5.7.25-el7-x86_64
mysql-commercial-8.0.16-el7-x86_64
...
```

- The exact version numbers might be different on your machine. However, there should be a 5.7 and 8.0 version, and both must be general availability (GA) releases.
7. Remove the symbolic link to /usr/local/mysql, which you created in the practice titled “Installing MySQL.” It links to the directory that contains the older version of the MySQL server and is no longer required.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# rm /usr/local/mysql
rm: remove symbolic link `/usr/local/mysql'? y
```

8. Create a symbolic link from the newly extracted MySQL directory to /usr/local/mysql.

Enter the following command at the Linux terminal prompt:

```
# ln -s /opt/mysql-commercial-8* /usr/local/mysql
```

9. Replace the existing mysqld.service file in the /usr/lib/systemd/system directory with the /stage/MySQL-Server/service/mysql-8.0/mysqld.service file.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# cp /labs/service/mysql-8.0/mysqld.service \
> /usr/lib/systemd/system
cp: overwrite '/usr/lib/systemd/system/mysqld.service'? y
```

10. Execute systemctl daemon-reload to make systemd aware of the new service unit configuration file.

Enter the following command at the Linux terminal prompt:

```
# systemctl daemon-reload
```

11. Start the mysqld service to launch the new version of the server.

Enter the following command at the Linux terminal prompt:

```
# systemctl start mysqld
```

12. Verify that the `mysqld` service is running.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# systemctl status mysqld
● mysqld.service - MySQL Server
  Loaded: loaded (/usr/lib/systemd/system/mysqld.service; enabled; vendor preset: disabled)
  Active: active (running) since ...
    Docs: man:mysqld(8)
           http://dev.mysql.com/doc/refman/en/using-systemd.html
   Main PID: 6079 (mysqld)
     Status: "SERVER_OPERATING"
      CGroupl: /system.slice/mysqld.service
              └─6079 /usr/local/mysql/bin/mysqld --defaults-
file=/etc/my.cnf
...
-
```

- The service is running.

13. Run `mysql_upgrade` to complete the upgrade process and ensure that there are no incompatibilities between the versions. Is this step required? Enter the following command at the Linux terminal prompt and receive the results shown:

```
# mysql_upgrade -uroot -p
Enter password: oracle
The mysql_upgrade client is now deprecated. The actions executed by the upgrade client are now done by the server.
To upgrade, please start the new MySQL binary with the older data directory. Repairing user tables is done automatically. Restart is not required after upgrade.
The upgrade process automatically starts on running a new MySQL binary with an older data directory. To avoid accidental upgrades, please use the --upgrade=NONE option with the MySQL binary. The option --upgrade=FORCE is also provided to run the server upgrade sequence on demand.
It may be possible that the server upgrade fails due to a number of reasons. In that case, the upgrade sequence will run again during the next MySQL server start. If the server upgrade fails repeatedly, the server can be started with the --upgrade=MINIMAL option to start the server without executing the upgrade sequence, thus allowing users to manually rectify the problem.
```

- This step is not required because MySQL server 8.0.16 or later will perform all of the upgrade process automatically when you start the server.
- If you are upgrading to an older release before 8.0.16, you may see the following result:

```
# mysql_upgrade -uroot -p
Enter password: oracle
Checking if update is needed.
```

```
Checking server version.  
Running queries to upgrade MySQL server.  
Upgrading system table data.  
Checking system database.  
mysql.columns_priv                                OK  
mysql.component                                    OK  
mysql.db                                           OK  
mysql.default_roles                               OK  
mysql.engine_cost                                 OK  
mysql.func                                         OK  
mysql.general_log                                 OK  
mysql.global_grants                             OK  
mysql.gtid_executed                            OK  
mysql.help_category                           OK  
mysql.help_keyword                            OK  
mysql.help_relation                           OK  
mysql.help_topic                                OK  
mysql.innodb_index_stats                      OK  
mysql.innodb_table_stats                      OK  
mysql.ndb_binlog_index                        OK  
mysql.password_history                         OK  
mysql.plugin                                     OK  
mysql.procs_priv                                OK  
mysql.proxies_priv                             OK  
mysql.role_edges                                OK  
mysql.server_cost                               OK  
mysql.servers                                    OK  
mysql.slave_master_info                         OK  
mysql.slave_relay_log_info                      OK  
mysql.slave_worker_info                         OK  
mysql.slow_log                                   OK  
mysql.tables_priv                                OK  
mysql.time_zone                                 OK  
mysql.time_zone_leap_second                     OK  
mysql.time_zone_name                            OK  
mysql.time_zone_transition                     OK  
mysql.time_zone_transition_type                OK  
mysql.user                                       OK  
Found outdated sys schema version 1.5.1.  
Upgrading the sys schema.  
Checking databases.  
sys.sys_config                                  OK
```

```
Upgrade process completed successfully.  
Checking if update is needed.
```

14. Use `systemctl` to view the full logs during the startup of the MySQL server. Can you find the logs related to the upgrade process?

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# systemctl status mysqld -l  
● mysqld.service - MySQL Server  
    Loaded: loaded (/usr/lib/systemd/system/mysqld.service;  
            enabled; vendor preset: disabled)  
    Active: active (running) since ...  
      ...  
date-and-time localhost.localdomain mysqld[6079]: date-and-time  
1 [System] [MY-011012] [Server] Starting upgrade of data  
directory.  
date-and-time localhost.localdomain mysqld[6079]: date-and-time  
2 [System] [MY-011003] [Server] Finished populating Data  
Dictionary tables with data.  
date-and-time localhost.localdomain mysqld[6079]: date-and-time  
5 [System] [MY-013381] [Server] Server upgrade from '50700' to  
'80016' started.  
date-and-time localhost.localdomain mysqld[6079]: date-and-time  
5 [System] [MY-013381] [Server] Server upgrade from '50700' to  
'80016' completed.  
...
```

15. Verify that the new version of MySQL server is now running on your system.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# mysqld --version  
/opt/mysql-commercial-8.0.16-el7-x86_64/bin/mysqld Ver 8.0.16-  
commercial for el7 on x86_64 (MySQL Enterprise Server - Commercial)
```

Note: The version on your machine might be different, but should be version 8.0.16 or later.

16. Close all open Linux terminal windows.

Practices for Lesson 3: Understanding MySQL Architecture

Practices for Lesson 3: Overview

Overview

In these practices, you explore the architecture of MySQL in terms of storage and memory requirements.

Assumptions

- You have completed the activities in the lesson titled “Installing and Upgrading MySQL” and have a Linux terminal window open as the `root` Linux user.
- SELinux is disabled (or permissive), and no other mandatory access control systems prevent MySQL from writing to the Linux file system. This is handled for you in Oracle classroom environments.

Practice 3-1: Configuring Tablespaces

Overview

In this practice, you create tables in different tablespaces and view the effects on the file system.

Note: You use a Linux command line and the `mysql` client for this practice. Use two terminals, one for the command line and one logged in to the `mysql` client.

Duration

This practice should take you approximately 20 minutes to complete.

Tasks

1. Open two Linux terminal windows. This practice refers to these terminal windows as `t1` and `t2`.
2. In `t1`, view the contents of the data directory.
3. In `t2`, launch the `mysql` client.
4. In `t2`, create a new database called `ts_test`.
5. In `t2`, create a table called `table1` in the new database, with a single integer column called `a`.
6. In `t2`, execute a `SHOW CREATE TABLE` statement to view the full statement that creates the table.
7. In `t1`, view the contents of the data directory again.
8. In `t1`, view the contents of the `ts_test` database directory.
9. In `t2`, create a tablespace called `general` with the data file `general.ibd`.
10. In `t2`, create a table in the new tablespace called `table2` with a single integer column called `b`.
11. In `t2`, execute a `SHOW CREATE TABLE` statement to view the full statement that creates the `table2` table.
12. In `t1`, view the contents of the data directory and the `ts_test` database directory again.
13. In `t1`, create a new directory in the root of the file system called `/tablespaces` and grant its ownership to the `mysql` user and group.
14. In `t1`, add the new `/tablespaces` directory to the `innodb_tablespaces` option in the `/etc/my.cnf` file and restart the MySQL server.
15. In `t2`, create a new general tablespace called `external` with a data file called `external.ibd` in the `/tablespaces` directory.
16. In `t2`, create a table in the new `external` tablespace called `table3` with a single integer column called `c`.
17. In `t1`, view the contents of the data directory and the `ts_test` database directory again.

18. In t1, view the contents of the /tablespaces directory.
19. In t2, create a table in its own default tablespace in the /tablespaces directory called table4 with a single integer column called d.
20. In t1, view the contents of the data directory, the ts_test database directory, and the /tablespaces directory again.
21. In t1, view the contents of the new /tablespaces/ts_test directory.
22. In t2, create a table called table5 in the system tablespace, with an integer column called e.
23. In t1, view the contents of the data directory, the ts_test database directory, and the /tablespaces/ts_test directory again.
24. In t2, drop the ts_test database.
25. In t1, view the changes in the file system.
26. In t2, drop the general and external tablespaces.
27. In t1, view the changes in the file system.
28. In t1, delete the /tablespaces/ts_test and /tablespaces directories.
29. Keep the mysql client and Linux terminal windows open for the next practice.

Solution 3-1: Configuring Tablespaces

Solution Steps

1. Open two Linux terminal windows. This practice refers to these terminal windows as t1 and t2.
2. In t1, view the contents of the data directory.

Enter the following command at the t1 Linux terminal prompt and receive the results shown:

```
# ls /var/lib/mysql
auto.cnf           ibdata1          performance_schema
binlog.000001      ib_logfile0     private_key.pem
binlog.000002      ib_logfile1     public_key.pem
binlog.index       ibtmp1          server-cert.pem
ca-key.pem         #innodb_temp    server-key.pem
ca.pem             mysql           sys
client-cert.pem   mysql.ibd      undo_001
client-key.pem    mysql.sock     undo_002
hostname.pid       mysql.sock.lock
ib_buffer_pool     mysql_upgrade_info
```

Note the following points:

- There are three database directories: mysql, performance_schema, and sys.
 - The system tablespace is contained within the ibdata1 file.
 - The temporary tablespace is contained within the ibtmp1 file.
 - The InnoDB redo logs are in the ib_logfile0 and ib_logfile1 files.
 - The PID file (shown as hostname.pid in the preceding output) has a default name that includes your host's name.
3. In t2, launch the mysql client.

Enter the following command at the t2 Linux terminal prompt and receive the results shown:

```
# mysql -uroot -p
Enter password: oracle
Welcome to the MySQL monitor. Commands end with ; or \g.
...
mysql>
```

4. In t2, create a new database called ts_test.

Enter the following statement at the mysql prompt in t2, and receive the results shown:

```
mysql> CREATE DATABASE ts_test;
Query OK, 1 row affected (#.# sec)
```

5. In t2, create a table called `table1` in the new database, with a single integer column called `a`.

Enter the following statements at the `mysql` prompt in t2, and receive the results shown:

```
mysql> USE ts_test
Database changed
mysql> CREATE TABLE table1 (a INT);
Query OK, 0 rows affected (#.# sec)
```

6. In t2, execute a `SHOW CREATE TABLE` statement to view the full statement that creates the table.

Enter the following statement at the `mysql` prompt in t2, and receive the results shown:

```
mysql> SHOW CREATE TABLE table1\G
***** 1. row *****
Table: table1
Create Table: CREATE TABLE `table1` (
  `a` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
          COLLATE=utf8mb4_0900_ai_ci
1 row in set (#.# sec)
```

- The table uses the InnoDB storage engine, and there are no `TABLESPACE` table options.

7. In t1, view the contents of the data directory again.

Enter the following command at the t1 Linux terminal prompt and receive the results shown:

```
# ls /var/lib/mysql
auto.cnf          ibdata1           performance_schema
binlog.000001     ib_logfile0      private_key.pem
binlog.000002     ib_logfile1      public_key.pem
binlog.index       ibtmp1           server-cert.pem
ca-key.pem        #innodb_temp    server-key.pem
ca.pem            mysql             sys
client-cert.pem   mysql.ibd       ts_test
client-key.pem    mysql.sock      undo_001
edvmr1p0.pid      mysql.sock.lock undo_002
ib_buffer_pool    mysql_upgrade_info
```

- There is a new directory containing the new `ts_test` database.

8. In t1, view the contents of the `ts_test` database directory.

Enter the following command at the t1 Linux terminal prompt and receive the results shown:

```
# ls /var/lib/mysql/ts_test
table1.ibd
```

- The database directory contains a tablespace file for the `table1` table.

9. In t₂, create a tablespace called general with the data file general.ibd.

Enter the following statement at the mysql prompt in t₂ and receive the results shown:

```
mysql> CREATE TABLESPACE general ADD DATAFILE 'general.ibd';
Query OK, 0 rows affected (#.## sec)
```

10. In t₂, create a table in the new tablespace called table2 with a single integer column called b.

Enter the following statement at the mysql prompt in t₂ and receive the results shown:

```
mysql> CREATE TABLE table2 (b int) TABLESPACE=general;
Query OK, 0 rows affected (#.## sec)
```

11. In t₂, execute a SHOW CREATE TABLE statement to view the full statement that creates the table2 table.

Enter the following statement at the mysql prompt in t₂ and receive the results shown:

```
mysql> SHOW CREATE TABLE table2\G
***** 1. row *****
      Table: table2
Create Table: CREATE TABLE `table2` (
    `b` int(11) DEFAULT NULL
) /*!50100 TABLESPACE `general` */ ENGINE=InnoDB
          DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
1 row in set (#.## sec)
```

- The table definition contains a TABLESPACE table option that refers to the general tablespace.

12. In t₁, view the contents of the data directory and the ts_test database directory again.

Enter the following commands at the t₁ Linux terminal prompt and receive the results shown:

```
# ls /var/lib/mysql
auto.cnf          ib_buffer_pool      mysql_upgrade_info
binlog.000001     ibdata1            performance_schema
binlog.000002     ib_logfile0        private_key.pem
binlog.index       ib_logfile1        public_key.pem
ca-key.pem        ibtmp1             server-cert.pem
ca.pem            #innodb_temp       server-key.pem
client-cert.pem   mysql              sys
client-key.pem    mysql.ibd         ts_test
edvmr1p0.pid      mysql.sock        undo_001
general.ibd      mysql.sock.lock  undo_002
# ls /var/lib/mysql/ts_test
table1.ibd
```

- The data directory contains a new .ibd tablespace file for the general tablespace.
- The ts_test database directory contains no tablespace file for the new table.

13. In t1, create a new directory in the root of the file system called /tablespaces and grant its ownership to the mysql user and group.

Enter the following commands at the t1 Linux terminal prompt and receive the results shown:

```
# mkdir /tablespaces
# chown mysql:mysql /tablespaces
```

14. In t1, add the new /tablespaces directory to the innodb_tablespaces option in the /etc/my.cnf file and restart the MySQL server.

Edit the /etc/my.cnf file and add the line shown in bold near the top.

```
[mysqld]
innodb_directories="/tablespaces"
datadir=/var/lib/mysql
...
```

Enter the following command at the t1 Linux terminal prompt:

```
# systemctl restart mysqld
```

15. In t2, create a new general tablespace called external with a data file called external.ibd in the /tablespaces directory.

Enter the following statement at the mysql prompt in t2 and receive the results shown:

```
mysql> CREATE TABLESPACE external
      -> ADD DATAFILE '/tablespaces/external.ibd';
ERROR 2006 (HY000): MySQL server has gone away
No connection. Trying to reconnect...
Connection id:    #
Current database: ts_test
Query OK, 0 rows affected (#.## sec)
```

- When you restart a running MySQL server, the mysql client loses its connection and must reconnect before executing a statement.

Note: If MySQL does not have permissions to write to the /tablespaces directory, or if it is prevented from doing so by a mandatory access control system such as SELinux, the preceding command fails.

16. In t2, create a table in the new external tablespace called table3 with a single integer column called c.

Enter the following statement at the mysql prompt in t2 and receive the results shown:

```
mysql> CREATE TABLE table3 (c int) TABLESPACE=external;
Query OK, 0 rows affected (#.## sec)
```

17. In t1, view the contents of the data directory and the ts_test database directory again.

Enter the following commands at the t1 Linux terminal prompt and receive the results shown:

```
# ls /var/lib/mysql
auto.cnf          ib_buffer_pool      performance_schema
binlog.000001     ibdata1           private_key.pem
binlog.000002     ib_logfile0       public_key.pem
binlog.000003     ib_logfile1       server-cert.pem
binlog.index      ibtmp1            server-key.pem
ca-key.pem        #innodb_temp      sys
ca.pem            mysql             ts_test
client-cert.pem   mysql.ibd         undo_001
client-key.pem    mysql.sock        undo_002
edvmr1p0.pid      mysql.sock.lock
general.ibd       mysql_upgrade_info
# ls /var/lib/mysql/ts_test
table1.ibd
```

- There are no visible artefacts that refer to table3 in the data and database directories.

18. In t1, view the contents of the /tablespaces directory.

Enter the following command at the t1 Linux terminal prompt and receive the results shown:

```
# ls /tablespaces/
external.ibd
```

- There is a data file for the new tablespace.

19. In t2, create a table in its own default tablespace in the /tablespaces directory called table4 with a single integer column called d.

Enter the following statement at the mysql prompt in t2 and receive the results shown:

```
mysql> CREATE TABLE table4 (d INT)
      -> DATA DIRECTORY='/tablespaces';
Query OK, 0 rows affected (#.## sec)
```

20. In t1, view the contents of the data directory, the ts_test database directory, and the /tablespaces directory again.

Enter the following command at the t1 Linux terminal prompt and receive the results shown:

```
# ls /var/lib/mysql
auto.cnf          ib_buffer_pool      performance_schema
binlog.000001     ibdata1           private_key.pem
binlog.000002     ib_logfile0       public_key.pem
binlog.000003     ib_logfile1       server-cert.pem
binlog.index      ibtmp1            server-key.pem
ca-key.pem        #innodb_temp      sys
ca.pem            mysql             ts_test
client-cert.pem   mysql.ibd         undo_001
client-key.pem    mysql.sock        undo_002
edvmr1p0.pid      mysql.sock.lock
general.ibd       mysql_upgrade_info
# ls /var/lib/mysql/ts_test
table1.ibd
# ls /tablespaces/
external.ibd     ts_test
```

Note the following points:

- There are no changes in the data or database directories.
- The /tablespaces directory contains a new ts_test directory.

21. In t1, view the contents of the new /tablespaces/ts_test directory.

Enter the following command at the t1 Linux terminal prompt and receive the results shown:

```
# ls /tablespaces/ts_test/
table4.ibd
```

- The directory contains the file-per-table tablespace for the table4 table.

22. In t2, create a table called table5 in the system tablespace, with an integer column called e.

Enter the following statement at the mysql prompt in t2 and receive the results shown:

```
mysql> CREATE TABLE table5 (e int) TABLESPACE=innodb_system;
Query OK, 0 rows affected (#.## sec)
```

23. In t1, view the contents of the data directory, the ts_test database directory, and the /tablespaces/ts_test directory again.

Enter the following commands at the t1 Linux terminal prompt and receive the results shown:

```
# ls /var/lib/mysql
auto.cnf          ib_buffer_pool      performance_schema
binlog.000001     ibdata1           private_key.pem
binlog.000002     ib_logfile0       public_key.pem
binlog.000003     ib_logfile1       server-cert.pem
binlog.index      ibtmp1            server-key.pem
ca-key.pem        #innodb_temp      sys
ca.pem            mysql             ts_test
client-cert.pem   mysql.ibd        undo_001
client-key.pem    mysql.sock       undo_002
edvmr1p0.pid      mysql.sock.lock
general.ibd       mysql_upgrade_info
# ls /var/lib/mysql/ts_test/
table1.ibd
# ls /tablespaces/ts_test/
table4.ibd
```

- There are no new tablespace files created. The table5 table is in the InnoDB system tablespace.

24. In t2, drop the ts_test database.

Enter the following statement at the mysql prompt in t2 and receive the results shown:

```
mysql> DROP DATABASE ts_test;
Query OK, 5 rows affected (#.# sec)
```

25. In t1, view the changes in the file system.

Enter the following commands at the t1 Linux terminal prompt and receive the results shown:

```
# ls /var/lib/mysql
auto.cnf          general.ibd      mysql.sock.lock
binlog.000001     ib_buffer_pool  mysql_upgrade_info
binlog.000002     ibdata1         performance_schema
binlog.000003     ib_logfile0     private_key.pem
binlog.index      ib_logfile1     public_key.pem
ca-key.pem        ibtmp1          server-cert.pem
ca.pem            #innodb_temp    server-key.pem
client-cert.pem   mysql           sys
client-key.pem    mysql.ibd      undo_001
edvmr1p0.pid      mysql.sock     undo_002
# ls /tablespaces/
external.ibd     ts_test
# ls /tablespaces/ts_test/
```

- The ts_test database directory in the data directory no longer exists.
- The /tablespaces directory contains an empty ts_test directory.

26. In t2, drop the general and external tablespaces.

Enter the following statement at the mysql prompt in t2 and receive the results shown:

```
mysql> DROP TABLESPACE general;
Query OK, 0 rows affected (#.## sec)

mysql> DROP TABLESPACE external;
Query OK, 0 rows affected (#.## sec)
```

27. In t1, view the changes in the file system.

Enter the following command at the t1 Linux terminal prompt and receive the results shown:

```
# ls /var/lib/mysql
auto.cnf          ib_buffer_pool      mysql_upgrade_info
binlog.000001     ibdata1           performance_schema
binlog.000002     ib_logfile0       private_key.pem
binlog.000003     ib_logfile1       public_key.pem
binlog.index      ibtmp1            server-cert.pem
ca-key.pem        #innodb_temp      server-key.pem
ca.pem            mysql             sys
client-cert.pem   mysql.ibd        undo_001
client-key.pem    mysql.sock       undo_002
edvmr1p0.pid     mysql.sock.lock
# ls /tablespaces/
ts_test
```

- The general and external tablespace files are no longer in their previous locations.

28. In t1, delete the /tablespaces/ts_test and /tablespaces directories.

Enter the following commands at the t1 Linux terminal prompt:

```
# rmdir /tablespaces/ts_test/
# rmdir /tablespaces/
```

29. Keep the mysql client (t2) and Linux terminal (t1) windows open for the next practice.

Practice 3-2: Configuring the Buffer Pool

Overview

In this practice, you view the default buffer pool configuration and modify it dynamically and in the MySQL configuration file.

Note: In the sample steps, the output shown in the solution might be different from the output you receive on your machine.

Duration

This practice should take you approximately 10 minutes to complete.

Tasks

1. In t1, run the `free` command to see how much free memory is available on your system.
2. In t2, run a `SHOW VARIABLES LIKE` command to view the current size of the InnoDB buffer pool in bytes.
3. In t2, change the size of the InnoDB buffer pool to a value that is approximately 75% of your system's RAM, in bytes.
4. In t2, view the changed size of the InnoDB buffer pool.
5. In t2, view the number of buffer pool instances.
6. In t1, edit the `[mysqld]` option group in the `/etc/my.cnf` file and set the following option values:
 - Set the buffer pool size to a round number of GB that is less than or equal to 75% of your system RAM (example: 12 GB).
 - Set the number of buffer pool instances so that you have one instance per GB in your buffer pool (example: 12).Save and close the file before performing the following step.
7. In t1, restart the MySQL service.
8. In t2, view the InnoDB buffer size and the number of buffer pool instances again.
9. Exit all `mysql` client sessions and close all Linux terminal windows.

Solution 3-2: Configuring the Buffer Pool

Solution Steps

1. In t1, run the `free` command to see how much free memory is available on your system.

Enter the following command at the t1 Linux terminal prompt and receive results similar to those shown:

```
# free
total        used        free      shared   ...
Mem:    16341084  7887536  8453548          0   ...
Swap:   8388604       0  8388604
```

- Linux uses available memory to improve the performance of I/O operations with write buffers and read caches.
- In the preceding output, the machine has 16 GB of RAM of which approximately 7.5 GB memory is being used and 8.5 GB is free. Your machine will show different numbers.

2. In t2, run a `SHOW VARIABLES LIKE` command to view the current size of the InnoDB buffer pool in bytes.

Enter the following statement at the `mysql` prompt in t2 and receive the results shown:

```
mysql> SHOW VARIABLES LIKE 'innodb_buffer_pool_size';
+-----+-----+
| Variable_name      | Value   |
+-----+-----+
| innodb_buffer_pool_size | 134217728 |
+-----+-----+
1 row in set (#.# sec)
```

- The default size of the InnoDB buffer pool is 128 MB.
3. In t2, change the size of the InnoDB buffer pool to a value that is approximately 75% of your system's RAM, in bytes.

Enter a statement similar to the following at the `mysql` prompt in t2, using a value that is approximately 75% of your system's RAM, and receive the results shown:

```
mysql> SET GLOBAL innodb_buffer_pool_size=12000 * 1024 * 1024;
Query OK, 0 rows affected (#.# sec) 1 warning (#.# sec)
```

- 75% of the RAM shown in step 1 (16 GB) is approximately 12 GB (12,000 MB.)

Note: If the new buffer pool size is not a multiple of 128 MB (the value of `innodb_buffer_pool_chunk_size`), the operation rounds up to the next multiple of 128 MB and produces a warning.

4. In t2, view the changed size of the InnoDB buffer pool.

Enter the following statement at the mysql prompt in t2 and receive the results shown:

```
mysql> SHOW VARIABLES LIKE 'innodb_buffer_pool_size';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| innodb_buffer_pool_size | 12884901888 |
+-----+-----+
1 row in set (0.01 sec)
```

5. In t2, view the number of buffer pool instances.

Enter the following statement at the mysql prompt in t2 and receive the results shown:

```
mysql> SHOW VARIABLES LIKE 'innodb_buffer_pool_instances';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| innodb_buffer_pool_instances | 1 |
+-----+-----+
1 row in set (#.# sec)
```

- The innodb_buffer_pool_instances variable is not a dynamic variable, so you cannot change it on a running system.

6. In t1, edit the [mysqld] option group in the /etc/my.cnf file and set the following option values:

- Set the buffer pool size to a round number of GB that is less than or equal to 75% of your system's RAM (example: 12 GB).
- Set the number of buffer pool instances so that you have one instance per GB in your buffer pool (example: 12).

Edit the /etc/my.cnf file and add the following highlighted lines near the top of the file.

```
[mysqld]
innodb_buffer_pool_size=12G
innodb_buffer_pool_instances=12
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
...
```

Save and close the file before performing the following step.

7. In t1, restart the MySQL service.

Enter the following command at the t1 Linux terminal prompt:

```
# systemctl restart mysqld
```

8. In t2, view the InnoDB buffer size and the number of buffer pool instances again.
Enter the following statements at the mysql prompt in t2 and receive the results shown:

```
mysql> SHOW VARIABLES LIKE 'innodb_buffer_pool_size';
ERROR 2006 (HY000): MySQL server has gone away
No connection. Trying to reconnect...
Connection id:      #
Current database: *** NONE ***

+-----+-----+
| Variable_name          | Value   |
+-----+-----+
| innodb_buffer_pool_size | 12884901888 |
+-----+-----+
1 row in set (#.# sec)

mysql> SHOW VARIABLES LIKE 'innodb_buffer_pool_instances';
+-----+-----+
| Variable_name          | Value   |
+-----+-----+
| innodb_buffer_pool_instances | 12      |
+-----+-----+
1 row in set (#.# sec)
```

- The InnoDB buffer pool options that you placed in the configuration file have been applied.
9. Exit all mysql client sessions and close all Linux terminal windows.

Practices for Lesson 4: Configuring MySQL

Practices for Lesson 4: Overview

Overview

In these practices, you configure the MySQL client and servers by using command-line arguments, dynamic system variables, and configuration files.

Assumptions

- SELinux is disabled (or permissive), and no other mandatory access control systems prevent MySQL from writing to the Linux file system or using nondefault port numbers.
- The `/labs/multi.cnf` file contains configuration settings for four MySQL instances controlled by `systemd mysqld@.service`.

Practice 4-1: Modifying a Setting by Using Command-Line Arguments

Overview

In this practice, you change MySQL settings by launching the server process with command-line arguments.

Duration

This practice should take you approximately 10 minutes to complete.

Tasks

1. Open two Linux terminal windows. This practice refers to these terminal windows as t₁ and t₂.
2. Launch a mysql client session in terminal t₁ and enter SQL commands to display the values of the port and max_connections system variables.
3. In terminal window t₂, use mysqladmin to stop the MySQL service.
4. In t₂, start the MySQL daemon as a background service by using the mysqld command, specifying command-line options for the following settings:
 - Port: 3300
 - Maximum number of connections: 50Note the process ID of the newly started process.
5. In t₁, display the values of the port and max_connections system variables.
6. In t₁, shut down the mysqld process by executing the SHUTDOWN command at the mysql prompt. Observe the status of the mysqld process that you started in t₂.
7. In t₂, start the MySQL daemon as a background service by using the mysqld command, specifying command-line options for the following settings:
 - Port: 3301
 - Maximum number of connections: 51
8. In t₁, display the values of the port and max_connections system variables.
9. In t₁, shut down the mysqld process by executing the SHUTDOWN command at the mysql prompt.
10. In t₂, use systemctl to start the mysqld process and verify that it is running.
11. At the mysql prompt in t₁, verify that the port and maximum_connections system variables are set to the default values (3306 and 151, respectively.)
12. Leave the t₁ and t₂ terminal windows open for the next practice.

Solution 4-1: Modifying a Setting by Using Command-Line Arguments

Solution Steps

1. Open two Linux terminal windows. This practice refers to these terminal windows as t1 and t2.
2. Launch a mysql client session in terminal t1 and enter SQL commands to display the values of the port and max_connections system variables.

Enter the following command at the t1 Linux terminal prompt and receive the results shown:

```
# mysql -uroot -p  
Enter password: oracle  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is ...  
  
mysql>
```

Enter the following statements at the mysql prompt in t1 and receive the results shown:

```
mysql> SHOW VARIABLES LIKE 'port';  
+-----+-----+  
| Variable_name | Value |  
+-----+-----+  
| port | 3306 |  
+-----+-----+  
1 row in set (#.# sec)  
  
mysql> SHOW VARIABLES LIKE 'max_connections';  
+-----+-----+  
| Variable_name | Value |  
+-----+-----+  
| max_connections | 151 |  
+-----+-----+  
1 row in set (#.# sec)
```

3. In terminal window t2, use mysqladmin to stop the MySQL service.

Enter the following command at the t2 Linux terminal prompt:

```
# mysqladmin -uroot -p shutdown  
Enter password: oracle
```

4. In t₂, start the MySQL daemon as a background service by using the `mysqld` command, specifying command-line options for the following settings:

- Port: 3300
- Maximum number of connections: 50

Note the process ID of the newly started process.

Enter the following command at the t₂ Linux terminal prompt and receive the results shown:

```
# mysqld --port=3300 --max-connections=50 &
[1] 25387
# date-and-time 0 [Warning] [MY-011070] [Server] 'Disabling
symbolic links using --skip-symbolic-links (or equivalent) is
the default. Consider not using this option as it' is deprecated
and will be removed in a future release.
...
```

- The ampersand (&) at the end of the command forces it to run in the background. You can return control to the terminal by pressing the Enter key.
- The process ID in the preceding output is 25387. The `mysqld` process reads the configuration for the socket file and other settings from the standard configuration file `/etc/my.cnf`.

5. In t₁, display the values of the `port` and `max_connections` system variables.

Enter the following statements at the `mysql` prompt in t₁ and receive the results shown:

```
mysql> SHOW VARIABLES LIKE 'port';
ERROR 2006 (HY000): MySQL server has gone away
No connection. Trying to reconnect...
Connection id: 8
Current database: *** NONE ***

+-----+-----+
| Variable_name | Value |
+-----+-----+
| port          | 3300  |
+-----+-----+
1 row in set (#.# sec)

mysql> SHOW VARIABLES LIKE 'max_connections';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| max_connections | 50    |
+-----+-----+
1 row in set (#.# sec)
```

- The mysql client displays a “server has gone away” message when it reconnects to a server that you have restarted.
 - The values of the variables have changed to those you specified with the command-line options you provided in step 4.
6. In t1, shut down the mysqld process by executing the SHUTDOWN command at the mysql prompt. Observe the status of the mysqld process that you started in t2.

Enter the following statement at the mysql prompt in t1 and receive the results shown:

```
mysql> SHUTDOWN;
Query OK, 0 rows affected (#.## sec)
```

The SHUTDOWN command shuts down the server but does not shut down the client. The client reconnects to the server the next time you run a command.

Switch to terminal window t2 and press Enter to return control to the terminal. You should see output similar to the following:

```
# date-and-time 0 [Warning] [MY-011070] [Server] 'Disabling
symbolic links using --skip-symbolic-links (or equivalent) is
the default. Consider not using this option as it' is deprecated
and will be removed in a future release.

...
# date-and-time 0 [System] [MY-010910] [Server] /opt/mysql-
commercial-version/bin/mysqld: Shutdown complete (mysqld
version) MySQL Enterprise Server - Commercial.

[1]+ Done mysqld --port=3300 --max-
connections=50
```

7. In t2, start the MySQL daemon as a background service by using the mysqld command, specifying command-line options for the following settings:
- Port: 3301
 - Maximum number of connections: 51

```
# mysqld --port=3301 --max-connections=51 &
[1] 25587
[root@edvmrlp0 ~]# date-and-time 0 [Warning] [MY-011070]
[Server] 'Disabling symbolic links using --skip-symbolic-links
(or equivalent) is the default. Consider not using this option
as it' is deprecated and will be removed in a future release.
date-and-time 0 [System] [MY-010116] [Server] /opt/mysql-
commercial-version/bin/mysqld (mysqld version-commercial)
starting as process 25587
date-and-time 0 [Warning] [MY-010068] [Server] CA certificate
ca.pem is self signed.
date-and-time 0 [System] [MY-010931] [Server] /opt/mysql-
commercial-version/bin/mysqld: ready for connections. Version:
'version' socket: '/var/lib/mysql/mysql.sock' port: 3301
MySQL Enterprise Server - Commercial.
```

8. In t1, display the values of the port and max_connections system variables.

Enter the following statements at the mysql prompt in t1 and receive the results shown:

```
mysql> SHOW VARIABLES LIKE 'port';
ERROR 2006 (HY000): MySQL server has gone away
No connection. Trying to reconnect...
Connection id:      8
Current database: *** NONE ***

+-----+-----+
| Variable_name | Value |
+-----+-----+
| port          | 3301  |
+-----+-----+
1 row in set (#.## sec)

mysql> SHOW VARIABLES LIKE 'max_connections';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| max_connections | 51   |
+-----+-----+
1 row in set (#.## sec)
```

9. In t1, shut down the mysqld process by executing the SHUTDOWN command at the mysql prompt.

Enter the following statement at the mysql prompt in t1 and receive the results shown:

```
mysql> SHUTDOWN;
Query OK, 0 rows affected (#.## sec)
```

10. In t2, use systemctl to start the mysqld process and verify that it is running.

Enter the following commands at the t2 Linux terminal prompt and receive the results shown:

```
# systemctl start mysqld
# systemctl status mysqld
● mysqld.service - MySQL Server
   Loaded: loaded (/usr/lib/systemd/system/mysqld.service;
             enabled; vendor preset: disabled)
     Active: active (running) since date-and-time UTC; duration
             ago
       Docs: man:mysqld(8)
             http://dev.mysql.com/doc/refman/en/using-systemd.html
    Main PID: 25644 (mysqld)
   Status: "SERVER OPERATING"
```

```
CGroup: /system.slice/mysqld.service
└─25644 /usr/local/mysql/bin/mysqld --defaults-
file=/etc/my.cnf

date-and-time edvmrlp0 systemd[1]: Starting MySQL Server...
...
date-and-time edvmrlp0 systemd[1]: Started MySQL Server.
Hint: Some lines were ellipsized, use -l to show in full.
```

11. At the mysql prompt in t1, verify that the port and maximum_connections system variables are set to the default values (3306 and 151, respectively).

Enter the following statement at the mysql prompt in t1 and receive the results shown:

```
mysql> SHOW VARIABLES LIKE 'port';
ERROR 2006 (HY000): MySQL server has gone away
No connection. Trying to reconnect...
Connection id: 8
Current database: *** NONE ***

+-----+-----+
| Variable_name | Value |
+-----+-----+
| port          | 3306  |
+-----+-----+
1 row in set (#.# sec)

mysql> SHOW VARIABLES LIKE 'max_connections';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| max_connections | 151   |
+-----+-----+
1 row in set (#.# sec)
```

12. Leave the t1 and t2 terminal windows open for the next practice.

Practice 4-2: Modifying the Configuration File

Overview

In this practice, you change MySQL settings by editing the `/etc/my.cnf` configuration file.

Duration

This practice should take you approximately five minutes to complete.

Tasks

1. In `t2`, use a text editor to edit the `/etc/my.cnf` file, adding options that configure the following settings:
 - Port: 3302
 - Maximum number of connections: 52
2. In `t2`, use `systemctl` to restart the MySQL service.
3. In the `mysql` prompt in terminal window `t1`, enter SQL commands to display the values of the `port` and `max_connections` variables.
4. In `t2`, use a text editor to edit the `/etc/my.cnf` file, removing the `port` and `max_connections` settings that you configured in step 1.
5. Close terminal window `t2`.
6. At the `mysql` prompt in terminal window `t1`, execute `RESTART` to restart the MySQL server.
7. Verify that the `port` and `max_connections` system variables are set to the default values (3306 and 151, respectively).
8. Leave the `mysql` client session open for the next practice.

Solution 4-2: Modifying the Configuration File

Solution Steps

1. In t2, use a text editor to edit the /etc/my.cnf file, adding options that configure the following settings:

- Port: 3302
- Maximum number of connections: 52

Modify the contents of the /etc/my.cnf file so that the first lines read as follows:

```
[mysqld]
port=3302
max-connections=52
innodb_buffer_pool_size=...
```

2. In t2, use systemctl to restart the MySQL service.

Enter the following commands at the t2 Linux terminal prompt and receive the results shown:

```
# systemctl restart mysqld
# systemctl status mysqld
● mysqld.service - MySQL Server
   Loaded: loaded (/usr/lib/systemd/system/mysqld.service; enabled; vendor preset: disabled)
     Active: active (running) since date-and-time UTC; duration
ago
      Docs: man:mysqld(8)
              http://dev.mysql.com/doc/refman/en/using-systemd.html
   Main PID: 25644 (mysqld)
     Status: "SERVER_OPERATING"
    CGroup: /system.slice/mysqld.service
              └─25644 /usr/local/mysql/bin/mysqld --defaults-
file=/etc/my.cnf

date-and-time edvmrlp0 systemd[1]: Starting MySQL Server...
...
date-and-time edvmrlp0 systemd[1]: Started MySQL Server.
Hint: Some lines were ellipsized, use -l to show in full.
```

3. In the mysql prompt in terminal window t1, enter SQL commands to display the values of the port and max_connections variables.

Enter the following statements at the mysql prompt in t1 and receive the results shown:

```
mysql> SHOW VARIABLES LIKE 'port';
ERROR 2006 (HY000): MySQL server has gone away
No connection. Trying to reconnect...
Connection id: 8
Current database: *** NONE ***

+-----+-----+
| Variable_name | Value |
+-----+-----+
| port          | 3302  |
+-----+-----+
1 row in set (#.# sec)

mysql> SHOW VARIABLES LIKE 'max_connections';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| max_connections | 52   |
+-----+-----+
1 row in set (#.# sec)
```

- The variables have the values that you set in the option file.
4. In t2, use a text editor to edit the /etc/my.cnf file, removing the port and max_connections settings that you configured in step 1.

The /etc/my.cnf file should now contain settings similar to those that follow, with any differences that you set in preceding practices:

```
[mysqld]
innodb_buffer_pool_size=12G
innodb_buffer_pool_instances=12
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Disabling symbolic-links is recommended to prevent assorted
# security risks
symbolic-links=0
innodb_directories='/tablespaces'
```

```
[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid

[client]
socket=/var/lib/mysql/mysql.sock
```

5. Close terminal window t2.

Enter the following command at the t2 Linux terminal prompt:

```
# exit
```

6. At the mysql prompt in terminal window t1, execute RESTART to restart the MySQL server.

Enter the following statement at the mysql prompt in t1 and receive the results shown:

```
mysql> RESTART;
Query OK, 0 rows affected (#.## sec)
```

7. Verify that the port and max_connections system variables are set to the default values (3306 and 151, respectively.)

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> SHOW VARIABLES LIKE 'port';
No connection. Trying to reconnect...
Connection id: 8
Current database: *** NONE ***

+-----+-----+
| Variable_name | Value |
+-----+-----+
| port          | 3306  |
+-----+-----+
1 row in set (#.## sec)

mysql> SHOW VARIABLES LIKE 'max_connections';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| max_connections | 151   |
+-----+-----+
1 row in set (#.## sec)
```

- As the server takes some time to restart, if you encounter an error connecting to the server, retry the SHOW statement.

8. Leave the mysql client session open for the next practice.

Practice 4-3: Changing Dynamic Settings

Overview

In this practice, you modify MySQL settings that do not require a server restart by changing the values of variables in the `mysql` client.

Duration

This practice should take you approximately five minutes to complete.

Tasks

1. At the `mysql` prompt you left open in the preceding practice, attempt to use a `SET GLOBAL` command to change the port number to 3303. What happens?
2. Use `SET GLOBAL` to change the value of the `max_connections` system variable to 1. What happens?
3. Display the value of the `autocommit` variable by using a `SHOW VARIABLES LIKE` command.
4. Change the value of the global `autocommit` variable to `OFF`.
5. Repeat step 3. Explain your results.
6. Execute `RESTART` at the `mysql` prompt to restore the `max_connections` system variable to its default value.
7. Leave the `mysql` client session open for the next practice.

Solution 4-3: Changing Dynamic Settings

Solution Steps

- At the mysql prompt you left open in the preceding practice, attempt to use a SET GLOBAL command to change the port number to 3303. What happens?

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> SET GLOBAL port=3303;
ERROR 1238 (HY000): Variable 'port' is a read only variable
```

- The mysql client displays an error because the port option is not dynamic. You can change it at the command line or in an option file, but you cannot change it in a running MySQL server.

- Use SET GLOBAL to change the value of the max_connections system variable to 1. What happens?

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> SET GLOBAL max_connections=1;
Query OK, 0 rows affected (#.# sec)
```

- The statement succeeds because the max_connections variable is dynamic. You can change it at the command line or in an option file by referring to its option name max-connections, and you can change it in a running MySQL server by referring to its system variable name max_connections.

- Display the value of the autocommit variable by using a SHOW VARIABLES LIKE command.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> SHOW VARIABLES LIKE 'autocommit';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| autocommit    | ON    |
+-----+-----+
1 row in set (#.# sec)
```

- Change the value of the global autocommit variable to OFF.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> SET GLOBAL autocommit=OFF;
Query OK, 0 rows affected (#.# sec)
```

5. Repeat step 3. Explain your results.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> SHOW VARIABLES LIKE 'autocommit';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| autocommit     | ON    |
+-----+-----+
1 row in set (#.## sec)
```

- The autocommit variable has both global and session versions. The SHOW VARIABLES LIKE command displays the session version of variables that have both global and session versions unless you provide the GLOBAL or SESSION modifier.

6. Execute RESTART at the mysql prompt to restore the max_connections system variable to its default value.

Enter the following statements at the mysql prompt and receive the results shown:

```
mysql> RESTART;
Query OK, 0 rows affected (#.## sec)

mysql> SHOW GLOBAL VARIABLES LIKE 'max_connections';
ERROR 2006 (HY000): MySQL server has gone away
No connection. Trying to reconnect...
Connection id: 8
Current database: *** NONE ***

+-----+-----+
| Variable_name | Value |
+-----+-----+
| max_connections | 151 |
+-----+-----+
1 row in set (#.## sec)
```

- As the server takes some time to restart, if you encounter an error connecting to the server, retry the SHOW statement.

7. Leave the mysql client session open for the next practice.

Practice 4-4: Persisting Global Variables

Overview

In this practice, you use the `SET PERSIST` SQL statement to persist global variables across server restarts.

Duration

This practice should take you approximately 10 minutes to complete.

Tasks

1. At the `mysql` prompt that you left open at the end of the preceding practice, use a `SET PERSIST` statement to change the maximum number of connections to 99 and verify that the change has taken effect.
2. Execute the `RESTART` SQL statement to restart the MySQL server.
3. Verify that the change to the `max_connections` global system variable is still in effect.
4. Examine the contents of the `mysqld-auto.cnf` file in the data directory by executing the following command through `mysql` client system command:

```
mysql> \! cat /var/lib/mysql/mysqld-auto.cnf | python -m json.tool
```
5. Query the Performance Schema `variable_info` table for information about the `max_connections` system variable.
6. Execute `SET PERSIST` again to change the maximum number of connections to `DEFAULT` and verify that the `max_connections` system variable now has the default value of 151.
7. Query the Performance Schema `variable_info` table for information about the `max_connections` system variable.
8. Examine the contents of the `mysqld-auto.cnf` file in the data directory.
9. Execute a `RESET PERSIST` statement to reset the `max_connections` system variable to a nonpersisted state.
10. Examine the contents of the `mysqld-auto.cnf` file in the data directory.
11. Execute the `RESTART` SQL statement to restart the MySQL server.
12. Query the Performance Schema `variable_info` table for information about the `max_connections` system variable.
13. Leave the `mysql` client session open for the next practice.

Solution 4-4: Persisting Global Variables

Solution Steps

- At the mysql prompt that you left open at the end of the preceding practice, use a SET PERSIST statement to change the maximum number of connections to 99 and verify that the change has taken effect.

Enter the following statements at the mysql prompt and receive the results shown:

```
mysql> SET PERSIST max_connections=99;
Query OK, 0 rows affected (#.## sec)

mysql> SHOW GLOBAL VARIABLES LIKE 'max_connections';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| max_connections | 99 |
+-----+-----+
1 row in set (#.## sec)
```

- Execute the RESTART SQL statement to restart the MySQL server.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> RESTART;
Query OK, 0 rows affected (#.## sec)
```

- Verify that the change to the max_connections global system variable is still in effect.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> SHOW GLOBAL VARIABLES LIKE 'max_connections';
ERROR 2006 (HY000): MySQL server has gone away
No connection. Trying to reconnect...
Connection id: 8
Current database: *** NONE ***

+-----+-----+
| Variable_name | Value |
+-----+-----+
| max_connections | 99 |
+-----+-----+
1 row in set (#.## sec)
```

- As the server takes some time to restart, if you encounter an error connecting to the server, retry the SHOW statement.

- Examine the contents of the mysqld-auto.cnf file in the data directory by executing the following command through mysql client system command:

```
mysql> \! cat /var/lib/mysql/mysqld-auto.cnf | python -m json.tool
```

Enter the following command at the `mysql` prompt and receive the results shown:

```
mysql> \! cat /var/lib/mysql/mysqld-auto.cnf | python -m json.tool
{
    "Version": 1,
    "mysql_server": {
        "max_connections": {
            "Metadata": {
                "Host": "localhost",
                "Timestamp": "timestamp",
                "User": "root"
            },
            "Value": "99"
        }
    }
}
```

- The `mysqld-auto.cnf` file contains details of the change you made to the global `max_connections` system variable by executing the `SET PERSIST` statement.
5. Query the Performance Schema `variable_info` table for information about the `max_connections` system variable.

Enter the following statement at the `mysql` prompt and receive the results shown:

```
mysql> SELECT * FROM performance_schema.variables_info
-> WHERE variable_name LIKE 'max_connections'\G
***** 1. row *****
VARIABLE_NAME: max_connections
VARIABLE_SOURCE: PERSISTED
VARIABLE_PATH: /var/lib/mysql/mysqld-auto.cnf
MIN_VALUE: 1
MAX_VALUE: 100000
SET_TIME: date-and-time
SET_USER: root
SET_HOST: localhost
1 row in set (#.## sec)
```

- Performance Schema also stores details of the persisted change. The MySQL server reads the `max_connections` system variable's value from the `mysqld-auto.cnf` file when the server starts.
6. Execute `SET PERSIST` again to change the maximum number of connections to `DEFAULT` and verify that the `max_connections` system variable now has the default value of 151.

Enter the following statements at the `mysql` prompt and receive the results shown:

```
mysql> SET PERSIST max_connections=DEFAULT;
Query OK, 0 rows affected (#.## sec)
mysql> SHOW GLOBAL VARIABLES LIKE 'max_connections';
```

```
+-----+-----+
| Variable_name | Value |
+-----+-----+
| max_connections | 151 |
+-----+-----+
1 row in set (#.# sec)
```

7. Query the Performance Schema `variable_info` table for information about the `max_connections` system variable.

Enter the following statement at the `mysql` prompt and receive the results shown:

```
mysql> SELECT * FROM performance_schema.variables_info
-> WHERE variable_name LIKE 'max_connections'\G
***** 1. row *****
VARIABLE_NAME: max_connections
VARIABLE_SOURCE: DYNAMIC
VARIABLE_PATH:
  MIN_VALUE: 1
  MAX_VALUE: 100000
  SET_TIME: 2018-06-06 13:52:27.578299
  SET_USER: root
  SET_HOST: localhost
1 row in set (#.# sec)
```

- The `max_connections` variable is now `DYNAMIC` instead of `PERSISTED`, because it was changed at runtime rather than from the startup files at startup.
 - After the MySQL server restarts, it will show `PERSISTED`.
8. Examine the contents of the `mysqld-auto.cnf` file in the data directory.

Enter the following command at the `mysql` prompt and receive the results shown:

```
mysql> \! cat /var/lib/mysql/mysqld-auto.cnf | python -m json.tool
{
  "Version": 1,
  "mysql_server": {
    "max_connections": {
      "Metadata": {
        "Host": "localhost",
        "Timestamp": 1528293147578250,
        "User": "root"
      },
      "Value": "151"
    }
  }
}
```

- The `max_connections` entry in the `mysqld-auto.cnf` file now shows that the value has changed to the default max number of connections: 151.

9. Execute a `RESET PERSIST` statement to reset the `max_connections` system variable to a nonpersisted state.

Enter the following statement at the `mysql` prompt and receive the results shown:

```
mysql> RESET PERSIST max_connections;
Query OK, 0 rows affected (#.## sec)
```

10. Examine the contents of the `mysqld-auto.cnf` file in the data directory.

Enter the following command at the `mysql` prompt and receive the results shown:

```
mysql> \! cat /var/lib/mysql/mysqld-auto.cnf | python -m json.tool
{
    "Version": 1,
    "mysql_server": {}
}
```

- The `mysqld-auto.cnf` file no longer has any configuration for the `max_connections` system variable.

11. Execute the `RESTART` SQL statement to restart the MySQL server.

Enter the following statement at the `mysql` prompt and receive the results shown:

```
mysql> RESTART;
Query OK, 0 rows affected (#.## sec)
```

12. Query the Performance Schema `variable_info` table for information about the `max_connections` system variable.

Enter the following statement at the `mysql` prompt and receive the results shown:

```
mysql> SELECT * FROM performance_schema.variables_info
      -> WHERE variable_name LIKE 'max_connections' \G
***** 1. row *****
VARIABLE_NAME: max_connections
VARIABLE_SOURCE: COMPILED
VARIABLE_PATH:
  MIN_VALUE: 1
  MAX_VALUE: 100000
  SET_TIME: NULL
  SET_USER: NULL
  SET_HOST: NULL
1 row in set (0.00 sec)
```

- The `max_connections` variable is now `COMPILED`, because it is not configured in any startup file.
- As the server takes some time to restart, if you encounter an error connecting to the server, retry the `SELECT` statement.

13. Leave the `mysql` client session open for the next practice.

Practice 4-5: Configuring the Client

Overview

In this practice, you configure the `mysql` client by creating a user-specific configuration file and by creating a login path containing encrypted credentials.

Duration

This practice should take you approximately 10 minutes to complete.

Tasks

1. Execute the `STATUS` command at the `mysql` client and use its output to determine the method the client uses to connect to the MySQL server.
2. Exit the `mysql` client.
3. Using a text editor, create the file `~/.my.cnf` with the following contents:

```
[mysql]
prompt='U[\d]> '
protocol=tcp
user=root
password=oracle
```
4. Launch the `mysql` client with no command-line arguments.
5. Change the default database to `mysql`.
6. Execute the `STATUS` command again to see how the client connects to the server.
7. Exit the `mysql` client.
8. The password in the configuration file is in plain text, which is not a secure configuration. Delete the `user` and `password` options from your user's `~/.my.cnf` configuration file.
9. Use the `mysql_config_editor` program to create a login path named `mysql` that specifies the `user` and `password` values that you deleted from the plain text configuration file in the preceding step.
10. Launch the `mysql` client with no command-line arguments.
11. Exit the `mysql` client.
12. Delete the `~/.my.cnf` and `~/.mylogin.cnf` files.
13. Leave the Linux terminal window open for the next practice.

Solution 4-5: Configuring the Client

Solution Steps

1. Execute the STATUS command at the mysql client and use its output to determine the method the client uses to connect to the MySQL server.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> STATUS
-----
mysql Ver 8.0.18 for Linux on x86_64 (MySQL Enterprise Server - Commercial)

Connection id: 8
Current database:
Current user: root@localhost
SSL: Not in use
Current pager: stdout
Using outfile:
Using delimiter: ;
Server version: 8.0.18-commercial MySQL Enterprise
Server - Commercial
Protocol version: 10
Connection: Localhost via UNIX socket
Server characterset: utf8mb4
Db characterset: utf8mb4
Client characterset: utf8mb4
Conn. characterset: utf8mb4
UNIX socket: /var/lib/mysql/mysql.sock
...
```

- Note that the connection uses a UNIX socket file at the following location:
`/var/lib/mysql/mysql.sock`.

2. Exit the mysql client.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> EXIT
Bye
```

3. Using a text editor, create the file `~/.my.cnf` with the following contents:

```
[mysql]
prompt='\U[\d]> '
protocol=tcp
user=root
password=oracle
```

- When you are logged in as `root` in Linux, the tilde (~) refers to the `/root` directory. If you are logged in as a different user, the tilde refers to that user's `$HOME` directory.
4. Launch the `mysql` client with no command-line arguments.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
...
root@localhost[(none)]>
```

Notes:

- You did not provide a password at the command line. The `mysql` client used the password that you placed in the option file.
- The `mysql>` prompt that previously appeared in the `mysql` client has been replaced with a prompt that contains the username, the host from which you connected, and the default database. As you have not yet selected a database, the prompt displays `(none)`.

5. Change the default database to `mysql`.

Enter the following statement at the modified `mysql` prompt and receive the results shown:

```
root@localhost[(none)]> USE mysql
Reading table information for completion of table and column
names
You can turn off this feature to get a quicker startup with -A

Database changed
root@localhost[mysql]>
```

- The prompt has changed to reflect the changed default database.

6. Execute the `STATUS` command again to see how the client connects to the server.

```
root@localhost[mysql]> STATUS
-----
mysql Ver 8.0.18 for x86_64 (MySQL Enterprise Server -
Commercial)

...
Connection:          localhost via TCP/IP
...
TCP port:            3306
```

- Note that the connection uses TCP/IP and the default port number 3306. You specified `protocol=tcp` in the option file but not the port option, so the `mysql` client connects using the default port.

7. Exit the `mysql` client.

Enter the following statement at the modified `mysql` prompt and receive the results shown:

```
root@localhost[mysql]> EXIT
Bye
```

8. The password in the configuration file is in plain text, which is not a secure configuration. Delete the `user` and `password` options from your user's `~/.my.cnf` configuration file. Edit the `~/.my.cnf` file so that it contains the following text:

```
[mysql]
prompt='\U[\d]> '
protocol=tcp
```

9. Use the `mysql_config_editor` program to create a login path named `mysql` that specifies the `user` and `password` values that you deleted from the plain text configuration file in the preceding step.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# mysql_config_editor set --login-path=mysql --user=root --password
Enter password: oracle
```

- This command creates an encrypted file called `~/.mylogin.cnf` in your home directory that contains the user, port, and password settings in an option group called `[mysql]`. The `mysql` client reads this option group by default, so you do not need to specify the `mysql` login path when you launch the client.

10. Launch the `mysql` client with no command-line arguments.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# mysql
Welcome to the MySQL monitor. Commands end with ; or \g.
...
root@localhost[(none)]>
```

- The client successfully connects using the encrypted credentials in the login path.

11. Exit the `mysql` client.

Enter the following statement at the modified `mysql` prompt and receive the results shown:

```
root@localhost[mysql]> EXIT
Bye
```

12. Delete the `~/.my.cnf` and `~/.mylogin.cnf` files.

Enter the following commands at the Linux terminal prompt and receive the results shown:

```
# rm ~/.my.cnf
rm: remove regular file '/root/.my.cnf'? y
# rm ~/.mylogin.cnf
rm: remove regular file '/root/.mylogin.cnf'? y
```

13. Leave the Linux terminal window open for the next practice.

Practice 4-6: Running Multiple MySQL Servers on the Same Host with `systemd`

Overview

In this practice, you configure and run multiple MySQL server instances using `systemd`.

Duration

This practice should take you approximately 20 minutes to complete.

Tasks

1. Use `mysqladmin` to shut down the running MySQL server.
2. Create a new empty directory in the root of your file system named `mysql`. Grant its ownership to the `mysql` user and group.
3. Using `mysqld`, initialize data directories in `/mysql/data1`, `/mysql/data2`, `/mysql/data3`, and `/mysql/data4`, respectively. Use the `--initialize-insecure` option (to avoid having to enter randomly generated temporary passwords for each instance) and the `--no-defaults` option (to prevent `mysqld` from using the default settings in `/etc/my.cnf`).
4. Examine the `/labs/multi.cnf` file that contains the following configuration entries. You will use this configuration file to specify the settings for four new server instances called `server1`, `server2`, `server3`, and `server4`.
5. Examine the `/labs/service/mysql-8.0/mysqld@.service` file. This is a `systemd` service unit configuration file. It is similar to the one you used in the “Configuring the MySQL Service” activity for the lesson titled “Installing MySQL.” The main difference is that it enables you to pass the name of the server as a parameter to `systemctl` (represented by `%I` in the file) and reads the appropriate configuration entry from `/labs/multi.cnf`.
6. Copy the `/labs/service/mysql-8.0/mysqld@.service` file to the `/usr/lib/systemd/system` directory.
7. Execute `systemctl daemon-reload` to reload all the unit files and re-create the dependency tree.
8. Start each new server instance in turn, using an appropriate `systemctl` command.
9. Use the Linux `ps` command to list the running `mysqld` processes.
10. Connect to the `server3` instance using the appropriate socket file, as specified in `/labs/multi.cnf`.
11. Enter a SQL command to display the port number that the MySQL instance is running on.
12. Exit the `mysql` client connected to `server3`.
13. Use `systemctl` to stop `server1`.
14. Enter the following `systemctl` command with a wildcard for the server number (*) to view the status of servers 1-4:

```
systemctl status mysqld@server*
```

15. Use `systemctl` to view the status of `server1`.
16. Enter `systemctl` commands with a wildcard for the server number (*) to stop all the servers and verify that they are stopped.
17. Use the Linux `ps` command to list the running `mysqld` processes.
18. Start the single instance MySQL server using `systemctl` command.
19. Close all open Linux terminal windows.

Solution 4-6: Running Multiple MySQL Servers on the Same Host with systemd

Solution Steps

1. Use mysqladmin to shut down the running MySQL server.

Enter the following statement at the mysql prompt and receive the results shown:

```
# mysqladmin -uroot -p shutdown
Enter password: oracle
```

2. Create a new empty directory in the root of your file system named mysql. Grant its ownership to the mysql user and group.

Enter the following commands at the Linux terminal prompt and receive the results shown:

```
# mkdir /mysql
# chown mysql:mysql /mysql
```

3. Using mysqld, initialize data directories in /mysql/data1, /mysql/data2, /mysql/data3, and /mysql/data4, respectively. Use the --initialize-insecure option (to avoid having to enter randomly generated temporary passwords for each instance) and the --no-defaults option (to prevent mysqld from using the default settings in /etc/my.cnf).

Enter the following commands at the Linux terminal prompt and receive the results shown:

```
# mysqld --no-defaults --initialize-insecure --user=mysql \
> --datadir=/mysql/data1
...
date-and-time 0 [System] [MY-013170] [Server] /opt/mysql-commercial-
version/bin/mysqld (mysqld version-commercial) initializing of server
has completed
# mysqld --no-defaults --initialize-insecure --user=mysql \
> --datadir=/mysql/data2
...
date-and-time 0 [System] [MY-013170] [Server] /opt/mysql-commercial-
version/bin/mysqld (mysqld version-commercial) initializing of server
has completed
# mysqld --no-defaults --initialize-insecure --user=mysql \
> --datadir=/mysql/data3
...
date-and-time 0 [System] [MY-013170] [Server] /opt/mysql-commercial-
version/bin/mysqld (mysqld version-commercial) initializing of server
has completed
# mysqld --no-defaults --initialize-insecure --user=mysql \
> --datadir=/mysql/data4
...
date-and-time 0 [System] [MY-013170] [Server] /opt/mysql-commercial-
version/bin/mysqld (mysqld version-commercial) initializing of server
has completed
```

4. Examine the `/labs/multi.cnf` file that contains the following configuration entries. You will use this configuration file to specify the settings for four new server instances called `server1`, `server2`, `server3`, and `server4`.

The contents of the `/labs/multi.cnf` file are as follows:

```
[mysqld@server1]
user=mysql
socket=/mysql/server1.sock
port=3311
datadir=/mysql/data1
log-error=/mysql/server1.err

[mysqld@server2]
user=mysql
socket=/mysql/server2.sock
port=3312
datadir=/mysql/data2
log-error=/mysql/server2.err

[mysqld@server3]
user=mysql
socket=/mysql/server3.sock
port=3313
datadir=/mysql/data3
log-error=/mysql/server3.err

[mysqld@server4]
user=mysql
socket=/mysql-server3.sock
port=3314
datadir=/mysql/data4
log-error=/mysql/server4.err
```

5. Examine the `/labs/service/mysql-8.0/mysqld@.service` file. This is a `systemd` service unit configuration file. It is similar to the one you used in the “Configuring the MySQL Service” activity for the lesson titled “Installing MySQL.” The main difference is that it enables you to pass the name of the server as a parameter to `systemctl` (represented by `%I` in the file) and reads the appropriate configuration entry from `/labs/multi.cnf`.

The contents of the `/labs/service/mysql-8.0/myqsl@.service` file are as follows:

```
[Unit]
Description=MySQL Multi Server for instance %i
Documentation=man:mysqld(8)
Documentation=http://dev.mysql.com/doc/refman/en/using-
systemd.html
After=network.target
After=syslog.target
```

```
[Install]
WantedBy=multi-user.target

[Service]
User=mysql
Group=mysql
PIDFile=mysql/server%I.pid
Type=notify

# Disable service start and stop timeout logic of systemd for
mysqld service.
TimeoutSec=0

# Start main service
ExecStart=/usr/local/mysql/bin/mysqld --defaults-
file=~/labs/multi.cnf --defaults-group-suffix=@%I $MYSQLD_OPTS

# Use this to switch malloc implementation
EnvironmentFile=-/etc/sysconfig/mysql

# Sets open_files_limit
LimitNOFILE = 10000

Restart=on-failure

RestartPreventExitStatus=1

# Set environment variable MYSQLD_PARENT_PID. This is required
for restart.
Environment=MYSQLD_PARENT_PID=1

PrivateTmp=false
```

6. Copy the `/labs/service/mysql-8.0/mysqld@.service` file to the `/usr/lib/systemd/system` directory.

Enter the following command at the Linux terminal prompt:

```
# cp /labs/service/mysql-8.0/mysqld@.service \
> /usr/lib/systemd/system/mysqld@.service
```

7. Execute `systemctl daemon-reload` to reload all the unit files and re-create the dependency tree.

Enter the following command at the Linux terminal prompt:

```
# systemctl daemon-reload
```

8. Start each new server instance in turn, using an appropriate `systemctl` command.

Enter the following commands at the Linux terminal prompt and receive the results shown:

```
# systemctl start mysqld@server1
# systemctl start mysqld@server2
# systemctl start mysqld@server3
# systemctl start mysqld@server4
```

9. Use the Linux `ps` command to list the running `mysqld` processes.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# ps aux | grep mysqld
mysql      PID ... /usr/local/mysql/bin/mysqld --defaults-
file=/labs/multi.cnf --defaults-group-suffix=@server1
mysql      PID ... /usr/local/mysql/bin/mysqld --defaults-
file=/labs/multi.cnf --defaults-group-suffix=@server2
mysql      PID ... /usr/local/mysql/bin/mysqld --defaults-
file=/labs/multi.cnf --defaults-group-suffix=@server3
mysql      PID ... /usr/local/mysql/bin/mysqld --defaults-
file=/labs/multi.cnf --defaults-group-suffix=@server4
root      PID ... grep --color=auto mysqld
```

- The output of the `ps` command shows five running servers: the existing instance that uses the `/etc/my.cnf` file and servers 1-4 that use the settings in the `/labs/multi.cnf` file.

10. Connect to the `server3` instance using the appropriate socket file, as specified in `/labs/multi.cnf`.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# mysql -uroot -S/mysql/server3.sock
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 9
...
mysql>
```

11. Enter a SQL command to display the port number that the MySQL instance is running on.

Enter the following statement at the `mysql` prompt and receive the results shown:

```
mysql> SELECT @@port;
+-----+
| @@port |
+-----+
| 3313 |
+-----+
1 row in set (0.00 sec)
```

12. Exit the mysql client connected to server3.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> EXIT  
Bye  
#
```

13. Use systemctl to stop server1.

Enter the following command at the Linux terminal prompt:

```
# systemctl stop mysqld@server1
```

14. Enter the following systemctl command with a wildcard for the server number (*) to view the status of servers 1-4:

```
systemctl status mysqld@server*
```

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# systemctl status mysqld@server*  
● mysqld@server3.service - MySQL Multi Server for instance server3  
    Loaded: loaded (/usr/lib/systemd/system/mysqld@.service; enabled; vendor preset: disabled)  
    Active: active (running) since date-and-time  
          ...  
    Main PID: PID (mysqld)  
          ...  
  
● mysqld@server4.service - MySQL Multi Server for instance server4  
    Loaded: loaded (/usr/lib/systemd/system/mysqld@.service; enabled; vendor preset: disabled)  
    Active: active (running) since date-and-time  
          ...  
    Main PID: PID (mysqld)  
          ...  
  
● mysqld@server2.service - MySQL Multi Server for instance server2  
    Loaded: loaded (/usr/lib/systemd/system/mysqld@.service; enabled; vendor preset: disabled)  
    Active: active (running) since date-and-time  
          ...  
    Main PID: PID (mysqld)  
          ...
```

- Servers 2, 3, and 4 are running. Server 1 does not appear in the list.

15. Use `systemctl` to view the status of `server1`.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# systemctl status mysqld@server1
● mysqld@server1.service - MySQL Multi Server for instance
server1
    Loaded: loaded (/usr/lib/systemd/system/mysql@.service;
enabled; vendor preset: disabled)
    Active: inactive (dead)
      Docs: man:mysqld(8)
             http://dev.mysql.com/doc/refman/en/using-systemd.html

...
date-and-time hostname systemd[1]: Stopping MySQL Multi Server
for instance server1...
date-and-time hostname systemd[1]: Stopped MySQL Multi Server
for instance server1.
```

- The output shows that `server1` is inactive (stopped).

16. Enter `systemctl` commands with a wildcard for the server number (*) to stop all the servers and verify that they are stopped.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# systemctl stop mysqld@server*
# systemctl status mysqld@server*
```

17. Use the Linux `ps` command to list the running `mysqld` processes.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# ps aux | grep mysqld
root     ... grep --color=auto mysqld
```

- The only running instance is the main instance that reads its configuration from `/etc/my.cnf`.

18. Start the single instance MySQL server using `systemctl` command.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# systemctl start mysqld
```

19. Close all open Linux terminal windows.

Practices for Lesson 5: Monitoring MySQL

Practices for Lesson 5: Overview

Overview

In these practices, you monitor a running MySQL instance by viewing the slow query log, `mysqladmin status` output, and by installing and using MySQL Enterprise Monitor.

Assumptions

- The `employees` database script and data files are in the `/stage/databases/employees_db` directory.
- The `mysqlmonitor-version-linux-x86_64-installer.bin` file is in the `/stage/MySQL-EM` directory. This is the installation file of MySQL Enterprise Monitor.
- The `slap-test-updates.sh` file is in the `/labs` directory.

Practice 5-1: Configuring the Slow Query Log

Overview

In this practice, you use the sample `employees` database to generate slow queries that you monitor by using the slow query log.

Duration

This practice should take you approximately 15 minutes to complete.

Tasks

1. Import the sample `employees` database by running the `employees.sql` script from the `/stage/databases/employees_db` directory. The operation takes several minutes to complete.
2. Start an interactive `mysql` client session.
3. Configure the global `slow_query_log_file` system variable to use the file name `slow-query.log`.
4. Configure the `long_query_time` session variable to log queries that take longer than half a second to execute.
5. Enable the slow query log.
6. Execute the following query against the `salaries` table in the `employees` database and record how long it takes to execute:

```
SELECT emp_no, salary FROM salaries  
WHERE from_date BETWEEN '1986-01-01' AND '1986-01-07'  
ORDER BY from_date, salary;
```

7. Examine the contents of the slow query log file by executing the following command through `mysql` client system command:

```
mysql> \! cat /var/lib/mysql/slow-query.log
```

8. Query the contents of the `mysql.slow_log` table.
9. Configure the slow query log so that it writes to the `mysql.slow_log` table instead of to a file.
10. Execute the query in step 6 again.
11. Query the contents of the `mysql.slow_log` table again.
12. Leave the `mysql` client session open for the next practice.

Solution 5-1: Configuring the Slow Query Log

Solution Steps

1. Import the sample employees database by running the employees.sql script from the /stage/databases/employees_db directory. The operation takes several minutes to complete.

Enter the following commands at a new Linux terminal prompt and receive the results shown:

```
# cd /stage/databases/employees_db
# mysql -uroot -p < employees.sql
Enter password: oracle
INFO
CREATING DATABASE STRUCTURE
INFO
LOADING departments
INFO
LOADING employees
INFO
LOADING dept_emp
INFO
LOADING dept_manager
INFO
LOADING titles
INFO
LOADING salaries
```

2. Start an interactive mysql client session.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# mysql -uroot -p
Enter password: oracle
...
```

3. Configure the global slow_query_log_file system variable to use the file name slow-query.log.

Enter the following statements at the mysql prompt and receive the results shown:

```
mysql> SET GLOBAL slow_query_log_file='slow-query.log';
Query OK, 0 rows affected (0.00 sec)
```

- The slow-query.log file will be created in the data directory when you enable it in a later step.

4. Configure the `long_query_time` session variable to log queries that take longer than half a second to execute.

Enter the following statement at the `mysql` prompt and receive the results shown:

```
mysql> SET SESSION long_query_time=0.5;
Query OK, 0 rows affected (#.## sec)
```

- The `long_query_time` variable is scoped both at the global level and the session level. If you set only the global variable, the session variable does not change until you reconnect the session.

5. Enable the slow query log.

Enter the following statement at the `mysql` prompt and receive the results shown:

```
mysql> SET GLOBAL slow_query_log=ON;
Query OK, 0 rows affected (#.## sec)
```

6. Execute the following query against the `salaries` table in the `employees` database and record how long it takes to execute:

```
SELECT emp_no, salary FROM salaries
WHERE from_date BETWEEN '1986-01-01' AND '1986-01-07'
ORDER BY from_date, salary;
```

Enter the following statements at the `mysql` prompt and receive the results shown:

```
mysql> USE employees
...
Database changed

mysql> SELECT emp_no, salary FROM salaries
-> WHERE from_date BETWEEN '1986-01-01' AND '1986-01-07'
-> ORDER BY from_date, salary;
+-----+
| emp_no | salary |
+-----+
| 82240 | 40000 |
| 241962 | 40000 |
...
| 97466 | 91912 |
+-----+
354 rows in set (0.98 sec)
```

- In the preceding output, the statement takes 0.98 seconds. It might take a different amount of time on your system. If it takes less time than the value of `long_query_time` that you set in step 4, adjust the value of `long_query_time` and execute the query again before proceeding with the next step.

7. Examine the contents of the slow query log file by executing the following command through mysql client system command:

```
mysql> \! cat /var/lib/mysql/slow-query.log
```

Enter the following command at the mysql prompt and receive the results shown:

```
mysql> \! cat /var/lib/mysql/slow-query.log
/usr/local/mysql/bin/mysqld, Version: version-commercial (MySQL
Enterprise Server - Commercial). started with:
Tcp port: 3306 Unix socket: /var/lib/mysql/mysql.sock
Time Id Command Argument
# Time: date-and-time
# User@Host: root[root] @ localhost [] Id: 10
# Query_time: 0.984945 Lock_time: 0.000148 Rows_sent: 354
Rows_examined: 2844401
use employees;
SET timestamp=1529659908;
SELECT emp_no, salary FROM salaries
WHERE from_date BETWEEN '1986-01-01' AND '1986-01-07'
ORDER BY from_date, salary;
```

The slow log records the following key metrics:

- Query time: 0.984945 seconds
- Lock time: 0.000148 seconds
- Rows sent: 354
- Rows examined: 2,844,401

Your query time and lock time are based on conditions on your local machine and differ from this output accordingly.

8. Query the contents of the mysql.slow_log table.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> SELECT * FROM mysql.slow_log;
Empty set (#.## sec)
```

- The slow query log table is empty because the slow query log outputs to a file by default.

9. Configure the slow query log so that it writes to the mysql.slow_log table instead of to a file.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> SET GLOBAL log_output='TABLE';
Query OK, 0 rows affected (#.## sec)
```

10. Execute the query in step 6 again.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> SELECT emp_no, salary FROM salaries
-> WHERE from_date BETWEEN '1986-01-01' AND '1986-01-07'
-> ORDER BY from_date, salary;
+-----+-----+
| emp_no | salary |
+-----+-----+
| 82240 | 40000 |
| 241962 | 40000 |
...
| 97466 | 91912 |
+-----+
354 rows in set (0.71 sec)
```

11. Query the contents of the mysql.slow_log table again.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> SELECT * FROM mysql.slow_log\G
***** 1. row *****
start_time: date-and-time
user_host: root[root] @ localhost []
query_time: 00:00:00.903977
lock_time: 00:00:00.000134
rows_sent: 354
rows_examined: 2844401
db: employees
last_insert_id: 0
insert_id: 0
server_id: 1
sql_text: SELECT emp_no, salary FROM salaries WHERE
from_date BETWEEN '1986-01-01' AND '1986-01-07' ORDER BY
from_date, salary
thread_id: 10
1 row in set (#.# sec)
```

- The mysql.slow_log table contains similar information to the slow query log file that you examined in step 7.

12. Leave the mysql client session open for the next practice.

Practice 5-2: Using Performance Schema

Overview

In this practice, you investigate the configuration of the Performance Schema and use it and the associated `sys` schema to diagnose potential performance problems.

Duration

This practice should take you approximately 20 minutes to complete.

Tasks

1. Enter the following statement at the `mysql` prompt to inspect the value of the `performance_schema` system variables and verify that the Performance Schema itself is enabled:

```
SHOW VARIABLES LIKE 'performance_schema%';
```
2. Execute a `SHOW ENGINE STATUS` command on the `PERFORMANCE_SCHEMA` storage engine to display the memory usage of the Performance Schema. The last line of the output contains this information.
3. Change the current database to `performance_schema` and execute a `SHOW` statement that lists all the Performance Schema tables with names that start with “setup”.
4. The `setup_actors` table configures which foreground threads to monitor. Foreground threads are those associated with client connections. Display the contents of the `setup_actors` table. Which foreground threads is Performance Schema monitoring?
5. The `setup_objects` table controls whether the Performance Schema monitors specific database objects. Display the contents of the `setup_objects` table. Which database objects is the Performance Schema monitoring?
6. The `setup_instruments` table contains one row for every instrumentation point in the source code and defines all the events that Performance Schema is able to collect. Display the first five rows of the `setup_instruments` table in vertical format.
7. The `setup_threads` table specifies which threads are instrumented and describes their properties. Display the contents of the `setup_threads` table.
8. The `setup_consumers` table lists the consumers of the events from the configured instruments. Display the contents of the `setup_consumers` table. Which consumers are enabled?
9. Read the help documentation on the `performance_schema.table_io_waits_summary_by_table` table:
<http://dev.mysql.com/doc/mysql/en/table-waits-summary-tables.html#table-io-waits-summary-by-table-table>
Note that this table aggregates all table I/O wait events generated by the `wait/io/table/sql/handler` instrument.
10. Verify that the `wait/io/table/sql/handler` instrument is enabled.

11. Truncate the contents of the `table_io_waits_summary_by_table` table. This resets any counter values and timing information to zero.
12. Execute the following SELECT statement on the `employees` table in the `employees` schema:

```
SELECT * FROM employees.employees;
```

13. Display the contents of the `OBJECT_SCHEMA`, `OBJECT_NAME`, `COUNT_STAR`, and `SUM_TIMER_WAIT` columns of the Performance Schema `table_io_waits_summary_by_table` table where the `OBJECT_SCHEMA` and `OBJECT_NAME` are the `employees` database and `employees` table name, respectively.

There are many tables in the Performance Schema, and they contain a lot of potentially valuable information for the DBA. However, it can be difficult to remember which tables you require and how to join the tables to retrieve only the information you want. For that reason, it is often easier to use the `sys` schema. The `sys` schema contains views and stored routines that make it easier to interact with the Performance Schema for common operations. For example, to identify slow queries, there is the `statements_with_runtimes_in_95th_percentile` view.

14. Execute the following query:

```
SELECT SLEEP(5);
```

15. Query the `statements_with_runtimes_in_95th_percentile` in the `sys` schema and locate the statement you executed in the preceding step:

16. Exit the mysql session.

Solution 5-2: Using Performance Schema

Solution Steps

- Enter the following statement at the mysql prompt to inspect the value of the performance_schema system variables and verify that the Performance Schema itself is enabled:

```
mysql> SHOW VARIABLES LIKE 'performance_schema%';
```

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> SHOW VARIABLES LIKE 'performance_schema%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| performance_schema | ON   |
| performance_schema_accounts_size | -1   |
| performance_schema_digests_size | 10000 |
...
44 rows in set (#.## sec)
```

- The performance_schema variable is set to ON because the Performance Schema is enabled by default.
- The other variables define the size of the various tables within the Performance Schema. Because all these tables are in memory, changing these values affects memory usage.

- Execute a SHOW ENGINE STATUS command on the PERFORMANCE_SCHEMA storage engine to display the memory usage of the Performance Schema. The last line of the output contains this information.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> SHOW ENGINE PERFORMANCE_SCHEMA STATUS;
+-----+-----+-----+
| Type      | Name          | Status    |
+-----+-----+-----+
...
| performance_schema | performance_schema.memory | 195333792 |
+-----+-----+-----+
243 rows in set (#.## sec)
```

Note: In earlier versions of MySQL, it was often necessary to tune the variables that specified the sizes of the Performance Schema tables. In later versions, many of these variables are autoscaled to use only as much memory as the table requires. For that reason, your output might differ from that shown.

3. Change the current database to `performance_schema` and execute a `SHOW` statement that lists all the Performance Schema tables with names that start with “setup”.

Enter the following statements at the `mysql` prompt and receive the results shown:

```
mysql> USE performance_schema
Reading table information
...
Database changed
mysql> SHOW TABLES LIKE 'setup%';
+-----+
| Tables_in_performance_schema (setup%) |
+-----+
| setup_actors                         |
| setup_consumers                      |
| setup_instruments                    |
| setup_objects                        |
| setup_threads                        |
+-----+
5 rows in set (#.# sec)
```

4. The `setup_actors` table configures which foreground threads to monitor. Foreground threads are those associated with client connections. Display the contents of the `setup_actors` table. Which foreground threads is Performance Schema monitoring?

Enter the following statement at the `mysql` prompt and receive the results shown:

```
mysql> SELECT * FROM setup_actors;
+-----+-----+-----+-----+
| HOST | USER | ROLE | ENABLED | HISTORY |
+-----+-----+-----+-----+
| %    | %    | %    | YES   | YES   |
+-----+-----+-----+-----+
1 row in set (#.# sec)
```

Answer: All foreground threads are monitored by default.

Note: The `ROLE` column is currently unused. The `HISTORY` column specifies whether to log historical thread information for foreground threads matched by the row.

5. The `setup_objects` table controls whether the Performance Schema monitors specific database objects. Display the contents of the `setup_objects` table. Which database objects is the Performance Schema monitoring?

Enter the following statement at the `mysql` prompt and receive the results shown:

```
mysql> SELECT * FROM setup_objects;
+-----+-----+-----+-----+
| OBJECT_TYPE | OBJECT_SCHEMA      | OBJECT_NAME | ENABLED | TIMED |
+-----+-----+-----+-----+
| EVENT       | mysql                 | %           | NO     | NO    |
+-----+-----+-----+-----+
```

EVENT	performance_schema	%	NO	NO	
EVENT	information_schema	%	NO	NO	
EVENT	%	%	YES	YES	
FUNCTION	mysql	%	NO	NO	
FUNCTION	performance_schema	%	NO	NO	
FUNCTION	information_schema	%	NO	NO	
FUNCTION	%	%	YES	YES	
PROCEDURE	mysql	%	NO	NO	
PROCEDURE	performance_schema	%	NO	NO	
PROCEDURE	information_schema	%	NO	NO	
PROCEDURE	%	%	YES	YES	
TABLE	mysql	%	NO	NO	
TABLE	performance_schema	%	NO	NO	
TABLE	information_schema	%	NO	NO	
TABLE	%	%	YES	YES	
TRIGGER	mysql	%	NO	NO	
TRIGGER	performance_schema	%	NO	NO	
TRIGGER	information_schema	%	NO	NO	
TRIGGER	%	%	YES	YES	

20 rows in set (#.## sec)

Answer: By default, Performance Schema monitors all events, functions, procedures, tables, and triggers, except for those in the `mysql`, `performance_schema`, and `information_schema` databases.

Notes

- The `ENABLED` column specifies whether Performance Schema monitors the database object to which the row refers. If the value of the `TIMED` column is also set to `YES`, Performance Schema not only counts the number of events but also provides timing information.
 - When a database object fits into several categories, Performance Schema uses the most specific match.
6. The `setup_instruments` table contains one row for every instrumentation point in the source code and defines all the events that Performance Schema is able to collect. Display the first five rows of the `setup_instruments` table in vertical format.

Enter the following statement at the `mysql` prompt and receive the results shown:

```
mysql> SELECT * FROM setup_instruments LIMIT 5\G
***** 1. row *****
      NAME: wait/synch/mutex/pfs/LOCK_pfs_share_list
      ENABLED: NO
      TIMED: NO
      PROPERTIES: singleton
      VOLATILITY: 1
      DOCUMENTATION: Components can provide their own performance_schema
                      tables. This lock protects the list of such tables definitions.
```

```
***** 2. row *****
    NAME: wait/synch/mutex/sql/TC_LOG_MMAP::LOCK_tc
    ENABLED: NO
    TIMED: NO
    PROPERTIES:
    VOLATILITY: 0
DOCUMENTATION: NULL
***** 3. row *****
    NAME: wait/synch/mutex/sql/MYSQL_BIN_LOG::LOCK_commit
    ENABLED: NO
    TIMED: NO
    PROPERTIES:
    VOLATILITY: 0
DOCUMENTATION: NULL
***** 4. row *****
    NAME: wait/synch/mutex/sql/MYSQL_BIN_LOG::LOCK_commit_queue
    ENABLED: NO
    TIMED: NO
    PROPERTIES:
    VOLATILITY: 0
DOCUMENTATION: NULL
***** 5. row *****
    NAME: wait/synch/mutex/sql/MYSQL_BIN_LOG::LOCK_done
    ENABLED: NO
    TIMED: NO
    PROPERTIES:
    VOLATILITY: 0
DOCUMENTATION: NULL
5 rows in set (0.00 sec) (#.## sec)
```

Notes

- The name of each instrument represents components that form a hierarchy, with each component separated by a forward slash (/).
 - If **ENABLED** is set to YES, the instrument produces events that are counted. If **TIMED** is set to YES, the events are timed.
7. The `setup_threads` table specifies which threads are instrumented and describes their properties. Display the contents of the `setup_threads` table.

Enter the following statement at the `mysql` prompt and receive the results shown:

mysql> SELECT * FROM <u>setup_threads</u> ;			
	NAME	ENABLED	HISTORY
+	+	+	+
	thread/performance_schema/setup	YES	YES
+	+	+	...
	thread/sql/bootstrap	YES	YES
	thread/sql/manager	YES	YES
			...

```

| thread/sql/main           | YES    | YES    | ...
| thread/sql/one_connection | YES    | YES    | ...
| thread/sql/signal_handler| YES    | YES    | ...
...
48 rows in set (#.## secs)

```

Note: The `setup_threads` table contains a column called `PROPERTIES` that can contain one of the following values:

- **singleton:** The instrument has a single instance. For example, there is only one thread for the `thread/sql/main` instrument.
 - **user:** The instrument is directly related to user workload (as opposed to system workload). For example, threads such as `thread/sql/one_connection` executing a user session have the `user` property to differentiate them from system threads.
8. The `setup_consumers` table lists the consumers of the events from the configured instruments. Display the contents of the `setup_consumers` table. Which consumers are enabled?

Enter the following statement at the `mysql` prompt and receive the results shown:

```

mysql> SELECT * FROM setup_consumers;
+-----+-----+
| NAME      | ENABLED |
+-----+-----+
| events_stages_current | NO      |
| events_stages_history | NO      |
| events_stages_history_long | NO      |
| events_statements_current | YES     |
| events_statements_history | YES     |
| events_statements_history_long | NO      |
| events_transactions_current | YES     |
| events_transactions_history | YES     |
| events_transactions_history_long | NO      |
| events_waits_current | NO      |
| events_waits_history | NO      |
| events_waits_history_long | NO      |
| global_instrumentation | YES     |
| thread_instrumentation | YES     |
| statements_digest | YES     |
+-----+-----+
15 rows in set (#.## sec)

```

Answer: The `events_statements_current`, `events_statements_history`, `events_transactions_current`, `events_transactions_history`, `global_instrumentation`, `thread_instrumentation`, and `statements_digest` consumers are enabled by default.

9. Read the help documentation on the

`performance_schema.table_io_waits_summary_by_table`:

<http://dev.mysql.com/doc/mysql/en/table-waits-summary-tables.html#table-io-waits-summary-by-table-table>

Note that this table aggregates all table I/O wait events generated by the `wait/io/table/sql/handler` instrument.

10. Verify that the `wait/io/table/sql/handler` instrument is enabled.

Enter the following statement at the `mysql` prompt and receive the results shown:

```
mysql> SELECT * FROM setup_instruments
-> WHERE NAME LIKE 'wait/io/table%';
+-----+-----+-----+...
| NAME           | ENABLED | TIMED | ...
+-----+-----+-----+...
| wait/io/table/sql/handler | YES     | YES    | ...
+-----+-----+-----+...
1 row in set (#.# sec)
```

- The `/wait/io/table/sql/handler` instrument is enabled and also records timing information.

11. Truncate the contents of the `table_io_waits_summary_by_table` table. This resets any counter values and timing information to zero.

Enter the following statement at the `mysql` prompt and receive the results shown:

```
mysql> TRUNCATE TABLE table_io_waits_summary_by_table;
Query OK, 0 rows affected (#.# sec)
```

12. Execute the following SELECT statement on the `employees` table in the `employees` schema:

```
SELECT * FROM employees.employees;
```

It takes a minute or two to display the results due to the size of the table.

Enter the following statement at the `mysql` prompt and receive the results shown:

```
mysql> SELECT * FROM employees.employees;
+-----+-----+-----+-----+-----+
| emp_no | birth_date | first_name | last_name | gender | hire_date |
+-----+-----+-----+-----+-----+
| 10001 | 1953-09-02 | Georgi      | Facello    | M       | 1986-06-26 |
| 10002 | 1964-06-02 | Bezalel     | Simmel    | F       | 1985-11-21 |
| 10003 | 1959-12-03 | Parto      | Bamford   | M       | 1986-08-28 |
| 10004 | 1954-05-01 | Chirstian  | Koblick   | M       | 1986-12-01 |
...
+-----+-----+-----+-----+-----+
300024 rows in set (#.# sec)
```

13. Display the contents of the OBJECT_SCHEMA, OBJECT_NAME, COUNT_STAR, and SUM_TIMER_WAIT columns of the Performance Schema table_io_waits_summary_by_table where the OBJECT_SCHEMA and OBJECT_NAME are the employees database and employees table name, respectively.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> SELECT OBJECT_SCHEMA, OBJECT_NAME,
-> COUNT_STAR, SUM_TIMER_WAIT
-> FROM table_io_waits_summary_by_table
-> WHERE OBJECT_SCHEMA='employees'
-> AND OBJECT_NAME='employees';
+-----+-----+-----+
| OBJECT_SCHEMA | OBJECT_NAME | COUNT_STAR | SUM_TIMER_WAIT |
+-----+-----+-----+
| employees     | employees   |      300024 |    340411266972 |
+-----+-----+-----+
1 row in set (#.# sec)
```

- The table records the table I/O wait events incurred by the query of the employees table in the employees database, together with timing information.
- The SUM_TIMER_WAIT column for waits depends on local conditions in your computer. It uses CYCLE timing units, which relate to CPU speed, as shown in the performance_timers table:

```
mysql> SELECT * FROM performance_timers;
+-----+-----+-----+
| TIMER_NAME | TIMER_FREQUENCY | TIMER_RESOLUTION | TIMER_OVERHEAD |
+-----+-----+-----+
| CYCLE      | 2691223880 | 1 | 27 |
| NANOSECOND | 1000000000 | 1 | 237 |
| MICROSECOND | 1000000 | 1 | 288 |
| MILLISECOND | 957 | 1 | 246 |
+-----+-----+-----+
4 rows in set (#.# sec)
```

- The TIMER_FREQUENCY in the sample output is approximately 2,600,000,000, on a system with a 2.6 GHz processor.
- The wait timer uses CYCLE. The idle, stage, statement, and transaction timers use NANOSECOND on platforms where the NANOSECOND timer is available, MICROSECOND otherwise.

14. Execute the following query:

```
SELECT SLEEP(5);
```

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> SELECT SLEEP(5);
+-----+
| SLEEP(5) |
+-----+
|          0 |
+-----+
1 row in set (5.00 sec)
```

- The call to the SLEEP(5) function generates an artificially slow query.

15. Query the statements_with_runtimes_in_95th_percentile in the sys schema and locate the statement you executed in the preceding step:

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> SELECT * FROM
    -> sys.statements_with_runtimes_in_95th_percentile\G
...
***** 2. row *****
query: SELECT `SLEEP` (?)  

db: employees  

full_scan:  

exec_count: 1  

err_count: 0  

warn_count: 0  

total_latency: 5.00 s  

max_latency: 5.00 s  

avg_latency: 5.00 s  

rows_sent: 1  

rows_sent_avg: 1  

rows_examined: 0  

rows_examined_avg: 0  

first_seen: date-and-time  

last_seen: date-and-time  

digest: f528ef963b2913583b6ebe482a623b4e
...
```

- You can use this information to identify and fix poorly performing queries. You will learn more about query optimization in the lesson titled “Optimizing Queries.”

16. Exit the mysql session.

Practice 5-3: Installing MySQL Enterprise Monitor

Overview

In this practice, you install and configure MySQL Enterprise Monitor. Use the default settings wherever possible. Specify a password of `oracle` for any account that requires a password.

Duration

This practice should take you approximately 15 minutes to complete.

Tasks

1. Open a Linux terminal window.
2. Launch the MySQL Enterprise Monitor installer from the `/stage/MySQL-EM` directory by entering the following command at the Linux terminal prompt:

```
# /stage/MySQL-EM/mysqlmonitor-*x86_64-installer.bin
```
3. Click OK in the Language Selection dialog box to select the default installation language of English.
4. Read the message in the Info dialog box and click OK.
5. In the MySQL Enterprise Monitor Setup dialog box, click Forward.
6. In the Installation directory dialog box, note that the default installation directory is `/opt/mysql/enterprise/monitor`. Click Forward.
7. Select “Small system” in the Select Requirements dialog box and click Forward.
8. Note the default Tomcat Server Port (18080) and Tomcat SSL Port (18443) and click Forward.
9. In the Service Manager User Account dialog box, note that the default User Account is `mysqlmem` and click Forward.
10. In the Database Installation dialog box, note that the default setting is “I wish to use the bundled MySQL database.” This setting installs an additional MySQL instance on the same host, which Enterprise Monitor uses as its repository. Click Forward to configure the bundled MySQL server.
11. In the Repository Configuration dialog box, enter the following details before clicking Forward:
 - Repository Username: `service_manager`
 - Password: `oracle`
 - Re-enter: `oracle`
 - MySQL Database Port: 13306
 - MySQL Database Name: `mem`
12. In the Ready to Install dialog box, click Forward.
13. MySQL Enterprise Monitor installation begins. When it completes, click Forward.
14. Read the message in the Completed installing files dialog box and click Forward.

15. Read the message labeled “WARNING” in the Completing MySQL Enterprise Monitor Setup Wizard dialog box and click Forward.
16. Deselect View Readme File and click Finish to launch Enterprise Monitor.

The default browser (Mozilla Firefox) launches and connects to the Enterprise Monitor web application on your local host, displaying a “Your connection is not secure” warning.

Note: Because the SSL certificate that is shipped with MySQL Enterprise Monitor is self-signed, it is not recognized by the browser as a certificate issued by a trusted certificate broker. In your organization, you can replace the provided certificate with your own trusted certificate, if you have one. If you do not have such a certificate, you must create a security exception in the browser to enable the self-signed certificate. You will do this in the following steps.

17. Click the Advanced button on the “Your connection is not secure” webpage.
18. Click the Add Exception button to add a security exception to the browser to enable secure connections using the supplied self-signed certificate. The Add Security Exception dialog box displays.
19. Click Confirm Security Exception in the Add Security Exception dialog box. The browser redirects to the MySQL Enterprise Monitor Dashboard Setup page.
20. Enter the following details in the “Create user with ‘manager’ role” section:
 - Username: memmanager
 - Password: oracle
 - Confirm Password: oracle
21. In the “Create user with ‘agent’ role” section, enter the following details:
 - Username: memagent
 - Password: oracle
 - Confirm Password: oracle
22. If you are in a training center that uses a network proxy, select Use HTTP Proxy and enter the proxy details in the form that appears. For example, in an Oracle education center, enter the proxy “ges-proxy.us.oracle.com:80” with no username or password.
23. Click Complete Setup.
24. In the Welcome dialog box, select your time zone and locale and click Save.
25. Select Overview from the list of links on the left of the page.
26. Bookmark the Overview page by pressing Ctrl + D in the browser and clicking Done in the Bookmark Added dialog box.
27. MySQL Enterprise Monitor is now installed and running. Close the browser window.
28. Leave the Linux terminal window open for the next practice.

Solution 5-3: Installing MySQL Enterprise Monitor

Solution Steps

There are no solution steps for this activity. Follow the practice steps.

Practice 5-4: Monitoring Server Activity

Overview

In this practice, you use `mysqladmin` and MySQL Enterprise Monitor to view the status of a running MySQL instance while using the `mysqlslap` load emulation client to generate a sample load.

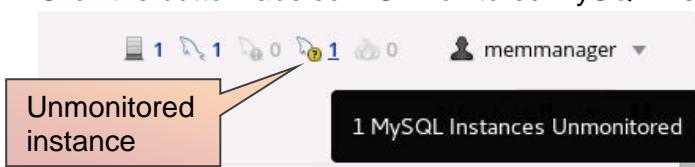
Note: In this practice, you open multiple Linux terminal prompts and a browser window containing MySQL Enterprise Monitor.

Duration

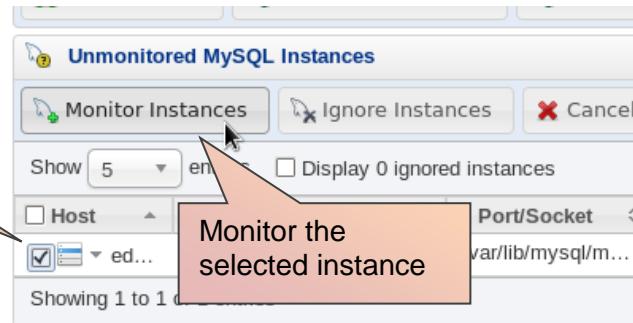
This practice should take you approximately 20 minutes to complete.

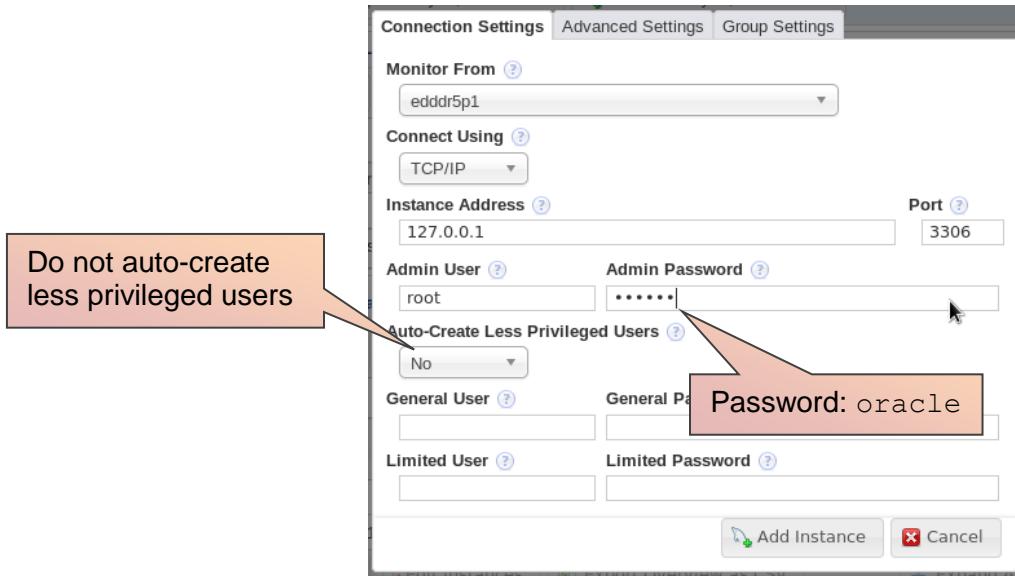
Tasks

1. Use `mysqladmin` to flush the status variables.
2. Use `mysqladmin` to display a status summary every 10 seconds.
3. Open the MySQL Enterprise Monitor Overview page.
 - a. Launch the Mozilla Firefox browser.
 - b. Select the Overview: MySQL Enterprise Dashboard bookmark that you created in the activities for the lesson titled “Installing MySQL Enterprise Monitor.”
 - c. Log in to Enterprise Monitor with the following credentials:
 - Username: `memmanager`
 - Password: `oracle`
4. Open the MySQL Instances page by clicking the Configuration > MySQL Instances link in the menu on the left-hand side of the Dashboard.
5. Add the instance that is running on port 3306 on your host to the list of monitored servers.
 - a. Click the button labeled 1 Unmonitored MySQL Instance near the top right.

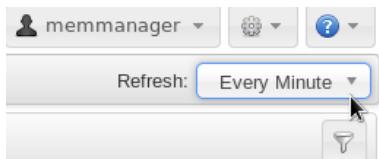


- b. In the Unmonitored MySQL Instances section that appears, select the MySQL instance running on port 3306 by using the check box in the Host column. Then click the Monitor Instances button.





- c. Select No in the Auto-Create Less Privileged Users selection box.
 - d. Type the password `oracle` into the Admin Password box.
 - e. Click Add Instance.
6. Remove the Enterprise Monitor bundled instance that is running on port 13306 from the monitored servers.
- a. Expand the section labeled All (2/2) by clicking the plus sign (+).
 - b. Select the instance running on port 13306.
 - c. Click Delete Instances.
 - d. In the Delete 1 Instance dialog box that appears, click Delete 1 Instance.
7. View the timeseries graphs for the configured instance. Click the Metrics > Timeseries Graphs in the menu on the left of the Dashboard. Examine the range of graphs available.
8. View the contents of the `/labs/slap-test-updates.sh` script and then execute it.
9. Return to the Enterprise Monitor window and change the Refresh option in the top-right



corner of the page so that it refreshes the graphs every 30 seconds.

10. View the differences in the graphs while the load emulation is running. In particular, note the changes in the number of connections, the CPU utilization, the disk and network I/O, and the number of row accesses and writes.
11. Kill the load emulation process.
12. Return to the Linux terminal window that contains the `mysqladmin status` output. Note the changes in the Threads, Questions, and Queries per second avg entries.
- The values have changed in several ways that you can explain with reference to your use of MySQL Enterprise Monitor and the load emulation client.

- The number of threads increased by a small amount when you connected to the instance with MySQL Enterprise Monitor and again when you launched the load emulation client.
- The number of questions increased rapidly while you ran the load emulation client.
- The average queries per second increased while you ran the load emulation client.

13. Kill the mysqladmin process.
14. Close the MySQL Enterprise Monitor web application page.
15. Stop the MySQL Enterprise Monitor service by entering the following command at a Linux terminal prompt and receiving the results shown:

```
# sh /opt/mysql/enterprise/monitor/mysqlmonitorctl.sh stop
Stopping tomcat service . [ OK ]
Stopping mysql service .. [ OK ]
```

Note: It is important to stop the MySQL Enterprise Monitor service. Not doing so will affect the outcome of later practices in the course.

16. Close all open Linux terminal windows.

Solution 5-4: Monitoring Server Activity

Solution Steps

1. Use mysqladmin to flush the status variables.

Enter the following command at a Linux terminal prompt:

```
mysqladmin -uroot -p flush-status
```

2. Use mysqladmin to display a status summary every 10 seconds.

Enter the following command at a Linux terminal prompt and receive the results shown:

```
# mysqladmin -uroot -p status -i10
Enter password: oracle
Uptime: 963185 Threads: 10 Questions: 60862 Slow queries: 3
Opens: 167 Flush tables: 1 Open tables: 132 Queries per
second avg: 0.063
Uptime: 963195 Threads: 10 Questions: 61460 Slow queries: 3
Opens: 167 Flush tables: 1 Open tables: 132 Queries per
second avg: 0.063
```

- The command displays status information every 10 seconds. The results will be different on your system.

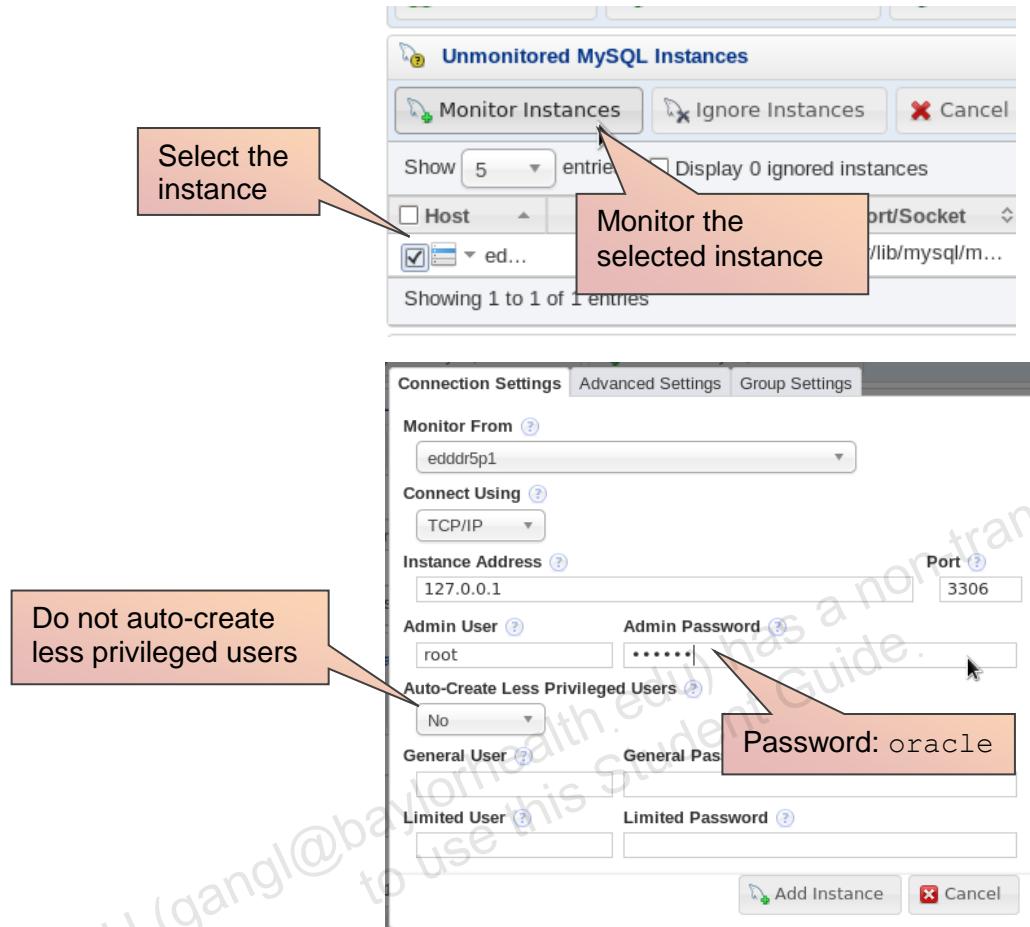
3. Open the MySQL Enterprise Monitor Overview page.

- a. Launch the Mozilla Firefox browser.
- b. Select the Overview: MySQL Enterprise Dashboard bookmark that you created in the activities for the lesson titled “Installing MySQL Enterprise Monitor.”
- c. Log in to Enterprise Monitor with the following credentials:
 - Username: memmanager
 - Password: oracle

4. Open the MySQL Instances page by clicking the Configuration > MySQL Instances link in the menu on the left-hand side of the Dashboard.
5. Add the instance that is running on port 3306 on your host to the list of monitored servers.
 - a. Click the button labeled 1 Unmonitored MySQL Instance near the top right.



- b. In the Unmonitored MySQL Instances section that appears, select the MySQL instance running on port 3306 by using the check box in the Host column. Then click the Monitor Instances button.



- c. Select No in the Auto-Create Less Privileged Users selection box.
- d. Type the password `oracle` in the Admin Password box.
- e. Click Add Instance.
6. Remove the Enterprise Monitor bundled instance that is running on port 13306 from the monitored servers.
- Expand the section labeled All (2/2) by clicking the plus sign (+).
 - Select the instance running on port 13306.
 - Click Delete Instances.
 - In the Delete 1 Instance dialog box that appears, click Delete 1 Instance.
7. View the timeseries graphs for the configured instance. Click the Metrics > Timeseries Graphs in the menu on the left of the Dashboard. Examine the range of graphs available.

8. View the contents of the `/labs/slap-test-updates.sh` script and then execute it. Open a new Linux terminal. Enter the following command at the Linux terminal prompt and receive the results shown:

```
# cat /labs/slap-test-updates.sh
#!/bin/bash
mysqlslap --user=root --password=oracle --concurrency=5 \
--iterations=100 --number-char-cols=4 --number-int-cols=7 \
--auto-generate-sql --number-of-queries=10000 \
--auto-generate-sql-load-type=update
```

The script uses the `mysqlslap` load emulation utility to run a benchmark with the following characteristics:

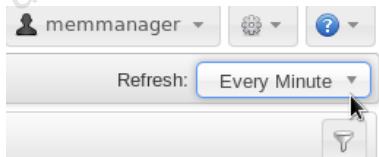
- 5 connections
- 100 iterations
- 10,000 statements per iteration
- Automatically generated load containing update operations
- Benchmark table containing four character columns and seven integer columns

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# sh /labs/slap-test-updates.sh
mysqlslap: [Warning] Using a password on the command line
interface can be insecure.
```

The command executes the load emulation client with the preceding configuration.

9. Return to the Enterprise Monitor window and change the Refresh option in the top right-



hand corner of the page so that it refreshes the graphs every 30 seconds.

10. View the differences in the graphs while the load emulation is running. In particular, note the changes in the number of connections, the CPU utilization, the disk and network I/O, and the number of row accesses and writes.
11. Kill the load emulation process.

Press `Ctrl + C` in the window that contains the `slap-test-updates.sh` script to terminate the process:

```
# sh /labs/slap-test-updates.sh
mysqlslap: [Warning] Using a password on the command line
interface can be insecure.

^C
```

12. Return to the Linux terminal window that contains the `mysqladmin status` output. Note the changes in the Threads, Questions, and Queries per second avg entries.

The values have changed in several ways that you can explain with reference to your use of MySQL Enterprise Monitor and the load emulation client.

- The number of threads increased by a small amount when you connected to the instance with MySQL Enterprise Monitor and again when you launched the load emulation client.
- The number of questions increased rapidly while you ran the load emulation client.
- The average queries per second increased while you ran the load emulation client.

13. Kill the mysqladmin process.

Press Ctrl + C in the window that contains the `slap-test-updates.sh` script to terminate the process:

```
# mysqladmin -uroot -p status -i10
...
Uptime: 963755 Threads: 10 Questions: 87554 Slow queries: 3
Opens: 177 Flush tables: 1 Open tables: 132 Queries per
second avg: 0.090
Uptime: 963765 Threads: 10 Questions: 88019 Slow queries: 3
Opens: 177 Flush tables: 1 Open tables: 132 Queries per
second avg: 0.091
^C
```

14. Close the MySQL Enterprise Monitor web application page.

15. Stop the MySQL Enterprise Monitor service by entering the following command at a Linux terminal prompt and receiving the results shown:

```
# sh /opt/mysql/enterprise/monitor/mysqlmonitorctl.sh stop
Stopping tomcat service . [ OK ]
Stopping mysql service .. [ OK ]
```

Note: It is important to stop the MySQL Enterprise Monitor service. Not doing so will affect the outcome of later practices in the course.

16. Close all open Linux terminal windows.

GANG LIU (gangli@baylorhealth.edu) has a non-transferable license
to use this Student Guide.

Practices for Lesson 6: Managing MySQL Users

Practices for Lesson 6: Overview

Overview

In these practices, you test your knowledge of user management in MySQL by answering quiz questions, creating and configuring users and roles, and modifying user permissions so that the user can execute queries on the database.

Assumptions

- You have created and populated the employees database in the practices for the lesson titled “Monitoring MySQL.”

Practice 6-1: Quiz—User Management

Overview

In this quiz, you answer questions about MySQL user management.

Duration

This practice should take you approximately five minutes to complete.

Quiz Questions

Choose the best answer from those provided for each multiple choice or True/False question.

- To determine which accounts can be used without specifying a password, use the following statement:

```
SELECT Host, User FROM mysql.user WHERE Password = '';
```

- a. True
- b. False

- Consider the following privilege settings for the accounts associated with a given MySQL username, where the `Select_priv` column indicates the setting for the global `SELECT` privilege:

```
mysql> SELECT Host, User, Select_priv FROM mysql.user
      -> WHERE User = 'Sasha';
+-----+-----+-----+
| Host | User | Select_priv |
+-----+-----+-----+
| 62.220.12.66 | Sasha | N   |
| 62.220.12.% | Sasha | Y   |
| 62.220.%     | Sasha | N   |
+-----+-----+-----+
```

Note: The `Select_priv` column indicates that the `SELECT` privilege for the second entry has been granted on a global scope (`*.*`). Assume that the `Sasha` accounts are not granted privileges in any of the other grant tables.

Can user `Sasha` select data from any table on the MySQL server when connecting from the following hosts?

- a. 62.220.12.66
 - b. 62.220.12.43
 - c. 62.220.42.43
 - d. localhost
- Which of the following user manipulation statements is true?
 - You can grant additional privileges to an existing user with either an `ALTER USER` or a `GRANT` statement.
 - You can remove a user with either a `DROP USER` or a `REVOKE` statement.

- c. You can set a password in both a `CREATE USER` and an `ALTER USER` statements.
4. Assume that a new user has been created, but that user account has not been granted any privileges yet. What can that user do at this point?
- Connect to the server
 - Browse all table data with statements such as `SELECT`
 - Browse all databases with statements such as `SHOW DATABASES`
5. Which of the following statements about the effects of privilege changes for existing connections are true?
- When a user is dropped, connections to the server by that user are terminated automatically.
 - Global privilege changes do not affect existing connections of a user. They take effect only the next time the user attempts to connect.
 - Database privilege changes take effect only at the time that a user next attempts to connect.
 - Database privilege changes take effect only at the time that a user next issues a `USE <database>` statement.
 - Table and column privileges take immediate effect.
6. A user wants to activate all the granted roles in the current session. Which of the following statements the user must run?
- `SET DEFAULT ROLE ALL`
 - `SET ROLE ALL`
 - `SET ROLE DEFAULT`
 - `SET activate_all_roles_on_login = ON;`
7. You can grant a user account to another user account. For example:

```
GRANT user1 TO user2;
```

- True
- False

Solution 6-1: Quiz—User Management

Quiz Solutions

1. **b.** False. The `mysql.user` table does not contain a `Password` column. It contains an `authentication_string` column that contains an encrypted password for some but not all authentication plugins.
2. Can user `Sasha` select data from any table on the MySQL server when connecting from the following hosts?
 - a. **No.** `62.220.12.66` is the most specific entry that matches the host that user `Sasha` is trying to connect from. Because the `SELECT` privilege for that entry is `N`, `Sasha` cannot select from any table on the server.
 - b. **Yes.** The most specific entry that matches `62.220.12.43` is `62.220.12.%`. Because the `SELECT` privilege for that entry is `Y`, user `Sasha` can select from any table on the server.
 - c. **No.** The most specific entry that matches `62.220.42.43` is `62.220.%`. Because the `SELECT` privilege for that entry is `N`, user `Sasha` cannot select from any table on the server.
 - d. **No.** There is no entry that matches `Sasha@localhost`. Therefore, user `Sasha` cannot even connect to the server from the `localhost`.
3. **c.** You can set a password in either a `CREATE USER` statement (for new users) or an `ALTER USER` statement (for existing users). `ALTER USER` can be used to change nonprivilege account settings such as expired status and password; if you want to grant additional privileges, use `GRANT`. You can remove a user completely with a `DROP USER` statement. Using `REVOKE`, you might be able to revoke all of the user's privileges, but the user account remains, as this example illustrates:

```
mysql> SHOW GRANTS FOR 'lennart';
+-----+
| Grants for lennart@%          |
+-----+
| GRANT USAGE ON *.* TO 'lennart'@'%'
+-----+
mysql> REVOKE ALL, GRANT OPTION FROM 'lennart';
mysql> SHOW GRANTS FOR 'lennart';
+-----+
| Grants for lennart@%          |
+-----+
| GRANT USAGE ON *.* TO 'lennart'@'%'
+-----+
mysql> DROP USER 'lennart';
mysql> SHOW GRANTS FOR 'lennart';
ERROR 1141 (42000): There is no such grant defined for user 'lennart'
on host '%'
```

4. a. Connect to the server. A new account with no explicit privileges can connect to the server and issue basic investigatory statements. Like any other user, that user gets the full output from statements that display information about the server, such as SHOW VARIABLES or SHOW STATUS. The user cannot get information about table data or other database objects except in very limited ways, because the account does not as yet have any privileges. SHOW DATABASES displays only the INFORMATION_SCHEMA database (which is a database that does not have a physical representation in the file system).
5. Which of the following statements about the effects of privilege changes for existing connections are true?
 - a. **False**. When a user is dropped, connections to the server by that user are not terminated automatically.
 - b. **True**. Global privilege changes do not affect the existing connections of a user. They take effect only the next time the user attempts to connect.
 - c. **False**. See the next item.
 - d. **True**. Database privilege changes take effect only at the time that a user next issues a USE <database> statement.
 - e. **True**. Table and column privileges take immediate effect.
6. b. SET ROLE ALL activates all the roles granted to the user in the current session. SET DEFAULT ROLE determines what roles are activated automatically when the user connects to the server. SET ROLE DEFAULT activates the user default role instead of all granted roles. SET activate_all_roles_on_login = ON; fails because this is a global variable.
7. a. True. The statement in the example will grant all privileges of user1 to user2; it works similar to granting a role to a user.

Practice 6-2: Creating Users and Roles

Overview

In this practice, you create and configure users and roles.

Duration

This practice should take you approximately 15 minutes to complete.

Tasks

1. Start a `mysql` client session in a Linux terminal window.
2. Create a new user called `kari` at the host `% .abc.com`.
3. Create 2 new roles called `r_manager` and `r_staff`.
4. Confirm that the new user and roles have been created by querying the `user`, `host`, and `authentication_string` columns of the `mysql.user` table.
5. View the full contents of the `r_staff` account row in the `mysql.user`.
6. Change the name of `r_staff` role to `r_clerk` using `RENAME USER` statement.
7. Confirm that the role name has been changed by querying the `user` and `host` columns of the `mysql.user` table.
8. Change the password for `kari` user on the `% .abc.com` host to `NewPass` by using an `ALTER USER` statement.
9. Configure the `kari@'% .abc.com'` account so that the password expires in 30 days.
10. Expire the password of the `kari@'% .abc.com'` account.
11. View the full contents of the `kari@'% .abc.com'` account row in the `mysql.user` table and observe the changes that you made in this practice.
12. Remove the `kari` user at the host `'% .abc.com'` using `DROP USER` statement. Confirm that the account no longer exists.
13. Remove the `r_clerk` role using `DROP ROLE` statement. Confirm that the role no longer exists.
14. Remove the `r_manager` role using `DROP USER` statement. Confirm that the role no longer exists.
15. Leave the `mysql` client session open for the next practice.

Solution 6-2: Creating Users and Roles

Solution Steps

1. Start a mysql client session in a Linux terminal window.

Enter the following command at a Linux terminal prompt and receive the results shown:

```
# mysql -uroot -p
Enter password: oracle
Welcome to the MySQL monitor. Commands end with ; or \g.
...
mysql>
```

2. Create a new user called kari at the host %.abc.com.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> CREATE USER kari@'%abc.com';
Query OK, 0 rows affected (#.## sec)
```

3. Create two new roles called r_manager and r_staff.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> CREATE ROLE r_manager, r_staff;
Query OK, 0 rows affected (#.## sec)
```

4. Confirm that the new user and roles have been created by querying the user, host, and authentication_string columns of the mysql.user table.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> SELECT user, host, authentication_string
-> FROM mysql.user\G
***** 1. row *****
      user: r_manager
      host: %
authentication_string:
***** 2. row *****
      user: r_staff
      host: %
authentication_string:
***** 3. row *****
      user: kari
      host: %.abc.com
authentication_string:
***** 4. row *****
      ...
7 row in set (#.## sec)
```

- Both the users and roles are stored in the mysql.user table.
- Roles have no password.

- The authentication string is blank for `kari@'%.abc.com'` because you did not set a password when you created the user account.
5. View the full contents of the `r_staff` account row in the `mysql.user`.

Enter the following statement at the `mysql` prompt and receive the results shown:

```
mysql> SELECT * FROM mysql.user WHERE user='r_staff'\G
***** 1. row ****
      Host: %
      User: r_staff
  Select_priv: N
Insert_priv: N
...
      plugin: caching_sha2_password
authentication_string:
  password_expired: Y
password_last_changed: YYYY-MM-DD HH:MI:SS
  password_lifetime: NULL
  account_locked: Y
...
1 row in set (#.## sec)
```

- When a role is created, it:
 - Is locked (`account_locked: Y`)
 - Has no password (`authentication_string` is blank)
 - Has default authentication plugin (`plugin: caching_sha2_password`)

6. Change the name of `r_staff` role to `r_clerk` using `RENAME USER` statement.

Enter the following statement at the `mysql` prompt and receive the results shown:

```
mysql> RENAME USER r_staff TO r_clerk;
Query OK, 0 rows affected (#.## sec)
```

7. Confirm that the role name has been changed by querying the `user` and `host` columns of the `mysql.user` table.

Enter the following statement at the `mysql` prompt and receive the results shown:

```
mysql> SELECT user, host FROM mysql.user;
+-----+-----+
| user           | host        |
+-----+-----+
| r_clerk        | %           |
| r_manager      | %           |
| kari           | %.abc.com   |
...
7 rows in set (#.## sec)
```

- `RENAME USER` statement can change user account and role name.

8. Change the password for `kari` user on the `% .abc .com` host to `NewPass` by using an `ALTER USER` statement.

Enter the following statement at the `mysql` prompt and receive the results shown:

```
mysql> ALTER USER kari@'% .abc .com' IDENTIFIED BY 'NewPass';
Query OK, 0 rows affected (#.## sec)
```

9. Configure the `kari@'% .abc .com'` account so that the password expires in 30 days.

Enter the following statement at the `mysql` prompt and receive the results shown:

```
mysql> ALTER USER kari@'% .abc .com'
      -> PASSWORD EXPIRE INTERVAL 30 DAY;
Query OK, 0 rows affected (#.## sec)
```

10. Expire the password of the `kari@'% .abc .com'` account.

Enter the following statement at the `mysql` prompt and receive the results shown:

```
mysql> ALTER USER kari@'% .abc .com' PASSWORD EXPIRE;
Query OK, 0 rows affected (#.## sec)
```

11. View the full contents of the `kari@'% .abc .com'` account row in the `mysql.user` table and observe the changes that you made in this practice.

Enter the following statement at the `mysql` prompt and receive the results shown:

```
mysql> SELECT * FROM mysql.user
      -> WHERE user='kari' AND host='% .abc .com' \G
***** 1. row *****
      Host: % .abc .com
      User: kari
      Select_priv: N
      Insert_priv: N
      ...
      plugin: caching_sha2_password
      authentication_string: $A$005$-1-;q%R
&d 008.^0PE7khAmgyEhFm5fbp4p.Sz0X0hjKUSjyV/q3WRU3Go8
      password_expired: Y
      password_last_changed: YYYY-MM-DD HH:MI:SS
      password_lifetime: 30
      account_locked: N
      ...
1 row in set (#.## sec)
```

- The `authentication_string` column now contains a hash value of the password according to the authentication scheme defined by the `caching_sha2_password` plugin.
- The `password_last_changed` contains today's date. The `password_lifetime` is 30 days, and the password is expired.
- The other settings have their default values.

12. Remove the `kari` user at the host '`%.abc.com`' using `DROP USER` statement. Confirm that the account no longer exists.

Enter the following statements at the `mysql` prompt and receive the results shown:

```
mysql> DROP USER kari@'%.abc.com';
Query OK, 0 rows affected (#.## sec)
mysql> SELECT user, host FROM mysql.user WHERE user='kari';
Empty set (#.## sec)
```

- The account no longer exists.

13. Remove the `r_clerk` role using `DROP ROLE` statement. Confirm that the role no longer exists.

Enter the following statements at the `mysql` prompt and receive the results shown:

```
mysql> DROP ROLE r_clerk;
Query OK, 0 rows affected (#.## sec)
mysql> SELECT user, host FROM mysql.user WHERE user='r_clerk';
Empty set (#.## sec)
```

- The role no longer exists.

14. Remove the `r_manager` role using `DROP USER` statement. Confirm that the role no longer exists.

Enter the following statements at the `mysql` prompt and receive the results shown:

```
mysql> DROP USER r_manager;
Query OK, 0 rows affected (#.## sec)
mysql> SELECT user, host FROM mysql.user WHERE user='r_manager';
Empty set (#.## sec)
```

- The role no longer exists.
- Both `DROP ROLE` and `DROP USER` statements can be used to remove roles.

15. Leave the `mysql` client session open for the next practice.

Practice 6-3: Granting Permissions

Overview

In this practice, you create a user and two roles. You grant some privileges on the `employees` database to the user and roles. You run some SQL statements to determine if the user session has the required privileges to access the table data. You activated the role granted to the user to test the privileges again.

Note: In this practice, you log in to a second `mysql` client as another user with different permissions. Ensure you do not mix up the two sessions. You might want to add a prompt to each `mysql` client session by executing a command such as “`PROMPT \U>`”.

Duration

This practice should take you approximately 20 minutes to complete.

Tasks

1. Create a new user `jan@localhost` with the password `Emp?Pass`.
2. Create two new roles called `r_mgr` and `r_emp`.
3. Grant the `SELECT` privilege on the `employees` tables in the `employees` database to `r_emp` role.
4. Grant the `INSERT`, `UPDATE`, and `DELETE` privileges on all tables in the `employees` database to `r_mgr` role.
5. Grant the `r_emp` role to `r_mgr` and `jan@localhost`.
6. Grant the `SELECT` privilege on the `salaries` tables in the `employees` database to `jan@localhost` user account.
7. Open a new Linux terminal window and start another `mysql` client session as the `jan@localhost` user.
8. Change the default database to `employees`. Try to retrieve five rows of data from the `employees` table and five rows of data from the `salaries` tables. Does the user have sufficient privileges to do so?
9. Check what roles are activated in `jan@localhost` session and show all the privileges granted to the user.
10. Activate all the roles granted to the user `jan@localhost` in the connected session and check what roles have been enabled. Show all the privileges granted to the user again.
11. Try to retrieve five rows of data from the `employees` tables. Does the user have sufficient privileges to do so?
12. Try to delete the employee with number 10001 from the `employees` table. Does the user have sufficient privileges to do so?
13. Exit the `mysql` clients that are logged in as `jan@localhost` and `root`.
14. Close all Linux terminal windows.

Solution 6-3: Granting Permissions

Solution Steps

1. Create a new user jan@localhost with the password Emp?Pass.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> CREATE USER jan@localhost IDENTIFIED BY 'Emp?Pass';
Query OK, 0 rows affected (#.## sec)
```

2. Create two new roles called r_mgr and r_emp.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> CREATE ROLE r_mgr, r_emp;
Query OK, 0 rows affected (#.## sec)
```

3. Grant the SELECT privilege on the employees tables in the employees database to r_emp role.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> GRANT SELECT ON employees.employees TO r_emp;
Query OK, 0 rows affected (#.## sec)
```

4. Grant the INSERT, UPDATE, and DELETE privileges on all tables in the employees database to r_mgr role.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> GRANT INSERT, UPDATE, DELETE ON employees.*
-> TO r_mgr;
Query OK, 0 rows affected (#.## sec)
```

5. Grant the r_emp role to r_mgr and jan@localhost.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> GRANT r_emp TO r_mgr, jan@localhost;
Query OK, 0 rows affected (#.## sec)
```

6. Grant the SELECT privilege on the salaries tables in the employees database to jan@localhost user account.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> GRANT SELECT ON employees.salaries TO jan@localhost;
Query OK, 0 rows affected (#.## sec)
```

7. Open a new Linux terminal window and start another `mysql` client session as the `jan@localhost` user.

Enter the following command at a new Linux terminal prompt and receive the results shown:

```
# mysql -u jan -p
Enter password: Emp?Pass
Welcome to the MySQL monitor. Commands end with ; or \g.
...
mysql>
```

8. Change the default database to `employees`. Try to retrieve five rows of data from the `employees` table and five rows of data from the `salaries` tables. Does the user have sufficient privileges to do so?

Enter the following statement at the `mysql` prompt logged in as `jan@localhost` and receive the results shown:

```
mysql> USE employees
Reading table information for completion of table and column
names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SELECT * FROM employees LIMIT 5;
ERROR 1142 (42000): SELECT command denied to user
'jan'@'localhost' for table 'employees'

mysql> SELECT * FROM salaries LIMIT 5;
+-----+-----+-----+-----+
| emp_no | salary | from_date | to_date   |
+-----+-----+-----+-----+
| 10001  | 60117  | 1986-06-26 | 1987-06-26 |
| 10001  | 62102  | 1987-06-26 | 1988-06-25 |
| 10001  | 66074  | 1988-06-25 | 1989-06-25 |
| 10001  | 66596  | 1989-06-25 | 1990-06-25 |
| 10001  | 66961  | 1990-06-25 | 1991-06-25 |
+-----+-----+-----+-----+
5 rows in set (#.## sec)
```

- The user `jan@localhost` does not have privilege to retrieve rows from the `employees` table. The privilege is granted through the `r_emp` role, which is not enabled in this session.
- The user `jan@localhost` has the privilege to retrieve rows from the `salaries` table. The privilege is granted directly to the user.

9. Check what roles are activated in `jan@localhost` session and show all the privileges granted to the user.

Enter the following statement at the `mysql` prompt logged in as `jan@localhost` and receive the results shown:

```
mysql> SELECT CURRENT_ROLE();
+-----+
| CURRENT_ROLE() |
+-----+
| NONE          |
+-----+
1 row in set (#.## sec)

mysql> SHOW GRANTS;
+-----+
| Grants for jan@localhost           |
+-----+
| GRANT USAGE ON *.* TO `jan`@`localhost`      |
| GRANT SELECT ON `employees`.`salaries` TO `jan`@`localhost` |
| GRANT `r_emp`@`%` TO `jan`@`localhost`        |
+-----+
3 rows in set (#.## sec)
```

- The session does not have any roles enabled.
- The user has been granted `r_emp` role, but it has not been activated.

10. Activate all the roles granted to the user `jan@localhost` in the connected session and check what roles have been enabled. Show all the privileges granted to the user again.

Enter the following statement at the `mysql` prompt logged in as `jan@localhost` and receive the results shown:

```

mysql> SET ROLE ALL;
Query OK, 0 rows affected (#.## sec)

mysql> SELECT CURRENT_ROLE();
+-----+
| CURRENT_ROLE() |
+-----+
| `r_emp`@`%`    |
+-----+
1 row in set (#.## sec)

mysql> SHOW GRANTS;
+-----+
| Grants for jan@localhost                                |
+-----+
| GRANT USAGE ON *.* TO `jan`@`localhost`                |
| GRANT SELECT ON `employees`.`employees` TO `jan`@`localhost` |
| GRANT SELECT ON `employees`.`salaries` TO `jan`@`localhost`   |
| GRANT `r_emp`@`%` TO `jan`@`localhost`                   |
+-----+
4 rows in set (0.00 sec)

```

- After a role has been activated in the session, SHOW GRANTS result contains all the privileges granted to the role.
11. Try to retrieve five rows of data from the employees tables. Does the user have sufficient privileges to do so?

Enter the following statement at the mysql prompt logged in as jan@localhost and receive the results shown:

```

mysql> SELECT * FROM employees LIMIT 5;
+-----+-----+-----+-----+-----+-----+
| emp_no | birth_date | first_name | last_name | gender | hire_date |
+-----+-----+-----+-----+-----+-----+
| 10001 | 1953-09-02 | Georgi      | Facello    | M       | 1986-06-26 |
| 10002 | 1964-06-02 | Bezalel     | Simmel     | F       | 1985-11-21 |
| 10003 | 1959-12-03 | Parto       | Bamford   | M       | 1986-08-28 |
| 10004 | 1954-05-01 | Chirstian   | Koblick   | M       | 1986-12-01 |
| 10005 | 1955-01-21 | Kyoichi     | Maliniak   | M       | 1989-09-12 |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

```

- The user jan@localhost has the privilege to retrieve rows from the employees table after the r_emp role has been enabled.

12. Try to delete the employee with number 10001 from the `employees` table. Does the user have sufficient privileges to do so?

Enter the following statement at the `mysql` prompt logged in as `jan@localhost` and receive the results shown:

```
mysql> DELETE FROM employees WHERE emp_no=10001;  
ERROR 1142 (42000): DELETE command denied to user  
'jan'@'localhost' for table 'employees'
```

- The user `jan@localhost` does not have the privilege to delete rows from the `employees` table.

13. Exit the `mysql` clients that are logged in as `jan@localhost` and `root`.

Enter the following statement at the `mysql` prompt logged in as `jan@localhost` and receive the results shown:

```
mysql> EXIT  
Bye
```

Enter the following statement at the `mysql` prompt logged in `root` and receive the results shown:

```
mysql> EXIT  
Bye
```

14. Close all Linux terminal windows.

GANG LIU (gangli@baylorhealth.edu) has a non-transferable license
to use this Student Guide.

Practices for Lesson 7: Securing MySQL

Practices for Lesson 7: Overview

Overview

In these practices, you test your knowledge of security in MySQL by answering quiz questions, explore encrypted connections that use SSL, and configure MySQL Enterprise Firewall.

Assumptions

- You have created and populated the `employees` database in the practices for the lesson titled “Monitoring MySQL.”
- The `firewall_training.sql` file is in the `/labs` directory.

Practice 7-1: Quiz – Securing MySQL

Overview

In this quiz, you answer questions about MySQL security.

Duration

This practice should take you approximately five minutes to complete.

Quiz Questions

Choose the best answer from those provided for each multiple choice or True/False question.

1. Which of the following are types of MySQL server and data security risks?
 - a) Eavesdropping
 - b) Altering
 - c) Playback
 - d) Denial of service
 - e) All of the above
2. Which of the following are the main security risks in a MySQL installation?
 - a) Networks, operating systems, and file systems
 - b) Networks, secure connections, and users
 - c) Users and data
3. True or false: MySQL password encryption is not enough to ensure the security of your database users if an attacker gains access to the `user` table.
 - a) True
 - b) False
4. Unencrypted connections are acceptable for moving data securely over a network, as long as users have passwords.
 - a) True
 - b) False
5. MySQL supports _____ connections between MySQL clients and the server for encryption of data.
 - a) SSL-only
 - b) SSL and SSH
6. MySQL Enterprise Firewall prevents _____.
 - a) client activity from unapproved applications
 - b) connections from hosts that are not in a defined whitelist
 - c) statements that are similar to those in a defined blacklist
 - d) statements that are unlike those in a defined whitelist
7. Which of the following actions must you perform to train MySQL Enterprise Firewall?
 - a) Contact MySQL Support to log a service request
 - b) Enable RECORDING mode for a specific user

- c) Insert prohibited statements into the `mysql.firewall_statements` table
- d) Run the `linux-train-firewall.sql` script (on Linux)

Solution 7-1: Quiz—Securing MySQL

Answers to Quiz Questions

1. e. All of the above
2. a. Networks, operating systems, and file systems
3. a. True. Attackers can use techniques such as brute force algorithms or rainbow tables to discover the plain text password if they gain access to the encrypted passwords in any password-protected system, including MySQL.
4. b. False. If you use unencrypted connections between the client and the server, a user with access to the network can watch all the traffic and observe (and modify) the data being sent or received.
5. b. SSL (secure socket layer) and SSH (secure shell) protocols
6. d. statements that are unlike those in a defined whitelist
7. b. Enable RECORDING mode for a specific user.

Practice 7-2: Enabling SSL for Secure Connections

Overview

In this practice, you explore the options that enable SSL for both client and server.

Note: This practice uses two terminal windows that the practice refers to as t1 and t2.

Duration

This practice should take you approximately 15 minutes to complete.

Tasks

1. Open two Linux terminal windows, which this practice refers to as t1 and t2.
2. In the first Linux terminal window t1, log in to the `mysql` client over TCP/IP.
3. In t1, check whether your MySQL server supports SSL by displaying the value of the `have_ssl` system variable.
4. In t1, check whether your current server connection uses SSL by executing `STATUS` (or `\s`).
5. In the second Linux terminal window t2, edit the `/etc/my.cnf` file with an option that disables SSL.
6. In the first terminal window t1, issue the `RESTART SQL` statement to restart the server and apply the changes you made to `/etc/my.cnf`.
7. In t1, enter a command such as `STATUS` to re-establish the connection to the MySQL server.
8. In t1, issue a statement to check whether the server still supports SSL.
9. In t1, check whether the current connection to the MySQL server uses SSL by executing `STATUS` (or `\s`).
10. In terminal window t2, remove the line that disables SSL from the `/etc/my.cnf` file.
11. In the first terminal window t1, issue the `RESTART SQL` statement to restart the server and apply the changes you made to `/etc/my.cnf`.
12. In t1, exit the `mysql` client.
13. In t1, log in to the `mysql` client as `root` using TCP and the `--ssl-mode=DISABLED` option.
14. In t1, check whether the server supports SSL.
15. In t1, check whether your current server connection uses SSL.
16. In t1, exit the `mysql` client.
17. In t1, log in to the `mysql` client over TCP.
18. In t1, display the list of TLS versions permitted by the server.
19. In t1, display the version of TLS the current connection is using.
20. In t1, display the list of ciphers that the server permits.

21. In t1, execute a single SQL statement that displays the current cipher (`Ssl_cipher_status` variable) and the list of ciphers permitted for the current session (`Ssl_cipher_list` status variable).
22. In t1, execute a single SQL statement that displays the names of the files that contain the following:
 - the digital certificate for the CA
 - the server's digital certificate
 - the server's private key
23. Leave both terminal windows open for the next practice.

Solution 7-2: Enabling SSL for Secure Connections

Solution Steps

1. Open two Linux terminal windows, which this practice refers to as t1 and t2.
2. In the first Linux terminal window t1, log in to the mysql client over TCP/IP.

Enter the following statement at the mysql prompt in t1 and receive the results shown:

```
# mysql -uroot -p --protocol=TCP
Enter password: oracle
Welcome to the MySQL monitor. Commands end with ; or \g.
...
...
```

3. In t1, Check whether your MySQL server supports SSL by displaying the value of the have_ssl system variable.

Enter the following statement at the mysql prompt in t1 and receive the results shown:

```
mysql> SHOW VARIABLES LIKE 'have_ssl';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| have_ssl     | YES   |
+-----+-----+
1 row in set (#.## sec)
```

- SSL is enabled for the server.
- 4. In t1, check whether your current server connection uses SSL by executing STATUS (or \s).

Enter the following statement at the mysql prompt in t1 and receive the results shown:

```
mysql> STATUS
...
SSL:          Cipher in use is DHE-RSA-AES128-GCM-SHA256
...
```

- The result shows that your connection uses SSL with the DHE-RSA-AES128-GCM-SHA256 cipher. The mysql client uses SSL when it is available.

5. In the second Linux terminal window t_2 , edit the `/etc/my.cnf` file with an option that disables SSL.

In terminal window t_2 , use a text editor such as `vi` to add the following line to the `/etc/my.cnf` file:

```
[mysqld]
...
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Disabling symbolic-links is recommended to prevent assorted
security risks
symbolic-links=0
ssl=0
...
```

6. In the first terminal window t_1 , issue the `RESTART` SQL statement to restart the server and apply the changes you made to `/etc/my.cnf`.

Enter the following statement at the `mysql` prompt in t_1 and receive the results shown:

```
mysql> RESTART;
Query OK, 0 rows affected (#.# sec)
```

7. In t_1 , enter a command such as `STATUS` to re-establish the connection to the MySQL server.

Enter the following statement at the `mysql` prompt in t_1 and receive the results shown:

```
mysql> STATUS
ERROR 2013 (HY000): Lost connection. Trying to reconnect...
Connection id:      97
Current database: *** NONE ***
```

- As the server takes some time to restart, if you encounter an error connecting to the server, retry the `STATUS` statement until you successfully connect to the server.

8. In t_1 , issue a statement to check whether the server still supports SSL.

Enter the following statement at the `mysql` prompt in t_1 and receive the results shown:

```
mysql> SHOW VARIABLES LIKE 'have_ssl';
+-----+-----+
| Variable_name | Value   |
+-----+-----+
| have_ssl     | DISABLED |
+-----+-----+
1 row in set (#.# sec)
```

- SSL is disabled for the server.

9. In t1, check whether the current connection to the MySQL server uses SSL by executing STATUS (or \s).

Enter the following statement at the mysql prompt in t1 and receive the results shown:

```
mysql> STATUS
...
SSL:           Not in use
...
```

- The result shows that your connection is not using SSL.

10. In terminal window t2, remove the line that disables SSL from the /etc/my.cnf file.

In terminal window t2, use a text editor such as vi to remove the ssl=0 line from the /etc/my.cnf file so that the [mysqld] section appears as follows:

```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Disabling symbolic-links is recommended to prevent assorted
# security risks
symbolic-links=0
...
```

11. In the first terminal window t1, issue the RESTART SQL statement to restart the server and apply the changes you made to /etc/my.cnf.

Enter the following statement at the mysql prompt in t1 and receive the results shown:

```
mysql> RESTART;
Query OK, 0 rows affected (#.## sec)
```

12. In t1, exit the mysql client.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> EXIT
Bye
```

13. In t1, log in to the mysql client as root using TCP and the --ssl-mode=DISABLED option.

Enter the following command at the t1 Linux terminal prompt and receive the results shown:

```
# mysql -uroot -p --protocol=TCP --ssl-mode=DISABLED
Enter password: oracle
Welcome to the MySQL monitor. Commands end with ; or \g.
...
```

14. In t1, check whether the server supports SSL.

Enter the following statement at the mysql prompt in t1 and receive the results shown:

```
mysql> SHOW VARIABLES LIKE 'have_ssl';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| have_ssl      | YES   |
+-----+-----+
1 row in set (#.# sec)
```

- The result shows that the server supports SSL even though you disabled it for this connection.

15. In t1, check whether your current server connection uses SSL.

Enter the following statement at the mysql prompt in t1 and receive the results shown:

```
mysql> STATUS
...
SSL:           Not in use
...
```

- The result shows that, although the server supports SSL, this connection does not use it.

16. In t1, exit the mysql client.

Enter the following statement at the mysql prompt in t1 and receive the results shown:

```
mysql> EXIT
Bye
```

17. In t1, log in to the mysql client over TCP.

Enter the following command at the t1 Linux terminal prompt and receive the results shown:

```
# mysql -uroot -p --protocol=TCP
Enter password: oracle
Welcome to the MySQL monitor. Commands end with ; or \g.
...
```

18. In t1, display the list of TLS versions permitted by the server.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> SHOW GLOBAL VARIABLES LIKE 'tls_version';
+-----+-----+
| Variable_name | Value          |
+-----+-----+
| tls_version   | TLSv1,TLSv1.1,TLSv1.2 |
+-----+-----+
1 row in set (#.# sec)
```

19. In t1, display the version of TLS the current connection is using.

Enter the following statement at the mysql prompt in t1 and receive the results shown:

```
mysql> SHOW SESSION STATUS LIKE 'Ssl_version';
+-----+-----+
| Variable_name | Value   |
+-----+-----+
| Ssl_version   | TLSv1.2 |
+-----+-----+
1 row in set (#.## sec)
```

- In the sample output, the current connection uses TLSv1.2.

20. In t1, display the list of ciphers that the server permits.

Enter the following statement at the mysql prompt in t1 and receive the results shown:

```
mysql> SHOW GLOBAL VARIABLES LIKE 'ssl_cipher';
+-----+-----+
| Variable_name | Value   |
+-----+-----+
| ssl_cipher    |       |
+-----+-----+
1 row in set (#.## sec)
```

- The result is blank, which means that the cipher selection is not restricted and that MySQL uses the strongest cipher available for both client and server.

21. In t1, execute a single SQL statement that displays the current cipher (Ssl_cipher status variable) and the list of ciphers permitted for the current session (Ssl_cipher_list status variable).

Enter the following statement at the mysql prompt in t1 and receive the results shown:

```
mysql> SHOW SESSION STATUS LIKE 'Ssl_cipher%\G
***** 1. row *****
Variable_name: Ssl_cipher
      Value: DHE-RSA-AES128-GCM-SHA256
***** 2. row *****
Variable_name: Ssl_cipher_list
      Value: ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-
GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-
ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-GCM-
SHA256:DHE-DSS-AES128-GCM-SHA256:DHE-RSA-AES128-SHA256:DHE-DSS-
AES128-SHA256:DHE-DSS-AES256-GCM-SHA384:DHE-RSA-AES256-
SHA256:...
2 rows in set (#.## sec)
```

- The output of the statement shows that DHE-RSA-AES128-GCM-SHA256 is the cipher for the session and lists all the ciphers that the current session permits.

22. In t1, execute a single SQL statement that displays the names of the files that contain the following:
- the digital certificate for the CA
 - the server's digital certificate
 - the server's private key

Enter the following statement at the mysql prompt in t1 and receive the results shown:

```
mysql> SHOW GLOBAL VARIABLES LIKE 'ssl_%';
+-----+-----+
| Variable_name | Value           |
+-----+-----+
| ssl_ca       | ca.pem         |
| ssl_capath    |                 |
| ssl_cert      | server-cert.pem |
| ssl_cipher    |                 |
| ssl_crl       |                 |
| ssl_crlpath   |                 |
| ssl_fips_mode | OFF             |
| ssl_key       | server-key.pem |
+-----+-----+
8 rows in set (#.## sec)
```

23. Leave both terminal windows open for the next practice.

Practice 7-3: Configuring MySQL Enterprise Firewall

Overview

In this practice, you configure the MySQL Enterprise Firewall by training it with whitelisted statements for a single user, enabling firewalled protection, and resetting the firewall.

Duration

This practice should take you approximately 20 minutes to complete.

Tasks

1. In the mysql session in t1 from the preceding practice, create the user employee@localhost with the password MySQL8.0!.
2. In t1, grant the employee@localhost user SELECT permissions on all tables in the employees database.
3. In t1, change the current database to the mysql system database.
4. In t1, install the MySQL Enterprise Firewall plugin by executing the SOURCE statement on the linux_install_firewall.sql script in the /usr/local/mysql/share directory.
5. In t1, execute the sp_set_firewall_mode stored procedure to enable RECORDING mode for the employee@localhost user.
6. At the Linux terminal prompt in t2 from the preceding practice, view the contents of the firewall_training.sql script in the /labs directory.
7. In t2, execute the firewall-training.sql script in the employees database as the employee user (password MySQL8.0!) by issuing the following command:

```
# mysql -uemployee -p employees < /labs/firewall_training.sql
```

8. In t1, enable PROTECTING mode for the employee@localhost user.
9. In t2, log in to mysql as the employee user.
10. In t2, manually execute the four SQL statements in the firewall_training.sql script as shown below:

```
1. USE employees  
2. SELECT dept_no, dept_name FROM departments;  
3. SELECT first_name, last_name FROM employees  
    WHERE emp_no=10001;  
4. SELECT departments.dept_name, employees.first_name,  
    employees.last_name  
    FROM departments NATURAL JOIN dept_emp  
    NATURAL JOIN employees  
    WHERE departments.dept_no='d001';
```

11. In t2, query the employees table to display the first and last name of employee number 20002.

12. In t2, select the emp_no, first_name, and last_name columns from the employees table for the employee with number 20002.
13. In t2, using a whitelisted pattern, select the department name and first and last names for all employees in the research department (`dept_no='d008'`).
14. In t2, change the first name of employee number 10001 to “Georgio”.
15. In t1 as the MySQL root user, reset the employee@localhost user’s firewall whitelist.
16. In t2, select the emp_no, first_name, and last_name columns from the employees table for the employee with number 20002.
17. In t2, exit the mysql client logged in as employee.
18. In t1 as the MySQL root user, display the global status variables that contain information about the firewall.
19. In t1, log out of the mysql client.
20. Close all open Linux terminal windows.

Solution 7-3: Configuring MySQL Enterprise Firewall

Solution Steps

1. In the mysql session in t1 from the preceding practice, create the user employee@localhost with the password MySQL8.0!.

Enter the following statement at the mysql prompt in t1 and receive the results shown:

```
mysql> CREATE USER employee@localhost  
-> IDENTIFIED BY 'MySQL8.0!';  
Query OK, 0 rows affected (#.## sec)
```

2. In t1, grant the employee@localhost user SELECT permissions on all tables in the employees database.

Enter the following statement at the mysql prompt in t1 and receive the results shown:

```
mysql> GRANT SELECT ON employees.* TO employee@localhost;  
Query OK, 0 rows affected (#.## sec)
```

3. In t1, change the current database to the mysql system database.

Enter the following statement at the mysql prompt in t1 and receive the results shown:

```
mysql> USE mysql  
Reading table information for completion of table and column  
names  
You can turn off this feature to get a quicker startup with -A  
  
Database changed
```

4. In t1, install the MySQL Enterprise Firewall plugin by executing the SOURCE statement on the linux_install_firewall.sql script in the /usr/local/mysql/share directory.

Enter the following statement at the mysql prompt in t1 and receive the results shown:

```
mysql> SOURCE  
-> /usr/local/mysql/share/linux_install_firewall.sql  
Query OK, 0 rows affected (#.## sec)  
  
Query OK, 0 rows affected (#.## sec)  
...
```

5. In t1, execute the `sp_set_firewall_mode` stored procedure to enable RECORDING mode for the `employee@localhost` user.

Enter the following statement at the `mysql` prompt in t1 and receive the results shown:

```
mysql> CALL mysql.sp_set_firewall_mode(
->      'employee@localhost', 'RECORDING'
-> );
+-----+
| read_firewall_whitelist(arg_userhost,FW.rule) |
+-----+
| Imported users: 0 Imported rules: 0           |
+-----+
1 row in set (#.## sec)

Query OK, 0 rows affected (#.## sec)
```

6. At the Linux terminal prompt in t2 from the preceding practice, view the contents of the `firewall_training.sql` script in the `/labs` directory.

Enter the following command at the t2 Linux terminal prompt and receive the results shown:

```
# cat /labs/firewall_training.sql
USE employees
SELECT dept_no, dept_name FROM departments;
SELECT first_name, last_name FROM employees
WHERE emp_no=10001;
SELECT departments.dept_name,
       employees.first_name, employees.last_name
  FROM departments NATURAL JOIN dept_emp
 NATURAL JOIN employees
 WHERE departments.dept_no='d001';
```

- The script contains statements that query the `employees` database.

7. In t2, execute the `firewall-training.sql` script in the `employees` database as the `employee` user (password `MySQL8.0!`) by issuing the following command:

```
# mysql -uemployee -p employees < /labs/firewall_training.sql
```

Enter the following command at the t2 Linux terminal prompt and receive the results shown:

```
# mysql -uemployee -p employees < /labs/firewall_training.sql
Enter password: MySQL8.0!
dept_no dept_name
d009 Customer Service
d005 Development
d002 Finance
...
Marketing Haldun Zaumen
Marketing Dharmaraja Ertl
Marketing Danai Hidayat
Marketing Siamak Salverda
```

- The command takes a few moments to execute and produces a lot of data.
8. In t1, enable PROTECTING mode for the `employee@localhost` user.

Enter the following statement at the `mysql` prompt in t1 and receive the results shown:

```
mysql> CALL mysql.sp_set_firewall_mode(
    ->     'employee@localhost', 'PROTECTING'
    -> );
Query OK, 5 rows affected (#.## sec)
```

9. In t2, log in to `mysql` as the `employee` user.

Enter the following command at the t2 Linux terminal prompt and receive the results shown:

```
# mysql -uemployee -p
Enter password: MySQL8.0!
...
```

10. In t2, manually execute the four SQL statements in the `firewall_training.sql` script as shown below:

```
1. USE employees
2. SELECT dept_no, dept_name FROM departments;
3. SELECT first_name, last_name FROM employees
   WHERE emp_no=10001;
4. SELECT departments.dept_name, employees.first_name,
   employees.last_name
   FROM departments NATURAL JOIN dept_emp
   NATURAL JOIN employees
   WHERE departments.dept_no='d001';
```

Enter the following statements at the mysql prompt in t2 and receive the results shown:

```
mysql> USE employees
Database changed

mysql> SELECT dept_no, dept_name FROM departments;
+-----+-----+
| dept_no | dept_name      |
+-----+-----+
| d009   | Customer Service |
| d005   | Development      |
| d002   | Finance          |
| d003   | Human Resources  |
| d001   | Marketing         |
| d004   | Production        |
| d006   | Quality Management|
| d008   | Research          |
| d007   | Sales             |
+-----+-----+
9 rows in set (#.## sec)

mysql> SELECT first_name, last_name FROM employees
-> WHERE emp_no=10001;
+-----+-----+
| first_name | last_name |
+-----+-----+
| Georgi     | Facello    |
+-----+-----+
1 row in set (#.## sec)

mysql> SELECT departments.dept_name,
->     employees.first_name, employees.last_name
->   FROM departments NATURAL JOIN dept_emp
->   NATURAL JOIN employees
-> WHERE departments.dept_no='d001';
+-----+-----+-----+
| dept_name | first_name   | last_name  |
+-----+-----+-----+
| Marketing | Cristinel  | Bouloucos  |
| Marketing | Georgy     | Dredge     |
| Marketing | Berhard    | McFarlin   |
| Marketing | Lunjin    | Giveon    |
| Marketing | Yucel     | Auria     |
+-----+-----+-----+
```

```

...
| Marketing | Haldun          | Zaumen      |
| Marketing | Dharmaraja       | Ertl        |
| Marketing | Danai           | Hedayat     |
| Marketing | Siamak          | Salverda    |
+-----+-----+-----+
20211 rows in set (#.## sec)

```

- The statements execute successfully.

11. In t2, query the employees table to display the first and last name of employee number 20002.

Enter the following statement at the mysql prompt in t2 and receive the results shown:

```

mysql> SELECT first_name, last_name FROM employees
      -> WHERE emp_no=20002;
+-----+-----+
| first_name | last_name |
+-----+-----+
| Jaber     | Brender   |
+-----+-----+
1 row in set (#.## sec)

```

- The statement succeeds.

12. In t2, select the emp_no, first_name, and last_name columns from the employees table for the employee with number 20002.

Enter the following statement at the mysql prompt in t2 and receive the results shown:

```

mysql> SELECT emp_no, first_name, last_name FROM employees
      -> WHERE emp_no=20002;
ERROR 1045 (28000): Statement was blocked by Firewall

```

- The firewall blocks the statement because it does not conform to a whitelisted pattern.

13. In t2, using a whitelisted pattern, select the department name and first and last names for all employees in the research department (`dept_no='d008'`).

Enter the following statement at the `mysql` prompt in t2 and receive the results shown:

```
mysql> SELECT departments.dept_name,
->     employees.first_name, employees.last_name
-> FROM departments NATURAL JOIN dept_emp
-> NATURAL JOIN employees
-> WHERE departments.dept_no='d008';
+-----+-----+
| dept_name | first_name      | last_name       |
+-----+-----+
| Research   | Tzvetan        | Zielinski      |
| Research   | Guoxiang       | Nooteboom      |
| Research   | Lillian        | Haddadi        |
| Research   | Weiyi          | Meriste        |
| Research   | Lucien         | Rosenbaum      |
...
| Research   | Nigel          | Solovay        |
| Research   | Bernd          | Baumann        |
| Research   | Masanao        | Ducloy         |
| Research   | Gila           | Lukaszewicz   |
+-----+-----+
21126 rows in set (#.## sec)
```

- The statement succeeds.

14. In t2, change the first name of employee number 10001 to “Georgio”.

Enter the following statement at the `mysql` prompt in t2 and receive the results shown:

```
mysql> UPDATE employees SET first_name='Georgio'
-> WHERE emp_no=10001;
ERROR 1045 (28000): Statement was blocked by Firewall
```

- The `employee` user does not have permission to update the `employees` table. However, the firewall blocks the statement before MySQL checks the permissions, because the statement does not conform to a whitelisted pattern.

15. In t1 as the MySQL root user, reset the `employee@localhost` user’s firewall whitelist.

Enter the following statement at the `mysql` prompt in t1 and receive the results shown:

```
mysql> CALL mysql.sp_set_firewall_mode(
->     'employee@localhost', 'RESET'
-> );
Query OK, 0 rows affected (#.## sec)
```

16. In t2, select the emp_no, first_name, and last_name columns from the employees table for the employee with number 20002.

Enter the following statement at the mysql prompt in t2 and receive the results shown:

```
mysql> SELECT emp_no, first_name, last_name FROM employees
-> WHERE emp_no=20002;
+-----+-----+-----+
| emp_no | first_name | last_name |
+-----+-----+-----+
| 20002 | Jaber      | Brender   |
+-----+-----+-----+
1 row in set (#.## sec)
```

- The statement succeeds because the firewall is no longer protecting the employee@localhost account.

17. In t2, exit the mysql client logged in as employee.

Enter the following statement at the mysql prompt in t2 and receive the results shown:

```
mysql> EXIT
Bye
```

18. In t1 as the MySQL root user, display the global status variables that contain information about the firewall.

Enter the following statement at the mysql prompt in t1 and receive the results shown:

```
mysql> SHOW GLOBAL STATUS LIKE 'Firewall%';
+-----+-----+
| Variable_name          | Value |
+-----+-----+
| Firewall_access_denied | 4     |
| Firewall_access_granted | 7     |
| Firewall_access_suspicious | 0     |
| Firewall_cached_entries | 0     |
+-----+-----+
4 rows in set (#.## sec)
```

- The Firewall_* status variables store counts of the occasions where access was granted, denied, or viewed as suspicious.

19. In t1, log out of the mysql client.

Enter the following statement at the mysql prompt in t1 and receive the results shown:

```
mysql> EXIT
Bye
```

20. Close all open Linux terminal windows.

Practices for Lesson 8: Maintaining a Stable System

Practices for Lesson 8: Overview

Overview

In these practices, you will test your knowledge of how to maintain a stable MySQL system by answering quiz questions and identifying the cause of poor database performance.

Assumptions

- You have created and populated the `employees` database in the practices for the lesson titled “Monitoring MySQL.”
- The `slap-test-updates.sh` file is in the `/labs` directory.

Practice 8-1: Quiz—Maintaining a Stable System

Overview

In this quiz, you answer questions about maintaining a stable MySQL system.

Duration

This practice should take you approximately 15 minutes to complete.

Quiz Questions

Choose the best answer from those provided for each multiple choice or True/False question.

1. Why is database server stability difficult to maintain?
 - a) Applications change due to growth or changes in business needs.
 - b) Computer hardware is expensive to purchase and maintain.
 - c) It is difficult to connect networked software to MySQL.
 - d) MySQL changes its configuration file to suit changes in usage.
2. Which of the following values does not belong in a performance baseline?
 - a) Configuration file contents (`/etc/my.cnf` on the server and all other configuration files used by the server or application client)
 - b) Connection username, password, host, and protocol
 - c) Hard disk space in use on the entire server including that used by MySQL's databases
 - d) Output of `SHOW GLOBAL VARIABLES`
 - e) RAM in use by the database server process
3. True or False: Application profiling is performed by developers and does not have any use when troubleshooting MySQL issues.
4. Which of the following is not a potential single point of failure?
 - a) Network adaptor connected to a network switch
 - b) SATA disk in a RAID 10 array
 - c) Server power supply connected to a UPS
 - d) Server room in the IT department
5. Which of the following is larger?
 - a) Logical InnoDB table size
 - b) Physical InnoDB table size
6. When you scale ___, you add more servers to the application environment. Which word is missing from the preceding sentence?
 - a) Out
 - b) Up
7. If a MySQL server fails to start, where is the first place you should look?
 - a) Configuration file (`/etc/my.cnf` or other files that the server reads)
 - b) Error log (in the location specified by `--log-error`)
 - c) Operating system log file (`syslog` or `messages`)

- d) Performance baseline
8. When you delete a row, which lock type does InnoDB hold while deleting the row?
- a) S
 - b) X
 - c) IS
 - d) IX
9. In the answers to the preceding question, what does the “I” stand for?
- a) Idempotent
 - b) Identity
 - c) Intention
 - d) Iterative
10. What does EXPLAIN display?
- a) An aggregate of the data affected by a statement
 - b) Details about the optimizer’s index choices (if any) for the statement
 - c) Help file information about the statement’s syntax
 - d) Summary of storage engine locks acquired by the statement
11. When EXPLAIN shows 1 warning after displaying its standard output, what do you see when you subsequently run SHOW WARNINGS?
- a) Length of time that the statement takes when you execute it
 - b) Reason you cannot run EXPLAIN for that statement type
 - c) Rephrased pseudo-SQL version of the statement
 - d) Standard MySQL disclaimer for optimizer precision
 - e) Syntax error that caused the warning
12. What does ANALYZE TABLE do?
- a) Displays details about the table structure and metadata
 - b) Ensures that table data conforms to all constraints
 - c) Finds inconsistencies and other problems in an InnoDB table
 - d) Stores key distribution statistics of a table
13. In a typical MySQL server configuration, to what value should --innodb_force_recovery be set?
- a) 0
 - b) 1
 - c) 4
 - d) 6
14. Which of the following products does not provide tools and configuration to enable high availability and automatic failover?
- a) MySQL Cluster
 - b) MySQL Enterprise Monitor
 - c) MySQL InnoDB Cluster
 - d) MySQL Replication

Solution 8-1: Quiz—Maintaining a Stable System

Quiz Solutions

1. **a.** As the user base grows and business adapts, the application changes how it interacts with the database in both volume and access patterns.
2. **b.** Server connections (credentials and network location) do not generally change as much as the application does. When they change (for example, after a server migration to a new subnet, or a security audit recommends compartmentalized application accounts), they do not affect the performance of the server.
3. **False.** An application profile might indicate that the largest delay in a function comes from a single query execution, and that information is useful to a DBA who is troubleshooting a slowdown.
4. **b.** A RAID 10 array has redundant disks. Often, a server has only one network interface or power supply, and a server room might be subject to natural disaster, fire, or some other localized problem.
5. **b.** Physical size includes metadata and empty space in the tablespace file.
6. **a.** Out. Scaling up involves adding more resources such as CPU power or RAM to a single server.
7. **b.** MySQL's error log shows why MySQL fails to start and usually indicates the next step in troubleshooting. If the failure is caused by a change in the configuration, you can identify the change by examining the `my.cnf` file or the baseline, and the operating system's log file might show the cause of some failures such as missing users or groups or security denials.
8. **b.** To modify a row, an InnoDB transaction must acquire an exclusive (`X`) write lock, which locks rows for the exclusive use of that transaction.
9. **c.** Intention. InnoDB transactions request an `IS` or `IX` lock before acquiring an `S` or `X` row lock, respectively, to indicate the transaction's intent.
10. **b.** Displayed information includes the indexes that are available to fulfill the query request, the actual index chosen by the optimizer, along with its key length in bytes and an estimate of the number of rows affected.
11. **c.** The optimizer refactors the statement to perform some optimizations, and `SHOW WARNINGS` displays the note-level warning that `EXPLAIN` generates, containing a pseudo-SQL version of that refactored statement.
12. **d.** The optimizer uses key distribution statistics to choose the best index for each statement. Such statistics are gathered and stored automatically by default when 10% of data in a table changes, and they are also generated when you run `ANALYZE TABLE`.
13. **a.** Zero. The default value of 0 causes InnoDB to recover from a shutdown or crash when it restarts. You should use only larger values (1 and higher) if InnoDB cannot start because of tablespace corruption.
14. **b.** MySQL Enterprise Monitor is a web-enabled monitoring solution and does not perform server administration or failover.

Practice 8-2: Identifying the Cause of Slow Performance

Overview

In this practice, you investigate blocked queries by using `SHOW PROCESSLIST` and the `sys` schema's `innodb_lock_waits` view.

Duration

This practice should take you approximately 15 minutes to complete.

Tasks

1. Open three Linux terminal windows. This practice refers to these terminal windows as `t1`, `t2`, and `t3`.
2. In `t1` and `t2`, log in to the `mysql` client and set the prompt to “`1>`” and “`2>`” respectively.
3. In `t1`, start a new transaction and execute a query for all rows and all columns in the `employees.departments` table with the `FOR UPDATE` modifier.
4. In `t2`, insert a new department called `Distribution` with a `dept_no` of `d010`.
5. In `t2`, increase the InnoDB lock timeout (system variable `innodb_lock_wait_timeout`) to 3600 seconds (1 hour) and re-execute the `INSERT` statement in the preceding step. Observe it for a minute or two before continuing with the next step.
6. In `t1`, display the `PROCESSLIST`.
7. In `t1`, use `sys.session` to display an alternative view of the running processes.
8. In `t1`, query the `sys` schema's `innodb_lock_waits` view by entering the following statement at the `mysql` prompt. Note the `blocking_pid` value. This is the process ID of the blocking thread.

```
SELECT
    waiting trx_id,
    waiting pid,
    waiting query,
    blocking trx_id,
    blocking pid,
    blocking query
FROM sys.innodb_lock_waits;
```

9. In `t3`, use the `mysqladmin kill` command to terminate the process that holds the blocking lock.
10. In `t2`, observe the result of the `INSERT` statement that has been issued in Step 5.
11. In `t3`, terminal window, view the contents of the `/labs/slap-test-updates.sh` file. The script within this file executes the `mysqlslap` load emulation client with options that automatically generate an update load with five concurrent sessions.

12. In t3, execute the `/labs/slap-test-updates.sh` script. Do not wait for it to complete; continue performing the remaining tasks.
13. In t2, display the `PROCESSLIST`.
14. In t2, use `sys.session` to display an alternative view of the running processes.
15. In t2, requery the `sys.innodb_lock_waits` view using the same statement that you executed in step 8. Repeat this query several times.
16. In t3, kill the `slap-test-update.sh` script by pressing `Ctrl + C`.
17. Exit all `mysql` client sessions and close all Linux terminal windows.

Solution 8-2: Identifying the Cause of Slow Performance

Solution Steps

1. Open three Linux terminal windows. This practice refers to these terminal windows as t1, t2, and t3.
2. In t1 and t2, log in to the mysql client and set the prompt to “1> ” and “2> ” respectively. Enter the following command at the t1 Linux terminal prompt and receive the results shown:

```
# mysql -uroot -p
Enter password: oracle
Welcome to the MySQL monitor. Commands end with ; or \g.
...
mysql> PROMPT 1> ;
PROMPT set to '1> '
```

Enter the following command at the t2 Linux terminal prompt and receive the results shown:

```
# mysql -uroot -p
Enter password: oracle
Welcome to the MySQL monitor. Commands end with ; or \g.
...
mysql> PROMPT 2> ;
PROMPT set to '2> '
```

3. In t1, start a new transaction and execute a query for all rows and all columns in the employees.departments table with the FOR UPDATE modifier.

Enter the following statements at the mysql prompt in t1 and receive the results shown:

```
1> USE employees
...
Database changed

1> START TRANSACTION;
Query OK, 0 rows affected (#.## sec)

1> SELECT * FROM departments FOR UPDATE;
+-----+-----+
| dept_no | dept_name |
+-----+-----+
| d009    | Customer Service |
| d005    | Development      |
...
| d008    | Research          |
| d007    | Sales             |
+-----+-----+
9 rows in set (#.## sec)
```

4. In t2, insert a new department called **Distribution** with a **dept_no** of d010.

Enter the following statements at the mysql prompt in t2 and receive the results shown:

```
2> USE employees
...
Database changed
2> INSERT INTO departments
-> VALUES ('d010', 'Distribution');
```

- The statement does not complete because it is waiting for a lock that the transaction in t1 currently holds. After 50 seconds, it times out with the following message:

```
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting
transaction
```

5. In t2, increase the InnoDB lock timeout (system variable **innodb_lock_wait_timeout**) to 3600 seconds (1 hour) and re-execute the **INSERT** statement in the preceding step. Observe it for a minute or two before continuing with the next step.

Enter the following statements at the mysql prompt in t2 and receive the results shown:

```
2> SET innodb_lock_wait_timeout=3600;
Query OK, 0 rows affected (#.## sec)

2> INSERT INTO departments VALUES ('d010','Distribution');
```

- The statement still does not complete.

6. In t1, display the **PROCESSLIST**.

Enter the following statement at the mysql prompt in t1 and receive results similar to those shown:

```
1> SHOW PROCESSLIST\G
*****
1. row ****
...
*****
2. row ****
...
*****
3. row ****
      Id: 2400
      User: root
      Host: localhost
      db: employees
      Command: Query
      Time: 63
      State: update
      Info: insert into departments values ('d010', 'Distribution')
3 rows in set (#.## sec)
```

- The **State** column of the third row of the above output shows that the process is in the **update** state. This might indicate that the statement is blocked, but might also indicate that it is executing normally.

7. In t1, use sys.session to display an alternative view of the running processes.

Enter the following statement at the mysql prompt in t1 and receive the results shown:

```
1> SELECT * FROM sys.session\G
***** 1. row ****
    thd_id: 2436
    conn_id: 2400
        user: root@localhost
        db: employees
    command: Query
    state: update
        time: 150
    current_statement: insert into departments values ('d010',
'Distribution')
    statement_latency: 2.49 m
        progress: NULL
    lock_latency: 142.00 us
    rows_examined: 0
        rows_sent: 0
    rows_affected: 0
        tmp_tables: 0
    tmp_disk_tables: 0
        full_scan: NO
    last_statement: NULL
last_statement_latency: NULL
    current_memory: 61.70 KiB
        last_wait: NULL
    last_wait_latency: NULL
        source: NULL
    trx_latency: 2.49 m
    trx_state: ACTIVE
    trx_autocommit: YES
        pid: 25707
    program_name: mysql
...
3 rows in set (#.## sec)
```

8. In t1, query the sys schema's innodb_lock_waits view by entering the following statement at the mysql prompt. Note the blocking_pid value. This is the process ID of the blocking thread.

Enter the following statement at the mysql prompt in t1 and receive the results shown:

```
1> SELECT waiting trx_id, waiting pid, waiting query,
   -> blocking trx_id, blocking pid, blocking query
   -> FROM sys.innodb lock waits\G
***** 1. row *****
waiting trx_id: 78159
waiting pid: 15
waiting query: insert into departments values ('d010',
'Distribution')
blocking trx_id: 78157
blocking pid: 14
blocking query: SELECT waiting trx_id, wai ... ery FROM
sys.innodb lock waits
1 row in set (#.# sec)
```

- In the preceding output, the blocking_pid value is 14.
 - The sys.innodb_lock_waits view reads information from the Performance Schema data_locks and data_lock_waits tables.
9. In t3, use the mysqladmin kill command to terminate the process that holds the blocking lock.

Enter the following statement at the mysql prompt in t3, substituting the process ID of the blocking transaction that you determined in the preceding step, and receive the results shown:

```
# mysqladmin -uroot -p kill 14
Enter password: oracle
```

- This command terminates the connection with the process ID that you specify, which rolls back any ongoing transaction in that thread.
10. In t2, observe the result of the INSERT statement that has been issued in Step 5.

In the t2 terminal window, the statement completes successfully:

```
2> INSERT INTO departments VALUES ('d010', 'Distribution');
Query OK, 1 row affected (#.# sec)
```

11. In t3, terminal window, view the contents of the `/labs/slap-test-updates.sh` file. The script within this file executes the `mysqlslap` load emulation client with options that automatically generate an update load with five concurrent sessions.
- Enter the following command at the t3 Linux terminal prompt and receive the results shown:

```
# cat /labs/slap-test-updates.sh
#!/bin/bash
mysqlslap --user=root --password=oracle --concurrency=5 \
--iterations=100 --number-char-cols=4 --number-int-cols=7 \
--auto-generate-sql --number-of-queries=10000 \
--auto-generate-sql-load-type=update
```

12. In t3, execute the `/labs/slap-test-updates.sh` script. Do not wait for it to complete; continue performing the remaining tasks.

Enter the following command at the t3 Linux terminal prompt and receive the results shown:

```
# sh /labs/slap-test-updates.sh
Warning: Using a password on the command line interface can be
insecure.
```

13. In t2, display the `PROCESSLIST`.

Enter the following statement at the `mysql` prompt in t2 and receive the results shown:

```
2> SHOW PROCESSLIST\G
...
***** 3. row *****
    Id: 2402
    User: root
    Host: localhost
    db: mysqlslap
...
***** 8. row *****
    Id: 2407
    User: root
    Host: localhost
    db: mysqlslap
Command: Query
    Time: 1
    State: updating
    Info: UPDATE t1 SET intcol1 = 493707480,intcol2 =
935888731,intcol3 = 1311686155,intcol4 = 163013917,intco
8 rows in set (#.## sec)
```

- The number of rows and the values displayed are different on your screen, because they depend on the dynamically generated activity of the `mysqlslap` load emulation client.

14. In t2, use sys.session to display an alternative view of the running processes.

Enter the following statement at the mysql prompt in t2 and receive results similar to those shown:

```
2> SELECT * FROM sys.session\G
*****
1. row ****
    thd_id: 2436
    conn_id: 2400
        user: root@localhost
        db: employees
    command: Query
        state: NULL
        time: 0
current_statement: SELECT * FROM sys.session
statement_latency: 70.58 ms
    progress: NULL
    lock_latency: 2.51 ms
rows_examined: 15798
    rows_sent: 0
    rows_affected: 0
        tmp_tables: 4
    tmp_disk_tables: 2
        full_scan: YES
    last_statement: NULL
last_statement_latency: NULL
    current_memory: 1.00 MiB
        last_wait: NULL
    last_wait_latency: NULL
        source: NULL
    trx_latency: 68.13 ms
    trx_state: ACTIVE
    trx_autocommit: YES
        pid: 25707
    program_name: mysql
*****
2. row ****
    thd_id: 2438
    conn_id: 2402
        user: root@localhost
        db: mysqlslap
    command: Query
        state: query end
        time: 0
current_statement: INSERT INTO t1 VALUES (1880074 ...
AF3FKMtKwCtCBdrTJKwuvLhKm')
statement_latency: 1.88 ms
    progress: NULL
    lock_latency: 60.00 us
rows_examined: 0
    rows_sent: 0
    rows_affected: 0
```

```
        tmp_tables: 0
        tmp_disk_tables: 0
            full_scan: NO
        last_statement: NULL
last_statement_latency: NULL
        current_memory: 93.78 KiB
            last_wait: NULL
last_wait_latency: NULL
            source: NULL
        trx_latency: 1.83 ms
        trx_state: ACTIVE
        trx_autocommit: YES
            pid: 25740
        program_name: mysqlslap
*****
      3. row *****
        thd_id: 43
        conn_id: 4
            user: sql/event_scheduler
            db: NULL
        command: Sleep
            state: Waiting on empty queue
            time: NULL
        current_statement: NULL
        statement_latency: NULL
            progress: NULL
        lock_latency: NULL
        rows_examined: NULL
            rows_sent: NULL
        rows_affected: NULL
            tmp_tables: NULL
        tmp_disk_tables: NULL
            full_scan: NO
        last_statement: NULL
last_statement_latency: NULL
        current_memory: 16.18 KiB
            last_wait: NULL
last_wait_latency: NULL
            source: NULL
        trx_latency: NULL
        trx_state: NULL
        trx_autocommit: NULL
            pid: NULL
        program_name: NULL
3 rows in set (#.# sec)
```

15. In t2, requery the `sys.innodb_lock_waits` view using the same statement that you executed in step 8. Repeat this query several times.

Enter the following statements at the `mysql` prompt in t1 several times and receive the results shown:

```

2> SELECT waiting trx_id, waiting pid, waiting query,
   -> blocking trx_id, blocking pid, blocking query
   -> FROM sys.innodb lock waits\G
***** 1. row *****
waiting trx_id: 186726
waiting pid: 2443
waiting query: UPDATE t1 SET intcoll = 489319 ...
GWQmhKxeRC550RSNWGbLceYsEhiao'
blocking trx_id: 186725
blocking pid: 2444
blocking query: UPDATE t1 SET intcoll = 489319 ...
GWQmhKxeRC550RSNWGbLceYsEhiao'
1 row in set (#.## sec)

2> SELECT waiting trx_id, waiting pid, waiting query,
   -> blocking trx_id, blocking pid, blocking query
   -> FROM sys.innodb lock waits\G
***** 1. row *****
waiting trx_id: 187740
waiting pid: 2451
waiting query: UPDATE t1 SET intcoll = 178150 ...
1rQsNlhwdPTaKC84PhtMDGNPvdHN5'
blocking trx_id: 187739
blocking pid: 2452
blocking query: UPDATE t1 SET intcoll = 178150 ...
1rQsNlhwdPTaKC84PhtMDGNPvdHN5'
1 row in set (#.## sec)

2> SELECT waiting trx_id, waiting pid, waiting query,
   -> blocking trx_id, blocking pid, blocking query
   -> FROM sys.innodb lock waits\G
***** 1. row *****
waiting trx_id: 221088
waiting pid: 2463
waiting query: UPDATE t1 SET intcoll = 591450 ...
5W9OE6eIFLcYJPv8o0FDmaaoZPgxr'
blocking trx_id: 221086
blocking pid: 2460

```

```
blocking_query: UPDATE t1 SET intcol1 = 591450 ...
5W9OE6eIFLcYJPv8o0FDmaaoZPgxr'
1 row in set (#.## sec)
```

- The preceding output shows details of the UPDATE statements that other threads are blocking.

16. In t3, kill the `slap-test-update.sh` script by pressing Ctrl + C.
17. Exit all mysql client sessions and close all Linux terminal windows.

Practices for Lesson 9: Optimizing Query Performance

Practices for Lesson 9: Overview

Overview

In these practices, you will identify poorly performing queries and then optimize them.

Assumptions

- You have created and populated the `employees` database in the practices for the lesson titled “Monitoring MySQL.”
- You have installed the MySQL Enterprise Monitor in the practices for the lesson titled “Monitoring MySQL.”

Practice 9-1: Improving Query Performance with Indexes

Overview

In this practice, you create indexes to improve the performance of specific queries, and measure the effects of creating those indexes.

Duration

This practice should take you approximately 20 minutes to complete.

Tasks

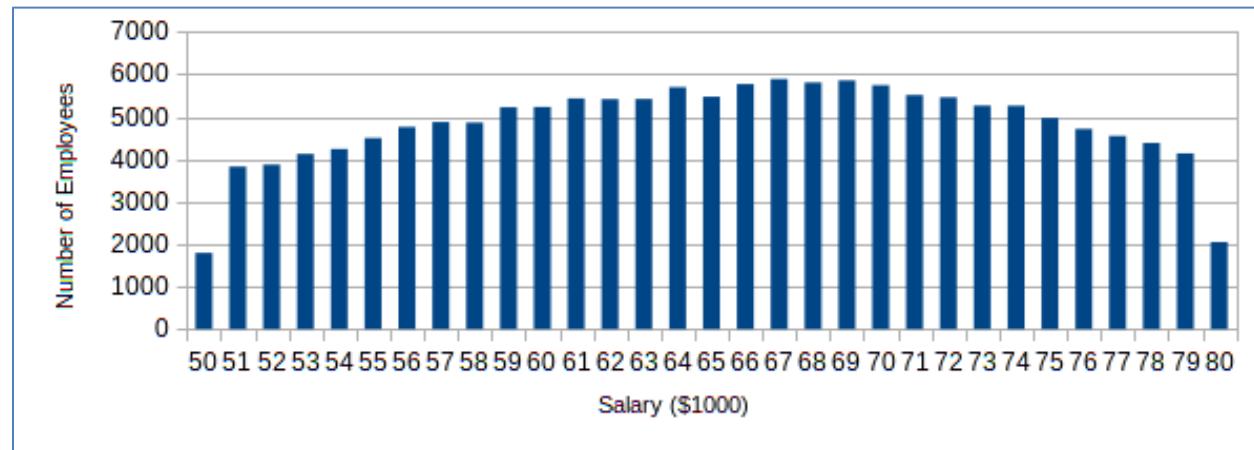
1. Execute the following query in the `employees` database. Note how long it takes.

```
SELECT emp_no, title FROM titles  
WHERE title='Manager' AND to_date > NOW();
```

2. Execute the `EXPLAIN` command on the previous statement. Note the conclusions that you reach about the optimizer's index choice and how many rows it must examine to execute the statement.
3. Create an index on the `title` column.
4. Execute the query in step 1 again and note how long it takes.
5. Execute the `EXPLAIN` command on the statement from step 1 again and note any conclusions you have about the new index.
6. Add another index, this one on the `title` and `to_date` columns, in that order.
7. Execute the `EXPLAIN` command on the statement from step 1 again and note any differences from the previous optimizer output.
8. Drop both of the indexes that you created in the preceding steps.
9. Execute the following query, and note how long it takes:

```
SELECT COUNT(*), ROUND(salary, -3) AS `base`  
FROM salaries  
WHERE salary BETWEEN 50000 AND 80000  
AND to_date > NOW()  
GROUP BY base  
ORDER BY base;
```

- The query generates a histogram of employees within each \$1,000 salary band between \$50,000 and \$80,000. The following chart is a visualization of this data:



10. Execute `EXPLAIN` on the preceding statement. Note the conclusions that you reach about the optimizer's index choice and how many rows it must examine to execute the statement.
11. Add an index on the `salary` column, which is one of the columns used in the `WHERE` clause.
12. Execute the query from step 9 again. Note how long it takes.
13. Execute the `EXPLAIN` command on the statement again and note any conclusions you have about the new index.
14. Create an index on the `salary` and `to_date` columns, in that order.
15. Execute the query again. Note how long it takes.
16. Execute the `EXPLAIN` command on the statement again and note any conclusions you have about the new index.
17. Create an index on the `to_date` and `salary` columns, in that order.
18. Execute the query again. Note how long it takes.
19. Execute the `EXPLAIN` command on the statement again and note any differences from the previous optimizer output.
20. Drop all of the indexes that you created in this activity.
21. Exit the `mysql` command-line client and keep the Linux terminal window open for the next practice.

Solution 9-1: Improving Query Performance with Indexes

Solution Steps

1. Execute the following query in the employees database. Note how long it takes.

```
SELECT emp_no, title FROM titles  
WHERE title='Manager' AND to_date > NOW();
```

Enter the following command at a Linux terminal prompt and receive the results shown:

```
# mysql -uroot -p  
Enter password: oracle  
Welcome to the MySQL monitor. Commands end with ; or \g.  
...  
mysql>
```

Enter the following statements at the mysql prompt and receive the results shown:

```
mysql> USE employees  
...  
Database changed  
mysql> SELECT emp_no, title FROM titles  
-> WHERE title='Manager' AND to_date > NOW();  
+-----+-----+  
| emp_no | title |  
+-----+-----+  
| 110039 | Manager |  
| 110114 | Manager |  
| 110228 | Manager |  
| 110420 | Manager |  
| 110567 | Manager |  
| 110854 | Manager |  
| 111133 | Manager |  
| 111534 | Manager |  
| 111939 | Manager |  
+-----+  
9 rows in set (0.44 sec)
```

- In the preceding output, the statement executes in 0.44 seconds. Your output might be different.

2. Execute the EXPLAIN command on the previous statement. Note the conclusions that you reach about the optimizer's index choice and how many rows it must examine to execute the statement.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> EXPLAIN SELECT emp_no, title FROM titles
   -> WHERE title='Manager' AND to_date > NOW() \G
***** 1. row *****
      id: 1
  select_type: SIMPLE
        table: titles
    partitions: NULL
       type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
         ref: NULL
        rows: 431190
  filtered: 3.33
     Extra: Using where
1 row in set, 1 warning (#.# sec)
```

- The optimizer does not recognize any possible indexes (`keys`), and therefore, it must examine every row in the table. The value in the `rows` column approximates the actual number of rows in the `titles` table, based on InnoDB's index statistics. The column value `type: ALL` indicates that the optimizer must perform a full table scan.

3. Create an index on the `title` column.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> CREATE INDEX titles_title ON titles(title);
Query OK, 0 rows affected (#.# sec)
Records: 0  Duplicates: 0  Warnings: 0
```

4. Execute the query in step 1 again and note how long it takes.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> SELECT emp_no, title FROM titles
      -> WHERE title='Manager' AND to_date > NOW();
+-----+-----+
| emp_no | title   |
+-----+-----+
| 110039 | Manager |
...
| 111939 | Manager |
+-----+-----+
9 rows in set (0.00 sec)
```

- In the preceding output, the statement executes in less than 0.01 seconds, which shows that it is considerably faster than before.
5. Execute the EXPLAIN command on the statement from step 1 again and note any conclusions you have about the new index.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> EXPLAIN SELECT emp_no, title FROM titles
      -> WHERE title='Manager' AND to_date > NOW()\G
***** 1. row *****
      id: 1
  select_type: SIMPLE
        table: titles
    partitions: NULL
       type: ref
possible_keys: titles_title
         key: titles_title
      key_len: 202
        ref: const
       rows: 24
  filtered: 33.33
     Extra: Using where
1 row in set, 1 warning (0.00 sec)
```

- The optimizer uses the new index. The optimizer estimates that it must examine only 24 rows based on a `ref` type query against a `const` value (`Manager`), and filter approximately a third of those rows (9) with the remainder of the `WHERE` clause condition.

6. Add another index, this one on the `title` and `to_date` columns, in that order.

Enter the following statement at the `mysql` prompt and receive the results shown:

```
mysql> CREATE INDEX titles_title_date ON titles(title, to_date);
Query OK, 0 rows affected (#.## sec)
Records: 0  Duplicates: 0  Warnings: 0
```

7. Execute the `EXPLAIN` command on the statement from step 1 again and note any differences from the previous optimizer output.

Enter the following statement at the `mysql` prompt and receive the results shown:

```
mysql> EXPLAIN SELECT emp_no, title FROM titles
      -> WHERE title='Manager' AND to_date > NOW()\G
*****
   id: 1
  select_type: SIMPLE
        table: titles
    partitions: NULL
       type: range
possible_keys: titles_title,titles_title_date
      key: titles_title_date
     key_len: 206
       ref: NULL
      rows: 9
 filtered: 100.00
    Extra: Using where; Using index
1 row in set, 1 warning (#.## sec)
```

- The optimizer considers both of the new indexes, and chooses the `titles_title_date` index. It must examine only nine rows, and does not require any additional filtering. It uses a `range` type query because it cannot compare indexed values for equality when you use an inequality comparison such as greater-than (the `>` operator).
 - Using `index` in the `extra` column indicates that the server does not need to read the data rows as all the required data can be obtained from the index.
8. Drop both of the indexes that you created in the preceding steps.

Enter the following statements at the `mysql` prompt and receive the results shown:

```
mysql> DROP INDEX titles_title ON titles;
Query OK, 0 rows affected (#.## sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> DROP INDEX titles_title_date ON titles;
Query OK, 0 rows affected (#.## sec)
Records: 0  Duplicates: 0  Warnings: 0
```

9. Execute the following query, and note how long it takes:

```
SELECT COUNT(*), ROUND(salary, -3) AS `base`
FROM salaries
WHERE salary BETWEEN 50000 AND 80000
AND to_date > NOW()
GROUP BY base
ORDER BY base;
```

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> SELECT COUNT(*), ROUND(salary, -3) AS `base` 
-> FROM salaries
-> WHERE salary BETWEEN 50000 AND 80000
-> AND to_date > NOW()
-> GROUP BY base
-> ORDER BY base;
+-----+-----+
| COUNT(*) | base   |
+-----+-----+
|      1791 | 50000  |
|      3826 | 51000  |
| ...      | ...    |
|      4136 | 79000  |
|      2046 | 80000  |
+-----+-----+
31 rows in set (1.67 sec)
```

- This query in the preceding output takes 1.67 seconds. The duration might be different in your output.
- The query generates a histogram of employees within each \$1,000 salary band between \$50,000 and \$80,000. The following chart is a visualization of this data.



10. Execute EXPLAIN on the preceding statement. Note the conclusions that you reach about the optimizer's index choice and how many rows it must examine to execute the statement.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> EXPLAIN SELECT COUNT(*), ROUND(salary, -3) AS `base`  
-> FROM salaries WHERE salary BETWEEN 50000 AND 80000  
-> AND to_date > NOW() GROUP BY base ORDER BY base\G  
***** 1. row *****  
      id: 1  
select_type: SIMPLE  
      table: salaries  
    partitions: NULL  
        type: ALL  
possible_keys: NULL  
          key: NULL  
    key_len: NULL  
      ref: NULL  
rows: 2838566  
filtered: 3.70  
Extra: Using where; Using temporary; Using filesort  
1 row in set, 1 warning (#.## sec)
```

- The optimizer cannot use any indexes when it runs this query, and must examine nearly three million rows in a full table scan.
11. Add an index on the salary column, which is one of the columns used in the WHERE clause.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> CREATE INDEX salary_value  
-> ON salaries(salary);  
Query OK, 0 rows affected (#.## sec)  
Records: 0  Duplicates: 0  Warnings: 0
```

12. Execute the query from step 9 again. Note how long it takes.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> SELECT COUNT(*), ROUND(salary, -3) AS `base`
-> FROM salaries
-> WHERE salary BETWEEN 50000 AND 80000
-> AND to_date > NOW()
-> GROUP BY base
-> ORDER BY base;
+-----+-----+
| COUNT(*) | base   |
+-----+-----+
|      1791 | 50000 |
|      3826 | 51000 |
...
|      2046 | 80000 |
+-----+-----+
31 rows in set (0.89 sec)
```

- This query in the preceding output takes 0.89 seconds, which is an improvement, but is still not optimal.

13. Execute the EXPLAIN command on the statement again and note any conclusions you have about the new index.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> EXPLAIN SELECT COUNT(*), ROUND(salary, -3) AS `base`
-> FROM salaries WHERE salary BETWEEN 50000 AND 80000
-> AND to_date > NOW() GROUP BY base ORDER BY base\G
***** 1. row *****
      id: 1
  select_type: SIMPLE
        table: salaries
    partitions: NULL
       type: ALL
possible_keys: salary_value
         key: NULL
    key_len: NULL
        ref: NULL
       rows: 2838912
  filtered: 16.66
     Extra: Using where; Using temporary; Using filesort
1 row in set, 1 warning (0.00 sec)
```

- The optimizer considers the salary_value index but does not use it. It chooses to perform a full table scan. This indicates that the index on the salary column is not useful for this particular query.

14. Create an index on the salary and to_date columns, in that order.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> CREATE INDEX salary_value_date
      -> ON salaries(salary, to_date);
Query OK, 0 rows affected (#.## sec)
Records: 0  Duplicates: 0  Warnings: 0
```

- The WHERE clause in the statement under consideration uses both the salary and to_date values, so the optimizer should be able to consider this index.

15. Execute the query again. Note how long it takes.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> SELECT COUNT(*), ROUND(salary, -3) AS `base`
      -> FROM salaries
      -> WHERE salary BETWEEN 50000 AND 80000
      -> AND to_date > NOW()
      -> GROUP BY base
      -> ORDER BY base;
+-----+-----+
| COUNT(*) | base   |
+-----+-----+
|      1791 | 50000 |
|      3826 | 51000 |
...
|      4136 | 79000 |
|      2046 | 80000 |
+-----+-----+
31 rows in set (0.44 sec)
```

- This query in the preceding output takes 0.44 seconds, which is somewhat quicker than the previous executions.

16. Execute the EXPLAIN command on the statement again and note any conclusions you have about the new index.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> EXPLAIN SELECT COUNT(*), ROUND(salary, -3) AS `base`
-> FROM salaries WHERE salary BETWEEN 50000 AND 80000
-> AND to_date > NOW() GROUP BY base ORDER BY base\G
*****
1. row ****
      id: 1
  select_type: SIMPLE
        table: salaries
    partitions: NULL
        type: range
possible_keys: salary_value,salary_value_date
      key: salary_value_date
    key_len: 7
        ref: NULL
       rows: 1419283
  filtered: 33.33
      Extra: Using where; Using index; Using temporary; Using
        filesort
1 row in set, 1 warning (#.## sec)
```

- The optimizer considers both of the new indexes and chooses the salary_value_date index. Rather than perform a table scan (as in the previous EXPLAIN output), it now performs a range type query, and must examine only 1.5 million rows, filtering out a third of those.

17. Create an index on the to_date and salary columns, in that order.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> CREATE INDEX salary_date_value
-> ON salaries(to_date, salary);
Query OK, 0 rows affected (#.## sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Note: This index contains the same columns as the index that you created in step 14, but in a different order.

18. Execute the query again. Note how long it takes.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> SELECT COUNT(*), ROUND(salary, -3) AS `base`
-> FROM salaries
-> WHERE salary BETWEEN 50000 AND 80000
-> AND to_date > NOW()
-> GROUP BY base
-> ORDER BY base;
+-----+-----+
| COUNT(*) | base   |
+-----+-----+
|      1791 | 50000 |
|      3826 | 51000 |
...
|      4136 | 79000 |
|     2046 | 80000 |
+-----+-----+
31 rows in set (0.10 sec)
```

- This time, the query is considerably faster.

19. Execute the EXPLAIN command on the statement again and note any differences from the previous optimizer output.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> EXPLAIN SELECT COUNT(*), ROUND(salary, -3) AS `base`
-> FROM salaries WHERE salary BETWEEN 50000 AND 80000
-> AND to_date > NOW() GROUP BY base ORDER BY base \G
***** 1. row *****
      id: 1
  select_type: SIMPLE
        table: salaries
    partitions: NULL
       type: range
possible_keys:salary_value,salary_value_date,salary_date_value
      key: salary_date_value
    key_len: 3
        ref: NULL
       rows: 444440
  filtered: 50.00
    Extra: Using where; Using index; Using temporary; Using
          filesort
1 row in set, 1 warning (#.# sec)
```

- The optimizer considers all three of the new indexes and chooses the salary_date_value index. It must examine only approximately 440,000 rows, which reduces the amount of additional filtering required to complete the query, and therefore, improves its overall performance.
 - The effect of the different column order in the index shows that index creation is part art and part science, and that you must always evaluate the results of any change.
20. Drop all of the indexes that you created in this activity.

Enter the following statements at the mysql prompt and receive the results shown:

```
mysql> DROP INDEX salary_value ON salaries;
Query OK, 0 rows affected (#.## sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> DROP INDEX salary_value_date ON salaries;
Query OK, 0 rows affected (#.## sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> DROP INDEX salary_date_value ON salaries;
Query OK, 0 rows affected (#.## sec)
Records: 0  Duplicates: 0  Warnings: 0
```

21. Exit the mysql command-line client and keep the Linux terminal window open for the next practice.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> EXIT
Bye
#
```

Practice 9-2: Using MySQL Enterprise Monitor Query Analyzer

Overview

In this practice, you execute a poorly performing query and use the information provided by MySQL Enterprise Monitor Query Analyzer to identify and improve it.

Duration

This practice should take you approximately 15 minutes to complete.

Tasks

1. Check the status of the MySQL Enterprise Monitor service and start it if it is not running, by entering the following commands at a new Linux terminal prompt and receive the results shown:

```
# sh /opt/mysql/enterprise/monitor/mysqlmonitorctl.sh status
MySQL Enterprise MySQL is not running
MySQL Enterprise Tomcat is not running

# sh /opt/mysql/enterprise/monitor/mysqlmonitorctl.sh start
Starting mysql service [ OK ]
date-and-time mysqld_safe Logging to
'/opt/mysql/enterprise/monitor/mysql/runtime/mysqld.log'.
date-and-time mysqld_safe Starting mysqld daemon with databases
from /opt/mysql/enterprise/monitor/mysql/data/
Starting tomcat service [ OK ]

# sh /opt/mysql/enterprise/monitor/mysqlmonitorctl.sh status
MySQL Enterprise MySQL is running
MySQL Enterprise Tomcat is running
```

2. Open the MySQL Enterprise Monitor dashboard in the Firefox web browser by using the bookmark you created for it in the practices for the lesson titled “Monitoring MySQL.” Log in as the manager user `memmanager`, with the password `oracle`.
Note: It might take a minute or two for the dashboard to load after you start the MySQL Enterprise Monitor Manager service.
3. From the left navigation menu, select “Queries” to open the Query Analyzer.
4. In the Refresh drop-down list in the top-right corner of the page, change the refresh interval



to every 30 seconds.

5. Open a new Linux terminal window and log in to the mysql command-line client.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# mysql -uroot -p  
Enter password: oracle  
Welcome to the MySQL monitor. Commands end with ; or \g.  
...  
mysql>
```

6. Change the current database to employees.

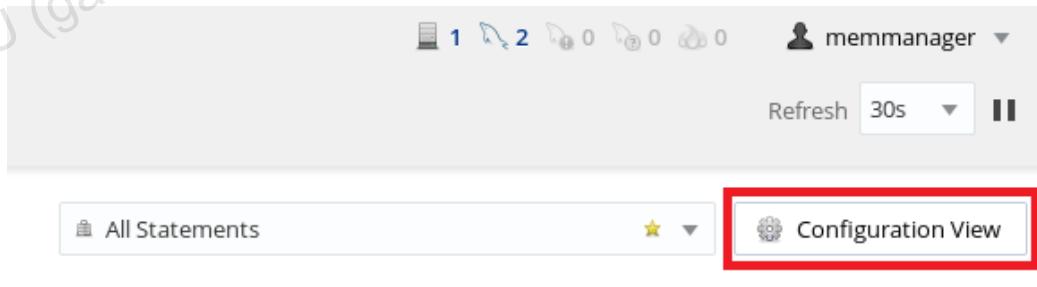
Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> USE employees  
Reading table information ...  
Database changed
```

7. In the terminal window containing the mysql command-line client prompt, issue the following statement several times against the employees database:

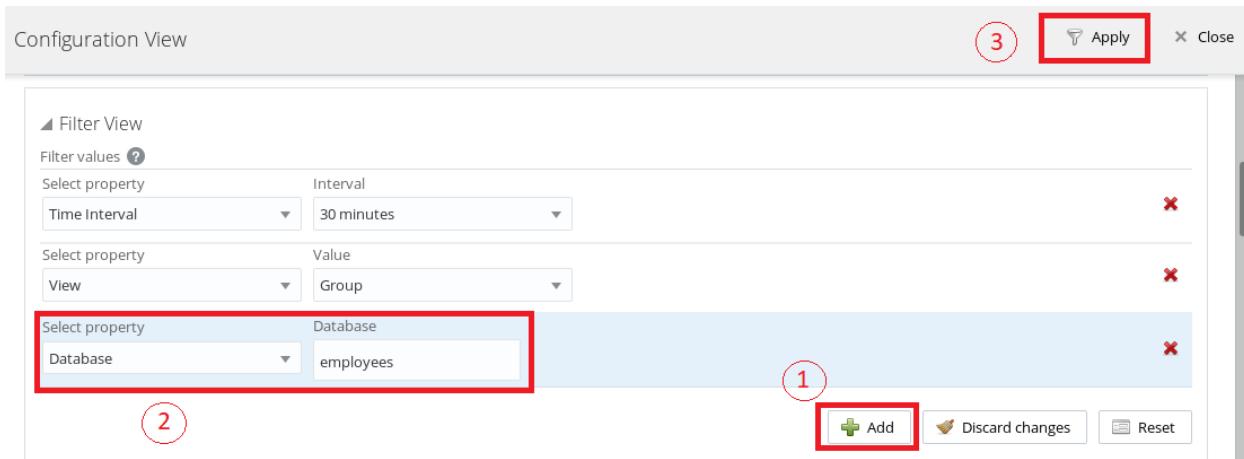
```
SELECT COUNT(*), ROUND(salary, -3) AS `base`  
FROM salaries  
WHERE salary BETWEEN 50000 AND 80000  
AND to_date > NOW()  
GROUP BY base  
ORDER BY base;
```

8. Return to the MySQL Enterprise Monitor Query Analyzer page.

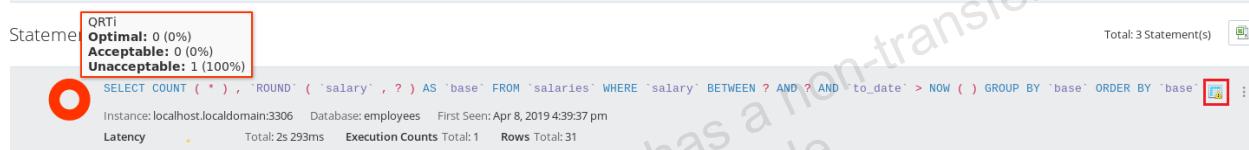


9. Click the "Configuration View" button:

10. Scroll down to the "Filter View" section and click the "Add" button. Click the "Select property" pull-down box and select "Database." Enter the database name employees in the Database text box. Then, click the "Apply" button on the top right. This limits the output of the Query Analyzer to show only those queries that affect the employees database.



- Wait until the Query Analyzer page refreshes and includes the query that you executed in step 7. This can take several minutes.



The Query Response Time Index pie chart should highlight the query response time in either red or yellow, which means that the response time for this query is either unacceptable or acceptable, but not optimal (green). In either case, you can improve the performance of this query.

Notes

- The response time of the query depends on a number of variables, including the hardware specification of your system.
- Allow sufficient time for the updated query results to appear; the page might refresh one or more times before the updated query metrics are available. Use the value in the column that shows the number of executions as an indicator of when the query metrics have been updated.
- An icon on the right of the statement shows that the query performs a full-table scan and is, therefore, a good candidate for optimization.

- Click the query in the statement list. The query details page is displayed. This shows the normalized query and execution metrics. Note that, from the number of rows examined in the “Statistics” section on the right, the query requires a full table scan and, therefore, would probably benefit from using an index.

Read the contents of the “Example Queries” section at the bottom of the details page.

When enabled, it displays more information about the most expensive query executed.

Note that, in order to enable this behavior, you must enable the

`events_statements_history_long` consumer in Performance Schema. The tab contents describe the `UPDATE` statement you must execute to achieve this. Execute the

statement provided at the mysql command-line prompt.

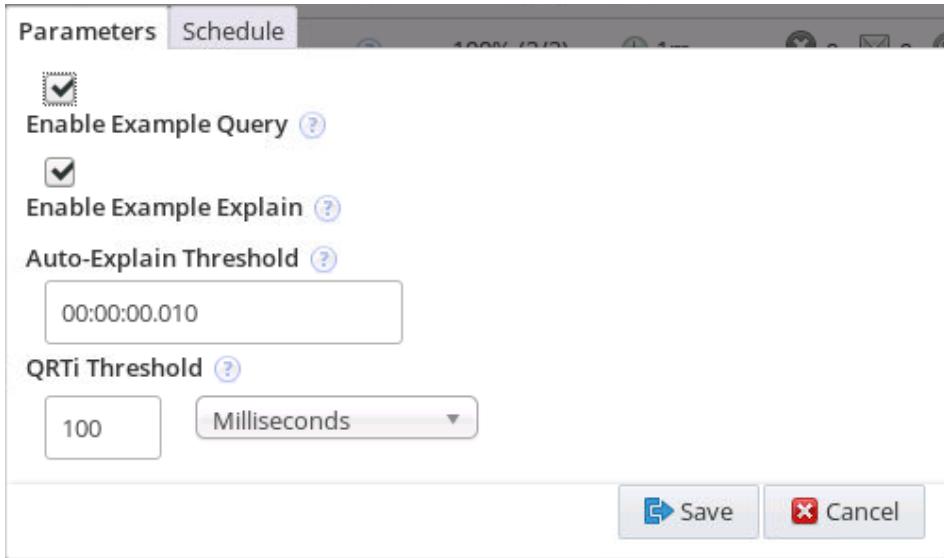
Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> UPDATE performance_schema.setup_consumers
->     SET enabled = 'YES'
-> WHERE name = 'events_statements_history_long';
Query OK, 1 row affected (#.## sec)
```

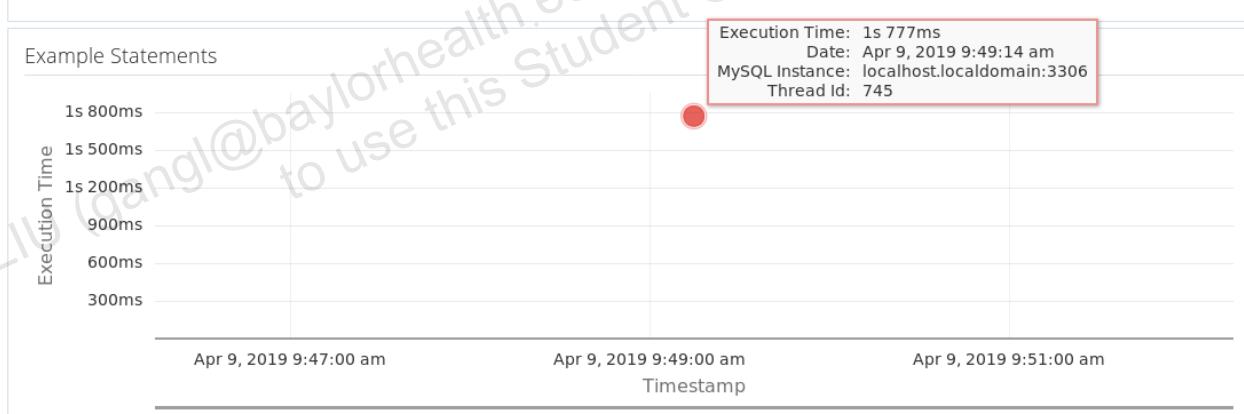
13. You must also configure MySQL Enterprise Monitor to collect this information. Click the “Configure Query Analysis” link. The list of advisors appears.
14. Check the Query Analysis Reporting advisor, and then click the Edit Selected button:

Item	Info	Coverage	Schedule	Event Handling	Parameters
Average Statement Execution Time Advisor	100% (2/2)	1m	<input checked="" type="checkbox"/>	0 <input checked="" type="checkbox"/> 0 <input checked="" type="checkbox"/>	Average Execution Time Thresholds: 500ms 1000ms 5000ms Minimum
Query Analysis Reporting	100% (2/2)	1m	<input checked="" type="checkbox"/>	0 <input checked="" type="checkbox"/> 0 <input checked="" type="checkbox"/>	Auto-Explain Threshold: 00:00:00.100 QRTi Threshold: 100ms
Query Pileup Advisor	100% (2/2)	1m	<input checked="" type="checkbox"/>	0 <input checked="" type="checkbox"/> 0 <input checked="" type="checkbox"/>	Window Size: 2m Growth Rate Thresholds: 20% 50% 80% Minimum
SQL Statement Generates Warnings or Errors	100% (2/2)	1m	<input checked="" type="checkbox"/>	0 <input checked="" type="checkbox"/> 0 <input checked="" type="checkbox"/>	Notification level when discovering queries with errors: CRITICAL Notification

15. On the Parameters tab of the dialog box that is displayed, ensure that the following options are set:
 - *Enable Example Query:* Selected
 - *Enable Example Explain:* Selected
 - *Auto-Explain Threshold:* Change to 00:00:00.010
 - *QRTi Threshold:* No change, remain as 100 milliseconds



16. Click the Save button.
17. Click “Queries” in the left navigation menu to return to the Query Analyzer.
18. At the mysql prompt, execute the query from step 7 again several times.
19. When the Query Analyzer display refreshes, click the query you executed in the preceding step and scroll down to the bottom. The graph shows the executions of this query. Place the cursor on a dot to view a summary of the execution.



20. Click a dot in the graph to show the full example statement and explain plan. Note that the possible_keys field contains the value NULL, which means that the query cannot use an index.

<p>Example SQL Statement</p> <pre>SELECT COUNT(*), ROUND(salary, -3) AS `base` FROM salaries WHERE salary BETWEEN 50000 AND 80000 AND to_date > NOW() GROUP BY base ORDER BY base</pre>	<p>Example Details</p> <table border="0"> <tr> <td>Date:</td> <td>Apr 9, 2019 9:49:14 am</td> </tr> <tr> <td>Execution Time:</td> <td>1s 777ms</td> </tr> <tr> <td>Thread ID:</td> <td>745</td> </tr> <tr> <td>From Host:</td> <td>localhost</td> </tr> <tr> <td>User:</td> <td>root</td> </tr> <tr> <td>MySQL Instance:</td> <td>localhost.localdomain:3306</td> </tr> </table>	Date:	Apr 9, 2019 9:49:14 am	Execution Time:	1s 777ms	Thread ID:	745	From Host:	localhost	User:	root	MySQL Instance:	localhost.localdomain:3306								
Date:	Apr 9, 2019 9:49:14 am																				
Execution Time:	1s 777ms																				
Thread ID:	745																				
From Host:	localhost																				
User:	root																				
MySQL Instance:	localhost.localdomain:3306																				
<p>Example EXPLAIN</p> <table border="1"> <thead> <tr> <th>ID</th> <th>select_type</th> <th>table</th> <th>type</th> <th>possible_keys</th> <th>key</th> <th>key_len</th> <th>ref</th> <th>rows</th> <th>extra</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>SIMPLE</td> <td>salaries</td> <td>ALL</td> <td></td> <td></td> <td>0</td> <td></td> <td>2838426</td> <td>Using where; Using temporary; Using filesort</td> </tr> </tbody> </table>	ID	select_type	table	type	possible_keys	key	key_len	ref	rows	extra	1	SIMPLE	salaries	ALL			0		2838426	Using where; Using temporary; Using filesort	
ID	select_type	table	type	possible_keys	key	key_len	ref	rows	extra												
1	SIMPLE	salaries	ALL			0		2838426	Using where; Using temporary; Using filesort												

Query Analyzer > Details

21. Scroll up and click the “Query Analyzer” link at the top to view the list of statements.
 22. Create the following indexes on the salaries table:
- Index name: salary_value, column to index: salary
 - Index name: salary_value_date, columns to index: salary, to_date, in that order
 - Index name: salary_date_value, columns to index: to_date, salary, in that order

Enter the following statements at a mysql prompt and receive the results shown:

```
mysql> CREATE INDEX salary_value ON salaries(salary);
Query OK, 0 rows affected (##.## sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> CREATE INDEX salary_value_date
      -> ON salaries(salary, to_date);
Query OK, 0 rows affected (##.## sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> CREATE INDEX salary_date_value
      -> ON salaries(to_date, salary);
Query OK, 0 rows affected (##.## sec)
Records: 0  Duplicates: 0  Warnings: 0
```

23. Re-execute the query from step 7 several times.

24. When the Query Analyzer display refreshes, locate the query in the statement list, which should now be displayed as partial green or orange in the QRTi pie chart.



25. At a mysql prompt, execute an EXPLAIN statement on the query from step 7 and receive the results shown in the following:

```
mysql> EXPLAIN SELECT COUNT(*), ROUND(salary, -3) AS `base`
   -> FROM salaries
   -> WHERE salary BETWEEN 50000 AND 80000
   -> AND to_date > NOW()
   -> GROUP BY base
   -> ORDER BY base \G
*****
1. row *****
      id: 1
  select_type: SIMPLE
        table: salaries
    partitions: NULL
        type: range
possible_keys: salary_value,salary_value_date,salary_date_value
      key: salary_value_date
     key_len: 7
        ref: NULL
       rows: 1419213
  filtered: 33.33
     Extra: Using where; Using index; Using temporary; Using
           filesort
1 row in set, 1 warning (#.# sec)
```

- Note that the optimizer now uses the `salary_value_date` index you created on the `salary` and `to_date` columns. The number of rows that must be read is considerably less than before.
- This information might not be available in the Query Analyzer interface, because the query execution time might be less than that configured in the Query Analysis Report advisor (Auto-Explain Threshold).

26. Drop the indexes you created in step 22 from the salaries table.

Enter the following statements at the mysql prompt and receive the results shown:

```
mysql> DROP INDEX salary_value ON salaries;
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> DROP INDEX salary_date_value ON salaries;
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> DROP INDEX salary_value_date ON salaries;
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

27. Exit the mysql command-line client session.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> EXIT
Bye
```

28. Close the Firefox browser window.

29. Stop the MySQL Enterprise Monitor service by entering the following commands at a Linux terminal prompt and receive the results shown:

```
# sh /opt/mysql/enterprise/monitor/mysqlmonitorctl.sh stop
Stopping tomcat service . [ OK ]
Stopping mysql service .2017-01-24T10:08:48.449141Z mysqld_safe
mysqld from pid file
/opt/mysql/enterprise/monitor/mysql/runtime/mysqld.pid ended
. [ OK ]
# sh /opt/mysql/enterprise/monitor/mysqlmonitorctl.sh status
MySQL Enterprise MySQL is not running
MySQL Enterprise Tomcat is not running
```

30. Close the Linux terminal window.

Solution 9-2: Using MySQL Enterprise Monitor Query Analyzer

There are no solution steps for this practice. Follow the task instructions.

GANG LIU (gangli@baylorhealth.edu) has a non-transferable license
to use this Student Guide.

Practices for Lesson 10: Choosing a Backup Strategy

Practices for Lesson 10: Overview

Overview

In these practices, you test your knowledge of the different types of backup and the factors you must consider when implementing a backup strategy.

Practice 10-1: Quiz – Backup Strategies

Overview

In this quiz, you answer questions about the different types of backup and how to implement a backup strategy.

Duration

This practice should take you approximately 10 minutes to complete.

Tasks

1. Which type of backup allows client applications to read, but not write, data?
 - a. Hot
 - b. Warm
 - c. Cold
 - d. Logical
2. Which of the following strategies perform “hot” backup only? (Select all that apply.)
 - a. SQL statements
 - b. OS copy commands
 - c. Snapshots
 - d. Replication
3. Which of the following are characteristics of logical backups? (Select all that apply.)
 - a. Are based on a binary representation of the data
 - b. Consist of a series of SQL statements that you can execute to restore the current state of the database
 - c. Are faster to restore than physical backups
 - d. Can be used on both local and remote servers
4. Which of the following are logical backup methods? (Select all that apply.)
 - a. SQL statements
 - b. Snapshots
 - c. mysqldump and mysqlpump utilities
 - d. MySQL Enterprise Backup
5. Which of the following are characteristics of physical backups? (Select all that apply.)
 - a. Are based on SQL statements that re-create the database
 - b. Are faster to restore than logical backups
 - c. Can be restored to servers with different architectures
 - d. Are always “cold” backups
6. Which of the following techniques can be used for physical backups? (Select all that apply.)
 - a. OS copy commands
 - b. Binary logging
 - c. MySQL Enterprise Backup
 - d. Replication

7. Which of the following are good reasons for creating a backup based on a snapshot of the data? (Select all that apply.)
 - a. To capture the state of the database at a particular point in time
 - b. To start up restored database faster by avoiding InnoDB auto-recovery
 - c. To minimize down time
 - d. To ensure a consistent view of the database
8. Which of the following are good reasons for using the binary log for backing up and restoring data? (Select all that apply.)
 - a. You can use binary logs to restore transactions that occurred since the last full backup.
 - b. You can identify specific transactions that caused damage to database integrity and skip them during the restore process.
 - c. You can restore from multiple binary log backups in any order because the server uses transaction identifiers to automatically determine the order in which transactions must be applied.
 - d. You can use binary logs to restore the system to a known state faster than any other backup technique.
9. When considering a backup strategy where the following factors apply, what is the most sensible option?
 - Down time is not an issue.
 - Small amounts of data (less than 2 GB)
 - A very limited budget for software, hardware, and network upgrades
 - a. Snapshots
 - b. MySQL Enterprise Backup
 - c. mysqldump or mysqlpump utilities
 - d. Cold (raw) backup
10. When considering a backup strategy where the following factors apply, what options are available? (Select all that apply.)
 - Zero down time
 - Large (and growing) amounts of data (greater than 1 TB)
 - a. Replication
 - b. Snapshots
 - c. MySQL Enterprise Backup
 - d. mysqldump or mysqlpump utilities

Solution 10-1: Quiz – Backup Strategies

Answers to Quiz Questions

1. **b.** Warm backups permit applications to read from the database, but not write to it.
2. **c.** SQL statements are “warm”; OS copy commands can be either “warm” or “cold.” Replication is not sufficient as a backup strategy. It can help protect against hardware failure on the master server, but does not protect against data loss because it can propagate statements that delete data to slave servers. See the lessons titled “Configuring a Replication Topology” and “Administering a Replication Topology” for more details on replication.
3. **b** and **d**.
4. **a** and **c**. The other techniques listed use physical backups.
5. **b** and **c**.
6. **a** and **c**. Replication is not a good strategy for backups for the reasons given in the answer to quiz question 2.
7. **a** and **c**. A backup based on a snapshot is not guaranteed to result in a consistent database image because some transactions might not have been successfully committed or rolled back at the point the snapshot was taken. If the database is not consistent, then InnoDB must perform its own recovery to ensure that there are no incomplete transactions.
8. **a** and **b**. The main benefit of backups based on the binary log is the fine-grained control they provide. You can use binary logs to isolate and restore specific transactions. However, you must restore binary logs in sequential order and the process can be slow depending on the number of logs you must restore.
9. **d**. Cold (raw) backup is suitable when down time is allowed. Snapshots require additional storage to keep a copy of the changed blocks. MySQL Enterprise Backup is a payable software. `mysqldump` and `mysqlpump` is more time consuming than raw backup.
10. **b** and **c**. The `mysqldump` and `mysqlpump` utilities are useful only for smaller data requirements. Replication is not an effective backup strategy for the reasons given in the answer to quiz question 2.

GANG LIU (gangli@baylorhealth.edu) has a non-transferable license
to use this Student Guide.

Practices for Lesson 11: Performing Backups

Practices for Lesson 11: Overview

Overview

In these practices, you will perform backups by using MySQL Enterprise Backup, and the `mysqldump` and `mysqlpump` utilities, and by backing up binary log files.

Assumptions

- You have created and populated the `employees` database in the practices for the lesson titled “Monitoring MySQL.”

Practice 11-1: Backing Up with MySQL Enterprise Backup

Overview

In this practice, you create a backup by using the `mysqlbackup` application. To accomplish this objective, you must:

- Modify the `my.cnf` file to configure MySQL Enterprise Backup
- Execute the `mysqlbackup` application while the MySQL server is running
- Delete database files from the `/var/lib/mysql` data directory
- Execute the `mysqlbackup` to restore the files that were deleted

Duration

This practice should take you approximately 30 minutes to complete.

Tasks

1. Create the `/backups` directory, and change its owner to the `mysql` user.
2. Open the `/etc/my.cnf` file in a text editor, and note the value of the `datadir` server parameter in the `[mysqld]` section.
3. Add a `[mysqlbackup]` section at the bottom of the file, and add the following settings to it:
 - `backup-dir: /backups/mebl`
 - `user: backupuser`
 - `socket: /var/lib/mysql/mysql.sock`
4. Save the changes to the `/etc/my.cnf` file and close the file.
5. Restart the MySQL server to apply the updated `/etc/my.cnf` file.
6. Open a second terminal window and start the `mysql` client.
7. In the second terminal window, which is connected to the MySQL server, create a new user called `backupuser`, identified by the password `MySQL8.0!`, and grant the user all the permissions that are required to use MySQL Enterprise Backup:
 - `RELOAD` on all tables, in all databases
 - `CREATE, INSERT, DROP, and UPDATE` on `mysql.backup_progress`
 - `CREATE, INSERT, DROP, UPDATE, SELECT, and ALTER` on `mysql.backup_history`
 - `CREATE, INSERT, DROP, and ALTER` on `mysql.backup_history_new`
 - `CREATE, INSERT, and DROP` on `mysql.backup_history_old`
 - `REPLICATION CLIENT, SUPER, PROCESS, BACKUP_ADMIN` on all tables, in all databases
 - `SELECT` on `performance_schema.replication_group_members`
 - `SELECT` on `performance_schema.log_status`
8. In the second terminal window, which is connected to the MySQL server, execute the `FLUSH LOGS` statement.

9. In the first terminal window, issue the following commands to run the mysqlbackup application as the mysql user.

```
# su mysql  
$ mysqlbackup --defaults-file=/etc/my.cnf -p backup-and-apply-log
```

10. In the first terminal window, list the files in the /backups/meb1 directory.
11. In the first terminal window, exit from the mysql Linux user shell back to the root shell. Then, in the /var/lib/mysql data directory, delete the employees database directory.
12. In the second terminal window, which is connected to the MySQL server, display all of the databases on the MySQL server. Does the list include the employees database?
13. Execute a SELECT query against the departments table of the employees database. Reexecute the query if the connection is lost. What is the result?
14. In the second terminal window, exit the mysql client.
15. In the first terminal window, shut down the MySQL server.
16. In the first terminal window, execute the following commands to restore the backup created to the /var/lib/mysql directory:

```
# su mysql  
$ mysqlbackup --defaults-file=/etc/my.cnf --force copy-back
```

17. In the first terminal window, exit the mysql Linux shell and start the MySQL server.
18. In the second terminal window, start the mysql client.
19. In the second terminal window, change the current database to employees and issue a query that lists the contents of the departments table.
20. In the second terminal window, exit the mysql client and close the terminal window.
21. Leave the first terminal window open for the next practice.

Solution 11-1: Backing Up with MySQL Enterprise Backup

Solution Steps

1. Create the `/backups` directory, and change its owner to the `mysql` user.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# mkdir /backups
# chown mysql:mysql /backups
```

2. Open the `/etc/my.cnf` file in a text editor, and note the value of the `datadir` server parameter in the `[mysqld]` section.

The contents of the `[mysqld]` section should resemble the following:

```
[mysqld]
datadir = /var/lib/mysql
...
```

- **Note:** You must specify the `datadir` parameter for cold backups. The directory paths must be absolute. MySQL Enterprise Backup specializes in hot/warm backups. For hot or warm backups, `mysqlbackup` queries the value from the server; it does not assume any defaults for file locations.

3. Add a `[mysqlbackup]` section at the bottom of the file, and add the following settings to it:

- `backup-dir: /backups/meb1`
- `user: backupuser`
- `socket: /var/lib/mysql/mysql.sock`

Add the following lines to the end of `/etc/my.cnf`:

```
...
[mysqlbackup]
backup-dir=/backups/meb1
user=backupuser
socket=/var/lib/mysql/mysql.sock
```

4. Save the changes to the `/etc/my.cnf` file and close the file.

5. Restart the MySQL server to apply the updated `/etc/my.cnf`.

Enter the following commands at the Linux terminal prompt:

```
# systemctl restart mysqld
# systemctl status mysqld
● mysqld.service - MySQL Server
   Loaded: loaded (/usr/lib/systemd/system/mysqld.service;
             enabled; vendor preset: disabled)
   Active: active (running) since ...
             Docs: man:mysqld(8)
                     http://dev.mysql.com/doc/refman/en/using-systemd.html
   Main PID: 28384 (mysqld)
   Status: "SERVER_OPERATING"
```

```
CGroup: /system.slice/mysqld.service
└─28384 /usr/local/mysql/bin/mysqld --defaults-
file=/etc/my.cnf
...
```

6. Open a second terminal window and start the `mysql` client.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# mysql -uroot -p
Enter password: oracle
Welcome to the MySQL monitor. Commands end with ; or \g.
...
```

7. In the second terminal window, which is connected to the MySQL server, create a new user called `backupuser`, identified by the password `MySQL8.0!`, and grant the user all the permissions that are required to use MySQL Enterprise Backup:

- RELOAD on all tables, in all databases
- CREATE, INSERT, DROP, and UPDATE on `mysql.backup_progress`
- CREATE, INSERT, DROP, UPDATE, SELECT, and ALTER on `mysql.backup_history`
- CREATE, INSERT, DROP, and ALTER on `mysql.backup_history_new`
- CREATE, INSERT, and DROP on `mysql.backup_history_old`
- REPLICATION CLIENT, SUPER, PROCESS, BACKUP_ADMIN on all tables, in all databases
- SELECT on `performance_schema.replication_group_members`
- SELECT on `performance_schema.log_status`

Enter the following statements at the `mysql` prompt and receive the results shown:

```
mysql> CREATE USER 'backupuser'@'localhost'
      -> IDENTIFIED BY 'MySQL8.0!';
Query OK, 0 rows affected (#.## sec)

mysql> GRANT RELOAD ON *.* TO 'backupuser'@'localhost';
Query OK, 0 rows affected (#.## sec)

mysql> GRANT CREATE, INSERT, DROP, UPDATE
      -> ON mysql.backup_progress TO 'backupuser'@'localhost';
Query OK, 0 rows affected (#.## sec)

mysql> GRANT CREATE, INSERT, DROP, UPDATE, SELECT, ALTER
      -> ON mysql.backup_history TO 'backupuser'@'localhost';
Query OK, 0 rows affected (#.## sec)

mysql> GRANT CREATE, INSERT, DROP, ALTER
      -> ON mysql.backup_history_new TO 'backupuser'@'localhost';
Query OK, 0 rows affected (#.## sec)
```

```

mysql> GRANT CREATE, INSERT, DROP
      -> ON mysql.backup_history_old TO 'backupuser'@'localhost';
Query OK, 0 rows affected (#.## sec)

mysql> GRANT REPLICATION CLIENT, SUPER, PROCESS, BACKUP_ADMIN
      -> ON *.* TO 'backupuser'@'localhost';
Query OK, 0 rows affected (#.## sec)

mysql> GRANT SELECT ON
      -> performance_schema.replication_group_members
      -> TO 'backupuser'@'localhost';
Query OK, 0 rows affected (#.## sec)

mysql> GRANT SELECT ON performance_schema.log_status
      -> TO 'backupuser'@'localhost';
Query OK, 0 rows affected (#.## sec)

```

8. In the second terminal window, which is connected to the MySQL server, execute the FLUSH LOGS statement.

Enter the following statement at the mysql prompt and receive the results shown:

```

mysql> FLUSH LOGS;
Query OK, 0 rows affected (#.## sec)

```

- This forces the current binary log to close and the next incremental binary log to open.

9. In the first terminal window, issue the following commands to run the mysqlbackup application as the mysql user.

```

# su mysql
$ mysqlbackup --defaults-file=/etc/my.cnf -p backup-and-apply-log

```

Enter the following commands at the Linux terminal prompt and receive the results shown:

```

# su mysql
$ mysqlbackup --defaults-file=/etc/my.cnf -p backup-and-apply-log
MySQL Enterprise Backup version 8.0.11 Linux-4.1.12-
94.2.1.el6uek.x86_64-x86_64 [date-and-time]
Copyright (c) 2003, 2018, Oracle and/or its affiliates. All
Rights Reserved.

date-and-time MAIN      INFO: A thread created with Id
'140202455238464'
date-and-time MAIN      INFO: Starting with following command
line ...
mysqlbackup --defaults-file=/etc/my.cnf -p backup-and-apply-log

date-and-time MAIN      INFO:

```

```

Enter password: MySQL8.0!
date-and-time MAIN      INFO: No SSL options specified.
date-and-time MAIN      INFO: MySQL server version is '8.0.11-
commercial'
date-and-time MAIN      INFO: MySQL server compile os version is
'linux-glibc2.12'
date-and-time MAIN      INFO: Got some server configuration
information from running server.

IMPORTANT: Please check that mysqlbackup run completes
successfully.

At the end of a successful 'backup-and-apply-log' run
mysqlbackup
    prints "mysqlbackup completed OK!".

...
date-and-time PCR1      INFO: We were able to parse
ibbackup_logfile up to
    lsn 6037486320.
date-and-time PCR1      INFO: Last MySQL binlog file position 0
155, file name binlog.000009
date-and-time PCR1      INFO: The first data file is
'/backups/meb1/datadir/ibdata1'
                                and the new created log files are
at '/backups/meb1/datadir'
date-and-time MAIN      INFO: Apply-log operation completed
successfully.
date-and-time MAIN      INFO: Full backup prepared for recovery
successfully.

mysqlbackup completed OK!

```

Note: When you use `su` to change from a `root` account to a `non-root` account, the prompt changes from `#` to `$`.

10. In the first terminal window, list the files in the `/backups/meb1` directory.

Enter the following command at the Linux terminal prompt and receive the results shown:

```

$ ls /backups/meb1/
backup-my.cnf  datadir  meta  server-all.cnf  server-my.cnf

```

11. In the first terminal window, exit from the `mysql` Linux user shell back to the `root` shell.
Then, in the `/var/lib/mysql` data directory, delete the `employees` database directory.

Enter the following commands at the Linux terminal prompt and receive the results shown:

```

$ exit
# cd /var/lib/mysql
# rm -rf employees

```

12. In the second terminal window, which is connected to the MySQL server, display all of the databases on the MySQL server. Does the list include the `employees` database?

Enter the following statement at the `mysql` prompt and receive the results shown:

```
mysql> SHOW DATABASES;
+-----+
| Database      |
+-----+
| employees     |
| information_schema |
| mysql          |
| mysqlslap      |
| performance_schema |
| sys            |
+-----+
6 rows in set (#.## sec)
```

- Does the list include the `employees` database? **Answer: Yes.** Details of the `employees` database are preserved in the transactional data dictionary.

13. Execute a `SELECT` query against the `departments` table of the `employees` database. Reexecute the query if the connection is lost. What is the result?

Enter the following statement at the `mysql` prompt and receive the results shown:

```
mysql> SELECT * FROM employees.departments;
ERROR 2013 (HY000): Lost connection to MySQL server during query

mysql> SELECT * FROM employees.departments;
ERROR 2006 (HY000): MySQL server has gone away
No connection. Trying to reconnect...
Connection id:      9
Current database: *** NONE ***
ERROR 1812 (HY000): Tablespace is missing for table
`employees`.`departments`.
```

- Connection is lost because MySQL server crashed and the service manager restarts it.
- MySQL cannot execute the query because you deleted the `employees` database directory from the file system, with all its tablespaces.

14. In the second terminal window, exit the `mysql` client.

Enter the following statement at the `mysql` prompt and receive the results shown:

```
mysql> EXIT
Bye
#
```

15. In the first terminal window, shut down the MySQL server.

Enter the following commands at the Linux terminal prompt and receive the results shown:

```
# systemctl stop mysqld
# systemctl status mysqld
● mysqld.service - MySQL Server
   Loaded: loaded (/usr/lib/systemd/system/mysqld.service; enabled; vendor preset: disabled)
     Active: inactive (dead) since date-and-time; 4s ago
       Docs: man:mysqld(8)
              http://dev.mysql.com/doc/refman/en/using-systemd.html
      Process: 28757 ExecStart=/usr/local/mysql/bin/mysqld --defaults-file=/etc/my.cnf $MYSQLD_OPTS (code=exited, status=0/SUCCESS)
     Main PID: 28757 (code=exited, status=0/SUCCESS)
        Status: "SERVER_SHUTTING_DOWN"
...
date-and-time edvmrlp0 systemd[1]: Stopped MySQL Server.
Hint: Some lines were ellipsized, use -l to show in full.
```

16. In the first terminal window, execute the following commands to restore the backup created to the /var/lib/mysql directory:

```
# su mysql
$ mysqlbackup --defaults-file=/etc/my.cnf --force copy-back
```

Enter the following commands at the Linux terminal prompt and receive the results shown:

```
# su mysql
$ mysqlbackup --defaults-file=/etc/my.cnf --force copy-back
MySQL Enterprise Backup version 8.0.11 Linux-4.1.12-
94.2.1.el6uek.x86_64-x86_64 [date-and-time]
Copyright (c) 2003, 2018, Oracle and/or its affiliates. All
Rights Reserved.

date-and-time MAIN      INFO: A thread created with Id
'139958497490752'
date-and-time MAIN      INFO: Starting with following command
line ...
mysqlbackup --defaults-file=/etc/my.cnf --force copy-back

date-and-time MAIN      INFO:
IMPORTANT: Please check that mysqlbackup run completes
successfully.

At the end of a successful 'copy-back' run
mysqlbackup
           prints "mysqlbackup completed OK!".
```

```

date-and-time MAIN      INFO: MySQL server version is '8.0.11-
commercial'
date-and-time MAIN      INFO: Restoring ...8.0.11-commercial
version
date-and-time MAIN WARNING: If you restore to a server of a
different version, the innodb_data_file_path parameter might
have a different default. In that case you need to add
'innodb_data_file_path=ibdata1:12M:autoextend' to the target
server configuration.
date-and-time MAIN WARNING: If you restore to a server of a
different version, the innodb_log_files_in_group parameter might
have a different default. In that case you need to add
'innodb_log_files_in_group=2' to the target server
configuration.
date-and-time MAIN WARNING: If you restore to a server of a
different version, the innodb_log_file_size parameter might have
a different default. In that case you need to add
'innodb_log_file_size=50331648' to the target server
configuration.
...
date-and-time MAIN WARNING: External plugins list found in
meta/backup_content.xml. Please ensure that all plugins are
installed in restored server.
date-and-time MAIN      INFO: Copy-back operation completed
successfully.
date-and-time MAIN      INFO: Finished copying backup files to
'/var/lib/mysql'

mysqlbackup completed OK! with 4 warnings

```

- The --force option instructs mysqlbackup to overwrite an existing data directory. In this case, the data directory is only partially complete, because you deleted only the employees database directory and the shared tablespace file ibdata1.
- Some warnings appear because the my.cnf file does not contain explicit values for options that affect the behavior of mysqlbackup. The implicit values appear along with the text of the warnings.
- You might also receive a warning reminding you to install plugins in the restored server, as shown in the sample output.

17. In the first terminal window, exit from the mysql Linux shell and start the MySQL server.

Enter the following commands at the Linux terminal prompt and receive the results shown:

```

$ exit
# systemctl start mysqld
# systemctl status mysqld
● mysqld.service - MySQL Server
    Loaded: loaded (/usr/lib/systemd/system/mysqld.service;
   enabled; vendor preset: disabled)

```

```

Active: active (running) since date-and-time; 6s ago
Docs: man:mysqld(8)
      http://dev.mysql.com/doc/refman/en/using-systemd.html
Main PID: 28841 (mysqld)
Status: "SERVER_OPERATING"
CGroup: /system.slice/mysqld.service
      └─28841 /usr/local/mysql/bin/mysqld --defaults-
file=/etc/my.cnf
...
date-and-time edvmrlp0 systemd[1]: Started MySQL Server.
Hint: Some lines were ellipsized, use -l to show in full.

```

- Press the Enter key to return the terminal prompt.
18. In the second terminal window, start the mysql client.

Enter the following command at the Linux terminal prompt and receive the results shown:

```

# mysql -uroot -p
Enter password: oracle
...
mysql>

```

19. In the second terminal window, change the current database to employees and issue a query that lists the contents of the departments table.

Enter the following statements at the mysql prompt and receive the results shown:

```

mysql> USE employees
Database changed
mysql> SELECT * FROM departments;
+-----+-----+
| dept_no | dept_name   |
+-----+-----+
| d009    | Customer Service |
| d005    | Development   |
| d010    | Distribution  |
| d002    | Finance       |
| d003    | Human Resources |
| d001    | Marketing     |
| d004    | Production    |
| d006    | Quality Management |
| d008    | Research      |
| d007    | Sales         |
+-----+-----+
10 rows in set (#.## sec)

```

- The employees database is restored.

20. In the second terminal window, exit the mysql client and close the terminal window.

Enter the following statements at the mysql prompt and Linux shell and receive the results shown:

```
mysql> EXIT  
Bye  
# exit
```

21. Leave the first terminal window open for the next practice.

Practice 11-2: Backing Up with mysqldump and mysqlpump

Overview

In this practice, you use the `mysqldump` application to create a backup copy of the `employees` database. You then create the same backup with `mysqlpump`. Finally, you restore most of the contents of the first backup you created to a new database.

Duration

This practice should take you approximately 20 minutes to complete.

Tasks

1. Display the value of the `secure_file_priv` system variable.
2. Change the value of the `secure_file_priv` system variable to the location of the `/backups` directory by entering a suitable value in `/etc/my.cnf`.
3. Restart the MySQL server.
4. Verify that the new value of `secure_file_priv` now applies.
5. Using the `mysqldump` tool, make a tab-delimited backup of the `employees` database to the `/backups` directory.
6. Review the contents of the `/backups` directory.
 - How many `*.sql` files are located in this directory?
 - How many `*.txt` files are located in this directory?
7. Review the contents of the `dept_manager.sql` file.
8. Review the contents of the `/backups/dept_manager.txt` file. What is the file format?
9. Create a subdirectory in `/backups`, called `pump`.
10. Use the `mysqlpump` tool to back up the `employees` database to a single SQL script file called `pump.sql` in the `/backups/pump` directory. Supply suitable arguments to `mysqlpump` so that it processes only the `employees` database and uses five threads.
11. Verify that `mysqlpump` created the `pump.sql` file in the `/backups/pump` directory.
12. Using the `mysqladmin` tool, create a new database called `emps2`.
13. Using the `mysql` application, load each of the `*.sql` files for the `departments`, `employees`, and `dept_manager` tables that you created by using the `mysqldump` tool in step 5 into the `emps2` database.

Note: You must create the `departments` and `employees` tables before you create the `dept_manager` table, because `dept_manager` has foreign key dependencies on the `departments` and `employees` tables.

14. Using the `mysqlimport` tool, load each of the `*.txt` files for the `departments`, `employees`, and `dept_manager` tables that you created in step 5.
Note: You must import data into the `departments` and `employees` tables before you import rows from the `dept_manager` table so that the import process does not violate the foreign key constraints in the `dept_manager` table.
15. Using the `mysql` client, review the tables in the `emps2` database.
16. Verify that the `emps2.employees` table and the `employees.employees` table contain the same number of rows.
17. Verify that the `departments` and `dept_manager` tables in both the `emps2` and `employees` databases contain the same number of rows.
18. Drop the `emps2` database.
19. Exit the `mysql` client session.
20. Leave the terminal window open for the next practice.

Solution 11-2: Backing Up with mysqldump and mysqlpump

Solution Steps

- Display the value of the `secure_file_priv` system variable.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# mysql -uroot -p -e"SHOW VARIABLES LIKE 'secure_file%'"  
Enter password: oracle  
+-----+-----+  
| Variable_name      | Value |  
+-----+-----+  
| secure_file_priv  | NULL  |  
+-----+-----+
```

- The value of the `secure_file_priv` system variable is `NULL` in Oracle classroom environments. This limits data import/export operations and prevents the use of `mysqldump` and `mysqlpump`.

- Change the value of the `secure_file_priv` system variable to the location of the `/backups` directory by entering a suitable value in `/etc/my.cnf`.

Edit the contents of `/etc/my.cnf` to read as follows:

```
[mysqld]  
...  
symbolic-links=0  
secure-file-priv=/backups  
  
[mysqld_safe]  
log-error=/var/log/mysqld.log  
pid-file=/var/run/mysqld/mysqld.pid  
...
```

- Restart the MySQL server.

Enter the following commands at the Linux terminal prompt and receive the results shown:

```
# systemctl restart mysqld  
# systemctl status mysqld  
● mysqld.service - MySQL Server  
   Loaded: loaded (/usr/lib/systemd/system/mysqld.service;  
             enabled; vendor preset: disabled)  
     Active: active (running) since Wed date-and-time; 11s ago  
           ...  
           date-and-time edvmrlp0 systemd[1]: Started MySQL Server.  
Hint: Some lines were ellipsized, use -l to show in full.
```

4. Verify that the new value of `secure_file_priv` now applies.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# mysql -uroot -p -e"SHOW VARIABLES LIKE 'secure_file%'";
Enter password: oracle
+-----+-----+
| Variable_name      | Value      |
+-----+-----+
| secure_file_priv  | /backups/  |
+-----+-----+
```

5. Using the `mysqldump` tool, make a tab-delimited backup of the `employees` database to the `/backups` directory.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# mysqldump -uroot -p --tab=/backups employees
Enter password: oracle
```

6. Review the contents of the `/backups` directory.

- How many `*.sql` files are located in this directory?
- How many `*.txt` files are located in this directory?

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# ls /backups -l
total 138224
-rw-r--r-- 1 root  root    1464 date-time departments.sql
-rw-rw-rw- 1 mysql mysql     153 date-time departments.txt
-rw-r--r-- 1 root  root    1747 date-time dept_emp.sql
-rw-rw-rw- 1 mysql mysql 11175033 date-time dept_emp.txt
-rw-r--r-- 1 root  root    1767 date-time dept_manager.sql
-rw-rw-rw- 1 mysql mysql     816 date-time dept_manager.txt
-rw-r--r-- 1 root  root    1547 date-time employees.sql
-rw-rw-rw- 1 mysql mysql 13821993 date-time employees.txt
drwxr-x--- 4 mysql mysql    4096 date-time meb1
-rw-r--r-- 1 root  root    1608 date-time salaries.sql
-rw-rw-rw- 1 mysql mysql 98781181 date-time salaries.txt
-rw-r--r-- 1 root  root    1615 date-time titles.sql
-rw-rw-rw- 1 mysql mysql 17718376 date-time titles.txt
```

- How many `*.sql` files are located in this directory? **Answer:** Six; one for each table in the `employees` database.
- How many `*.txt` files are located in this directory? **Answer:** Six; one for each table in the `employees` database

7. Review the contents of the `dept_manager.sql` file.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# cat /backups/dept_manager.sql
-- MySQL dump 10.13  Distrib 8.0.16, for linux-glibc2.12 (x86_64)
--
-- Host: localhost      Database: employees
-- -----
-- Server version      8.0.16-commercial

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
    SET NAMES utf8mb4 ;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE=''' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `dept_manager`
--

DROP TABLE IF EXISTS `dept_manager`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
    SET character_set_client = utf8mb4 ;
CREATE TABLE `dept_manager`(
    `dept_no` char(4) NOT NULL,
    `emp_no` int(11) NOT NULL,
    `from_date` date NOT NULL,
    `to_date` date NOT NULL,
    PRIMARY KEY (`emp_no`, `dept_no`),
    KEY `emp_no` (`emp_no`),
    KEY `dept_no` (`dept_no`),
    CONSTRAINT `dept_manager_ibfk_1` FOREIGN KEY (`emp_no`) REFERENCES `employees`(`emp_no`) ON DELETE CASCADE,
    CONSTRAINT `dept_manager_ibfk_2` FOREIGN KEY (`dept_no`) REFERENCES `departments`(`dept_no`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on date-and-time
```

8. Review the contents of the /backups/dept_manager.txt file. What is the file format?
 Enter the following command at the Linux terminal prompt and receive the results shown:

```
# cat /backups/dept_manager.txt
d001    110022  1985-01-01      1991-10-01
d001    110039  1991-10-01      9999-01-01
d002    110085  1985-01-01      1989-12-17
d002    110114  1989-12-17      9999-01-01
d003    110183  1985-01-01      1992-03-21
d003    110228  1992-03-21      9999-01-01
d004    110303  1985-01-01      1988-09-09
d004    110344  1988-09-09      1992-08-02
d004    110386  1992-08-02      1996-08-30
d004    110420  1996-08-30      9999-01-01
d005    110511  1985-01-01      1992-04-25
d005    110567  1992-04-25      9999-01-01
d006    110725  1985-01-01      1989-05-06
d006    110765  1989-05-06      1991-09-12
d006    110800  1991-09-12      1994-06-28
d006    110854  1994-06-28      9999-01-01
d007    111035  1985-01-01      1991-03-07
d007    111133  1991-03-07      9999-01-01
d008    111400  1985-01-01      1991-04-08
d008    111534  1991-04-08      9999-01-01
d009    111692  1985-01-01      1988-10-17
d009    111784  1988-10-17      1992-09-08
d009    111877  1992-09-08      1996-01-03
d009    111939  1996-01-03      9999-01-01
```

- What is the file format? **Answer:** It is a tab-delimited text file.
9. Create a subdirectory in /backups, called pump.
 Enter the following commands at the Linux terminal prompt and receive the results shown:

```
# mkdir /backups/pump
# ls -l /backups
total 138220
-rw-r--r-- 1 root  root      1460 date-time departments.sql
-rw-r--r-- 1 root  root      1743 date-time dept_emp.sql
-rw-rw-rw- 1 mysql mysql 11175033 date-time dept_emp.txt
-rw-r--r-- 1 root  root      1763 date-time dept_manager.sql
-rw-rw-rw- 1 mysql mysql      816 date-time dept_manager.txt
-rw-r--r-- 1 root  root      1543 date-time employees.sql
-rw-rw-rw- 1 mysql mysql 13821993 date-time employees.txt
drwxr-x--- 4 mysql mysql     4096 date-time meb1
drwxr-xr-x  2 root  root     4096 date-time pump
-rw-r--r-- 1 root  root      1631 date-time salaries.sql
-rw-rw-rw- 1 mysql mysql 98781181 date-time salaries.txt
-rw-r--r-- 1 root  root      1611 date-time titles.sql
-rw-rw-rw- 1 mysql mysql 17718376 date-time titles.txt
```

10. Use the `mysqlpump` tool to back up the `employees` database to a single SQL script file called `pump.sql` in the `/backups/pump` directory. Supply suitable arguments to `mysqlpump` so that it processes only the `employees` database and uses five threads.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# mysqlpump -uroot -p --databases employees \
> --default-parallelism=5 > /backups/pump/pump.sql
Enter password: oracle
Dump progress: 0/5 tables, 250/3469408 rows
Dump progress: 0/6 tables, 2524435/3896234 rows
Dump completed in 1928 milliseconds
```

11. Verify that `mysqlpump` created the `pump.sql` file in the `/backups/pump` directory.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# ls -al /backups/pump
total 165084
drwxr-xr-x 2 root root 4096 date-time .
drwxr-xr-x 3 mysql mysql 4096 date-time ..
-rw-r--r-- 1 root root 169037412 date-time pump.sql
```

The `mysqlpump` tool created the backup in a single script file. You could restore this file by executing the statements it contains. For example, using the `mysql` command-line client:

```
# mysql < pump.sql
```

Note: Do not perform this step. You will use `mysqlimport` in subsequent steps to restore the backup from `mysqldump`.

12. Using the `mysqladmin` tool, create a new database called `emps2`.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# mysqladmin -uroot -p create emps2
Enter password: oracle
```

13. Using the `mysql` application, load each of the `*.sql` files for the `departments`, `employees`, and `dept_manager` tables that you created by using the `mysqldump` tool in step 5 into the `emps2` database.

Enter the following commands at the Linux terminal prompt and receive the results shown:

```
# cd /backups
# mysql -uroot -p emps2 < departments.sql
Enter password: oracle
# mysql -uroot -p emps2 < employees.sql
Enter password: oracle
# mysql -uroot -p emps2 < dept_manager.sql
Enter password: oracle
```

Note: You must create the `departments` and `employees` tables before you create the `dept_manager` table, because `dept_manager` has foreign key dependencies on the `departments` and `employees` tables.

14. Using the `mysqlimport` tool, load each of the `*.txt` files for the `departments`, `employees`, and `dept_manager` tables that you created in step 5.

Enter the following commands at the Linux terminal prompt and receive the results shown:

```
# mysqlimport -uroot -p emps2 /backups/departments.txt
Enter password: oracle
emps2.departments: Records: 10 Deleted: 0 Skipped: 0
Warnings: 0
# mysqlimport -uroot -p emps2 /backups/employees.txt
Enter password: oracle
emps2.employees: Records: 300024 Deleted: 0 Skipped: 0
Warnings: 0
# mysqlimport -uroot -p emps2 /backups/dept_manager.txt
Enter password: oracle
emps2.dept_manager: Records: 24 Deleted: 0 Skipped: 0
Warnings: 0
```

- The `mysqlimport` tool uses the file name to determine the name of the table into which it imports the rows.

Note: You must import data into the `departments` and `employees` tables before you import rows from the `dept_manager` table so that the import process does not violate the foreign key constraints in the `dept_manager` table.

15. Using the `mysql` client, review the tables in the `emps2` database.

Enter the following commands at the Linux terminal prompt and receive the results shown:

```
# mysql -uroot -p
Enter password: oracle
Welcome to the MySQL monitor. Commands end with ; or \g.
...
mysql> USE emps2
...
Database changed
mysql> SHOW TABLES;
+-----+
| Tables_in_emps2 |
+-----+
| departments      |
| dept_manager     |
| employees        |
+-----+
3 rows in set (#.## sec)
```

16. Verify that the `emps2.employees` table and the `employees.employees` table contain the same number of rows.

Enter the following statements at the `mysql` prompt and receive the results shown:

```
mysql> SELECT COUNT(*) FROM emps2.employees;
+-----+
| count(*) |
+-----+
|      300024 |
+-----+
1 row in set (#.## sec)

mysql> SELECT COUNT(*) FROM employees.employees;
+-----+
| count(*) |
+-----+
|      300024 |
+-----+
1 row in set (#.## sec)
```

17. Verify that the departments and dept_manager tables in both the emps2 and employees databases contain the same number of rows.

Enter the following statements at the mysql prompt and receive the results shown:

```
mysql> SELECT COUNT(*) FROM emps2.departments;
+-----+
| count(*) |
+-----+
|      10 |
+-----+
1 row in set (#.## sec)

mysql> SELECT COUNT(*) FROM employees.departments;
+-----+
| count(*) |
+-----+
|      10 |
+-----+
1 row in set (#.## sec)

mysql> SELECT COUNT(*) FROM emps2.dept_manager;
+-----+
| count(*) |
+-----+
|      24 |
+-----+
1 row in set (#.## sec)

mysql> SELECT COUNT(*) FROM employees.dept_manager;
+-----+
| count(*) |
+-----+
|      24 |
+-----+
1 row in set (#.## sec)
```

18. Drop the emps2 database.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> DROP DATABASE emps2;
Query OK, 3 rows affected (#.## sec)
```

19. Exit the mysql client session.

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> EXIT
Bye
```

20. Leave the terminal window open for the next practice.

Practice 11-3: Backing Up by Using the Binary Log

Overview

In this practice, you back up the `employees` database by using a full logical backup and subsequent binary log backups. You then restore the database, including later events, to a new server instance.

Duration

This practice should take you approximately 20 minutes to complete.

Tasks

1. List the first 15 rows of the `titles` table in the `employees` database. You will modify this table later in this practice.
2. In a new terminal window, use `mysqldump` to back up the `employees` database to a file called `employees_full.sql` in the `/backups` directory, by executing the following command:

```
# mysqldump -uroot -p --single-transaction --master-data=2 \
> employees > /backups/employees_full.sql
```
3. Locate the binary log position in the dump file, by opening the `/backups/employees_full.sql` file in a text editor and finding the line that contains the `CHANGE MASTER TO` comment. Make a note of the binary log file and log position in that comment.
4. Close the `employees_full.sql` file.
5. Switch to the terminal window logged in to the MySQL server and use the `REPLACE` function in an `UPDATE` statement to change the word “Assistant” to the abbreviation “Asst” in the `titles` table.
6. Switch back to the window that contains the Linux terminal prompt and identify which binary log files contain all events after the log position recorded in the full backup. Copy those binary log files from the data directory to the `/backups` directory.
7. Flush the binary log by executing the `FLUSH BINARY LOGS` SQL statement in the `mysql` terminal window.
8. Use the `REPLACE` function in an `UPDATE` statement to change the word “Senior” to the abbreviation “Snr” in the `titles` table.
9. Exit the `mysql` client.
10. Copy the binary log files that contain all events after the log position recorded in step 3 from the data directory to the `/backups` directory.
11. Use `systemctl` to stop the MySQL server process.

12. Create a new MySQL instance to restore the modified `employees` database by performing the following steps:

- a. Create the `/datadir` directory and grant ownership to the `mysql` user:

```
# mkdir /datadir  
# chown mysql:mysql /datadir
```

- b. Use the `mysqld --initialize-insecure` command to set up a new instance with a temporary data directory in the `/datadir` directory and no `root` password:

```
# mysqld --datadir=/datadir --initialize-insecure &
```

– Press Enter to return to the Linux command prompt.

- c. Start the new instance as a background process, providing the `--datadir=/datadir` option:

```
# mysqld --datadir=/datadir &
```

13. List the databases that the new instance hosts.

14. Create a new database called `employees`, and make it the current database.

15. Restore the full `employees` dump that you created in step 2.

16. Use the `mysqlbinlog` program to restore all events that took place after the full backup from the binary logs that you backed up in step 10. Enter the following command at the Linux terminal prompt, replacing the file names and start position with those you determined in step 3:

```
# mysqlbinlog --start-position=8250859 \  
> /backups/binlog.000012 \  
> /backups/binlog.000013 | mysql -uroot
```

17. Verify that the `employees` database contains the data from both the full database backup and all subsequent logged changes.

18. Exit the `mysql` client.

19. Use `mysqladmin` to stop the server instance you started in step 12.

20. Delete the `/datadir` directory.

21. Use `systemctl` to restart the MySQL server that was running at the beginning of this practice.

22. Close any open terminal windows.

Solution 11-3: Backing Up by Using the Binary Log

Solution Steps

1. In the terminal window (`t1`), start `mysql` client and list the first 15 rows of the `titles` table in the `employees` database. You modify this table in later steps in this practice.

Enter the following command at the `t1` Linux terminal prompt and receive the results shown:

```
# mysql -uroot -p
Enter password: oracle
Welcome to the MySQL monitor. Commands end with ; or \g.
...
mysql> USE employees
...
Database changed
mysql> SELECT * FROM titles LIMIT 15;
+-----+-----+-----+
| emp_no | title           | from_date   | to_date    |
+-----+-----+-----+
| 10001  | Senior Engineer  | 1986-06-26  | 9999-01-01 |
| 10002  | Staff            | 1996-08-03  | 9999-01-01 |
| 10003  | Senior Engineer  | 1995-12-03  | 9999-01-01 |
| 10004  | Engineer         | 1986-12-01  | 1995-12-01 |
| 10004  | Senior Engineer  | 1995-12-01  | 9999-01-01 |
| 10005  | Senior Staff     | 1996-09-12  | 9999-01-01 |
| 10005  | Staff            | 1989-09-12  | 1996-09-12 |
| 10006  | Senior Engineer  | 1990-08-05  | 9999-01-01 |
| 10007  | Senior Staff     | 1996-02-11  | 9999-01-01 |
| 10007  | Staff            | 1989-02-10  | 1996-02-11 |
| 10008  | Assistant Engineer| 1998-03-11  | 2000-07-31 |
| 10009  | Assistant Engineer| 1985-02-18  | 1990-02-18 |
| 10009  | Engineer          | 1990-02-18  | 1995-02-18 |
| 10009  | Senior Engineer   | 1995-02-18  | 9999-01-01 |
| 10010  | Engineer          | 1996-11-24  | 9999-01-01 |
+-----+-----+-----+
15 rows in set (0.00 sec)
```

- The `titles` table contains over 440,000 rows. The preceding output shows a sample of the data contained in the table.
2. Open a new terminal window (`t2`), use `mysqldump` to back up the `employees` database to a file called `employees_full.sql` in the `/backups` directory, by executing the following command:

```
# mysqldump -uroot -p --single-transaction --master-data=2 \
> employees > /backups/employees_full.sql
Enter password: oracle
```

3. In t2, locate the binary log position in the dump file, by opening the /backups/employees_full.sql file in a text editor and finding the line that contains the CHANGE MASTER TO comment. Make a note of the binary log file and log position in that comment.

The following output is an example.

```
# less /backups/employees_full.sql
-- MySQL dump 10.13 Distrib 8.0.version
--
-- Host: localhost      Database: employees
-- -----
-- Server version      8.0.version

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
SET NAMES utf8mb4 ;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Position to start replication or point-in-time recovery from
--

-- CHANGE MASTER TO MASTER_LOG_FILE='binlog.000012',
MASTER_LOG_POS=8250859;

--
-- Table structure for table `departments`
--


DROP TABLE IF EXISTS `departments`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
SET character_set_client = utf8mb4 ;
CREATE TABLE `departments` (
...
```

- In the preceding example, the point at which the backup completed is located in the binary log file binlog.000012, with log position 8250859.

4. In t2, close the employees_full.sql file.
If you use less to view the file, press q key to exit.

5. In t1, use the REPLACE function in an UPDATE statement to change the word “Assistant” to the abbreviation “Asst” in the titles table.

Enter the following statement at the mysql prompt in t1 and receive the results shown:

```
mysql> UPDATE titles SET title =
-> REPLACE(title, 'Assistant', 'Asst');
Query OK, 15128 rows affected (#.## sec)
Rows matched: 443308  Changed: 15128  Warnings: 0
```

- The REPLACE function in the statement replaces all instances of “Assistant” in the title column to “Asst.” There is no WHERE clause, so the UPDATE statement operates on all rows.

6. In t2, identify which binary log files contain all events after the log position recorded in the full backup. Copy those binary log files from the data directory to the /backups directory.

Enter the following command at the t2 Linux terminal prompt and receive the results shown:

```
# ls -1 /var/lib/mysql/binlog*
/var/lib/mysql/binlog.000001
/var/lib/mysql/binlog.000002
/var/lib/mysql/binlog.000003
/var/lib/mysql/binlog.000004
/var/lib/mysql/binlog.000005
/var/lib/mysql/binlog.000006
/var/lib/mysql/binlog.000007
/var/lib/mysql/binlog.000008
/var/lib/mysql/binlog.000009
/var/lib/mysql/binlog.000010
/var/lib/mysql/binlog.000011
/var/lib/mysql/binlog.000012
/var/lib/mysql/binlog.index
```

- The -1 option of the ls command produces single-column output.
- The preceding output shows 12 binlog files and an index file. The binlog position in the example backup output in step 3 identifies the binlog.000012 file. In this example, you would need to copy only that file to capture all events since the backup. If your output shows an additional binary log file after the file that you identified in step 3, you must execute the following command twice: once for the file specified, and once for the new subsequent binary log file.

Enter the following command in the t2 Linux terminal, replacing the file name with the file name that you derived in step 3:

```
# cp /var/lib/mysql/binlog.000012 /backups
```

- Execute the command a second time if required, specifying any subsequent file.
- In a production environment, you might host the /backups directory on network storage to provide physical redundancy. In such a case, you could recover from a disaster (including complete server loss) by restoring the contents of the dump file to a new server and applying changes from the binary log.

7. In t1, flush the binary log by executing the FLUSH BINARY LOGS SQL statement in the mysql terminal window.

Enter the following statement at the mysql prompt in t1 and receive the results shown:

```
mysql> FLUSH BINARY LOGS;
Query OK, 0 rows affected (#.# sec)
```

- This command closes and reopens the binary log, and creates a new binary log file. It does not delete old events.

8. In t1, use the REPLACE function in an UPDATE statement to change the word “Senior” to the abbreviation “Snr” in the titles table.

Enter the following statement at the mysql prompt in t1 and receive the results shown:

```
mysql> UPDATE titles SET title =
      -> REPLACE(title, 'Senior', 'Snr');
Query OK, 190603 rows affected (#.# sec)
Rows matched: 443308  Changed: 190603  Warnings: 0
```

9. In t1, exit the mysql client.

Enter the following statement at the mysql prompt in t1 and receive the results shown:

```
mysql> EXIT
Bye
```

10. In t2, copy the binary log files that contain all events after the log position recorded in step 3 from the data directory to the /backups directory.

Enter the following command at the t2 Linux terminal prompt and receive the results shown:

```
# ls -1 /var/lib/mysql/binlog*
/var/lib/mysql/binlog.000001
/var/lib/mysql/binlog.000002
/var/lib/mysql/binlog.000003
/var/lib/mysql/binlog.000004
/var/lib/mysql/binlog.000005
/var/lib/mysql/binlog.000006
/var/lib/mysql/binlog.000007
/var/lib/mysql/binlog.000008
/var/lib/mysql/binlog.000009
/var/lib/mysql/binlog.000010
/var/lib/mysql/binlog.000011
/var/lib/mysql/binlog.000012
/var/lib/mysql/binlog.000013
/var/lib/mysql/binlog.index
```

- The preceding output shows a new binary log file. The FLUSH BINARY LOGS command created this file.

Enter the following command at the `t2` Linux terminal prompt, replacing the file names with those that apply on your computer:

```
# cp /var/lib/mysql/binlog.000012 \
> /var/lib/mysql/binlog.000013 /backups
cp: overwrite `/backups/binlog.000012'? y
```

- The preceding command overwrites the copy of `binlog.000012` that you created in step 6.

At this point, you have a complete dump of the `employees` database and all subsequent changes recorded in two binary log files.

11. In `t2`, use `systemctl` to stop the MySQL server process.

Enter the following commands at the `t2` Linux terminal prompt and receive the results shown:

```
# systemctl stop mysqld
# systemctl status mysqld
● mysqld.service - MySQL Server
    Loaded: loaded (/usr/lib/systemd/system/mysqld.service;
              enabled; vendor preset: disabled)
    Active: inactive (dead) since date-and-time UTC; 12s ago
      Docs: man:mysqld(8)
             http://dev.mysql.com/doc/refman/en/using-systemd.html
      Process: 29088 ExecStart=/usr/local/mysql/bin/mysqld --defaults-file=/etc/my.cnf $MYSQLD_OPTS (code=exited,
              status=0/SUCCESS)
     Main PID: 29088 (code=exited, status=0/SUCCESS)
       Status: "SERVER_SHUTTING_DOWN"
...
Aug 02 16:55:49 edvmr1p0 systemd[1]: Stopped MySQL Server.
Hint: Some lines were ellipsized, use -l to show in full.
```

12. In `t2`, create a new MySQL instance to restore the modified `employees` database by performing the following steps:

- a. Create the `/datadir` directory and grant ownership to the `mysql` user:

```
# mkdir /datadir
# chown mysql:mysql /datadir
```

- b. Use the `mysqld --initialize-insecure` command to set up a new instance with a temporary data directory in the `/datadir` directory and no `root` password:

```
# mysqld --datadir=/datadir --initialize-insecure &
```

- Press Enter to return to the Linux command prompt.

- c. Start the new instance as a background process, providing the `--datadir=/datadir` option:

```
# mysqld --datadir=/datadir &
```

13. In t1, start mysql client without the password option and list the databases that the new instance hosts.

Enter the following at the t1 Linux terminal prompt, and receive the result shown as follows:

```
# mysql -uroot
Welcome to the MySQL monitor.  Commands end with ; or \g.
...
mysql> SHOW DATABASES;
+-----+
| Database      |
+-----+
| information_schema |
| mysql          |
| performance_schema |
| sys            |
+-----+
4 rows in set (#.## sec)
```

- This is a new, empty instance with no password, and only the system databases appear. In particular, the employees database is not present.

14. In t1, create a new database called employees, and make it the current database.

Enter the following statements at the mysql prompt in t1 and receive the results shown:

```
mysql> CREATE DATABASE employees;
Query OK, 1 row affected (#.## sec)

mysql> USE employees
Database changed
```

15. In t1, restore the full employees dump that you created in step 2.

Enter the following statement at the mysql prompt in t1 and receive the results shown:

```
mysql> SOURCE /backups/employees_full.sql
Query OK, 0 rows affected (#.## sec)

Query OK, 0 rows affected (#.## sec)
...
```

- The operation takes a few minutes.

16. In t2, use the mysqlbinlog program to restore all events that took place after the full backup from the binary logs that you backed up in step 10. Enter the following command at the Linux terminal prompt, replacing the file names and start position with those you determined in step 3:

```
# mysqlbinlog --start-position=8250859 \
> /backups/binlog.000012 \
> /backups/binlog.000013 | mysql -uroot
```

17. In t1, verify that the employees database contains the data from both the full database backup and all subsequent logged changes.

Enter the following statement at the mysql prompt in t1 and receive the results shown:

```
mysql> SELECT * FROM titles LIMIT 15;
+-----+-----+-----+-----+
| emp_no | title      | from_date | to_date   |
+-----+-----+-----+-----+
| 10001  | Srn Engineer | 1986-06-26 | 9999-01-01 |
| 10002  | Staff       | 1996-08-03 | 9999-01-01 |
| 10003  | Srn Engineer | 1995-12-03 | 9999-01-01 |
| 10004  | Engineer    | 1986-12-01 | 1995-12-01 |
| 10004  | Srn Engineer | 1995-12-01 | 9999-01-01 |
| 10005  | Srn Staff   | 1996-09-12 | 9999-01-01 |
| 10005  | Staff       | 1989-09-12 | 1996-09-12 |
| 10006  | Srn Engineer | 1990-08-05 | 9999-01-01 |
| 10007  | Srn Staff   | 1996-02-11 | 9999-01-01 |
| 10007  | Staff       | 1989-02-10 | 1996-02-11 |
| 10008  | Asst Engineer | 1998-03-11 | 2000-07-31 |
| 10009  | Asst Engineer | 1985-02-18 | 1990-02-18 |
| 10009  | Engineer    | 1990-02-18 | 1995-02-18 |
| 10009  | Srn Engineer | 1995-02-18 | 9999-01-01 |
| 10010  | Engineer    | 1996-11-24 | 9999-01-01 |
+-----+-----+-----+-----+
15 rows in set (#.## sec)
```

- All instances of “Senior” and “Assistant” that appeared in the output of step 1 have changed to “Srn” and “Asst,” respectively, indicating that the full backup and all changes that were in the binary log are in the new database.

18. In t1, exit the mysql client.

Enter the following statement at the mysql prompt in t1 and receive the results shown:

```
mysql> EXIT
Bye
```

19. In t2, use mysqladmin to stop the server instance you started in step 12.

Enter the following command at the t2 Linux terminal prompt and receive the results shown:

```
# mysqladmin -uroot shutdown
date-and-time 0 [System] [MY-010910] [Server] /opt/mysql-
commercial-8.0.version/bin/mysqld: Shutdown complete (mysqld
8.0.version) MySQL Enterprise Server - Commercial.
[2]+ Done                      mysqld --datadir=/datadir
```

20. In t2, delete the /datadir directory.

Enter the following command at the t2 Linux terminal prompt:

```
# rm -rf /datadir
```

21. In t2, use systemctl to restart the MySQL server that was running at the beginning of this practice.

Enter the following commands at the t2 Linux terminal prompt and receive the results shown:

```
# systemctl start mysqld
# systemctl status mysqld
● mysqld.service - MySQL Server
   Loaded: loaded (/usr/lib/systemd/system/mysqld.service; enabled; vendor preset: disabled)
   Active: active (running) since date-and-time UTC; 5s ago
     Docs: man:mysqld(8)
           http://dev.mysql.com/doc/refman/en/using-systemd.html
 Main PID: 32399 (mysqld)
   Status: "SERVER_OPERATING"
    CGroupl: /system.slice/mysqld.service
              └─32399 /usr/local/mysql/bin/mysqld --defaults-
file=/etc/my.cnf
...
Aug 02 17:16:07 edvmr1p0 systemd[1]: Started MySQL Server.
Hint: Some lines were ellipsized, use -l to show in full.
```

22. Close any open terminal windows.

GANG LIU (gangli@baylorhealth.edu) has a non-transferable license
to use this Student Guide.

Practices for Lesson 12: Configuring a Replication Topology

Practices for Lesson 12: Overview

Overview

In these practices, you will test your knowledge of MySQL replication and configure a variety of replication topologies.

Assumptions

- You have created and populated the `employees` database in the practices for the lesson titled “Monitoring MySQL.”
- You have created the data directories `/mysql/data1`, `/mysql/data2`, `/mysql/data3`, and `/mysql/data4` in the practices for the lesson titled “Configuring MySQL.”

Practice 12-1: Quiz—Configuring Replication

Overview

In this practice, you answer questions about replication.

Duration

This practice should take you approximately five minutes to complete.

Quiz Questions

Choose the best answer from those provided for each multiple choice or True/False question.

1. Which of the following is true about MySQL master servers?
 - a. There is no limit on the number of slaves a single master can have.
 - b. It is possible for a slave to have a different MySQL version from the master.
 - c. It is common to limit the number of slaves to less than 30 in most production setups.
 - d. All of the above.
2. At its simplest, MySQL replication works through a one-way, log-shipping, asynchronous mechanism, making it a master-slave relationship.
 - a. True
 - b. False
3. Which of the following is a common use for replication?
 - a. Scale-out solutions
 - b. High availability
 - c. Analytics
 - d. All of the above
4. MySQL replication uses a log-shipping system in which all data changes that occur on the master are stored in a log and then retrieved by the slave and executed from these received log files. What is the name of this log file in MySQL?
 - a. Slave log
 - b. Master log
 - c. Binary log
 - d. Error log
5. Slaves need to be connected permanently to receive updates from the master.
 - a. True
 - b. False
6. Which of the following is a disadvantage of using statement-based replication?
 - a. Disk space usage and bandwidth requirements for replication are larger.
 - b. Replication occurs at the row level.
 - c. Some functions might not replicate correctly to a remote server.
 - d. None of the above

7. Which thread is responsible for downloading the binary logs from the master into a local file set called relay logs?
- a. BINARY_THREAD
 - b. IO_THREAD
 - c. SQL_THREAD
 - d. MASTER_THREAD

Solution 12-1: Quiz—Configuring Replication

Quiz Solutions

1. **d.** All of the above
2. **a.** True
3. **d.** All of the above
4. **c.** Binary log
5. **b.** False. Slaves do not need to be connected permanently to receive updates from the master.
6. **c.** Some functions might not replicate correctly to a remote server of a different version.
7. **b.** IO_THREAD

Practice 12-2: Configuring Replication

Overview

In this practice, you start four server instances of MySQL, configure one server as a slave of another, create some data on the master, and see that it replicates to the slave.

Duration

This practice should take you approximately 20 minutes to complete.

Tasks

1. In a Linux terminal window, enter a `systemctl` command to stop the MySQL server.
2. Examine the contents of the `/labs/multi.cnf` file that you used to launch multiple instances of `mysqld` in the “Running Multiple `mysqld` Instances on the Same Host with `systemd`” activity in the lesson titled “Configuring MySQL.”
3. Examine the contents of the `/labs/repl.cnf` file. This file enables you to run four servers in a replication topology. Note the differences between the configuration files in this step and the previous step.

Note: In this activity, you will use only `server1` (as the master) and `server2` (as the slave).
4. Modify the `/usr/lib/systemd/system/mysqld@.service` service unit configuration file to use `/labs/repl.cnf` as the default configuration file.
5. Execute `systemctl daemon-reload` to reload all the unit files and re-create the dependency tree.
6. Start the four servers in turn, using an appropriate `systemctl` command.
7. Use the Linux `ps` command to list the running `mysqld` processes.
8. In a new terminal window, use the `mysql` client to connect to the first server as `root`, and set the prompt to “`1>`”. This is `server1` for the remaining steps in this practice.

Note: For the sake of simplicity in a training environment, these server instances do not have a `root` password.
9. Execute a query to find the log coordinates of `server1`.
10. On `server1`, create a user called `repl`, with the password `oracle`, and grant that user the `REPLICATION SLAVE` permission. Ensure that `repl` uses the `mysql_native_password` authentication plugin instead of the default `caching_sha2_password`.
11. On `server1`, create the `world` database and populate it from the `/stage/databases/world.sql` script.

12. Open a new terminal window, use the `mysql` client to connect to the `server2` as `root`, and set the prompt to “`2>`”. This is `server2` for the remaining steps in this practice.
13. On `server2`, issue a `CHANGE MASTER TO` command to configure the second server as a slave of the first, using the log coordinates noted in step 9. Display the text of any warnings generated.
14. Display the databases that exist on `server2`.
15. Start the slave threads on `server2`.
16. Execute `SHOW PROCESSLIST` on `server1` and `server2` to display the processes running on each server.
17. Display the databases on `server2` again. Note the differences.
18. Leave the Linux terminal and `mysql` client session windows open for the next practice.

Solution 12-2: Configuring Replication

Solution Steps

1. In a Linux terminal window, enter a `systemctl` command to stop the MySQL server.

Enter the following commands at a Linux terminal prompt and receive the results shown:

```
# systemctl stop mysqld
# systemctl status mysqld
● mysqld.service - MySQL Server
   Loaded: loaded (/usr/lib/systemd/system/mysqld.service; enabled; vendor preset: disabled)
     Active: inactive (dead) since date-and-time UTC; 8s ago
       Docs: man:mysqld(8)
              http://dev.mysql.com/doc/refman/en/using-systemd.html
      Process: 32399 ExecStart=/usr/local/mysql/bin/mysqld --defaults-file=/etc/my.cnf $MYSQLD_OPTS (code=exited, status=0/SUCCESS)
     Main PID: 32399 (code=exited, status=0/SUCCESS)
        Status: "SERVER_SHUTTING_DOWN"
...
date-and-time edvmrlp0 systemd[1]: Stopped MySQL Server.
Hint: Some lines were ellipsized, use -l to show in full.
```

2. Examine the contents of the `/labs/multi.cnf` file that you used to launch multiple instances of `mysqld` in the “Running Multiple `mysqld` Instances on the Same Host with `systemd`” activity in the lesson titled “Configuring MySQL.”

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# cat /labs/multi.cnf
[mysqld@server1]
user=mysql
socket=/mysql/server1.sock
port=3311
datadir=/mysql/data1
log-error=/mysql/server1.err

[mysqld@server2]
user=mysql
socket=/mysql/server2.sock
port=3312
datadir=/mysql/data2
log-error=/mysql/server2.err

[mysqld@server3]
user=mysql
```

```
socket=/mysql/server3.sock
port=3313
datadir=/mysql/data3
log-error=/mysql/server3.err

[mysqld@server4]
user=mysql
socket=/mysql/server4.sock
port=3314
datadir=/mysql/data4
log-error=/mysql/server4.err
```

- The /labs/multi.cnf file configures four MySQL instances named server1, server2, server3, and server4 with their data directories and connection parameters.
3. Examine the contents of the /labs/repl.cnf file. This file enables you to run four servers in a replication topology. Note the differences between the configuration files in this step and the previous step.

Note: In this activity, you will use only server1 (as the master) and server2 (as the slave).

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# cat /labs/repl.cnf
mysqld@server1]
server-id=11
user=mysql
socket=/mysql/server1.sock
port=3311
datadir=/mysql/data1
log-error=/mysql/server1.err
log-bin=server1-bin
relay-log=server1-relay-bin
# gtid-mode=ON
# enforce-gtid-consistency

[mysqld@server2]
server-id=12
user=mysql
socket=/mysql/server2.sock
port=3312
datadir=/mysql/data2
log-error=/mysql/server2.err
log-bin=server2-bin
relay-log=server2-relay-bin
```

```
# gtid-mode=ON
# enforce-gtid-consistency

[mysqld@server3]
server-id=13
user=mysql
socket=/mysql/server3.sock
port=3313
datadir=/mysql/data3
log-error=/mysql/server3.err
log-bin=server3-bin
relay-log=server3-relay-bin
# gtid-mode=ON
# enforce-gtid-consistency

[mysqld@server4]
server-id=14
user=mysql
socket=/mysql/server4.sock
port=3314
datadir=/mysql/data4
log-error=/mysql/server4.err
log-bin=server4-bin
relay-log=server4-relay-bin
# gtid-mode=ON
# enforce-gtid-consistency
```

- Note that each server has a unique `server-id` to identify it within a replication topology.
- The server that will be the master (`server1`) must have binary logging enabled, which is the default behavior. The `log-bin` and `relay-log` entries provide custom names for these files.
- The `gtid-mode` and `enforce-gtid-consistency` configuration settings are commented out. You use these in the practice titled “Enabling GTID and Configuring Circular Replication” later in this lesson’s activities.

4. Modify the `/usr/lib/systemd/system/mysqld@.service` service unit configuration file to use `/labs/repl.cnf` as the default configuration file.

Edit the `/usr/lib/systemd/system/mysqld@.service` service unit configuration file as shown below:

```
[Unit]
Description=MySQL Multi Server for instance %i
Documentation=man:mysqld(8)
Documentation=http://dev.mysql.com/doc/refman/en/using-systemd.html
After=network.target
After=syslog.target

[Install]
WantedBy=multi-user.target

[Service]
User=mysql
Group=mysql
PIDFile=/mysql/server%I.pid
Type=notify

# Disable service start and stop timeout logic of systemd for mysqld
# service.
TimeoutSec=0

# Start main service
ExecStart=/usr/local/mysql/bin/mysqld --defaults-file=/labs/repl.cnf --
--defaults-group-suffix=@%I $MYSQLD_OPTS

# Use this to switch malloc implementation
EnvironmentFile=-/etc/sysconfig/mysql

# Sets open_files_limit
LimitNOFILE = 10000

Restart=on-failure

RestartPreventExitStatus=1

# Set environment variable MYSQLD_PARENT_PID. This is required for
# restart.
Environment=MYSQLD_PARENT_PID=1

PrivateTmp=false
```

- Execute `systemctl daemon-reload` to reload all the unit files and re-create the dependency tree.

Enter the following command at the Linux terminal prompt:

```
# systemctl daemon-reload
```

- Start the four servers in turn, using an appropriate `systemctl` command.

Enter the following commands at the Linux terminal prompt and receive the results shown:

```
# systemctl start mysqld@server1
# systemctl start mysqld@server2
# systemctl start mysqld@server3
# systemctl start mysqld@server4
```

- Use the Linux `ps` command to list the running `mysqld` processes.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# ps aux | grep mysqld
mysql      PID ... /usr/local/mysql/bin/mysqld --defaults-
           file=/labs/repl.cnf --defaults-group-suffix=@server1
mysql      PID ... /usr/local/mysql/bin/mysqld --defaults-
           file=/labs/repl.cnf --defaults-group-suffix=@server2
mysql      PID ... /usr/local/mysql/bin/mysqld --defaults-
           file=/labs/repl.cnf --defaults-group-suffix=@server3
mysql      PID ... /usr/local/mysql/bin/mysqld --defaults-
           file=/labs/repl.cnf --defaults-group-suffix=@server4
root      PID ... grep --color=auto mysqld
```

- The output of the `ps` command shows four running servers.

- In a new terminal window, use the `mysql` client to connect to the first server as `root`, and set the prompt to “`1>`”. This is `server1` for the remaining steps in this practice.

Note: For the sake of simplicity in a training environment, these server instances do not have a `root` password.

Enter the following command at a new Linux terminal prompt and receive the results shown:

```
# mysql -uroot -h127.0.0.1 -P3311
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
...
mysql> PROMPT 1> ;
PROMPT set to '1> '
```

9. Execute a query to find the log coordinates of server1.

At the server1 mysql prompt, enter the following statement and receive the results shown:

```
1> SHOW MASTER STATUS\G
*****
File: server1-bin.000001
Position: 155
Binlog_Do_DB:
Binlog_Ignore_DB:
Executed_Gtid_Set:
1 row in set (#.## sec)
```

- Note that the log file is `server1-bin.000001` and the log position is 155.

10. On server1, create a user called `repl`, with the password `oracle`, and grant that user the `REPLICATION SLAVE` permission. Ensure that `repl` uses the `mysql_native_password` authentication plugin instead of the default `caching_sha2_password`.

At the server1 mysql prompt, enter the following statement and receive the results shown:

```
1> CREATE USER 'repl'@'127.0.0.1'
-> IDENTIFIED WITH mysql_native_password BY 'oracle';
Query OK, 0 rows affected (#.## sec)

1> GRANT REPLICATION SLAVE ON *.* TO 'repl'@'127.0.0.1';
Query OK, 0 rows affected (#.## sec)
```

- The default authentication plugin for new users in MySQL 8.0 and later is `caching_sha2_password`, which requires a secure connection as described in the lesson titled “Securing MySQL.” To minimize the amount of configuration required in this practice, you will use `mysql_native_password`, which does not require a secure connection.

11. On server1, create the world database and populate it from the /stage/databases/world.sql script.

At the server1 mysql prompt, enter the following statements and receive the results shown:

```
1> CREATE DATABASE world;
Query OK, 1 row affected (#.## sec)

1> USE world
Database changed

1> SOURCE /stage/databases/world.sql
Query OK, 1 row affected (#.## sec)

Query OK, 1 row affected (#.## sec)
...
```

12. Open a new terminal window, use the mysql client to connect to server2 as root, and set the prompt to “2>”. This is server2 for the remaining steps in this practice.

Enter the following commands at a new Linux terminal prompt and receive the results shown:

```
# mysql -uroot -h127.0.0.1 -P3312
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 1
...
mysql> PROMPT 2> ;
PROMPT set to '2> '
2>
```

13. On server2, issue a CHANGE MASTER TO command to configure the second server as a slave of the first, using the log coordinates noted in step 9. Display the text of any warnings generated.

At the server2 mysql prompt, enter the following statement and receive the results shown:

```
2> CHANGE MASTER TO
-> MASTER_HOST='127.0.0.1' ,
-> MASTER_PORT=3311 ,
-> MASTER_LOG_FILE='server1-bin.000001' ,
-> MASTER_LOG_POS=155 ;
Query OK, 0 rows affected (#.## sec)
```

14. Display the databases that exist on server2.

At the server2 mysql prompt, enter the following statement and receive the results shown:

```
2> SHOW DATABASES;
+-----+
| Database      |
+-----+
| information_schema |
| mysql          |
| performance_schema |
| sys            |
+-----+
4 rows in set (#.## sec)
```

- You installed the world database on server1. It is not installed on server2.

15. Start the slave threads on server2.

At the server2 mysql prompt used in the preceding step, enter the following statement and receive the results shown:

```
2> START SLAVE USER='repl' PASSWORD='oracle';
Query OK, 0 rows affected (#.## sec)
```

16. Execute SHOW PROCESSLIST on server1 and server2 to display the processes running on each server.

- At the server1 mysql prompt, issue the following statement and receive the results shown:

```
1> SHOW PROCESSLIST\G
...
***** 3. row *****
    Id: 14
    User: repl
    Host: localhost:47661
    db: NULL
    Command: Binlog Dump
    Time: 93
    State: Master has sent all binlog to slave; waiting for more updates
    Info: NULL
3 rows in set (#.## sec)
```

- b. At the server2 mysql prompt, enter the following statement and receive the results shown:

```
2> SHOW PROCESSLIST\G
...
***** 3. row *****
    Id: 12
    User: system user
    Host:
    db: NULL
Command: Connect
    Time: 140
    State: Waiting for master to send event
    Info: NULL
***** 4. row *****
    Id: 13
    User: system user
    Host:
    db: NULL
Command: Query
    Time: 239
    State: Slave has read all relay log; waiting for more updates
    Info: NULL
4 rows in set (#.## sec)
```

17. Display the databases on server2 again. Note the differences.

At the server2 mysql prompt, enter the following statement and receive the results shown:

```
2> SHOW DATABASES;
+-----+
| Database      |
+-----+
| information_schema |
| mysql          |
| performance_schema |
| sys            |
| world         |
+-----+
5 rows in set (#.## sec)
```

- The **world** database is in the list, because it was created on server1 after you noted the log coordinates in step 9. The slave process on server2 replicated every change after those coordinates, including all data in the **world** database, and the user created in step 10.

18. Leave the Linux terminal and mysql client session windows open for the next practice.

Practice 12-3: Adding a New Slave

Overview

In this practice, you provision a new server as a new slave of the existing slave (using mysqldump), change some data on the master, and see that it replicates to both slaves.

Duration

This practice should take you approximately ten minutes to complete.

Tasks

1. Using mysqldump, take a backup of the world database on server2, including the information needed to create and use the database, and to configure a slave.

Enter the following command at a Linux terminal prompt:

```
# mysqldump -uroot -h127.0.0.1 -P3312 --master-data=2 \  
 > -B world > /tmp/server2.sql
```

Note: The --master-data=2 option writes a CHANGE MASTER TO statement to the dump file that is commented out, enabling you to make changes before using the dump file. Setting --master-data=1 writes an uncommented version of the statement which executes when you load the dump file.

2. Use a text editor *other than* gedit, (such as nano, emacs, or vi), to edit the backup file created in the preceding step. Modify the CHANGE MASTER TO ... line so that it points to the second server, and is uncommented.

Note: Do not use gedit to edit the /tmp/server2.sql file. The gedit text editor might fail due to a bug in the version used in Oracle classroom environments.

3. Open a new terminal window, connect to the third server using the mysql client, and set the prompt to "3> ". This is server3 for the remaining steps in this practice.
4. Apply the backup taken in step 1 and modified in step 2 to the third server.
5. Start the slave process on the third server, and display the slave status.
6. On server1, delete all rows where the ID is greater than 4070 from the city table, and verify that the changes replicate to server2 and server3.
7. To prepare for the next practice, create a user on server3 called repl, with the password oracle, and grant the REPLICATION SLAVE permission.
8. Leave the Linux terminal and mysql client session windows open for the next practice.

Solution 12-3: Adding a New Slave

Solution Steps

1. Using mysqldump, take a backup of the world database on server2, including the information needed to create and use the database, and to configure a slave.

Enter the following command at a Linux terminal prompt:

```
# mysqldump -uroot -h127.0.0.1 -P3312 --master-data=2 \
> -B world > /tmp/server2.sql
```

Note: The --master-data=2 option writes a CHANGE MASTER TO statement to the dump file that is commented out, enabling you to make changes before using the dump file. Setting --master-data=1 writes an uncommented version of the statement, which executes when you load the dump file.

2. Use a text editor *other than* gedit (such as nano, emacs, or vi) to edit the backup file created in the preceding step. Modify the CHANGE MASTER TO ... line so that it points to the second server, and is uncommented.

Note: Do not use gedit to edit the /tmp/server2.sql file. The gedit text editor might fail due to a bug in the version used in Oracle classroom environments.

Using an editor such as nano, emacs, or vim, open the /tmp/server2.sql file and locate the following line:

```
-- CHANGE MASTER TO MASTER_LOG_FILE='server2-bin.000001',
MASTER_LOG_POS=723074;
```

Change that line as follows, and save the file before exiting the editor:

```
CHANGE MASTER TO MASTER_HOST='127.0.0.1', MASTER_PORT=3312,
MASTER_LOG_FILE='server2-bin.000001', MASTER_LOG_POS=723074;
```

- The log coordinates and binary log file name might be different from that shown in your output.
- The replication user `rep1` on server2 was created on server1 and replicated to server2 in the preceding practice.

3. Open a new terminal window, connect to the third server using the mysql client, and set the prompt to “3>”. This is server3 for the remaining steps in this practice.

Enter the following command at the new Linux terminal prompt and receive the results shown:

```
# mysql -uroot -h127.0.0.1 -P3313
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 1
...
```

Enter the following statement at the mysql prompt and receive the results shown:

```
mysql> PROMPT 3> ;
PROMPT set to '3> '
3>
```

4. Apply the backup taken in step 1 and modified in step 2 to the third server.

At the server3 mysql prompt, enter the following command and receive the results shown:

```
3> SOURCE /tmp/server2.sql
Query OK, 0 rows affected (#.## sec)

Query OK, 0 rows affected (#.## sec)

Query OK, 0 rows affected (#.## sec)

...
```

5. Start the slave process on the third server, and display the slave status.

In the mysql prompt used in the preceding step, enter the following commands and receive the results shown:

```
3> START SLAVE USER='repl' PASSWORD='oracle';
Query OK, 0 rows affected, 1 warning (#.## sec)

3> SHOW SLAVE STATUS\G
***** 1. row *****
Slave_IO_State: Waiting for master to send event
Master_Host: 127.0.0.1
Master_User: repl
Master_Port: 3312
Connect_Retry: 60
Master_Log_File: server2-bin.000001
Read_Master_Log_Pos: 722644
Relay_Log_File: server3-relay-bin.000002
Relay_Log_Pos: 722813
Relay_Master_Log_File: server2-bin.000001
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
...
Exec_Master_Log_Pos: 722644
...
Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
Last_IO_Erno: 0
Last_IO_Error:
Last_SQL_Erno: 0
Last_SQL_Error:
Replicate_Ignore_Server_Ids:
Master_Server_Id: 12
```

```

Master_UUID: 471e248c-707f-11e8-acc4-
aabb00018482
Master_Info_File: mysql.slave_master_info
SQL_Delay: 0
SQL_Remaining_Delay: NULL
Slave_SQL_Running_State: Slave has read all relay log;
waiting for more updates
Master_Retry_Count: 86400
Master_Bind:
Last_IO_Error_Timestamp:
Last_SQL_Error_Timestamp:
Master_SSL_Crl:
Master_SSL_Crlpath:
Retrieved_Gtid_Set:
Executed_Gtid_Set:
Auto_Position: 0
Replicate_Rewrite_DB:
Channel_Name:
Master_TLS_Version:
Master_public_key_path:
Get_master_public_key: 0
1 row in set (#.## sec)

```

6. On server1, delete all rows where the ID is greater than 4070 from the city table, and verify that the changes replicate to server2 and server3.

- a. On server1, enter the following statement at the mysql prompt and receive the results shown:

```

1> DELETE FROM world.city WHERE ID > 4070;
Query OK, 9 rows affected (#.## sec)

```

- b. On server2, enter the following statement at the mysql prompt and receive the results shown:

```

2> SELECT ID, Name FROM world.city
    -> ORDER BY ID DESC LIMIT 5;
+----+-----+
| ID      | Name           |
+----+-----+
| 4070    | Chitungwiza   |
| 4069    | Bulawayo       |
| 4068    | Harare         |
| 4067    | Charlotte Amalie |
| 4066    | Charleston     |
+----+-----+
5 rows in set (#.## sec)

```

- c. On server3, enter the following statement at the mysql prompt and receive the results shown:

```
3> SELECT ID, Name FROM world.city
   -> ORDER BY ID DESC LIMIT 5;
+-----+-----+
| ID   | Name      |
+-----+-----+
| 4070 | Chitungwiza |
| 4069 | Bulawayo   |
| 4068 | Harare    |
| 4067 | Charlotte Amalie |
| 4066 | Charleston |
+-----+
5 rows in set (#.# sec)
```

- The changes you made to the city table on the first server have replicated to the second and third servers.
7. To prepare for the next practice, create a user on server3 called repl, with the password oracle, and grant the REPLICATION SLAVE permission.

On server3, enter the following statement at the mysql prompt and receive the results shown:

```
3> CREATE USER 'repl'@'127.0.0.1'
   -> IDENTIFIED WITH mysql_native_password BY 'oracle';
Query OK, 0 rows affected (#.# sec)

3> GRANT REPLICATION SLAVE ON *.* TO 'repl'@'127.0.0.1';
Query OK, 0 rows affected (#.# sec)
```

- At this point, the repl user exists on three servers, having been created manually on the first server, replicated to the second, and again created manually on the third.

Note: The repl user did not replicate to the third server; it was created on its master server—the second server—before the point at which the third server started to replicate. The backup that you restored to the third server contained information only from the world database.

8. Leave the Linux terminal and mysql client session windows open for the next practice.

Practice 12-4: Enabling GTID and Configuring Circular Replication

Overview

In this practice, you enable GTID on the three servers, connect the master so that it becomes a slave of the second slave, and test the newly created circular topology by changing some data.

Duration

This practice should take you approximately 15 minutes to complete.

Tasks

1. Use `systemctl` to stop all running servers.
2. Edit the `/labs/repl.cnf` file to uncomment the following GTID configuration for each server:

```
gtid-mode=ON
enforce-gtid-consistency
```
3. Use `systemctl` to start `server1`, `server2`, and `server3` in turn.
4. Ensure that `server1`, `server2`, and `server3` are running before continuing.
5. On `server2` and `server3`, enter a command such as `STATUS` to re-establish the connection and stop the slave threads.
6. Issue `RESET MASTER` commands on `server1`, `server2`, and `server3` so that the log files contain only events that use GTIDs.
7. Issue appropriate `CHANGE MASTER TO...` commands on `server2` and `server3` to use the GTID replication protocol.
8. Start the slave threads on `server2` and `server3`.
9. On `server1`, delete all rows where the ID is greater than 4060 from the `city` table and ensure that the changes replicate to `server2` and `server3`.
10. Note the server UUIDs for the first, second, and third servers.
11. On `server3`, view the slave status.
12. Issue an appropriate `CHANGE MASTER TO...` statement on `server1`, configuring it as a slave to `server3`, and start the slave threads.
13. On `server2`, delete all rows where the ID is greater than 4050 from the `city` table and ensure that the changes replicate to `server1` and `server3`.
14. Exit all `mysql` client sessions and close all open terminal windows.

Solution 12-4: Enabling GTID and Configuring Circular Replication

Solution Steps

1. Use `systemctl` to stop all running servers.

Enter the following commands at the Linux terminal prompt and receive the results shown:

```
# systemctl stop mysqld@server*
# systemctl status mysqld@server*
#
```

- In the sample output, `systemctl status` shows no running `mysql@server*` instances.

2. Edit the `/labs/repl.cnf` file to uncomment the following GTID configuration for each server:

```
gtid-mode=ON
enforce-gtid-consistency
```

Make the following changes to the `/labs/repl.cnf` file:

```
[mysqld@server1]
server-id=11
user=mysql
socket=/mysql/socket1
port=3311
datadir=/mysql/data1
log-error=/mysql/err1
log-bin=server1-bin
relay-log=server1-relay-bin
gtid-mode=ON
enforce-gtid-consistency

[mysqld@server2]
server-id=12
user=mysql
socket=/mysql/socket2
port=3312
datadir=/mysql/data2
log-error=/mysql/err2
log-bin=server2-bin
gtid-mode=ON
enforce-gtid-consistency

[mysqld@server3]
server-id=13
user=mysql
```

```

socket=/mysql/socket3
port=3313
datadir=/mysql/data3
log-error=/mysql/err3
log-bin=server3-bin
gtid-mode=ON
enforce-gtid-consistency

mysqld@server4]
server-id=14
user=mysql
socket=/mysql/socket4
port=3313
datadir=/mysql/data4
log-error=/mysql/err4
log-bin=server4-bin
gtid-mode=ON
enforce-gtid-consistency

```

- **Note:** This practice does not use server4. Make the changes to server4 for a later practice.
3. Use `systemctl` to start server1, server2, and server3 in turn:

Enter the following commands at the Linux terminal prompt:

```

# systemctl start mysqld@server1
# systemctl start mysqld@server2
# systemctl start mysqld@server3

```

4. Ensure that server1, server2, and server3 are running before continuing.

Enter the following command at the Linux terminal prompt and receive the results shown:

```

# systemctl status mysqld@server*
● mysqld@server3.service - MySQL Multi Server for instance
server3
   Loaded: loaded (/usr/lib/systemd/system/mysqld@.service;
enabled; vendor preset: disabled)
   Active: active (running) since ...
             Loaded: loaded (/usr/lib/systemd/system/mysqld@.service;
enabled; vendor preset: disabled)
             Active: active (running) since ...

```

```
● mysql@server2.service - MySQL Multi Server for instance
server2
   Loaded: loaded (/usr/lib/systemd/system/mysql@.service;
enabled; vendor preset: disabled)
     Active: active (running) since ...

```

5. On server2 and server3, enter a command such as STATUS to re-establish the connection and stop the slave threads.

On server2, enter the following statements at the mysql prompt and receive the results shown:

```
2> STATUS
ERROR 2006 (HY000): MySQL server has gone away
No connection. Trying to reconnect...

2> STOP SLAVE;
Query OK, 0 rows affected (#.## sec)
```

On server3, enter the following statement at the mysql prompt and receive the results shown:

```
3> STATUS
ERROR 2006 (HY000): MySQL server has gone away
No connection. Trying to reconnect...

3> STOP SLAVE;
Query OK, 0 rows affected (#.## sec)
```

6. Issue a RESET MASTER command on server1, server2, and server3 so that the log files contain only events that use GTIDs.

- a. On server1, issue the following statements and receive the results shown:

```
1> STATUS
ERROR 2006 (HY000): MySQL server has gone away
No connection. Trying to reconnect...

1> RESET MASTER;
Query OK, 0 rows affected (#.## sec)
```

- b. On server2, issue the following statement and receive the result shown:

```
2> RESET MASTER;
Query OK, 0 rows affected (#.## sec)
```

- c. On server3, issue the following statement and receive the result shown:

```
3> RESET MASTER;
Query OK, 0 rows affected (#.## sec)
```

7. Issue an appropriate CHANGE MASTER TO... command on server2 and server3 to use the GTID replication protocol.

- a. On server2, issue the following statement and receive the result shown:

```
2> CHANGE MASTER TO MASTER_AUTO_POSITION=1;
Query OK, 0 rows affected (#.## sec)
```

- b. On server3, issue the following statement and receive the result shown:

```
3> CHANGE MASTER TO MASTER_AUTO_POSITION=1;
Query OK, 0 rows affected (#.## sec)
```

8. Start the slave threads on server2 and server3.

- a. On server2, issue the following statement and receive the result shown:

```
2> START SLAVE USER='rep1' PASSWORD='oracle';
Query OK, 0 rows affected (#.## sec)
```

- b. On server3, issue the following statement and receive the result shown:

```
3> START SLAVE USER='rep1' PASSWORD='oracle';
Query OK, 0 rows affected (#.## sec)
```

9. On server1, delete all rows where the ID is greater than 4060 from the city table and ensure that the changes replicate to server2 and server3.

- a. On server1, enter the following statement and receive the result shown:

```
1> DELETE FROM world.city WHERE ID > 4060;
Query OK, 10 rows affected (#.## sec)
```

- b. On server2, enter the following statement and receive the result shown:

```
2> SELECT * FROM world.city ORDER BY ID DESC LIMIT 5;
+----+-----+-----+-----+
| ID | Name      | CountryCode | District      | Population |
+----+-----+-----+-----+
| 4060 | Santa Monica | USA          | California    | 91084     |
| 4059 | Cary        | USA          | North Carolina | 91213     |
| 4058 | Boulder     | USA          | Colorado      | 91238     |
| 4057 | Visalia     | USA          | California    | 91762     |
| 4056 | San Mateo   | USA          | California    | 91799     |
+----+-----+-----+-----+
5 rows in set (#.## sec)
```

- c. On server3, enter the following statement and receive the results shown:

```
3> SELECT * FROM world.city ORDER BY ID DESC LIMIT 5;
+-----+-----+-----+-----+
| ID   | Name      | CountryCode | District      | Population |
+-----+-----+-----+-----+
| 4060 | Santa Monica | USA          | California    | 91084      |
| 4059 | Cary        | USA          | North Carolina | 91213      |
| 4058 | Boulder     | USA          | Colorado      | 91238      |
| 4057 | Visalia     | USA          | California    | 91762      |
| 4056 | San Mateo   | USA          | California    | 91799      |
+-----+-----+-----+-----+
5 rows in set (#.# sec)
```

- The changes you made to the city table on server1 have replicated to server2 and server3.

10. Note the server UUIDs for the first, second, and third servers.

On server1, enter the following statement and receive a result similar to that shown:

```
1> SELECT @@server_uuid;
+-----+
| @@server_uuid           |
+-----+
| e34aab07-d97b-11e6-ac55-3417eb99981d |
+-----+
1 row in set (#.# sec)
```

On server2, enter the following statement and receive a result similar to that shown:

```
2> SELECT @@server_uuid;
+-----+
| @@server_uuid           |
+-----+
| f67acd3c-d97b-11e6-aecd-3417eb99981d |
+-----+
1 row in set (#.# sec)
```

On server3, enter the following statement and receive a result similar to that shown:

```
3> SELECT @@server_uuid;
+-----+
| @@server_uuid           |
+-----+
| 06bd7cbe-d97c-11e6-b09a-3417eb99981d |
+-----+
1 row in set (#.# sec)
```

Note: Your UUIDs are different from those shown; the server UUID is unique by design.

11. On server3, view the slave status.

On server3, enter the following statement and receive a result similar to that shown:

```
3> SHOW SLAVE STATUS\G
***** 1. row *****
Slave_IO_State: Waiting for master to send event
Master_Host: 127.0.0.1
Master_User: repl
Master_Port: 3312
Connect_Retry: 60
Master_Log_File: mysql2-bin.000001
Read_Master_Log_Pos: 729
Relay_Log_File: mysql3-relay-bin.000002
Relay_Log_Pos: 944
Relay_Master_Log_File: mysql2-bin.000001
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
Replicate_Do_DB:
Replicate_Ignore_DB:
Replicate_Do_Table:
Replicate_Ignore_Table:
Replicate_Wild_Do_Table:
Replicate_Wild_Ignore_Table:
Last_Error:
Skip_Counter: 0
Exec_Master_Log_Pos: 729
Relay_Log_Space: 1152
Until_Condition: None
Until_Log_File:
Until_Log_Pos: 0
Master_SSL_Allowed: No
Master_SSL_CA_File:
Master_SSL_CA_Path:
Master_SSL_Cert:
Master_SSL_Cipher:
Master_SSL_Key:
Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
Last_IO_Errno: 0
Last_IO_Error:
Last_SQL_Errno: 0
Last_SQL_Error:
Replicate_Ignore_Server_Ids:
Master_Server_Id: 2
Master_UUID: f67acd3c-d97b-11e6-aecd-3417eb99981d
```

```

Master_Info_File: mysql.slave_master_info
      SQL_Delay: 0
SQL_Remaining_Delay: NULL
Slave_SQL_Running_State: Slave has read all relay log; waiting
for more updates
      Master_Retry_Count: 86400
      Master_Bind:
Last_IO_Error_Timestamp:
Last_SQL_Error_Timestamp:
      Master_SSL_Crl:
      Master_SSL_Crlpath:
Retrieved_Gtid_Set: e34aab07-d97b-11e6-ac55-3417eb99981d:1
Executed_Gtid_Set: e34aab07-d97b-11e6-ac55-3417eb99981d:1
      Auto_Position: 1
Replicate_Rewrite_DB:
      Channel_Name:
      Master_TLS_Version:
1 row in set (#.## sec)

```

- Note that `Master_UUID` identifies server2, but the GTID for the data modification in step 9 identifies the first server.
12. Issue an appropriate `CHANGE MASTER TO...` statement on server1, configuring it as a slave to server3, and start the slave threads.

On the first server, enter the following statements and receive the results shown:

```

1> CHANGE MASTER TO MASTER_HOST='127.0.0.1',
   -> MASTER_PORT=3313,
   -> MASTER_AUTO_POSITION=1;
Query OK, 0 rows affected, 1 warning (#.## sec)

1> START SLAVE USER='rep1' PASSWORD='oracle';
Query OK, 0 rows affected (#.## sec)

```

13. On server2, delete all rows where the ID is greater than 4050 from the `city` table and ensure that the changes replicate to server1 and server3.

- a. On server2, enter the following statement and receive the results shown:

```

2> DELETE FROM world.city WHERE ID > 4050;
Query OK, 10 rows affected (#.## sec)

```

- b. On server1, enter the following statement and receive the results shown:

```
1> SELECT * FROM world.city ORDER BY ID DESC LIMIT 5;
+----+-----+-----+-----+
| ID | Name | CountryCode | District | Population |
+----+-----+-----+-----+
| 4050 | Roanoke | USA | Virginia | 93357 |
| 4049 | Brockton | USA | Massachusetts | 93653 |
| 4048 | Albany | USA | New York | 93994 |
| 4047 | Richmond | USA | California | 94100 |
| 4046 | Norman | USA | Oklahoma | 94193 |
+----+-----+-----+-----+
5 rows in set (#.## sec)
```

- c. On server3, enter the following statement and receive the results shown:

```
3> SELECT * FROM world.city ORDER BY ID DESC LIMIT 5;
+----+-----+-----+-----+
| ID | Name | CountryCode | District | Population |
+----+-----+-----+-----+
| 4050 | Roanoke | USA | Virginia | 93357 |
| 4049 | Brockton | USA | Massachusetts | 93653 |
| 4048 | Albany | USA | New York | 93994 |
| 4047 | Richmond | USA | California | 94100 |
| 4046 | Norman | USA | Oklahoma | 94193 |
+----+-----+-----+-----+
5 rows in set (#.## sec)
```

– Circular replication is in effect, and a change on server2 replicated back to server1 via server3.

14. Exit all mysql client sessions and close all open terminal windows.

Practices for Lesson 13: Administering a Replication Topology

Practices for Lesson 13: Overview

Overview

In these practices, you will perform tasks associated with administering an existing replication setup. You will monitor and troubleshoot the replication.

Assumptions

- You have configured a circular replication by using GTID on server1, server2, and server3 in the practices for the lesson titled “Configuring a Replication Topology.”
- You have installed the MySQL Enterprise Monitor in the practices for the lesson titled “Monitoring MySQL.”

Practice 13-1: Monitoring Replication with MySQL Enterprise Monitor

Overview

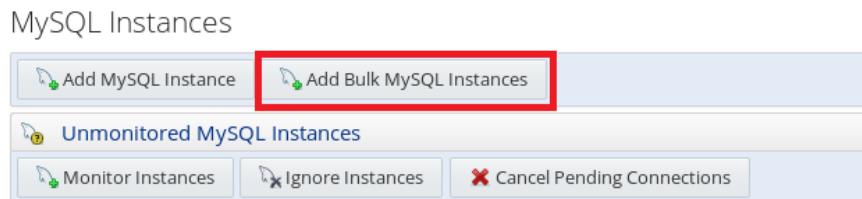
In this practice, you configure MySQL Enterprise Monitor to monitor an existing replication and view the information available in Replication Dashboard.

Duration

This practice should take you approximately 15 minutes to complete.

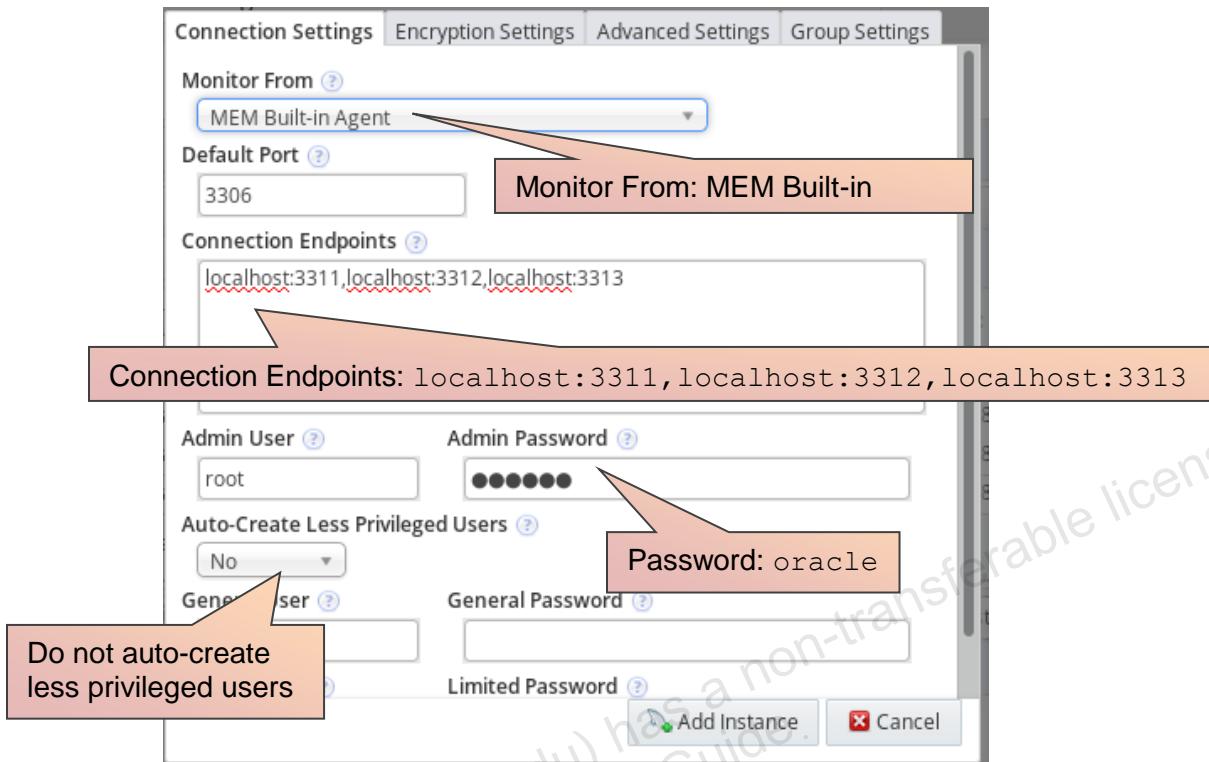
Tasks

1. Check the status of the MySQL Enterprise Monitor service and start it if necessary.
2. Ensure that `server1`, `server2`, and `server3` are running by executing `systemctl status mysql@server*` on a terminal window.
3. Open three new terminal windows. In the first terminal window, use the `mysql` client to connect to `server1` as the `root` user, and set the prompt to `1>`. Change the password of the `root` user to `oracle`, the change should replicate to `server2` and `server3`. In the second terminal window, use the `mysql` client to connect to `server2` as the `root` user with the password `oracle`, and set the prompt to `2>`. In the third terminal window, use the `mysql` client to connect to `server3` as the `root` user with the password `oracle`, and set the prompt to `3>`.
4. Open the MySQL Enterprise Monitor dashboard in the Firefox web browser by using the bookmark you created for it in the practices for the lesson titled “Monitoring MySQL.” Log in as the manager user `memmanager`, with the password `oracle`.
5. Add the three MySQL server instances to be monitored. Open the MySQL Instances page by clicking the Configuration > MySQL Instances link in the menu on the left of the Dashboard.
 - a. Click the Add Bulk MySQL Instances button near the top.

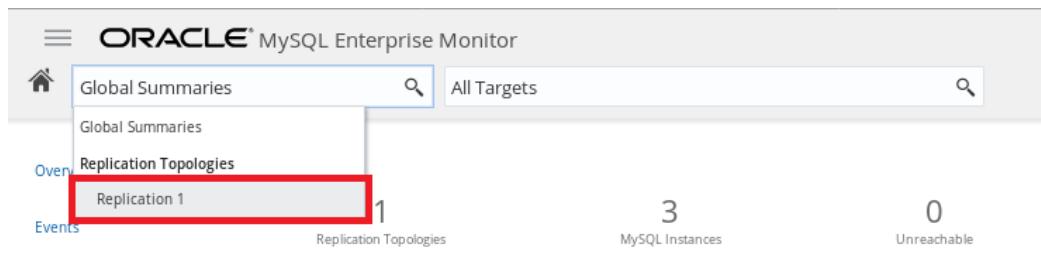


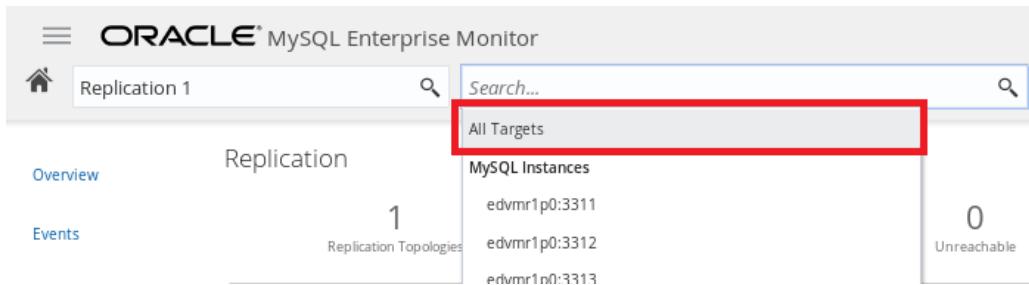
- b. Select MEM Built-in Agent in the Monitor From selection box.
- c. Type `localhost:3311,localhost:3312,localhost:3313` in the Connection Endpoints box.
- d. Type the password `oracle` in the Admin Password box.
- e. Select No in the Auto-Create Less Privileged Users selection box.

- f. Click Add Instance.



6. Open the Replication Dashboard. From the left navigation menu, select “Replication.” The screen shows an overview of all the replication topologies monitored by MySQL Enterprise Monitor. It may take a few minutes to discover the replication topology of the newly added instances. If the Replication Dashboard is empty, click the Refresh button on the top right of the page. Identify the following:
 - a. How many replication topologies are there?
 - b. How many MySQL Instances are there?
 - c. Are there any unreachable instances?
 - d. Are there any replication errors?
 - e. Are there any stopped instances?
7. View the details of a replication topology. From the Global Summaries selection box, select Replication 1. Then select All Targets. The screen shows the replication dashboard for a single replication topology. Check the information available on the three tabs; status, statistics, and error history by clicking them.





8. View the replication topology in a graphical form. From the left navigation menu, click “Topology”. What is the topology of the replication?
9. Keep the three mysql client connections and browser open for the next practice.

Solution 13-1: Monitoring Replication with MySQL Enterprise Monitor

Solution Steps

1. Check the status of the MySQL Enterprise Monitor service and start it if necessary. Enter the following commands at a new Linux terminal prompt and receive the results shown:

```
# sh /opt/mysql/enterprise/monitor/mysqlmonitorctl.sh status
MySQL Enterprise MySQL is not running
MySQL Enterprise Tomcat is not running

# sh /opt/mysql/enterprise/monitor/mysqlmonitorctl.sh start
Starting mysql service [ OK ]
date-and-time mysqld_safe Logging to
'/opt/mysql/enterprise/monitor/mysql/runtime/mysqld.log'.
date-and-time mysqld_safe Starting mysqld daemon with databases
from /opt/mysql/enterprise/monitor/mysql/data/
Starting tomcat service [ OK ]

# sh /opt/mysql/enterprise/monitor/mysqlmonitorctl.sh status
MySQL Enterprise MySQL is running
MySQL Enterprise Tomcat is running
```

2. Ensure that server1, server2, and server3 are running by executing `systemctl status mysqld@server*` on a terminal window.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# systemctl status mysqld@server*
● mysqld@server3.service - MySQL Multi Server for instance
server3
    Loaded: loaded (/usr/lib/systemd/system/mysqld@.service;
enabled; vendor preset: disabled)
    Active: active (running) since ...
              Loaded: loaded (/usr/lib/systemd/system/mysqld@.service;
enabled; vendor preset: disabled)
              Active: active (running) since ...
● mysqld@server1.service - MySQL Multi Server for instance
server1
    Loaded: loaded (/usr/lib/systemd/system/mysqld@.service;
enabled; vendor preset: disabled)
    Active: active (running) since ...
● mysqld@server2.service - MySQL Multi Server for instance
server2
    Loaded: loaded (/usr/lib/systemd/system/mysqld@.service;
enabled; vendor preset: disabled)
    Active: active (running) since ...
```

3. Open three new terminal windows. In the first terminal window, use the `mysql` client to connect to `server1` as the `root` user, and set the prompt to `1>`. Change the password of the `root` user to `oracle`, the change should replicate to `server2` and `server3`. In the second terminal window, use the `mysql` client to connect to `server2` as the `root` user with the password `oracle`, and set the prompt to `2>`. In the third terminal window, use the `mysql` client to connect to `server3` as the `root` user with the password `oracle`, and set the prompt to `3>`.

Enter the following command at the first Linux terminal prompt and receive the results shown:

```
# mysql -uroot -h127.0.0.1 -P3311
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 1
...
mysql> PROMPT 1> ;
PROMPT set to '1> '
1> ALTER USER USER() IDENTIFIED BY 'oracle';
Query OK, 0 rows affected (#.## sec)
```

Enter the following command at the second Linux terminal prompt and receive the results shown:

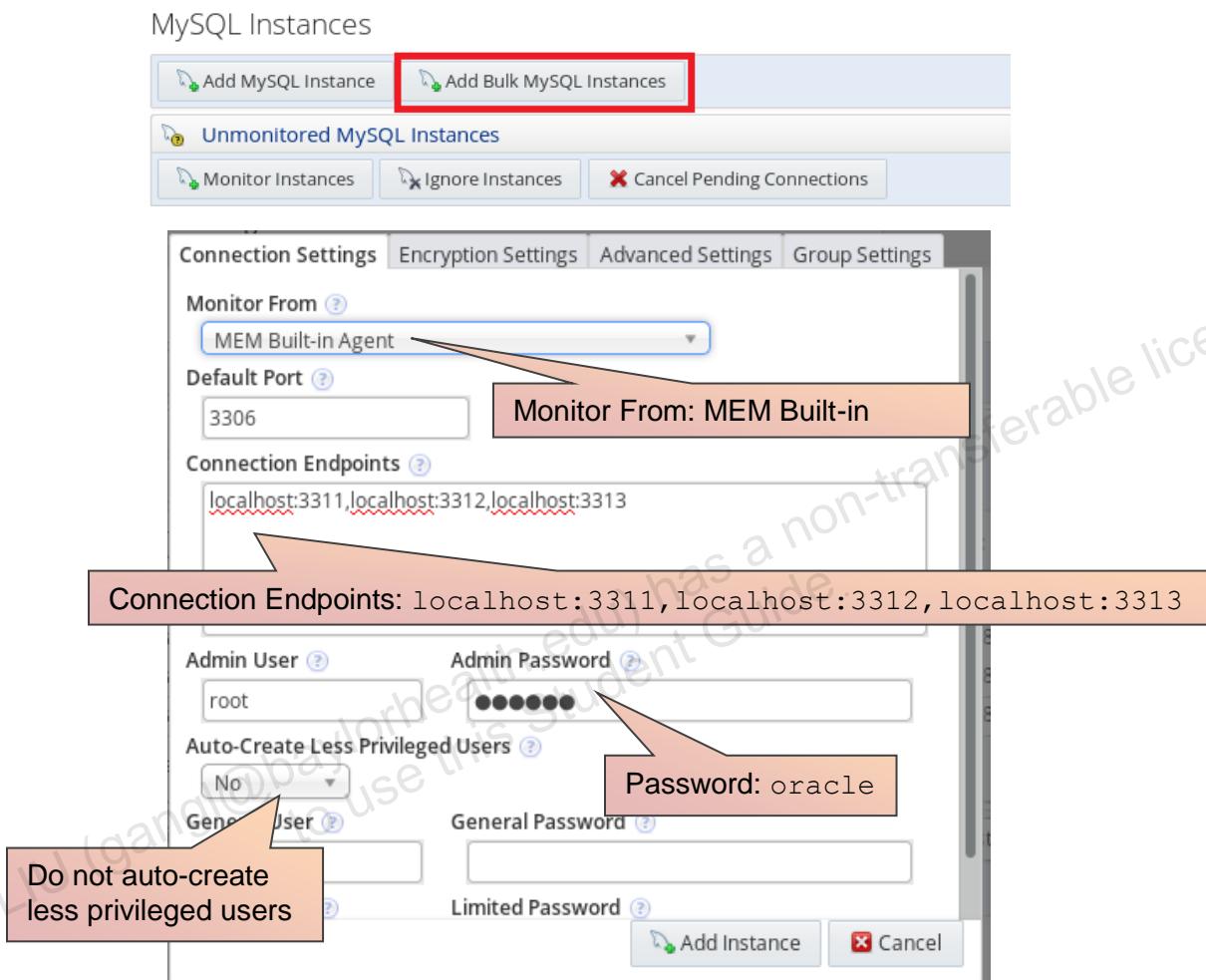
```
# mysql -uroot -h127.0.0.1 -P3312 -p
Enter password: oracle
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 1
...
mysql> PROMPT 2> ;
PROMPT set to '2> '
2>
```

Enter the following command at the third Linux terminal prompt and receive the results shown:

```
# mysql -uroot -h127.0.0.1 -P3313 -p
Enter password: oracle
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 1
...
mysql> PROMPT 3> ;
PROMPT set to '3> '
3>
```

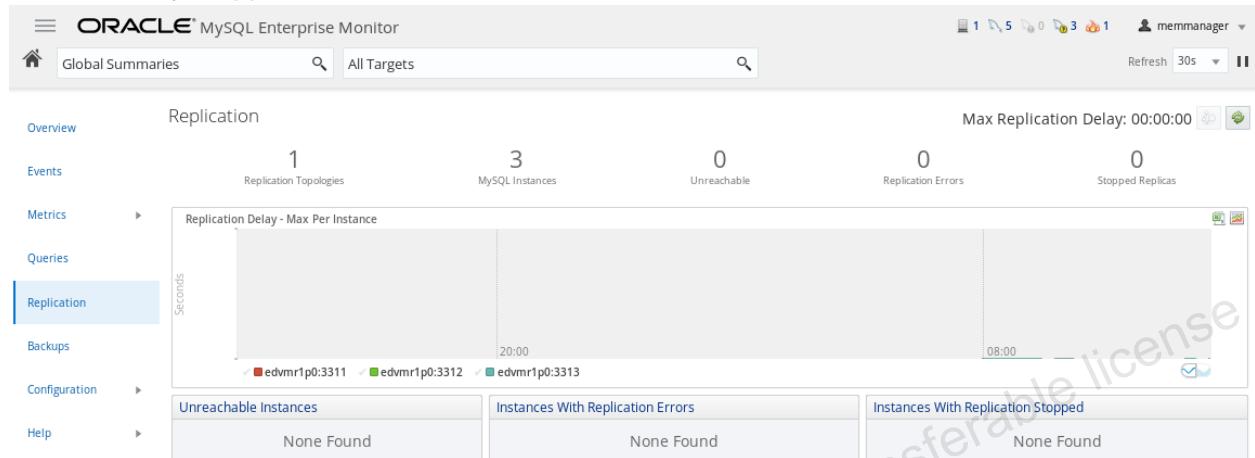
4. Open the MySQL Enterprise Monitor dashboard in the Firefox web browser by using the bookmark you created for it in the practices for the lesson titled “Monitoring MySQL.” Log in as the manager user `memmanager`, with the password `oracle`.

5. Add the three MySQL server instances to be monitored. Open the MySQL Instances page by clicking the Configuration > MySQL Instances link in the menu on the left side of the Dashboard.
- Click the Add Bulk MySQL Instances button near the top.

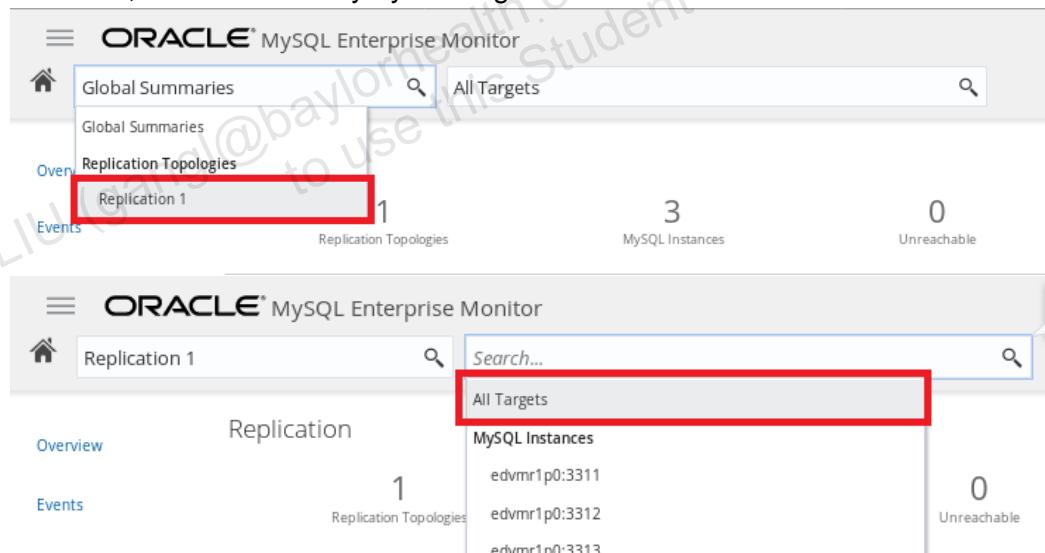


- Select MEM Built-in Agent in the Monitor From selection box.
 - Type localhost:3311,localhost:3312,localhost:3313 into the Connection Endpoints box.
 - Type the password oracle in the Admin Password box.
 - Select No in the Auto-Create Less Privileged Users selection box.
 - Click Add Instance.
6. Open the Replication Dashboard. From the left-navigation menu, click “Replication.” The screen shows an overview of all the replication topologies monitored by MySQL Enterprise Monitor. It may take a few minutes to discover the replication topology of the newly added instances. If the Replication Dashboard is empty, click the Refresh button on the top right of the page. Identify the following:
- How many replication topologies are there? 1

- b. How many MySQL Instances are there? **3**
- c. Are there any unreachable instances? **None**
- d. Are there any replication error? **None**
- e. Are there any stopped instances? **None**



7. View the details of a replication topology. Click the Global Summaries selection box and select Replication 1. Then, select All Targets. The screen shows the replication dashboard for a single replication topology. Check the information available on the three tabs: Status, Statistics, and Error History by clicking them.



This is an example of the Status tab. If you expand the row by clicking the plus symbol on the left, it shows all the status information of an individual node.

Replication

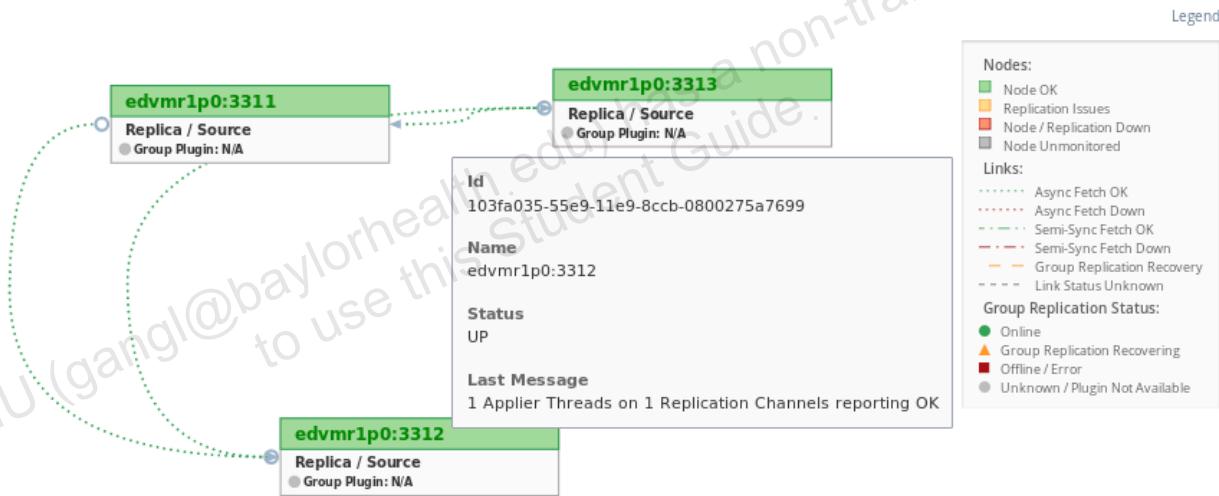
Max Replication Delay: 00:00:00  

Status		Statistics		Error History									
Show All entries		Search: <input type="text"/>				Show / hide columns		First	Previous	1	Next	Last	
Instance	Fetch State	Apply State	Time Behind	Read Only	GTID Enabled	Binary Log Format	Node Type	Channels	Version				
 edvmr1p0:3313			00:00:00	OFF	ON		Replica / Source	1	8.0.16-commercial				
 edvmr1p0:3312			00:00:00	OFF	ON		Replica / Source	1	8.0.16-commercial				
 edvmr1p0:3311			00:00:00	OFF	ON		Replica / Source	1	8.0.16-commercial				

Showing 1 to 3 of 3 entries First Previous 1 Next Last

8. View the replication topology in a graphical form. In the left navigation menu, click "Topology." What type of topology is this replication?

The screen shows the replication topology for the replication. This is a **circular** replication where edvmr1p0:3311 → edvmr1p0:3312 → edvmr1p0:3313 → edvmr1p0:3311.



9. Keep the three mysql client connections and browser open for the next practice.

Practice 13-2: Troubleshooting Replication Errors

Overview

In this practice, you troubleshoot replication errors. You run a provided script to break the replication, perform troubleshooting to identify the issues and fix them.

Duration

This practice should take you approximately ten minutes to complete.

Assumptions

A circular replication using GTID on `server1`, `server2`, and `server3` is running.

You have configured MySQL Enterprise Monitor to monitor the replication in the first practice of this lesson.

Tasks

1. Open a new terminal and run the `/labs/replerror.sh` script to break the replication.
2. Open the Firefox browser from the previous practice. Use the Replication Overview, Replication Dashboard, and Topology pages to identify the issue. What is the issue?
3. Fix the issue by starting the replication on `server2`. Does this fix the issue?
4. Restart the replication providing the correct user account and password. Does this fix the issue?
5. Keep the three `mysql` client connections and browser open for the next practice.

Solution 13-2: Troubleshooting Replication Errors

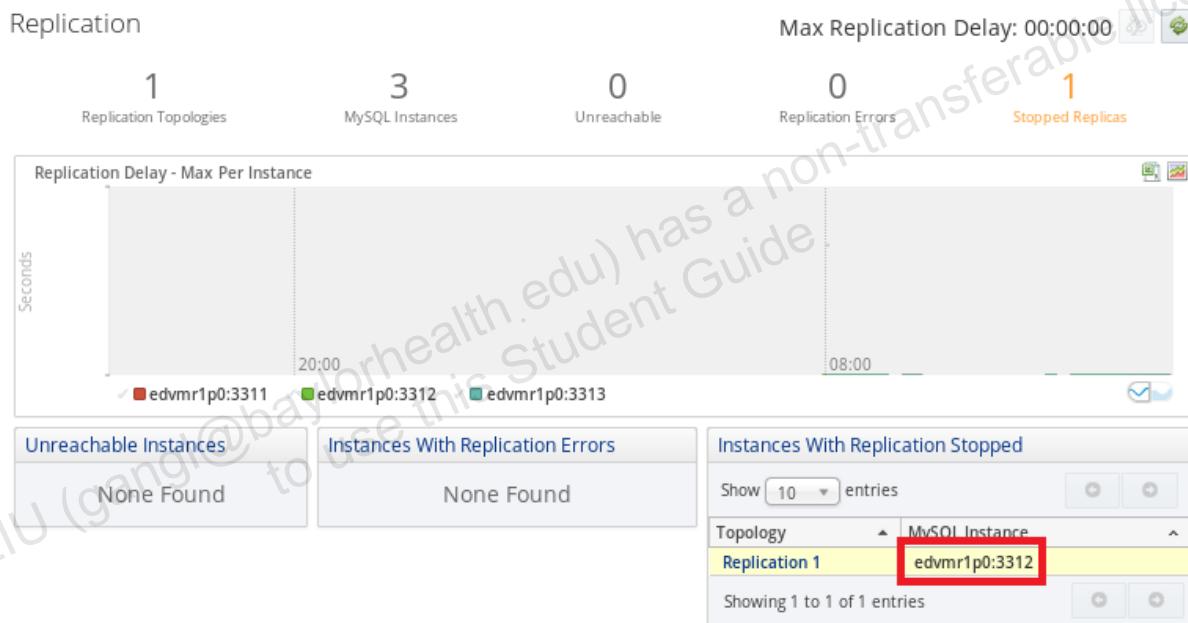
Solution Steps

1. Open a new terminal and run the `/labs/replerror.sh` script to break the replication. Enter the following command at a Linux terminal prompt and receive the results shown:

```
# sh /labs/replerror.sh  
#
```

2. Open the Firefox browser from the previous practice. Use the Replication Overview, Replication Dashboard, and Topology pages to identify the issue. What is the issue?

The Replication Overview page shows one instance has stopped replicating and the instance is `edvmr1p0:3312`.



The Replication Dashboard shows replication is off without error for instance `edvmr1p0:3312`.

Replication

Status Statistics Error History

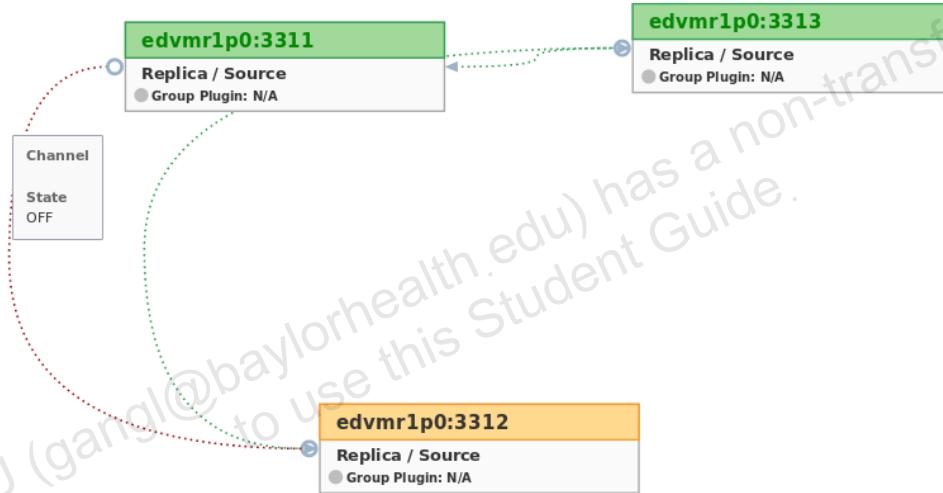
Show All entries

Instance	Fetch State	Apply State	Time Behind	Read Only	GTID Enabled	Binary Log F
edvmr1p0:3313	Up	Up	00:00:00	OFF	ON	ROW
edvmr1p0:3311	Up	Up	00:00:00	OFF	ON	ROW
edvmr1p0:3312	Up	Up	Unknown	OFF	ON	ROW

Showing 1 to 3 of 3 entries

Replication is OFF, but no errors are reported

The Topology page shows the channel between edvmr1p0:3311 and edvmr1p0:3312 is off.



The issue is that MySQL instance edvmr1p0:3312 has stopped replicating.

You can also use `SHOW SLAVE STATUS` for troubleshooting. The result shows `Slave_IO_Running` and `Slave_SQL_Running` are No.

```

2> SHOW SLAVE STATUS\G
***** 1. row *****
Slave_IO_State:
      Master_Host: 127.0.0.1
      Master_User: test
      Master_Port: 3311
      Connect_Retry: 60
      Master_Log_File: server1-bin.000004
      Read_Master_Log_Pos: 275
      Relay_Log_File: server2-relay-bin.000011
      Relay_Log_Pos: 453
      Relay_Master_Log_File: server1-bin.000004
  
```

```
Slave_IO_Running: No  
Slave_SQL_Running: No
```

```
...
```

- Fix the issue by starting the replication on server2. Does this fix the issue?

At the server2 mysql prompt, enter the following statement and receive the results shown:

```
2> START SLAVE;  
Query OK, 0 rows affected (#.## sec)  
  
2> SHOW SLAVE STATUS\G  
*****  
      Slave_IO_State: Connecting to master  
      Master_Host: 127.0.0.1  
      Master_User: test  
      Master_Port: 3311  
      Connect_Retry: 60  
      Master_Log_File: server1-bin.000004  
      Read_Master_Log_Pos: 275  
      Relay_Log_File: server2-relay-bin.000011  
      Relay_Log_Pos: 453  
      Relay_Master_Log_File: server1-bin.000004  
      Slave_IO_Running: Connecting  
      Slave_SQL_Running: Yes  
  
...  
      Last_IO_Errno: 1045  
      Last_IO_Error: error connecting to master  
'test@127.0.0.1:3311' - retry-time: 60 retries: 3  
...
```

- Slave_SQL_Running shows Yes.
- Slave_IO_Running remains Connecting after a few retries. The IO thread cannot connect to the master server.
- This does not fix the issue. The hint of the issue is the Master_User: test line, the username is incorrect.
- Note:** You may check the related pages in MySQL Enterprise Monitoring to check whether the replication is working or not.

4. Restart the replication providing the correct user account and password. Does this fix the issue?

At the server2 mysql prompt, enter the following statement and receive the results shown:

```
2> STOP SLAVE;
Query OK, 0 rows affected (#.## sec)

2> START SLAVE USER='rep1' PASSWORD='oracle';
Query OK, 0 rows affected (#.## sec)

2> SHOW SLAVE STATUS\G
***** 1. row *****
Slave_IO_State: Waiting for master to send event
Master_Host: 127.0.0.1
Master_User: rep1
Master_Port: 3311
Connect_Retry: 60
Master_Log_File: server1-bin.000004
Read_Master_Log_Pos: 275
Relay_Log_File: server2-relay-bin.000012
Relay_Log_Pos: 453
Relay_Master_Log_File: server1-bin.000004
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
...
```

- The issue is fixed with both the `Slave_IO_Running` and `Slave_SQL_Running` show Yes.

Note: You may check the related pages in MySQL Enterprise Monitoring to verify that the replication is working.

5. Keep the three mysql client connections and browser open for the next practice.

Practice 13-3: Performing a Replication Failover

Overview

In this practice, you perform a replication failover because one server has failed and unable to restart.

Duration

This practice should take you approximately 15 minutes to complete.

Assumptions

A circular replication using GTID on `server1`, `server2`, and `server3` is running.

You have configured MySQL Enterprise Monitor to monitor the replication in the first practice of this lesson.

Tasks

1. Open a new terminal and run the `/labs/replfailover.sh` script to break the replication.
2. Open the Firefox browser from the previous practice. Use the Replication Overview, Replication Dashboard, and Topology pages to identify the issue. What is the issue?
3. Fix the issue by starting the `server2` instance. Does this fix the issue?
4. When an instance in a circular replication stops working, the circle is broken. To restore the circular topology, remove the stopped instance and reconfigure the downstream slave instance to connect to the master of the stopped instance. Reconfigure the replication to restore the circular topology.
 - a. Which instance has stopped?
 - b. Which is the downstream slave instance?
 - c. Which is the master of the stopped instance?
5. Exit all `mysql` client sessions and close the Firefox browser window.
6. Stop the MySQL Enterprise Monitor service by entering the following command at a Linux terminal prompt:

```
# sh /opt/mysql/enterprise/monitor/mysqlmonitorctl.sh stop
```

Note: It is important to stop the MySQL Enterprise Monitor service. Not doing so will affect the outcome of later practices in the course.

7. Close all open Linux terminal windows.

Solution 13-3: Performing a Replication Failover

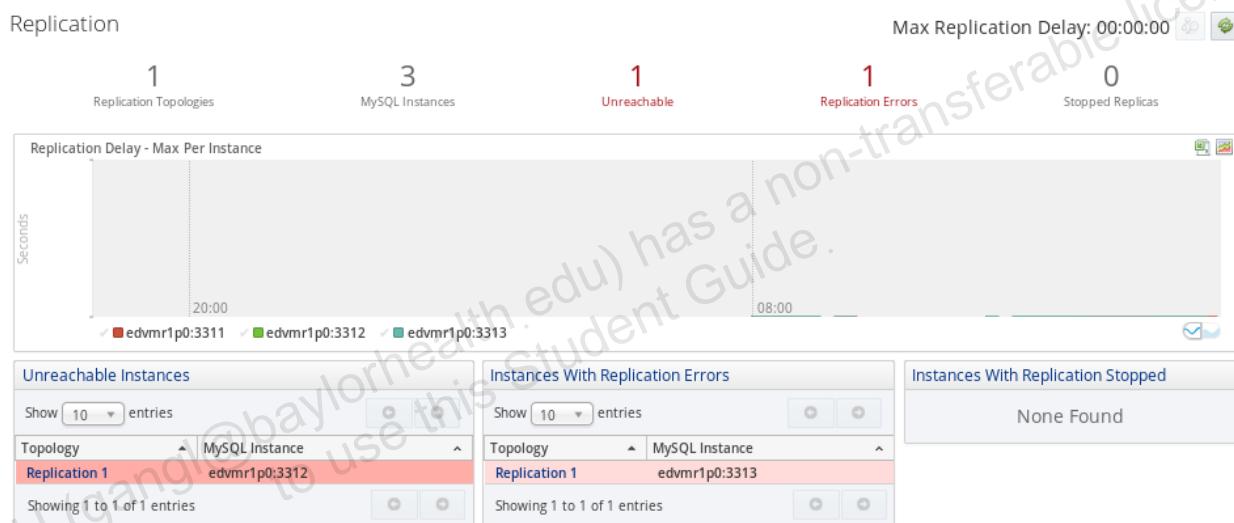
Solution Steps

1. Open a new terminal and run the `/labs/replfailover.sh` script to break the replication.

Enter the following command at a Linux terminal prompt and receive the results shown:

```
# sh /labs/replfailover.sh
#
```

2. Open the Firefox browser from the previous practice. Use the Replication Overview, Replication Dashboard, and Topology pages to identify the issue. What is the issue? The Replication Overview page shows instance `edvmr1p0:3312` is unreachable and instance `edvmr1p0:3313` has a replication error.



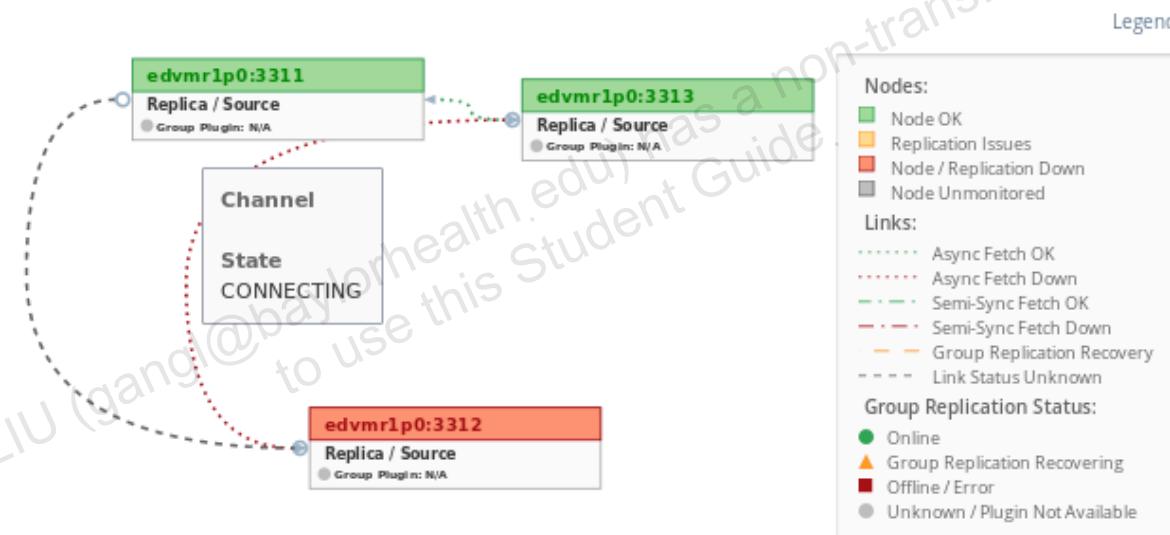
The Replication Dashboard shows instance `edvmr1p0:3312` is down and instance `edvmr1p0:3313` has fetch state (IO thread) error.

Replication

Replication Status							
Status	Statistics	Error History					
Show All entries		Search:			Actions		
Instance	Fetch State	Apply State	Time Behind	Read Only	GTID Enabled	Binary Log Format	
edvmr1p0:3313	Unknown	OFF	ON	ROW			
edvmr1p0:3311	00:00:00	OFF	ON	ROW			
edvmr1p0:3312	MySQL Instance is down!						

Showing 1 to 3 of 3 entries

The Topology page shows instance edvmr1p0:3313 is trying to connect to instance edvmr1p0:3312 which is down.



- The issue is that MySQL instance edvmr1p0:3312 has stopped.

You can verify whether the instance edvmr1p0:3312 is still running or not. Enter the following command at the Linux terminal prompt and receive the results shown:

```
# systemctl status mysqld@server2
● mysqld@server2.service - MySQL Multi Server for instance server2
   Loaded: loaded (/usr/lib/systemd/system/mysqld@.service; disabled; vendor preset: disabled)
   Active: inactive (dead)
     Docs: man:mysqld(8)
           http://dev.mysql.com/doc/refman/en/using-systemd.html

date-and-time edvmr1p0 systemd[1]: Starting MySQL Multi Server for instan.....
```

```
date-and-time edvmr1p0 systemd[1]: Started MySQL Multi Server
for instance...2.
```

Hint: Some lines were ellipsized, use -l to show in full.

- The instance `edvmr1p0:3312` is not running. So, the issue is `edvmr1p0:3312` instance is down or crashed.

3. Fix the issue by starting the `server2` instance. Does this fix the issue?

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# systemctl start mysqld@server2
Job for mysqld@server2.service failed because the control
process exited with error code. See "systemctl status
mysqld@server2.service" and "journalctl -xe" for details.

# systemctl status mysqld@server2 -l
● mysqld@server2.service - MySQL Multi Server for instance
server2
   Loaded: loaded (/usr/lib/systemd/system/mysqld@.service;
disabled; vendor preset: disabled)
     Active: failed (Result: exit-code) since ...
       Docs: man:mysqld(8)
              http://dev.mysql.com/doc/refman/en/using-systemd.html
      Process: 28727 ExecStart=/usr/local/mysql/bin/mysqld --
default-file=/labs/repl.cnf --defaults-group-suffix=@%I
$MYSQLD_OPTS (code=exited, status=1/FAILURE)
     Main PID: 28727 (code=exited, status=1/FAILURE)
        Status: "SERVER_BOOTING"
       Error: 2 (No such file or directory)

...
date-and-time edvmr1p0 mysqld[28727]: date-and-time [ERROR] [MY-
013276] [Server] Failed to set datadir to '/mysql/data2/' (OS
errno: 2 - No such file or directory)
date-and-time edvmr1p0 mysqld[28727]: date-and-time [ERROR] [MY-
010119] [Server] Aborting
date-and-time edvmr1p0 mysqld[28727]: date-and-time [System]
[MY-010910] [Server] /usr/local/mysql/bin/mysqld: Shutdown
complete (mysqld 8.0.15-commercial) MySQL Enterprise Server -
Commercial.
date-and-time edvmr1p0 systemd[1]: mysqld@server2.service: main
process exited, code=exited, status=1/FAILURE
date-and-time edvmr1p0 systemd[1]: Failed to start MySQL Multi
Server for instance server2.
date-and-time edvmr1p0 systemd[1]: Unit mysqld@server2.service
entered failed state.
date-and-time edvmr1p0 systemd[1]: mysqld@server2.service
failed.
```

- The issue is not fixed because the `server2` instance fails to start. The error message indicates the data directory `/mysql/data2` is missing. It may take a long time to restore the data directory from a backup or from another server. A quicker solution is to exclude the failed server from the replication and add it back after the data has been recovered.
4. When an instance in a circular replication stops working, the circle is broken. To restore the circular topology, remove the stopped instance and reconfigure the downstream slave instance to connect to the master of the stopped instance. Reconfigure the replication to restore the circular topology.
- a. Which instance has stopped? `edvmr1p0:3312 or server2`
 - b. Which is the downstream slave instance? `edvmr1p0:3313 or server3`
 - c. Which is the master of the stopped instance? `edvmr1p0:3311 or server1`

At the `server3 mysql` prompt, enter the following statement and receive the results shown:

```
3> SHOW SLAVE STATUS\G
***** 1. row *****
Slave_IO_State: Reconnecting after a failed
master event read
      Master_Host: 127.0.0.1
      Master_User: repl
      Master_Port: 3312
      Connect_Retry: 60
      Master_Log_File: server2-bin.000005
      Read_Master_Log_Pos: 275
      Relay_Log_File: server3-relay-bin.000009
      Relay_Log_Pos: 453
      Relay_Master_Log_File: server2-bin.000005
      Slave_IO_Running: Connecting
      Slave_SQL_Running: Yes
      ...
      Last_IO_Errno: 2003
      Last_IO_Error: error reconnecting to master
'repl@127.0.0.1:3312' - retry-time: 60 retries: 5
      ...

3> STOP SLAVE;
Query OK, 0 rows affected (#.## sec)

3> CHANGE MASTER TO MASTER_HOST='127.0.0.1', MASTER_PORT=3311;
Query OK, 0 rows affected (#.## sec)
```

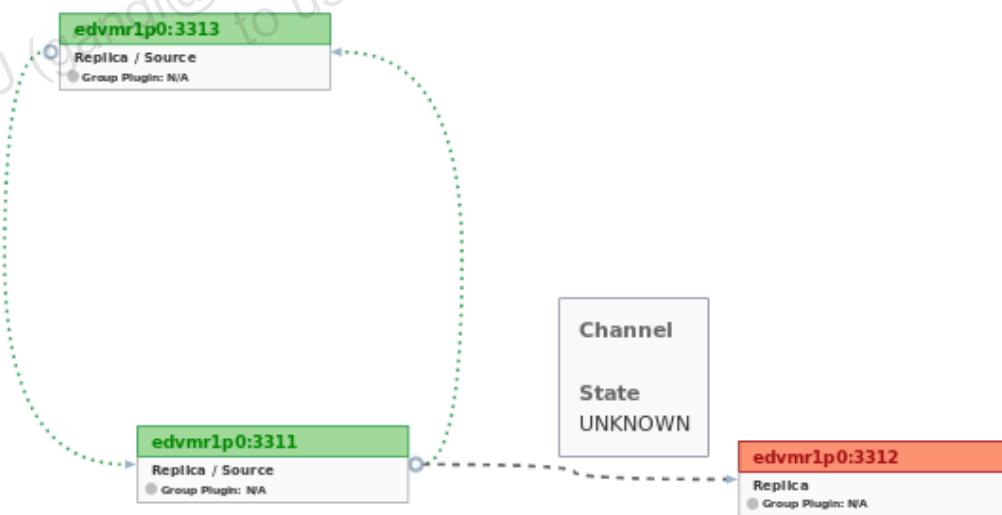
```

3> START SLAVE USER='rep1' PASSWORD='oracle';
Query OK, 0 rows affected (#.## sec)

3> SHOW SLAVE STATUS\G
***** 1. row *****
Slave_IO_State: Waiting for master to send event
Master_Host: 127.0.0.1
Master_User: rep1
Master_Port: 3311
Connect_Retry: 60
Master_Log_File: server1-bin.000004
Read_Master_Log_Pos: 275
Relay_Log_File: server3-relay-bin.000002
Relay_Log_Pos: 373
Relay_Master_Log_File: server1-bin.000004
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
...

```

The following shows the new topology diagram from MySQL Enterprise Monitor. edvmr1p0:3313 replicates to edvmr1p0:3313 and edvmr1p0:3313 replicates to edvmr1p0:3311. The stopped instance edvmr1p0:3312 is excluded from the replication.



5. Exit all mysql client sessions and close the Firefox browser window.
6. Stop the MySQL Enterprise Monitor service by entering the following command at a Linux terminal prompt:

```
# sh /opt/mysql/enterprise/monitor/mysqlmonitorctl.sh stop
```

Note: It is important to stop the MySQL Enterprise Monitor service. Not doing so will affect the outcome of later practices in the course.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# sh /opt/mysql/enterprise/monitor/mysqlmonitorctl.sh stop  
Stopping tomcat service . [ OK ]  
Stopping mysql service . [ OK ]
```

7. Close all open Linux terminal windows.

Practice Scripts for Lesson 13

Script: /labs/replerror.sh

```
#!/bin/bash
mysql -uroot --protocol=tcp -poracle -P3312 -e"stop slave" 2>/dev/null
```

Script: /labs/replfailover.sh

```
#!/bin/bash
mysqladmin -uroot --protocol=tcp -poracle -P3312 shutdown 2>/dev/null
mv /mysql/data2 /mysql/data22
```

GANG LIU (gangl@baylorhealth.edu) has a non-transferable license
to use this Student Guide.

Practices for Lesson 14: Achieving High Availability with MySQL InnoDB Cluster

Practices for Lesson 14: Overview

Overview

In these practices, you will install the tools necessary to work with MySQL InnoDB Cluster and use them to configure a production deployment. You will then test that the cluster ensures high availability by simulating and then recovering from a server outage.

Assumptions

- The `repl.cnf` file is in the `/labs` directory.
- The `idc-setup.sh` file is in the `/labs` directory.
- MySQL Shell has been installed in Practice 2-4.

Practice 14-1: Creating Member Instances

Overview

In this practice, you create three MySQL Server instances with separate static IP addresses.

Duration

This practice should take you approximately ten minutes to complete.

Tasks

1. At a Linux terminal prompt, change the current working directory to /etc/sysconfig/network-scripts and list its contents.
2. Create three new virtual network interfaces based on ifcfg-eth0 by copying the ifcfg-eth0 file to ifcfg-eth0:1, ifcfg-eth0:2, and ifcfg-eth0:3.
3. Edit the ifcfg-eth0:* files as follows, leaving all other configuration settings intact:
 - a. The ifcfg-eth0:1 file must contain the following entries:

```
DEVICE=eth0:1
BOOTPROTO=static
ONBOOT=yes
IPADDR=10.1.1.1
NETMASK=255.255.255.0
```

- b. The ifcfg-eth0:2 file must contain the following entries:

```
DEVICE=eth0:2
BOOTPROTO=static
ONBOOT=yes
IPADDR=10.1.1.2
NETMASK=255.255.255.0
```

- c. The ifcfg-eth0:3 file must contain the following entries:

```
DEVICE=eth0:3
BOOTPROTO=static
ONBOOT=yes
IPADDR=10.1.1.3
NETMASK=255.255.255.0
```

4. Enable the new network interfaces by using the ifup interface command.
5. Edit the /etc/hosts file to provide the following name/IP address mappings:
 - 10.1.1.1 maps to “server1”
 - 10.1.1.2 maps to “server2”
 - 10.1.1.3 maps to “server3”
6. Ensure that you can ping server1, server2, and server3 before continuing.

7. Edit the `/labs/repl.cnf` file to add the following entries:
 - For server1: `report-host=server1, report-port=3311`
 - For server2: `report-host=server2, report-port=3312`
 - For server3: `report-host=server3, report-port=3313`

Remove the section for server4; this practice does not use it.
8. Examine the contents of the `/labs/idc-setup.sh` file.
9. Execute the `/labs/idc-setup.sh` file to create the MySQL Server instances you will use in your cluster.
10. Leave the Linux terminal prompt open for the next practice.

Solution 14-1: Creating Member Instances

Solution Steps

- At a Linux terminal prompt, change the current working directory to /etc/sysconfig/network-scripts and list its contents.

Enter the following commands at the Linux terminal prompt and receive the results shown:

```
# cd /etc/sysconfig/network-scripts
# ls -l
total 244
-rw-r--r--. 1 root root    369 Aug 22 10:00 ifcfg-eth0
-rw-r--r--. 1 root root    126 Apr 14  2016 ifcfg-eth0-1
-rw-r--r--. 1 root root    106 Aug 22 10:00 ifcfg-eth1
-rw-r--r--. 1 root root    273 Apr  4  2016 ifcfg-eth2
-rw-r--r--. 1 root root    254 Sep 12  2016 ifcfg-lo
...
```

- This directory contains `ifcfg-eth*` files that configure the network interfaces on your system.

- Create three new virtual network interfaces based on `ifcfg-eth0` by copying the `ifcfg-eth0` file to `ifcfg-eth0:1`, `ifcfg-eth0:2`, and `ifcfg-eth0:3`.

Enter the following commands at the Linux terminal prompt and receive the results shown:

```
# cp ifcfg-eth0 ifcfg-eth0:1
# cp ifcfg-eth0 ifcfg-eth0:2
# cp ifcfg-eth0 ifcfg-eth0:3
```

- Edit the `ifcfg-eth0:*` files as follows, leaving all other configuration settings intact:

- The `ifcfg-eth0:1` file must contain the following entries:

```
DEVICE=eth0:1
BOOTPROTO=static
ONBOOT=yes
IPADDR=10.1.1.1
NETMASK=255.255.255.0
```

- The `ifcfg-eth0:2` file must contain the following entries:

```
DEVICE=eth0:2
BOOTPROTO=static
ONBOOT=yes
IPADDR=10.1.1.2
NETMASK=255.255.255.0
```

- c. The `ifcfg-eth0:3` file must contain the following entries:

```
DEVICE=eth0:3
BOOTPROTO=static
ONBOOT=yes
IPADDR=10.1.1.3
NETMASK=255.255.255.0
```

4. Enable the new network interfaces by using the `ifup interface` command.

Enter the following commands at the Linux terminal prompt:

```
# ifup eth0:1
# ifup eth0:2
# ifup eth0:3
```

5. Edit the `/etc/hosts` file to provide the following name/IP address mappings:

- 10.1.1.1 maps to “server1”
- 10.1.1.2 maps to “server2”
- 10.1.1.3 maps to “server3”

Edit the `/etc/hosts` file as follows:

```
127.0.0.1 localhost localhost.localdomain localhost4
localhost4.localdomain4
::1 localhost localhost.localdomain localhost6
localhost6.localdomain6
10.x.y.z hostname.us.oracle.com hostname

10.1.1.1 server1
10.1.1.2 server2
10.1.1.3 server3
```

6. Ensure that you can ping server1, server2, and server3 before continuing.

Enter the following commands at the Linux terminal prompt and receive the results shown:

```
# ping server1
PING server1 (10.1.1.1) 56(84) bytes of data.
64 bytes from server1 (10.1.1.1): icmp_seq=1 ttl=64 time=0.016 ms
64 bytes from server1 (10.1.1.1): icmp_seq=2 ttl=64 time=0.031 ms
64 bytes from server1 (10.1.1.1): icmp_seq=3 ttl=64 time=0.033 ms
^C
--- server1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.016/0.026/0.033/0.009 ms

# ping server2
PING server2 (10.1.1.2) 56(84) bytes of data.
64 bytes from server2 (10.1.1.2): icmp_seq=1 ttl=64 time=0.013 ms
64 bytes from server2 (10.1.1.2): icmp_seq=2 ttl=64 time=0.016 ms
64 bytes from server2 (10.1.1.2): icmp_seq=3 ttl=64 time=0.016 ms
^C
--- server2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.013/0.015/0.016/0.001 ms

# ping server3
PING server3 (10.1.1.3) 56(84) bytes of data.
64 bytes from server3 (10.1.1.3): icmp_seq=1 ttl=64 time=0.020 ms
64 bytes from server3 (10.1.1.3): icmp_seq=2 ttl=64 time=0.033 ms
64 bytes from server3 (10.1.1.3): icmp_seq=3 ttl=64 time=0.030 ms
^C
--- server3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.020/0.027/0.033/0.008 ms
```

- Press Ctrl + C to exit the ping command after you start to receive packets.
- If you do not receive packets, check the configuration of your virtual network interfaces and the /etc/hosts file before proceeding with the remaining practices.

7. Edit the /labs/repl.cnf file to add the following entries:

- For server1: report-host=server1, report-port=3311
- For server2: report-host=server2, report-port=3312
- For server3: report-host=server3, report-port=3313

Remove the section for server4; this practice does not use it.

The contents of the /labs/repl.cnf file should be as shown below:

```
[mysql@server1]
server-id=11
user=mysql
```

```
socket=/mysql/socket1
port=3311
datadir=/mysql/data1
log-error=/mysql/err1
log-bin=server1-bin
relay-log=server1-relay-bin
report-host=server1
report-port=3311
gtid-mode=ON
enforce-gtid-consistency

[mysqld@server2]
server-id=12
user=mysql
socket=/mysql/socket2
port=3312
datadir=/mysql/data2
log-error=/mysql/err2
log-bin=server2-bin
relay-log=server2-relay-bin
report-host=server2
report-port=3312
gtid-mode=ON
enforce-gtid-consistency

[mysqld@server3]
server-id=13
user=mysql
socket=/mysql/socket3
port=3313
datadir=/mysql/data3
log-error=/mysql/err3
log-bin=server3-bin
relay-log=server3-relay-bin
report-host=server3
report-port=3313
gtid-mode=ON
enforce-gtid-consistency
```

8. Examine the contents of the /labs/idc-setup.sh file.

The /labs/idc-setup.sh file's contents are as follows:

```
#!/bin/bash

echo "*** Stopping existing servers ..."
systemctl stop mysqld@server1
systemctl stop mysqld@server2
systemctl stop mysqld@server3
systemctl stop mysqld@server4

echo "*** Removing existing servers ..."
rm -rf /mysql/*
echo
echo "== Done"
echo

echo "*** Initializing servers 1-3..."
echo "**** SERVER 1"
mysqld --no-defaults --initialize-insecure --user=mysql --
datadir=/mysql/data1
echo
echo "**** SERVER 2"
mysqld --no-defaults --initialize-insecure --user=mysql --
datadir=/mysql/data2
echo
echo "**** SERVER 3"
mysqld --no-defaults --initialize-insecure --user=mysql --
datadir=/mysql/data3
echo

echo "**** Starting servers 1-4..."
systemctl start mysqld@server1
systemctl start mysqld@server2
systemctl start mysqld@server3

echo
sleep 10s

mysql -uroot -h127.0.0.1 -P3311 -e"SET sql_log_bin=OFF;UPDATE
mysql.user SET Host='%', plugin='mysql_native_password' WHERE
User='root';FLUSH PRIVILEGES;SET sql_log_bin=ON"
mysql -uroot -h127.0.0.1 -P3312 -e"SET sql_log_bin=OFF;UPDATE
mysql.user SET Host='%', plugin='mysql_native_password' WHERE
User='root';FLUSH PRIVILEGES;SET sql_log_bin=ON"
mysql -uroot -h127.0.0.1 -P3313 -e"SET sql_log_bin=OFF;UPDATE
mysql.user SET Host='%', plugin='mysql_native_password' WHERE
User='root';FLUSH PRIVILEGES;SET sql_log_bin=ON"
```

```
echo "== Done"
echo
```

- This script creates three new MySQL Server instances to replace the four that you used in the practices for the lesson titled “Administering a Replication Topology.” It creates a `root` user account on each that can log in from any host and disables binary logging while this operation takes place to avoid replication conflicts.
9. Execute the `/labs/idc-setup.sh` file to create the MySQL Server instances you will use in your cluster.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# sh /labs/idc-setup.sh
** Stopping existing servers ...
** Removing existing servers ...

== Done

** Initializing servers 1-3...
*** SERVER 1
date-and-time [System] [MY-013169] [Server] /opt/mysql-commercial-version-el7-x86_64/bin/mysqld (mysqld version-commercial) initializing of server in progress as process 6590
date-and-time [Warning] [MY-010453] [Server] root@localhost is created with an empty password ! Please consider switching off the --initialize-insecure option.
date-and-time [System] [MY-013170] [Server] /opt/mysql-commercial-version-el7-x86_64/bin/mysqld (mysqld version-commercial) initializing of server has completed

*** SERVER 2
date-and-time [System] [MY-013169] [Server] /opt/mysql-commercial-version-el7-x86_64/bin/mysqld (mysqld version-commercial) initializing of server in progress as process 6636
date-and-time [Warning] [MY-010453] [Server] root@localhost is created with an empty password ! Please consider switching off the --initialize-insecure option.
date-and-time [System] [MY-013170] [Server] /opt/mysql-commercial-version-el7-x86_64/bin/mysqld (mysqld version-commercial) initializing of server has completed

*** SERVER 3
date-and-time [System] [MY-013169] [Server] /opt/mysql-commercial-version-el7-x86_64/bin/mysqld (mysqld version-commercial) initializing of server in progress as process 6682
```

```
date-and-time [Warning] [MY-010453] [Server] root@localhost is  
created with an empty password ! Please consider switching off  
the --initialize-insecure option.  
date-and-time [System] [MY-013170] [Server] /opt/mysql-  
commercial-version-el7-x86_64/bin/mysqld (mysqld version-  
commercial) initializing of server has completed  
  
*** Starting servers 1-4...  
== Done
```

10. Leave the Linux terminal prompt open for the next practice.

Practice 14-2: Creating MySQL InnoDB Cluster

Overview

In this practice, you will install MySQL Shell and use it to configure a cluster with the MySQL Server instances you created in the preceding practice.

Duration

This practice should take you approximately 15 minutes to complete.

Tasks

1. At a Linux terminal prompt, switch to the `oracle` Linux user and perform the remaining steps as `oracle`.
Note: You will receive a warning if you attempt to configure MySQL InnoDB Cluster as the Linux `root` user.
2. Start MySQL Shell by executing `mysqlsh` at the Linux terminal prompt.
3. Use the `\c root@server1:3311` command to connect to `server1`.
Note: `\c` is a shorthand for `shell.connect()`. These practices use the shorthand notation for commands when they are available.
4. Enter `\?` at the MySQL Shell prompt to view the available commands and objects.
5. Enter `dba.help()` at the MySQL Shell prompt to view information about the methods available for working with MySQL InnoDB Cluster.
6. Attempt to create the cluster on `server1` by executing
`dba.createCluster("myCluster")`.
7. Configure the current instance (`server1`) by executing `dba.configureInstance()` with no parameters.
8. Check the configuration of `server2` by executing the
`dba.checkInstanceConfiguration` method, passing the connection details for `server2` as a string parameter.
9. Configure `server2` by executing `dba.configureInstance`, passing in the connection parameters for `server2`.
10. Configure `server3` by executing `dba.configureInstance`, passing in the connection parameters for `server3`.
11. Attempt to create the cluster again using the same command that you executed in step 7, but this time assign the result of the operation to a variable called `cluster`.
12. Add `server2` to the cluster by calling the `cluster.addInstance` method, passing in the connection details for `server2`.
13. Add `server3` to the cluster by calling the `cluster.addInstance` method, passing in the connection details for `server3`.

14. Execute `cluster.status()` to view the current status of the cluster. Which instance acts as the primary and accepts writes?
15. Execute `\quit` or `\q` at the MySQL Shell prompt to disconnect from the seed instance and terminate the session.
16. Enter `exit` at the Linux terminal prompt to become the Linux `root` user.
17. Leave the Linux terminal prompt open for the next practice.

Solution 14-2: Creating MySQL InnoDB Cluster

Solution Steps

- At a Linux terminal prompt, switch to the `oracle` Linux user and perform the remaining steps as `oracle`.

Note: You will receive a warning if you attempt to configure MySQL InnoDB Cluster as the Linux `root` user.

Enter the following commands at the Linux terminal prompt and receive the results shown:

```
# su oracle  
$ whoami  
oracle
```

- Start MySQL Shell by executing `mysqlsh` at the Linux terminal prompt.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
$ mysqlsh  
MySQL Shell version  
  
Copyright (c) 2016, 2018, Oracle and/or its affiliates. All rights reserved.  
  
Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.  
  
Type '\help' or '\?' for help; '\quit' to exit.  
  
MySQL-JS >
```

Note: The “JS” in the prompt tells you that MySQL Shell is in the default JavaScript scripting mode. You could enter `\py` or `\sql` to enter the Python or SQL mode, respectively.

- Use the `\c root@server1:3311` command to connect to `server1`.

Note: `\c` is a shorthand for `shell.connect()`. These practices use the shorthand notation for commands when they are available.

Enter the following command at the MySQL Shell prompt and receive the results shown:

```
MySQL-JS > \c root@server1:3311  
Creating a session to 'root@server1:3311'  
Enter password: [press Enter]  
Save password for 'root@server1:3311'? [Y]es/[N]o/Ne[v]er (default No): [press Enter]  
Fetching schema names for autocompletion... Press ^C to stop.  
Your MySQL connection id is 7303  
Server version: version-commercial MySQL Enterprise Server - Commercial
```

```
No default schema selected; type \use <schema> to set one.
```

MySQL-server1:3311-ssl-JS >

- The MySQL Shell prompt changes to show the server you are connected to and the type of connection (SSL).

4. Enter \? at the MySQL Shell prompt to view the available commands and objects.

Enter the following command at the MySQL Shell prompt and receive the results shown:

```
MySQL-server1:3311-ssl-JS > \?
===== Global Commands =====
\help      (\?,\h) Print this help.
\sql       Switch to SQL processing mode.
\js        Switch to JavaScript processing mode.
\py        Switch to Python processing mode.
\source    (\.) Execute a script file. Takes a file name as an argument.
\
Start multi-line input when in SQL mode.
\quit     (\q) Quit MySQL Shell.
\exit      Exit MySQL Shell. Same as \quit
\connect   (\c) Connect to a server.
\reconnect
Reconnect with a server.
\option
Manage MySQL Shell options.
\warnings  (\W) Show warnings after every statement.
\nowarnings (\w) Don't show warnings after every statement.
\status    (\s) Print information about the current global connection.
\use      (\u) Set the current schema for the active session.
\rehash
Update the auto-completion cache with database names.
\history
View and edit command line history.
```

For help on a specific command use the command as \? <command>

```
===== Global Objects =====
dba      Enables you to administer InnoDB clusters using the AdminAPI.
mysql    Used to work with classic MySQL sessions using SQL.
mysqlx   Used to work with X Protocol sessions using the MySQL X DevAPI.
session  Represents the currently open MySQL session.
shell    Gives access to general purpose functions and properties.
sys      Gives access to system specific parameters.
util     Global object that groups miscellaneous tools like upgrade checker.
```

For more help on a global variable use <var>.help(), e.g. dba.help()

- In this lesson's activity, you will use the `dba` object to configure and administer MySQL InnoDB Cluster.
5. Enter `dba.help()` at the MySQL Shell prompt to view information about the methods available for working with MySQL InnoDB Cluster.

Enter the following command at the MySQL Shell prompt and receive the results shown:

```
MySQL-server1:3311-ssl-JS > dba.help()

The global variable 'dba' is used to access the AdminAPI functionality and perform DBA operations. It is used for managing MySQL InnoDB clusters.

The following properties are currently supported.

- verbose Enables verbose mode on the Dba operations.

The following functions are currently supported.

- checkInstanceConfiguration      Validates an instance for MySQL InnoDB Cluster usage.
- configureInstance              Validates and configures an instance for MySQL InnoDB Cluster usage.
- configureLocalInstance         Validates and configures a local instance for MySQL InnoDB Cluster usage.
- createCluster                  Creates a MySQL InnoDB cluster.
- deleteSandboxInstance          Deletes an existing MySQL Server instance on localhost.
- deploySandboxInstance          Creates a new MySQL Server instance on localhost.
- dropMetadataSchema            Drops the Metadata Schema.
- getCluster                     Retrieves a cluster from the Metadata Store.
- help                          Provides help about this class and it's members.
- killSandboxInstance            Kills a running MySQL Server instance on localhost.
- rebootClusterFromCompleteOutage Brings a cluster back ONLINE when all members are OFFLINE.
- startSandboxInstance           Starts an existing MySQL Server instance on localhost.
- stopSandboxInstance            Stops a running MySQL Server instance on localhost.
```

```
For more help on a specific function use:
dba.help('<functionName>')

e.g. dba.help('deploySandboxInstance')
```

6. Attempt to create the cluster on server1 by executing
`dba.createCluster("myCluster")`.

Enter the following command at the MySQL Shell prompt and receive the results shown:

```
MySQL-server1:3311-ssl-JS > dba.createCluster("myCluster")
A new InnoDB cluster will be created on instance 'root@server1:3311'.

Validating instance at server1:3311...

This instance reports its own address as server1

Some configuration options need to be fixed:
+-----+-----+-----+
-----+
| Variable | Current Value | Required Value | Note
|
+-----+-----+-----+
-----+
| binlog_checksum | CRC32 | NONE | Update the server
variable |
+-----+-----+-----+
-----+
Please use the dba.configureInstance() command to repair these issues.

ERROR: Instance must be configured and validated with
dba.checkInstanceConfiguration() and dba.configureInstance() before it
can be used in an InnoDB cluster.

Dba.createCluster: Instance check failed (RuntimeError)
```

- The cluster cannot be created because the server1 instance is not configured correctly.
7. Configure the current instance (server1) by executing `dba.configureInstance()` with no parameters.

Enter the following command at the MySQL Shell prompt and receive the results shown:

```
MySQL-server1:3311-ssl-JS > dba.configureInstance()
Configuring local MySQL instance listening at port 3311 for use
in an InnoDB cluster...

This instance reports its own address as server1

Some configuration options need to be fixed:
```

```
+-----+-----+-----+
| Variable | Current Value | Required Value | Note
|
+-----+-----+-----+
| binlog_checksum | CRC32 | NONE | Update the
server variable |
+-----+-----+-----+
```

Do you want to perform the required configuration changes?
[y/n]: **y**
Configuring instance...
The instance 'server1:3311' was configured for use in an InnoDB cluster.

- Check the configuration of server2 by executing the dba.checkInstanceConfiguration method, passing the connection details for server2 as a string parameter.

Enter the following command at the MySQL Shell prompt and receive the results shown:

```
MySQL-server1:3311-ssl-JS >
dba.checkInstanceConfiguration("root@server2:3312")
Please provide the password for 'root@server2:3312': [press Enter]
Save password for 'root@server1:3311'? [Y]es/[N]o/[E]ver (default No): [press Enter]
Validating local MySQL instance listening at port 3312 for use
in an InnoDB cluster...

This instance reports its own address as server2

Checking whether existing tables comply with Group Replication
requirements...
No incompatible tables detected

Checking instance configuration...

Some configuration options need to be fixed:
+-----+-----+-----+
| Variable | Current Value | Required Value | Note
|
+-----+-----+-----+
```

- server2 also requires configuration changes. server3 is identical to server2 and will, therefore, also require these changes.

9. Configure server2 by executing `dba.configureInstance`, passing in the connection parameters for server2.

Enter the following command at the MySQL Shell prompt and receive the results shown:

```
MySQL-server1:3311-ssl-JS >
dba.configureInstance("root@server2:3312")
Please provide the password for 'root@server2:3312': [press Enter]
Save password for 'root@server2:3312'? [Y]es/[N]o/Ne[v]er (default No): [press Enter]
Configuring local MySQL instance listening at port 3312 for use in an InnoDB cluster...
```

This instance reports its own address as server2

Some configuration options need to be fixed:

Variable	Current Value	Required Value	Note

```
| binlog_checksum | CRC32           | NONE          | Update the
server variable |
+-----+-----+-----+
-----+
Do you want to perform the required configuration changes?
[y/n]: y
Configuring instance...
The instance 'server2:3312' was configured for use in an InnoDB
cluster.
```

10. Configure server3 by executing dba.configureInstance, passing in the connection parameters for server3.

Enter the following command at the MySQL Shell prompt and receive the results shown:

```
MySQL> dba.configureInstance("root@server3:3313")
Please provide the password for 'root@server3:3313': [press Enter]
Save password for 'root@server3:3313'? [Y]es/[N]o/Ne[v]er (default No): [press Enter]
Configuring local MySQL instance listening at port 3313 for use in an InnoDB cluster...

This instance reports its own address as server3

Some configuration options need to be fixed:
+-----+-----+-----+
| Variable | Current Value | Required Value | Note
|          |
+-----+-----+-----+
| binlog_checksum | CRC32 | NONE | Update the
server variable |
+-----+-----+-----+
Do you want to perform the required configuration changes?
[y/n]: y
Configuring instance...
The instance 'server3:3313' was configured for use in an InnoDB cluster.
```

11. Attempt to create the cluster again using the same command that you executed in step 7, but this time assign the result of the operation to a variable called `cluster`.

Enter the following command at the MySQL Shell prompt and receive the results shown:

```
MySQL-server1:3311-ssl-JS > var cluster =
  dba.createCluster("myCluster")
A new InnoDB cluster will be created on instance
'root@server1:3311'.

Validating instance at server1:3311...

This instance reports its own address as server1

Instance configuration is suitable.
Creating InnoDB cluster 'myCluster' on 'root@server1:3311'...
Adding Seed Instance...

Cluster successfully created. Use Cluster.addInstance() to add
MySQL instances.

At least 3 instances are needed for the cluster to be able to
withstand up to one server failure.
```

- The cluster was created successfully and you must now add instances to it.

12. Add `server2` to the cluster by calling the `cluster.addInstance` method, passing in the connection details for `server2`.

Enter the following command at the MySQL Shell prompt and receive the results shown:

```
MySQL-server1:3311-ssl-JS >
cluster.addInstance("root@server2:3312")
A new instance will be added to the InnoDB cluster. Depending on
the amount of data on the cluster this might take from a few
seconds to several hours.

Please provide the password for 'root@server2:3312': [press
Enter]
Save password for 'root@server2:3312'? [Y]es/[N]o/[Ne]ver (default
No): [press Enter]
Adding instance to the cluster ...

Validating instance at server2:3312...

This instance reports its own address as server2

Instance configuration is suitable.
The instance 'root@server2:3312' was successfully added to the
cluster.
```

13. Add server3 to the cluster by calling the `cluster.addInstance` method, passing in the connection details for server3.

Enter the following command at the MySQL Shell prompt and receive the results shown:

```
MySQL-server1:3311-ssl-JS > cluster.addInstance("root@server3:3313")
A new instance will be added to the InnoDB cluster. Depending on
the amount of data on the cluster this might take from a few
seconds to several hours.

Please provide the password for 'root@server3:3313': [press Enter]
Save password for 'root@server3:3313'? [Y]es/[N]o/[E]ver (default
No): [press Enter]
Adding instance to the cluster ...

Validating instance at server3:3313...

This instance reports its own address as server3

Instance configuration is suitable.
The instance 'root@server3:3313' was successfully added to the
cluster.
```

14. Execute `cluster.status()` to view the current status of the cluster. Which instance acts as the primary and accepts writes?

Enter the following command at the MySQL Shell prompt and receive the results shown:

```
MySQL-server1:3311-ssl-JS > cluster.status()
{
  "clusterName": "myCluster",
  "defaultReplicaSet": {
    "name": "default",
    "primary": "server1:3311",
    "ssl": "REQUIRED",
    "status": "OK",
    "statusText": "Cluster is ONLINE and can tolerate up to
ONE failure.",
    "topology": {
      "server1:3311": {
        "address": "server1:3311",
        "mode": "R/W",
        "readReplicas": {},
        "role": "HA",
        "status": "ONLINE"
      },
    }
  }
},
```

```
"server2:3312": {
    "address": "server2:3312",
    "mode": "R/O",
    "readReplicas": {},
    "role": "HA",
    "status": "ONLINE"
},
"server3:3313": {
    "address": "server3:3313",
    "mode": "R/O",
    "readReplicas": {},
    "role": "HA",
    "status": "ONLINE"
}
},
"groupInformationSourceMember": "mysql://root@server1:3311"
}
```

- server1 acts as the primary and accepts both reads and writes. server2 and server3 are read-only secondaries.
15. Execute `\quit` or `\q` at the MySQL Shell prompt to disconnect from the seed instance and terminate the session.

Enter the following command at the MySQL Shell prompt and receive the results shown:

```
MySQL-server1:3311-ssl-JS > \quit
Bye!
$
```

16. Enter `exit` at the Linux terminal prompt to become the Linux `root` user.

Enter the following commands at the Linux terminal prompt and receive the results shown:

```
$ exit
exit
# whoami
root
```

17. Leave the Linux terminal prompt open for the next practice.

Practice 14-3: Deploying MySQL Router and Testing the Cluster

Overview

In this practice, you configure MySQL Router to enable clients to connect to the cluster without needing to know details of the individual instances within the cluster.

Duration

This practice should take you approximately 15 minutes to complete.

Tasks

1. At the Linux terminal prompt, `su` to the `oracle` user and change the current working directory to the home directory.
2. Bootstrap MySQL Router with the primary (metadata) instance as the `oracle` user by executing:

```
mysqlrouter --bootstrap instance:port --directory=mysqlrouter
```

You have now configured MySQL Router. Make a note of the connection details for connections displayed in the output for the read/write (primary) and read-only (secondary) instances.

3. Start `mysqlrouter` as a background process. Refer to the command found in the output of Step 2.
4. Verify that the router is configured correctly by using MySQL Shell to connect to the read/write port (primary instance) displayed in the output of step 2.
5. Change the MySQL Shell mode from the default JavaScript to SQL, by entering `\sql`. The MySQL Shell prompt changes to show that you are now in SQL mode.
6. Enter a `SELECT` statement to display the port number of the host that the MySQL Shell client is connected to. Verify that this is the primary (read/write) instance running on port 3311.
7. Create a database on the primary instance called `clustertest`, with a table called `tbl_cluster`. The `tbl_cluster` table should have an auto-incrementing primary key column called `ID`, and a `VARCHAR(255)` column called `SomeText`.
8. Insert a row into the `tbl_cluster` table with "MySQL InnoDB Cluster" as the value for the `SomeText` column.
9. Exit the MySQL Shell session.
10. Use MySQL Shell to connect to one of the read-only (secondary) instances using the port number you determined in step 2.
11. Switch to SQL mode and verify that you are connected to an instance on either port 3312 or 3313.
12. Verify that the `clustertest` database you created on the primary instance in step 8 has replicated to the secondary instance you are currently connected to.
13. Leave the MySQL Shell connection to the secondary instance open for the next practice.

Solution 14-3: Deploying MySQL Router and Testing the Cluster

Solution Steps

- At the Linux terminal prompt, `su` to the `oracle` user and change the current working directory to the home directory.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# su oracle
$ cd
$ pwd
/home/oracle
$
```

- Bootstrap MySQL Router with the primary (metadata) instance as the `oracle` user by executing:

```
mysqlrouter --bootstrap instance:port --directory=mysqlrouter
```

You have now configured MySQL Router. Make a note of the connection details for connections displayed in the output for the read/write (primary) and read-only (secondary) instances.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
$ mysqlrouter --bootstrap server1:3311 --directory=mysqlrouter
Please enter MySQL password for root: [press Enter]
# Bootstrapping MySQL Router instance at '/home/oracle/mysqlrouter'...
- Checking for old Router accounts
  - No prior Router accounts found
- Creating mysql account mysql_router1_per0ea9xb2i5@'%' for cluster
  management
- Storing account in keyring
- Adjusting permissions of generated files
- Creating configuration /home/oracle/mysqlrouter/mysqlrouter.conf

# MySQL Router configured for the InnoDB cluster 'myCluster'
After this MySQL Router has been started with the generated
configuration
$ mysqlrouter -c /home/oracle/mysqlrouter/mysqlrouter.conf
the cluster 'myCluster' can be reached by connecting to:
## MySQL Classic protocol
- Read/Write Connections: localhost:6446
- Read/Only Connections: localhost:6447
## MySQL X protocol
- Read/Write Connections: localhost:64460
- Read/Only Connections: localhost:64470
```

- Start `mysqlrouter` as a background process. Refer to the command found in the output of Step 2.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
$ mysqlrouter -c /home/oracle/mysqlrouter/mysqlrouter.conf &
[1] 10353
Loading all plugins.
  plugin 'logger:' loading
  plugin 'metadata_cache:myCluster' loading
  plugin 'routing:myCluster_default_ro' loading
  plugin 'routing:myCluster_default_rw' loading
  plugin 'routing:myCluster_default_x_ro' loading
  plugin 'routing:myCluster_default_x_rw' loading
Initializing all plugins.
  plugin 'logger' initializing
logging facility initialized, switching logging to loggers
specified in configuration
```

- Note:** The Process ID (10353) shown in the sample output will likely be different on your system.
- Verify that the router is configured correctly by using MySQL Shell to connect to the read/write port (primary instance) displayed in the output of step 2.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
$ mysqlsh --uri root@localhost:6446
Creating a session to 'root@localhost:6446'
Enter password: [press Enter]
Save password for 'root@localhost:6446'? [Y]es/[N]o/[Never]
(default No): [press Enter]
Fetching schema names for autocompletion... Press ^C to stop.
Your MySQL connection id is 4010
...
MySQL-localhost:6446-ssl-JS >
```

- Change the MySQL Shell mode from the default JavaScript to SQL, by entering `\sql`. The MySQL Shell prompt changes to show that you are now in SQL mode.

Enter the following command at the MySQL Shell prompt and receive the results shown:

```
MySQL-localhost:6446-ssl-JS > \sql
Switching to SQL mode... Commands end with ;

MySQL-localhost:6446-ssl-SQL >
```

6. Enter a `SELECT` statement to display the port number of the host that the MySQL Shell client is connected to. Verify that this is the primary (read/write) instance running on port 3311.

Enter the following statement at the MySQL Shell prompt and receive the results shown:

```
MySQL-localhost:6446-ssl-SQL > SELECT @@port;
+-----+
| @@port |
+-----+
| 3311 |
+-----+
1 row in set (#.#### sec)
```

7. Create a database on the primary instance called `clustertest`, with a table called `tbl_cluster`. The `tbl_cluster` table should have an auto-incrementing primary key column called `ID`, and a `VARCHAR(255)` column called `SomeText`.

Enter the following statements at the MySQL Shell prompt and receive the results shown:

```
MySQL-localhost:6446-ssl-SQL > CREATE DATABASE clustertest;
Query OK, 1 row affected (#.#### sec)

MySQL-localhost:6446-ssl-SQL > USE clustertest;
Query OK, 0 rows affected (#.#### sec)

MySQL-localhost:6446-ssl-SQL > CREATE TABLE tbl_cluster (
    -> ID INT NOT NULL AUTO_INCREMENT,
    -> SomeText VARCHAR(255),
    -> PRIMARY KEY(ID));
Query OK, 0 rows affected (#.#### sec)
```

8. Insert a row into the `tbl_cluster` table with "MySQL InnoDB Cluster" as the value for the `SomeText` column.

Enter the following statement at the MySQL Shell prompt and receive the results shown:

```
MySQL-localhost:6446-ssl-SQL > INSERT INTO tbl_cluster
    -> (SomeText)
    -> VALUES ('MySQL InnoDB Cluster');
Query OK, 1 row affected (#.#### sec)
```

9. Exit the MySQL Shell session.

Enter the following command at the MySQL Shell prompt and receive the results shown:

```
MySQL-localhost:6446-ssl-SQL > \q
Bye!
$
```

10. Use MySQL Shell to connect to one of the read-only (secondary) instances using the port number you determined in step 2.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
$ mysqlsh --uri root@localhost:6447
Creating a session to 'root@localhost:6447'
Enter password: [press Enter]
Save password for 'root@localhost:6447'? [Y]es/[N]o/[N]ever
(default No): [press Enter]
Fetching schema names for autocompletion... Press ^C to stop.
Your MySQL connection id is 28
...
MySQL-localhost:6447-ssl-JS >
```

11. Switch to SQL mode and verify that you are connected to an instance on either port 3312 or 3313.

Enter the following commands at the MySQL Shell prompt and receive the results shown:

```
MySQL-localhost:6447-ssl-JS > \sql
Switching to SQL mode... Commands end with ;

MySQL-localhost:6447-ssl-SQL > SELECT @@port;
+-----+
| @@port |
+-----+
| 3313 |
+-----+
1 row in set (#.#### sec)
```

- If the output of `SELECT @@port` displays either 3312 or 3313, you are connected to a read-only (secondary) instance.

12. Verify that the `clustertest` database you created on the primary instance in step 8 has replicated to the secondary instance you are currently connected to.

Enter the following commands at the MySQL Shell prompt and receive the results shown:

```
MySQL-localhost:6447-ssl-SQL > USE clustertest;
Query OK, 0 rows affected (#.#### sec)

MySQL-localhost:6447-ssl-SQL > SELECT * FROM tbl_cluster;
+----+-----+
| ID | SomeText           |
+----+-----+
| 7  | MySQL InnoDB Cluster |
+----+-----+
1 row in set (#.#### sec)
```

13. Leave the MySQL Shell connection to the secondary instance open for the next practice.

Practice 14-4: Testing High Availability

Overview

In this practice, you simulate failure of the primary instance in the cluster and observe how the cluster reconfigures itself to promote a secondary instance to become the new primary.

Duration

This practice should take you approximately ten minutes to complete.

Tasks

1. Open a new Linux terminal prompt and execute a `systemctl` command to shut down the primary instance, `server1`.
2. At the MySQL Shell command prompt, connected to a secondary instance from the previous practice, switch to JavaScript mode (`\js`).
3. Execute `dba.getCluster("myCluster")` to retrieve the object that references the cluster and store a reference to it in a variable named `cluster`.
4. Execute `cluster.status()` to display the current status of the cluster. What has happened?
5. At a Linux terminal prompt, start `server1`.
6. At the MySQL Shell prompt, execute `cluster.status()` to display the current status of the cluster again. What has happened?
7. At the MySQL Shell prompt, execute `cluster.dissolve({force:true})` to dissolve the cluster.
8. Exit MySQL Shell.
9. Enter `exit` at the Linux terminal prompt to become the root Linux user.
10. Stop `server1`, `server2`, and `server3`.
11. Start the main `mysqld` service.
12. Close any open terminal windows.

Solution 14-4: Testing High Availability

Solution Steps

1. Open a new Linux terminal prompt and execute a `systemctl` command to shut down the primary instance, `server1`.

Enter the following commands at the Linux terminal prompt and receive the results shown:

```
# systemctl stop mysqld@server1
# systemctl status mysqld@server1
● mysqld@server1.service - MySQL Multi Server for instance
server1
    Loaded: loaded (/usr/lib/systemd/system/mysqld@.service;
enabled; vendor preset: disabled)
    Active: inactive (dead)
      Docs: man:mysqld(8)
             http://dev.mysql.com/doc/refman/en/using-systemd.html
...
date-and-time host systemd[1]: Stopped MySQL Multi Server for
instance server1.
```

2. At the MySQL Shell command prompt, connected to a secondary instance from the previous practice, switch to JavaScript mode (`\js`).

Enter the following command at the MySQL Shell prompt and receive the results shown:

```
MySQL-localhost:6447-ssl-SQL > \js
Switching to JavaScript mode...
```

3. Execute `dba.getCluster("myCluster")` to retrieve the object that references the cluster and store a reference to it in a variable named `cluster`.

Enter the following command at the MySQL Shell prompt and receive the results shown:

```
MySQL-localhost:6447-ssl-clustertest-JS > var cluster =
dba.getCluster("myCluster")
MySQL-localhost:6447-ssl-clustertest-JS >
```

4. Execute `cluster.status()` to display the current status of the cluster. What has happened?

Enter the following command at the MySQL Shell prompt and receive the results shown:

```
MySQL-localhost:6447-ssl-clustertest-JS > cluster.status()
{
  "clusterName": "myCluster",
  "defaultReplicaSet": {
    "name": "default",
    "primarystatusstatusText1 member is not active",
    "topology": {
      "server1:3311": {
        "address": "server1:3311",
        "mode": "R/O",
        "readReplicas": {},
        "role": "HA",
        "statusmode
```

- The cluster has elected server2 as the new primary. server1 reports as MISSING and the cluster, having only two available member, cannot tolerate another instance failure.
5. At a Linux terminal prompt, start server1.

Enter the following commands at the Linux terminal prompt and receive the results shown:

```
# systemctl start mysqld@server1
# systemctl status mysqld@server1
● mysqld@server1.service - MySQL Multi Server for instance
server1
    Loaded: loaded (/usr/lib/systemd/system/mysqld@.service;
enabled; vendor preset: disabled)
    Active: active (running) since Thu 2018-08-30 13:37:32 UTC;
8s ago
      Docs: man:mysqld(8)
              http://dev.mysql.com/doc/refman/en/using-systemd.html
   Main PID: 8088 (mysqld)
     Status: "SERVER_OPERATING"
      CGroup: /system.slice/system-
mysqld.slice/mysqld@server1.service
           └─8088 /usr/local/mysql/bin/mysqld --defaults-
file=/labs/repl.cnf --defaults-group-suffix=@serve...
date-and-time host systemd[1]: Starting MySQL Multi Server for
instance server1...
date-and-time host systemd[1]: Started MySQL Multi Server for
instance server1.
```

6. At the MySQL Shell prompt, execute `cluster.status()` to display the current status of the cluster again. What has happened?

Enter the following command at the MySQL Shell prompt and receive the results shown:

```
MySQL-localhost:6447-ssl-clustertest-JS > cluster.status()
{
  "clusterName": "myCluster",
  "defaultReplicaSet": {
    "name": "default",
    "primary

```

```

        "readReplicas": {},
        "role": "HA",
        "status": "ONLINE"
    },
    "server2:3312": {
        "address": "server2:3312",
        "mode": "R/W",
        "readReplicas": {},
        "role": "HA",
        "status": "ONLINE"
    },
    "server3:3313": {
        "address": "server3:3313",
        "mode": "R/O",
        "readReplicas": {},
        "role": "HA",
        "status": "ONLINE"
    }
},
"topologyMode": "Single-Primary"
},
"groupInformationSourceMember": "server2:3312"
}

```

- server1 is back online and the cluster is tolerant of a single failure again.
server2 is still the primary (read-write) instance.

7. At the MySQL Shell prompt, execute `cluster.dissolve({force:true})` to dissolve the cluster.

Enter the following command at the MySQL Shell prompt and receive the results shown:

```

MySQL-localhost:6447-ssl-clustertest-JS >
cluster.dissolve({force:true})

WARNING: You are about to dissolve the whole cluster and lose
the high availability features provided by it. This operation
cannot be reverted. All members will be removed from their
ReplicaSet and replication will be stopped, internal recovery
user accounts and the cluster metadata will be dropped. User
data will be maintained intact in all instances.

Are you sure you want to dissolve the cluster? [y/N]: y

Instance 'server1:3311' is attempting to leave the cluster...
Instance 'server3:3313' is attempting to leave the cluster...
Instance 'server2:3312' is attempting to leave the cluster...
The cluster was successfully dissolved.

Replication was disabled but user data was left intact.

```

8. Exit MySQL Shell.

Enter the following command at the MySQL Shell prompt and receive the results shown:

```
MySQL-localhost:6447-ssl-clustertest-JS > \q  
Bye!  
$
```

9. Enter `exit` at the Linux terminal prompt to become the root Linux user.

Enter the following commands at the Linux terminal prompt and receive the results shown:

```
$ exit  
exit  
# whoami  
root
```

10. Stop `server1`, `server2`, and `server3`.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# systemctl stop mysqld@server*  
# systemctl status mysqld@server*  
#
```

11. Start the main `mysqld` service.

Enter the following command at the Linux terminal prompt and receive the results shown:

```
# systemctl start mysqld  
# systemctl status mysqld  
● mysqld.service - MySQL Server  
    Loaded: loaded (/usr/lib/systemd/system/mysqld.service;  
            enabled; vendor preset: disabled)  
    Active: active (running) since Thu 2018-08-30 13:48:33 UTC;  
          7s ago  
      Docs: man:mysqld(8)  
            http://dev.mysql.com/doc/refman/en/using-systemd.html  
    Main PID: 8206 (mysqld)  
       Status: "SERVER_OPERATING"  
      CGroup: /system.slice/mysqld.service  
              └─8206 /usr/local/mysql/bin/mysqld --defaults-  
                  file=/etc/my.cnf  
...  
date-and-time host systemd[1]: Started MySQL Server.  
Hint: Some lines were ellipsized, use -l to show in full.
```

12. Close any open terminal windows.

Practices for Lesson 15: Conclusion

Practices for Lesson 15

There are no practices for this lesson.