Hands-on Lab: Automating MySQL Database Tasks Using Shell Scripts

Estimated time needed: 30 minutes

In this lab, you will use the MySQL command line interface (CLI) to automatically back up the database and restore the database when required.

Software Used in This Lab

To complete this lab, you will use the MySQL relational database service available as part of the IBM Skills Network Labs (SN Labs). SN Labs is a virtual lab environment used in this course.

Database Used in This Lab

You will use a modified version of the Sakila database for the lab, which is an adapted version from the following source: https://dev.mysql.com/doc/sakila/en/ under New BSD license [Copyright 2021 - Oracle Corporation].

Objectives

After completing this lab, you will be able to use the MySQL command line to:

- Create the shell script to back up the database.
- Create a cron job to run the backup script thereby creating a backup file.
- Truncate the tables in the database.
- Restore the database using the backup file.

Exercise

In this exercise you will create a database, backup the database using an automated script, and finally truncate and restore it back.

Task A: Create a Database

1. Go to **Terminal > New Terminal** to open a terminal from the side-by-side launched Cloud IDE.

2. Copy the command below and run it on the terminal to fetch the sakila_mysql_dump.sql file to the Cloud IDE.

   ```
   wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0110EN-SkillsNetwork/datasets/sakila/sakila_mysql_dump.sql
   ```

3. Start the MySQL service session using the `Start MySQL in IDE button` directive.

Open MySQL Page in IDE

4. On the launching page, click on the **Create** button.

5. Open the `.my.cnf` as root user, with nano editor in terminal to configure your mysql password .

   ```
   sudo nano ~/.my.cnf
   ```

6. Add the password you noted in the previous step to the `.my.cnf` file. This aids in not entering the password over and over again and keeps the password hidden in the config file.

   Note: Once you open the ~/.my.cnf file enter the line `password = <Your MySQL Password>` and replace the password with your password noted before.

   ```
   [client]
   host = mysql
   port = 3306
   user = root
   password = <Your MySQL Password>
   ```

7. Press Ctrl+O, next followed by the Enter key to save the file.

8. Press Ctrl+X to quit the nano editor.

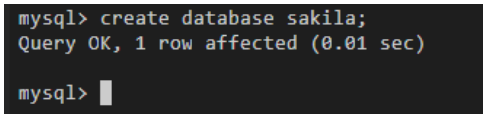
9. Initiate the mysql command prompt session within the MySQL service session using the command below in the terminal:

   ```
   mysql
   ```

Here you find that you are able to login to mysql without entering the password as it is already configured in the ~/.my.cnf file.

9. Create a new database **sakila** using the command below in the terminal and proceed to Task B:

```
create database sakila;
```



Task B: Restore the Structure and Data of a Table

1. To use the newly created empty sakila database, use the command below in the terminal:

```
use sakila;
```

2. Restore the sakila mysql dump file (containing the sakila database table definitions and data) to the newly created empty sakila database using the command below in the terminal:

```
source sakila_mysql_dump.sql;
```

3. To check, list all the table names from the sakila database using the command below in the terminal

```
SHOW FULL TABLES WHERE table_type = 'BASE TABLE';
```

Task C: Understanding the Process Involved in Creating MySQL Database Backups

You will create a shell script that does the following:

- Writes the database to an sqlfile created with a timestamp, using the `mysqldump` command
- Zips the sqlfile into a zip file using the `gzip` command
- Removes the sqlfile using rm command
- Deletes the backup after 30 days

Before you create the script, let's understand each of the command blocks you will be adding to the file.

- To start with, you have a database that you want to back up. You will store the name of the database in a variable.

```
DATABASE='sakila'
```

- It is always a good practice to print some logs, which can help in debugging.

```
echo "Pulling Database: This may take a few minutes"
```

- You will also set the backup folder where the SQL and zipped files will be stored.

```
backupfolder=/home/theia/backups
```

- You will decide and set the number of days the backup will need to be kept.

```
keep_day=30
```

- You will set the name of the SQL file where you will dump the database as "all-database-" suffixed with the current timestamp and .sql extension, and the zip file in which you will compress the SQL file as "all-database-" suffixed with the current timestamp and .gz extension.

```
sqlfile=$backupfolder/all-database-$(date +%d-%m-%Y_%H-%M-%S).sql
zipfile=$backupfolder/all-database-$(date +%d-%m-%Y_%H-%M-%S).gz
```

- Now that all the placeholders are created, you will create the SQL backup. In MySQL, it can be accomplished with the `mysqldump` command. Depending on whether the operation is successful or not, a log is printed. If the operation is successful, you will compress the .sql file into a .gz and delete the the .sql file.

```
if mysqldump  $DATABASE > $sqlfile ; then
   echo 'Sql dump created'
    # Compress backup
    if gzip -c $sqlfile > $zipfile; then
        echo 'The backup was successfully compressed'
    else
        echo 'Error compressing backupBackup was not created!'
        exit
    fi
    rm $sqlfile
else
   echo 'pg_dump return non-zero code No backup was created!'
   exit
```

```
fi
```

- Finally, you will remove any backups that are in the system for longer than the time you decided to retain the backup.

```
find $backupfolder -mtime +$keep_day -delete
```

Now that you have examined the components and understood what the shell script does, let's create a file and save the script in it.

Task D: Creating a Shell Script for MySQL Database Backups

1. Select **File** menu and then select **New File** to create a new shell script named `sqlbackup.sh`.

Enter the following code in the new file:

```
#!/bin/sh
# The above line tells the interpreter this code needs to be run as a shell script.
# Set the database name to a variable.
DATABASE='sakila'
# This will be printed on to the screen. In the case of cron job, it will be printed to the logs.
echo "Pulling Database: This may take a few minutes"
# Set the folder where the database backup will be stored
backupfolder=/home/theia/backups
# Number of days to store the backup
keep_day=30
sqlfile=$backupfolder/all-database-$(date +%d-%m-%Y_%H-%M-%S).sql
zipfile=$backupfolder/all-database-$(date +%d-%m-%Y_%H-%M-%S).gz
# Create a backup
if mysqldump  $DATABASE > $sqlfile ; then
   echo 'Sql dump created'
    # Compress backup
    if gzip -c $sqlfile > $zipfile; then
        echo 'The backup was successfully compressed'
    else
        echo 'Error compressing backupBackup was not created!'
        exit
    fi
    rm $sqlfile
else
   echo 'pg_dump return non-zero code No backup was created!'
   exit
fi
# Delete old backups
find $backupfolder -mtime +$keep_day -delete
```

2. Save your script using the**Save** option or pressing Commands+S (in Mac) or Ctrl+S (Windows).

3. Now you need to give executable permission for the shell script file, to the owner (yourself), by running the following command in the terminal. `u` stands for user or creator, `x` stands for execute, and `r` stands for read permission:

```
sudo chmod u+x+r sqlbackup.sh
```

4. You decided to create the backups in a folder, but that folder doesn't exist in the environment. You need to create it by running the following command:

```
  mkdir /home/theia/backups
```

Task E: Setting Up a Cron Job

- Cron is a system that helps Linux users schedule any task. It can be a shell script or a simple bash command.
- A cron job helps us automate our routine tasks and it can be hourly, daily, monthly, etc.
- A crontab (cron table) is a text file that specifies the schedule of cron jobs.
- Each line in the crontab file contains six fields separated by a space followed by the command to be run.

The first five fields may contain one or more values separated by a comma or a range of values separated by a hyphen.

* The asterisk operator means any value or always. If you have the asterisk symbol in the Hour field, it means the task will be performed each hour.

, The comma operator allows you to specify a list of values for repetition. For example, if you have 1,3,5 in the Hour field, the task will run at 1 a.m., 3 a.m. and 5 a.m.

- The hyphen operator allows you to specify a range of values. If you have 1-5 in the Day of week field, the task will run every weekday (from Monday to Friday).

/ The slash operator allows you to specify values that will be repeated over a certain interval between them. For example, if you have */4 in the Hour field, it means the action will be performed every four hours. It is same as specifying 0,4,8,12,16,20. Instead of an asterisk before the slash operator, you can also use a range of values. For example, 1-30/10 means the same as 1,11,21.

To understand how a crontab works, let's set up a cron job that happens every 2 minutes.

1. To start the crontab, run the following command in the terminal:

   ```
   crontab -e
   ```

This will open a crontab editor as follows.

2. Scroll to the bottom of the editor page using the down arrow key and copy and paste the following code:

   ```
   */2 * * * * /home/project/sqlbackup.sh > /home/project/backup.log
   ```

3. Press Ctrl+O followed by the Enter key to save the file.
4. Press Ctrl+X to quit the cron editor.

5. The cron service needs to be explicitly started. Start the cron service by executing the following command:

   ```
   sudo service cron start
   ```

6. After 2 minutes, execute the following command to check whether the backup file are created:

   ```
   ls -l /home/theia/backups
   ```

   In a real-life scenario, the cron service is a long-running service that runs in the background. To stop the cron job you can run `sudo service cron stop`.

7. In this example, the cron is set for backup every 2 minutes. Stop the cron service using the below command:

   ```
   sudo service cron stop
   ```

Practice Exercise

1. Change the crontab schedule to create a backup every week on Monday at 12:00 a.m.
   ▶ Click here for solution
2. Change the crontab schedule to create a backup every day at 6:00 a.m.
   ▶ Click here for solution

Task F: Truncate the Tables in the Database

Now that you have automated the backup task, let's replicate a scenario where the data is corrupted or lost and you will remove all the data in the database and restore the data from the backup.

We will create a truncate script that does the following:

- Connects to mysql RDBMS using the credentials.
- Lists tables using `show tables` and feeds the output using pipe(|) operator to the next command.
- Iterates through each table using a while loop and truncates the table.

1. Create a new file named **truncate.sh** under home/project.

2. Copy this script and paste it in the new file.

   ```
   #!/bin/sh

   DATABASE=sakila
   mysql -Nse 'show tables' sakila | \
       while read table; do mysql \
       -e "use sakila;SET FOREIGN_KEY_CHECKS=0;truncate table $table;SET FOREIGN_KEY_CHECKS=1;" ;done
   ```

3. Change the permission of the file by running the following command:

   ```
   sudo chmod u+x+r truncate.sh
   ```

4. Execute the script to truncate the tables.

   ```
   bash truncate.sh
   ```

5. To check whether the tables in the database are truncated, log in to the database with the credentials.

   ```
   mysql
   ```

The mysql interface appears

6. Switch to the **sakila** database.

```
use sakila;
```

7. Check all the tables in the database.

```
show tables;
```

8. Retrieve all the rows from staff table. If the `truncate` was successful, the output should be an **Empty set**.

```
select * from staff;
```

9. Quit the mysql prompt.

```
\q
```

Task G: Restore the Database

To restore the database:

- You pick up a compressed zip file present in the backup folder and unzip it to extract the sql file using the gunzip command.
- You connect to the mysql database and restore the database with the sqlfile.

1. In the terminal window, run the following command to find the list of backup files that have been created.

```
ls -l /home/theia/backups
```

2. Select the file that you want to restore the data from and copy the file name.
3. Unzip the file and extract the SQL file from the backup file.

```
gunzip /home/theia/backups/<backup zip file name>
```

4. Populate and restore the database with the sqlfile that results from the unzip operation.

```
mysql sakila < /home/theia/backups/<backup sql file name>
```

Once the command is executed, you will be prompted to enter your mysql login password. Paste the password that you have copied before using Ctrl+V and press Enter.

5. To check the restored database, go to the mysql prompt.

```
mysql
```

6. Use the **sakila** database:

```
use sakila;
```

7. Select all the rows from any one of the tables, as given below. You should find that the database is restored.

```
select * from staff;
```

8. Quit the MySQL command prompt session using the command below in the terminal and proceed to Task D:

```
\q
```

Practice Exercise

1. Create a shell script which takes the database name and back up directory as parameters and backs up the database as `<dbname>_timestamp.sql` in the backup directory. If the database doesn't exist, it should display appropriate message. If the backup dir doesn't exist, it should create one.

▶ Click here for the solution

2. Write a shell script which takes the database name and the script file as parameters and restores the database from the sql file.

▶ Click here for the solution

Optional Exercise

1. You can clean up the backups folder by using the following command:

```
sudo rm -rfv /home/theia/backups
```

Author(s)

- Lakshmi Holla

Other Contributor(s)

- Malika Singla
- Lavanya