Start all 01:30:00

Debugging and Solving Software Problems

1 hour 30 minutes · Free · ★★★★☆

Introduction
Accessing the virtual machine
Debug lab
Improve performance
Congratulations!
End your lab

Help

Introduction

You're a member of your company's IT department. A colleague that recently left the company wrote a program that's 90% complete. It's designed to read some data files with information on employees and then generate a report. It's up to you to finish the code. (this includes fixing any errors, bugs, and slowness that might be in the unfinished code.

Prerequisites:

You should have a sound knowledge of the following things prior to performing the lab:

- Debugging (gathering information, root cause analysis, and remediation)
- Identifying and understanding system performance (CPU, Network, DIO, Memory)
- Understanding and troubleshooting the environment around the program (file system, DB, etc.)

You'll have 90 minutes to complete this lab.

Start the lab

You'll need to start the lab before you can access the materials in the virtual machine VM. To do this, click the green "Start Lab" button at the top of the screen.

Note: For this lab you are going to access the **Linux VM** through your **Qwiklabs SSH Client**, and not use the **Google Console Open SCP Console** button is not available for this lab.

Start Lab

After you click the "Start Lab" button, you will see all the SSH connection details on the left-hand side of your screen. You should have a screen that looks like this:

Accessing the virtual machine

Please find one of the three relevant options below based on your device's operating system:

Note: Working with Qwiklabs may be similar to the work you perform as an **IT Support Specialist**, you'll be interfacing with a contractor's technology that requires multiple steps to access, and perhaps healthy amounts of documentation/instructions. Keep in mind that this is a feature of the lab - a virtual aid to IT Support that you'll be able to practice through this lab.

Option 1: Windows Users: Connecting to your VM

In this section, you will use the PuTTY Secure Shell (SSH) client and your VM's External IP address to connect.

Download your PPK key file

You can download the SSH private key file in the PuTTY compatible **PPK** format from the Qwiklabs Start Lab page. Click on **Download PPK**.

Connect to your VM using SSH and PuTTY

1. You can download Putty from here.
2. In the **Host Name (or IP address)** box, enter username@external_ip_address.

Note: Replace **username** and **external_ip_address** with values provided in the lab.

3. In the **Connection** list, expand **SSH**.
4. Then expand **Auth** by clicking on it once.
5. Now, select the **Credentials** from the **Auth** list.
6. In the **Private key file for authentication** box, browse to the PPK file that you downloaded and provide it.
7. Click on the **Open** button.

Note: PPK file is to be imported into PuTTY tool using the Browse option available in it. It should not be opened directly but only to be used in PuTTY.

8. Click **Yes** when prompted to allow a first connection to this remote SSH server because it doesn't have a RSA host key cached for authentication, you will not be prompted for a password.

Common Issues

If PuTTY fails to connect to your Linux VM, verify that:

- You entered **username@external ip address** in PuTTY
- You downloaded the fresh new PPK file for this lab from Qwiklabs
- You are using the downloaded PPK file in PuTTY

Option 2: OSX and Linux users: Connecting to your VM via SSH

Download your PPK private key file

You can download the private key file in PEM format from the Qwiklabs Start Lab page. Click on **Download PEM**.

Connect to the VM using the local Terminal application

A **terminal** is a program which provides a **text-based interface for typing commands**. Here you will use your terminal as an SSH client to connect with lab provided Linux VM.

1. Open the Terminal application.
   - To open the terminal in Linux use the shortcut key **Ctrl+Alt+t**.
   - To open terminal in **Mac** (OSX) enter enter **cmd + space** and search for **terminal**.
2. Enter the following commands.

Note: Substitute the **path/filename for the PEM file** you downloaded, **username** and **External IP address**.

You will more likely find the PEM file in **Downloads**. If you have not changed the permissions of your private key file, use the path of the PEM key set for **~/Downloads/qwiklabs-gcp-00000.pem**

```
chmod 400 ~/Downloads/qwiklabs-00000.pem
```

```
ssh -i ~/Downloads/qwiklabs-00000.pem username@external-ip-address
```


Option 3: Chrome OS users: Connecting to your VM via SSH

Note: Make sure you are not in **Incognito/Private mode** while launching the application.

Download your PPK private key file.

You can download the private key file in PEM format from the Qwiklabs Start Lab page. Click on **Download PEM**.

Connect to your VM

1. Add Secure Shell from here to your Chrome browser.
2. Click the Secure Shell app and click on **[New Connection]**.

3. In **Connection** section, enter the username given in the Connection details Panel of the lab, and for the **hostname** section enter the external IP of your VM instance that is mentioned in the Connection Details Panel of the lab.

4. In the **Identity** section, import the downloaded PEM key by clicking on the **Import...** button beside the field. Choose your PEM key and click on the **OPEN** button.

Note: If the key is still not available after importing it, refresh the application, and select it from the **Identity** drop down menu.

5. Once your key is uploaded, click on the **[SAVE]** button.

6. For any prompts, type **yes** to continue.
7. You have now successfully connected to your Linux VM.

You're now ready to continue with the lab!

Debug Issue

You have a `start_data_report.py` Python script with a bunch of functions (`gc_get_start_date()`, `hd_convert()` and others. This script will operate on the data files in the directory specified by the variable `DATA_DIR` and write the output to `report.txt`. The script then generates a report of all employees that worked on the given start date.

To list the files in the home directory, use the following command:

```
ls
```

Output

```
data1  data2  data3  data4  data5  data6  data7  data8  data9  data10
```

Grant the executable and writable file permission to the `start_data_report.py`

```
sudo chmod 777 ~/start_data_report.py
```

Now, run the python program `start_data_report.py`

```
~/start_data_report.py
```

Enter the values for the year, month, and day respectively as the prompt appears.

Output

```
Traceback (most recent call last):
  File "/home/student_00_0000000000/start_data_report.py", line 75, in <module>
    start = gc_get_start_date()
  File "/home/student_00_0000000000/start_data_report.py", line 15, in gc_get_start_date
    year = int(raw_input('Enter value for the year: '))
NameError: name 'raw_input' is not defined
```

The program crashes with a **TypeError**. This is because it reads the value entered as prompt as a string. Refer to the function `gc_get_start_date()` within the script. The arguments passed to the `gc_get_start_date()` function should be of type: the input values are expected to be integers.

In order to fix this **ERROR**, open `start_data_report.py` by using the following command:

```
nano ~/start_data_report.py
```

Now, search for **gc_start_date()** function and operand the string variable that's taken from user input to the integer. Here, we have to explicitly cast the data type of these three variables `year`, `month`, and `day` from string to integer.

```
gc_start = int(raw_input('Enter value for the year: '))
```

Similarly you can cast the values of `month` and `day` to an integer.

The **gc_set_start()** function should now looks like this

```
def gc_set_start():
    global gc_start, gc_end, gc_days
    gc_start = int(raw_input('Enter value for the year: '))
    gc_end = int(raw_input('Enter value for the month: '))
    gc_days = int(raw_input('Enter value for the day: '))
    print('Searching for employees who started on ', gc_start, '-', gc_end, '-', gc_days)
```

Save the `start_data_report.py` script file by clicking Ctrl+x, the Enter key, and Y key.

Run the `start_data_report.py` Python script:

```
~/start_data_report.py
```

Output

```
Traceback (most recent call last):
  File "/home/student_00_0000000000/start_data_report.py", line 75, in <module>
    start = gc_get_start_date()
  File "/home/student_00_0000000000/start_data_report.py", line 15, in gc_get_start_date
    year = int(raw_input('Enter value for the year: '))
NameError: name 'raw_input' is not defined
```

Click Check my progress to verify the objective.

Debug and fix Issue

Check my progress

Improve performance

Once you debug the issue, the program will start processing the flat file it takes as input. However, it takes a long time to generate the report. Instead of processing the report quickly. You need to debug why the program is slow and fixes it. In this section, you need to find bottlenecks, improve the code, and make it fresh faster.

The problem with the script is that it's downloading the whole file and then processing it. Instead, it should be processing the data as it's being downloaded. It compiles for 20:01:01. An optimized script should generate reports for the same date within a few seconds.

To check the execution time of a script, add a prefix "time" and run the script.

Example

```
time ~/test.py
```

Now, run the `start_data_report.py` script using `time` command:

```
time ~/start_data_report.py
```

This is a pretty challenging task that you have to complete by modifying the **gc_core_or_invest()** function.

1. Download the file only once from the URL.
2. Pre-process it so that the same calculation doesn't need to be done over and over again. This can be done in two ways. You can choose any one of them.
   - To create a dictionary with the start dates and then use the dates in the dictionary to find the employees who started.
   - To sort the data by start_date and then use it to find the dates.

Choose any one of the above preprocessing options and modify the script accordingly.

Once you've completed modifying the Python script, save the file by clicking Ctrl+x the Enter key and Y key.

Run the `start_data_report.py` python script:

```
~/start_data_report.py
```

Output

```
Enter value for the year: 2001
Enter value for the month: 1
Enter value for the day: 1
('Searching for employees who started on ', 2001, '-', 1, '-', 1)
('Number of employees found: ', 1)
('Time taken to generate report: ', 1.0)
('Report generated successfully!!!')
real    0m1.000s
user    0m0.000s
sys     0m0.000s
```

Now, you've improved the performance of the script.

Click Check my progress to verify the objective.

Improve performance

Check my progress

Congratulations!

Congrats! You've successfully fixed errors, bugs, and increased the performance of execution. Debugging an issue from a program and reducing execution time by fixing a repeatable call will be beneficial as an IT Specialist.

End your lab

When you have completed your lab, click **End Lab**. Qwiklabs removes the resources you've used and cleans the account for you.

You will be given an opportunity to rate the lab experience. Select the applicable number of stars, type a comment, and then click **Submit**.

The number of stars indicates the following:

- 1 star = Very dissatisfied
- 2 stars = Dissatisfied
- 3 stars = Neutral
- 4 stars = Satisfied
- 5 stars = Very satisfied

You can close the dialog box if you don't want to provide feedback.

For feedback, suggestions, or corrections, please use the **Support** tab.