Start Lab 01:30:00

Working with Python Scripts

1 hour 30 minutes Free

−/20

Introduction

Welcome to your first lab on fixing problems in Python. In this lab, you'll first have to fix an incorrect Python script. This includes:

- Fixing the file permissions to make it executable.
- Fixing a bug in the code.

After that, you'll write your own Python module and use it from the original script.

You'll have 90 minutes to complete this lab.

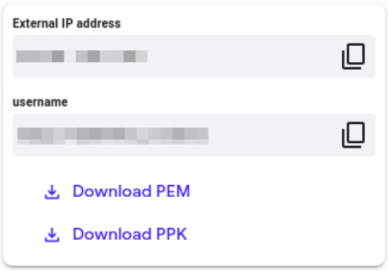
Start the lab

You'll need to start the lab before you can access the materials in the virtual machine OS. 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 **local SSH Client**, and not use the **Google Console** (**Open GCP 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'd perform as an **IT Support Specialist**; you'll be interfacing with a cutting-edge technology that requires multiple steps to access, and perhaps healthy doses of patience and persistence(!). You'll also be using **SSH** to enter the labs -- a critical skill in IT Support that you'll be able to practice through the labs.

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

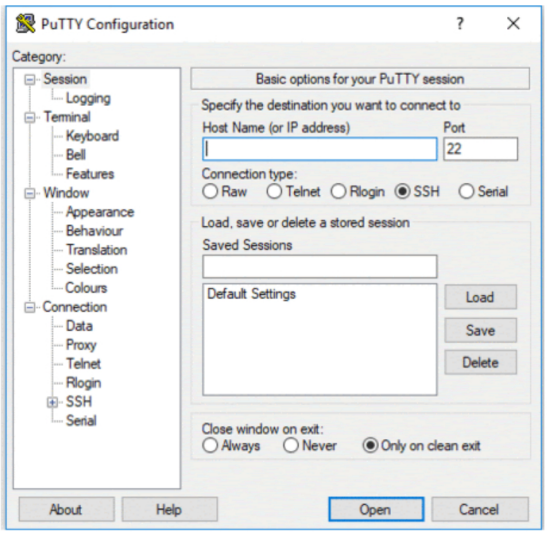
You can download the VM's private key file in the PuTTY-compatible **PPK** format from the Qwiklabs Start Lab page. Click on **Download PPK**.

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 **Category** list, expand **SSH**.

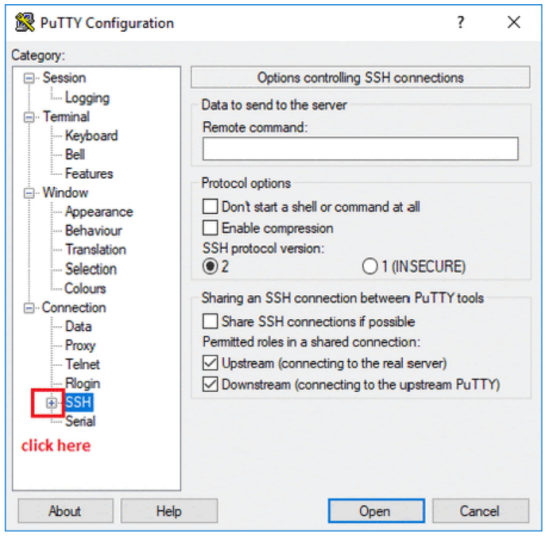
4. Click **Auth** (don't expand it).

5. In the **Private key file for authentication** box, browse to the PPK file that you

downloaded and double-click it.

6. 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.



7. Click **Yes** when prompted to allow a first connection to this remote SSH server. Because you are using a key pair 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 VM's 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 **cmd + space** and search for

- To open terminal in **Mac** (OSX) enter **cmd + space** and search for **terminal**.

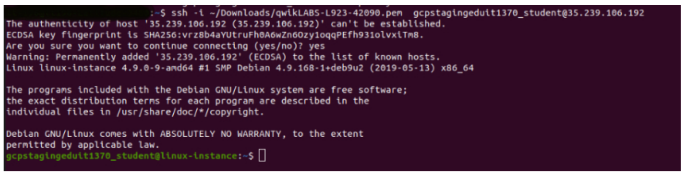
2. Enter the following commands.

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

You will most likely find the PEM file in **Downloads**. If you have not changed the download settings of your system, then the path of the PEM key will be **~/Downloads/qwikLABS-XXXXX.pem**

```
chmod 600 ~/Downloads/qwikLABS-XXXXX.pem
```

```
ssh -i ~/Downloads/qwikLABS-XXXXX.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 VM's 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. Open the Secure Shell app and click on **[New Connection]**.

3. In the **username** 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 **[ENTER] Connect** button below.

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!

Fix file permissions

We have a python script ready for you. From your home directory (~), use the following command to navigate to the scripts directory:

```
cd scripts
```

List the files to find the script using the following command:

```
ls
```

To view the contents of this file, enter the following command:

```
cat health_checks.py
```

This Python file consists of a script to check disk and cpu usage. You can see shutil and psutil modules are imported here.

The *shutil* module offers a number of high-level operations on files and collections of files. In particular, it provides functions that support file copying and removal. It comes under Python's standard utility modules. *disk_usage()* method is used to get disk usage statistics of the given path. This method returns a named tuple with the attributes *total*, *used*, and *free*. The *total* attribute represents the total amount of space, the *used* attribute represents the amount

of used space, and the *free* attribute represents the amount of available space, in bytes.

psutil (Python system and process utilities) is a cross-platform library for retrieving information on the processes currently running and system utilization (CPU, memory, disks, network, sensors) in Python. It's useful mainly for system monitoring, profiling, limiting process resources, and managing running processes. *cpu_percent()* returns a float showing the current system-wide CPU use as a percentage. When the interval is 0.0 or None (default), the function compares process times to system CPU times elapsed since the last call, returning immediately (non-blocking). That means that the first time it's called it will return a meaningful 0.0 value. When the interval is > 0.0, the function compares process times to system CPU times elapsed before and after the interval (blocking).

This script begins with a line containing the #! character combination, which is commonly called hash bang or shebang and continues with the path to the interpreter.

#!/usr/bin/env python3 uses the operating system env command, which locates and executes Python by searching the PATH environment variable. Unlike Windows, the Python interpreter is usually already in the $PATH variable on linux, so you don't have to add it.

Now that you understand what the script does, and the functions within it, let's run the Python file using the following command:

```
./health_checks.py
```

We got a permission denied error.

This is because the above command tries to run your script directly as a program. The program is parsed by the interpreter specified in the first line of the script, i.e. shebang. If the kernel finds that the first two bytes are #! it uses the rest of the line as an interpreter and passes the file as an argument. So, to do this, the file needs to have execute permission.

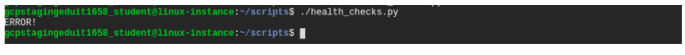
To run this file, we need it to have execute permission (x). Let's update the file permissions and then try running the file. Use the following command to add execute permission to the file:

```
sudo chmod +x health_checks.py
```

Now try running the file again by using the following command:

```
./health_checks.py
```

This time, the output shows "ERROR".



Click *Check my progress* to verify the objective.

Fix file permissions

Check my progress

Debug the issue

The Python script returns ERROR only if there's not enough disk usage or CPU usage. Try to debug this issue.

Hint: The problem is that the function **check_cpu_usage** should return **true** if the CPU usage is less than 75%, but in this case, it returns false.

Use a nano editor to open the file health_checks.py.

```
nano health_checks.py
```

Make the necessary changes now. And once the changes are done, save the file by clicking Ctrl-o, enter key and Ctrl-x.

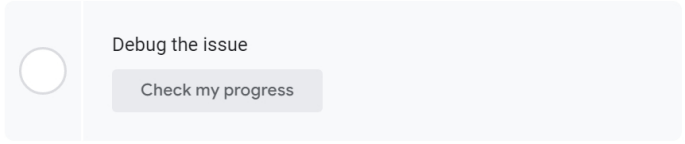
Once you have debugged the issue, try running the file again by using the command:

```
./health_checks.py
```

This time, if the script is correct, the output should be "Everything ok".

Congratulations! You fixed the script!

Click *Check my progress* to verify the objective.



Create a new Python module

In this section, you are going to write a Python module. A module is a file containing Python definitions and statements. The file name is the module name with the suffix .py appended.

The module you are going to write will be used to test the network connections. We will guide you step by step through this process. Throughout the lab, we will refer to this module as the network module.

Let's start writing this network module. Since, the network module will check whether the network is correctly configured on the computer, we will use the requests module.

What is the requests module?

Requests is a Python module that you can use to send all kinds of HTTP requests. It's an easy-to-use library with a lot of features ranging from passing parameters in URLs to sending custom headers and SSL verification. You can

add headers, form data, multi-part files, and parameters with simple Python dictionaries.You can then access the response data using the same request.

To use the requests module, you first need to install it. Use the following command to install the request module. If you receive any prompts, continue by clicking **Y**.

```
sudo apt install python3-requests
```

Create a file named `network.py`. The file should be created in the same directory as `health_checks.py`, i.e., **scripts**. If you are not present in the scripts directory, navigate to the scripts directory first and then create the file.

```
cd ~/scripts
```

Use nano editor to create a new file `network.py`:

```
nano network.py
```

Add a shebang line to define where the interpreter is located. In this case, the shebang line would be /usr/bin/env python3.

```
#!/usr/bin/env python3
```

Import the request module into the file using the import statements.

```
import requests
```

To check whether the local host is correctly configured, we use the socket module.

Now, import the socket module.

```
import socket
```

Next, write a function **check_localhost**, which checks whether the local host is correctly configured. We do this by calling the *gethostbyname* within the function.

```
localhost = socket.gethostbyname('localhost')
```

The above function translates a host name to IPv4 address format. Pass the parameter **localhost** to the function *gethostbyname*. The result for this function should be **127.0.0.1**.

Edit the function **check_localhost** so that it returns true if the function returns **127.0.0.1**.

Now, we will write another function called **check_connectivity**. This checks whether the computer can make successful calls to the internet.

A request is when you ping a website for information. The **Requests** library is designed for this task. You will use the request module for this, and call the GET method by passing *http://www.google.com* as the parameter.

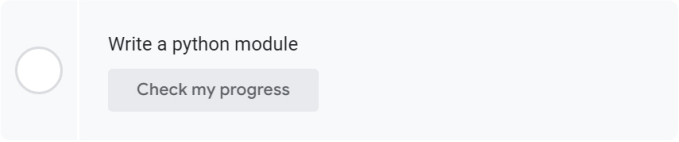
```
request = requests.get("http://www.google.com")
```

This returns the website's status code. This status code is an integer value. Now, assign the result to a response variable and check the *status_code* attribute of that variable. It should return **200**.

Edit the function **check_connectivity** so that it returns true if the function returns **200** status_code.

Once you have finished editing the file, press Ctrl-o, Enter, and Ctrl-x to exit.

Click *Check my progress* to verify the objective.



Use the Python module

Now, you're going to re-edit the file **health_checks.py** to make it call the checks in the network module. For this, you will need to import the module into health_checks.py script.

To do this, open the script `health_checks.py`

```
nano health_checks.py
```

Now import network module at the beginning of the file.

```
from network import *
```

Call the checks to the network module by adding an elif clause after the if clause in the script `health_checks.py`.

Replace the **else** part with an elif clause.

```
elif check_localhost() and check_connectivity():
    print("Everything ok")
```

Add an **else** part at the end of the file.

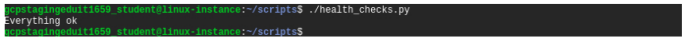
```
else:
    print("Network checks failed")
```

Once you have completed editing the file, press Ctrl-o, Enter, and Ctrl-x to exit.

Now, run the file.

```
./health_checks.py
```

It should return "Everything ok".



Click *Check my progress* to verify the objective.

Click Check my progress to verify the objective.

Use the python module

Check my progress

Congratulations!

You've successfully fixed an incorrect Python script, created a new Python module, and used it in the original script. These are common tasks for IT support roles, and mastering the skills you've practiced in this lab will greatly help you in your IT career. You can now close the RDP/SSH window, manually end the lab, and continue onto the next module. This lab will automatically end when the time runs out.

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.