

# Introduction

It's time to put your new skills to the test! In this lab, you'll have to find the users using an old email domain in a big list using regular expressions. To do so, you'll need to write a script that includes:

- Replacing the old domain name (abc.edu) with a new domain name (xyz.edu).
  - Storing all domain names, including the updated ones, in a new file.
- You'll have 90 minutes to complete this lab.

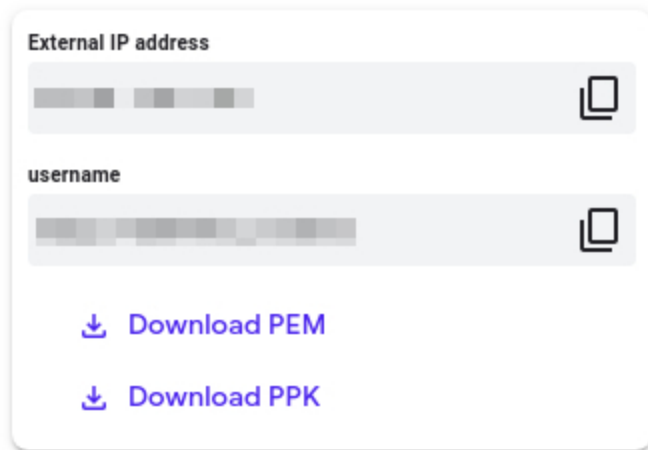
## Start the lab

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).

A green rectangular button with the text "Start Lab" in white.

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:



External IP address

username

Download PEM

Download PPK

## 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**.

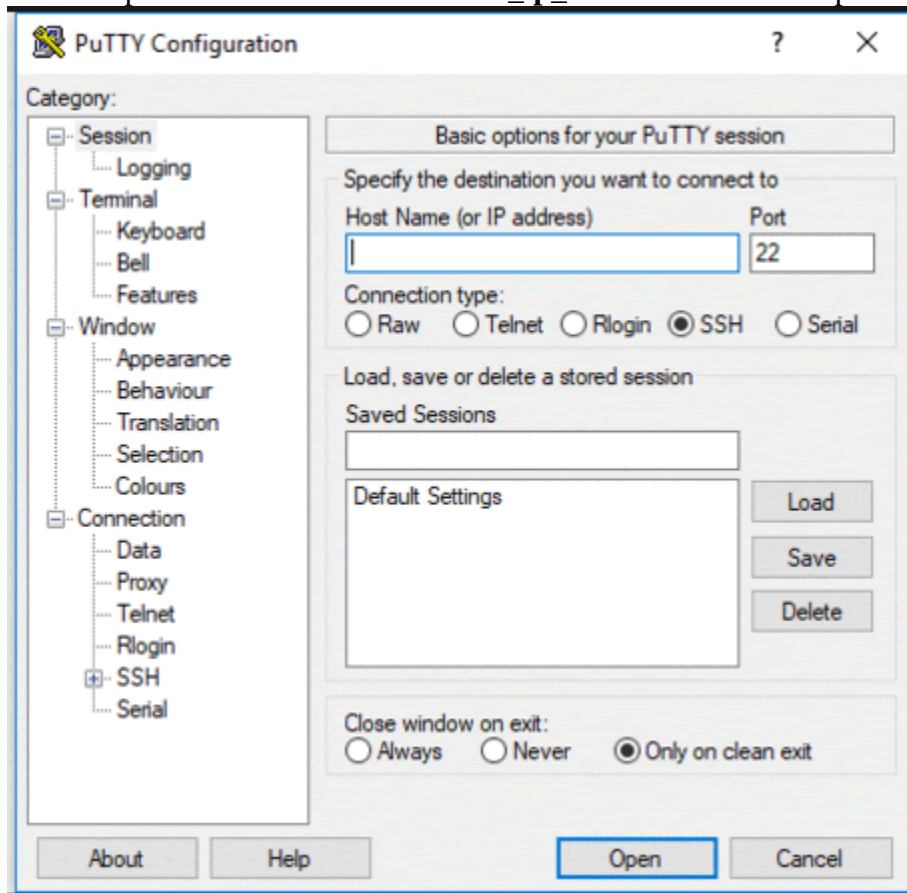
↓ Download PEM

↓ Download PPK

### Connect to your VM using SSH and PuTTY

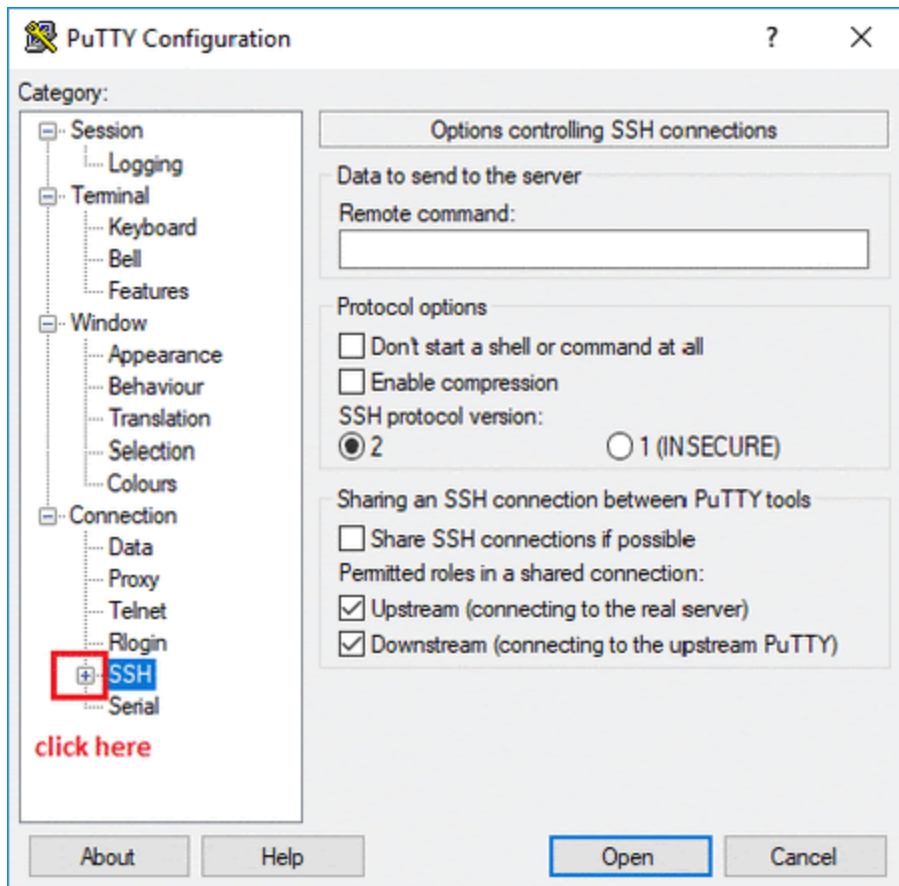
1. You can download Putty from [here](#)
2. In the **Host Name (or IP address)** box, enter username@external\_ip\_address.

**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 **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

gcpstagingdui1370_student@linux-instance:~$ ssh -i ~/Downloads/qwikLABS-L923-42090.pem gcpstagingdui1370_student@35.239.106.192
The authenticity of host '35.239.106.192 (35.239.106.192)' can't be established.
ECDSA key fingerprint is SHA256:vrz8b4aYUtruFh0A6wZn60zy1oqqPEfh931olvxITm8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '35.239.106.192' (ECDSA) to the list of known hosts.
Linux linux-instance 4.9.0-9-amd64 #1 SMP Debian 4.9.168-1+deb9u2 (2019-05-13) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
gcpstagingdui1370_student@linux-instance:~$
```

## 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]**.

This is a screenshot of the "New Connection" dialog box in the Secure Shell application. The dialog has a title bar that says "[New Connection]". Below the title bar is a large text input field with the placeholder text "username@hostname or free form text". Underneath this is a section with three input fields: "username", "hostname", and "port". Below these is a field for "SSH relay server options". Further down is a section for "Identity" with a dropdown menu currently set to "[default]" and an "Import..." button next to it. Below that is a field for "SSH Arguments" with the placeholder "extra command line arguments". Then is a field for "Current profile" set to "default". Finally, there is a field for "Mount Path" with the placeholder "the default path is the user's home directory". At the bottom left are buttons for "[DEL] Delete" and "Options". At the bottom right are buttons for "SFTP Mount", "SFTP", and "[ENTER] Connect". A red arrow points from the left towards the "[New Connection]" title bar.

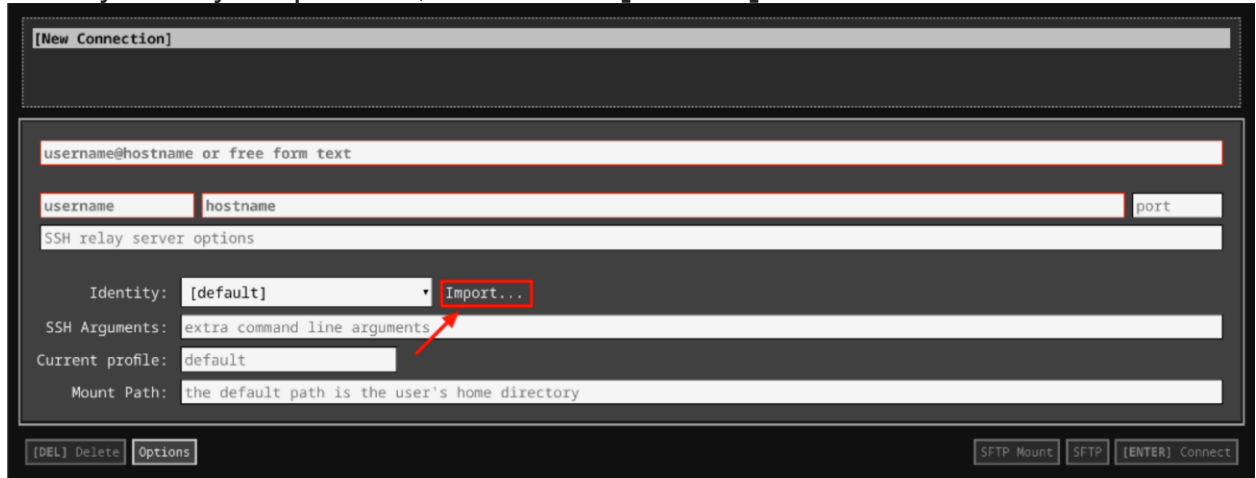
3. In the **username** section, enter the username given in the Connection Details Panel of the lab. And for the **hostname** section, enter the external IP of your VM instance that is mentioned in the Connection Details Panel of the lab.

This is another screenshot of the "New Connection" dialog box, similar to the one above. In this version, the "username" and "hostname" input fields are highlighted with red rectangular boxes. A red arrow points from the left towards the "username" field.

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!

## Prerequisites

We've created a list containing user names and their email addresses. Navigate to the **data** directory using the following command:

```
cd data
```

To find the data, list the files using the following command:

```
ls
```

You can now see a file named **user\_emails.csv**. This is where you will find the required data. To view the contents of the **user\_emails.csv** file, enter the following command:

```
cat user_emails.csv
```

You can also access a python script that contains function definitions for the task. Navigate to the **scripts** directory using the following command:

```
cd ~/scripts
```

Now list the contents within the **scripts** directory using the following command:

```
ls
```

Here, you will find a file named **script.py**. The aim of this script is to use regex to find all instances of the old domain ("abc.edu") in the **user\_emails.csv** file and then replace them with the new domain ("xyz.edu").

This file already has the functions defined for you. You have to now complete the function's body to make it work as intended.

Let's update the file's permissions.

```
sudo chmod 777 script.py
```

We will use nano editor to edit **script.py** file.

```
nano script.py
```

Before we start writing the script, let's import libraries to use in the script. To do this, open the file with nano editor. To deal with CSV file operations, Python has a CSV module that effectively handles CSV data. Let's import the CSV module using the following:

```
import csv
```

Import the regex Python module (i.e the regular expression module) to this script. A regular expression(RegEx) is a sequence of characters that defines a search pattern.

```
import re
```

## Identify the old domain



In this section, we will write the body of the function named *contains\_domain*. This function uses *regex* to identify the domain of the user email addresses in the *user\_emails.csv* file.

The function takes address and domain as parameters, and its primary objective is to check whether an email address belongs to the old domain(abc.edu).

To do this, we will use a regular expression stored in the variable named *domain\_pattern*. This variable will now match email addresses of a particular domain. If the old domain is found, then the function returns true.

```
domain_pattern = r'[\w\.-]+@'+domain+'$'
if re.match(domain_pattern, address):
    return True
```

The function *contains\_domain* should now look like this:

```
def contains_domain(address, domain):
    domain_pattern = r'[\w\.-]+@'+domain+'$'
    if re.match(domain_pattern, address):
        return True
    return False
```

## Replace the domain name

In this section, we will replace the old domain name with the new one. The second function defined in the **script.py** file is *replace\_domain*.

The *replace\_domain* function takes in one email address at a time, as well as the email's old domain name and its new domain name. This function's primary objective is to replace the email addresses containing the old domain name with new domain name.

In order to replace the domain name, we will use the regular expression module and make a pattern that identifies sub-strings containing the old domain name within email addresses. We will then store this pattern in a variable called *old\_domain\_pattern*. Next, we will use substitution function *sub()* from **re** module to replace the old domain name with the new one and return the updated email address.

```
old_domain_pattern = r' ' + old_domain + '$'
address = re.sub(old_domain_pattern, new_domain, address)
```

The function *replace\_domain* should now look similar to the following:

```
def replace_domain(address, old_domain, new_domain):
```

```
old_domain_pattern = r' ' + old_domain + '$'  
address = re.sub(old_domain_pattern, new_domain, address)  
return address
```

## Write a CSV file with replaced domain from main

In this section, we're going to call the above defined functions: `contains_domain()` and `replace_domain` from the `main()`. This will allow us to find the old domain email address, replace it with the newer one, and write the updated list to a CSV file in the data directory.

In the previous sections, you might have seen variables named `old_domain` and `new_domain`, which are passed as parameters to the functions. Let's declare them here within `main()`.

```
old_domain, new_domain = 'abc.edu', 'xyz.edu'
```

Now store the path of the list **user\_emails.csv** in the variable `csv_file_location`. Also, give a file path for the resulting updated list within the variable `report_file`. This updated list should be generated within the **data** directory.

```
csv_file_location = '<csv-file-location>'  
report_file = '<data-directory>' + '/updated user emails.csv'
```

Replace `<csv_file_location>` by the path to the `user_emails.csv`. `<csv_file_location>` is similar to the path `/home/<username>/data/user_emails.csv`. For variable `report_file`, replace `<data_directory>` by the path to `/data` directory. `<data_directory>` is similar to the path `/home/<username>/data`. Replace `<username>` with the one mentioned in the Connection Details Panel on the left-hand side.

Then, initialize an empty list where you will store the user email addresses. This is then passed to the function `contains_domain`, where a regular expression is used to match them and finally replace the domains using the `replace_domain` function.

Next, initialize the two different lists, **old\_domain\_email\_list** and **new\_domain\_email\_list**.

The **old\_domain\_email\_list** will contain all the email addresses with the old domain that the regex would match within the function **contains\_domain**. Since

the function *contains\_domain* takes in email address passed as parameter, we will iterate over the **user\_email\_list** to pass email addresses one by one. For every matched email address, we will append it to the list **old\_domain\_email\_list**.

```
user_email_list = []
old_domain_email_list = []
new_domain_email_list = []
```

The CSV module imported earlier implements classes to read and write tabular data in CSV format. The CSV library provides functionality to both read from and write to CSV files. In this case, we are first going to read data from the list (which is a CSV file). The data is read from the **user\_emails.csv** file and passed to the **user\_data\_list**. So the **user\_data\_list** now contains the same information as that present in **user\_emails.csv** file. While we do this, we will also add all the email addresses into the **user\_email\_list** that we initialized in the previous step.

```
with open(csv_file_location, 'r') as f:
    user_data_list = list(csv.reader(f))
    user_email_list = [data[1].strip() for data in user_data_list[1:]]
```

The list **old\_domain\_email\_list** should contain all the email addresses with the old domain. This will be checked by the function *contains\_domain*. The function *replace\_domain* will then take in the email addresses (with old domain) and replace them with the new domains.

```
for email_address in user_email_list:
    if contains_domain(email_address, old_domain):
        old_domain_email_list.append(email_address)
        replaced_email = replace_domain(email_address, old_domain,
new_domain)
        new_domain_email_list.append(replaced_email)
```

Now, let's define the headers for our output file through the **user\_data\_list**, which contains all the data read from **user\_emails.csv** file.

```
email_key = ' ' + 'Email Address'
email_index = user_data_list[0].index(email_key)
```

Next, replace the email addresses within the **user\_data\_list** (which initially had all the user names and respective email addresses read from the **user\_emails.csv** file) by iterating over the **new\_domain\_email\_list**, and replacing the corresponding values in **user\_data\_list**.

Finally, close the file using the `close()` method. A closed file no longer be read or written. It is good practice to use the `close()` method to close a file.

```
for user in user_data_list[1:]:
    for old_domain, new_domain in zip(old_domain_email_list,
new_domain_email_list):
        if user[email_index] == ' ' + old_domain:
            user[email_index] = ' ' + new_domain
f.close()
```

Now write the list to an output file, which we declared at the beginning of the script within the variable **report\_file**.

```
with open(report_file, 'w+') as output_file:
    writer = csv.writer(output_file)
    writer.writerows(user_data_list)
```

Finally, call the `main()` method.

The script should now look like this:

```
f.close()

with open(report_file, 'w+') as output_file:
    writer = csv.writer(output_file)
    writer.writerows(user_data_list)
    output_file.close()

main()
```

Save the file by clicking **Ctrl-o**, **Enter** key, and **Ctrl-x**.  
Now run the file.

```
./script.py
```

On a successful run, this should generate a new file  
named **updated\_user\_emails** within the **data** directory.  
To view the newly generated file, enter the following command:

```
ls ~/data
```

You should now be able to see a new file named **updated\_user\_emails.csv**. To  
view the contents of this file, enter the following command:

```
cat ~/data/updated user emails.csv
```

Great job! You have successfully replaced the old domain names with the new  
ones and generated a new file containing all the user names with their respective  
email addresses.

The report file should be similar to the one below image:

```
gcpstagingeduit1884_student@linux-instance:~/data$ cat updated_user_emails.csv
Full Name, Email Address
Blossom Gill, blossom@xyz.edu
Hayes Delgado, nonummy@utnisia.com
Petra Jones, ac@xyz.edu
Oleg Noel, noel@liberomauris.ca
Ahmed Miller, ahmed.miller@nequenonquam.co.uk
Macaulay Douglas, mdouglas@xyz.edu
Aurora Grant, enim.non@xyz.edu
Madison McIntosh, mcintosh@nisiaenean.net
Montana Powell, montanap@sem magna.org
Rogan Robinson, rr.robinson@xyz.edu
Simon Rivera, sri@xyz.edu
Benedict Pacheco, bpacheco@xyz.edu
Maisie Hendrix, mai.hendrix@xyz.edu
Xavier Gould, xlg@utnisia.net
Oren Rollins, oren@sem magna.com
Flavia Santiago, flavia@utnisia.net
Jackson Owens, jackowens@xyz.edu
Britanni Humphrey, britanni@ut.net
Kirk Nixon, kirknixon@xyz.edu
Bree Campbell, bree@utnisia.net
gcpstagingeduit1884_student@linux-instance:~/data$
```