

Working with Regular Expressions

1 hour 30 minutesFree

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.

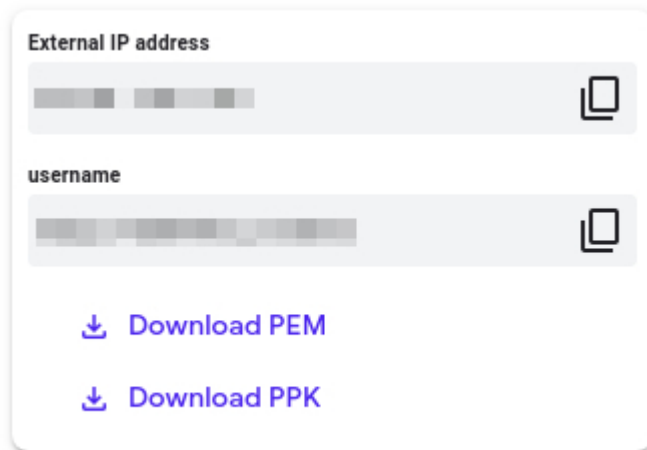
Start the lab

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).

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:



The screenshot shows a white rectangular panel with rounded corners. At the top, it has the label "External IP address" above a light gray input field containing a placeholder IP address (10.10.10.10). To the right of the input field is a copy icon. Below this, the label "username" is above another light gray input field containing a placeholder username (root). To the right of this input field is also a copy icon. At the bottom of the panel, there are two blue links, each preceded by a download icon: "Download PEM" and "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](#)

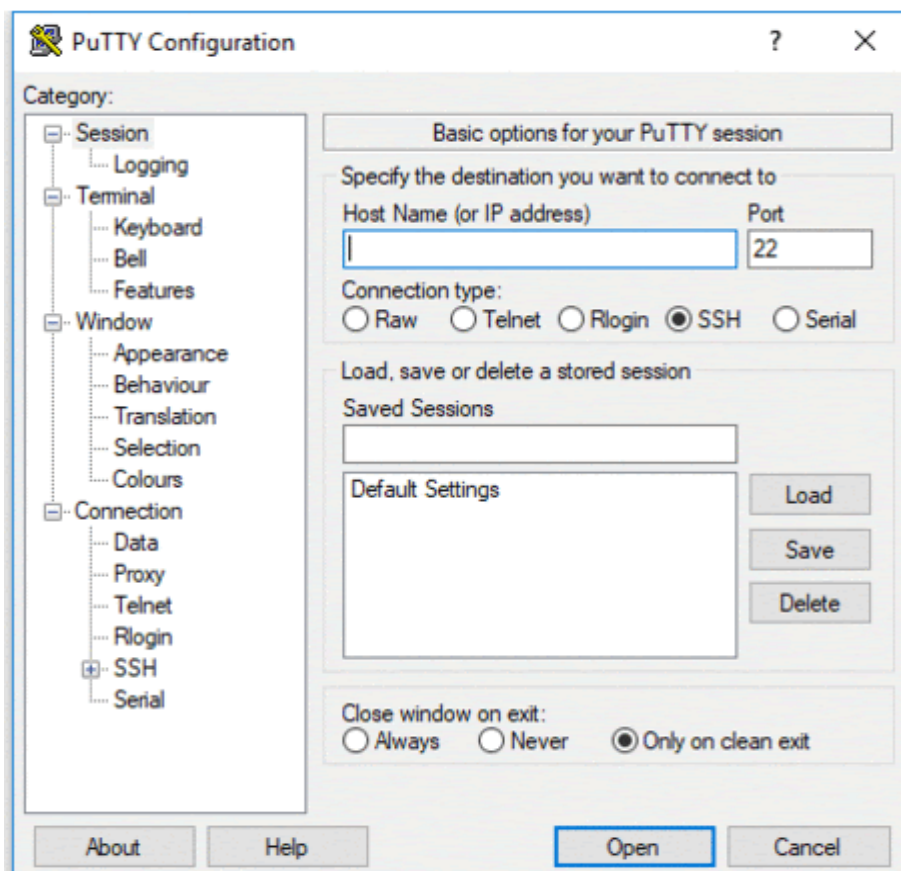
 [Download PPK](#)



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.

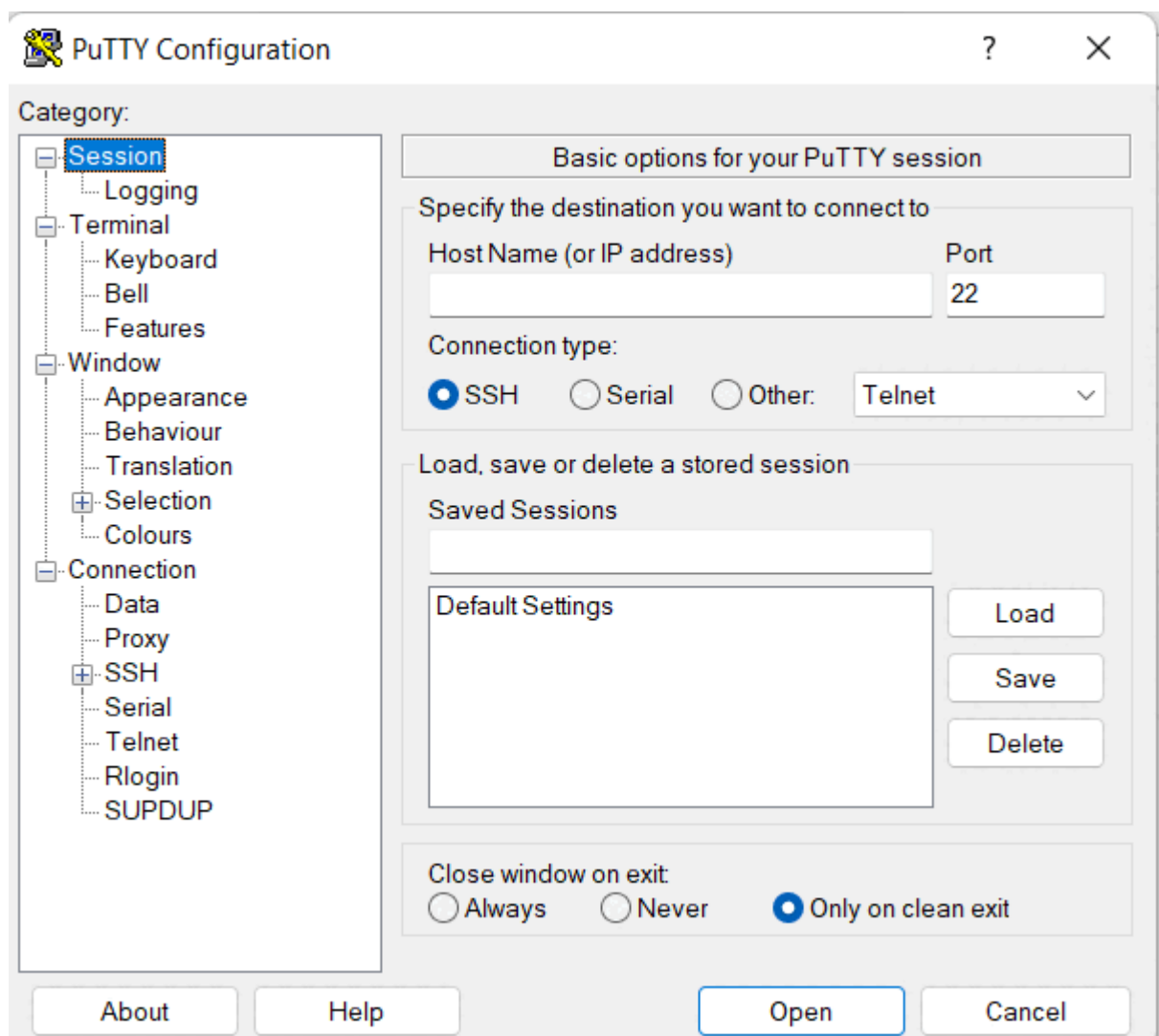
Note: Replace **username** and **external_ip_address** with values provided in the lab.



3. In the **Connection** list, expand **SSH**.

4. Then expand **Auth** by clicking on + icon.
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 double-click 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 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
```

Copied!

content_copy

```
ssh -i ~/Downloads/qwikLABS-XXXXX.pem username@External Ip Address
```

Copied!

content_copy

```
~$ ssh -i ~/Downloads/qwikLABS-L923-42090.pem gcpstagingedit1370_stu
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.
gcpstagingedit1370_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**].

[New Connection]

username@hostname or free form

username

hostname

SSH relay server options

Identity: [default]

SSH Arguments: extra command line

Current profile: default

Mount Path: the default path

[DEL] Delete

Options

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.

[New Connection]

username@hostname or free form text

username

hostname

SSH relay server options

Identity: [default]

SSH Arguments: extra command line arg

Current profile: default

Mount Path: the default path is th

[DEL] Delete

Options

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.

[New Connection]

username@hostname or free form text

username

hostname

SSH relay server options

Identity: [default]

SSH Arguments: extra command line arguments

Current profile: default

Mount Path: the default path is the

[DEL] Delete

Options

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
Copied!
content_copy
```

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

```
ls
Copied!
content_copy
```

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
Copied!
content_copy
```

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
Copied!
content_copy
```

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

```
ls
```

Copied!

content_copy

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
```

Copied!

content_copy

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

```
nano script.py
```

Copied!

content_copy

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
```

Copied!

content_copy

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
```

Copied!

content_copy

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  
Copied!  
content_copy
```

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  
Copied!  
content_copy
```

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)  
Copied!  
content_copy
```

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  
Copied!  
content_copy
```

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'
```

Copied!

content_copy

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'
```

Copied!

content_copy

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 = []
```

Copied!

content_copy

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:]]
```

Copied!

content_copy

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)
```

Copied!

content_copy

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)
```

Copied!

content_copy

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()
```

Copied!

content_copy

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)
```

```
output_file.close()
```

Copied!

content_copy

Finally, call the main() method.

```
main()
```

Copied!

content_copy

The script should now look like this:

```
#!/usr/bin/env python3
import re
import csv
def contains_domain(address, domain):
    """Returns True if the email address contains the given, domain, in the domain position, false if not."""
    domain = r'[w\.-]+@'+domain+'$'
    if re.match(domain, address):
        return True
    return False
def replace_domain(address, old_domain, new_domain):
    """Replaces the old domain with the new domain in the received address."""
    old_domain_pattern = r'' + old_domain + '$'
    address = re.sub(old_domain_pattern, new_domain, address)
    return address
def main():
    """Processes the list of emails, replacing any instances of the old domain with the new domain."""
    old_domain, new_domain = 'abc.edu', 'xyz.edu'
    csv_file_location = '<csv_file_location>'
    report_file = '<path_to_home_directory>' + '/updated_user_emails.csv'
    user_email_list = []
    old_domain_email_list = []
    new_domain_email_list = []
    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:]]
        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)
        email_key = ' ' + 'Email Address'
        email_index = user_data_list[0].index(email_key)
        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()
    with open(report_file, 'w+') as output_file:
        writer = csv.writer(output_file)
        writer.writerows(user_data_list)
        output_file.close()
main()
Copied!
content_copy
```

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

Now run the file.

```
./script.py  
Copied!  
content_copy
```

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  
Copied!  
content_copy
```

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  
Copied!  
content_copy
```

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 u
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@semmagna.org
Rogan Robinson, rr.robinson@xyz.edu
Simon Rivera, sri@xyz.edu
Benedict Pacheco, bpacheco@xyz.edu
Maisie Hendrix, mai.hendrix@xyz.edu
Xaviera Gould, xlg@utnisia.net
Oren Rollins, oren@semmagna.com
Flavia Santiago, flavia@utnisia.net
Jackson Owens, jackowens@xyz.edu
Britanni Humphrey, britanni@ut.net
Kirk Nixon, kirknixon@xyz.edu
Bree Campbell, breee@utnisia.net
gcpstagingeduit1884_student@linux-instance:~/data$
```

Click Check my progress to verify the objective.

Replace domain name

Check my progress

Congratulations!

You successfully wrote a Python script that achieves two tasks. First, it changed the domain name to the new domain name. Second, it stored all the updated domain names in a new file.

Creating reports using Python with CSV and using regular expressions to find a pattern in a string are very useful tools in IT support. You'll likely complete similar tasks regularly throughout your career, so feel free to go through this lab as many times as you need. Remember, practice makes perfect.

You can now close the RDP/SSH window. You can manually end the lab, or it 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.