Create, modify, and remove file and folder permissions in Linux

1 hour 1 Credit

Rate Lab

Introduction

In this lab, you'll learn the foundations of how managing user permissions work on a Linux machine. Using the new commands you learned in Bash, you'll fix up the permissions of some files and folders.

Head's up: You'll experience a delay as the labs initially load (particularly for Windows labs). So, please **wait a couple of minutes for the labs to load**. Please also make sure to access the labs **directly through Coursera** and not in the Qwiklabs catalog. If you access the labs through the Qwiklabs catalog, you will *not* receive a grade. (As you know, a passing grade is required to matriculate through the course.) The grade is calculated when the lab is complete, so be sure to hit "**End Lab**" when you're done!

You'll have 60 minutes to complete this lab.

What you'll do

- Familiarize yourself with the process of changing permissions within a file and folder in Linux
- Change the ownership of a specific file and folder

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).

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

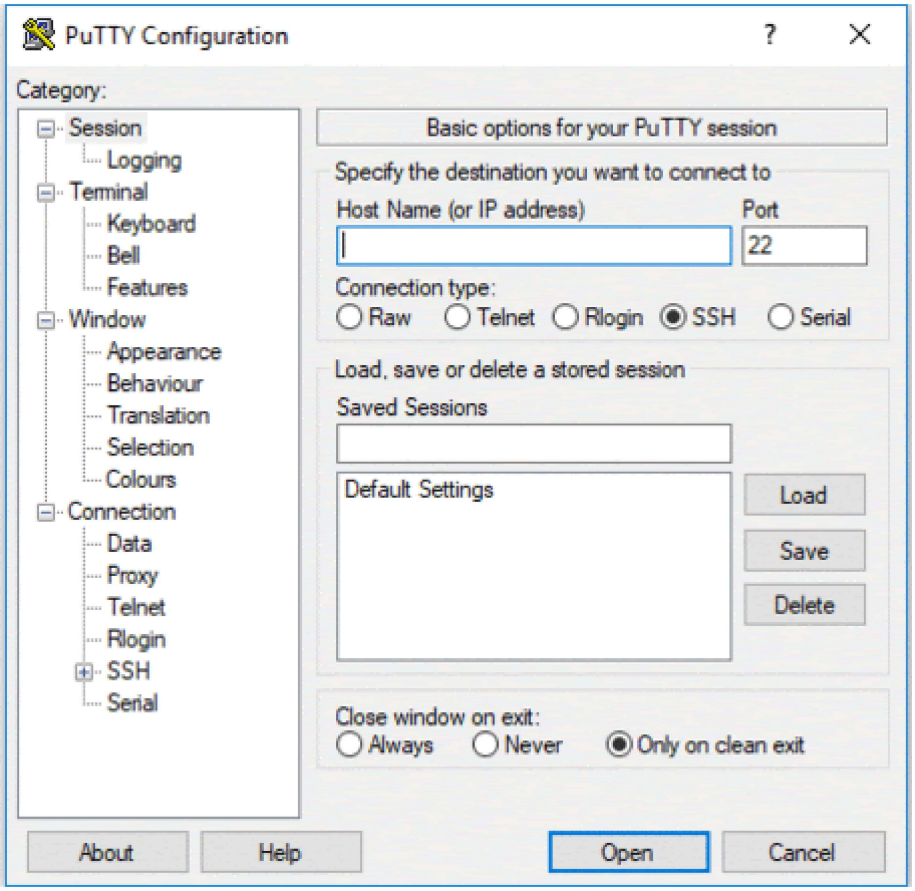
You can download the VM’s private key file in the PuTTY-compatible **PPK** format from the Qwiklabs Start Lab page. Click on **Download PPK**.

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.



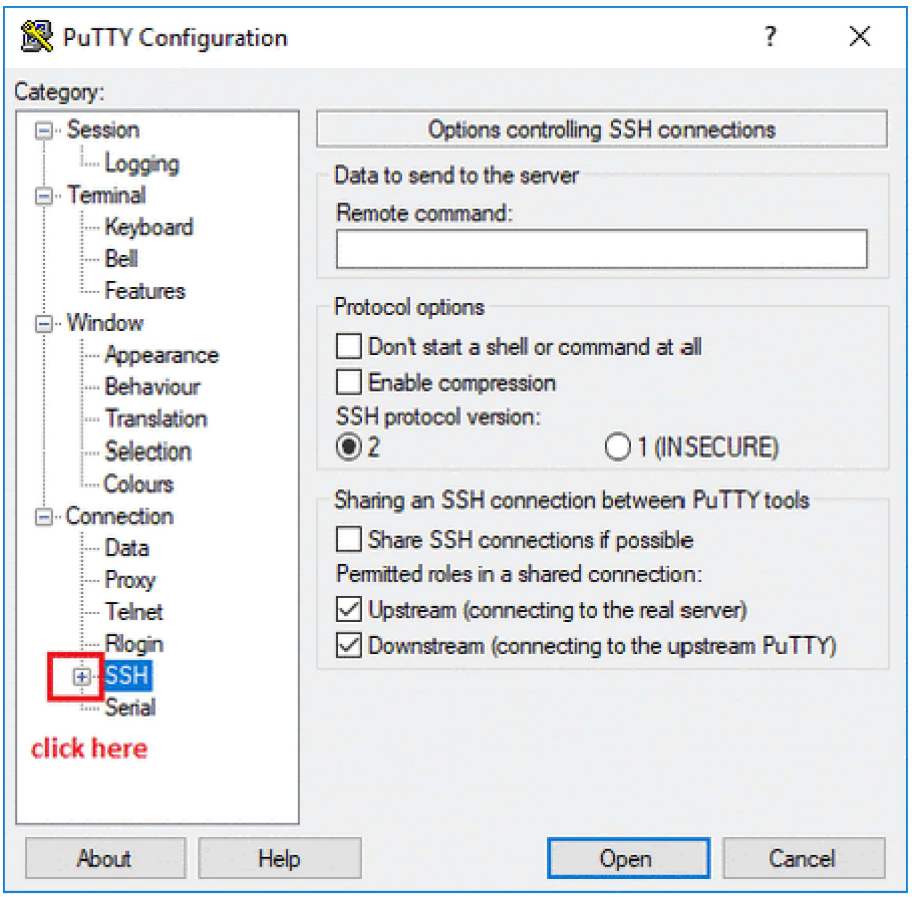
3. In the **Category** list, expand **SSH**.

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

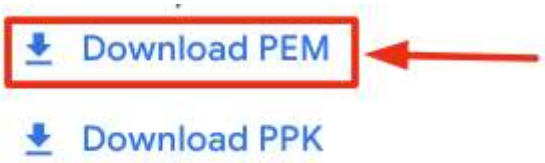
If PuTTY fails to connect to your Linux VM, verify that:

- 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 **terminal**.
2. Enter the following commands.

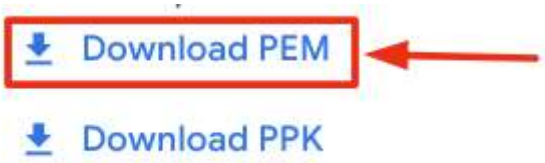
Note: Substitute the **path/filename for the PEM** file you downloaded, **username** and **External IP Address**.
You will most likely find the PEM file in **Downloads**. If you have not changed the download settings of your system, then the path of the PEM key will be **~/Downloads/qwikLABS-XXXXX.pem**

```
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!

Checking permissions

Before you can even begin changing the permission of a file or folder, you need to first check the permissions of the specific file/folder. To display ownership and permissions for a file, you can use **ls** with the **-l** flag and the name of the file you're interested in with the command `ls -l [FILENAME]`

There's a file named "**important_document**" on your machine in the "**/home/qwiklab/documents**" directory. You can change to this directory from your current one using this command:

```
cd ../qwiklab/documents
```

Check out its current permissions with this command, and take a look at the output below:

```
ls -l important_document
```

```
gcpstagingedu1t1082_student@linux-instance:/home/qwiklab/documents$ ls -l important_document
-rw------- 1 root root 16 Apr  9 02:13 important_document
```

You can see that it's owned by the "root" user, and that the owner has read and write (but not execute) permissions while everyone else has none at all.

Changing file permissions

Now, change the permissions of "important_document" (from the previous step) so that the owner has execute permissions on top of their current permissions. To do this, you'll use the **chmod** command, with the argument **700**. The two zeros keep everyone, but the owner, from having any permissions at all, and the seven grants all available permissions to the owner (including execute). Keep in mind that because the file is owned by "root" you'll need to use **sudo**:

```
sudo chmod 700 important_document
```

You can check the permissions from the below command. You'll now see that the execution permission has been added:

```
ls -l important_document
```

Click *Check my progress* to verify the objective. Modify permissions on important_document

Changing folder permissions

Now you'll do a similar process, this time on a directory. First, move up one directory:

```
cd ..
```

In this directory there's a folder called "secret_folder". View its current permissions using **ls**, this time with the **-ld** flag rather than **-l** because you're viewing a directory instead of a normal file:

```
ls -ld secret_folder/
```

As you can see, the owner of the file (the root user) has read and write permissions, and everyone else can read only.

The goals for the lab, related to this file, are below:

1. The owner should have all permissions.
2. The group should have only write permission.
3. People other than the owner and the group should have no permissions.

Head's up: When using **chmod** on a directory, files within that directory are not affected. While this isn't relevant to this specific lab, it's important to remember.

Previously, we used a numerical argument to set the permissions for a file. If you want to avoid figuring out the number that matches the permission levels, you can use an alternate syntax. To satisfy the first condition, you only need to add the execute permission to the owner, since they already have read and write permissions. To add execute to the owner's permission, you can use the command below. (Note that "u" stands for "user" and "x" stands for "execute".)

```
sudo chmod u+x secret_folder/
```

You can check the permissions again to see that the owner can now read, write, and execute:

```
ls -ld secret_folder/
```

```
gcpstagingeduit1082_student@linux-instance:/home/qwiklab$ ls -ld secret_folder/
drwxr--r-- 2 root root 4096 Apr  9 02:13 secret_folder/
gcpstagingeduit1082_student@linux-instance:/home/qwiklab$
```

Now you can fix the group's permission. They currently have read permission and don't have write permission, which you can fix with two similar commands. These can be done in either order; "g" stands for "group" (like "u" from before), and "w" and "r" stand for "write" and "read" respectively:

```
sudo chmod g+w secret_folder/
sudo chmod g-r secret_folder/
```

You can check the permissions again to see that the group now has only write permissions, and the owner has the same permissions as before:

```
ls -ld secret_folder/
```

```
gcpstagingeduit1082_student@linux-instance:/home/qwiklab$ ls -ld secret_folder/
drwx-w-r-- 2 root root 4096 Apr  9 02:13 secret_folder/
```

Finally, you can remove read permissions from everyone else using the command below ("o" stands for "other"):

```
sudo chmod o-r secret_folder/
```

You can see that all the criteria for this file are now met using **ls** again:

```
ls -ld secret_folder/
```

```
gcpstagingeduit1082_student@linux-instance:/home/qwiklab$ ls -ld secret_folder/
drwx-w---- 2 root root 4096 Apr  9 02:13 secret_folder/
```

Using **chmod** this way is easier to remember, but takes lots more commands. All the previous steps could also have been done using the numerical notation, like this:

```
sudo chmod 720 secret_folder/
```

Click *Check my progress* to verify the objective. Modify permissions on secret_folder

Changing owners

Now you'll change the owner of a file. In your current directory, there's a folder called "taco". Use **ls** to examine its permissions and see who the owner of the file is:

```
ls -ld taco/
```

```
gcpstaging7284_student@linux-instance:/home/qwiklab$ ls -ld taco/
drwxr-xr-x 2 root root 4096 Oct 25 12:17 taco/
```

You can see from this that the root user currently owns this file. There's another user account on the machine called "cook". Go ahead and make "cook" the owner of the file, using the **chown** command like this:

```
sudo chown cook /home/qwiklab/taco
```

Now you can use **ls** again to see that the owner of the file has been successfully changed:

```
ls -ld taco/
```

Click *Check my progress* to verify the objective. Change owner of Taco

More practices

There are a few more files present on your machine that you can practice on. First, move into the "documents" folder:

```
cd documents/
```

There's a file in this folder called "not_so_important_document". View its permissions to see its current state:

```
ls -l not_so_important_document
```

The owner can read and write, the group can read, and everybody else has no permissions at all. Now, use **chmod** to change the permissions so that these criteria are met:

1. The owner has all permissions.
2. The group has read and write permissions.
3. Everyone has read permissions.

To give the owner execution permissions, you can use the same command from earlier:

```
sudo chmod u+x not_so_important_document
```

Remember to use **ls** to double-check that everything you do behaves how you expect:

```
ls -l not_so_important_document
```

The group already has read permissions, so all you need to do is add write permissions:

```
sudo chmod g+w not_so_important_document
```

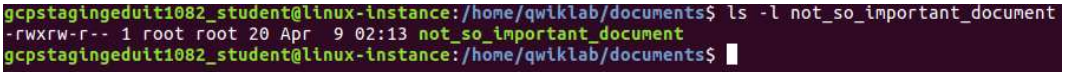
```
ls -l not_so_important_document
```

Finally, you need to give everyone else read permissions. You can use the "o+r" argument to add read permissions to people other than the owner or group, but you can also use "a+r". This adds read permission to everyone (owner, group, and other). Because the owner and the group already have read permissions, this will only change the permissions for everyone else, but the end result is the same:

```
sudo chmod a+r not_so_important_document
```

You should be able to view the permissions again and see that all criteria for this file have been met:

```
ls -l not_so_important_document
```



Again, you can accomplish the same result using a numerical argument to set the permissions, rather than incrementally changing them. Here's the command that meets all three criteria at once:

```
sudo chmod 764 not_so_important_document
```

Click *Check my progress* to verify the objective. Change permissions of not_so_important_document

Adding multiple permissions at once

Finally, you'll learn how to use the non-numeric argument to add multiple permissions at once. There's one more file in the current directory, named "public_document". First, view its current permissions:

```
ls -l public_document
```

For this file, you want everyone (owner, group, and anyone else) to have all permissions. You can add read, write, and execute permissions to everyone at once using this command:

```
sudo chmod a+rwx public_document
```

This should make the file as open as possible, where every user has every permission:

```
ls -l public_document
```

```
gcpstagingeduit1082_student@linux-instance:/home/qwiklab/documents$ ls -l public_document
-rwxrwxrwx 1 root root 14 Apr  9 02:13 public_document
```

Using the numeric argument form of **chmod**, this same result could be accomplished with this command:

```
sudo chmod 777 public_document
```

Click *Check my progress* to verify the objective. Change owner of public_document

Conclusion

Congrats! You've successfully used **chmod** on both directories and normal files, in multiple formats. You can directly set a file's permissions numerically, or add and remove specific permissions one at a time. You've also successfully used **chown** to change the owner of a file. Really great work!

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.