

# Hands on with Hashing

1 hour 1 Credit

[Rate Lab](#)

## Introduction

In this lab, you'll have hands-on practice demonstrating hashing and hash verification using `md5sum` and `shasum` tools.

*Md5sum* is a hashing program that calculates and verifies 128-bit MD5 hashes. As with all hashing algorithms, theoretically, there's an unlimited number of files that will have any given MD5 hash. `Md5sum` is used to verify the integrity of files.

Similarly, *shasum* is an encryption program that calculates and verifies SHA hashes. It's also commonly used to verify the integrity of files.

In this lab, you'll see that almost any change to a file will cause its MD5 hash or SHA hashes to change.

### What you'll do

- **Compute:** You'll create a text file and generate hashes using the `md5sum` and `shasum` tools.
- **Inspect:** After you generate the hash digests, you'll inspect the resulting files.
- **Verify:** You'll verify the hash using the `md5sum` and `shasum` tools.
- **Modify:** You'll modify the text file and compare these results to the original hash to observe how the digest changes and how the hash verification process fails.

### 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 blue input field containing a blurred IP address. To the right of the field is a copy icon. Below this is the label "username" above another light blue input field containing a blurred username, also with a copy icon to its right. At the bottom of the panel are two buttons: "Download PEM" and "Download PPK", both in blue text with a small download icon to their left.

## 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

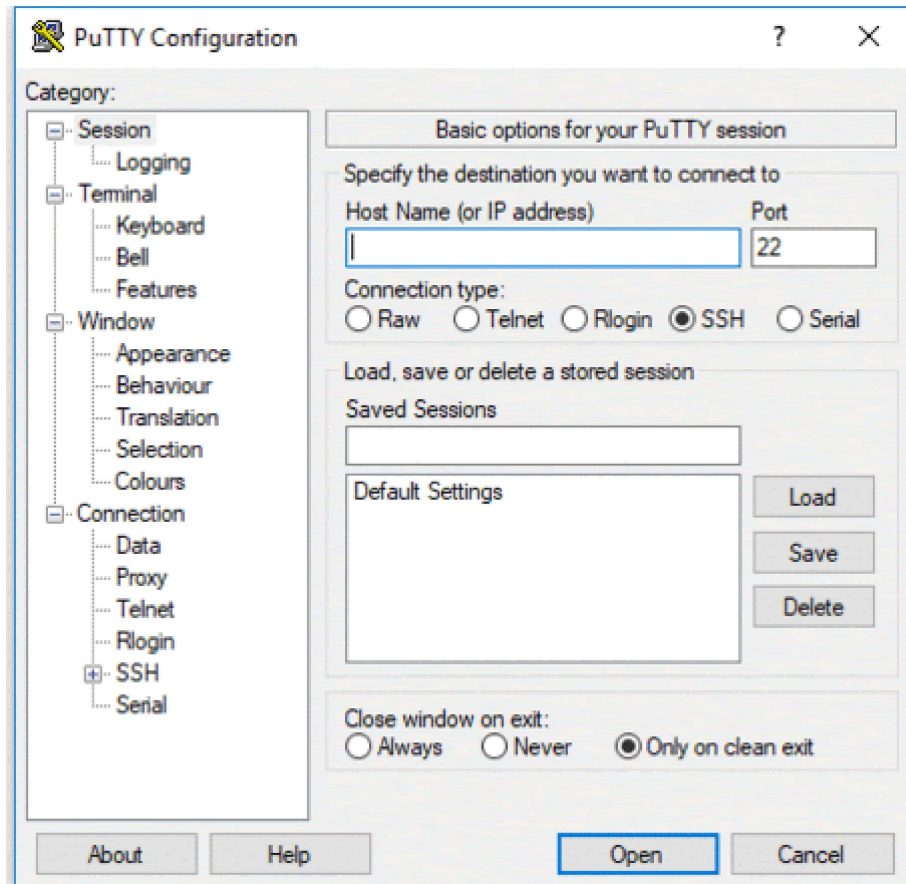
↓ Download PPK



**Connect to your VM using SSH and PuTTY**

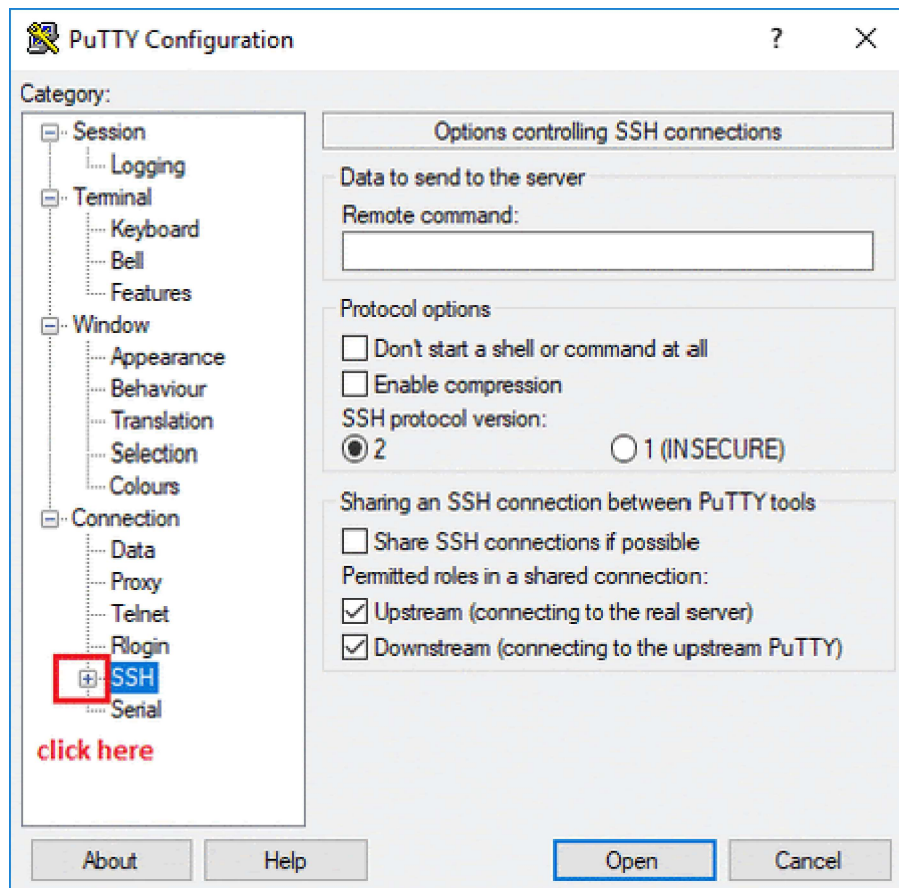
1. You can download Putty from [here](#)
2. In the **Host Name (or IP address)** box, enter  
username@external\_ip\_address.

**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

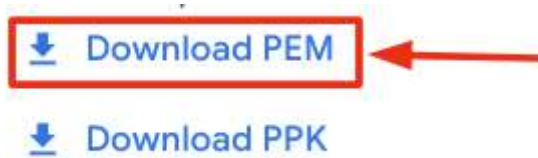
If PuTTY fails to connect to your Linux VM, verify that:

- 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
```

```
gcpstagingeduit1370_student@35.239.106.192:~$ ssh -i ~/Downloads/qwikLABS-L923-42090.pem gcpstagingeduit1370_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:vrz8b4aYUtrufH0A6wZn6Ozy1oqqPEfh931olvxITn8.
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.

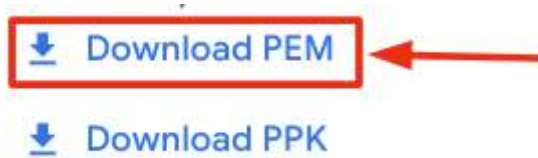
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
gcpstagingeduit1370_student@linux-instance:~$
```

## 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.

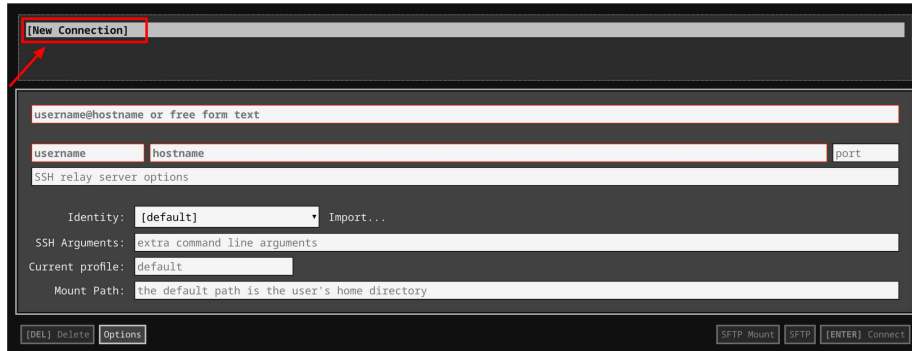
**Download your VM's private key file.**

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

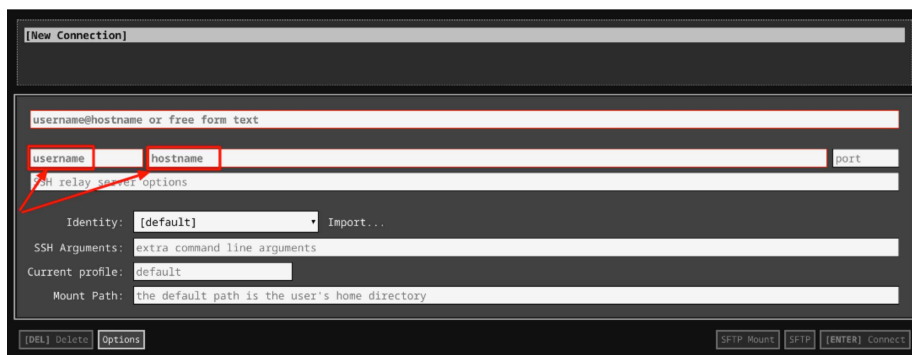


## Connect to your VM

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



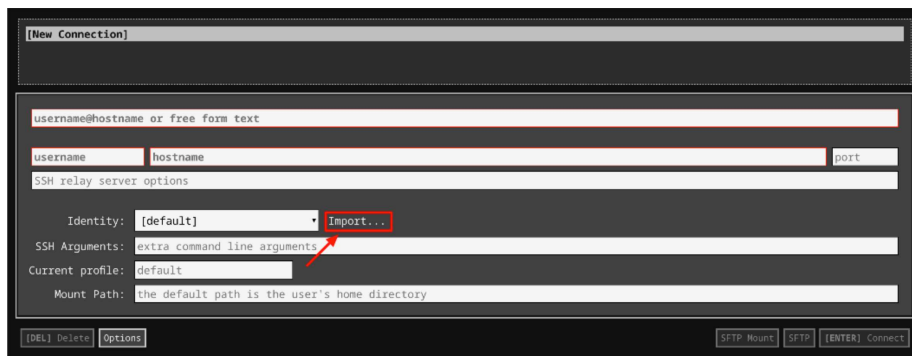
3. In the **username** section, enter the username given in the Connection Details Panel of the lab. And for the **hostname** section, enter the external IP of your VM instance that is mentioned in the Connection Details Panel of the lab.



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!

## MD5

Let's kick things off by creating a text file containing some data. Feel free to substitute your own text data, if you want. This command creates a text file called "file.txt" with a single line of basic text in it:

```
echo 'This is some text in a file, just so we have some data' > file.txt
```

Click *Check my progress* to verify the objective. Create Text File

You'll now generate the MD5 sum for the file and store it. To generate the sum for your new file, enter this md5sum command:

```
md5sum file.txt > file.txt.md5
```

This creates the MD5 hash, and saves it to a new file. You can take a look at the hash by printing its contents to the screen, using this command:

```
cat file.txt.md5
```

This should print the hash to the terminal, which should look something like this:

```
gcpstaging3643_student@linux-instance:~$ cat file.txt.md5
c7a8ef893898f9a6b380eb4ec1e87113  file.txt
gcpstaging3643_student@linux-instance:~$
```

More importantly, you can also verify that the hash is correct, and that the original file hasn't been tampered with since the sum was made. To do this, enter this command and see the following output, which indicates that the hash is valid:

```
md5sum -c file.txt.md5
```

```
gcpstaging3643_student@linux-instance:~$ md5sum -c file.txt.md5
file.txt: OK
gcpstaging3643_student@linux-instance:~$
```



Click *Check my progress* to verify the objective. md5sum

## Verifying an invalid file

Next, we'll demonstrate the security of this process by showing how even a single-character change to the file results in a different hash. First, you'll create a copy of the text file, and insert a single space at the end of the file. Feel free to use any text-editor that you'd like. Head's up that we've included instructions for making this change in Nano. To make a copy of the file, enter this command:

```
cp file.txt badfile.txt
```

Then generate a new md5sum for the new file:

```
md5sum badfile.txt > badfile.txt.md5
```

Note that the resulting hash is **identical** to the hash for our original file.txt despite the filenames being different. This is because hashing only looks at the data, not the metadata of the file.

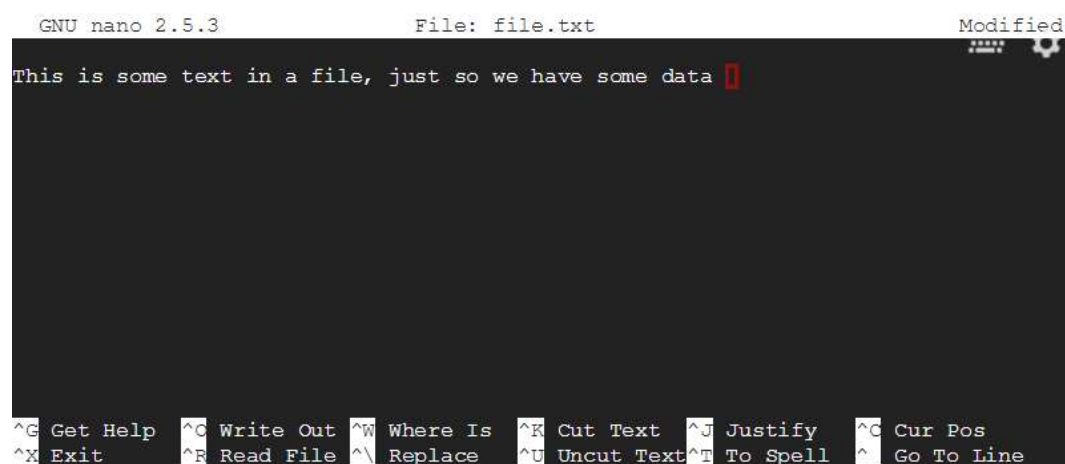
```
cat badfile.txt.md5
```

```
cat file.txt.md5
```

To open the text file in Nano, enter this command:

```
nano badfile.txt
```

This will open the file in the text editor. To add a space to the end of the file, use the arrow keys (not the mouse!) to move the cursor to the end of the line of text. Then, press the spacebar to add a space character to the end of the file. Your screen should look like this image:



To save the file, press **ctrl+X**. You should see this message:



```
GNU nano 2.5.3 File: file.txt Modified
This is some text in a file, just so we have some data

Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ?
Y Yes
N No ^C Cancel
```

Confirm by typing **Y** for **yes**, then press **Enter** to confirm.

This will take you back to the normal terminal screen. Now that you've made a very minor change to the file, try verifying the hash again. It should fail verification this time, showing that any change at all will result in a different hash. Try to verify it by entering this command again:

```
md5sum -c badfile.txt.md5
```

You should see a message that shows that the verification wasn't successful:

```
gcpstagingeduit259_student@linux-instance:~$ md5sum -c badfile.txt.md5
badfile.txt: FAILED
md5sum: WARNING: 1 computed checksum did NOT match
gcpstagingeduit259_student@linux-instance:~$
```

Click *Check my progress* to verify the objective. md5sum failure

To see how different the hash of the edited file is, generate a new hash and inspect it:

```
md5sum badfile.txt > new.badfile.txt.md5
```

```
cat new.badfile.txt.md5
```

Check out how it's different from our previously generated hash:

```
gcpstagingeduit259_student@linux-instance:~$ md5sum badfile.txt > new
gcpstagingeduit259_student@linux-instance:~$ cat new.badfile.txt.md5
3d121c97162462ce6e3448636ba13359 badfile.txt
gcpstagingeduit259_student@linux-instance:~$
```

For reference, here are the contents of the original sum:

```
gcpstaging3643_student@linux-instance:~$ cat file.txt.md5
c7a8ef893898f9a6b380eb4ec1e87113 file.txt
gcpstaging3643_student@linux-instance:~$
```

Click *Check my progress* to verify the objective. Recompute MD5 Sum

## SHA

Let's do the same steps, but for SHA1 and SHA256 hashes using the shasum tool. Functionally, the two work in very similar ways, and their purpose is the same. But SHA1 and SHA256 offer stronger security than MD5, and SHA256 is more secure than SHA1. This means that it's easier for a malicious third party to attack a system using MD5 than one using SHA1. And because SHA256 is the strongest of the three, it's currently widely used.

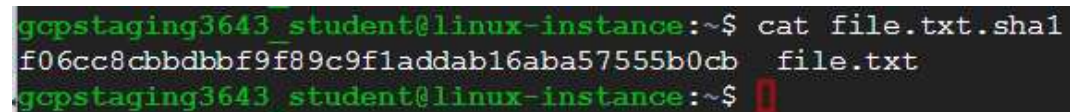
## SHA1

To create the SHA1 sum and save it to a file, use this command:

```
shasum file.txt > file.txt.sha1
```

View it by printing it to the screen, like you've done before:

```
cat file.txt.sha1
```

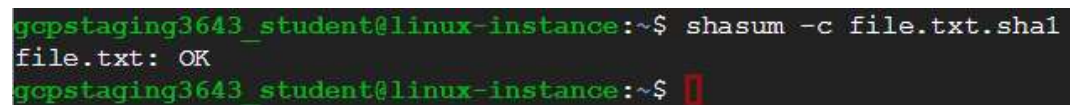


```
gcpstaging3643_student@linux-instance:~$ cat file.txt.sha1
f06cc8cbbdbbf9f89c9f1addab16aba57555b0cb file.txt
gcpstaging3643_student@linux-instance:~$
```

Now, verify the hash using the command below. (Like before, this would fail if the original file had been changed.)

```
shasum -c file.txt.sha1
```

You should see the following output, indicating that the verification was a success:



```
gcpstaging3643_student@linux-instance:~$ shasum -c file.txt.sha1
file.txt: OK
gcpstaging3643_student@linux-instance:~$
```

Click *Check my progress* to verify the objective. SHA1 Hash

## SHA256

The same tool can be used to create a SHA256 sum. The "-a" flag specifies the algorithm to use, and defaults to SHA1 if nothing is specified. To generate the SHA256 sum of the file, use this command:

```
shasum -a 256 file.txt > file.txt.sha256
```

You can output the contents of this file, the same as before:

```
cat file.txt.sha256
```

SHA256's increased security comes from it creating a longer hash that's harder to guess. You can see that the contents of the file here are much longer than the SHA1 file:

```
gcpstaging3643_student@linux-instance:~$ cat file.txt.sha256
911166859be8d9fa946ed2badd2bfc7a504466384cee3e44b0a4f63db20030eb  file.txt
gcpstaging3643_student@linux-instance:~$
```

Finally, to verify the SHA256 sum, you can use the same command as before:

```
shasum -c file.txt.sha256
```

Click *Check my progress* to verify the objective. SHA256 Hash

## Conclusion

Congrats! You've successfully created and verified hashes using three of the most common hashing algorithms: MD5, SHA1, and SHA256. You've also demonstrated the security of these hashes by showing that even very small changes to the file result in a new hash and cause the verification to fail.

## 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.