Unknown Title

Introduction to Github

Introduction

In this lab, you'll practice the basics of interacting with GitHub. You'll practice setting up an account, logging in, creating a repository, making changes on the local machine, and pushing changes back to the remote repository. We use these git operations to share changes from the remote repository to the local repository and vice-versa.

What you'll do

- Create a Github account
- Create a git repository
- Git clone to create a local copy on your local machine
- Add a file to this repository
- Create snapshot/snapshots of the local repository
- Push the snapshots to the main branch

You'll have 90 minutes to complete this lab.
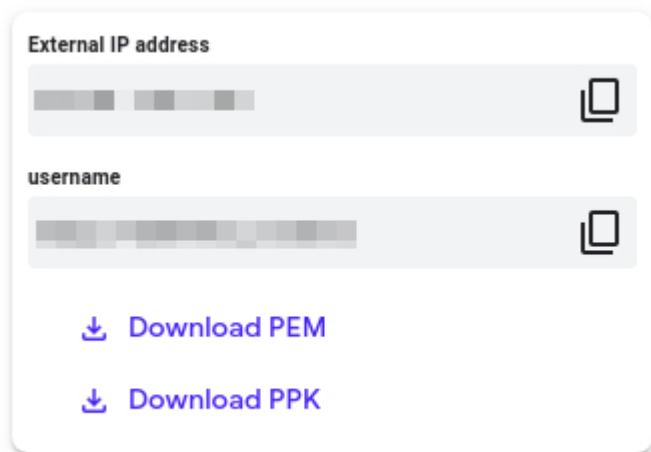
Start the lab

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).

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**.
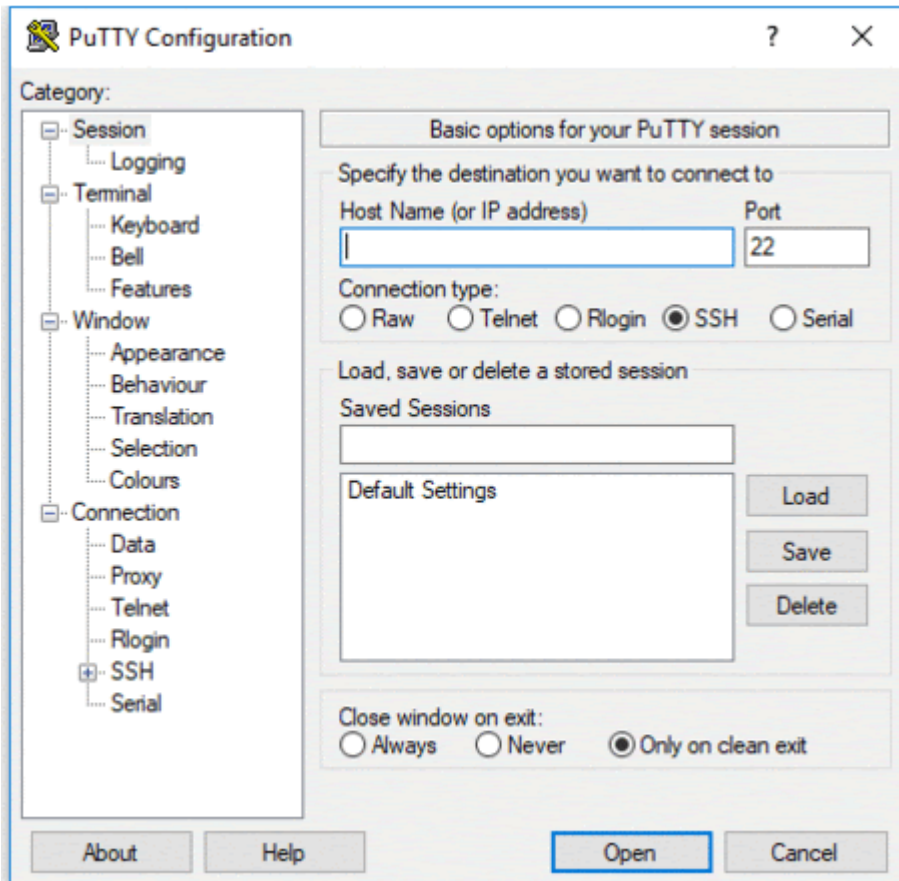

Connect to your VM using SSH and PuTTY

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.
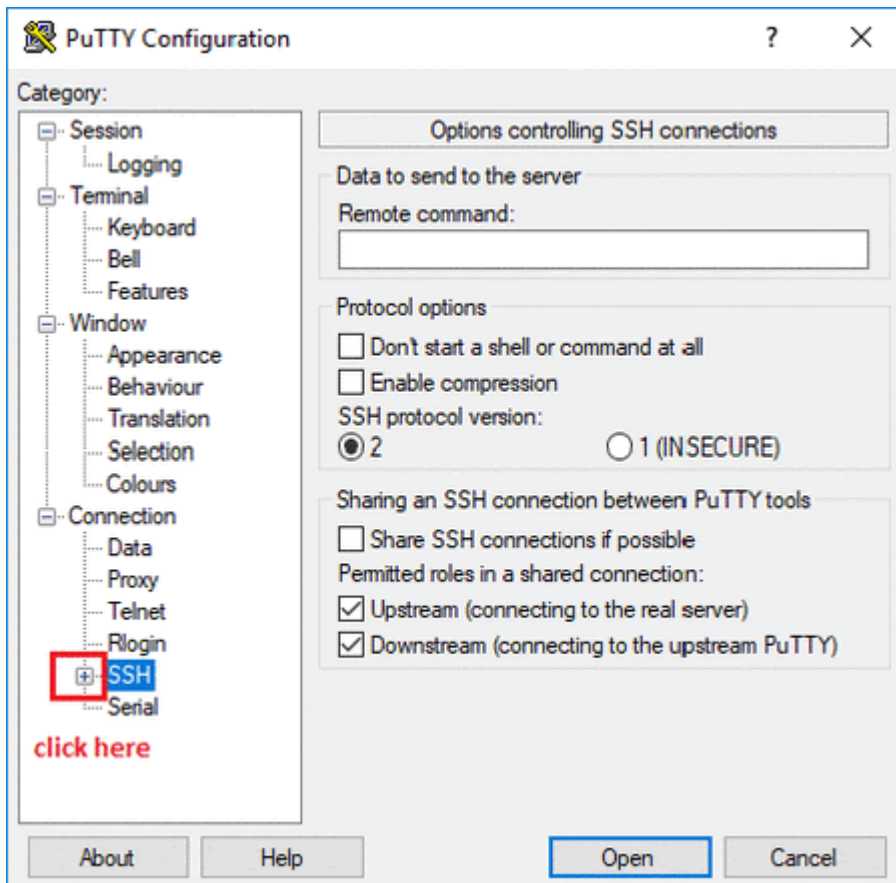
7. Click **Yes** when prompted to allow a first connection to this remote SSH server. Because you are using a key pair for authentication, you will not be prompted for a password.

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!

Create a git repository

To create a git repository, you need to have a Github account. Follow the steps below to create a github account and a git repository:

- Open Github. If you don't already have a Github account, create one by entering a username, email, and password. If you already have a Github account proceed to the next step.
- Log in to your account from the Github login page.
- Click the **+** sign in the top-right corner of the page and click then on **New repository**.
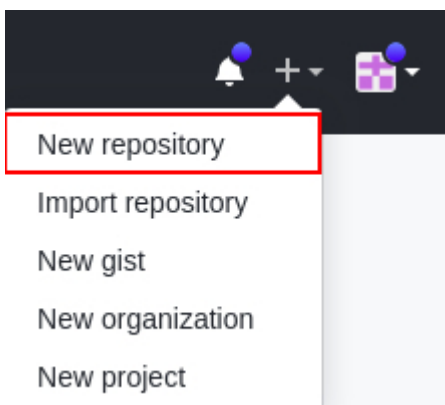
- Enter your repository name in the field **Repository name** and add a project description in the **Description** field.

- You can either select **public** or **private** to restrict repository accessibility. If **public**, anyone can see the repository but you still choose who can commit to it. If **private**, you choose who can see and commit to the repository.

- Check the option **Initialize this repository with a README** to initialize the repository with a README file. Leave all the other values to their default.

- Click the **Create repository** button.

Git operations

You now need to create a local copy of this remote repository on your machine. We'll do this by cloning the repository. The syntax for this is:

git clone [URL]

For the URL, you can either choose an **SSH** or an **HTTPS** link as a URL. We will use HTTPS to clone the Git repository. Click on **Clone or download** and select HTTPS. Copy the **HTTPS** link by clicking on the Copy button beside the link.
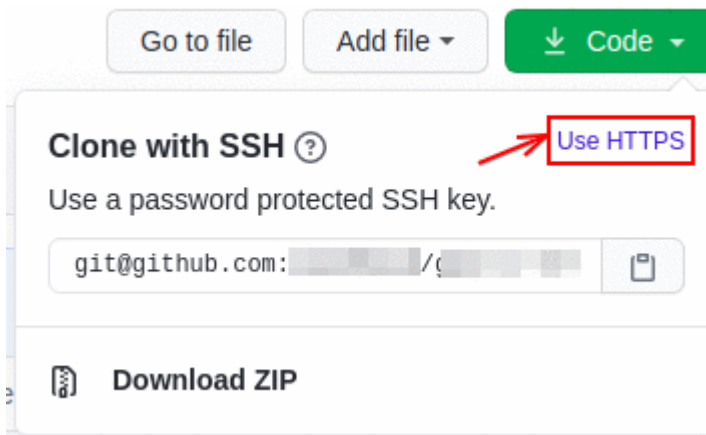
Next, go to your **linux-instance** terminal and replace [URL] from the above syntax with the link you copied. The command should now look similar to:

git clone https://github.com/[username]/[git-repo].git

Here, **username** is the Git username and **git-repo** is the name of the remote repository you created.

This creates a directory with the same name as your repository, initializes a **.git** directory inside it, pulls down all the data for that repository, and creates a working copy of the latest version.

You can now list the files using the **ls** command and find your new repository. Move into your repository using **cd** command. There, you'll see the project files, which are ready to be worked on or used.

cd directory_name

Replace the **directory_name** with your repository's name that you just initialized.

If you want to clone the repository into another directory of your choice, you can do that by passing the name of the directory. This automatically creates a new directory with the specified name and initializes the repository inside it.

Syntax:

git clone [URL] directory_name

Configure Git

Git uses a username to associate commits with an identity. It does this by using the **git config** command. Set Git username with the following command:

git config --global user.name "Name"

Replace **Name** with your name. Any future commits you push to GitHub from the command line will now be represented by this name. You can use **git config** to even change the name associated with your Git commits. This will only affect future commits and won't change the name used for past commits.

Let's set your email address to associate them with your Git commits.

git config --global user.email "user@example.com"

Replace **user@example.com** with your email-id. Any future commits you now push to GitHub will be associated with this email address. You can also use **git config** to even change the user email associated with your Git commits.

Edit the file and add it to the repository

Now, edit the README file by using nano editor:

nano README.md

Add any text within the file, or you can use the following text:

I am editing the README file. Adding some more details about the project description.

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

We can check the status using the following command:

git status

The git status command shows the different states of files in your working directory and staging area, like files that are modified but unstaged and files that are staged but not yet committed.

You can now see that the README.md file shows that it's been modified.

```
student-01-80cc852cdf10@linux-instance:~/my-git-repo$ git status
On branch main
Your branch is up-to-date with 'origin/main'.
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
```

Now, let's add the file to the staging area using the following command:

git add README.md

Use the **git add** command to add content from the working directory into the staging area for the next commit. When the git commit command is run, it looks at this staging area. So you can use git add to craft what you'd like your next commit snapshot to look like. To check the files in staging area use `git status`.

```
student-01-80cc852cdf10@linux-instance:~/my-git-repo$ git status
On branch main
Your branch is up-to-date with 'origin/main'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   README.md
```

Let's now commit the changes. A Git commit is like "saving" your work.

Commit the changes using the following command:

git commit

This now opens an editor that asks you to type a commit message. Every commit has an associated commit message. A commit message is a log message from the user describing the changes.

Enter the commit message of your choice or you can use the following text:

I am editing the README file.

Once you've entered the commit message, save it by clicking Ctrl-o and the Enter key. To exit click Ctrl-x.

The **git commit** command captures a snapshot of the project's currently staged changes. It stores the current contents of the index in a new commit along with the commit message.

You've successfully committed your file!

Now, push the committed changes from your local repository to a remote repository on the **main** branch by using:

git push origin main

Next, enter your Github username/email ID and password to prompt the associated remote repository to push the changes.

Note: If you enabled two-factor authentication in your Github account you won't be able to push via HTTPS using your account's password. Instead you need to generate a personal access token. This can be done in the application settings of your Github account. Using this token as your password should allow you to push to your remote repository via HTTPS. Use your username as usual. For more help to generate a personal access token, click here.

Click *Check my progress* to verify the objective. Edit the file and add it to repository

You can check the changes made to the local README.md file on the remote repository on Github. You can see the last time when the README.md file was added/updated.

You can also see the commit ID just above the list of files in the repository. Click on the Commit ID to get more details related to the commit.
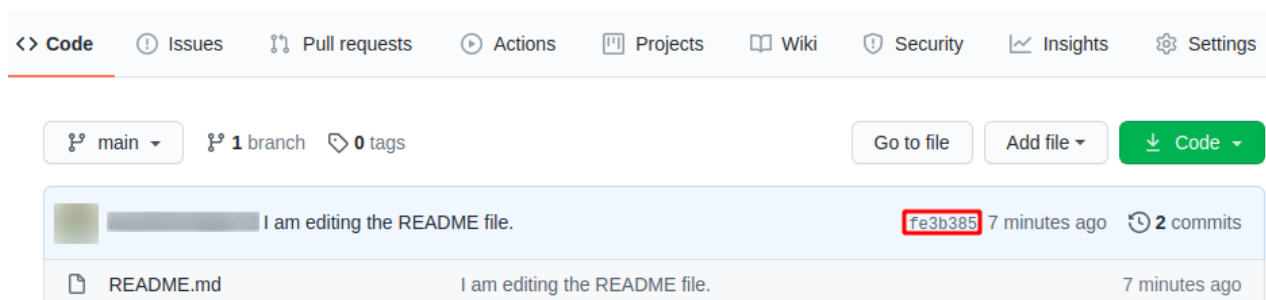
Create a new file and commit it to the repository

You now need to create a new file **example.py** on the local git repository in the working directory. To do this, use the following command:

nano example.py

Add the following Python script to the **example.py** file:

def git_opeation(): print("I am adding example.py file to the remote repository.") git_opeation()

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

Now, repeat the same procedure by adding a file to the staging area for next commit:

git add example.py

Commit the changes:

git commit

Enter a commit message and save it by pressing Ctrl-o and the Enter key. To exit click Ctrl-x.

We will push these changes later in the lab.

Click *Check my progress* to verify the objective. Create a new file and commit it to the repository

Add an empty file to the repository through web UI

Now, let's create an empty file on the remote repository using the Github website.

1. Go to your repository on the Github website and click on the **Add file** button, then click on **Create new file**. This will open a new page.
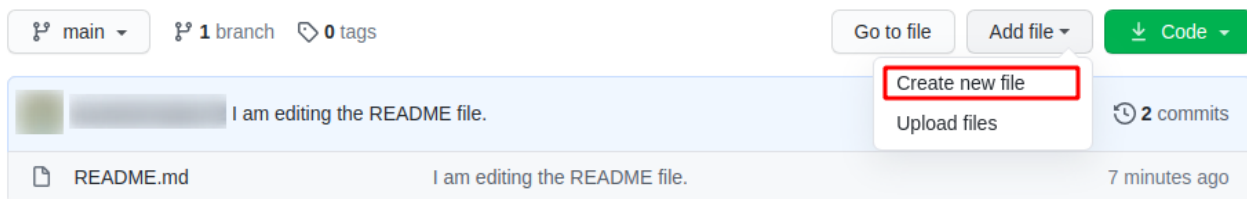
2. Enter the file's name in the box beside your repository's name. Leave the contents of the file empty.
3. Scroll down and enter a commit message in the first box under **Commit new file** section.
4. Leave the rest on its default value and click the **Commit new file** button.

You've successfully committed a new file through the website.

Now, let's push the changes made on the local repository that weren't pushed. Switch back to your terminal and enter the following command:

git push origin main

Output:

```
student-01-80cc852cdf10@linux-instance:~/my-git-repo$ git push origin main
Username for 'https://github.com':
Password for 'https://            @github.com':
To https://github.com/                    /my-git-repo.git
 ! [rejected]        main -> main (fetch first)
error: failed to push some refs to 'https://github.com/                    /my-git-repo.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

The last command throws an **error**. This is because the files added or change done on a remote repository (the Github website) isn't present yet on your local repository, but we're trying to push something from the local repository to the remote repository. To push changes from the local repository, we need to first update the local repository from the remote repository.

Let's now pull the current snapshot/commit in the remote repository to the local repository:

git pull origin main

This opens an editor that asks you to enter a commit message for the merge operation (remote repository to local repository).

You can simply accept the default message or type your own message. To continue, save the file by pressing Ctrl-o, Enter key, and Ctrl-x.

The git pull command is used to fetch and download content from a remote repository and update the local repository to match that content.

Output:

```
student-01-80cc852cdf10@linux-instance:~/my-git-repo$ git pull origin main
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/                    /my-git-repo
 * branch            main       -> FETCH_HEAD
   fe3b385..9afc635  main       -> origin/main
Merge made by the 'recursive' strategy.
 data.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 data.txt
```

Now try pushing the changes again.

git push origin main

Output:

```
student-01-80cc852cdf10@linux-instance:~/my-git-repo$ git push origin main
Username for 'https://github.com':
Password for 'https://                    @github.com':
Counting objects: 5, done.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 704 bytes | 0 bytes/s, done.
Total 5 (delta 0), reused 0 (delta 0)
To https://github.com/                    /my-git-repo.git
   9afc635..79c9314  main -> main
```

This shows that your local repository is now up-to-date with your remote repository and you successfully pushed the changes to the remote repository.

Click *Check my progress* to verify the objective. Add an empty file to the repository through web UI

Congratulations!

Great job! You've successfully set up a Github account, logged in, created a repository, made changes on the local machine, and pushed changes back to the remote repository. You've learned how to share local changes from a local to remote repository and vice-versa. These skills will help you as an IT specialist to work with a remote user on the same repository.

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.