**Setting up and using the KitClient Linux Environment**

7. Running and interacting with the KitClient requires that you install (or have already installed) some dependencies. Complete the following steps:

a. Download, install and run *Docker Desktop*, which will be used to run the KitClient:

* + <https://www.docker.com/>

b. Install the *TigerVNC Viewer*, which you will use to interact with the KitClient:

* + Windows: <https://sourceforge.net/projects/tigervnc/files/stable/1.12.0/vncviewer64-1.12.0.exe/download>
  + Mac: <https://sourceforge.net/projects/tigervnc/files/stable/1.12.0/TigerVNC-1.12.0.dmg/download>

8. Now let's create and run the KitClient and connect to it.

a. Open a Terminal Window or Command Prompt on your computer and then copy and paste the command below to run the KitClient:

docker create --name KitClient --mount source=gitkitvol,target=/home/student -p 6901:6901 -p 5901:5901 registry.gitlab.com/hfossedu/kits/kitclient

It may take a little while for the KitClient to download to your machine. Fortunately, you will only have to use this command once.

When the download completes, the final line of output from the above command should be a long string of letters and numbers. Copy and paste that string here. If you do not see that string, check the output for error messages and try the command above again.

b. Now that you have the KitClient image you can start it running using the following command:

docker start KitClient

While this command doesn’t appear to do much, it has in fact started a server on your machine that contains the KitClient. If this worked successfully, the command will simply output KitClient.

There is nothing required here. But you need to be sure that the docker start command has outputted KitClient to indicated that the KitClient is running. If not, revisit #7 and #8 to ensure that you have everything installed correctly.

c. With the KitClient running you can connect to it using the TigerVNC viewer.

* + Launch the TigerVNC viewer.
  + Put the address: localhost:5901 into the “VNC server” text field.
  + Click the “Connect” button.

When you click connect a window should open with the KitClient’s Linux environment running inside. Take a screenshot of that desktop and paste it here.

Note: If you have been unable to install the Tiger VNC viewer you can also access the KitClient using a web browser by visiting the URL: http://localhost:6901

If you access the KitClient via your browser:

* + Open the “noVNC” menu (the little tab on the left) and then use the settings (the the gear) to set the “Scaling Mode” to “Remote Resizing.” This will resize the KitClient desktop when you resize your browser window.
  + Copy and pasting between your OS and the KitClient requires that you use the clipboard in the “noVNC” menu. This is a little inconvenient, but it is functional.

9. At some point you will complete this assignment or want to take a break and do other things. When you do so, you should stop the KitClient and then restart it when you are ready to work again.

a. Stop the KitClient by opening a Terminal Window or a command prompt on your computer (not in the KitClient) use the command:

docker stop KitClient

Take a screenshot of your command and its output and paste it here.

b. Use the steps in #8b and #8c to restart the KitClient and reconnect to it with TigerVNC (or noVNC in your browser). You should notice that it starts much more quickly this time.

Nothing is required here. But be sure you have the KitClient running and open in a VNC Viewer window or in a browser. **The remainder of this activity (and the next several) will assume that you are working within the KitClient.**

**Cloning Your FarmData2 Fork:**

In order to work with the files in your forked repository you will need to create a *clone* of it onto your local machine as shown in Figure 2.

Figure 1 - Cloning your Origin

10. To make a clone of your fork you will first need a *Git client*. There are lots of different Git clients and most Integrated Development Environments (IDEs) will provide a graphical interface to Git. For these exercises however, you will use the command line interface (CLI) to Git. This is the most general way to interact with Git and will provide a foundation for using any of the graphical interfaces that you might encounter.

Open a terminal window (inside the KitClient) and enter the command git. Use the output that you see to determine how to display the version of git that is installed.

Paste a screenshot of the command you used to display the git version and the output that it generated here.

11. Recall that Git repositories maintain a complete history of all changes. To do this, git needs to know a little bit of information about you. This question will have you do a little configuration to provide git with the information that it needs.

Enter the following commands one by one on the command line replacing the <…> with the appropriate values:

git config --global user.name <your GitHub username>

git config --global user.email <your e-mail>

Now run the following command:

git config --global --list

Give a screenshot of the commands you used and their output here. Note: You should see the information that you entered in the above commands. If not, try those commands again.

12. Now with git configured you can use it to create a local clone as was illustrated in Figure 2.

a. On the GitHub page for your fork, find the “Code” ( Graphical user interface

Description automatically generated with low confidence ) button. When you click this button, you will be shown a URL that can be used to clone the repository. That URL should include your GitHub username. If it does not, then you are not on the page for your fork (Go to your GitHub profile and find your fork). Also that URL should end with .git. If it does not, then you have not copied the correct URL (Check under the Code button again). Copy and paste the URL you found here.

b. Open a Terminal (in the KitClient) and use the URL from part a in the following command:

git clone <URL>

Give a screenshot of the command you used and its output here.

c. In the output above from part b, there should be a line that starts “Cloning into”. This line tells you the name of the directory into which the repository has been cloned. If you do not see this line, check the output from part b for errors and try again.

What directory was your repository cloned into?

d. Examine the remaining output from your command in part a. What is the “Kit-tty”?

Be sure to keep your eyes open for messages from the Kit-tty, they will help you to stay on track with these activities and may save you a lot of time by preventing you from going down an incorrect path.

e. What does the final “IMPORTANT” line of output in the terminal tell you to do?

There is a technical reason for having to close your current terminal and reopen a new one at this point. But the details for why are unimportant, just please be sure that you do so now.

f. Examine the files that are in the directory you found in part c and compare them to the files that are in your fork on GitHub. Remember that you can use ls -a to see all files including hidden files. Is your clone an exact copy of your fork? If you notice any differences describe them here.

13. In Figure 2, there is a dotted line from your clone to your fork indicating that your clone knows about the remote repository from which it was cloned. The Git CLI will allow us to see and confirm this connection.

a. Using a Terminal, ensure that your working directory is the directory containing your cloned repository and use the following command:

git remote -v

This command lists all of the remote repositories that your clone knows about. In your case there should be two lines of output (one for “push” and one for “fetch”).

Give a screenshot of your command and its output here.

b. In class we saw that the name *origin* is used to refer to your fork on GitHub. You should also see this name in the output from part a. This indicates that your clone knows the URL of your origin. How is the information about the *origin* remote represented in Figure 2?

c. In class we discussed a *workflow* for making contributions to FOSS projects. For that workflow to operate it is essential that your clone (i.e. local copy) know about the origin repository from which it was cloned. Which of the four operations discussed in class would require this information?