**Setting up and using the KitClient Visual Studio Code Environment**

7. Running and interacting with the VSCode 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. Download and install, which will be used for version control:

* + https://git-scm.com/downloads

c. Download and install Visual Studio Code, which will be used for your IDE:

* + - * + https://code.visualstudio.com/download

8. Start Visual Studio Code and install the *DevContainers* extension:

a. <https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.remote-containers>

**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

9. 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 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.

10. Recall that Git repositories maintain a complete history of all changes, including who made each change. To do this, git needs to know a little bit of information about you. This question will have you do some configuration to provide git with the information that it needs to attribute changes that you make to you.

Git identifies the author of each change using two pieces of information: A *name* and an *e-mail*. For name, you might enter your full name, or a nickname, or your GitHub username. Just keep in mind that whatever you use will be associated with all of the changes you make. For e-mail, you can again use any valid e-mail associated with you.

Enter the following commands one by one on the command line replacing *<name>* and

<e-mail> with appropriate values:

git config --global user.name "*<name>*"

git config --global user.email "*<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.

11. 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. In your Terminal, 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?

12. Open Visual Studio Code (VSCode)

a. Inside VSCode, choose File>Open Folder and select the GitKit-FarmData2 folder. Click Open.

b. VSCode will give you the option to reopen the folder in a container. Click the **Reopen in Container** button.

c. At this point VSCode may take a while building the dev container.

d. Once it finishes, it will open a terminal. Examine the output in the terminal. 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 “Done” 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 URL of 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 workflow steps would require this information?