**1 – FOSS Community and Collaboration**

**Activities**

**Name:**

Top of FormThis topic has focused on FOSS communities. It looked at what they are, how they are organized, the principles on which they operate and the roles that the members take on. We also began to look at some of the tools and processes that these communities use to support those principles and coordinate their work. In particular we saw how git and GitHub work together to allow FOSS communities to share their work and to collaborate. We saw how forks and clones make it possible for a diverse and distributed group of contributors to work asynchronously and to contribute useful changes back to the upstream. These activities will strengthen your understanding of FOSS communities and use a copy of the FarmData2 project repository to provide you with hands-on experience using use git and GitHub.

**FOSS Communities:**

1. In class we discussed some of the key principles that guide FOSS communities. These were *shared values*, *collaboration*, *transparency*, *inclusivity*, *meritocracy* and *release early & often*. Consider each of the following statements about the operation of a FOSS community and label it with the principle (or principles if multiple apply) with which it most closely aligns.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | **Statement** | **FOSS Community Principle(s)** |  |
|  | By building on each other’s work the community can solve problems that no one could solve alone. |  |  |
|  | Decisions and the rationale for them are available to the community. |  |  |
|  | The mission and goals of the community are more important than individual agendas. |  |  |
|  | The best ideas should win, regardless of where they come from. |  |  |
|  | Incorporating new changes and features quickly generates feedback and leads to rapid improvement. |  |  |
|  | Community members enhance and extend what others contribute in unanticipated ways. |  |  |
|  | Decision makers continually seek diverse perspectives. |  |  |
|  | All community members have access to the information necessary to do their best work. |  |  |
|  |  |  |  |

2. In class we also discussed some of the roles that tend to exist in FOSS communities. These included *Users*, *Requestors*, *Contributors*, *Maintainers* and *Leaders*. Consider each of the actions described below. For each action, label it with the role of the individual(s) that is *most likely* to be responsible for the action.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | **Action** | **Role** |  |
|  | Choosing the license under which the project will be released. |  |  |
|  | Deploying the software in a new and creative way. |  |  |
|  | Asking that a useful new feature be added to the software. |  |  |
|  | Discovering a bug in the software. |  |  |
|  | Providing a code patch that fixes a bug in the software. |  |  |
|  | Submitting an improved set of installation instructions. |  |  |
|  | Documenting a bug in the issue tracker so that it can be fixed. |  |  |
|  | Defining the goals for the next year of work on the project. |  |  |
|  | Incorporating a contributed bug fix into the main branch. |  |  |
|  | Redesigning a core software module in the system. |  |  |
|  |  |  |  |

**Creating a GitHub Account:**

You will be using git and GitHub for the remainder of these activities. In order to use GitHub it is necessary that you have an account.

3. Give the full URL to your GitHub profile (e.g. github.com/MyUserName) below. If you do not have a GitHub account, or would like to have a separate account just for this class you can create a GitHub account here: <https://github.com/signup>.

**Forking FarmData2:**

Recall that the first step in working on a FOSS project in GitHub is to make a *fork* of the *upstream* repository. As shown in Figure 1, creating a fork makes a full copy of the main project repository (i.e. the upstream) into your own GitHub space.

Figure 1 - Forking the Upstream

For the purposes of this activity (and then next few) you will not be interacting with the main FarmData2 repository. Instead, you will be interacting with a copy of FarmData2 that was created just for these activities. We will treat this copy as if it is the “Main Project Repo” or the upstream. This will allow you to practice interacting with an upstream for learning purposes without affecting the main FarmData2 repository, which is under active development.

The URL for the copy of FarmData2 that we will use as the upstream for these activities is:

* PLACE URL OF THE KITAINER VERSION OF FarmData2 HERE

4. Using your web browser of choice go to the upstream repository given just above.

a. Find the “Fork” button ( Text

Description automatically generated ) on the upstream repository page.

b. Click the “Fork” button to create a new fork of the upstream repository. Make sure that you create the fork into the GitHub account that you listed above in question #3.

c. Give the full URL of your fork. The URL for your fork will include your GitHub username.

5. In Figure 1 there is a dotted line from the fork back to the upstream showing that the fork knows about its upstream repository.

a. There some text and a link on the GitHub page for your fork that indicates the upstream repo from which it was forked. Copy that text and link here.

b. Click the link in the text you found for part a to go to the upstream repo. Use the back and forward buttons of your browser to compare the list of files contained in the upstream repo to the list of files contained in your fork. Is your fork an exact copy of the upstream repo? If you notice any differences describe them here.

c. In terms of the way that FOSS contributions are made, why is it important that your fork know about the upstream from which it was forked? Hint: Which operation requires this information?

6. Because a fork is a complete copy of the upstream repo there is no reason that there cannot be a fork of a fork, or a fork of a fork of a fork. That is, there can be multiple levels of upstream.

a. What is the full URL of the upstream of your fork?

b. What is the full URL of the upstream repository for the repository you found in part a?

**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. Because local environments can vary greatly, you will be working within a KitClient which is a Docker container that provides a known Linux development environment.

Figure 2 - Cloning your Origin

7. First let's install the software we need to run and interact with the KitClient.

a. Install and run Docker: <https://www.docker.com/>

b. Optionally install TigerVNC: <http://tigervnc.bphinz.com/nightly/>   
 This will provide a nicer interface for interacting with the KitClient.

8. Now let's run the KitClient.

a. Start the KitClient

On Windows with Docker running, open a Command Prompt and copy and paste the following:

mkdir kitclient

cd kitclinet

docker run --name KitClient --detach -v %cd%:/home/headless -p 6901:6901 -p 5901:5901 ghcr.io/hfossedu/kitclient:1.1

On Mac or Linux with Docker running, open a Terminal and enter:

mkdir -p kitclient

cd kitclinet

docker run --name KitClient --detach -v %cd%:/home/headless -p 6901:6901 -p 5901:5901 ghcr.io/hfossedu/kitclient:1.1

The above commands create a directory named kitclient, moves into that directory, and then starts the KitClient. All of the KitClient's data will be saved in the kitclient directory. If the KitClient ever stops, restart it by running the long docker command from within this directory. That way it will have access to all the data from your previous session.

b. If you installed TigerVNC open it to localhost:5901;  
 otherwise, open a browser to http://localhost:6901

c. The username and password are both: headless

The remainder of the activity will assume that you are working within the KitClient.

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.

a. Open a terminal window (inside the KitClient) and enter the command git. What output appears in the terminal when you attempt to run git?

b. If part a indicated that Git was not installed, then install the Git CLI. You can use apt or the Synaptic package manager, or some other package manager if you prefer. If you installed Git, what was the name of the package that you installed?

c. What version of Git is installed on your machine? Hint: Use a web search or man git to determine how to display the Git version number.

10. Now that you have Git installed, you need to do a little configuration to make it easier to use.

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

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

What output is generated by the above command?

11. Now with Git installed and 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 a terminal, and from your home directory, use the URL from part a in the following command:

git clone <URL>

What output is generated when you run this command?

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. What directory was your repository cloned into?

d. What are some of the files that are in the directory you found in part c. Compare those files to those that are in your fork on GitHub. Is your clone an exact copy of your fork? If you notice any differences describe them here.

12. 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, change to the directory containing your clone 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”). What output is generated by this command when you run it?

b. When you clone a fork to your local machine there Git uses a special name to refer to that fork. What name does Git use to refer to the fork that has been cloned?

c. Write a few sentences describing how your answer in part b connects to the output you observed in part a and how that relates what is shown in Figure 2.

d. In terms of the way that FOSS contributions are made, why is it important that your clone know about the origin repository? Hint: Which operation requires this information?

**The Issue Tracker:**

You have now forked and cloned the our FarmData2 repository and are in the state shown in Figure 3. From here you are now ready to do some work.

Figure 3 - Ready to Work

As described in class, FOSS projects often use an *issue tracker* to maintain a list of bugs to be fixed, features to be added and other tasks to be completed. Thus, the issue tracker is a good place to go in order to find something to work on in a FOSS project.

13. The issue tracker for the FarmData2 repository that you forked contains a list of issues that we will use for practice. This question explores that issue tracker.

a. Every issue has a *number* and a *title*. What is the title of issue #10?

b. Each issue can have *tags* that can be used to sort and categorize issues. What tags are associated with issue #8?

c. To which file in the repo does the typo described in issue #11 apply?

d. In what different orders can you ask the GitHub issue tracker to sort the issues?

e. Why might it be useful to sort issues by oldest or least commented or least recently updated?

14. Most projects will have a process by which you can *claim an issue* to work on or have an issue you want to work on *assigned* to you. You can claim an issue in the version of FarmData2 that you are working by commenting on the issue with the **exact** **message**:

I would like to work on this please!

The first person who comments on an issue with this message will have the issue assigned to them.

a. Visit the upstream repository **and comment on an issue that is tagged Round1 to claim it**. Double check to be sure you have been assigned the issue as it may be that someone else slipped a comment in just ahead of you while you were writing yours. If you get scooped like that, just pick another issue that hasn’t yet been assigned and try again.

What was the title and number of the issue that you claimed?

b. Some projects intentionally do not allow issues to be claimed or assigned to individuals. What do you think some advantages and disadvantages of claiming or assigning issues might be?

15. Do not attempt to fix your issue. That is, please do not make any changes to the files in your local repo at this time. We will be learning about the proper way to work with a repo next class period. If you accidentally made changes, please delete your clone and recreate it (see #9a, b). Type “I Understand” in the following box to confirm that your local repo has not been changed.

16. When you forked the upstream, GitHub made a copy of the repository into your account. Did it also make a copy of the issue tracker and its issues into your GitHub space? What reasons can you think of for why the designers of GitHub decided that a fork should behave this way?

If you are curious, you can explore the issue tracker for the main FarmData2 repository here:

* <https://github.com/DickinsonCollege/FarmData2/issues>

**Review & Reflection:**

17. Give a short description, in your own words, of what each of the following Git CLI commands does. Your description should do more than simply restate the command. I.e. “Clones the repository” is not a sufficient description for the git clone command.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | **Command** | **Description** |  |
|  | git config |  |  |
|  | git remote -v |  |  |
|  | git clone |  |  |
|  |  |  |  |

**Optional:** To help us improve and scope these activities for future semesters please consider providing the following feedback.

a. Approximately how much time did you spend on this activity outside of class time?

b. Please comment on any particular challenges you faced in completing this activity.

**Acknowledgements:**

Some materials, questions and resources have been adapted from activities posted on foss2serve.org:

* <http://foss2serve.org/index.php/Git:_Cloning>
* <http://foss2serve.org/index.php/Git:_Git_Intro_Activity>
* <http://foss2serve.org/index.php/Git:_GitHub_Issues_and_Pull_Requests>
* <http://foss2serve.org/index.php/Git:_GitHub_Workflow_Activity>
* <http://foss2serve.org/index.php/Intro_to_GitHub_(Activity)>
* <http://foss2serve.org/index.php/Version_Control_(Activity)>
* <http://foss2serve.org/index.php/Work_Locally_with_Git_from_the_Command_Line_(Activity)>

Some materials, questions and resources have been adapted from opensource.com under the :

* <https://opensource.com/open-source-way> (question #1)
* [*https://www.theopensourceway.org/the\_open\_source\_way-guidebook-2.0.html*](https://www.theopensourceway.org/the_open_source_way-guidebook-2.0.html)