

Title: Introduction to Repositories for Collaboration

Student Name: Danielle Valerie Guir

Date: 11/8/13

Executive summary:

This document will seek to provide an introduction to software development repositories and provide a guide to setting up a GitHub repository. An overview on how to initialize a GitHub repository and set up an environment, including a new branch, will also be within the scope of this document.

Keywords: Programming, Collaborative Projects, GitHub

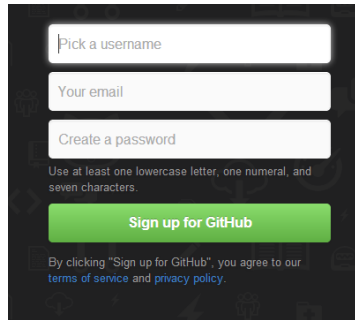
Repositories

Often in projects that require software development, it is necessary for more than one individual to contribute to the overall product. As a result of this needed collaboration, a version-control system becomes necessary to keep track of changes to the documents of the project. It also becomes a necessity to have ease of access to the documents, possibly from a remote location – i.e. remote laptops – as working on the same station to work on a project is inefficient and can become incredibly complicated. Therefore, a solution – a system that would be able to handle ease of file accessibility, as well as, distribution – is needed. In other words, a code repository is needed. Code repositories a convenient solution to the need of coordinating schedules to be able to efficiently develop the software needed for a particular project. In this way, each member of the software development team can contribute to the development without significant scheduling and location issues. Many institutions have their own code-revision software, though there are also a number of different web-hosted repositories available for download. Some of the kinds of easily accessible, and commonly known repositories, include: SourceForge, Google Code, Codeplex, GitHub, etc.

GitHub

GitHub is a web-based hosting service for software development. It supports private repositories for a fee and a free account for open source projects. It provides an extremely useful collaborative revision control and has a very convenient GUI application available for Windows and Mac. In addition, the site also offers: task delegation features, private versus public repository support, contributor workflow analysis, and file commit flow tracking. These features make it a very convenient choice for working on collaborative projects.

- I. Set Up: Initial set it up using GitHub
 - a. Create a user ID on the GitHub website: www.github.com



Pick a username

Your email

Create a password

Use at least one lowercase letter, one numeral, and seven characters.

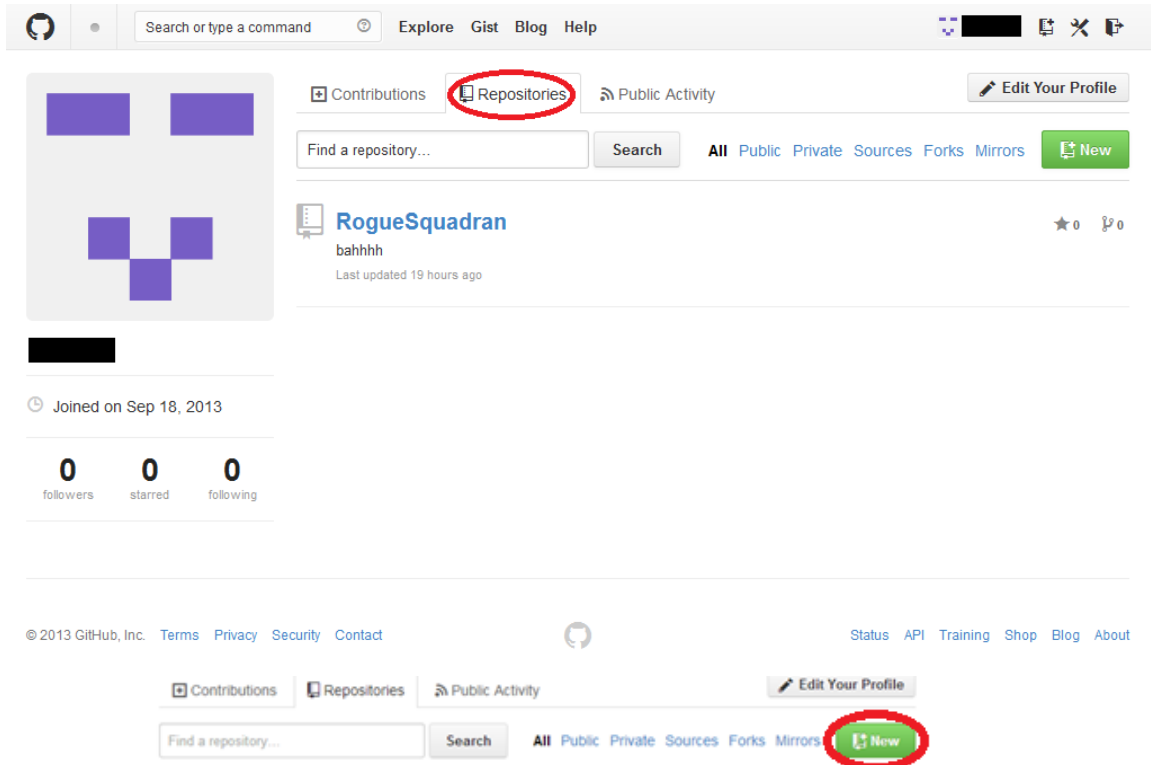
[Sign up for GitHub](#)

By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy policy](#).

b. Complete the steps to finish creating an account.

 Completed Set up a personal account	 Step 2: Choose your plan	 Step 3: Go to your dashboard
---	--	--

c. In completing the creation of a profile, now you can create a new repository by clicking the site's, convenient



Search or type a command

Explore Gist Blog Help

Contributions **Repositories** Public Activity Edit Your Profile

Find a repository... Search All Public Private Sources Forks Mirrors **New**

RogueSquadran
bahhhh
Last updated 19 hours ago

Joined on Sep 18, 2013

0 followers 0 starred 0 following

© 2013 GitHub, Inc. Terms Privacy Security Contact Status API Training Shop Blog About

Contributions Repositories Public Activity Edit Your Profile

Find a repository... Search All Public Private Sources Forks Mirrors **New**

d. Remote setup using a command line

i. Once you are in the preferred directory for your development

1. To create a new directory

```
git init
```

2. To add a new file and commit changes to the remote repository

```
git add README.md
```

```
git commit -m "first commit"
```

3. Add the remote repository using the path given by GitHub.com

```
git remote add origin [path provided by  
Github.com]
```

e.g.

```
git remote add origin  
https://github.com/<user_name>/<repository_name>.git
```

4. Update the GitHub repository

```
git push -u origin master
```

5. Create a branch

```
git checkout -b [name of new branch]
```

By doing this you will have created a remote repository with a README.md file and a new branch where you can save your features during development, and then merge back to the master branch.

e. Cloning an existing repository

i. Get a copy of the data available on the GitHub server

```
git clone [url provided on the GitHub website]
```

e.g.

```
git clone https://github.com/<user\_name>/<repository\_name>.git
```

or

```
git clone  
https://github.com/<user\_name>/<repository\_name>.git <branch_name>
```

This will create a directory based on the branch that you are cloning. All the updated data within that branch will also be pulled down, ready to be used.

Note: ¹You can also clone a branch from this original url by adding the name of the branch following the url.

²There are a number of transfer protocols available for use:

1. Git://
2. https:// or <user>@<server>:/

For the purpose of this guide, the SSH transfer protocol <https://> will be used for the url paths provided by the website.

f. Updating local version of files

- i. Pull down the data from the web-hosted repository

```
git pull
```

```
http://github.com/<user\_name>/<repository\_name>  
<branch_name>
```

g. To add a new file and update the repository

- i. Once you have finished creating and editing the file you wish to add.

1. Update the remote version of the repository

```
git pull <remote> <branch>
```

2. Merge any conflicts between your modified, remote, version and the version in the GitHub repository, add the file if it is a new file.

```
git merge <branch>
```

```
git add <file_name>
```

3. Commit the new changes

```
git commit -m "<commentary>"
```

4. Push the changes to the

```
git push
```

```
https://github.com/<user\_name>/<repository\_name>  
> <branch>
```

II. Issues: with using a repository, possible errors

- a. Merge conflicts

A merge conflict occurs when versions of the same file on two branches, are merged together and one version does not match the version in the other branch.

There are a couple of ways of resolving these issues, one way is using the GitHub for a Mac client. Another way of resolving the issues is through a command line.

There are also, multiple types of merge conflicts. One common example is version conflicts. This is when the same file in two branches, being merged together, are not identical – due to edits. These conflicts are marked with conflict markers, beginning with <<<<<<, ending with >>>>>>, and are divided by a =====. These conflicts must be resolved by hand, before being committed.

Another common type of merge conflict is a removed file conflict. This is when the current branch contains a file that has been removed from the local copy. In other words, one branch has the file and the other does not. This conflict can be resolved in two ways: either remove all copies of the file in question, or revert the version of the branch with the missing file to re-create the file.

b. Other errors

GitHub has a fairly extensive help link which is a good resource for other issues, including HTTPS cloning errors, commit issues, repository errors, user and token errors, etc. The link is as follows:

<https://help.github.com>

Conclusion

In completing the guide, you have now been able to create and initialize a code repository, add new files to the web-hosted repository, commit changes made, know how to update the local version of the files, and now know how to clone an existing repository to your local station using GitHub through a command terminal. It must be noted that there are resources on the GitHub.com website with instructions to access the features using the OS application. You have also been able to create a branch, add and commit a new file to the new repository. From this point, using and testing code on a remote, local, station is relatively easy. Adding additional features to the display and the output file storage management, which will later be ported to a microcontroller, are a couple of examples of how our team expects to use the repository.