HYDROSHARE

HOW TO PUSH/PULL CHANGES IN GITHUB

October 7, 2013

by:

Stephanie Mills
Renaissance Computing Institute
University of North Carolina at Chapel Hill

Table of Contents

CHANGE NOTES

Date	Who	Notes
Oct 7, 2013	Mills	Initial version
Oct 16, 2013	Valentine	Added a workflows, change log
Oct 21, 2013	Mills	Edited new content for consistency & format
Feb 4, 2014	Mills	Updated some out-of-date information

1. New to GitHub?

1.1. Overview and Help

Git is a *distributed version control system*. Git allows programmers to share code easily with each other while maintaining separate production versions as well as a record of previous versions of code.

GitHub is a website for sharing *repositories*. Repositories on GitHub must be open source. GitHub provides social networking around these repositories, so users can leave comments, *fork* repositories into their own accounts, and submit *pull requests* to suggest changes, as well as use many other collaboration tools.

Hydroshare is located in a repository on GitHub. In order to contribute to Hydroshare, you will need to be logged into your GitHub account, which you can register for free at GitHub.com. You will also need to set up Git on your local computer. Instructions for this process can be found at GitHub Help: Set Up Git.

This document is specific to using Git/GitHub to contribute to Hydroshare. For more general assistance, please see GitHub Help.

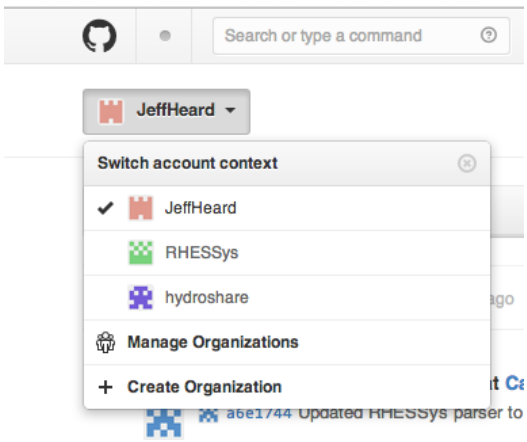
1.2. SSH Keys

You will need an *SSH identity* in order to push changes from your local repository to GitHub. An SSH identity is composed of a public and private key. Your private key stays on your local computer, and you must provide GitHub with your public key in order to connect to your own repositories and, with permission, others' repositories such as Hydroshare. Once you've completed this setup, you will not have to re-authenticate your local computer in the future.

Your key can be found in `.ssh/id_rsa.pub` or `.ssh/id_dsa.pub` in your Linux home directory. Once you have your key, you will need to enter it in your "Account Settings: SSH Keys" on GitHub.com. Once logged in, simply navigate to Settings: SSH Keys and select "Add SSH key" on the top right; then enter your key and select "Add key" on the bottom left. For more detailed instructions on how to complete this necessary setup, please refer to GitHub Help: Generating SSH Keys.

2. Pushing/Pulling Changes: Within Hydroshare Organization

Users within the Hydroshare organization will use GitHub's *shared repository model* to *push* and *pull changes* to Hydroshare. Users in this category will have push access to the Hydroshare *repository*, and will push their changes directly to this repository. You can check to see if you are member of the Hydroshare organization on GitHub.com, by clicking this dropdown menu on the top left:



If you see "hydroshare" on the dropdown list, you are a member of the Hydroshare organization. If you do not fall into this category, see section 3. Pushing/Pulling Changes: Outside Hydroshare Organization.

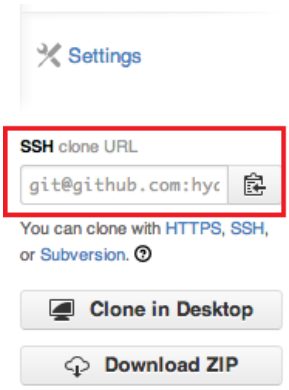
2.1. How to: Clone a Repository

Before you can begin contributing to Hydroshare, you need to *clone* the repository. This will essentially copy all Hydroshare files, including all past and current versions, to your local computer.

In order to clone a repository, you will simply need to run the following command:

```
$ git clone git@github.com:<username>/<repository>.git
```

You can also find the appropriate clone URL by navigating to the Hydroshare repository on GitHub and looking for "HTTPS clone URL" on the bottom right, outlined in red, shown in the image to the below:



You may then proceed to work on the project on your local computer.

2.2. How to: Commit

When you *commit* using Git, you save changes to your local repository. This differs from other revision control systems (such as Subversion) in which commits save changes both locally and remotely. For details on this process, please see Step 2: Commit your README in GitHub Help: Create a Repo.

To commit to your local repository, use this command:

```
$ git commit -m "commit message"
```

It is encouraged to commit often. If you later find that the number of commits in your history is confusing, or you just want to revise them later, Git allows this. See Rewriting Git History for more information.

2.3. How to: Pull Changes from Hydroshare

Before you pull any new changes from Hydroshare, it is best to make sure that all of the changes you've made on your local computer are committed to your local repository (see section 2.2. Committing).

Once your changes are committed on your local computer, you can pull changes from Hydroshare. This will take all changes made to Hydroshare and automatically *merge* them with your local copy. To do this, you can use the following command:

```
$ git pull <remotename> <branchname>
```

Specific to Hydroshare, this command will look like this:

```
$ git pull origin development
```

Typically, however, simply using:

```
$ git pull
```

should suffice.

If you have a merge conflict or forgot to commit all of your changes to your local repository before pulling, you can use the following command to return your local repository to the last version prior to your pull:

```
$ git merge –abort
```

2.4. How to: Push Changes to Hydroshare

Before pushing any new changes to Hydroshare, you should ensure that you have pulled (and consequently merged) to your local repository any changes that have been pushed to Hydroshare while you were working. If you have not pulled all new changes, you may be unable to push. For information on pulling changes from Hydroshare, see section 2.3 How to: Pull Changes from Hydroshare.

You can now push your changes to Hydroshare with the following command:

```
$ git push <remotename> <branchname>
```

Specific to Hydroshare, the command will look like this:

```
$ git push origin development
```

Typically, however, simply using `$ git push` should suffice.

3. Pushing/Pulling Changes: Outside Hydroshare Organization

Users who are not members of the Hydroshare organization will use the *fork and pull request model* to add to Hydroshare. If you fall into this category, you must first *fork* the *repository*, then *clone* it to your local computer, and then *push changes* to your personal forked repository. When you want to share your changes, you will issue a *pull request* to Hydroshare, after which the project maintainer may choose to *pull your changes* into the Hydroshare repository. For more details about this process, see GitHub Help: Fork a Repo.

If you need to check if you're a member of the HydroShare organization, please see the introduction in section 2. Pushing/Pulling Changes: Within Hydroshare Organization.

3.1. How to: Fork a Repository

Forking the Hydroshare repository will create a personal copy of the entire Hydroshare repository on your account on GitHub. To do this, navigate to the Hydroshare repository on GitHub, and click the "Fork" button on the top right:

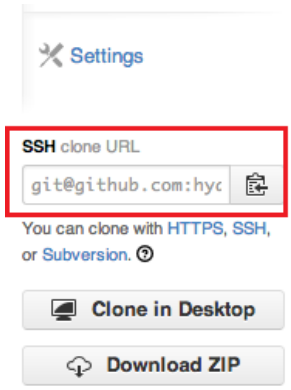
3.2. How to: Clone your Repository

Before you can begin contributing to Hydroshare, you need to clone your personal fork of the Hydroshare repository. This will essentially copy all Hydroshare files, including all past and current versions, to your local computer.

In order to clone your repository, you will simply need to run the following command, substituting your username and the name of your forked repository:

```
$ git clone git@github.com:<yourusername> /<yourrepository>.git
```

You can also find the appropriate clone URL by navigating to your fork of the Hydroshare repository and looking for "HTTPS clone URL" on the bottom right, outlined in red, shown in the image to the below:



You may then proceed to work on the project on your local computer.

3.3. How to: Commit

When you commit using Git, you save changes to your local repository. This differs from other revision control systems (such as Subversion) in which commits save changes both locally and remotely. For details on this process, please see Step 2: Commit your README in GitHub Help: Create a Repo.

To commit to your local repository, use this command:

```
$ git commit -m "commit message"
```

It is encouraged to commit often. If you later find that the number of commits in your history is confusing, or you just want to revise them later, Git allows this. See Rewriting Git History for more information.

3.4. How to: Pull Changes from Hydroshare

Before you pull any new changes from Hydroshare, it is best to make sure that all of the changes you've made on your local computer are committed to your local repository by running this command:

```
$ git commit -m "commit message"
```

For details on this process, please see Step 2: Commit your README in GitHub Help: Create a Repo.

Once your changes are committed on your local computer, you can pull changes from Hydroshare. This will take all changes made to Hydroshare and automatically *merge* them with your local copy. To do this, you can use the following command:

```
$ git pull upstream <branchname>
```

Specific to Hydroshare, this command will look like this:

```
$ git pull upstream development
```

Typically, however, simply using:

```
$ git pull
```

should suffice.

If you have a merge conflict or forgot to commit all of your changes to your local repository before pulling, you can use the following command to return your local repository to the last version prior to your pull:

```
$ git merge –abort
```

3.5. How to: Push Changes to your Forked Repository

Before pushing any new changes to your forked Hydroshare repository, you should ensure that you have pulled (and consequently merged) to your local repository any changes that have been pushed to the original Hydroshare repository while you were working. If you have not pulled all new changes, you may be unable to push. For information on pulling changes, see section 3.4 How to: Pull Changes from Hydroshare.

You can now push your changes to your forked Hydroshare repository with the following command:

```
$ git push <remotename> <branchname>
```

Specific to your forked Hydroshare repository, the command will look like this:

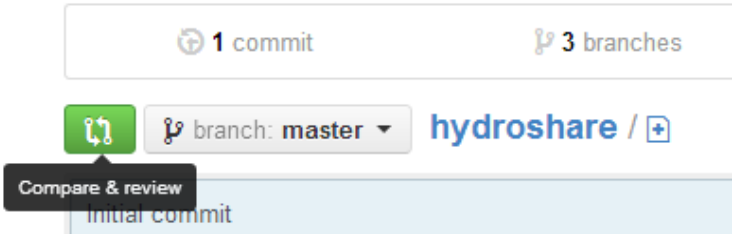
```
$ git push origin development
```

Typically, however, simply using `$ git push` should suffice.

3.6. How to: Send a Pull Request to Hydroshare

If you would like to contribute your changes back to the original Hydroshare repository, you can send a pull request to the project maintainer. For more general information about pull requests, see GitHub Help: Using Pull Requests.

To send a pull request, navigate to your forked repository and the appropriate branch, and select the green pull request button labeled “Compare & review,” pictured here:



You pull request can then be evaluated by the Hydroshare project maintainer.

4. Workflow Patterns and use cases

The previous sections present GitHub commands. This section focuses on the workflow pattern specific to Hydroshare, or the best practices and procedural steps needed to contribute to Hydroshare. For more general information about GitHub workflows, see this article on GitHub flow.

4.1. Notes

Hydroshare uses a *distributed source control system*. Under this system, contributors may work on code for Hydroshare on their local computers. When using this model, it is best to commit to your local machine often. This acts as a "save point" and allows you to easily reverse unwanted changes to a stable state.

When you have a working codebase that is ready for use, commit it to Hydroshare on GitHub.

Note that if you add a content type via the administrative interface, it will be in your local test database and not in the code. Code must to be exported via features.

4.2. Development on Hydroshare Branch

Developing on the main branch means that the code will be available to other developers. Any commits to the Hydroshare GitHub will be seen by other developers. To work on the Hydroshare branch, follow these steps. Each step in the process links to a section in this guide (if one exists) with more detailed instructions.

1. Clone the Hydroshare repository to your local computer, and/or pull any changes.
2. Make your edits to the code.
3. Commit your changes locally.
4. Pull any new changes if necessary.
5. Run tests.
6. Push your changes to Hydroshare, or preferably, send a pull request. Even if you are within the Hydroshare organization, it may be a good idea to send a pull request instead of simply pushing changes.

4.3. Create a Module Using a Fork

If you are working on a feature that will take a while to develop, or if you fear breaking Hydroshare, then you may want to work on a fork. This will allow for you to develop without fear of your code being mixed into the beta and production Hydroshare codebases. To work on a fork, follow these steps. Each step in the process links to a section in this guide (if one exists) with more detailed instructions.

1. Create a fork.
2. Clone your fork.
3. Work on your fork (make edits, etc.).
4. Commit changes frequently as you work.
5. Pull any new changes from the main Hydroshare repository and merge them with yours.
6. Commit all changes. Shelve any changes you do not wish to commit. See Git: Shelving changes and bringing them back later for more information.
7. Push changes to your fork.
8. Sent a pull request to the main Hydroshare repository.
9. Optional: Restore any shelved changes on your local repository. See Git: Shelving changes and bringing them back later for more information.

5. Glossary

A glossary of GitHub terminology can be found at GitHub Help: GitHub Glossary.

5.1. Clone

A clone is a copy of a repository that exists on your local computer. Editing your clone does not require you to be online. Changes that you make to your clone can later be pushed to the corresponding repository on GitHub.

5.2. Commits

A commit is a saved change made to file. Rather than overwriting previous versions, Git keeps every commit that you make. Each commit records where changes were made and who made them, and also includes a message describing the changes.

5.3. Distributed Version Control System

Git uses a distributed version control system to keep track of all revisions to software. This allows contributors to work on a project simultaneously on their local computers, and does not require a network connection during editing.

5.4. Fork

A fork is a personal copy of another user's repository. It differs from a clone in that it is saved on GitHub, not your local computer. Forks can be updated by pulling changes from the original repository, and can contribute back to the original repository by sending pull requests.

5.5. Fork and Pull Model

Under this model, users who wish to contribute to a repository first fork that repository, make their changes, and then send a pull request back to the original repository's project maintainer, who can decide to accept these changes or not.

5.6. Git

Git is an open source program that tracks changes made to files.

5.7. GitHub

GitHub is a website that allows users to view, share, comment, and contribute to other users' code. GitHub is a social and user interface built on Git.

5.8. Merge

Merging incorporates changes made in one branch back into the main body of work.

5.9. Pull Changes

Users pull changes when they wish to taking changes that others have made to a repository and incorporate them into their local copy.

5.10. Push Changes

Users push changes when they have made changes on their local copy and wish to add these changes to a remote repository so others can access them.

5.11. Pull Request

Users may issue a pull request when they've made changes to their forked copy of a repository and wish to share those changes back to the original repository so others can access them.

5.12. Repositories

Repositories are essentially a project's entire folder, including all previous versions, branches, and documentation.

5.13. Shared Repository Model

Under this model, users who wish to contribute to a repository are added to that organization and may push changes directly to that shared repository.

5.14. SSH Identity

An SSH identity includes a public and a private key. The private key stays on your local computer, and the public key is provided to GitHub. This allows GitHub to identify you and give you access to your repositories.