Jim Tang

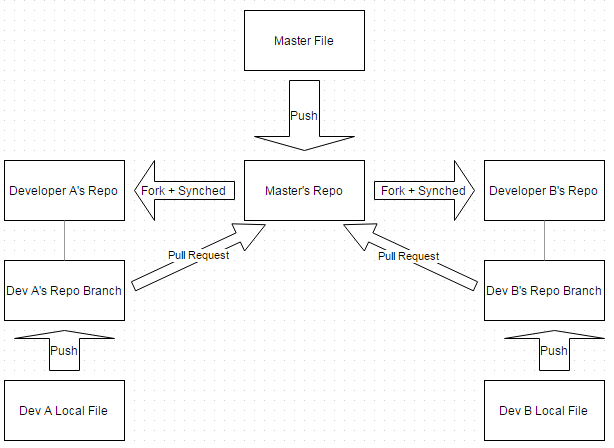
IconCells

**INTRODUCTION**

This documentation serves as guide to help fellow developers become more familiar with the process of file management using GitHub. IconCells mandates that you have WordPress (WP) setup locally and ready for version control using GitHub documentation and its file sharing properties.

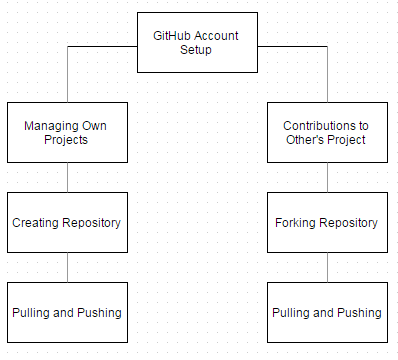
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Why use GitHub? Answer: Version control; managing what comes in and what can't. The schematics below, shows the master being the source of the project. Developers are able to copy (clone) the files into their own repository (repo) and sync them so that it can fetch updates. Developers will want to create a branch so they can work on improving the source without altering their forked repo (clone of another's repo) and also note; the forked repo is left untouched solely for the purpose of fetching updates from the master. Once developers finish their changes and push the updates to their branch, they will submit a pull request, requesting a merger between their own branch and the master repo. In return, all developers' synched repo will retrieve the update(s).



**Figure 1: GitHub's Version Control Schematics Diagram.**

All developers will need to create an account to participate in GitHub's file sharing and version control. Depending on which project objective you are pursuing (either acting as the original developer/ managing what is updated OR contributing to help other's improve their project), will lead to different method of approach as seen in the flow diagram, below.



**Figure 2: Flow Diagram; Left path for managing own repo. Right path for contributing to other's repo.**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**GITHUB ACCOUNT SETUP**

1. Access github.com and create an account

2. Download git shell depending on operating system used (i.e. windows, OSX, Linux)

*Assumption: I will be using Git for Windows*

Notes: A Repository is a folder for project location

3. Open Git Shell (or whichever command prompt you are using) and enter the following

while replacing thisIsYourNameHere and thisIsYourEmail@address.com with your user name

and your email address

git config --global user.name thisIsYourNameHere

git config --global user.email thisIsYourEmail@adrress.com

(this will set your username and email for every repository on your local device)

Test: git config --global user.name

git config --global user.email

(entering the above command should display the username and email you have chosen)

Notes: Verify that your email is registered on GitHub's setting page followed by in the email section

(you may need to restart browser after the request but before confirming)

*(continued)*

4. Activate email notifications on GitHub site (this is used to keep updated of changes to repo you

are watching:

4a. View your profile on GitHub site; select repositories tab, select the name of the

repository you want to get email notifications on

4b. Select the settings icon, then webhooks&services, then add services and select email

4c. Enter the email address that would like to receive notifications

Test: Create a repository> settings> webhooks&services>add the email services> edit the readme.md on GitHub site. You should receive an email of the change

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***IF YOU WANT TO MANAGING YOUR OWN PROJECT:***

**CREATE A REPOSITORY**

1. On your account’s GitHub homepage, select: create repository

2. Create the project name, add a description, and initialize the repository

3. Create the repository

Test: Refresh your homepage to see the newly created repository

\*\*\* This new repository is provided with an URL that can be used for you to make copies to your

local computer and update it, see Pulling and Pushing

***IF YOU WANT TO MANAGE YOUR CONTRIBUTIONS TO OTHERS’ PROJECTS:***

**FORK A REPOSITORY** (creates a repo to your account using other's repo)

1. Visit the targeted repo page on GitHub and select the fork button

\*\*\* This will create a copy of the file to your repository on your own GitHub

Test: Refresh your home page to see that the file has been added as one of your repository

2. Once the repo is created; create a branch either through the GitHub site or using the command in

the file directory (you must be in the file directory, change directory if needed)

git checkout -b newBranchName

replace "newBranchName" with your chosen name for the copy of the forked repo

Test: Refresh GitHub, select the branch tab, a scroll menu will show master and branch

Or use git status on command prompt

\*\*\* When you make changes on the branch, simply treat it as your own repo. Update the

branch, not the master because the master is where you will pull updates from the original

In order for your master to pull updates from original, you must SYNC the fork:

1. Using the command line, change directory to the same directory IN the fork you created

locally (the fork created is your copy of the original master) replacing Owner and file with the original owner and file name of the repo you forked

git remote add upstream https://github.com/Owner/file.git

Test: git remote -v will show the your origin file pulling from which upstream

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**PULLING REPOSITORY = IMPORT TO YOUR COMPUTER BY MAKING A COPY**

\*\*\* This will create a copy of your repo to your local device using its Git URL from your repo

1. Create a folder for the clone file on your local device

2. Using command line change directory to the file location you want the destination to be

Example:

If my folder is named "myFolder" and is located on my desktop under user "Jim", enter the following in Git Shell command prompt:

cd c:/users/Jim/desktop/myFolder

3. Making the copy of an existing project from GitHub to local

(For practices, we will use my project URL. But in the future, you will have created your own

repo or forked it prior to using its URL)

git clone https://github.com/jimt2189/restaurant.git

Test: Check the local folder to verify that the files have been “pulled” to the folder

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**PUSHING REPOSITORY = UPDATING REPOSITORY ON GITHUB SERVER**

\*\*\* You have made changes to your local file and want to push it to your GitHub repo

1. Using the command prompt, change directory to the file location; apply: git status to see

status (anything in red has not been updated to the repository)

2. git add fileName.txt to add to the repository (will be colored green)

3. git commit -m “use the quotations with a comment as well”

this allows the file ready to be uploaded and forces you to comment on the changes within the

quotes

4. git push will add the changes to the repository on GitHub

Test: Refresh the repository page on github.com to see changes made

--- --- --- --- --- --- --- --- --- --- --- --- ---

*MERGE CONFLICT*

\*\*\* Occurs when a file is attempted to be pushed before it can be updated

Example:

Two individuals pull a repo at the same time, one pushes and file is merged before the other.

When the other tries to push, merge conflict occurs because the file he/she was working on is

considered outdated.

1. When this occurs, use git pull on command prompt to view the conflict in the modified file

2. Make the changes on the conflicted file and push it back to the main repo on GitHub

Example:

<<<<<<

Your changes are shown here

=======

Others' changes are shown here

>>>>>>

*REMOVED FILE CONFLICT*

\*\*\* Occurs when a file is removed before it can be updated

1. Using git status will display the conflict

1A. Resolve by adding it back; simply by git add followed by git commit -m

then git push (do not forget to comment)

1B. Resolve by deleting it; use git rm instead of git add, then git commit -m

then git push (do not forget to comment)

Test: Either case, reload the page to see the changes on GitHub webpage

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\*\*\* Updates are automatic on GitHub server. Locally, you will need to git pull to update.

\*\*\* If you are the owner of the repo and someone has submitted a pull request, you will see

the notification in your email, select the link to bring you to the page where you can either merge

or close pull request; always make a comment.

\*\*\* If you want to update other's repo, submit a pull request comparing your branch to their master,

after you have push the updates to your branch, this will request the owner merge your changes to

his master file (owner will decide on this request)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**SOCIAL**

\*\*\* Don’t forget to Follow others and/or Watch projects to be posted on updates

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**COMMAND LINE KEYS MOSTLY USED**

ls Lists all documents in the file u have just access

cd Changes directory

cd .. Moves backward out of a destination

git clone Creates copy

git add . Adds all modified files

git pull Updates current directory with all up to date changes