**Git and GitHub Tutorial**

|  |  |
| --- | --- |
| **Version Control System(VCS)**   * Manages source code by tracking changes over time * (SCM) Source code management * Ex. Having different version of files for Microsoft Word when you save it. * Ex. Photoshop’s History panel * Ex. Ctrl+Z or undo is another example of Version Control |  |
| **Who should use Git?**  Answer: Anyone wanting to track edits   * Review a history log of changes made * View difference between versions * Retrieve old versions   Answer: anyone needing to share changes with collaborators | **Answer:** Anyone not afraid of command-line tools (LOL)! |
| **Git also keeps track of non-coding files like movies, music, or images, pdf’s, Word files…**   * However, it isn’t as efficient in terms of pointing out differences between the different versions |  |
| **Distributed Version Control**   * Different users (or teams of users) maintain their own repositories, instead of working from a central repository * Changes are stored as “change sets” or “patches” * Tracks CHANGES not VERSIONS. * NO single MASTER repository; just many working copies (each with their own combination of change sets.) | **Unlike other version controls where users have to communicate with the master/central server to make a change…with Git, you can make any change you want freely.**  **This means not network is required, everything is faster, no single failing point because your not working with a synced copy of the master.**  **You literally have your own master if you call it that of one master goes down, everyone else has their own.**  **Encourages “forking” of projects.(you have total freedom. You can leave if you want and just use your version)**  **All repositories are equal in Git.** |
| **Git Tutorial (newBoston) Config Username and Email**   * $ git config --global user.name "Zombie Slayer" * **git config –global** (configures name, email, or other code for it to have meaning to the rest of the program) * **git help** (displays all the topics for you to choose from to get help in) * **git help [topic]** (gives you more information about a particular topic) |  |
| **Git Tutorial (newBoston) Creating Our First Repository**   * **cd=**change directory * **~=**home directory * **pwd=**command that lets you know what directory you’re in(ex. C drive) * **ls=**list all the folders inside the directory * **repository=** a git project. * **git init=**starts new git repository * **git commit -m “message”=**a commit is a snapshot of a point in time where you edited code. * **git add .**=command to add information changed to previous version(git purgatory) * **ls -la=** shows you hidden files. DO NOT TOUCH!!!! |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **Git Tutorial(newBoston)Adding Files and the Commit Log**   * **git log:** shows the history of edits/commits in repository * **git status:** shows you if there are any edits that have not been commited to the repository. * **git log --author=":** allows you to see changes from a specific person |  |
| **Git Tutorial (newBoston) How to Edit Files**   * **git add (FileName)=** how to add an edited file * **git add .=**how to add/edit multiple files |  |
| **Git Tutorial (newBoston) Viewing the Changes That You Made**   * **git diff=**shows you the difference between your edited file and previous one before you add it to repository(purgatory) |  |
| **Git Tutorial (newBoston) Comparing the Staging Area with the Repository**   * **git diff --staged=** shows difference between “staged file” and repository * **git diff=** shows difference between your unstaged edited file and repository . |  |
| **Git Tutorial (newBoston) How to Delete Files**   * **git rm (FileName)=**removes file from repository and computer |  |
| **Git Tutorial (newBoston) How to Move and Rename Files**   * **git mv (FileName)(other FileName)=**how to rename file * **git mv(FileName)(FolderName)(otherFileName)=**moving a file to another folder | **Why cant I move my files to another folder outside of my working one?**  Probably because it isn’t initialized. |
| **Git Tutorial (newBoston) Working with an Actual Website**  **Git Tutorial (newBoston) How to Commit Directly to the Repository**  **git commit -am “”:**   * This commits your changes directly to the repository vs having to go into the staging area (purgatory) first. |  |
| **Git Tutorial(newBoston) Checkout this Video (Git it?)**  **git checkout – (FileName)**   * This undo’s your last change. |  |
| **Git Tutorial(newBoston) Unstage Files**  **git reset HEAD (FileName)**   * The command to unstage something in the STAGING area |  |
| **Git Tutorial(newBoston) Getting Old Versions form the Repository**  **git checkout (file#) – (FileName)**   * How to revert to old versions of your project. * VERY USEFUL!!! |  |

|  |  |
| --- | --- |
| **Git Tutorial (newBoston) Git Hub**  **Github**   * A site to post your projects out in the open | The reason for using Git w github is that you can download projects you think is cool from github, make changes, save those changes as different version on git, then publish back on github. |
| **Git Tutorial (newBoston) Pushing to a GitHub Repository**  **Remote**   * This is a connection you make between your local computer and the github server * **git remote add (nameOfConnection…can be whatever) (git link when repository is created)**   **Push**   * outputting changes to the server from your computer   **Fetch**   * retrieving or downloading items from server to your computer   **git push -u (nameOfConnection) master**   * makes the file or whatever you put on github, the master copy |  |
| **Git Tutorial (newBoston) GitHub Desktop**   * To have changes on github appear on GitHub Desktop, click on the plus sign on the top left corner and clone repository. |  |
| **Git Tutorial (newBoston) Commiting Changes to GitHub**  Make change and then commit to master and then click sync. |  |
| **Git Tutorial (newBoston) Branches**  **Branch**   * A copy of your repository. That we you can make edits and your friends make edits, ask to get accepted and you merge the file with the changes while at the same time preserving the original copy.   **ReadMe**   * File that is like a summary or brief description about your project   **New Pull Request**   * This asks the master or head of the project to update the master repository with the new changes * **MergePullRequest**=Yes * **ClosePullRequest**=No |  |
| **Git Tutorial (newBoston) Watch Star and Fork**  **Raw**   * The text version of your code…that way when you try to copy and paste it doesn’t pick up the coding functions as well.   **Blame**   * Shows you who edited your code. Ex. Line 12-16 by Jerry. No, you know who to blame if someone messes up lol.   **History**   * Shows history of your **FILE.** Commits shows history of the entire Repository.   **Watch**   * This allows you to see all the updates that are going on your website   **Star**   * To bookmark something of interest.   **Fork**   * Is how you make copies of someone else’s repository to work on. |  |
| **GitHub Tutorial (newBoston) Issues and Labels**  **Issues**   * These are task that you can delegate to other people on your team * On your issues you can upload files as well as code snippets   **Labels**   * These tell you the type of issue. (bug, enhancement, help, etc.) * You can also create custom labels.   **Assign**   * Assigns task |  |
| **GitHub Tutorial (newBoston) GitHub Wiki** |  |
| **GitHub Tutorial (newBoston) GitHub Organizations and Teams** |  |