Git-Hub Training

1. Check applications on all PCs
   1. GitHub account config? (<https://github.com/>)
   2. GitHub downloaded? (<http://windows.github.com/> or <http://mac.github.com/>)
2. Fork my repo to your own (<https://github.com/SMcCalden/GitHubTraining>)
3. Initialising or cloning a Git repo
   1. Initialise blank repo…
      1. Demo online manual creation of repo
      2. Demo manually initialising a blank repo from MAC command line following on-screen init instructions
   2. Clone the repo…
      1. Direct clone from the browser demo
      2. Clone request from the GitHub Client demo
   3. Create repo from existing files…
      1. Generate repo from GitHub Client demo
   4. **Exercise…**
      1. Everyone clone the forked training repo from your own GitHub.com <https://github.com/SMcCalden/GitHubTraining> to a folder of your choice locally
4. Pull Requests
   1. Important to keep local repo fresh & up-to-date
   2. Regular pulls recommended & 100% before any branch or feature work is begun
   3. Manually make a change to a file on the browser
   4. Demo the file’s current state locally
   5. Execute pull request
   6. Confirm the file has updated
   7. **Exercise…**
      1. Everyone execute a pull request for your training repo
      2. Confirm the updated file has changed
5. Adding, Editing & Commiting
   1. Add
      1. Demo adding a new folder
      2. Demo adding a new file within that new folder
   2. Editing
      1. Demo editing an existing file
   3. Committing
      1. Demo commit of a file
         1. Commit message important!!!
   4. NB : Multiple users editing a common file may cause commit/merge problems so best to coordinate yourselves to only edit completely distinct file OR employ feature branching as file.txt can be edited within Branch1 and Branch2 perfectly fine.
   5. **Exercise…**
      1. Add a new root folder of your surname
      2. Add a “firstname”.txt file within that have “Hello World!”
      3. Stage all the changes made
      4. Commit all the changes made with unique commit messages
6. Push Requests
   1. Demo pushing via command line
      1. Explain the branch 🡪 branch confirmation
      2. Confirm the push success in the browser online
      3. Confirm the commit messages / files / etc. online
   2. **Exercise…**
      1. Push all of your local changes up into the training repo
      2. Confirm the changes online have been pushed successfully
7. Branching
   1. Never work directly on the master branch
   2. Always employ feature branching
   3. Create a branch
      1. Demo git checkout -b NEW\_BRANCH
      2. Confirm changed to the new branch via git branch
      3. Edit an existing file with a personal & unique change
      4. Stage & commit the change
      5. Push the change up to the training repo IN THE NEW BRANCH (**\*not master\***) >> git push NEW\_BRANCH
   4. **Exercise…**
      1. Create a new branch of your surname
         1. NB : You may need to PULL the training repo again to get his/her new folder/file!!! ;-)
      2. Change to the branch
      3. Edit the ../surname/firstname.txt
      4. Stage, commit & push the file back up to the training repo IN YOUR BRANCH!!
      5. Confirm via the browser that the master and new branches differ for this specific file
8. Merging
   1. Always have merges executed by ANOTHER person – essentially a code review
   2. Merge…
      1. Demo online how to trigger a merge request
      2. Assign it to another person
      3. Examine the diffs in the merge request
      4. Execute the merge
      5. Delete the old branch
   3. **Exercise…**
      1. \*Probably no time for this but try in your own time to issue a merge request from **Branch B** into **Branch A** and confirm the merge is successful & Branch B deleted\*
9. Miscellaneous
   1. Gitignore
      1. File at root level which holds extensions of files NOT to be included within commits
      2. Useful for different project IDEs (NetBeans vs Eclipse)
   2. README.md
      1. This is an intro file useful as a first port of call for new users
      2. Ideally should contain description, installation, execution info but really anything is possible
      3. Very precise coding standards so “m” to view the syntax available for layouts
      4. Multi-layered README files are good to explain complex nested sections within the code base
   3. Stashing
      1. This is any idea that a change is made locally but not ready to go up
      2. Stash it locally to exclude from the commit/push and return to later
      3. I’m not a fan as clearly this is an example where featured branching should’ve been employed
   4. SSH keys
      1. Private github ssh key can be added to the PC
      2. Public github ssh key can be uploaded to GitHub
         1. Demo the SSH key location
      3. 2nd secure layer of confirming secure access to the repo
   5. Git Reference (<http://gitref.org/index.html>)
      1. Online manual of command-line options if you need to automate the git pull, edit, commit process