**GIT/MAVEN COMMAND CHEAT SHEET**

Key:

< > : replace with your local identifiers

( ) : optional commands that may be chained

Maven Commands

* Cleaning: *mvn clean*
* Packaging: *mvn (clean) package*
  + Recommended you use the *install* command below unless you specifically want to package
* Installing: *mvn (clean) install*
  + Recommended that you include *clean* as part of the *install* command
* Running: *mvn spring-boot:run*
  + It should run on localhost:8081

GIT Commands

Notes:

* **Remember to create a local directory for you to work in**
* The remote repository is at <https://github.com/scottwares/chessapp>
* Remote contains two branches, *develop* and *master*
  + *master* is for production ready builds **only**
  + *develop* is for, well, developing
* Do **not** push to *master*. When cloning the repo initially I recommend avoiding cloning *master* and just cloning *develop* (see *cloning commands*).
* Create new branches for your individual work. You can do this by either creating a new branch within GitHub (you **will need** to be registered as a contributor to do so), or by creating a new local branch and then pushing it to the remote repo (see *branch commands*)
* **If** you don’t want to work by creating branches then fork the repo and submit pull requests that way. If you opt for this route remember that you **must** **keep your fork up to date** to avoid conflicts.

*Cloning Commands*

* Clone the whole repo: *git clone* [*https://github.com/scottwares/chessapp*](https://github.com/scottwares/chessapp)
  + This is **not** recommended to avoid any unwanted merges into *master*
* Clone the *develop* branch: *git clone –b develop* [*https://github.com/scottwares/chessapp*](https://github.com/scottwares/chessapp)
  + This will clone the project into *<yourDirectory>/ChessApp* and create a local branch *develop*

*Branch Commands*

* Checking available local branches: *git branch*
* Creating new local branches: *git –b <branchName>*
* Changing branches: *git checkout <branchName>*
  + Note that you **must** have *committed* all changes in your current branch first (see *Committing Changes*)
* Pushing a local branch to remote: *git push <remoteName> <branchName>*
  + Unless you’ve changed it your *<remoteName>* is likely *origin*
  + I recommend creating your personal branches and pushing them to remote this way rather than creating a branch at the remote via GitHub since the branch names need to match
* Deleting a local branch: *git branch –d <branchName>*

*Committing Changes*

* When working in a local branch, any changes (new files, modifications, deletions etc) need to be *committed* to your branch.
* Commit: *git commit*
  + If you get an error aboutunstaged changes, see *Unstaged Changes*
* Committing **should** open a text editor for you to include a commit message
  + **If** for some reason Git decides to force you to do this in Vim in your command shell then @Ross in the group chat because he has lots of experience in this

*Unstaged Changes*

* You might not want to commit all of your changes. To achieve this, Git requires that you tell it exactly what changes you want to *commit*
* For simplicity, unless you’re perfectly comfortable with Git, use *git add –A*
  + This adds all the changes you’ve made since the last *commit* to this *commit*

*Pushing to Remote*

* To push your changes to remote: *git push*
  + You may need to log in
  + You may need to configure your branch to point to an upstream branch if it isn’t already (see *Configuring Upstream*)

*Pulling from Remote*

* To fetch most recent changes from your upstream branch: *git pull*
* To fetch metadata changes, such as new branches: *git fetch*
  + This **does not** retrieve file changes, only metadata changes. Use *pull* in most cases

*Configuring Upstream*

* Each of your local branches should point to a branch in remote
* For example, after cloning *develop*, your local *develop* branch will point to the remote *develop* branch
* If you choose to create a new branch via GitHub manually and then a new local branch to match it, you will need to point this to the remote branch
* Set current active branch to remote branch **of the same name**: *git branch –u <remoteName>/<remoteBranchName>*
  + As above, your *<remoteName>* is likely *origin* unless you’ve specifically changed it

*Merging Branches*

* Merging branches happens when you want to incorporate the changes of one branch into another
* For example, you finish working on your own personal branch, push them to remote and want to merge that into the remote *develop* branch
* To do this, go to GitHub and make a pull request with the base branch as develop and the merging branch as your remote branch
  + This will need to be reviewed by someone else
  + **DO NOT** approve your own pull requests
* You can merge local branches together using *git merge <branchName>*
  + This will merge the branch *<branchName>* into the branch that your **currently active branch**
  + **CHECK** that you are definitely merging the correct way around

*Example Flow*

This outlines an example of what your Git command flow might look like for cloning the project initially and getting started

1. Clone the *develop* branch from remote to local
   1. *git clone –b develop* [*https://github.com/scottwares/chessapp*](https://github.com/scottwares/chessapp)
2. Check that your local *develop* branch exists
   1. Change directory into *./chessapp*
   2. *git branch*
3. Create new local branch for work, and push it to remote
   1. *git push <remoteName> <branchName>*
4. Confirm your local branch is created
   1. *git branch*
5. Do some work
6. Commit your changes
   1. *git commit* 
      1. Set the unstaged changes if needed with *git add –A*
      2. If this is required then run *git commit* again after
7. Push your changes to your remote branch
   1. *git push*
8. Cycle on the turbo for 13 days
9. Come back to do more work
10. Pull any new changes from the remote branch
    1. *git pull*
11. Acquire any new changes from the remote *develop* branch
    1. Switch your local develop branch with *git checkout develop*
    2. Pull the changes with *git pull*
12. You **may** wish to merge your local *develop* branch into whatever local branch you’re currently working on so that it is up to date
    1. *git checkout <otherLocalWorkBranch>* to switch into your current working branch
    2. *git merge develop* to merge the changes in
    3. *git push* to push the changes of your local working branch to the remote branch
       1. You will need to *commit* before pushing