**Working with GitHub**

1. Install Git, https://git-scm.com/downloads
2. Open the git bash
3. Check your git version: type: git –version
4. Type: git config –global color.ui auto
   1. Give you color coding print out
5. git config –global core.autocrlf true
   * 1. to fix window’s conversion of LF to CRLF
6. Navigate to the folder you like to work with
7. Clone the repository of your choice using: git clone and URL of the repository

e.g. git clone https://github.com/.../someonerepo.git

git clone https://github.com/.../someonerepo.git mycopy

**GIT Commands**

1. git log -> print out all committed changed
   1. the print out has:
   2. commit id
   3. the Author
   4. The date of commit
   5. The message associated with the commit
   6. git log -n 1, which show only one commit
2. git diff -> compares two versions files or commits
   * 1. it has color coded comparison
     2. green lines, section or parts added to old file
     3. red lines, section or parts removed from the old file
     4. black lines, didn’t change
     5. without commit ID, will show changes to file blw working directory and staging area
     6. git diff –staged, will show the difference blw files in staging area and the repository
3. git log –stat -> print statistics about which files changed at each commit in the entire repository. Besides git log information, gives
   * 1. which files that changed
     2. total number of lines changed
     3. color coded comparison,
     4. green plus: for each added line in each file
     5. red minus: for each line removed from each file
     6. the statistics
        1. number of changed files
        2. total number of insertions (+) for entire repository
        3. total number of deletions (-) for entire repository
4. git checkout -> to revert to earlier version of your repository

**Creating, Editing and Modifying GIT Repository**

# https://github.com/github/training-kit/blob/master/downloads/github-git-cheat-sheet.pdf

1. cd into the directory of your choice
   1. ls -a
2. Initialize a git repository using: git init
3. git status
4. stage files for the repository using: git add file1
   1. git add file1
   2. git add file1 file2
   3. git add .
5. To remove a staged file use: git rm -cached file1
   1. git rm -cached file1
   2. git reset file1
6. Commit the changes to the repository using: git commit
   1. git commit
      1. this will open a text editor for your commit message
      2. Add you commit message as a command
7. Close the text editor, git commit done!
8. git commit -m “commit message”
   1. this will automatically add your commit message
   2. will not open the text editor
9. To undo last staging and commit use: git reset –hard
   * 1. git reset –hard
     2. don’t do often

**Git Branches**

1. git branch – show the branches
2. if not branches, will show the default \* master
   1. this is the current checkout branch
3. git branch yourbranch, will create a new branch named yourbranch
4. git branch will show
   1. yourbranch
   2. \* master
5. git checkout -b newbranch is actually two commands
   1. create a branch newbranch
   2. checkout to newbranch
6. git checkout yourbranch, to checkout to yourbranch
   1. git branch will now show
      1. \* yourbranch
      2. Master
      3. git status, will confirm the active branch
7. To see the commit history or diagram use
   1. git log –graph –online master yourbranch
   2. displays the commit diagram
8. Merge the branches using git merge
   1. Checkout on the master branch
   2. Run git merge master yourbranch
      1. git merge master yourbranch
      2. git merge yourbranch
         1. This run Auto-merging
         2. Using recursive strategy
         3. Prints the commit stats
   3. Sorts the commit by time
      1. Two close commits are not essential sequential
      2. Use git show commitID
         1. Show difference compared to parent
         2. Solved the recursive problem
   4. Clean closed branch using git branch -d yourbranch
      1. git branch -d yourbranch
      2. delete the branch labels
      3. the commitID will be reached through the master

**Git for collaborative work**

1. git remote
2. git remote add origin githubremoteUrl
3. git remote -v, (-v for verbose)
4. git push origin master
5. git clone githubremoteUrl
6. git branch -a
7. git pull origin master, is sequential combination of
   1. git fetch origin
   2. git merge master origin/master
8. git log origin/master
9. git diff origin/master master

**Git collaborative Branching**

1. create your own branch,
   1. git branch yourbranch22
2. checkout on yourbranch
   1. git checkout yourbranch22
3. make changes on file2 on yourbranch22
4. stage using -- git add file2
5. commit changes using -- git commit -m “change file2”
6. then push to remote: git push origin yourbranch22

**Git conflict resolution.**

1. Go to the file with the conflict
2. Update the file
3. Add the file using git add conflictfile
4. Then commit, using git commit
5. Push to remote using: git push origin master

**GitHub pull request**

1. Go the GitHub, checkout on the branch
2. Click pull request
3. Make sure the base fork is the master, use edit button for this
4. Add a header
5. Optionally write a comment
6. Click the create pull request button
7. GitHub sends email notification to your collaborator for pull request
   1. Your collaborator will review and comment on the changes
   2. They can merge the branches

***Updating Forked Repository***

* 1. Add the original repository as a remote in your clone, as upstream
     1. git remote add upstream originalRepoUrl
  2. pull the master branch from the original repository
     1. checkout into master branch – git checkout master
     2. git pull upstream master
  3. Merge the master branch into your local branch
     1. First checkout into local branch
        1. git checkout mylocalbranch
     2. git merge master mylocalbranch
     3. Resolve any conflicts
     4. Add file to the staging area: git add updatedFile
     5. Commit the staged file: git commit
  4. Push your local branch to your forked repository
     1. git push origin mylocalbranch
     2. Don’t push to the original repository, upstream!!
  5. Update your local master as well
     1. Checkout into master: git checkout master
     2. git push