**WORKING LOCALLY – LOCAL REPOSITORY**

1. inside your project folder

* **git init** // initializes a repository for the project inside the project folder .git
* initially the files/changes etc. inside that folder are not tracked

2. **git add filename** // moves this file to *staging* and makes it *tracked* which includes all of the changes made to that file as of this time

3. **git status** // lets you look at the status of tracked and untracked files and what’s in staging

4. **git commit** -m “message” // commits the staging to the local repository with a commit message for the log

5. Other **git add** commands

* **git add .** will add the entire directory
* **git** **add \*.java** would add all java files from the directory
* **.gitignore** file will tell git which files to exclude.   
  Create the **.gitignore** file in the directory using **touch .gitignore**edit the file in a text editor and add the files/directories you wish to ignore (\*.log), etc.
* Stage and commit the .gitignore file

6. **Branching** – if you wanted to work on adding a new feature to your code, make a branch of your main line

* **git branch MyBranch** // creates a new branch named MyBranch
* **git checkout MyBranch** // switches to MyBranch as working
* do your work, make your changes
* add your changes to the branch **git add .**
* commit the staging to the branch **git commit -m”new feature added”**
* switch back to main branch **git checkout master**
* **git merge MyBranch** //will merge the branch back to the master main branch

7. **Stash** – if you’re working in a branch and need to move to a different branch or back to main, but are not ready to commit your changes you can stash them and come back to them at another time

* **git stash** // stashes changes in all tracked files [if a file is untracked you should add that to staging manually before stashing **git add filename.** This makes your working directory clean so you can checkout a different directory
* **git stash apply** // reapplies your stash when you come back to complete your work on that directory

**REMOTE REPOSITORY**

1. **Clone the remote repository**:
   * on GitHub - from the project click the **Clone or download button** and copy the clone URL (example: <https://github.com/D2CTeam/D2CProject.git>)
   * from gitBash: **git clone** [**https://github.com/D2CTeam/D2CProject.git**](https://github.com/D2CTeam/D2CProject.git) // creates an exact copy of the remote repository along with entire commit history in its own folder
   * the remote repository is *origin* – from gitBash change to the new folder **git remote**

2. Now we have a local repository that we can work with – but we need to keep it up to date: by **fetching (pulling)** and **pushing**

3. **Fetching from the remote repository**: i

* **git fetch origin** // goes out to GitHub and gets any changes made since you last cloned or fetched. Fetching will bring changes into your local repository but will not merge it into your work. You have to merge manually.
* **git pull origin** // will automatically fetch and merge the changes from the remote branch into your current branch

4. When you are ready to submit your changes commit your work **git commit -a m ‘this will add and commit with a message at the same time’** // remember this is just committing changes to the *local repository*

5. **Push to the remote repository**

* **git push origin master** // we are telling Git to push our changes to the remote repository known to us as origin and commit them to the master branch