Git for the DSI quick reference

What you want to do	**`Git commands at command line`**
Clone a repo on Github locally	`git clone <url_from_github>`
Check to see what files are changed	`git status`
Show file differences that haven't been staged	`git diff`
See what branch you are on	`git branch`
Add a file to the staging area (for tracking)	`git add <filename>`
Remove a file from the staging area	`git reset <filename>`
Show file differences between staged and last commit	`git diff --staged`
Commit changes in staging area to a checkpoint	`git commit -m "<a short, descriptive commit message>"`
See where on Github you will push your commit	`git remote -v`
Push your commit to Github	`git push <name_of_remote> <name_of_branch>`
Pull your latest changes on Github down locally	`git pull <name_of_remote> <name_of_branch>`
For pair programming	
Assumes you and your partner are working on the master (default) branch	
After your partner has added you as a collaborator on his/her Github repository:	
Add your partner as a remote (for push/pulling)	`git remote add <their_name> <url_of_their_repo_on_Github>`
Pull their changes into your master branch	`git pull <their_name> master`
You do some work (you "drive" aka code)	
Add your changes to the staging area	`git add <filename_1> <filename_2>`
Commit changes in staging area to a checkpoint	`git commit -m "<a short, descriptive commit message>"`
Push your commit to your partner's remote	`git push <their_name> master`
Push your commit to your remote	`git push origin master`
Now your partner starts to drive (you navigate)	
First your partner pulls your changes down	`git pull origin master`
For group case studies	
Assumes you are following the Feature Branch Workflow	
https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow	
In this workflow ***you never work on the master branch*** *(for final code & code to be shared)*	
One of the group members (A) forks the Galvanize case study, and then clones it locally.	
On Github, A adds everyone as collaborators, and then they clone A's repository (not Galvanize's).	
Each member (incl. A) makes a branch named after them	`git branch <your_name>`
Now each member (incl. A) checks out their branch	`git checkout <your_name>`
Now everyone does their work	
You have some code you want to share with the group, so:	
Add your changes to the staging area	`git add <filename>`
Commit changes in staging area to a checkpoint	`git commit -m "<a short, descriptive commit message>"`
Push your commit (on your branch) to A's remote	`git push origin <your_name>`
Now A will pull down your branch, and merge it into master	
A gets on the master branch	`git checkout master`
A pulls down whatever is new on the master branch	`git pull origin master`
A verifies that your branch exists on the remote	`git branch -r`
A fetches your branch	`git fetch origin <your_name>:<your_name>  #  Github:local`
A verifies he/she is on the master branch	`git checkout master`
A merges your branch into master	`git merge <your_name>`
A pushes the master branch up to Github	`git push origin master`
Others can pull down the master branch to access your code	`git pull origin master`
To use it, you and they should make a branch off master	`git checkout master`
Delete the old branch (if all changes have been committed)	`git branch -d <your_name>`
Make a new branch	`git branch <your_name>`
Check it out (to work on it)	`git checkout <your_name>`
Fetching a remote branch	
Clone a repo on Github locally	`git clone <url_from_github>`
Fetch the remote branch	`git fetch origin <remote_branch>:<remote_branch>  # Github:local`
If you want to merge a branch into master (after you've fetched it above)	
Make sure you are on master	`git checkout master`
Merge the branch into the master branch	`git merge <branch_name>`
Delete the branch	`git branch -d <branch_name>`