GIT DOCUMENTATION  
with GitHub

<http://git-scm.com/download/win>

<http://cmder.net>

# Questions

Different user.name for different computers? To see from which computer I uploaded.

# Basic Commands

git init // Initialize local git repository

git add . // Adds every file to staging area

git add <file> // Add File(s) to staging area

git commit // Commit changes in index

git push // Push changes in staging area to remote repository

git pull // Pull any changes from remote repository

git clone // Clone repository into a new directory

Git config –global user.name “<yourname>” // Set your name  
Global = the settings for this user

Git config –global user.name // Display current name  
Git config user.name // Displays current name

Git config –list // displays a list of config settings

# Create a new repository

Create a new folder  
Git init // initialize git repository (creates .git folder in the repo)

# Staging files

Git status // shows which files have changed and which files are in the staging area

Git add <filename> // adds the file to the staging area  
Git add . // adds all changed files to the staging area  
Git rm –cached <file> // to unstage a file

# Commit

Git commit -m “info” // Committing file(s) with a descriptive message

Git log // display commit log  
Git log –oneline // displays commit log but just the id and descriptive message

## Undoing things (in order of danger)

**Checkout commit (read only?)**Git checkout <id> // Detaches and go to a specific commit (id)  
Git checkout master // goes back to master branch

**Revert commit**  **(changes)**Git revert <id> // Adds a new commit, that removes changes done on specified commit (id)

**Reset commit**  **(Resets, unable to undo)**Git reset <id> // Resets back to that commit id. But keeps changes in editor  
Git reset <id> --hard // Resets back to that commit id. Do not keep changes in editor (no going back)

# Branches

Git branch <NameOfBranch> // creates a branch   
Git checkout -b <NameOfBranch> // Creates a branch and switches to that branch

Git branch -a // Displays all current branches

Git checkout <NameOfBranch> // Switches to this branch

Add files to staging area and commit files as normal

Git branch -d <NameOfBranch> // Deletes branch if it is merged  
Git branch -D <NameOfBranch> // Deletes branch even if it is not merged

## Merging branches

Switch to master branch

Git merge <NameOfBranch> // Merges branch with master

# GitHub

## Local repo and push up to git hub

Git push <repo URL> <branch> // pushes local to remote

Always name the URL the same as <something> so you don’t have to remember the URL??

## No local repo, pull from git hub

# Lexicon

## Repository (aka Repo’s)

A repository is a place in which data is kept and maintained in an organized way. It can be local or remote.

## Branch, branches

A "branch" is an active line of development. The most recent commit on a branch is referred to as the tip of that branch. The tip of the branch is referenced by a branch head, which moves forward as additional development is done on the branch. A single Git repository can track an arbitrary number of branches, but your working tree is associated with just one of them (the "current" or "checked out" branch), and HEAD points to that branch.

## Commit

As a noun: A single point in the Git history; the entire history of a project is represented as a set of interrelated commits. The word "commit" is often used by Git in the same places other revision control systems use the words "revision" or "version". Also used as a short hand for commit object.

As a verb: The action of storing a new snapshot of the project’s state in the Git history, by creating a new commit representing the current state of the index and advancing HEAD to point at the new commit.

Pull

Stage area