Git Cheat Sheet

Overview

When you first setup Git, set up your user name and email address so your first commits record them properly.

```
git config --global user.name "My Name"
git config --global user.email "user@email.com"
```

About Git, GitHub and Heroku.
Git is a free & open source, distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

GitHub is the best way to collaborate around your code. Fork, send pull requests and manage all your public and private git repositories.

Heroku is a cloud application platform that supports a number of different programming languages including Java, Ruby, Node.js, and Clojure - it's a new way of building and deploying web apps.

Basic Git Workflow Example

Initialize a new git repository, then stage all the files in the directory and finally commit the initial snapshot.

```
$ git init
$ git add .
$ git commit -m 'initial commit'
```

Create a new branch named featureA, then check it out so it is the active branch. then edit and stage some files and finally commit the new snapshot.

```
$ git branch featureA
$ git checkout featureA
$ (edit files)
$ git add (files)
$ git commit -m 'add feature A'
```

Switch back to the master branch, reverting the featureA changes you just made, then edit some files and commit your new changes directly in the master branch context.

```
$ git checkout master
$ (edit files)
$ git commit -a -m 'change files'
```

Merge the featureA changes into the master branch context, combining all your work. Finally delete the featureA branch.

```
$ git merge featureA
$ git branch -d featureA
```

Setup & Init

Git configuration, and repository initialization & cloning.

`git config [key] [value]`	set a config value in this repository
`git config --global [key] [value]`	set a config value globally for this user
`git init`	initialize an existing directory as a Git repository
`git clone [url]`	clone a Git repository from a URL
`git help [command]`	get help on any Git command

Stage & Snapshot

Working with snapshots and the Git staging area.

`git status`	show the status of what is staged for your next commit and what is modified in your working directory
`git add [file]`	add a file as it looks now to your next commit (stage)
`git reset [file]`	reset the staging area for a file so the change is not in your next commit (unstage)
`git diff`	diff of what is changed but not staged
`git diff --staged`	diff of what is staged but not yet committed
`git commit`	commit your staged content as a new commit snapshot
`git rm [file]`	remove a file from your working directory and unstage
`git gui`	tcl/tk GUI program to make all of these commands simpler

Branch & Merge

Working with Git branches and the stash.

`git branch`	list your branches. a * will appear next to the currently active branch
`git branch [branch-name]`	create a new branch at the current commit
`git checkout [branch]`	switch to another branch and check it out into your working directory
`git checkout -b [branch]`	create a branch and immediately switch to it
`git merge [branch]`	merge another branch into your currently active one and record the merge as a commit
`git log`	show commit logs
`git stash`	stash away the currently uncommitted modifications in your working directory temporarily
`git stash apply`	re-apply the last stashed changes

Share & Update

Fetching, merging and working with updates from another repository.

`git remote add [alias] [url]`	add a git URL as an alias
`git fetch [alias]`	fetch down all the branches from that Git remote
`git merge [alias]/[branch]`	merge a branch on the server into your currently active branch to bring it up to date
`git push [alias] [branch]`	push the work on your branch to update that branch on the remote git repository
`git pull`	fetch from the URL tracked by the current branch and immediately try to merge in the tracked branch

Inspect & Compare

Examining logs, diffs and object information.

`git log`	show the commit history for the currently active branch
`git log branchB..branchA`	show the commits on branchA that are not on branchB
`git log --follow [file]`	show the commits that changed file, even across renames
`git diff branchB...branchA`	show the diff of what is in branchA that is not in branchB
`git show [SHA]`	show any object in Git in human-readable format
`gitx`	tcl/tk program to show the commit log in a GUI

Contributing on GitHub

To contribute to a project that is hosted on GitHub you can fork the project on github.com, then clone your fork locally, make a change, push back to GitHub and then send a pull request, which will email the maintainer.

fork project on github

```
$ git clone https://github.com/my-user/project
$ cd project
$ (edit files)
$ git add (files)
$ git commit -m 'Explain what I changed'
$ git push origin master
```

go to github and click ‘pull request’ button

Deploying to Heroku with Git

Use the heroku command-line tool to create an application and git remote:

```
$ heroku create
```

```
[Creating glowing-dusk-965... done, stack is bamboo-mri-1.9.2
http://glowing-dusk-965.heroku.com/ <http://glowing-dusk-965.
heroku.com/> | git@heroku.com:glowing-dusk-965.git <x-msg://536/
git@heroku.com:glowing-dusk-965.git> Git remote heroku added]
```

Use git to deploy the application.

```
$ git push heroku master
```

Create an additional Heroku app for staging, and name the git remote "staging".

```
$ heroku create my-staging-app --remote staging
```

Use git to deploy the application via the staging remote.

```
$ git push staging master
```

github
SOCIAL CODING

http://github.com

heroku™

http://heroku.com

082011