Workflow Components

Git Cheat Sheet

## Checkouts

### Get the latest code base

## Change to your desired parent directory, like C:/repos/ or ~/repos/. For DataShop servers, use /datashop/workflow\_components/

## git clone https://github.com/PSLCDataShop/WorkflowComponents /datashop/workflow\_components

### Check out a tag

You can’t really check out a tag in Git, since they can’t be moved around. If you want to put a version of your repository in your working directory that looks like a specific tag, you can create a new branch at a specific tag with git checkout -b [branchname] [tagname]:

$ git checkout -b version2 v2.0.0

Switched to a new branch 'version2'

Of course if you do this and do a commit, your version2 branch will be slightly different than your v2.0.0 tag since it will move forward with your new changes, so do be careful.

### Update your local repo

git pull

## Commit changes and push

### Adding or removing files

git add

git rm

### Check in modifications to versioned files

git commit –a

### Push changes to branch

git push <REMOTENAME> <BRANCHNAME>

The git push command takes two arguments:

* A remote name, for example, origin
* A branch name, for example, master

### Make last N commits disappear

git reset HEAD~N # where N is a positive int

### Redo last commit with new files

$ git commit --amend

As an example, if you commit and then realize you forgot to stage the changes in a file you wanted to add to this commit, you can do something like this:

$ git commit -m 'initial commit'

$ git add forgotten\_file

$ git commit --amend

You end up with a single commit – the second commit replaces the results of the first.

### Revert changes

$ git checkout -- CONTRIBUTING.md

$ git status

On branch master

Changes to be committed:

(use "git reset HEAD <file>..." to unstage)

renamed: README.md -> README

### 

## Branching

### List all changeset descriptions

$ git log

### List all changesets belonging to a specific file

git log net/ieee80211/ieee80211\_module.c

### List all branches

$ git branch

Rename a branch

git push <REMOTENAME> <LOCALBRANCHNAME>:<REMOTEBRANCHNAME>

e.g. git push origin master newname

Delete a remote branch or tag

git push <REMOTENAME> :<BRANCHNAME>

e.g. git push origin :v1.4

### View branch names in Git's log output with --decorate

git log --oneline --graph **--decorate**

### Make desired branch current in working directory

$ git checkout $branch

### Create a new branch, and make it current

$ git checkout -b my-new-branch-name master

### Examine which branch is current

$ git status

### Obtain a diff between current branch, and master branch

$ git diff master..HEAD

### Obtain a list of changes between current branch, and master branch

$ git log master..HEAD

(this is equivalent to **git log**, when used with HEAD)

### Merge changes from one branch into another

Let us suppose that you do work on branch *A* and branch *B*, and after work on those two branches is complete, you merge the work into mainline branch *M*.

$ git checkout M # switch to branch M

$ git merge A # merge A into M

$ git merge B # merge B into M

## Tags

### Download tags

git pull only downloads sha1-indexed object data, and the requested remote head. This misses updates to the .git/refs/tags/ and .git/refs/heads/ directories. For tags, run

git fetch --tags $URL

### List tags

$ git tag

v0.1

v0.2 …

### Show tag info

$ git show v1.2

tag v1.2

Tagger: Scott Chacon <schacon@gee-mail.com>

Date: Mon Feb 9 15:32:16 2009 -0800

version 1.2

commit 9fceb02d0ae598e95dc970b74767f19372d61af8

Author: Magnus Chacon <mchacon@gee-mail.com>

Date: Sun Apr 27 20:43:35 2008 -0700

### Release tagging - Use Annotated Tags

Annotated tags contain a full copy of repo with comments, identifying info, and md5 checksum.

Lightweight tags only store a checksum for a changeset and nothing more. I prefer we don't use them.

Annotated tag

git tag -a v1.0 -m "Workflow Components v9.x.y"

### Sharing Tags

You must explicitly push tags to a shared server after you have created them. This process is like sharing remote branches – you can run git push origin [tagname].

$ git push origin v1.0

Username for 'https://github.com':

Password for 'https://mkomisin@github.com':

Counting objects: 1, done.

Writing objects: 100% (1/1), 228 bytes | 0 bytes/s, done.

Total 1 (delta 0), reused 0 (delta 0)

To https://github.com/PSLCDataShop/WorkflowComponents

\* [new tag] v1.0 -> v1.0

If you have a lot of tags that you want to push up at once, you can also use the --tags option to the git push command. This will transfer all of your tags to the remote server that are not already there.

$ git push origin --tags

Now, when someone else clones or pulls from your repository, they will get all your tags as well.