2 ways to create a repo:

-start one from scratch

-fork an existing repo

Starting from scratch

Go to github and create new repo and initialize w/README

Create a local copy: make a dir to store the repo, then cd into that dir

Initialize a local git repo:

$git init

Point local repo at the remote repo create on github:

$git remote add origin <https://github.com/yourUserName/yourRepo.git>

Fork exisiting repo

<https://help.github.com/articles/fork-a-repo/>

Go to that repo on github. Click the Fork icon (you now have a copy of that repo on your github account)

Now make a local copy of that repo on your local machine:$git clone [https://github.com/yourUserName/repoYouWanted.git](https://github.com/yourUserName/yourRepo.git)

-This will clone the repo into your current directory

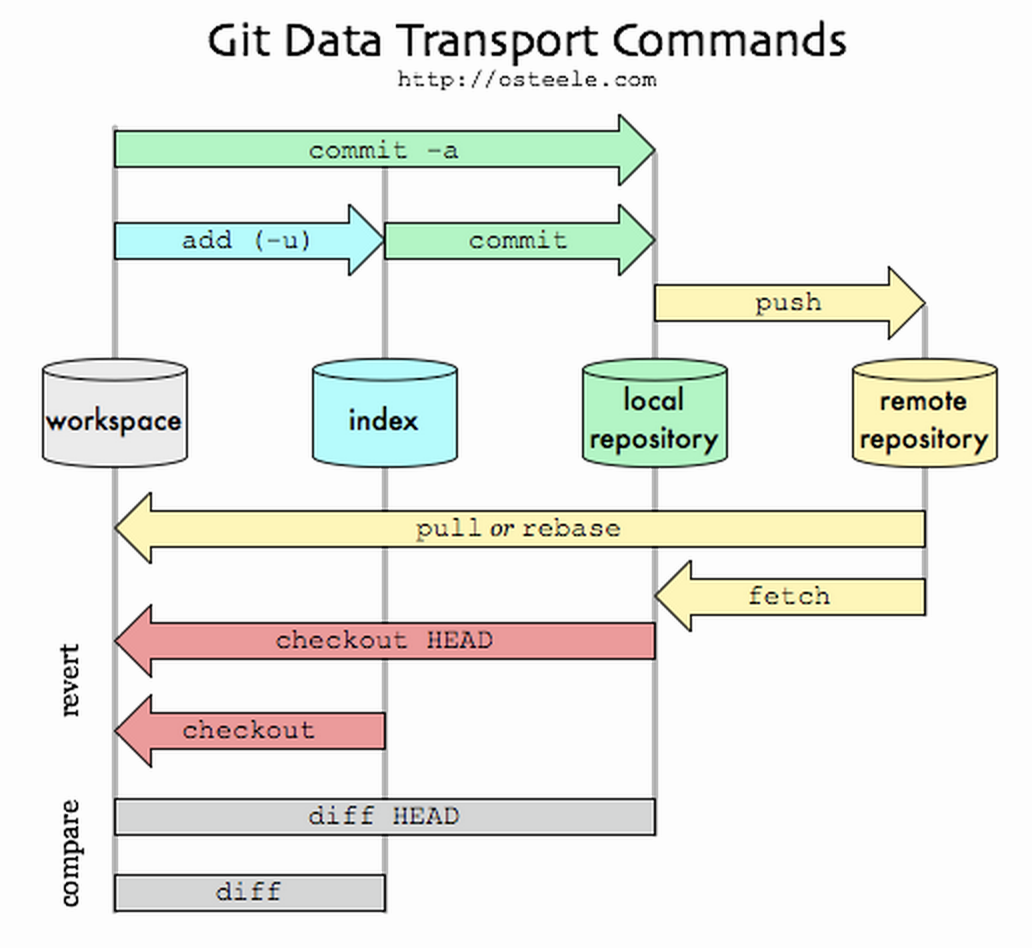
Great workflow overview:<http://blog.osteele.com/posts/2008/05/my-git-workflow/>

Workspace= directories on local that contain files you’re actually working on

Index/stage=tells git which files it should be controlling under version control

local repo=files that are stored/version controlled on local computer

remote repo=Generally github, but could be other



Add (adds to index), commit(commits changes to local repo), push(commits changes to remote repo)

-All commands executed from the working directory.

Adding(follow arrows from ‘workspace’):

$git add . //adds all new files from the workspace(current working dir) to the index where they can be tracked

$git add -u //updates tracking for files that changed names or were deleted

$git add -A <file>// does both of the above

-You should do this before committing to the local repository

Committing (to your local repo):

$git commit -m “useful message”

Pushing (to the remote repo):

$git push -u origin master (from working directory)

$git push <nameOfRemoteRepo> <branchNameInThatRepo>

Branches:

Sometimes you are working on a project with a version being used by many people. You may not want to edit that version.

While practices vary team by team, here’s 2 common scenarios to explain why you might want to use a branch;

1.The master branch is treated as the release branch, in between releases the only reason to edit the master is to fix bugs. Other branch(es) are development branches. This branch gets checkout and edited until it’s time to provide a new release. Then the dev branch is merged back into the master. and the process starts over.

2.The master branch is the version of the software that ALWAYS works. When you want to create a new feature, you create a branch to work on that feature. When dev of that feature is determined to be complete, merge that branch back into the master.

-A highbred of the 2 scenarios above could be having a release branch and and dev branch and then to additionally branch the dev branch to work on each new feature. When each feature is done, merge it back to dev branch. When it’s time for a new release, merge the dev branch back to the master.

-Branches never NEED to be push to remote repos, they can be kept completely local.

Also see:<http://stackoverflow.com/questions/3611256/forking-vs-branching-in-github>

Create a branch

$git checkout -b giveYourBranchThisName

See what branch you are on

$git branch

To switch back to the master branch

$git checkout master

Pull Requests:

Go to github and navigate to the branch that you’re working on.

Click icon to ‘compare and pull request’