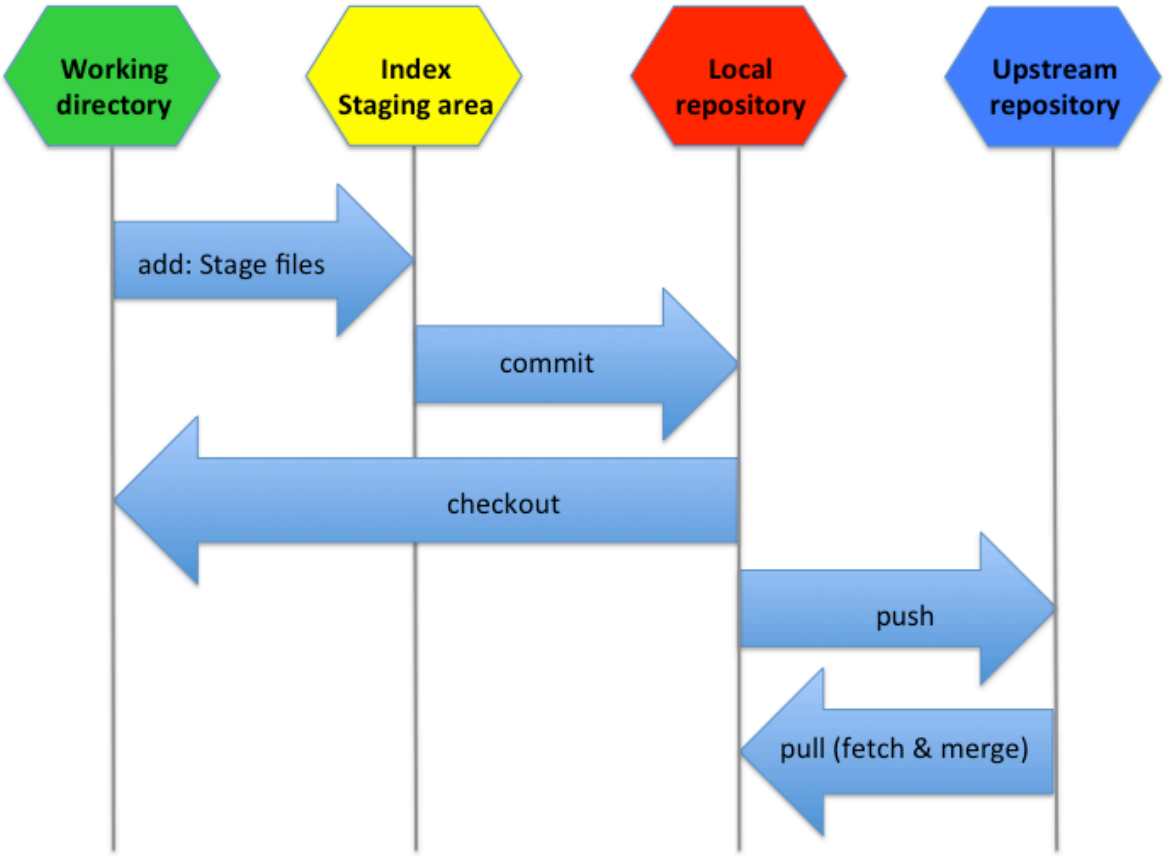
Introduction to Using GitHub

Overview



Definitions

Stash - place to hide modifications while working on something else
Workspace - local checkout

Index - staging area that holds a snapshot of the content of the working tree to be used in the next commmit
Working Tree
Local Repository - A subdirectory named .git that contains all of your repository files
Upstream Repository - version of your project hosted on the internet/server/network. Default is `origin`
branch - used to develop features isolated from each other. The *master* branch is the default branch. Use other branches for development and merge them back into the master when ready
master - master branch of repository
origin - default upstream repository name
HEAD - current commit object
HEAD~1 - parent of HEAD (HEAD~2 = grandparent of HEAD)
.gitignore - text file specifying contents of directory to ignore (ex. *.dmg,* .exe, *.Rhistory)
Fork - copy of someone else's repository. You can manipulate and keep for your own purposes or improve utility and send pull request to original user so they can incorporate changes into their repository

git commands
`git init` - Initializes (creates) a new local repository in the existing directory named .git
`git add <file1> <file2> <dir1>...` - adds the contents specified to the index, staging it for the next commit. `git clone <repo>` - download the repository specified by `<repo>` and checkout HEAD of the master branch (i.e. creates a copy of an existing project)
`git add .` to add all files in the current directory
`git rm <file>` - removes file from your index and working directory so no longer tracked
`git commit -m <"message">` - commits staged changes with a message explaining reason for the changes
`git status` - display the paths that have differences between the index file and the current HEAD commit, paths that have differences between the workspace and the index file, and paths in the workspace that are not tracked by git
`git branch` - list all the local branches
`git branch <branch>` - create a new branch named `<branch>`
`git checkout <branch>` - switch branches by updating the index and workspace to reflect the specified branch and updating HEAD to be the branch
`git fetch <remote>` - retrieves all branches from the remote repository (updates without merging)
`git merge <branch>` - merge branch into current branch
`git pull` - fetch changes from the server and merge them into the current branch (keep up to date with a remote repository)
`git push <remote> <branch>` - update the server with commits made to `<branch>` since the last push
`git reset HEAD <file>` - remove the specified files from the next commit
`git reset [--hard, --soft, --mixed] <remote>/<branch>`
`git diff` - compares changes to tracked files
`git stash` - temporarily saves changes you aren't ready to commit to yet
`git remote` - shows all of the remote versions of your repository

`git remote add <remote> <remote_URL>` - adds a remote repository to your git config. `git revert HEAD` - revert to the last commit
`git revert $id` - revert to a specific commit
`git cherry-pick [--edit] [-n] [-m parent-number] [-s] [-x] <commit>` - selectively merge a single commit from another local branch, Example: `git cherry-pick 1610a7130d9457f11a23423b3453m345h3456`
`git help <command>` - get help on `<command>`
`git log` - shows a listing of commits on a branch including the related details

More detailed list of commands and their use can be found at http://cheat.errtheblog.com/s/git

Other good information http://stackoverflow.com/questions/315911/git-for-beginners-the-definitive-practical-guide

Directions

This assumes you have set up git on your local machine. If you have not yet done so, you can find step-by-step instructions at https://help.github.com/articles/set-up-git

Initial Repository Setup

OPTION 1: If you do not yet have a folder or files on your local machine

1. Log into github.com
2. Create new repository (+ symbol in upper right)
3. Add the name of the repository, a README.md file (check the box "Initialize this repository with a README "), a .gitignore file (optional: allows you to have things that are not tracked, can be added later), a license, and a brief description of the repository
4. Decide if the repository will be public (free) or private (pay service)
5. Click "Create Repository"
6. Highlight and copy the the **HTTPS** clone URL link on the bottom of the right-hand column
7. Open your command line interface (Terminal app on Mac OS)
8. cd to your desired directory where you want to add (clone) your new folder
9. type `git clone https://URL-Copied-From-GitHub` (if you have the finder window open you can see new folder appear)
10. Now you can populate this local folder and push changes to the GitHub repository

OPTION 2: If you already have a folder on your computer that you want to create a GitHub repository for

1. Log into github.com
2. Create new repository (+ symbol in upper right)
3. Add the name of the repository, **DO NOT** check the box "Initialize this repository with a README ") or add a .gitignore file. It should be okay to add license information and a brief

description of the repository
4. You should add a README.md file to your folder. This can be done using any markdown or text editor or on the command line using `touch README.md`
5. Now use `git init` to initialize a git file within the folder (cd to the folder in the Terminal first)
6. Now use `git add README.me` to stage the README file
7. `git commit -m "initial commit" will commit the change
8. Highlight and copy the the **HTTPS** clone URL link on the bottom of the right-hand column of the GitHub repository you created
9. On the command line type `git remote add origin https://URL-Copied-From-GitHub` (if you have the finder window open you can see new folder appear)
10. `git push -u origin master` will push everything that has been committed to the GitHub repository

Push changes to folder to GitHub

1. Check what files have been changes with `git status` command
2. Add (stage) the files you want to change in the next commit with the `git add <file_name>` command or use `git add .` to add all
3. Commit to the changes that were staged using `git commit -m "Message about why you made the changes"`. Now it has saved your changes on your local machine
4. Push the changes on your local remote repository to GitHub using `git push origin master`. This means you are pushing your local remote repository named origin to the GitHub branch named master. You can use different local remote names and you can also work in and push to different branches on GitHub. You can use the `git remote` command to see what local remote you're working in (default is `origin`). Now if you check on GitHub.com you will find your files in the master branch updated to match your local files.

Using Git via GUI

RStudio has git functionality built into the projects option.

There is a nice little tutorial at Molecular Ecologist

SourceTree has nice visualization and is available for Windows and Mac.

SmartGit works on Windows, Mac, and Linux and is free for non-commercial use.

GitHub has it's own GUIs for Windows and for Mac. They must assume Linux users are comfortable enough with command line.