---

title: Git FUN!damentals
subtitle: Remote workflows
minutes:

The Origin

While git is useful to use locally, it is invaluable when there are lots of people contributing to the same project. Generally, when there are lots of people working on the same thing, it is good to have one authoritative source of the thing. In git, this is called the origin (but it could also be called dinosaur or banana, if you like). This origin is usually not in *YOUR* local filesystem (i.e. not in your home directory), but in a place where everyone can read from it, even if they can't write to it.

When you `clone` a repo, git automatically adds the source you cloned from as the remote origin. Let's all clone this repo:

```
$ git clone https://github.com/dlab-berkeley/git-fundamentals
```

Now when you type `ls` you should see the folder in your pwd. If you `cd` into `git-fundamentals` and type `ls` you will see all the information you've learned today. You can learn more about where the repo resides by typing:

```
git remote -v
```

And you should see an output that looks something like this:

```
origin  https://github.com/dlab-berkeley/git-fundamentals (fetch)
origin  https://github.com/dlab-berkeley/git-fundamentals (push)
```

You can get information about that remote with:

```
git remote show origin
```

Which, for me, yields this:

```
* remote origin
  Fetch URL: https://github.com/dlab-berkeley/git-fundamentals
  Push  URL: https://github.com/dlab-berkeley/git-fundamentals
  HEAD branch: master
  Remote branches:
    gh-pages tracked
    master   tracked
  Local branch configured for 'git pull':
    master merges with remote master
  Local ref configured for 'git push':
    master pushes to master (up to date)
```

You can add your own remotes to a project with `git remote add <alias> <url>`. For example, if we wanted to add our home directory as a remote for this repo, we could type:

```
git init ../test
git remote add test ../test
```

And now when we run `git remote -v`, we see:

```
test    /home/mylaptopusername/test/.git (fetch)
test    /home/mylaptopusername/test/.git (push)
origin  git@github.com:mygithubusername/git-fundamentals.git (fetch)
origin  git@github.com:mygithubusername/git-fundamentals.git (push)
```

This is useful if you have already initialized a git repository in your local directory that you want to live at a URL or server where other people are able to access it. To send our data to that remote, we'll push it. Try changing a file, adding it, committing it, and then typing:

```
git push
```

What happens? You should see:

```
remote: Invalid username or password.
fatal: Authentication failed for 'https://github.com/mygithubusername/git-
fundmantals.git/'
```

This is because if you don't specify the remote, git assumes that you mean `origin` (just like BASH assumes you mean pwd), and you don't have write access to my repository.

The general, best-practices workflow with a remotely hosted git repository looks like this:

1. Fetch
2. Merge
3. Branch
4. Modify
5. Commit
6. Merge
7. Push

Fetch means to retrieve data from a remote. Push means to send data to a remote. Usually, when you retrieve data from a remote, you let git retrieve the data and merge it into your local data automatically - this is done with the `pull` command, like:

```
git pull origin
```

`pull` is essentially equivalent to first `fetch` and then `merge`. You would use `fetch` and `merge` if you wanted to look at the changes other have made before changing your own files immediately. Otherwise `pull` does this all in one command.

After you have modified and committed your changes, you send the changes to the remote with:

```
git push
```

Since we haven't specified which remote, git assumes the origin, but sends us a fussy message:

```
warning: push.default is unset; its implicit value is changing in
Git 2.0 from 'matching' to 'simple'. To squelch this message
and maintain the current behavior after the default changes, use:

  git config --global push.default matching

To squelch this message and adopt the new behavior now, use:

  git config --global push.default simple

When push.default is set to 'matching', git will push local branches
to the remote branches that already exist with the same name.

In Git 2.0, Git will default to the more conservative 'simple'
behavior, which only pushes the current branch to the corresponding
remote branch that 'git pull' uses to update the current branch.

See 'git help config' and search for 'push.default' for further information.
(the 'simple' mode was introduced in Git 1.7.11. Use the similar mode
'current' instead of 'simple' if you sometimes use older versions of Git)

Counting objects: 7, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 477 bytes | 0 bytes/s, done.
Total 4 (delta 3), reused 0 (delta 0)
To git@github.com:dlab-berkeley/cornerstone-2015-unix-FUNdamentals.git
   0022eb6..fef3ee5  master -> master
```

It is very important to fetch and merge changes before you start modifying files, because files modified in serial can be merged automatically, but files modified in parallel often require human intervention (more on this later).