**Add A Remote Repository**

A remote repository is a repository that's just like the one you're using but it's just stored at a different location. To manage a remote repository, use the git remote command:

$ git remote

* It's possible to have links to multiple different remote repositories.
* A shortname is the name that's used to refer to a remote repository's location. Typically the location is a URL, but it could be a file path on the same computer.
* git remote add is used to add a connection to a new remote repository.
* git remote -v is used to see the details about a connection to a remote.

**Further Research**

* [Working with Remotes](https://git-scm.com/book/en/v2/Git-Basics-Working-with-Remotes#_showing_your_remotes) from the Git book
* [the git remote command](https://git-scm.com/docs/git-remote) from the Git docs

e.g.

$ **git remote add origin** <https://github.com/2046people0data0story/my-travel-plans.git>

**Push Changes to a Remote Repo**

The git push command is used to send commits from a local repository to a remote repository.

$ git push origin master

The git push command takes:

* the shortname of the remote repository you want to send commits to
* the name of the branch that has the commits you want to send

**Pulling Changes From A Remote**

If there are changes in a remote repository that you'd like to include in your local repository, then you want to *pull* in those changes. To do that with Git, you'd use the git pull command. You tell Git the shortname of the remote you want to get the changes from and then the branch that has the changes you want:

$ git pull origin master

When git pull is run, the following things happen:

* the commit(s) on the remote branch are copied to the local repository
* the local tracking branch (origin/master) is moved to point to the most recent commit
* the local tracking branch (origin/master) is merged into the local branch (master)

Also, changes can be manually added on GitHub (but this is not recommended, so don't do it).

**Pull vs. Fetch**

You can think of the git pull command as doing two things:

1. fetching remote changes (which adds the commits to the local repository and moves the tracking branch to point to them)
2. merging the local branch with the tracking branch

The git fetch command is just the first step. It just retrieves the commits and moves the tracking branch. It *does not* merge the local branch with the tracking branch. The same information provided to git pull is passed to git fetch:

* the shortname of the remote repository
* the branch with commits to retrieve

$ git fetch origin master

<https://github.com/2046people0data0story/course-collaboration-travel-plans.git>

**Reviewing Exiting Work**

The git log command is extremely powerful, and you can use it to discover a lot about a repository. But it can be especially helpful to discover information about a repository that you're collaborating on with others. You can use git log to:

* group commits by author with git shortlog
* $ git shortlog
* filter commits with the --author flag
* $ git log --author="Richard Kalehoff"
* filter commits using the --grep flag
* $ git log --grep="border radius issue in Safari"

grep is a complicated topic and you can find out more about it [here on the Wiki page](https://en.wikipedia.org/wiki/Grep) or in our [Shell Workshop course](https://www.udacity.com/course/shell-workshop--ud206).

**Determining What To Work On**

Before you start doing any work, make sure to look for the project's CONTRIBUTING.md file.

Next, it's a good idea to look at the GitHub issues for the project

* look at the existing issues to see if one is similar to the change you want to contribute
* if necessary create a new issue
* communicate the changes you'd like to make to the project maintainer in the issue

When you start developing, commit all of your work on a topic branch:

* do not work on the master branch
* make sure to give the topic branch clear, descriptive name

As a general best practice for writing commits:

* make frequent, smaller commits
* use clear and descriptive commit messages
* update the README file, if necessary