Git Usage Guide

Contents

1 Introduction to Version Control

At the most basic level, a version control system is used to help automate the following tasks:

- Store files and folders in a centralized master repository.
- Tracks changes made to files and files.
- Allows multiple people to edit the same set of files.
- Allows multiple people to share their changes with others.
- Allows you to determine who made a particular change.
- Allows you to roll back changes as needed.

Now, you might be thinking at this point: **That's all very nice, but what does it all really mean?_**

I'm so glad you asked, read on and I'll do my best to break it all down for you.

2 Introduction to Git

Git is a distributed version control system written by Linus (***the terrible***) Torvalds, the ***BDFL*** of the Linux operating system. Fortunately, Git isn't terrible at all. In fact it's one of the best version control systems ever.

While Git is quite powerful and provides many different capabilities (*some of which may be a little intimidating at first*), only a small subset of commands are actually required for working with OpenIndiana Docs.

Read on to learn more….

3 Git Vocabulary - Repositories

Before diving into the mechanics of what Git does, or even how to use it, let's review some basic Git vocabulary as it pertains to working with Git repositories.

Term	Definition
Upstream Repository	The ***Upstream*** repository is the repository under the direct control of the project. It contains the ***Golden Copy*** of all the files and folders held in Git version control. Nobody can make changes to this repository, unless the project allows it.
Fork	Before contributing to a project using Git, you first need to make a server side copy of the upstream repository. When you do this, you are ***Forking*** the upstream repository. This copy of the repository is under your control. When you submit changes, they are stored in this copy of the upstream repository.

Term	Definition
Clone	Before editing any files, you first need to ***Clone*** your forked copy of the upstream repository. Cloning creates a local working copy of the files and folders which resides on your computer. When you make changes, you are making changes to your cloned copy of the files and folders. You may locally clone the repository on as many computers as you wish.

4 Git Vocabulary - Commands

Now let's review some of the most commonly used Git commands and how you might use them.

Command	Description
`git clone`	In the previous section, we already mentioned a little about clones and what they are. Therefore, you would use the ***git clone*** command to create your editable local working copy of the files and folders.
`git status`	The ***git status*** command shows you whether any local files have been changed.
`git diff`	The ***git diff*** command shows you precisely what changed, for each local file which changed.
`git add`	The ***git add*** command lets you stage local files so their changes may be recorded locally by Git.
`git commit`	The ***git commit*** command allows you to locally record changes to staged local files and even include a nice little message describing your changes.
`git pull`	The ***git pull*** command allows you to pull in (merge) changes originating from a remote repository (either your fork or the upstream repository).
`git push`	The ***git push*** command lets you push locally recorded changes (your commits) up to your forked copy of the upstream repository.

5 Git Pull Requests

Now that we've explained several of the most basic Git commands, you might be left thinking:

Great, so how do changes which now reside in my forked copy of the repository, find their way to the upstream repository?

That's an excellent question; I am so happy you asked. The answer is of course, by submitting a ***pull request***.

When you submit a ***pull request*** you are ***requesting*** the differences found in your forked copy of the upstream repository to be ***pulled*** into the upstream repository itself. At this point other members of project will perform a quality assurance review of your changes. If your changes are accepted by the project, they will be pulled (merged) into the upstream repository.

6 Git Usage Scenario

Git does not limit you to working on a single computer. In fact, you can work with the very same set of files using multiple computers. Two different git commands make this possible.

- `git push`
- `git pull`

Sound pretty simple huh? Well that's because it is.

Here is a practical examples of how you might use Git:

6.1 Scenario 1

You're at home in the morning drinking your coffee, wearing your favorite bunny slippers, and busily working on the computer. Morning is your most creative time so you've made lots of changes to your files.

When finished, you do the following:

- View a list of changed files using `git status`.
- View the file differences using `git diff`.
- Locally stage changed files for commit using `git add <file name>` or using `git add .` to stage all changed files..
- Locally record your changes using `git commit -m <message>`.
- Safeguard your local commits by pushing them up to your forked copy of the upstream repository using the `git push` command.

6.2 Scenario 2

Now it's time to go to work and while on lunch break you want to make some more edits to your files. That's simple enough, perform the following steps.

If the local clone already exists:

- Execute the `git pull` command to pull in (merge) the changes you made earlier in the morning.

If the clone does not already exist:

- Pull down a local working copy of your forked upstream repository using the command: `git clone <URL of fork>`

Now let's throw a little twist to this scenario.

You have just received an email from another member of the project development team. In the email contains a note advising you of a completed (accepted and merged) pull request relating to the upstream repository. No problem, you'll simply do the following:

- Pull in (merge) those upstream changes into your local working copy using the command `git pull upstream master`.

Now your local working copy contains all the changes from both your fork as well the upstream repository.

When you're finished making changes, you'll safeguard those changes with another cycle of ***stage***, ***commit***, and ***push***. And whenever you're ready, submit a ***pull request*** to have the differences found in your fork pulled into the upstream repository.

Now that wasn't so bad was it?

7 Installing Git

Installing Git on OpenIndiana Hipster is simple.

```
pkg install git
```

After Git is installed, be sure to configure your name and email address.

For further details about configuring GIT, see: https://help.github.com/articles/set-up-git/

8 Further Reading

If you're really excited about Git and desire to become a Git Ninja, here is a great book to further your education: https://git-scm.com/book/en/v2

There is also the book Pro Git which is available for free under a Creative Commons license.