Chapter 3: The Master, the Origin, and a Remote

By Ryan Peters

Table of Contents

[Clarifying the Conventions of Git Terminology 25](#_Toc496525401)

[Recognizing the origin, origin/master, and master. 25](#_Toc496525402)

[The Origin is a remote repository, but origin/master is not. 25](#_Toc496525403)

[Command Descriptions 26](#_Toc496525404)

[Pulling from remote origin 26](#_Toc496525405)

[Committing, tagging, and pushing updates. 26](#_Toc496525406)

[References 27](#_Toc496525407)

This chapter contains information on the difference between your project’s master, the origin repository, and a remote repository. For as you saw in reading chapter 1, GitHub is built on Git’s Version Control System(VCS), so this chapter will dig a bit more into the underlying design and syntax of Git’s tools for managing remote repositories and code branching.

# Clarifying the Conventions of Git Terminology

To better understand the content in the rest of this chapter, and the rest of the book, you will want to solidify your understanding of Git’s naming conventions and technical terminology.

Recognizing the origin, origin/master, and master.

Implicit in the preceding heading is the fact that names like Origin and Origin Master are somehow different from one another. If you’ll recall from chapter 1, you were introduced to the idea of a repository, repo(s) for short, as your coding project’s workspace, and branches as snapshots of your code in that repository across stages of development. (Git - Documentation)

For the purposes of clarifying how origin, origin/master, and master are distinct from one another we need to introduce the idea of local and remote repos. Where remote vs. local describes the repo’s location relative to the computer you are working on. It naturally follows that if you have local and remote repos, you will also have local and remote branches.

The Origin is a remote repository, but origin/master is not.

In your coding experience, outside of working with Git or GitHub, you have undoubtedly come across the issue of defining the scope for preventing naming conflicts between your numerous variables and functions. In Java this issue is handled by using packages, and in the c-languages it is handled with namespace; in both instances, the structure of the solution is essentially a folder/file directory. The same is true in Git, where you can think of repositories as folders in a file system. (Git - Documentation)

Each folder, or repository in this case, has its own name. Under that repository the programmer is free to name files and sub-folders however they like. Though, note that creating a sub-folder will not create a sub-repository, as there is no need or reason for such a structure.

Considering this perspective, we are now safe to define the Git origin as a remote repository that you cloned to create your local repository. The origin/master in turn, is simply the conventional folder-like name for your clone of origin’s master branch.

# Command Descriptions

To understand the differences between local and remote repositories, it’s helpful to practice interacting with them. The following example scenarios describe the use of git to accomplish simple version control tasks. To see samples of the Git commands mentioned bellow, please see Table 1.

These scenarios are written under the hypothetical context that you are one of many programmers contributing to a software utilities library that is being centrally hosted on a private server.

SSH keys and access privileges have already been handled, so all you need to worry about is entering you Git commands into the Bash terminal. You will need to fetch a clone of origin’s master branch, merge that clone into your local repo’s branch, Git-commit your updates with version annotations, and finally push that commit back to the origin for confirmation by the team.

## Pulling from remote origin

Your first steps will be entering Git’s fetch and merge commands into the terminal. Remember, the origin is not the actual name of a specific remote repo, it’s how your local workstation refers to any remote repo it fetches from or pushes to.

As a side note, the pull command wraps the process of fetching and merging into a one step process for you, the user. However, you should delay in adopting Pull as it will obstruct your learning how to properly set up a remote reference. As a result, you will not have the foundational understanding needed to trouble shoot future problems in your VCS.

## Committing, tagging, and pushing updates.

A frequent beginner mistake is to overlook the importance of properly tagging updates. According to the official Git documentation, “[Git] like most VCSs, has the ability to tag specific points in history as being important. Typically people use this functionality to mark release points”(Git - Tagging).

The good news is that tagging is easy to do on a fundamental level, see the samples in Table 1. Aft

#### Table 1: Quick reference for Git commands

|  |  |  |
| --- | --- | --- |
| Command name | Purpose | Examples |
| Fetch | Retrieves a clone of the master branch stored on a target remote origin repository. | git fetch origin |
| Retrieve branch clones from multiple repositories. | git fetch –multiple[<params>] remote\_name1 remote\_name2 |
| You can configure Git retain lists of multiple remote repositories, such as all those of your team members.   Using this command allows you to fetch branch clones from all of them at once. Useful for synchronizing a decentralized array of workstations. | git fetch –all [<params>] |
| Merge | Joins the clone branch origin/master into your local master branch | git merge origin/master master |
| If you are prompted for action where there is a version history conflict between branches, and **you** **don’t** **want to resolve the issue yet.** | git merge –abort |
| If you are prompted for action where there is a version history conflict between branches, and **you** **want to resolve the issue using default resolution settings.** | git merge –continue |
| Push | Update remote references along with associated objects. | git push origin |
| This will push any staged commits you have to the master branch of the remote origin repository.  This will automatically initiate a merge command with a default conflict resolution approach. | git push origin master |
| Tag | Produces a list of available tags in alphabetical order. | git tag |
| Adds your chosen tag to the commit, what you’ve entered in place of the <>, thus making your commit searchable and referenceable by others. | git tag <your chosen tag> |
| Will search the available tags for any that match your provided search pattern. Note, this pattern needs to go inside of the quotations, sans <>. | git tag -l "< your string pattern to search for>" |
| Produces an annotated tag which others can later use to gain insight into the details of your commit. | git tag -a v1.4 -m "<your brief comment or description>" |
| https://git-scm.com/docs/git-fetch  https://git-scm.com/docs/git-merge  https://git-scm.com/docs/git-push | | |
| * A note on the use of this table –The contents of this table are in alphabetical order, and in no way do they represent the order in which commands should be used. The time and place for using a command is subject to the context you are working in. * Pay attention to white space in the commands, as scripting languages in the Bash command terminal don’t use comma delimiters between their parameters. * Arguments appearing inside of diamond brackets <> indicate documentation descriptions of what should normally belong at that point in the line of code. * Square brackets [ ] are used in to denote optional arguments in an example, so you may omit that argument if it suits your needs. | | |

# References

Git - Documentation. https://git-scm.com/doc. Accessed 16 Oct. 2017.

Git - Git-Fetch Documentation. https://git-scm.com/docs/git-fetch. Accessed 16 Oct. 2017.

Git - Git-Merge Documentation. https://git-scm.com/docs/git-merge. Accessed 16 Oct. 2017.

Git - Git-Push Documentation. https://git-scm.com/docs/git-push. Accessed 16 Oct. 2017.

Git - Tagging. https://git-scm.com/book/en/v2/Git-Basics-Tagging. Accessed 16 Oct. 2017.

**[word count 734]**