shutterstock.com/ venimo

Collaborating and Contributing in GitHub

By NICKY BLEIEL | *Associate Fellow*

GITHUB (*http://github.com*) is a Web-based repository for software projects and is reportedly the world's largest open source community, hosting over 35 million repositories that include both code and the documentation for that code. It includes version control, project management, and social features that technical communicators can use to manage and document software projects.

Technical communicators can collaborate and contribute in GitHub in a variety of ways, including: writing content and managing issues; commenting on, reviewing, and merging proposed changes; documenting GitHub projects; and managing the wiki community.

GitHub is also an excellent choice for version control of documentation files and management of documentation issues. Docs aren't siloed in GitHub—they live with the code, follow the same workflow as the code, and are reviewed with the code—which is especially useful in an Agile environment, where docs are part of the "definition of done."

GitHub projects can be public, private, or hosted internally behind your company firewall on GitHub Enterprise. While the features and options are the same for all, your responsibilities can vary for each. Company projects hosted on public GitHub need customer-facing documentation and possibly community monitoring. Projects hosted behind a firewall in GitHub Enterprise need internal documentation, issue review and tracking, and project file management. When contributing to public GitHub projects, you can pick and choose what you want to do—write or edit the documentation, or contribute code, graphics, or comments.

It All Starts with Git

GitHub uses Git as a repository. Git (a command-line tool initially released in 2005) is a distributed, nonlinear version control system for software development by Linus Torvalds, creator of Linux. Git has wide adoption; according to a survey released in May 2014 by the Eclipse Foundation, Git is being used by one third of software developers and is the #1 code management tool.

Why GitHub?

GitHub could be considered a "one-stop-development-shop" because, in addition to version control, it includes issue tracking (bugs and feature requests), notifications, diffs, status dashboards, and documentation. It also has social features. You can "follow" other contributors and GitHub will automatically alert you of their activities. You can also "watch" specific projects—you will receive notices about those also. If you prefer to tag a repository, but skip the notifications, you can "star" it. You can then go to your "stars page" to catch up on those projects, as well as take a look at the "stars" of your friends.

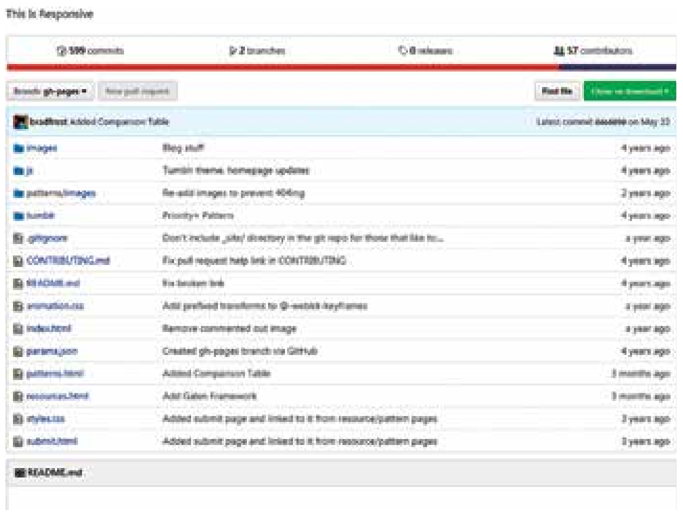
One of the main reasons GitHub has become so popular is that it has simplified the process of contributing to open source projects. According to Gregg Pollack of Code School (in the article "What Exactly Is GitHub Anyway?"), before GitHub, contributing to a project meant a developer had to download the source code, make changes, create a patch, then email that patch to be evaluated. With GitHub, you can create a local copy (a "fork") of the source code, make your changes, and submit a request for consideration (a "pull" request). And there is a public record of all of this, so you can build a reputation in the community.

Organizations can post projects on the public version of GitHub for free, as well as purchase GitHub Enterprise for their internal use behind a firewall. Individuals can create public repositories and contribute to projects with a free account, but must pay a fee to create private repositories that are accessible only to them and their invited collaborators.

GitHub Terminology

Following is a list of selected GitHub terminology from the GitHub Glossary (*https://help.github.com/articles/github-glossary/*):

- **Repository** (also referred to as a "repo"): A repository is the most basic element of GitHub. They're easiest to imagine as a project's folder. A repository contains all of the project files (including documentation), and stores each file's revision history.
- **Fork**: A fork is a personal copy of another user's repository that lives on your account. Forks allow you to freely make changes to a project without affecting the original.
- **Pull request**: Pull requests are proposed changes to a repository submitted by a user and accepted or rejected by a repository's collaborators.



This Is Responsive

599 commits | 2 branches | 0 releases | 57 contributors

Branch: gh-pages ▾ | New pull request | Find file | Clone or download ▾

bradfrost Added Comparison Table | Latest commit [illegible] on May 23

images	Blog stuff	4 years ago
js	Tumblr theme, homepage updates	4 years ago
patterns/images	Re-add images to prevent 404ing	2 years ago
tumblr	Priority+ Pattern	4 years ago
.gitignore	Don't include _site/ directory in the git repo for those that like to...	a year ago
CONTRIBUTING.md	Fix pull request help link in CONTRIBUTING	4 years ago
README.md	Fix broken link	4 years ago
animation.css	Add prefixed transforms to @-webkit-keyframes	a year ago
index.html	Remove commented out image	a year ago
params.json	Created gh-pages branch via GitHub	4 years ago
patterns.html	Added Comparison Table	3 months ago
resources.html	Add Galen Framework	3 months ago
styles.css	Added submit page and linked to it from resource/pattern pages	3 years ago
submit.html	Added submit page and linked to it from resource/pattern pages	3 years ago

README.md

Figure 1. A Sample GitHub Repository

- **Merge**: Merging takes the changes from one branch (in the same repository or from a fork), and applies them into another.
- **Collaborator***:* A collaborator is a person with read and write access to a repository who has been invited to contribute by the repository owner.
- **Contributor***:* A contributor is someone who has contributed to a project by having a pull request merged but does not have collaborator access.

GitHub Documentation Options

There are three options to document a GitHub project: READMEs, GitHub Pages, and a Wiki. Public GitHub projects may use all three to explain and publicize the project, as well as build community. An internal GitHub Enterprise project could include more than one README (for example, one for development plans and one for notes on final customer-facing docs); the Wiki could document processes (or not be used); or GitHub Pages may not be used at all.

READMEs

When you create a project in GitHub, a README file is created automatically. Unlike most READMEs, GitHub readmes are front and center and are the "hub" of the repository. You can create more than one if you need it, but the one named README.md will be displayed by default when the project is opened.

*Figure 2. The Octocat is the official mascot of GitHub (*https://github.com/cameronmcfee*). The Ordered Listocat, a variation from the Octodex, is by Cameron McEfee (*https://octodex.github.com/*).*

GitHub could be considered a “one-stop-development-shop.”

READMEs are edited in Markdown, and have the file extension of .md. Markdown is a lightweight markup language that can be converted to HTML easily. If you have used wiki syntax, it will look very familiar. Markdown was originally developed in 2004 by John Gruber and has splintered into different variations. GitHub uses “GitHub Flavored Markdown,” which has features such as syntax highlighting, task lists, tables, and @mentions (a quick way to notify someone that you need them to take a look at something). Markdown is also used for GitHub Wikis, GitHub Pages, for commenting, and when creating sharable reusable snippets, called Gists.

In addition to contributing content to the readme, any proposed changes need to be reviewed, commented on, and merged. You can also review the READMEs in public repositories and propose changes to them. (This README checklist on GitHub by Daniel Beck is a useful guide to writing quality READMEs: *https://github.com/ddbeck/readme-checklist/blob/master/checklist.md*.)

Wikis

You can expand upon the information in the README by adding wiki pages to your project repository. Wiki pages are easy to create, and can be edited in Markdown or one of the other eight edit modes. By default, anyone can edit a GitHub project wiki, but you can change the settings and make your wiki read-only. If your project wiki is public, contributions from the community will need to be monitored. To learn how to setup a wiki in GitHub, see the Mastering Wikis tutorial in GitHub Guides at *https://guides.github.com/features/wikis/*. (Examples of GitHub wikis can be found online at *https://github.com/showcases/projects-with-great-wikis*.)

As a **member,** you get the **tools you need** to be **successful**.

- Education and Training
- Job Bank
- Award-Winning Publications
- Affinity Program
- Networking
- Chapters and Special Interest Groups
- Annual Conference
- Body of Knowledge
- Discounts and Special Offers

STC Society for Technical Communication

www.stc.org

GitHub Pages

Another way to provide documentation for your project is to use GitHub Pages. These are Web pages hosted and published on GitHub. They are authored in Markdown, and you can use GitHub-provided themes to create a custom look. You can even add your Google Analytics tracking ID to each of your Pages. After you are satisfied with your content and the theme, you can publish your GitHub Pages and the default URL will be: *http://[accountName].github.io/[repoName]*. GitHub Pages are always public, even if your repository is private. (Examples of GitHub Pages can be found online at *https://github.com/showcases/github-pages-examples*.)

Using GitHub for Version Control

GitHub has, at its core, the Git version control system. Files are stored in Git repositories, and Git is a powerful tool. If you have used Git for version control before, that knowledge will be useful when working in GitHub. (A quick reference to Git commands can be found online at *https://git-scm.com/docs*.)

If your company is using GitHub Enterprise, it makes sense to use it for the version control of your documentation files. And storing documentation files along with code and other project artifacts means that docs are not siloed, can follow the same workflow as code, and that the project’s collaborators can review and contribute.

There are some things you need to keep in mind when using version control in GitHub. First, you need to determine what workflow you will use. In public GitHub, it is fairly straightforward: you “fork” (make a copy on your account) of the project, clone or download a local copy of the project to your machine, make your changes, commit (“push”) them on your fork, and then submit a request for consideration (a “pull request”). The project owners and the community can comment on the changes, and the owners can merge them if they approve. You can do all of this in the GitHub graphical user interface (GUI), but you can also work strictly on the command-line using Git commands, or with a mixture of both.

In GitHub Enterprise, the workflow will be similar, but since you will be a collaborator, you can upload the documentation files to the “master” branch, and create new branches to make your revisions in. (In GitHub, the “master” branch is defined as the one that can be

One of the main reasons GitHub has become so popular is that it has simplified the process of contributing to open source projects.

released at any given time, but could also be considered the "original" version that you are refining incrementally on branches.) Your company may develop guidelines you need to follow around the naming of branches and who should do reviews and merges. You should be involved in the development, and, of course, the documentation of these guidelines. (See *https://help.github.com/articles/what-is-a-good-git-workflow/* for an overview of Git workflows.)

In GitHub, any file that can be read with a text editor can be opened and edited. When working with Markdown files and many other file types, editing and reviewing can be done right in the GitHub GUI. But for some files (like .dita files), you may prefer to use your XML editing tool. And binary files (images, Word files, etc.) can't be opened within GitHub at all. This is part of the reason why working on a local copy ("clone") of your GitHub project is a best practice. On your local copy, you can open any file in the application you wish.

Managing Documentation Issues in GitHub

You can manage documentation issues (bug and feature requests) in your project using GitHub's issue tracking. If you have used other issue tracking systems, many of the features will be familiar, and as with any system, you need to think about how granular issues should be, how you want to tag them, and what best practices your team should follow. GitHub issues can be authored in Markdown, so you can add formatting, create task lists to track progress, use emojis, and more.

It is important to create all the Labels (such as "Documentation") and Milestones you need so that you can tag the issues in your project properly. Every issue should be tagged with one or more labels, a milestone, and an assignee (owner). It is easy to find issues using the Filters and Search, and you can click on any label or milestone to find all issues in a project that contain those tags. You can also search for issues across repositories, see *https://help.github.com/articles/searching-issues/* for details. If you need to break a larger documentation project down into subprojects, you can create Epics and assign issues to them. To reference another issue in your project, enter "#" in the description or a comment and the entire list of issues for that project will display. The chosen issue will become a link you can use to navigate between the issues.

After an issue is created, anyone with Collaborator status in that project can comment on it. If you would like a specific team member to comment on an issue, use an @ mention in the issue description or a comment. As soon as you enter "@", the list of collaborators will display and you can choose one.

A fun GitHub convention you may want to adopt is including the "Ship It Squirrel" emoji (:shipit:) in a comment if you believe the proposed change is ready to ship.

To view and manage your issues on a virtual task board (which is great for agile shops and virtual teams), check out ZenHub, a productivity tool that can be integrated with GitHub. (More information about GitHub issues can be found in the Mastering Issues tutorial in GitHub Guides online at *https://guides.github.com/features/issues/*.)

Integrations with Other Tools

In addition to ZenHub, over 70 productivity tools can be integrated with GitHub, including Slack, a team messaging and collaboration application. If you would prefer a Git GUI client instead of working at the command line to clone, commit, push, etc., check out GitKraken and GitHub Desktop. GitBook can be used to host, write, and publish documentation (outputs include PDF, ePub, mobi or a website). (See the complete list of tools online at *https://github.com/integrations*.)

Other Uses for GitHub

GitHub's origins are in software collaboration, but it can also be used to collaborate on other types of projects, and is a great place to post your portfolio or class projects. According to the article "From Collaborative Coding to Wedding Invitations: GitHub Is Going Mainstream," it is also being used to collaborate on projects as diverse as Gregorian chants, licensing agreements, and wedding invitations.

NICKY BLEIEL (nableiel@us.ibm.com) *is a Watson Information Developer at IBM and a GitHub user. She is a Past President and Associate Fellow of the Society for Technical Communication and has over 20 years of experience writing and designing content for software products in a variety of industries. She is a popular speaker at many conferences, including the STC Summit, WritersUA, tcworld, CIDM, and LavaCon; and has been published in STC's* Intercom, tcworld magazine, ISTC Communicator, *and more. Learn more about her at* nickybleiel.com.

REFERENCES

Finley, Klint. "What Exactly Is GitHub Anyway?" *TechCrunch* (14 July 2012). *http://techcrunch.com/2012/07/14/what-exactly-is-github-anyway/*.

McMillan, R. "From Collaborative Coding to Wedding Invitations: GitHub Is Going Mainstream," Wired (2 September 2013). *www.wired.com/2013/09/github-for-anything/*.

Skeritt, Ian. Eclipse Community Survey 2014 Results (2014). *https://ians.kerrett.wordpress.com/2014/06/23/eclipse-community-survey-2014-results/*.