GitHub 101 for Government

By Bryan Hirsch, Solutions Architect at Acquia

Table of Contents

Introduction

Imagine this scenario: Your government agency has just adopted open source with Drupal. As you look ahead to this re-platforming project and how you'll manage it, you wonder where to start. Someone you respect suggests creating an account on GitHub.com, but you don't really understand what GitHub is. You want to know why you need it, what you should do with it, what new communications channels it opens up, and how to steer clear of landmines. You don't want unapproved spokespeople communicating with the public on your agency's behalf or putting out any sort of communications that could embarrass or confuse anyone. And you don't want to get into anything without understanding all of the security and cost implications.

If this scenario sounds familiar to you, I have some basic background and recommendations for you to get started with GitHub for Government.

In the coming pages, we'll discuss what the public can see and do with GitHub, what Drupal code should and shouldn't be in a public repository, users and permissions on organizational accounts, and developer workflows. I hope you find this useful.

Bryan Hirsch is an Acquia Solutions Architect. He works with federal, state, and local government agencies to craft Drupal-based solutions to digital problems. Bryan has led agencies' tech teams through building and deploying new web applications, open-sourcing applications to make them shareable across agencies, releasing data through web APIs, managing high traffic events, and executing plans for full application lifecycle management. He has also helped shape and launch Acquia projects including: Site Factory, Shield, Edge CDN, and Edge Protect.

What GitHub Does

GitHub is a user-friendly website that helps people and organizations share code and collaborate on projects, like a social network oriented around code. The site's main feature is hosted Git code repositories, like a "track changes" feature for software projects. This lets developers fully understand changes in code over time and improves their ability to use released code. Organizations hosting code on GitHub can make their projects public or private. When releasing software to the public, a repository must be made public for others to view, download, copy, and comment on the project's code and history of changes.

The sections that follow include a GitHub crash course for nontechnical audiences, screenshots so you can see what these things look like, and recommendations for getting started. You'll learn about:

- What the public can see and do with what you put on GitHub
- What Drupal code should (not) be included in a public repository
- Users and permissions and creating an organization in GitHub
- Developer workflows

What the Public Can See and Do with the Information You Put on GitHub

When you make a GitHub code repository (or repo) public, certain pieces of information and code become visible, and available for interaction. Code becomes downloadable and publicly viewable online. Visitors can browse through folders, click files, and view code through their web browser.

The code's change history (the git "log") also becomes public. This is like a "track changes" history. For every saved change, Git keeps track of the changes made, the user who made the change, the date the change was made, and a message logged by the user. These changes can be viewed online and in copies of the source code. Here are a few examples:

- A list of logged changes, authors, and messages viewable online on GitHub
- A single change, its author, and the author's message on GitHub
- A list of changes in a visitor's copy of the repository
- A single change in a visitor's copy of the repository
- All past releases online and in a visitor's copy of the code base

Although code maintainers have no obligation to respond to public input, public repositories on GitHub do open a few new channels for the public to interact with a project:

- GitHub users can submit comments about saved changes, also known as "commits".
- GitHub users can submit proposed changes to code as "pull requests" for the project maintainers to consider "pulling" into the main codebase.
- GitHub users can copy ("fork") and maintain their own versions of any project they have access to.

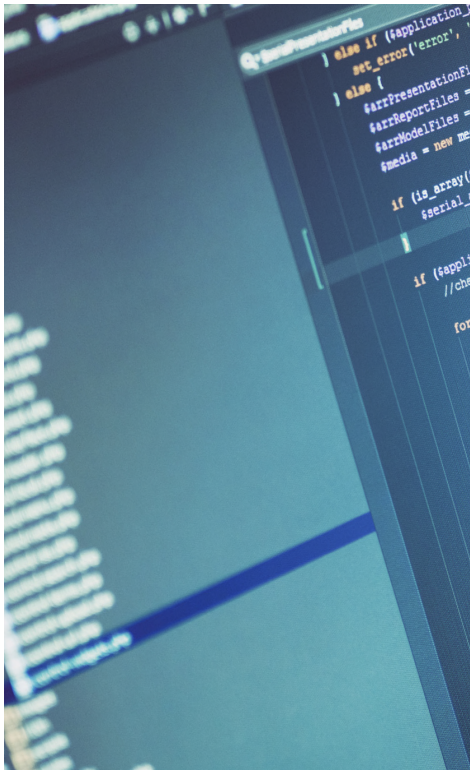
Git keeps track of the changes made, the user who made the change, the date the change was made, and a message logged by the user.

How to Use Public Repositories for Your Agency

Your government agency should use an organizational GitHub account to publicly host open source code. GitHub is the easiest and most user-friendly way to release code to the public.

If you are interested in “open sourcing” your code, but people in your organization are wary of creating new, unknown, or unsupervised channels of communication with the public, disable the Wikis and Issues features. To the public, releases on GitHub will seem like a normal but “bare bones” use of the software.

Making code public for “bare bones” use is perfectly respectable. Just because a repository is public, the repository owner is not obligated to communicate with the GitHub community or manage development in public view. Linus Torvalds, the creator of Linux and Git, uses GitHub to host Linux, but he doesn’t publicly manage development there. Similarly, a mirror of Drupal’s authoritative git repository is on GitHub, but the Wiki and Issue trackers are disabled. Discussion, collaboration, and issue tracking for these projects happen elsewhere. For your agency, you can decide on a project-by-project basis if your roadmap and issue tracker should be private or public and take place on GitHub or elsewhere.



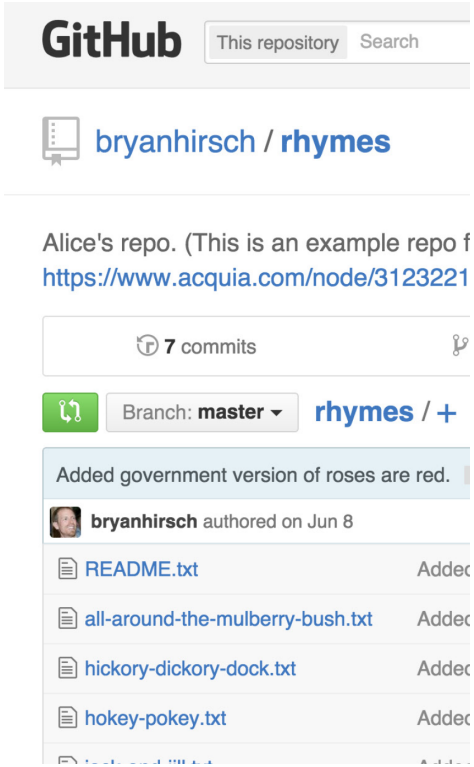
For your agency, you can decide on a project-by-project basis if your roadmap and issue tracker should be private or public, and if they’ll take place on GitHub or elsewhere.

What Drupal Code Should/Should Not Be Included in a Public Repository (Public Repo)

What code can be safely made public and what code should be made public are not always the same. What custom code should be public depends on what your agency's objectives are in releasing your source code. (If you know you're expected to open source your code, but you're not sure what objectives are driving the decision, I recommend finding out. That knowledge can help you avoid a lot of headache and confusion down the road.)

Possible objectives for releasing source code to the public include:

- Complying with an organizational source code policy that mandates "open source by default"
- Releasing code as a matter of transparency, so the public knows what it's paying for
- Releasing code with author names in the commit history to give developers a sense of pride, ownership, and peer pressure to do their best work
- Providing code as an example or starting point for other agencies solving similar problems, whether the code is reusable or not
- Cultivating a community of users and contributors so your agency is not paying the total cost of ownership
- Encouraging other agencies to reuse your code, so you're not using taxpayer dollars to solve the same problems over and over again



The following lists describe the types of code that power Drupal-based web applications. Keep your agency's particular objectives in mind, and this information will help clarify what you should release in your own particular situation.

Code in a Drupal application and the types of projects that people contribute on drupal.org can be shared. If your objective is reuse, here are the most important pieces of code to share:

- Modules: These provide features and functionality that extend Drupal's core feature set.
- Themes: Graphic designs are implemented as "themes."
- Install profiles / Distributions ("distros"): Whole web applications can be made sharable and reusable as
- install profiles or distros. If developers build and maintain install profiles from the beginning, these are really easy to release later; trying to reverse engineer an install profile after the application is built can be laborious and costly.
- Make files: A Drush make file is like a recipe for an application including a list of all included projects and version numbers. You can use make files to assemble a whole application from all its constituent parts. These are usually included in install profiles on drupal.org to create a "distribution," but you can release these on your own on GitHub, too.

Whole web applications can be made sharable and reusable as install profiles or distros.

There are some pieces of code that developers do not release on drupal.org, but that might be valuable to others as an example or starting point for similar projects. These include site-specific modules including features or functionality that are specific to your government agency, as well as site-specific themes including branding assets, hard-coded privacy policies, or site names specific to your government agency.

Some code should NOT be released publically, including:

- Site-specific--potentially private or sensitive--configuration stored in settings.php
- Site-specific files such as .htaccess
- Copies of other people's work maintained elsewhere, such as on drupal.org
- A full copy of the exact site code base

How Do You Create a GitHub Organization?

Only existing GitHub users can create an organization, so you'll need at least one member of your organization to have a GitHub user account. Once a user is logged in, he can create a GitHub organization (follow the steps outlined here). Organizations are similar to users in that they can be named and have their own code repositories. Different paid account options correspond to different numbers of private repositories that your organization can own. You also have a free organization option, which contains your free open source code projects, but you have to pay to work on any private projects.

Once set up, an organization works similarly to a user account—it is paid, it can add users to the organization, and it can add users to its private projects. Organizations own the projects that are created within them, so no single user controls the fate of a project. For reference, Acquia has a GitHub organization that you can review. In it, you'll see different actions taken by members of the organization, and you can also see some of the Acquia GitHub organization members.

Organizations are similar to user profiles in that they can be named and have their own code repositories.

Who Controls an Organization?

Organization repositories have varying levels of permissions. Authorized members of an organizationcan manage issues, which means they can include issues or label them, can associate issues with milestones on the agency's project roadmaps, and can delegate issues to members of the team who will work on them.

Does the Public See All an Organization's Members and Their Activity?

The short answer is no. As mentioned, when you look at the Acquia GitHub organization page, you can see the public members listed, but some organization members might not be public, and therefore won't be listed. User visibility can be toggled on and off, depending on whether a project, and user association, needs to be kept private or is something that can be shared publicly. This is a helpful feature if you're working on projects requiring discretion.

User activity on public projects, however, is always public, no matter the user. But users who are private members of an organization can comment on issues as themselves, without the public knowing that they have any association with that organization. Here's an example:

- Jane Smith is a private member of Example Organization, so when you visit the Example Organization page, you won't see her listed as a member. Jane replies to an issue in the Example Organization issue queue, but appears to be a member of the public, because she has chosen to make her association with Organization Y private. There are instances, however, when Jane may want to show her association with Example Organization—especially in a situation where authority is needed, and she can prove her authority by sharing her association with the organization.

GitHub Developers and Workflows

We've discussed the basics of GitHub for Government, what the public can and can't see, what codeshould and should not be included in a public repo, and GitHub users and permissions. Now let's dive into developers and workflows.

How Are Things Organized on GitHub, and Who Does What?

GitHub organizes projects by associating them with an organization. Let's use Agency Z as an example. Agency Z has 10 projects on GitHub. The Agency Z copy of the code is the authoritative versions of these projects. A select group of developers has administrative access to this code, but the vast majority of users

—interns, junior developers, consultants—have read-only access. If one of those read-only users wants to make changes to a project—say a summer intern wants to propose a new widget—he will fork the code of a particular project to his account, which essentially creates a copy of the code in his account for editing. This won't give that summer intern access to the company's original code, but it will give him a copy of the code to play around with.

When he's made the changes he wants—in this case to implement a new widget—he can send those proposed edits back to the agency as a pull request. This pull request becomes visible to all members of that intern's team, and each of those users can review the changes. If more changes are needed, a user can send it back for development, but if the user likes the changes, that user can accept them by clicking the "Agree" button, which merges the pull request with the authoritative version of the code.

Anyone with read-only access can review changes, but only people with administrative permissions can actually accept the changes. In this case, other interns or junior developers can review the new widget and leave comments, such as "this widget works as expected," but only administrators can click Merge and Accept to make it a part of the authoritative version of the project.

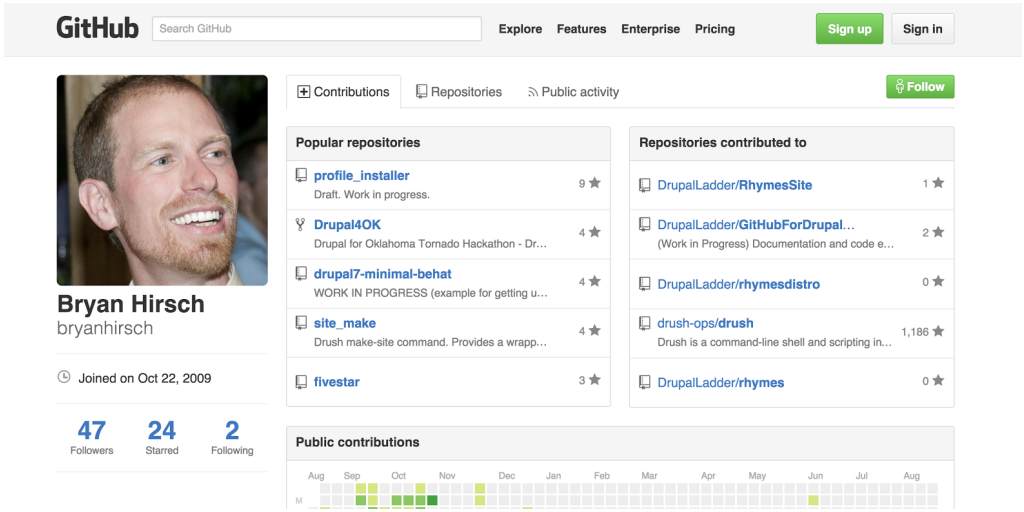
Can You Ensure No Code Gets in Without Peer Review?

The simple answer is yes. A review process can be dictated by permission levels. So if your organization is very conservative about your permissions, give almost everyone read-only access and only a very small number of people administrative access. This ensures a large pool of users can review the work, but only a select few administrators and gatekeepers can merge and accept pull requests.

It's also highly effective to have a rule in place that doesn't allow developers to merge their own code. I've seen regression rates drop to nearly zero by enforcing a workflow like this. Whether it's actually enforced with permissions, or it's just enforced by having an organizational policy that says, "No developers are allowed to merge their own code. All code must be peer reviewed and merged through a pull request."

How Does a GitHub User Submit Changes?

To make changes to a project, a user need to clone the project, which creates a copy of the project in her own user profile. Within the copied repository will be a few simple project files. The user should open the READ.ME file in a text editor, change the desired code and save it. Once changes are saved to the local copy within the user's profile, she can submit a pull request with the proposed changes to other team members for peer review.
For more in-depth, step-by-step instructions for the change and review process, check out this blog post.



What Do

You Do When a User Spots a Problem, or Something Must Be Fixed Before a Change Is Accepted?

Let's say I've submitted a pull request with my proposed changes, and it's currently being peer-reviewed. Then, I find an issue that needs to be addressed immediately, but my changes haven't yet been accepted and merged. GitHub users can leave comments on a pull request as a whole, but they can also leave very specific line-by-line comments in the code when they're doing a peer review. If they want, they can leave comments with check boxes. That way, a user could say, "Here's a list of issues that this code needs resolved before that step is committed." You can leave it for someone else to work through those issues. A user can leave the pull request open and push more revisions up. Users can require these conversations be reviewed before the changes can actually be accepted.

IMPLEMENTING Github for Government: Where to Start

The "Getting Started: Collaborative development with Git" blog is a great starting point. It walks you through a real-life example, with graphical support. Start there, and you'll be on your way to mastering GitHub for Government.

GitHub users can leave comments on a pull request as a whole, but they can also leave very specific line-by-line comments in the code when they're doing a peer review.

LET'S TALK