GitHub and Git

Standard Operating Procedures (SOP)

NMFS R User Group[1]

October 2022

Version 3.0

Introduction

NOAA's official GitHub policy from 2017 may be found here. Since 2017, GitHub use has expanded greatly across NMFS and all the federal agencies (link). The 2017 memo guidelines are not designed for science teams and scientific work that is done on GitHub. In addition, the 2017 memo largely focuses on the @NOAAGov GitHub organization, but NMFS GitHub use is not on this organization. This SOP was developed by the NMFS R User Group admins to provide practical help and guidelines for the wide variety of ways that Git and GitHub are used within NOAA Fisheries. This SOP was influenced by the SEFSC GitHub SOP released in October 2021 and by the work from the Fisheries Integrated Toolbox team who developed standardized GitHub guidelines and "best practices" for NMFS tool developers.

These guidelines are intended for scientific products that are low FISMA for which FedRAMP is not required. Scientific products are not generally considered official communication or business, and a disclaimer is added to GitHub repositories to state this. However, those who are handling higher FISMA data, relying on GitHub as the main data repository for official data, or working on products that will be official communications will need to seek further guidance for using GitHub.

1.0 Guidelines for Use of GitHub at NOAA Fisheries

The information here is intended to provide employees and affiliates of NOAA Fisheries (NMFS) with practical guidance and "best practices" for how to use GitHub. "NOAA allows use of GitHub to share code and content in the spirit of collaboration and open government." (2017 memo). NOAA has a strong history of scientific collaboration, coordination, and close engagement with other government partners, non-government organizations, academic institutions, international colleagues, and other members of the scientific research community.

1.1 Glossary

NOAA Branded Product - NOAA branded Organizations are created by a science center (or regional office), division, or program for its official products. These GitHub organizations are distinguished by clear branding that indicates that the repositories are official agency products.
GitHub Organization - Organizations are shared accounts where members of the organization can collaborate across many repositories at once. Owners and administrators can manage member access to the organization's data and projects with sophisticated security and administrative features.

Repository - A repository is hosted online and contains all of a project's files and each file's revision history.

1.1 What Content Can Be Shared on GitHub?

Generally, content on GitHub is limited to NOAA's scientific products as defined in the NOAA Scientific Integrity Policy. This policy defines such products as “The results of scientific activities including the analysis, synthesis, compilation, or translation of scientific information and data into electronic and hardcopy formats for the use of NOAA, the Department of Commerce, or the Nation. These products include, but are not limited to, experimental and operational models, forecasts, graphics, and verbal and written communications of all kinds relating to scientific activities, including NOAA social media accounts.”

The open source nature of GitHub allows content to be available for other developers to build upon or contribute to via fork, clone, or pull request. Embracing this open source workflow often facilitates open review by allowing others to comment and offer solutions for open issues, improving bug reports by allowing users to see source code, and providing the full history of the project changes (i.e., version control). Note, “open source” is not equivalent to making content publicly accessible. The level of visibility of a repository to the general public is a separate decision and is project dependent.

2.0 Account Guidelines

2.1 GitHub Personal Account Guidelines

In order to collaborate with colleagues and contribute to open science and open government over GitHub, you will need a GitHub account. This will allow you to create GitHub repositories, participate in GitHub organizations, use version control with GitHub, fork or clone repositories, contribute to other GitHub repositories, among other features.

- There should be clear separation between any use of GitHub associated with your NOAA activities and non-NOAA activities. You are not required to append “-NOAA” to your NOAA GitHub username, but it may be helpful for distinguishing between accounts.
- Your NOAA supervisor should be aware of your use of GitHub and have a clear understanding of what content is being shared on GitHub.
- When creating an account on GitHub.com use your @noaa.gov email as primary (under Settings/account) and as the notification email (under Settings/notifications). The latter ensures that notifications on work related repositories are sent to your NOAA email, not to a personal or university email address.
- Fill out your profile with your name and NOAA affiliation.
- Provide a real photograph of yourself as your Github profile.
- Enable Two-Factor Authentication.

- Official NOAA work products hosted under individual GitHub accounts are subject to special rules and notes. “Work products include, and are not limited to R scripts and functions, scripts for a markdown report, data, an app or tool, a citable software package in a repository on GitHub.
    - The repository ownership must be transferred to another NOAA individual GitHub owner at NOAA or to a NOAA GitHub Organization when/if an individual leaves NOAA. The individual can continue to contribute to the repository after leaving NOAA via pull requests, issues, etc., using their personal, not-NOAA affiliated accounts. Direct push permission to the repository for the now offboarded user must be ended.
    - Best practice is to maintain and contribute to NOAA work products within a NOAA GitHub organization associated with your center, division, or program rather than your individual GitHub account. This streamlines project onboarding and offboarding.

2.2 GitHub Repository Guidelines: All Repositories

GitHub provides a platform to host official work products, however GitHub repositories are used for a variety of purposes and not all repositories are “products”. Repositories are also used for project management, development, training, and sandboxing ideas. However, all repositories, regardless of purpose, must follow these guidelines:

- PII and BII should not be shared (on purpose or inadvertently) on GitHub (regardless of whether the repository is in a private or public repository). It is up to the end user to ensure best practices and safeguards are in place to prevent this. See Security section 3.0.
- No sensitive information should be shared in repositories. Sensitive information includes, but is not limited to, usernames, passwords, login information, port numbers, IP addresses, server names, Application Programming Interface (API) keys, Personally Identifiable Information (PII), Business Identifiable Information (BII), or confidential data.
- GitHub is **not a back-up service nor is it a data repository**. Other tools are designed for this purpose. At the minimum, this means maintaining a clone on at least one government furnished laptop or server. **Managers of NOAA branded GitHub organizations should work with their IT department to ensure automated regular backups of the organization repositories.** In other guidelines, this is often referred to as a “gold standard copy”.
- Only scientific content (section 1.1) that can be reasonably classified as FISMA Low (section 3.1) should be shared on GitHub.

Repositories that have code that interacts with APIs using IP addresses, usernames, passwords, secrets, or credentials must take steps to prevent committing of "secrets" to GitHub. (See section 3.3).

2.3 GitHub Organization Guidelines

GitHub organizations can be used by multiple GitHub users to collaborate on a collection of individual and team repositories, manage projects and tasks, document discussions, and allow permission customization for the organization members. The following are basic best practices for GitHub organizations.

- Fill out the organization profile with the NOAA affiliation.
- Provide contact information for the organization owner(s) or maintainer(s). Ensure that the organization has multiple owners who are able to manage the site. Ensure that when an organization owner leaves NOAA, another owner is able to manage the organization and all repositories.
- Create a README.md file for the organization in the .github/Profile folder. This README will then appear on the front page of the organization and can be used to describe the organization's purpose, affiliations, and repositories.
- **NOAA branded Organizations:** NOAA branded Organizations are often created by a center, division, or program for its official products. GitHub organizations that are NOAA Branded (i.e., have NOAA Fisheries logos and other branding indicating that the repositories represent official products), will need to have formal guidelines for participation in the organization. Examples of the guidelines include only allowing NOAA members to participate, no direct push access by non-NOAA accounts, guidelines for code review and tests, push access to repositories or visibility limited to specific members, formal naming conventions for repositories, and formal guidelines for repository formats. Branded organizations should have the disclaimer (section 4.2) added to the organization top-level README (located in the profile folder in the .github repository).

2.4 Additional GitHub Repository Guidelines

If your repository represents something you produced in the course of your work at NOAA and could be considered a 'work product', then your repository should include a README (section 4.1), the government product DISCLAIMER (section 4.2), and LICENSE file (section 4.3). If the work is something that can be cited, then also include citation information.

2.5 Collaboration with non-NOAA GitHub users

From the 2017 guidelines: “NOAA has a strong history of scientific collaboration, coordination, and close engagement with other government partners, non-government organizations, academic institutions, international colleagues, and other members of the scientific research community.” GitHub facilitates collaboration with non-NOAA collaborators and this is encouraged, however care needs to be taken for NOAA branded GitHub organizations and repositories. See discussion above on common restrictions places on NOAA branded GitHub organizations and repositories.

3.0 Security

3.1 FISMA Low

The scientific product must be reasonably classifiable as FISMA Low, as outlined by the Federal Information Security Management Act of 2002. FISMA Low classification includes only information for which the unauthorized disclosure, unauthorized modification, unauthorized destruction, or disruption of access can be expected to have a limited adverse effect on organizational operations, organizational assets, or individuals. If the effect of such events would be serious, severe, or catastrophic, the information cannot be released under this authority.

3.2 Sensitive information cannot be shared on GitHub

No usernames, passwords, login information, port numbers, IP addresses, server names, Application Programming Interface (API) keys, Personally Identifiable Information (PII), Business Identifiable Information (BII), or confidential data may be stored in any file hosted on GitHub. Read Section 3.3 on how to properly store and use credentials. If you have GitHub Actions or Pages that use credentials, then Encrypted Secrets inside of GitHub is also acceptable for API (Application Programming Interface) keys and similarly credentialed interfaces.

3.3 Preventing inadvertent committing of secrets or credentials to GitHub

If your repository code uses confidential data or connects to APIs using IP addresses, usernames, passwords, secrets, or credentials, then you must take steps to ensure that you do not inadvertently commit these to GitHub. Ensuring that confidential data or secrets do not get pushed to GitHub requires diligence and deliberate choices in your workflow.

Your approach must be tailored to the nature of your work and the content of the repository. For example, a repository that is a package for fitting species distribution models to generic data or a demo repository for teaching purposes is very different from a repository that produces an official report using confidential fisheries data.

3.3.1 Prevent Damage

Establish a workflow that keeps confidential information separate from non-confidential information

- Separate your files into “public” and “secrets” folders. Confidential data or secrets cannot be dispersed throughout your code and files. Clearly distinguishing between “public” and “secrets” in your code and files will require a deliberate and carefully designed file organization.
- Once you have “safe” and “unsafe” folders delineated for your repository, use the `.gitignore` file to prevent these from being pushed to GitHub. You may need to add these files to the `.gitignore` file before you push to your GitHub repository.
- Alternatively, for code that integrates data or databases, develop a public version of the package. This public version of the package would not include any of the data or credentials and can then be used locally and called with arguments to create the output.

Passwords and API links

- Never leave usernames or passwords into your code or scripts. While it may be tempting to save time, this habit is a common cause of inadvertent sharing of credentials and is a security risk.
- The {dotenv} and {keyring} packages can be used to securely store API links and credentials for connecting to databases, APIs, and other services. These packages are easy to use and implement into your workflow. Other APIs (like Google Drive) use a secrets folder which has encrypted tokens for accessing their APIs.
- The basic idea is to keep the “keys” on your local computer so you’ll need to add the folders or files with the secrets to your `.gitignore` file.
- GitHub Actions or Pages that use API (Application Programming Interface) keys and similarly credentialed interfaces can use Encrypted Secrets inside of GitHub.

Customized pre-commit hooks to prevent committing secrets

Repositories that are in danger of committing access keys to cloud computing resources or similar credentials with high consequences will need to implement mechanisms to prevent committing secrets to GitHub in the first place. {gitleaks} provides this mechanism, but implementing it is involved. It uses “git hooks”, which are a special file in the .git/hooks folder within a repository. The gitleaks documentation gives instructions on how to set this up. You will need to customize the “hook” to detect your particular type of secret.

Restrict push permissions and require pull-requests and code review

You can also incorporate code and content review but only allow pull requests and no direct push access. This allows time for a manual review of the material being pushed to GitHub.

3.3.2 Detect Damage

If a secret has mistakenly made it onto GitHub, you will need to remove it as soon as possible and change keys/passwords that were shared.

Enabling repository scanning via {gitleaks} can provide an alert if you inadvertently committed secrets to your repository. The scan will be automatically run using a "GitHub Action" every time a "push" or "pull request" is completed. Depending on what credentials you are worried about, the generic {gitleaks} scan using a GitHub Action will be sufficient and is easy to apply. However the default types of secrets that {gitleaks} detects may not be useful without customization. For example, {gitleaks} provides a means of automating a security scan by scanning for API keys and other credentials accidentally left in the code. However, the default behavior would not detect confidential fisheries data nor would it detect a password hard-coded into a script. Note, previously, NOAA policy required the use of a *git-secrets* addon; {gitleaks} is a newer and improved program. {gitleaks} requires customization for the types of secrets that might be pushed to your repo, and it is essential that you test that {gitleaks} will detect your secrets after you have customized it.

This generic Gitleaks scan can be enabled by following these steps:

1. Add the below folder structure to a given local directory

```
local_repo/
├── .git/
├── .github/
│ ├── workflows/
│ │ └── secretScan.yml
```

1. Copy and paste the following code to the secretScan.yml file, making sure to keep the spaces exactly as typed below:

```
name: gitleaks

on: [push,pull_request]

jobs:
 gitleaks:
  runs-on: ubuntu-latest
  steps:
```

```
- uses: actions/checkout@v2
  with:
    fetch-depth: '0'
- name: gitleaks-action
  uses: zricethezav/gitleaks-action@master
```

2. Customize the {gitleaks} scan accordingly for the types of secrets that the organization repositories use. Some custom secrets or credentials may not be caught by the default {gitleaks} scan. This situation would be most likely to occur at the GitHub organization level. See the gitleaks documentation.

4.0 GitHub Repository Components

4.1 README.md File

This file should provide a description of the repository. The contents of the README file will vary greatly depending on the application. Here are some tips:

- **An official work product released on a NOAA-branded GitHub organization**. This is the most formal instance and the organization leaders should develop template READMEs to provide a consistent format across the products.
- **A software package (e.g., R package)**. Standard components are: Badges indicating build status and version, how to install, how to use or link to documentation, where to report issues, citation.
- **A report or paper.** Author, description and citation.
- Add the DISCLAIMER (below) for any repository that can be considered a 'work product'.

4.2 DISCLAIMER File

Repositories and web content shared on GitHub should make it clear to the audience that no information should be considered or interpreted as official communication of NOAA. The simplest method for doing this is to include the following disclaimer within a DISCLAIMER.md file at the root of each repository. Additionally, it's also good practice to include the disclaimer text as a footnote within the repository's README.md and any web content available from that repository. Be careful not to use NOAA logos and and NOAA Fisheries branding in a way that could imply an official communication. NOAA logos should be used to indicate your affiliation and acknowledge support and funding. The repositories within the NOAA Fisheries Integrated Toolbox GitHub organization are good examples of how to prepare README files with the Disclaimer and NOAA Fisheries logos. https://github.com/nmfs-fish-tools

The following disclaimer should be included within a DISCLAIMER.md file in the root of the repository.

> *This repository is a scientific product and is not official communication of the National Oceanic and Atmospheric Administration, or the United States Department of Commerce. All NOAA GitHub project code is provided on an 'as is' basis and the user assumes responsibility for its use. Any claims against the Department of Commerce or Department of Commerce bureaus stemming from the use of this GitHub project will be governed by all applicable Federal law. Any reference to specific commercial products, processes, or services by service mark, trademark, manufacturer, or otherwise, does not constitute or imply their endorsement, recommendation or favoring by the Department of Commerce. The Department of Commerce seal and logo, or the seal and logo of a DOC bureau, shall not be used in any manner to imply endorsement of any commercial product or activity by DOC or the United States Government.*

4.3 LICENSE Files

The work of U.S. Government employees is not subject to copyright in the U.S. Thus you should include an appropriate LICENSE file with your repositories. There are two types of licenses that are commonly used: Creative Commons Zero v1.0 Universal (CC0 1.0 Universal) and (GNU General Public License v3.0) GPL-3. CC0. The GPL-3. CC0 license is more commonly used for US Government products but the NOAA Fisheries Integrated Toolbox prefers the GPL-3 license since it is more tailored for code products. To add a LICENSE file to your repository, navigate to the repository on GitHub.com, select “add file,”, select “create new file”, and type LICENSE. You will then see the button in the upper right hand of the page to choose a license file.

Additionally, the following statement applies with regards to licensing for any code published in any NOAA repository. This should be included within a LICENSE.md file at the root of each repository. Repositories that are code packages should also add this to the repository README.md.

> *Software code created by U.S. Government employees is not subject to copyright in the United States (17 U.S.C. §105). The United States/Department of Commerce reserves all rights to seek and obtain copyright protection in countries other than the United States for Software authored in its entirety by the Department of Commerce. To this end, the Department of Commerce hereby grants to Recipient a royalty-free, nonexclusive license to use, copy, and create derivative works of the Software outside of the United States.*

5.0 Contact and Resources

These guidelines were developed by the NMFS R User Group administrators with input from (scientific) administrators of GitHub organizations that collaborate and deliver science products on GitHub. The NMFS R User Group supports agency staff in the use of open science and reproducible science tools within the agency. See the NMFS Open Science website for contact information.