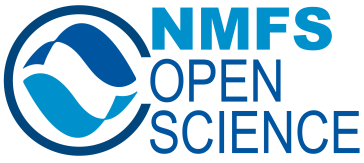
GitHub and Git

Guidance and Best Practices for NMFS Users
version 3.8

NMFS Open Science GitHub working group



0.1 Introduction

This "best practices" guide was developed by and is maintained by the NMFS Open Science GitHub working group (Section 0.11). This document provides practical help and guidelines for using Git and GitHub within NOAA Fisheries. This guide was heavily influenced by the work from the Fisheries Integrated Toolbox team who developed standardized GitHub "best practices" documents for NOAA Fisheries tool developers, by the GitHub SOPs developed by SEFSC, NEFSC and NWFSC, and other guidance documents. See references (Section 0.12).

0.2 Quick Start

Here are the steps for getting started with GitHub and GitHub Enterprise at NOAA Fisheries with detailed instructions in the subsections. See the GitHub Governance Users Page for more information on on-boarding.

1. Create a GitHub user account with your NOAA email.
2. Turn on 2-Factor Authentication on your account.
3. Create a GitHub repository.
4. Add LICENSE+INTENT files (or LICENSE+NOTICE files) to the repo and fill out a README. See Section 0.9.

The next instructions are for access to NOAA Fisheries GitHub Enterprise organizations.

3. Download the NMFS GitHub Enterprise Cloud user agreement and have your supervisor (or NOAA sponsor if a contractor) sign it.
4. Sign up with the Google form on this on-boarding page and upload the form there.
5. Request access to your offices' GitHub Enterprise Cloud organization.
6. Watch for an invite to the GitHub organization in your email and on-boarding instructions.

0.2.1 Create a GitHub user account

You will need a GitHub user account that is specific to your NOAA work and that uses your NOAA email for notifications. If you have an existing GitHub account that you only use for NOAA work, you can simply add your NOAA email as the primary contact for notifications. If you have an existing GitHub account that you use for non-NOAA work, e.g. another job, university work, or personal work, then you will need to create a new GitHub account for your NOAA work.

1. Go to www.github.com.
2. Create an account with your NOAA email. Your username should include your name, e.g. FirstLast or initialslastname. Some users add "-NOAA" to the end of their username. This is not required but helpful if you have another non-NOAA account.
3. Edit your profile and add your NOAA affiliation and your real name.

0.2.2 Turn on 2-Factor Authentication on your account

Go to https://github.com/settings/security and turn on 2-Factor Authentication. Add multiple methods for authentication in case your primary method does not work.

You will not be asked to authenticate with your 2nd factor every time you log in, but you will be asked if you try to do something that might be dangerous. For example, if you are adding a collaborator to a repository, you will be asked for your 2nd factor.

0.2.3 NMFS GitHub Enterprise Cloud user agreement

If you are requesting access to the NOAA Fisheries GitHub Enterprise organizations, download the GitHub Enterprise Cloud user agreement (NOAA internal) and have your supervisor sign it. You will upload this when you request access to your office's GitHub Enterprise organization.

0.2.4 Request access to NMFS GitHub Enterprise Cloud

Request access to the GitHub organization for your the office where you work and any other GitHub organization that you need access to. Use this Google form to request access. You will need to upload your user agreement.

Wait for an email inviting your to become a member of the GitHub organization. Once you accept the invitation, you see that you are listed as a member of your office's GitHub organization here: NMFS GHEC site. Note, you will get a 404 error unless you are a NMFS GHEC member.

0.2.5 Wait for the organization invite and on-boarding instructions

Each NMFS GitHub Enterprise organization has a different on-boarding process. This Google spreadsheet (NOAA internal) shows you the admins for the GitHub Enterprise organizations.

Be on the look out for an invite to the GitHub Enterprise organization. Accept the invitation and then you will be able to see the NMFS GHEC organizations. Once you

are a member of the organization, review its SOP (in this Google drive folder (NOAA internal)). Reach out to your local NMFS GHEC organization admins (in the spreadsheet) if you need help.

0.2.6 Authenticating to GitHub

To push and pull changes to GitHub from your computer, you will need to authenticate to GitHub.

- GitHub Desktop. Sign into GitHub under GitHub Desktop > Settings > Account. For both GitHub Free and GitHub Enterprise Cloud, sign into GitHub.com with your username and password (not token). Ignore the GitHub Enterprise sign in. The latter is for an on-premise installation of Enterprise not for Enterprise in the Cloud (which NMFS has).
- VSCode, GitKraken, Positron. Similar to GitHub Desktop, these allow you to authenticate by signing into your GitHub account with your username and password.
- Using a Personal Access Token. When using RStudio or the terminal to interact with GitHub, you can use a token. Go to https://github.com/settings/tokens and create a token, which you will use when asked for a password from GitHub. Typically, the scope only needs repo. If you use GitHub Actions, add workflow scope. Copy the token as you will not see it again after it is created. If you need to use the token for GitHub Enterprise, select the Configure SSO button and select your GitHub Enterprise organization.
- Using SSH. https://www.geeksforgeeks.org/using-github-with-ssh-secure-shell/

Please see the NMFS Open Science Resource Book on installing Git and Authenticating to GitHub for details on the specifics of set-up for RStudio, VSCode, terminal, etc.

RStudio Users

This is a callout with a title.

0.2.7 Step 1

Go to Tools > Global Options > Git/SVN. Make sure the path to the git executible is there. Not there? Open a terminal and type `where git` (Windows) or `which git` (Mac). Copy the path shown into the git executible box. Still can't find the location of your git executible and you know it is installed (because you have, say, GitHub Desktop installed)?

- On a Mac? It is at /usr/bin/git Just paste that in.
- On a PC? Look (by opening a finder window) in the following places. Try

the GitHubDesktop one first.

- C:/Program Files/Git/bin/git.exe
- C:/Users/your.username/AppData/Local/Programs/Git/bin/git.exe
- C:/Users/your.username/AppData/Local/GitHubDesktop/app-2.8.3/resources/app/git/cmd/git.exe

0.2.8 Step 2

Open up RStudio and run this from the console:

```
install.packages("usethis")
usethis::use_git_config(user.name = "YourName", user.email = "your@mail.com")
# this will open a browser to create a token
usethis::create_github_token() #Accept defaults scope. Copy the token. Copy that into Y
credentials::set_github_pat("YourPAT")
```

0.3 Background

In 2017, GitHub use was authorized for scientific products: CIO memo authorizing GitHub use at NOAA. Subsequently, in 2017, NOAA released official GitHub guidelines here. Some individual centers and offices prepared local GitHub guidelines and SOPs to help their staff interested in using GitHub. See references (Section 0.12). Since 2017, GitHub use has expanded greatly across NMFS and all the federal agencies (link), and GitHub has become integrated into modern scientific workflow. When the 2017 memo was written, GitHub was not Fedramp authorized. GitHub Enterprise became Fedramp authorized in 2018 and GitHub use expanded rapidly in other agencies (e.g. NIH, Census, GSA and VA) using Enterprise. In 2023, NOAA Fisheries secured an Authorization to Use (ATU) for GitHub Enterprise Cloud for Low FISMA scientific products; Memo (NOAA internal).

The 2017 memo tells us what can be shared on GitHub (scientific products) but there is a need for guidelines that cover and recognize the wide variety of scientific work (and learning) that is done on GitHub and much has changed in the tools provided by GitHub since 2017. Local GitHub SOPs have been developed by individual centers (see references Section 0.12) to help their staff. These are very helpful, but even with these, staff are often confused by how to follow the guidelines in practice. For example, if I fork a repo in statistics workshop I am in, am I supposed to have a gold standard backup for that? What does it mean to keep NOAA work separate? I work with confidential fisheries data; what are my options? I am collaborating with someone at a university on a Python package; what are my options? I am collaborating with a regional office on an assessment that will be released as an official report from my center; what are my options?

> **Note**
>
> GitHub Enterprise Cloud (GHEC) versus GitHub Cloud free. GHEC does not require a different username or account. You simply need to be invited into one (or more) of the NMFS GHEC organizations. Any repositories, projects, or other resources you contribute to within the GHEC organization will be under your GHEC license. For resources under your individual account or non-GHEC organizations, you'll be collaborating under your GitHub Cloud free license.
> The NNMFS GHEC organizations have more security controls in place and will require additional authenticaion with your CAC card OR NOAA Google account (just like other single sign-on (SSO) websites for NOAA resources). You do not need to be on VPN to access a NMFS GHEC organization. You will not need to authenticate with SSO every time you visit resources on GitHub.com and, if you access GitHub resources via your development environment software (e.g. VS Code, RStudio) you will need to set up or re-generate a Personal Access Token.

0.3.1 Scientific products

These guidelines are intended for scientific products that are low FISMA. 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.

0.3.2 NOAA and Open Source Software

NAO 201-118: Software Governance and Public Release Policy describes our obligations regarding the code we write for NOAA mission work, in particular code that underlies analyses, models or packages that are core to our products and advice. Key points of this policy:

- To the extent possible, i.e. no confidentiality contraints, code should be released openly and with an open source licence. NOAA Fisheries supports GitHub Enterprise organizations at each science center and office to support code sharing and collaboration.
- You should follow basic good practices regarding code development to improve code robustness and useability. A core good practice is version-control (Git) within a user interface that encourages bug tracking and testing (GitHub).
- You should follow good practices for ensuring the security of your code. Use of GitHub Enterprise will take care of this by ensuring back-ups and providing access control (SAML sign-on for your NOAA collaborators) and logging of activity.

- You should apply an open license to your repositories and ensure that code developed under contracts or grants is under an open license and we have the source code (at least a copy). See Section 0.9.3 for recommended licenses.
- Software that is used for products or advice needs a formal review and testing process. Note, in scientific contexts such reviews are typically conducted by statistical review teams. For R software, submission to CRAN (or bioconductor) ensures that the packages follows standards agreed upon by the R community. A higher level of review is the rOpenSci peer review but even if you do not submit a package to rOpenSci, their development guide provides best practices for R scientific packages in addition to the Wickham and Bryan guide.

0.4 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 GitHub memo) and to support NOAA's obligation to share code developed with federal funds (NAO 201-118). 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.

0.4.1 Glossary

NOAA Branded Product - NOAA Branded Organizations are created by a science center, 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.

Non-Enterprise GitHub Organization (org) - Organizations are shared accounts where members of the org 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. Org in this context is not related to organizations on an org chart. It is more like a managed folder of folders.

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

0.4.2 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 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, usually Git). 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.

0.4.3 Sharing data: Alternatives to Git Large File Storage

Oftentimes, data also needs to be shared with code. Small datasets can be directly committed to a repository in a variety of formats. However, when datasets are large can be more challenging. GitHub restricts pushing of files large than 100 MiB from the command line. There is GitHub Large File Storage for files up to 5 GB on GitHub Enterprise Cloud, but in practice, staff have found some difficulties with Large File Storage. Issues include:

1. It is difficult to delete a file once it has been committed, even if the version history is rewritten to remove the file. GitHub suggests deleting the repository, which may not be possible for established repositories.
2. There are limits to the the included space for Large File Storage with GitHub accounts.

Because of these issues, other ways of sharing data may be preferable. Some options include: - archiving data at NCEI. - storing data in an on-premise database (contact your office's IT department for more information about what is available to you). - sharing public data via the NOAA Open Data Desemination program (NODD). For example, some Alaska Fisheries Science Center data is shared through the NODD, which provides access to cloud storage. - for large files as part of a release, the piggyback R package. - storing and sharing datasets via Google Drive.

0.5 Repository Guidelines

0.5.1 GitHub Personal Account Settings

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 if you use GitHub for NOAA work and non-NOAA work[1].
- 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.
- Enable Two-Factor Authentication.

> **Note**
>
> Your NOAA supervisor should be aware of your use of GitHub and have a clear understanding of what content is being shared on GitHub. Your supervisor can 'follow' repositories on GitHub if they need to be aware as changes are pushed to GitHub repos.

0.5.2 GitHub Repository Guidelines

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 testing out ideas.

All repositories, regardless of purpose, must follow these general guidelines:

- PII and BII should never be shared (on purpose or inadvertently) on GitHub regardless of whether the repository is in a private or public repository. Best practices and safeguards must be followed to prevent this.

[1] Work when not employed by NOAA or work that is not part of your job and you are not paid for this work by NOAA.

- 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 with archiving. Other tools are designed for this purpose. See Backups (Section 0.8).
- Only scientific content (Section 0.4.2) that can be reasonably classified as FISMA Low (Section 0.6.1) 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 0.6.2).

0.5.3 Disclaimers and Licenses

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 0.9.1) and LICENSE file (Section 0.9.3). If the work is something that can be cited, then also include citation information and a DOI[2]. Scientific works that are not official communications should include the DISCLAIMER (Section 0.9.2).

0.5.4 Repositories Under Individual Accounts

GitHub repositories can be created under your individual GitHub account or under a GitHub organization that you are a member of. If you have NOAA work products[3] hosted under your individual GitHub account, you need to do the following to ensure your NOAA work is transferred to a new NOAA owner when you leave NOAA.

- The repository ownership must be transferred to another NOAA individual GitHub owner at NOAA or to a NOAA GitHub organization when an individual leaves NOAA.
- Under Settings, specify a successor who can take over your account if you are unable to transfer your repositories.

What about a repository associated with a journal article if writing this was part of your NOAA job? Transferring that kind of repository would probably not make sense. The transferring policy is for things that someone who is on-boarding for your job will

[2]DOIs are easy to generate with the Zenodo plugin for your GitHub repositories. Read how here.

[3]What is a NOAA work product? First, this is part of your job and you are being paid by NOAA to do this work. Second, it is a product. A repo that you throw onto GitHub as part of something you are testing out or during a workshop you are taking is not a 'product'. Would you put a DOI on this thing? Yes? That's probably a product. Note, NOAA Fisheries provides GitHub Enterprise for this kind of work and having the work under your center's GitHub organization is best practice.

need. Think about the kinds of work you would transfer off your computer during off-boarding.

Note, you should not have mission critical repositories (software products or the main code and scripts for official NOAA reports) in your personal (work) GitHub account. That is how we lose access to code – when something happens to you or you lose access to your account. This type of work should be in a GitHub organization.

> **Non-Enterprise GitHub Organizations**
>
> You can also create your work repositories in a GitHub organization rather than your individual GitHub account. GitHub organizations can be set-up so that its members can create and manage their repositories freely as they would in their individual account. Creating work repositories in a GitHub organization greatly streamlines on-boarding and off-boarding. It also makes it easier for all team members to use similar templates for their repositories.
> NOAA Fisheries has an Enterprise Cloud version of GitHub and there are existing organizations within the Enterprise that you can use. See the NOAA Fisheries GitHub Governance Team contact page (NOAA internal only) for more information on NOAA Fisheries Enterprise Cloud.

0.5.5 Non-Enterprise GitHub Organization Settings

A GitHub organization is simply an account where multiple GitHub users can be named as members who are then given permission to create repositories. The organization can have as many admins as you need, but 2-3 is common. Anyone can create a GitHub organization and it is a convenient way for teams or groups to organize and collaborate on a thematic set of repositories. In addition, a GitHub organization allows members to manage project boards, tasks, and discussions, and allows the admins to customize permission settings 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.

> **Naming Conventions**
>
> If your GitHub organization is specific to one center, be aware that some centers use a naming convention, e.g., `nwfsc-mathbio`.

0.5.5.1 NOAA Branded Organizations

There is nothing inherently official or formal about a GitHub organization; it is simply an account with multiple GitHub users (members). However a GitHub organization can be formal and highly regulated. A GitHub organization that delivers official products for the public would fall under this category and these organizations would use clear NOAA logos, text and other branding.

GitHub organizations that deliver official NOAA 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, naming conventions for repositories, and guidelines for repository structure. The NOAA Fisheries GitHub Enterprise organizations are an example of GitHub organizations that are branded and have formal guidelines and access controls.

0.5.6 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 Section 0.5.5.1 on common restrictions placed on NOAA branded GitHub organizations and repositories.

0.6 Security

0.6.1 FISMA Low

The scientific product on GitHub 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.

0.6.2 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.

0.6.3 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.

> **Note**
>
> 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.

0.6.3.1 Prevent Damage

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

- Separate your files into "public" and "secret" folders. Confidential data or secrets cannot be dispersed throughout your code and files. Clearly distinguishing between "public" and "secret" in your code and files will require a deliberate and carefully designed file organization.
- Once you have "public" and "secret" 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.

Restrict push permissions and require pull-requests and code review

You can 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.

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 should be used. It uses "git hooks", which are a special file in the `.git/hooks` folder within a repository. See the {gitleaks} documentation. Another option is the git-secrets add-on which you install locally.

Note, secrets scanners like gitlinks need to be customized and tested. Also, They are designed to catch secrets that look like AWS keys or API keys and not confidential fisheries data or a simple text password hard-coded into a script.

0.6.3.2 Detect Damage

For all public repositories, GitHub offers a feature called secret scanning. Learn more in the GitHub doumentation for secret scanning alerts.

For private repositories, enabling repository scanning via a secrets scanner such as gitleaks GitHub Action can provide an alert if someone inadvertently commit secrets to your repository. The scan will be automatically run using a "GitHub Action" every time a "push" or "pull request" is completed. GitLeaks is free, but requires a license for organizations.

0.7 Recover from 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.

Follow GitHub's guidance to remove sensitive data from a repository. Note that you must re-write git history rather than just making a new commit with the confidential data removed, otherwise the confidential data will continue to exist in the commit history.

0.8 Backups

GitHub is not a backup service and if a repository were deleted on GitHub, it is gone. Note if it is not deleted, one can generally recover the state of the repository at a past time (unless something really bad were done to the Git record). At the minimum, this means you should maintain a clone on at least one government furnished laptop or server. Alternatively you can back-up to Google Drive with a script (below). *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".

Example of back-up scripts to Google Drive:

- ghbackup by NOAA Fisheries Integrated Toolbox
- backup-gdrive

0.9 GitHub Repository Components

Your repository should include a README (Section 0.9.1) and LICENSE (+ INTENT or NOTICE) files (Section 0.9.3). If the work is something that can be cited, then also include citation information and a DOI. Scientific products that are not official communications should include the government product DISCLAIMER (Section 0.9.2).

0.9.1 README.md

This file should provide a description of the repository. The contents of the README file will vary greatly depending on the application, but there are a few common elements for NOAA Fisheries work products: title, description, developer information, disclaimer (Section 0.9.2), NOAA Fisheries logo, and DOC footer. See examples from repositories linked on the Fisheries Integrated Toolbox.

Specialized repositories will have additional elements in their README:

- **A software or code package (e.g., R package)**. Standard components are: Badges indicating build status and version, description, how to install, how to use or link to documentation, where to report issues, authors, citation. Example and tools linked in the Fisheries Integrated Toolbox.
- **A report or paper.** Authors, description and citation.

0.9.2 DISCLAIMER.md

Repositories and web content shared on GitHub that are scientific products and not official NOAA communications should make this clear to the audience. The simplest method for doing this is to include the disclaimer text as a footnote within the repository's `README.md` and any web content available from that repository. For this type of scientific repository, NOAA logos and and NOAA Fisheries branding should not be used in a way that could imply official communication. NOAA logos should only be used to indicate your affiliation and acknowledge support and funding. The repositories within the NOAA Fisheries Integrated Toolbox GitHub organization are examples of how to prepare README files with the Disclaimer and NOAA Fisheries logos.

The following `DISCLAIMER.md` file is put in the root of the repository that are scientific products and not official communications (per 2027 CIO GitHub Memo in references).

DISCLAIMER.md

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.

0.9.3 LICENSE Files

Federally funded work, whether by federal employees or by contractors or grantees, must be publically released as open source (unless there are specific legal prohibitions) with rights sufficient to allow government-wide access; see the Share It Act. NAO 201-118 on Software Governance and Public Release Policy describes code in tiers: Tier 0: Trivial software for individual use. Tier 1: Simple software for one-time publicly visible use

(e.g., plotting scripts for a publication). Tier 2: Software for daily or one-off use (e.g., supporting academic papers), Tier 3: Software tools intended for repeated use or public release (e.g., development of simulation models). Tier 4: Similar to tier 3, but mature enough for applied research or broad public distribution. Tier 5: Tier 4 software with active full support. Tier 0, 1, and 2 software do not require public release.

Federally funded code should have an open license file added to the GitHub repository before work begins. This establishes from the moment work starts that all work is open source (as required for publically released federally funded work). Difficulties can arise when work is done with a non-federal contributor (contractor or grantee) with federal funds, but it was not established from the beginning that the work is under an open source license. The recommended license is Apache 2.0 for code/software and CC0 for data and documentation.

To add a license to your repository, navigate to the base level of your repository (so not in a subdirectory) and add a file called `LICENSE`. GitHub will ask if you want to use one of the template licenses, but the guidance from the draft handbook to the NAO 201-118 is to use a custom license format. See below. Note if the instructions below seem complicated, keep in mind that the key is to add an open license from the start. The exact details of which one is less important than actually having one on the repository before contributions are added.

Copyright and authorship

You will notice that the license statements have a copyright statement. You are stating who have the right to release the work under the stated license. If the author is a federal employee, then they cannot claim copyright but you can still be listed as the author in the notice, if you wish. Why the "Intent" statements? Works by federal employees are in the public domain so technically you cannot apply a license to something you don't have ownership of. The intent statement makes it 100% clear that the intent is to release under the stated license.

Software and code

Section 7D of NAO 201-118 specifies that software developed by NOAA or with NOAA funding specifically for its mission will be developed and publicly released as Open Source unless legally prohibited or superseded by formal, written agreements. For software jointly developed jointly with contractors, grantees, cooperative institutes, private entities, interagency partners, international partners, or Cooperative Research and Development Agreement partners, text should be included in the contract or agreement to ensure that the software and code will be publicly released as open source.

Per the draft handbook on NAO 201-118, Apache 2.0 is the recommended license for software and code developed with NOAA funding. The recommended way (per NAO 201-118 handbook) is to include this in a GitHub repository in a file called `LICENSE` with the following text and then add a INTENT file. However, some software package

repositories (like CRAN) will not allow custom license files; in that case use the LICENSE and NOTICE approach below. Both approaches have the same affect.

1) LICENSE and INTENT file

Text for `LICENSE` file:

```
Copyright [year developed] U.S. Federal Government (in countries where recognized)
Copyright [year developed] name of non-federal employee contributor. Add any needed info

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software is distributed on an
```

Text for `INTENT` file:

```
The intent is that this software and documentation ("Project") should be treated as if it

Unless required by applicable law or agreed to in writing, software is distributed on an
```

2) LICENSE and NOTICE file

What to do if you cannot use a custom license (as required for CRAN or some code repositories)? In that case, the normal approach for Apache 2.0 is to have the `LICENSE` file be the unchanged Apache 2.0 text and include a file called `NOTICE` with the copyright and intent information. For example, your `NOTICE` file would look like

```
Copyright [year developed] U.S. Federal Government (in countries where recognized)
Copyright [year developed] name of non-federal employee contributor. Add any needed info

The intent is that this software and documentation ("Project") should be treated as if it

Unless required by applicable law or agreed to in writing, software is distributed on an
```

Permissive (as in open) alternatives to Apache 2.0 are the MIT license and the LGPL licenses. A less permissive alternative, The GNU General Public License v3.0 (GPL-3) is sometimes used when a copyleft licenses is needed (e.g., copying in code that has a GPL-3 license requires the project using the code to also have a GPL-3 license), however whenever possible the more permissive licenses should be used.

Optionally, you can add authorship information to the top of the NOTICE. For example:

```
Authors:
- Your Name, NOAA Fisheries (U.S. Federal Government)
- Their Name, University of Example - Contributed data analysis and visualization modules
```

Data and documentation

Per the NAO 212-15B on Management of NOAA Data and Information, the Creative Commons Zero v1.0 Universal (CC0 1.0 Universal) license is used for data and content products. Your `LICENSE` file would be similar to the Apache 2.0 example.

1) LICENSE and INTENT files

Text for `LICENSE` file:

```
Copyright [year developed] U.S. Federal Government (in countries where recognized)
Copyright [year developed] name of non-federal employee contributor. Add any needed info

Licensed under the Creative Commons Zero v1.0 Universal (CC0 1.0 Universal) (the "License

https://creativecommons.org/publicdomain/zero/1.0/deed.en

Unless required by applicable law or agreed to in writing, data and content is distribute
```

Text for `INTENT` file:

```
The intent is that this software and documentation ("Project") should be treated as if it

Unless required by applicable law or agreed to in writing, software is distributed on an
```

2) LICENSE and NOTICE files

What to do if you cannot use a custom license (as required by some data archives)? In that case, the `LICENSE` file be the unchanged CC0 text and include a file called `NOTICE` with the copyright and intent information. For example, your `NOTICE` file would look like

```
Copyright [year developed] U.S. Federal Government (in countries where recognized)
Copyright [year developed] name of non-federal employee contributor. Add any needed info

The intent is that this data and documentation ("Project") should be treated as if it is

Unless required by applicable law or agreed to in writing, this product is distributed or
```

Optionally, you can add authorship information to the top of the NOTICE. For example:

```
Authors:
- Your Name, NOAA Fisheries (U.S. Federal Government)
- Their Name, University of Example - Contributed data analysis and visualization modules
```

0.10 GitHub Governance Team and access to GitHub Enterprise Cloud

The NOAA Fisheries GitHub Governance Team provides access to and governs the use of GitHub Enterprise Cloud at NOAA Fisheries. See the NMFS GitHub Governance Team website (NOAA internal) for information on access to GitHub Enterprise Cloud.

0.11 GitHub Guide authors

The NMFS Open Science GitHub working group is a group of NMFS scientists who are active developers on GitHub and with wide experience is the many ways that GitHub is used within NOAA Fisheries. The group is involved in helping NMFS scientists with GitHub use by developing this best-practice document, helping with trainings and content, and helping govern on the GitHub Governance Team.

Lead editors:

- Eli Holmes, Northwest Fisheries Science Center, GGT rep
- Josh London, Alaska Fisheries Science Center, GGT rep
- Emily Markowitz, Alaska Fisheries Science Center
- Kathryn Doering, Office of Science and Technology, GGT rep

The editors assembled the material into a cohesive format, but significant sections were developed by other individuals in other contexts. See also the references (Section 0.12).

- Much of the section on setting up your GitHub individual account was adapted from the SEFSC October 2021 GitHub SOP. This SOP effort was led by James Primrose, SEFSC. Note the SEFSC SOP was developed from the NEFSC SOP led by David Chevrier, NEFSC.
- Most of the information on standard elements of a NMFS repository (disclaimers, readme, logos, etc) was adapted or taken directly from the Fisheries Integrated Toolbox Resource pages. Corinne Bassin, Christine Stawitz, Kathryn Doering, and Bai Li were the developers of the FIT material.
- The section on security used information from a presentation on "Keeping secrets secret" in the context of accessing fisheries databases via APIs by Adyan Rios, Southwest Fisheries Science Center.

- Some of the material was first assembled in a less formal format in the NMFS Open Science Resource Book, a project of the 2021 Openscapes cohorts at NWFSC and AKFSC.
- Thanks to the many NMFS R User Group members who commented on the initial drafts over 2022.

0.12 References

- 2017 CIO Memo authorizing GitHub use at NOAA Note this document was written before Microsoft bought GitHub and before GitHub Enterprise existed.
- 2017 NOAA GitHub Guidelines for NOAAGov GitHub Org This document appears to be intended to apply only to content on https://github.com/NOAAGov. See top of page 2.
- 2023 NOAA Fisheries Authorization to Use for GitHub Enterprise Cloud (NOAA internal)
- NOAA GitHub Checklist Note this checklist appears to be intended to apply to contributions to https://github.com/NOAAGov. References the GitHub Guidelines document above which is written for the NOAAGov GitHub organization.
- NOAA Scientific Integrity Policy (NAO 202-735D.2) website. final signed
- Fisheries Integrated Toolbox GitHub Guide
- Using GitHub at NOAA Practice and Policy PMEL presentation by Eugene Burger, PMEL.
- DOC Open Source Policy
- Federal Open Access Memo 2022
- GitHub Enterprise Fedramp authorized