Fork styleguide

This is the styleguide for the way Sean Patrick Myrick (@seanpm2001/@seanwallawalla) does Git/GitHub forks. This guide predominantly covers the GitHub style.

Prerequisites for reading

- Basic knowledge of Git or GitHub
- Medium knowledge of English
- Basic understanding of Seanpm2001 and Seanpm2001 projects
- 18 to 25 minutes of available time

Table of contents

There are 3 chapters. The general chapter is an overview of all of the rules and conditions that apply to all accounts. Chapter 2 is for forks to @seanpm2001 only, while chapter 3 is for forks to @seanwallawalla only.

- - Cannot fork a fork rule

- - Cannot fork an empty repository rule

- - Cannot easily fork an archived repository, but still done rule

- Don't copy the master branch only

- When GitHub deletes users who were forked from, classic times (stays a forked repository) and modern times (turned into source repository)

- Star and watch

- Almost all forks are unmodified

- <username>_<repository-name>

- Underscore problem (when they are in the original name)

- License disagreements

- When to clone

- Google, Amazon, and other enemy forks

- Template repositories

- DCO signoff

- LFS objects

- Mirrors

- Dead links

- Taking inspiration and learning

- As a warm-up exercise

- GitHub static content dying

- Leaving in spelling errors (it isn't my job to fix them)

- Rate of forking

- GitHub no longer counts forks as commits (2023.09.19)

- Preservation

- Offline archive

- Deleted contents in git history of forks

Chapter 2 – Forking to Seanpm2001

- Logging forks into data files

- Gathering forks

- Exceptions for what to fork

- Sanctions

- Fork gather tunneling and rabbitholes

- Following users before forking

- Organization fork process

- Prioritizing forks of already-starred, but not forked repositories

- Prioritizing forks of already-forked (by one of my former organizations) but not forked by my main account repositories

- When forks are modified

Chapter 3 – Forking to SeanWallaWalla

- Keep the name the same

- Fork once a day

- How forks were previously logged into data files

- How forks are now logged into data files

- Holding off forks for vacation times

- Forking non-Seanpm2001 repositories

- Going back to fork once-private repositories

- Why fork Seanpm2001 repositories

- Scope file

- Note file

Forking (general)

This is the guide for general forking information.

Why fork?

Repositories are forked as they are discovered via the archive rule. Every repository that I come across is forked. Additionally, when an organization is present, all organization repositories are included, and additionally, the source of their forks is also forked, which can lead to a rabbit-hole of organizations being covered.

Name rules

There are several rules for naming forked repositories.

.github rule

Repositories with only the name `.github` don’t have the underscore rule, they will be named as `username.github`

.github.io rule

Repositories with the `username.github.io` name scheme are not renamed at all. Note that the older `username.github.com` naming scheme does not apply to this rule. See below

.github.com rule

Repositories with the `username.github.com` name scheme are renamed as `username_username.github.com`

.com rule

Repositories with the `username.com` / `name.com` name scheme are renamed as `username_username.com` or `username_name.com`

.net rule

Repositories with the `username.net` / `name.net` name scheme are renamed as `username_username.net` or `username_name.net` just like .com and .org URLs, unless it is particularly noted that Microsoft Dot NET is in use.

DotNET rule

If a repository ends in `dotnet` or consists only of `dotnet` the repository is renamed as `username_projectname_dotNET`

Other website repository name rules

There currently aren’t any other website repository name rules.

Capitalization rules

These are the rules for capitalizing repository names.

CamelCase and hyphen-case rules

CamelCase is most common in pre-existing names, but hyphen case is used to separate existing words over the usage of an underscore (unless it is in the repository name already, more on this later in the document)

MacOS, iOS, and other name rules

MacOS, iOS and some other names have special capitalization rules
macos → MacOS
ios → iOS
ipados → iPadOS
reactos → ReactOS
iot → IoT
No other examples listed.

Capitalize Every Word rule

In forked repository names, every word has its first letter capitalized, with the exception of brand names like iOS and iPadOS

LIBraries rule

The word Libraries is usually capitalized as LIBraries as part of my effort to normalize people to the word LIB meaning library, a form of learning through osmosis.

Usernames

I try to enter usernames according to how they are spelled, or how the user says they spelled. They are commonly copy and pasted to prevent errors. If no spelling guide is found, I will either guess, or only capitalize the first letter of the username, or (if it starts with a number or hyphen) capitalize every letter.

Cannot fork rules

These are the rules for what I cannot fork.

Cannot fork a fork rule

I cannot fork a forked repository, as this comes into conflict with the source repository. If I ever fork and star the source repository, I will be unable to fork any of its forks. So when repositories are came across are merely forks, I try to go for the source only. In some cases, if I find the repository significant enough, I will clone the repository as a source repository to my account, with the same forked repository name rules.

Cannot fork an empty repository rule

I cannot fork empty repositories, as GitHub doesn't allow it. If I was able to, empty repositories would still be forked, as there is a repository name involved. Empty repositories are usually just logged as empty.

Cannot easily fork an archived repository, but still done rule

I am unable to easily fork archived repositories, but I still do so. Usually, I would just go to create a new file, and have the fork created this way. Since you can't do that in an archived repository, I instead press the fork button, turn off "Copy the master branch only" and then continue the fork process.

Don't copy the master branch only

GitHub has an option to copy only the master branch. After how many times I have deselected this, and for the 14+ years this wasn't a thing, it should register that I don't want to copy only the master/main branch.

When GitHub deletes users who were forked from, classic times (stays a forked repository) and modern times (turned into source repository)

In the classic GitHub times (before 2022) when GitHub deletes a user who you forked a repository from, you could still give credit to the user, because it would still show where you forked it from, even though this would be a dead link, it was still very helpful. Unfortunately, in 2022, this ability was removed, and now forked repositories are converted to source repositories when a user is deleted.

Star and watch

All repositories that are forked are also starred and subscribed to (watched) I try to star and fork on the same day, so that when I come across the repository later, I will know it has been forked. This forking rule hasn't always been present, and I commonly have to go back and fork repositories that were

starred, but not forked. In my star notes, a comment is added: `//// ALREADY STARRED` and in my fork notes, this comment is added: `///// BUT NOT FORKED UNTIL TODAY`

Almost all forks are unmodified

Almost all forks I create are never modified. There are rare cases in which I will work on a fork and submit a pull request/keep working on my fork, but this happens less than 0.1% of the time.

<username>_<repository-name>

I use underscores to distinguish a username from a repository name. The standard name format is `<username>_<repository-name>`

Underscore problem (when they are in the original name)

As said above, I use underscores for distinguishing names. Unfortunately, some repositories already use underscores in their names, so I have to include existing underscores. To continue distinguishing, just remember that everything after the first underscore is the repository name.

License disagreements

I fork repositories regardless of license, even if it is with a license I have strong disagreements with (such as the Apache license) I prefer the GPL3, GPL2, and Vim licenses, but exceptions had to be made to fit with the archive rule.

When to clone

Repositories are only cloned when a fork is notable enough to be separated, and when the source cannot be found.

Google, Amazon, and other enemy forks

I commonly fork the repositories of my enemies (Google, Amazon, Facebook, etc.) it does not mean that I support them. It is just part of the archive process.

Template repositories

Template repositories are forked, and their status as a template is not disabled.

DCO signoff

Repositories with a DCO signoff rule are still forked, and this rule isn’t disabled.

LFS objects

Repositories with LFS objects are still forked.

Mirrors

I fork mirrored repositories when they are in my queue.

Dead links

When dead links are found, they are added to a dead link list.

Taking inspiration and learning

When doing the forking process daily, I take inspiration from the repositories I fork, and learn things from them.

As a warm-up exercise

The forks are commonly done as a daily warm-up exercise before I get to my actual work. It is a routine, and part of my work, but it still helps me get going.

GitHub static content dying

GitHub static content dies over time, but unfortunately, I don’t have the time to save it.

Leaving in spelling errors (it isn't my job to fix them)

I leave in spelling errors in the names of forked repositories for preservation purposes. It isn’t my job to fix them, but nonetheless, errors are kept.

Rate of forking

My rate of forking has become consistent since some time in 2021 or 2022. I try to create 25 repositories per day, so usually 17 to 25 forks are created (so there is room for 0-8 source repositories) Starting 2023, September 10th, I have been creating 100 repositories per day, meaning I create 92 to 100 forks per day (with room for an additional 0-8 source repositories)

GitHub no longer counts forks as commits (2023.09.19)

As of 2023, September 19th, GitHub stopped counting my forks as commits per day on the commit calendar. I feel this was due to the huge increase in daily forks.

Preservation

Forks are done for preservation purposes.

Offline archive

Forks are done to create a sizable, offline, searchable database of Git content Sean has seen. It is a part of the Senapm2001 Life Archive project.

Deleted contents in git history of forks

Unfortunately, a lot of developers have the strategy of deleting all of the contents in an old repository when transitioning to a new one, instead of simply archiving it and leaving a note. Luckily, with Git file history, almost all of the repository is still there (with the exception of large files and binary files, such as images, videos, audio, rich documents, etc.)

Forking to Seanpm2001

These are my rules for forking repositories to the @seanpm2001 GitHub account. In the archive process, this is known as GitHub forks part A.

Logging forks into data files

Forks are logged into data files, which became a common practice in mid 2022. Forks are logged in URLL format. The format is

`seanpm2001/original-fork-name -> seanpm2001/User42_Original-Fork-Name`
The arrow is important. In some instances, there are multiple arrows for a single fork entry, as I make errors sometimes, and have to correct them.

`seanpm2001/original-fork-name -> seanpm2001/Usr42_Orignal-fork-Name -> seanpm2001/User42_Original-Fork-Name`

This is the standard process for repository fork data files.

Gathering forks

To gather forks to work with over days, weeks, months, and years, entire organizations are searched, and random repositories are discovered and logged into a secret file known as PIN which is a personal blockchain of notes (not a cryptocurrency blockchain, just a very large text file that is full of dated entries, and updated frequently) repositories from users are usually put into a miscellaneous queue, while organizations are added to a dated queue, with 17-100 entries listed per day. When the time comes, they are all opened, then starred, forked, renamed, and logged. This is a process that is done on a daily basis.

Exceptions for what to fork

There are exceptions for what to fork. The exceptions are:
- Empty repositories
- Private/deleted repositories
- Forked repositories
- Sanctioned repositories

Sanctions

Some organizations may be sanctioned before information on them can be gathered. Currently, there are only 2 sanctioned organizations: Alibaba and Bilibili. These were sanctioned due to special aid being given to an aggressor in a war (giving support to Russia in the 2022 Russian Invasion of Ukraine) they were already in a bad spot to begin with (Chinese SaaSS, personal data harvesting, spying, and censorship)

Fork gather tunneling and rabbitholes

Forks are gathered organization by organization. When an organization has a fork that is of another organization, the path will continue, and that organization will be gathered as well. This can easily lead to rabbit holes, as multiple organizations will be documented, and it will go from organization to organization, sometimes leading to 100+ organizations being documented from a single organization.

Following users before forking

Users and organizations are always followed before forking their repositories. If this isn’t done, they will be followed immediately after. This is a newer rule from mid 2022, and some users from projects that were starred and forked prior to 2022 are still being discovered again.

Organization fork process

The organization fork process goes as follows
- Find an organization

- Gather its repository links
- Schedule it into a queue
- Wait for the day to come
- Star, fork, and log the repositories

Going back to fork starred repositories

I have to go back and fork repositories that were previously starred, but not forked.

Prioritizing forks of already-starred, but not forked repositories

I have been prioritizing forks of already-starred (but not forked) repositories as an extra task to do whenever comfortable and possible (when there is room available)

Prioritizing forks of already-forked (by one of my former organizations) but not forked by my main account repositories

Most currently starred, but not forked repositories are related to my former GitHub organizations, where the fork is still retained (but just returns 404) these are commonly discovered, and are prioritized in the miscellaneous queue.

When forks are modified

When to-be-forked repositories are modified before I get to them, and something bad happens (source code rewrite, file deletion) they are still forked.
I occasionally modify forked repositories for use in pull requests, or for general development (extremely rare)

Forking to SeanWallaWalla

These are my rules for forking repositories to the @seanwallawalla GitHub account. In the archive process, this is known as GitHub forks part B.

Keep the name the same

The names of repositories of seanpm2001 repositories are kept the same, and are logged as just `seanwallawalla/forkname`

Fork once a day

Seanpm2001 repositories are forked once per day. I create new repositories on @seanpm2001, and fork them to @seanwallawalla the next day.

How forks were previously logged into data files

Forks were previously logged into a separate data file (the note file)

How forks are now logged into data files

Forks are logged into the same data file as the Seanpm2001 fork file. They are in a section near the bottom labeled SeanWallaWalla.

Holding off forks for vacation times

Some forks are held off from normal development days to be added to a vacation queue. On vacations, I try to keep the forks going on a daily basis, and I begin to dig into this queue, at a slower rate (normally 2-4 repositories per day)
Before going on vacation, I purposefully hold off some daily forks to put into a vacation queue, as I cannot create new repositories on vacations.

Forking non-Seanpm2001 repositories

In the beginning, I used to sometimes fork non-Seanp2001 repositories, as to populate my alt account, and promote other projects. This is done extremely rarely now.

Going back to fork once-private repositories

Some repositories are created as private repositories, and are made public later on. I occasionally go back and fork projects after I make them public again.

Why fork Seanpm2001 repositories

Seanpm2001 repositories are forked, as a way of providing a backup. The motivation to fork my repositories daily started in 2022 August, after the suspension of my main account was listed, which lasted nearly a month. Seanpm2001 repositories are forked to Seanwallawalla, so that if my main account is suspended, there will still be a way to view my projects without going to another site.

Scope file

I used to maintain a scope file for Seanpm2001 projects, built off of data from SeanWallaWalla forks. Currently, this file is unmaintained.

Note file

I used to maintain a note file for SeanWallaWalla forks, but it stopped daily development after a severe kernel panic in 2023 June. The file contained information on forks, what would be forked next, and information about them. This is now maintained in the PIN file.

File info

File version: 3 (2023, Monday, September 25th at 10:21 pm PST)

Page count: 10

Word count: 2,943

Character count: 17,644
