How to contribute to the Gen3 source code

Gen3 Community Forum
20 March 2024
The Agenda

● Introduction
● Open-source principles
● How to engage with Gen3
  ○ Ask questions
  ○ Submit issues in GitHub
  ○ Submit pull requests in GitHub
  ○ Collaborate
● Examples of past external contributions
● Q&A
Gen3 Open Source Principles
Gen3 Open-source principles

The Gen3 software is open source, which means:

- All source code is available on GitHub
- You may copy and modify the code however you wish (subject to permissive Apache 2.0 license)
- The maintainer organization, Center for Translational Data Science at the University of Chicago, would like to hear from you about bugs or feature requests in order to guide the development of Gen3
- CTDS accepts code contributions via Pull requests. PRs are then reviewed and accepted or rejected.
- There are limitations on what code we can accept due to concerns about security, maintainability, and lack of alignment with product vision
How to engage with the Gen3 Team
Ask questions

- It can often be more efficient to start a conversation on Slack when you first encounter an issue (e.g. bug) or have a question about whether Gen3 supports a specific use case. You may also reach out via a forum post or email.
- #gen3_community includes CTDS maintainers as well as community members
- You don’t have to start a conversation before submitting an issue or pull request, but it may be helpful
Submit issues in GitHub

- Every Gen3 repo has an option to submit an issue, which could be a bug report or feature request.
- If you are not sure which repo is appropriate (there are a LOT of Gen3 repos) then please use gen3_community.
- Look for existing issues before you post!
- **DO NOT post any Personally Identifiable Information (PII) in a GitHub Issue.** This includes IP address, hostnames, and tokens.
Submit issues in GitHub

- Once created, the issue will be reviewed by a member of the CTDS staff. Initial status will be ‘waiting for triage’. After that the status will depend on whether it is under active development, requires further review, requires additional information, or eventually closed.
- Internal to CTDS a JIRA ticket will be created and tracked.
Submit a pull request in GitHub

If you know what code changes are needed please submit a pull request:

1. Note that all code contributions are subject to our Apache 2.0 license attached to each of our repositories ([https://www.apache.org/licenses/LICENSE-2.0](https://www.apache.org/licenses/LICENSE-2.0)).

2. Should you create an issue first? Not necessarily - just create the pull request and use the description to provide context and motivation, as you would for an issue.

3. Always check out the main branch and submit pull requests against it. PRs must include creating or updating unit tests as needed.

4. Run all relevant tests against your modified code before you submit a pull request. This will require that you have a version of Gen3 up and running.
5. Choose the granularity of your commits consciously. Squash commits that represent multiple edits or corrections of the same logical change.

6. Write clear and informative commit messages.
   - The Gen3 Contributor Guidelines includes references for best practices for streamlining commits and writing commit messages.

7. If there is a related GitHub issue, reference the issue number in the description of the pull request.

8. Once your PR is made, a CTDS staff member may reach out with additional questions and/or comment directly on your PR in GitHub.
Submit a pull request in GitHub

- PRs (and issues) are tracked via GrimoireLab and individually in JIRA. Each PR will be discussed and reviewed by the engineering team.
- If your PR is not approved, we’ll provide feedback about the rationale and possible steps to get it approved. Reasons for not approving PRs include:
  - Does not follow Gen3 coding conventions
  - Inconsistent with Gen3 product vision
  - Includes new features or packages that would be hard to maintain
  - Inconsistent with security requirements
- If accepted, your contribution may be heavily modified as needed prior to merging. You may also be asked to rework the submission.
- If asked to make corrections, simply push the changes against the same branch, and your pull request will be updated.
How to track submitted issues or PRs

- Review status in GitHub. It will change over time as it is reviewed and acted upon.
- We are aiming to at least provide an initial review of PRs and Bug Reports relatively quickly and give feedback including a quick rejection if appropriate.
- Reach out to Michael Fitzsimons or other staff member you have been working with on the ticket who may be able to offer additional status updates.
Collaborate with Gen3 Team

- While PRs do not have a hard limit in terms of complexity, practically speaking larger integrations will require significant coordination and discussion with the Gen3 Team.
- If you have a larger improvement to Gen3 in mind please reach out to discuss rather than spending significant time on a PR that could later be rejected.
- In the future we will plan to organize working groups of external contributors that will collaborate with the Gen3 Team on larger initiatives.
Example Gen3 Contributions
Use file_name provided in manifest


- Initial issue
  - Some Gen3 upload commands required the local file name to be the same as the ObjectID in the upload manifest in order to validate file and upload data.
  - Part of a broader PR that adds an option to select a specific bucket for data uploads.

- Expected outcome
  - Files can be uploaded that are named based on the file_name attribute rather than the ObjectID
Delete files support for non-AWS S3 compatible systems

- [https://github.com/uc-cdis/fence/pull/1127](https://github.com/uc-cdis/fence/pull/1127)
- Initial issue
  - When deleting indexd records in the Gen3 SDK from non-AWS S3 buckets (like MinIO), we encounter an "AWS Access Key Id you provided does not exist" error. The `delete_file_locations()` method should handle deletion for all S3-compatible buckets.
- Back and forth
  - Adding unit test
  - Backwards Compatibility
Arborist “POST /auth/mapping” token support

● Initial issue
  ○ The need for “client_credentials” access token support in Guppy:
    Tokens that are linked to an OIDC client and not to a user.
  ○ The Arborist authorization endpoint used by Guppy does not accept a token as input.

● Updates
  ○ Arborist’s “POST /auth/mapping” endpoint now accepts access tokens as input, including “client_credentials” tokens.
    https://github.com/uc-cdis/arborist/pull/159
  ○ Guppy now calls Arborist’s “POST /auth/mapping” endpoint.
    https://github.com/uc-cdis/guppy/pull/254

● Back and forth
  ○ Adding unit tests
  ○ Documentation
  ○ Backwards compatibility
  ○ Consistency in Arborist behavior when the user is unauthenticated
Discussion
- **Gen3-community GitHub Repo** - for submitting issues and feature requests that span multiple Gen3 repos or you are not sure about the appropriate service/repo
- **Gen3 contributor guidelines**
Acknowledgements

● **Speakers**
  ○ Robert Grossman
  ○ Michael Fitzsimons
  ○ Chris Meyer
  ○ Peter Vassilatos
  ○ Binam Bajracharya
  ○ Pauline Ribeyre

● **Gen3 Forum Steering Committee**
  ○ Robert Grossman - Center for Translational Data Science, University of Chicago
  ○ Steven Manos - Australian BioCommons
  ○ Claire Rye - New Zealand eScience Infrastructure
  ○ Plamen Martinov - Open Commons Consortium
  ○ Michael Fitzsimons - Center for Translational Data Science, University of Chicago