Participate in Hydra-in-a-Box Development

So, you want to help develop the technologies used in the Hydra-in-a-Box project? Great! If you haven’t done so before, you may have some questions about how to get started. This FAQ collection contains questions commonly asked by developers new to the Hydra-in-a-Box project or new to the Samvera community work cycles.

Prerequisites

First, you’ll want to get in touch with Michael J. Giarlo (Technical Manager of the Hydra-in-a-Box project) who will help coordinate your participation. The Hydra-in-a-Box team will work with you to make sure you have access to all the shared resources you need to participate, including access to:

- Appropriate GitHub repositories (see below)
- Planning documentation on the Samvera wiki and Google Docs
- Samvera Slack

If you don’t yet have accounts for Google Docs, the DuraSpace wiki, or GitHub, you’ll want to sign up for each.

If you’ll be participating in a community work cycle, we’ll also send you calendar appointments for daily standups, sprint retrospectives, and sprint planning meetings.

GitHub Repositories

Hyku (the Hydra-in-a-Box repository application) is based on Samvera community components. Most of the work that we do in our sprints are on Hyku and Hyrax, though we also touch a number of other Samvera components when necessary. Hyku is a Rails application based on Hyrax.

- https://github.com/samvera-labs/hyku
- https://github.com/samvera/hyrax

Development Tools and Practices

Since Hyku is based on Hyrax, development of Hyku tends to follow the same practices and use the same tools as Hyrax. A good starting point would be to review the Hyrax Development Guide and the Hyrax README. These will give you an idea of what tools and dependencies you’ll need installed and help you get started running the test suite. Note that there are differences between working with a Rails engine like Hyrax and working with an application like Hyku.

There is also an in-progress Samvera Technical On-Boarding document, which provides details about how the Samvera community communicates, how we track issues, code guidelines, and the code review process.

What database do I use?

Unlike Hyrax, which permits you to use any database management system supported by Rails, Hyku has a hard dependency on PostgreSQL. Why? Hyku uses the Apartment gem to enable multi-tenancy and that gem prefers Postgres. In practice we’ve found Postgres as easy to work with and just as robust as MySQL (if not more so!).

How do I build / run Hyku locally?

1. Hyku is a Hyrax-based application, so you’ll need to start out by ensuring you have all the necessary Hyrax prerequisites installed. Generally speaking, you should also review the Hyrax installation:
   - Install the Hyku codebase itself (e.g. by cloning https://github.com/samvera-labs/hyku)
   - Install all gem dependencies: bundle install
   - Create your database (should be PostgreSQL as noted above): rake db:create
   - Run all database migrations: rake db:migrate
   - Start servers (which starts Solr, Fedora, and Rails together): rake hydra:server
     a. Alternatively, if you are doing a lot of development, you may find it easier to start/stop each of these individually by running each of the following in different terminals:
        i. solr_wrapper (Starts Solr on port 8983)
        ii. fcrepo_wrapper (Starts Fedora on port 8984)
        iii. rails server (Starts Hyku on port 3000)
   7. Ensure you have loaded the Hyrax Workflows and created the default Hyrax Administrative Set. Currently, these are not run automatically, and the default AdminSet is required before you can add Works.
      a. rake hyrax:workflow:load
      b. rake hyrax:default_admin_set:create
      a. To create an initial administrator account, simply register with a new login/password.

How do I know what issue to work on?
That depends on a series of factors, including what the team is currently focused on – we do sprint planning and goal setting as a team – and what you or your institution are interested in. You should first take a glance at the Hydra-in-a-Box project’s issue tracker. The Waffle-based tracker shows you issues from Hyku and Hyrax, and issues are divided into five columns:

- **Done**: These issues are done, so you can get a sense of what the team has been working on recently. **Don’t pick an issue from this column.**
- **In Progress**: These issues are currently being worked on. **Don’t pick an issue from this column.**
- **Ready**: These issues are ready to be picked up, and are usually in rough priority order. **Review this column for issues of interest and, ideally, select one from near the top of the column.** If there are items of interest in this column that aren’t near the top, chat with Michael J. Giarlo and he can give you a sense of how appropriate an issue is for the current sprint.
- **Design Needed**: These issues need (conceptual, technical, or UI/UX) design. **If you are interested in design or have design skills to contribute to the team, feel free to help with one of these tickets.** Do let Michael J. Giarlo and Hannah Frost know you’re interested in helping with design.
- **Backlog**: These issues are not a high priority for the current sprint. **Don’t pick an issue from this column.** If you’re interested in an issue in this column and think it should be considered in-scope, chat with Michael J. Giarlo and Hannah Frost.

I’ve chosen an issue to work on! Now what?

When you find an issue you’d like to work on, please do assign yourself to the issue in GitHub. This is an important step that signals to other developers that you’re working on the issue and that they shouldn’t pick it up too.

It is also recommended to link to the GitHub branch where you have begun your work (once you’ve begun). That gives others the opportunity to review your work early.

**How do I run the test suite to make sure changes I made work?**

In general you will run Hyku’s tests the same way as you’d run Hyrax’s, but there’s one major difference: since Hyku is a Rails application and not an engine, there’s no such thing as a “test application” because we already have an application. Assuming 1) you’ve got the code checked out and 2) you have an up-to-date Ruby (2.3.1 is recommended) and 3) you’ve run `bundle install` to pick up all of Hyku’s dependencies, you have a couple steps remaining.

1. First, start up Solr and Fedora per the Hyrax Development Guide. Keep in mind they both should be running on different ports than you normally use for development:
   a. Start Solr on port 8985 for testing, for example: `solr_wrapper -v -d solr/config/ -n hydra-test -p 8985`
   b. Start Fedora on port 8986 for testing, for example: `fcrepo_wrapper -v -p 8986 --no-jms`
2. Then you should be able to run the test suite.
   a. For example, to run all tests use: `rake spec`

**How do I contribute my changes to the project?**

Generally speaking, we follow the guidelines / recommendations in samvera Technical On-Boarding regarding both Code Guidelines and Code Reviews.

1. Create a pull request in GitHub. We recommend creating one as early as you are comfortable doing so, as it allows others to begin to give feedback/suggestions on your work. If your work is still “in progress”, please prefix the PR’s name with `[WIP]`. That way others know it’s not complete, but feedback is welcome.
   a. You may choose to create the PR from either a branch within your own fork of the project, or alternatively create a new branch in the main project (the latter would let others contribute to your PR more easily, as needed).
2. Before pushing any new commits to your pull request, you may wish to do the following:
   a. Run `rubocop` to verify the well-formedness of your Ruby code
   b. Please ensure your new code comes with RSpec tests (as code coverage will be auto-verified by Coveralls when you update your PR), and that all tests currently pass (see “How do I run the test suite...” above).
   c. Once you finish your work, we recommend squashing all changes down to a single commit. You are welcome to keep commits separate while you are still working though, if it’s easier.
3. If any of your PR changes depend on brand new code in Hyrax, then you will need to ensure Hyku’s Gemfile.lock gets updated in the codebase. This file controls the version of all gems (including Hyrax) that are currently in use by Hyku.
   a. Just run `bundle update` AND commit the changes made to your local Gemfile.lock
   b. NOTE: if you forget to update the Gemfile.lock file in your PR, then Travis CI will not pick up the latest versions of Hyrax, and therefore may throw unexpected errors.
4. Once your PR is ready for review, remove the `[WIP]` prefix from your PRs name and ping the development team (on Slack or GitHub).
   a. You are encouraged to ask for early review while your work is still in progress especially if you haven’t had discussed the technical design of the work you’re doing. Other developers may have hints/tips on easier ways to achieve your goal, or help note improvements that may be necessary before the PR can be merged.

**People and Institutions**

https://wiki.duraspace.org/display/samvera/Hydra-in-a-Box+Project+Team

**Resources**

You may find the following project resources helpful to keep around:
• Wiki pages: https://wiki.duraspace.org/display/samvera/Hydra-in-a-Box
• Website/blog/FAQ: http://hydrainabox.samvera.org/
• Issue tracking: https://waffle.io/samvera-labs/hyku
• Hyku UI Mockups: https://ggeisler.github.io/lerna-dashboard/admin.html
• Videoconference for team standups and sprint meetings: varies each work cycle. For August-November 2016, it's https://bluejeans.com/4326546636/