Hyku and Hyrax: How are they related and how are they different?

Overview

Hyku is the polished, turnkey, feature-full repository application product created by the Hydra-in-a-Box project in response to the call by the Institute of Museum and Library Services for a National Digital Platform. Hyku offers the Samvera community's best of breed features, using commonly adoptable components wherever possible in the interest of interoperability and maintenance ease, and at the same time serves as a staging ground for some of the latest innovations developed in Samvera's growing, collaborative, well-supported software framework.

Hyku is functionally competitive with other top-tier, general purpose, open source repository products. The components at the core of Hyku's capabilities are Hyrax, Fedora 4 and the Portland Common Data Model. Hyku has been optimized for a scalable, hosted service implementation with support for multi-tenancy and a Super Administrator's interface. Hyku is cloud-provider agnostic; CloudFormation templates are available for ready use on Amazon Web Services. Hyku has also been optimized for the Digital Public Library of America's national network of content contributors by publishing harvestable metadata that easily crosswalks to the DPLA profile.

The Technical Perspective

Hyku is a Rails application generated from Hyrax (the Samvera community's follow-on effort to Sufia and CurationConcerns). All of its features come from Hyrax with a few exceptions. Currently these exceptions are: multi-tenancy, IIIF Image & Presentation API support, the Universal Viewer, and bulk import scripts.

Furthermore a number of technical features distinguish Hyku from Hyrax:

- In providing a user interface for easy configuration, Hyku is more opinionated in its customization options. (Advanced users should use Hyrax to generate their own applications that are open to code customization, as forking Hyku will be harder to maintain.)
- It is easier to keep dependencies pinned more tightly in Hyku.
- Hyku's user interface may diverge from Hyrax's in ways the Hydra-in-a-Box team finds appealing; the Hydra-in-a-Box team expects to continue designing and refining the Hyku UI over time.
- Hyku is opinionated about its choice of database -- currently PostgreSQL -- for compatibility with the software component we use for multi-tenancy.
- Hyku is opinionated about content types, currently shipping with two types (Generic Work and Image) and more to come.
- Hyku uses Fedora-supplied UUIDs for object identifiers instead of NOIDs.
- Hyku is easily deployable to Amazon Web Services using CloudFormation templates and every commit in the Hyku code repository's master branch triggers an automatic deployment to an AWS-hosted demo instance.
- Hyku is deployable locally via Docker Compose.

Faced with the decision to implement Hyku or Hyrax, there are key factors to consider. Hyku has been developed with an orientation toward the provision of hosted services. As such the development focus has been on multi-tenancy and making deployment, configuration, and maintenance easier and less resource intensive. Hyrax offers most of the same features as Hyku, and is the primary gem where new development is taking place within the Samvera community. Generally Hyrax is deployed locally and requires some degree of expertise in system administration as well as Ruby and Rails development to operate and maintain. Hyrax is more amenable to local customizations; when customized, the need for an ongoing commitment to maintain and migrate the repository, its contents and its customizations is heightened.

In terms of a maintenance and enhancement strategy, the Hyku developers seek to locate code in the best possible codebase for the sake of long-term maintenance and general benefit to the Samvera community. Code will be added to Hyrax, to other gems lower in the stack, such as ActiveFedora, or extracted to plugins. When the code can't be located in Hyrax or another gem, we will put it in Hyku.