Fulcrum (github) recently upgraded to Hyrax 4 in order to move away from Rails 5 which was end-of-life. Having kept up with upgrades since 2016 from Curation Concerns 0.9, this is the longest upgrade we've done taking over 300 hours. Fulcrum predates Hyku (and Hyrax itself really) and therefore has its own "multi-tenancy" solution meaning we have over 50+ individual "themes" (presses/collections) that needed to be converted to Bootstrap 4 which was the bulk of the effort. While the Samvera community is mostly engaged with Hyrax 5 (rightly so) we hope these notes might help other longer running Hyrax installs in getting through Hyrax 4.0 on their way to 5.0.

Getting specs to pass with Hyrax 4.0

Generally when we do a Hyrax update the largest part of the job is getting our specs to pass due to the amount of customizations we've done in Fulcrum over the years. Because this time the Hyrax 4 update primarily involved new Blacklight and Bootstrap, and we generally don't write view specs, getting the specs to pass wasn't where we spent most of the effort. As is usual when we do an upgrade, as we go through and fix the specs we create a running list of notes of debatable usefulness. The notes include getting feature specs to pass so touch on blacklight 7 and bootstrap 4 updates as well as Hyrax itself.

Spec development notes, "Fulcrum: Hyrax 3.5 to Hyrax 4 Specs"

The Blacklight 7 upgrade

A number of years ago we added quite a few accessibility fixes to Fulcrum and in the interest of expediency copied quite a bit of Blacklight directly into Fulcrum to add the a11y fixes locally. Predictability, this became a problem when updating to a new major version of Blacklight. We've made other choices with regard to Blacklight such as a thing that acts and looks like a facet that is not that caused some pain too. Since blacklight now uses view components we needed to learn that technology in order to support our local changes. Blacklight 7 also has presenters, a good thing, but it caused problems with our overrides that used the older helpers exclusively and needed to be updated. Supporting our local a11y changes, using view components and switching to presenters all took some time to sort out but generally seemed like good things for our project.

The Bootstrap 4 upgrade

Fulcrum just has a lot of css and custom templates. Whole areas of the application exist that used Bootstrap 3 that are not in Hyrax but needed to be upgraded alongside the Hyrax pieces to get everything aligned with Bootstrap 4. We're actually still working on a few of the less egregious non user facing pieces that we'll get to over time. We originally attempted to hand the

entire bootstrap 4 upgrade over to a “front end” team member but in the end it was an all hands situation with everyone either modifying code or reviewing various parts of the application that got things done. There was really just too much for one person to handle all at once. Bootstrap has pretty good migration documentation that we leaned hard on. Due to the specificity of Fulcrum’s code, listing out all the changes isn’t helpful but we have a couple of documents that suggest the scope of the changes we needed to make.

"Notes on Hyrax 4 Upgrade: Bootstrap 3 to Bootstrap 4"
"QA for Hyrax 4 Upgrades"

Skylight, Rails 6 and the Hyrax 4.0 Actor stack

Some time ago we were able to get Skylight for Open Source which has been pretty great for general monitoring and zeroing in on performance problems. So when we discovered that skylight was causing problems with the actor stack, meaning no creating/updating/deleting Works (an obvious deal breaker) we were originally resigned to disabling skylight in Fulcrum. However in looking at it again we came up with a partial solution, essentially disabling skylight only for the Actor stack itself.

I suspect that this is a problem with Rails 6, ActionDispatch::MiddlewareStack::InstrumentationProxy and how the Actor stack works. In Hyrax 3.5 with Rails 5.2 the Actor stack looks like this:

```
[1] pry(main)> Hyrax::CurationConcern.actor_factory.build
=> #<Hyrax::Actors::OptimisticLockValidator:0x0000560f5a99bca8
 @next_actor=
  #<HeliotropeActor:0x0000560f5a99bcf8
   @next_actor=
    #<RegisterFileSetDoisActor:0x0000560f5a99bd48
     @next_actor=
      #<CreateWithImportFilesActor:0x0000560f5a99bd98
       @next_actor=
        #<Hyrax::Actors::CreateWithRemoteFilesActor:0x0000560f5a99bde8
         @next_actor=
         …
```

In Hyrax 4 with Rails 6 it’s this:

```
[1] pry(main)> Hyrax::CurationConcern.actor_factory.build
=> #<ActionDispatch::MiddlewareStack::InstrumentationProxy:0x00005570777d2fe8
 @middleware=
  #<Hyrax::Actors::OptimisticLockValidator:0x00005570777d3038
   @next_actor=
       #<ActionDispatch::MiddlewareStack::InstrumentationProxy:0x00005570777d30d8
       @middleware=
       #<HeliotropeActor:0x00005570777d3150
```

```
       @next_actor=
#<ActionDispatch::MiddlewareStack::InstrumentationProxy:0x00005570777d31c8
       @middleware=
       #<RegisterFileSetDoisActor:0x00005570777d3268
              @next_actor=
#<ActionDispatch::MiddlewareStack::InstrumentationProxy:0x00005570777d3308
              @middleware=
              #<CreateWithImportFilesActor:0x00005570777d3358
              @next_actor=
#<ActionDispatch::MiddlewareStack::InstrumentationProxy:0x00005570777d3420
              @middleware=
                     #<Hyrax::Actors::CreateWithRemoteFilesActor:0x00005570777d34c0
                     @next_actor=
                     …
```

The ActionDispatch::MiddlewareStack::InstrumentationProxy is inserting itself between the Actors in the stack. This is a good thing for observability and Skylight makes use of it. However, it totally breaks the Actor stack and the next_actor chain. Skylight is supposed to allow modifications of it's probes/middleware, one should be able to add configuration with one of these:

```
config.skylight.probes -= ['middleware']
config.skylight.probes -= ['middleware', 'action_dispatch']
```

However in practice we couldn't get it to work. In the end we overrode https://github.com/samvera/hyrax/blob/hyrax-v4.0.0/app/services/hyrax/default_middleware_stack.rb changing:

```
def self.build_stack
  ActionDispatch::MiddlewareStack.new.tap do |middleware|
    # Ensure you are mutating the most recent version
    middleware.use Hyrax::Actors::OptimisticLockValidator
```

To:

```
def self.build_stack
  ActiveSupport::Notifications.unsubscribe('process_middleware.action_dispatch')
  ActionDispatch::MiddlewareStack.new.tap do |middleware|
    # Ensure you are mutating the most recent version
    middleware.use Hyrax::Actors::OptimisticLockValidator
```

We're turning off ActiveSupport::Notifications in the actor stack so InstrumentationProxies don't get inserted into the stack. This works ok. Skylight still works for what we want it to do, which is monitoring performance on user facing routes. We've never really used Skylight with the Actor stack anyway. Most of that activity is in background jobs.

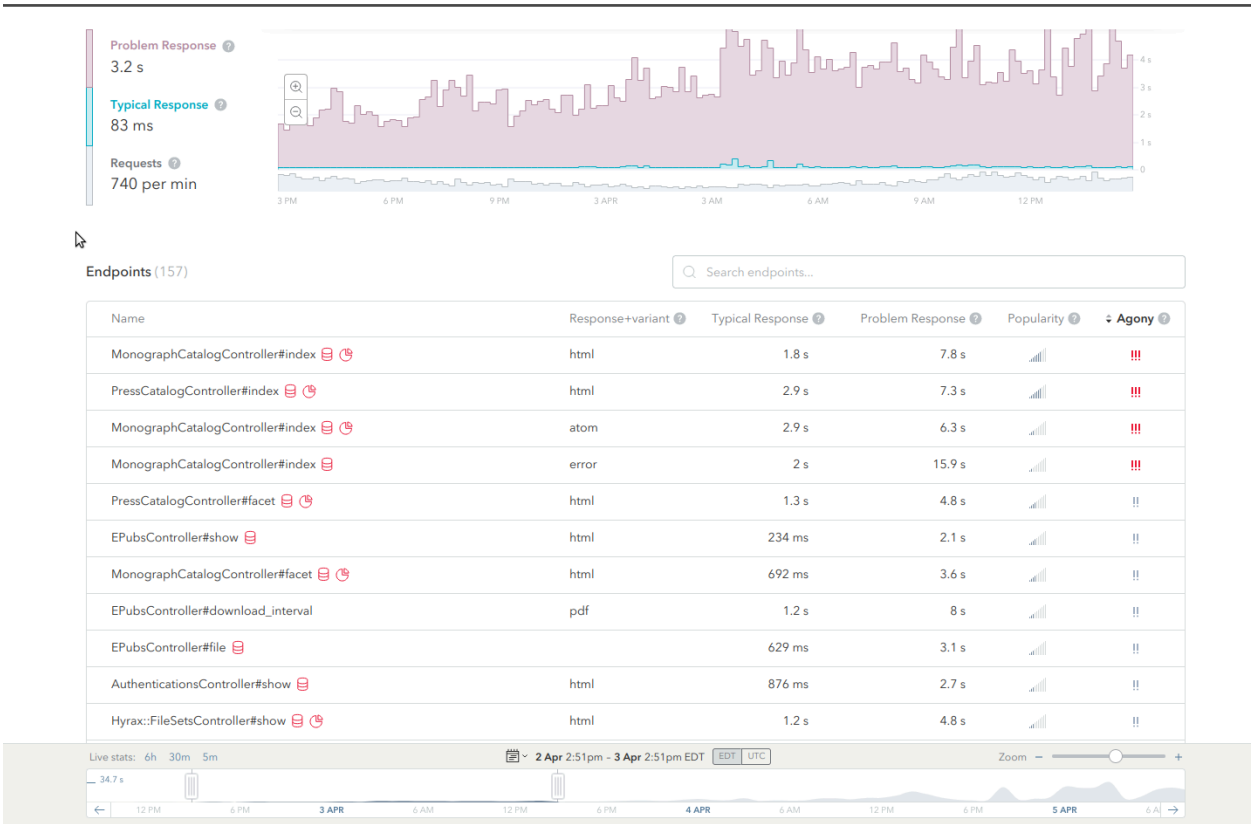
Blacklight deprecations and performance issues in production

It was good that we were able to get Skylight working. It helped diagnose some performance problems we were seeing with Blacklight 7. Above, in “The Blacklight 7 upgrade” I mention that we switched some code over to use the new view components and presenters. At the time of release of Fulcrum 4.0 (we historically version Fulcrum releases based on major Hyrax releases) this was less true. Some of our Blacklight overrides needed to be updated, but a lot of it we left alone and simply silenced the many deprecation warnings thinking we’d address them later. There were pretty big problems with that approach. Here’s what catalog/_document.html.erb in Blacklight 6 vs. Blacklight 7 rendering looks like

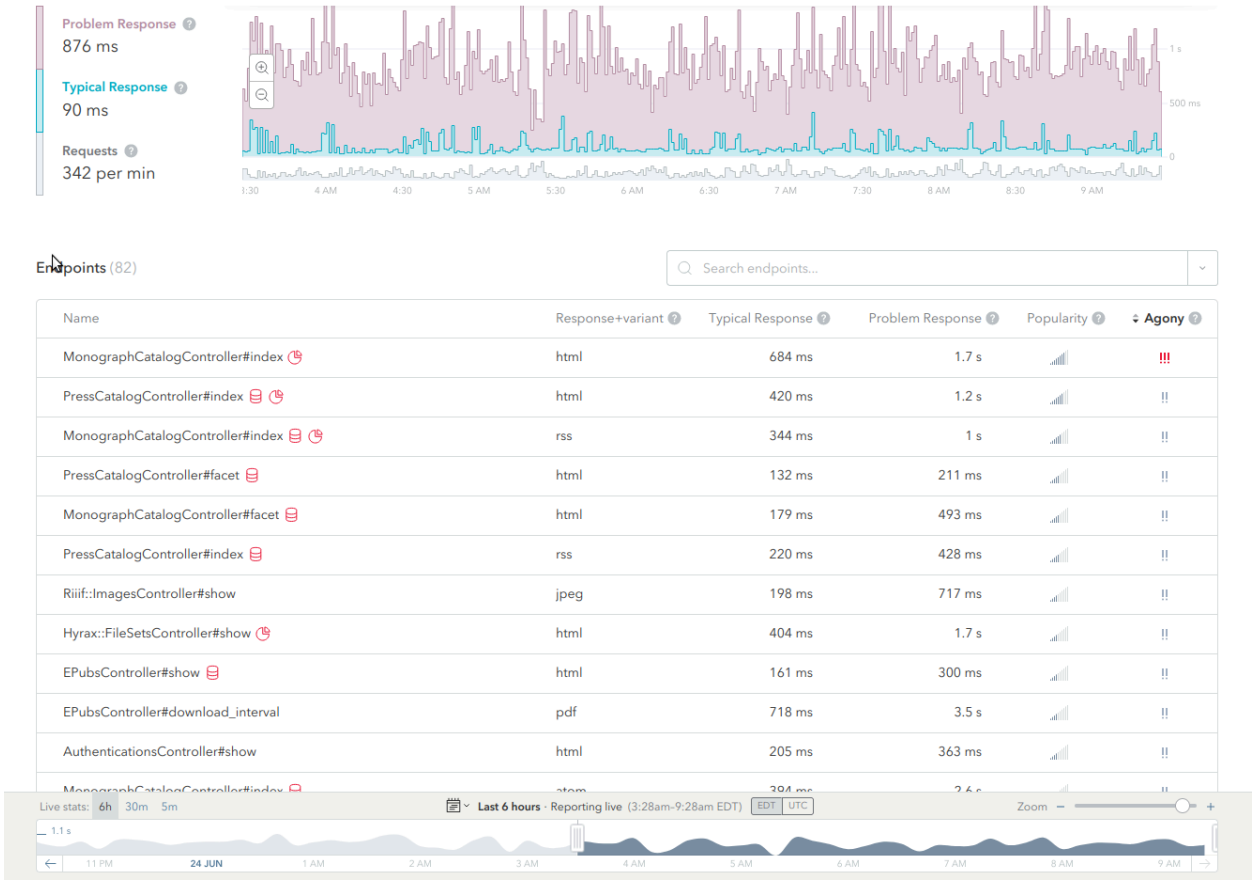
```
blacklight6:        29ms,  19,000+ allocations

blacklight7:       841ms, 548,000+ allocations
```

We didn’t pay close enough attention and put Fulcrum 4.0 into production like this which caused a lot of headaches. In the end, by updating our blacklight overrides to use the conventions in Blacklight 7 like the presenters and view components, and to no longer produce deprecation warnings, even when they were silenced, we were able to once again get down to the 20,000 allocations and 30 ms timings for the catalog templates. It was actually a good thing for our team in some ways, we’d been putting off a lot of performance tuning in our code and letting things slide a bit and the necessity of fixing this in our blacklight code led us to fix even more performance bottlenecks in other places. We’re still not done, but we’re in a better place now. In the next few pages are a couple of images from Skylight that show the before and after of our performance tuning.



This was right after our Hyrax 4 upgrade and shows some pretty unacceptable response times that overloaded puma and caused site degradation.



Things are looking better now, although there's still room for improvement.