Second Time’s the Charm… (sort of)
Lessons learned from two attempts to migrate from Dspace 6.3 to Dspace 7.6

Erica Johns (Lead Librarian for the eCommons Institutional Repository Service)
Robert Wilson (Cloud systems administrator)
Brandon Kowalski (Senior Software Engineer)
Cornell University Library IT
Opportunity Strikes / we got very excited about doing good things in the world!

Security considerations and operational objectives yield a unique opportunity.

Our Land-Grant Legacy
As a land-grant university, we exist to bring positive change into the world.

University Cloudification Push
Cornell University has been pushing for the moving of applications hosted on-prem to the cloud.

Library IT Containerization
Cornell University Library IT (CUL-IT) has embraced serverless technologies, including containers, for new apps.

Dspace 6.x End of Life
Support ended July 2023 - need to migrate for continued software patching.
The Elevator Pitch
We want “Turn-Key Repositories” for the masses!

The Dream
Democratize the process of deploying / running a DSpace Repository

Containerize
Research if the software can be run inside of a container

Automate
If software run successfully, automate the configuration and deployment process

Celebrate!
1. Answer a few questions
2. Press Deploy Button
3. Receive bacon Repository
4. Fulfill UN Sustainability Goal #10
Old School

1. Deploy by Hand
   One must install all software by hand and then keep it up to date.

2. Server Patching
   Not only do you need to keep software up to date, also have to patch servers.

3. Personnel Requirements
   IT Team to maintain infrastructure and software.

Too Cool

1. Automated = Repeatable
   Software configures itself. Need to update? Change the version number and rebuild.

2. Serverless
   Software is all bundled up in the container image and infrastructure is handled for you.

3. Less is more
   Single person / smaller team can operate. Can embrace the AWS Shared Security Model.
Opportunity = Challenges

**CHALLENGE 1**
Migrations are hard in general without any changes in architecture.

**CHALLENGE 2**
Containerization introduces unique new considerations.

**CHALLENGE 3**
Our DSpace instance is sizable, has complicated permissions, and features local customizations.

**CHALLENGE 4**
Attempting to migrate, containerize and cloudify all at once is a little... Ambitious ;)

Container Woes
Had a few growing pains to navigate

1. No command-line available to perform batch uploads (or other administrative tasks).

2. Shibboleth single sign on support when running behind an SSL-Terminating Load Balancer

3. Solr hogs memory from other containers.

4. Intercommunication between containers across services.

5. Managing DSpace configuration files via environment variables.
What our containerized instance looked like!

Sorry! Tons of tech to fit. You’ll have to download these slides.
Ok, so you already know. It didn’t work out.
BUT! We very nearly accomplished it!

- We migrated all content (bitstreams), PostgreSQL database, and SOLR.
- Through our testing with concurrent users, we discover a significant performance delay, and a couple bugs. (e.g. taking 1 minute+ to load the submission form)
- So, we reverted to our DSpace 6.3 instance
  - A let down, but we learned a lot!
  - We continued to experiment with our infrastructure.
Questions Linger

Is the performance the fault of the containerization?

How have others migrated?
Identifying the problem

• Since we had the majority of the DSpace 7 deployment containerized as a monolith task, we were not entirely sure of the origin of the performance issues.

• The monolith only showed a composite graph for CPU and memory usage, useless for finding the problem child.

• We then separated each component into its own task (we’re still in container land)
  • Java API
  • Angular UI with PM2
  • Apache2 Reverse Proxy
  • Prefect Agent
  • SOLR (already lived alone on an EC2 backed container instance)
  • PostgreSQL excluded as it lives on AWS Relational DB Service
Identifying the problem continued

• Splitting the container monolith worked!

• The Angular UI was hanging when hit with traffic.
  • The slow response tied up worker threads in the Apache Reverse Proxy
  • This in turned caused health checks to fail at the load balancer
  • Failed Health Checks = HTTP 500 Level Errors = Sad Migration Team 😞

• The DSpace community has documented some performance issues with the UI, but we needed to dig in further
WE CAN FIX THAT, BOOM! FIXED!

Source: We Bare Bears on Cartoon Network
If only it were that easy!
The makings of a second migration attempt...
Main infrastructure tactics:

First, we de-containerized and installed dspace with the same monolith (everything running on single server instance) set up as our 6.3 instance albeit off-premises servers (AWS EC2). We didn’t have performance issues with this 6.3, maybe it’ll solve our issues! Spoiler alert: it didn’t. This is because Dspace 7 has a totally different infrastructure.

Since then, we’ve been slowly breaking up the monolith with SOLR first. Are you seeing a pattern yet?

We’re still experiencing performance issues, but its better. We migrated in September with the understanding that we would performance tune and decided not to let perfection get in the way of “it works.... Ok.”

Currently in progress is separating reverse proxy and experimenting with nginx instead of Apache. The reverse proxy is necessary for both tomcat and angular to be accessible on commons web traffic ports: 80/443. Nginx has a smaller footprint and is often utilized as a reverse proxy.

Next, we will be separating out API and UI from each other.
Current Infrastructure
In Progress
More performance tuning for September 2023 migration and it continues...

- Experimented with different SSR cache values for bot and anonymous
  [https://wiki.lyrasis.org/display/DSDOC7x/User+Interface+Configuration#UserInterfaceConfiguration-CacheSettings-ServerSideRendering(SSR)](https://wiki.lyrasis.org/display/DSDOC7x/User+Interface+Configuration#UserInterfaceConfiguration-CacheSettings-ServerSideRendering(SSR))
- Tuning of PM2 cluster mode with CPU and memory limits
  - Significantly increased cores in use: from 4 cpu cores with 2gb each to 24 cores with 4gb each
- Default http protocol in was http/1.1. Changed to http/2
  - http/2 allows multiplexing. Less bottleneck potential when loading resources compared to http/1.1 which would load one resource at a time.
- Once we made the domain cutover, RDS (AWS database service: PostgreSQL) instance got hit pretty hard. Had to up the instance size significantly to deal with increased load.
- EC2 instances suffered same fate with not enough memory or CPU. Changing instance sizes as needed was done.
  - Instance type in ec2 is a combination of CPU and memory. Just kept increasing as needed. Started with 4cpu /16gb and have increased as traffic has warranted. For tomcat and solr memory, increased those as instance sizes were increased. Increasing resources for services has always helped. Identifying ways to reduce resources needed will help with costs however.
- Significantly increased Tomcat memory
- Significantly increased Solr memory
- Modified robots.txt to reduce potential impact from bots per documentation: [https://wiki.lyrasis.org/display/DSDOC7x/Search+Engine+Optimization](https://wiki.lyrasis.org/display/DSDOC7x/Search+Engine+Optimization), since we have sitemaps in place, we were able to further reduce paths a good bot would follow thereby reducing load on box.

Before many of these changes, we'd regularly get F or D grades from site performance testing like GTMatrix: [https://gtmetrix.com](https://gtmetrix.com). But now regularly we are in B range. Audit recommendations are specific to UI structure. This might be from our customization or out of the box UI behavior.
WINNING! Or... at least better!
BETTER = Continued performance issues

- Sometimes thumbnails never load
- Sometimes pages never load OR we experience what we call “forever elipses”. It’s much better than before.
- Still, it’s enough that I have this guidance on our help pages:

---

**eCommons DS 7.6 Migration Status**

**Update**

We migrated to a new DSpace infrastructure on September 15, 2023. There are some known issues with the new system and we expect you may find others! If you are unable to find the guidance you need, please reach out to eCommons administrators with questions or issues you may be experiencing. Thank you in advance for your patience!

**Known Issues**

- The load times are slower in this version of eCommons than our previous instance due to the caching behavior in the angular infrastructure. Our first suggested fix is to refresh the page if it fails to load. If this doesn’t work, please clear your cache and try again.
- We are aware that sometimes, customized thumbnail images do not load.
Future: Due to the high volume of resources required to tune performance, we must consider cost savings and asset store durability.

Did you see the number of resources we are putting towards this? 24 cores with 4GB of memory each?! ($$$)
And we’re still experiencing performance issues.

Plan to move from EFS storage to S3

Then containerize each EC2 service, starting with SOLR
This does not address UX, a whole other can of worms

We have received a fair amount of feedback about the user experience of Dspace 7’s UI

I will be leading a Dspace Community Advisory Team (DCAT) usability project at the next Dspace testathon in 2024

We hope to focus on search, browse, filter and sort, general navigation, and statistics

We hope that the feedback will prove consistent, and those observations will be used by Dspace leadership and development for future UI improvements.
We still have dreams to democratize the hosting of repositories and create a “turn-key” solution but, the more pressing daydream is 3-5 second page loads 😊

Desired Outcome

Provide anyone the means of hosting a repository and sharing its data.

Step 1

Demonstrate acceptable performance with a DSpace 7 instance powered by containers.

Step 2

Remove CUL-IT Customization and create means to handle configuration simply.

Step 3

Distribute container images along with a method for populating a configuration file.
Do you have a decent size Dspace 7.x repository experiencing optimal performance?

Let’s chat! (please!)
Questions? Shoot an email!

Robert Wilson  
Cloud Systems Administrator  
rtwilson@cornell.edu

Brandon T. Kowalski  
Senior Software Engineer  
kowalski@cornell.edu

Erica M. Johns  
Repository Manager  
emj73@cornell.edu