DSpace Docker and Cloud Deployment Goals

OUTDATED / OBSOLETE: Please see DSpace and Docker for latest info.

Problem Summary

Configuring Test Environments

Configuring and managing DSpace test and development environments requires significant effort.

When developing and testing against multiple versions of DSpace, the management of system software becomes significantly more complex especially when there is a need to downgrade versions of software.

DSpace 4, 5, and 6 require Postgres and Tomcat as prerequisite software.

A full DSpace 7 instance will require Postgres, Tomcat, Node (angular), and Solr as pre-requisite software. A developer may wish to focus on either the full set of components or a subset of these components.

Assumption

The complexity of managing these development and test environments has likely prevent some institutions from contributing to the platform.

The project might be able to engage a larger audience of developers if the prerequisites for creating a development environment could be simplified.

Many DSpace stakeholders are not developers. If it was easier to deploy a snapshot instance of DSpace to the cloud, these stakeholders could play a more active role in system testing.

Deployment Options

The following grid documents some of the potential ways DSpace Docker images could be deployed. Currently, only the first option is supported by the published DSpace images.

This grid has been designed to help recommend how our images will need to evolve to support multiple deployment options.

Deployment Resources

- https://kubernetes.io/docs/setup/

Potential DSpace code changes required for implementation

1. INGEST: Move docker ingest tools from docker-compose mount directories into the docker image file system.
2. VARIABLES: Simplify variable overrides by not using dashes and docs in variable names.
   a. Could a config map be used? https://kubernetes.io/docs/tasks/configure-pod-container/configure-pod-configmap/
3. CFG: Create a mechanism to pass in a config file when starting an image
4. CLI: Migrate all CLI functionality to REST API calls. Create a queue system to allow some tasks to run in an isolated memory space.
5. VOLS: Implement volumes as cloud storage. Cloud storage could be accessed from multiple nodes. Ensure that all CLI tasks (esp filter-media tasks) do not rely on a file system.
6. REPL: Allow the database to be replicated to multiple nodes.
7. EMAIL: Integrate email service into image.
8. AUTH: Integrate authentication providers into image.

Deployment Options	**Uptime**	**DSpace Nodes: Tomcat, SOLR, Angular**	**DB Nodes**	**Persistence**	**CLI Access and cron tasks**	**Notes**	**Changes needed to DSpace code**
Local workstation/test server deployment with Docker Compose	Limited Duration for Testing	1	1	Docker Compose Volumes	docker exec	https://github.com/DSpace-Labs/DSpace-Docker-Images	INGEST
Server deployment with Docker Compose (shell access to server and network drives)	Extended Availability	1	1	Docker Compose Volumes bound to file system storage	SSH + docker exec		INGEST, CFG
Docker swarm options	Extended Availability	?	?	?	?	https://docs.docker.com/v17.09/get-started/part6/	
Local workstation/test server deployment with Kubernetes	Limited Duration for Testing	1	1	Use "local" volume. https://kubernetes.io/docs/concepts/storage/volumes/#types-of-volumes	?	Some experimentation here: https://github.com/DSpace-Labs/DSpace-Docker-Images/pull/140	
Server deployment with Kubernetes (shell access to server and network drives)	Extended Availability	1	1	?	?		
Server deployment with Portainer management	Extended Availability	?	?	?	?	https://www.portainer.io	
Public cloud deployment using vendor-specific deployment option (such as Amazon ECS or Fargate)	Extended Availability	1	1	Assetstore in cloud storage	No access		CLI, VOLS
Public cloud deployment using Kubernetes	Extended Availability	1	1	Assetstore in cloud storage	?		CLI, VOLS
Public cloud deployment using vendor-specific deployment option (such as Amazon ECS or Fargate)	Extended Availability	N	1 or N	Assetstore in cloud storage	No access		CLI, VOLS, REPL
Public cloud deployment using Kubernetes	Extended Availability	N	1 or N	1 or N	?		CLI, VOLS, REPL

Deploying to Production / to the Cloud with Docker

There have been multiple questions in the community regarding production deployment with Docker.

The public cloud providers offer some compelling ways to deploy docker containers. How suitable is DSpace for such a deployment?

Current Goals

Goal 3: Create a simple workflow to deploy a DSpace branch or snapshot to the cloud

- Status: Some prototyping

Each of the major cloud providers provides a mechanism for running Docker containers.

If we could streamline the process for deploying published Docker images to the cloud, we could support a unique test environment for each major branch of DSpace. This would allow users to compare and contrast functionality on each version of DSpace. It would also allow an institution to evaluate the functionality on a specific DSpace branch.

Here is a write up of my experimentation with Docker Images running on specific cloud providers.

A prototype implementation is here: http://bit.ly/dspace-launcher-dashboard

Goal 4: Manage hosted instances of DSpace for each supported branch of the system

- Status: Some prototyping, inactive

If the prior goals are achieved, it would be possible to manage multiple hosted instances of DSpace for each major branch of the system in addition to the production reference version at demo.dspace.org.

A prototype implementation is here: http://bit.ly/dspace-launcher-dashboard

Goal 6: Explore Kubernetes Deployment of DSpace

- Status: Some prototyping
- https://github.com/DSpace-Labs/DSpace-Docker-Images/pull/140

Docker Compose is an excellent solution for deploying DSpace Docker for testing.

As institutions explore the possibility of a production deployment of DSpace Docker or a public cloud deployment of DSpace Docker, the dependency on Docker Compose may not be suitable. The public cloud providers offer mechanisms for deploying containers without fully provisioning a server.

- AWS offers Elastic Container Service and Elastic Kubernetes Service
- Google Cloud offers Google Kubernetes Engine
- Azure offers (TBD)

It is also possible to run a local kubernetes instance for testing.

The tools that we use to ingest content on startup are currently supplied through docker compose volumes. As these tools are stabilized, they should be migrated into the DSpace docker images.

- Ingest Tools: https://github.com/DSpace-Labs/DSpace-Docker-Images/blob/master/add-ons/dspace-docker-tools/ingestAIP.sh
- Target location: https://github.com/DSpace/DSpace/tree/master/dspace/src/main/docker

Goal 7: Document Recommendations for Deploying DSpace Docker in Production

- Status: Not started

The following changes will be needed to run DSpace Docker in Production.

- Eliminate the dependency of docker compose volumes (see above)
- In a cloud context, terminal access may or may not be available.
    - Migrate CLI actions to REST endpoints
- In a cloud context, the local file system may not be available
    - Migrate assetstore to cloud storage objects
- Which DSpace services can be replicated to multiple nodes?
- Application life-cycle questions: how to update the docker container (f.e. a security update of a library) while preserving content
- **Log additional items here**

Inactive

Goal 2: Publish a standard set of AIP files (archival ingest packages) to facilitate system testing

- Status: Some prototyping, inactive

In order to expedite and simplify DSpace testing, it would be useful to provide developers with content suitable for testing a majority of DSpace use cases.

Fortunately, the DSpace AIP ingest process provides a mechanism for constructing a DSpace repository that is DSpace version agnostic.

What test resources exist?

We have generated one simple set of re-usable AIP files posted on GitHub. https://github.com/DSpace-Labs/AIP-Files

These files are too simplistic for testing real DSpace use cases.

Demo.dspace.org is populated with a set of AIP files that are stored on Amazon S3. These files are approximately 1.5G in size.

A regular GitHub repository is not appropriate for sharing large files. See https://help.github.com/articles/working-with-large-files/.

What test resources should exist?

- A community/collection hierarchy containing items of various sample document types (PDF, image) that illustrate DSpace features including some access-restricted material.
    - These examples should be created from contrived examples or from open-access resources.
- A community/collection hierarchy containing multilingual metadata
- A community/collection hierarchy containing some digital collection material

How should these test resources be distributed?

The collections of material that are likely to be shared will exceed the space and bandwidth limitations of a GitHub code repository, so there are likely to be some storage and bandwidth costs associated with sharing these resources.

A test collection would need to be downloadable as a single zip file or as a collection of individual zip files.

- Explore GitHub LFS (large file storage) as an option
- Share resources through a cloud service such as AWS S3.
- Share resources using a collaboration platform such as Box or Google Drive
- Upload these asses to a data repository and share them from that platform

Goal 5: Publish instructions for a replicable development environment for on-boarding, tutorials, and troubleshooting

- Status: Some prototyping, inactive

Frequent contributors to DSpace are likely to manage and maintain a robust development environment that includes best of class development tools such as IDE's.

There is also a need to assist new contributors and potential contributors with the creation of a simple and basic development environment for DSpace.

While Docker can standardize the runtime and test environments, it may be possible to document a replicable development environment to help with on-boarding.

See Docker Workshop.

Completed Goals

Goal 1: Simplify and standardize the process for creating a development environment for all supported versions of DSpace

- Status: Complete: ✓

Docker provides a consistent, predictable environment

Docker provides users with a consistent and predictable runtime environment. With such an environment, new DSpace developers will be able to more easily isolate local environment inconsistencies when seeking help from the DSpace community.

Docker allows a user to manage multiple (and incompatible) environments

Docker containers allow a developer to manage and run multiple system configurations from a single desktop or test server.

Significant work has already been done in this area. https://github.com/DSpace-Labs/DSpace-Docker-Images

Each major branch of DSpace and each recent release of DSpace has been published as a Docker image.