Schedule 1: Installation Quote for Islandora / Fedora

David Livingstone Archive

Version History

Version	Description	Author	Date
1.0	Initial Draft	Nigel Banks	November 11th, 2015
1.1	Revised Quote	Nigel Banks	November 24th, 2015
1.2	Added deliverables / corrected mistakes	Nigel Banks	Dec 1st, 2015

Overview

The following quote will be to set up a Islandora installation based on the latest stable release, along with a number of additional modules / themes that were developed by UCLA Library across a number of servers to be provided by the University of Maryland. The installation will cover the setup and configuration of three environments, *production, stage, development*

Production

The production environment will require the most resources as it will be responsible for: storing all the data; generating derivatives; and serving the content to the public. The public will be able to access the site at livingstoneonline.org.

Recommended Hardware Requirements & Partition Layout

Purpose	OS	# CPU(s)	RAM	Hard Disk
Host public site & data	Red Hat Linux	8 CPU(s)	12 GB	1 TB

Partition Name / Purpose	Mount Point	Size
Root / Filesystem	/	32 GB
Swap		4 GB
Fedora Repository	/data	988 GB

Stage

The stage environment is to serve as a test bed for changes that have been made in the development environment and are considered ready for deployment, but are not yet approved for deployment to production. The site will be available url to be determined, this site is not meant for public consumption.

Recommended Hardware Requirements & Partition Layout

Purpose	OS	# CPU(s)	RAM	Hard Disk
Test new features / work	Red Hat Linux	2 CPU(s)	4 GB	32 GB

Partition Name / Purpose	Mount Point	Size
Root / Filesystem	/	28 GB
Swap		4 GB

Development

The development environment will not be hosted, but will be deployable locally (on one's laptop for example) via Vagrant. It's meant to serve as a place to testing features that are not ready for deployment.

Requirements

Although I would prefer to have full access to the existing server which is currently hosted at http://livingstoneonline.org, this shouldn't be a requirement as Kathy will be able to export the site configuration and Drupal nodes via Drupal Features.

The University of Maryland will be expected to deliver the stage and production VMs as described in the previous section.

To ensure the work can be done correctly and in a timely fashion **SSH Access & Root access** to all servers **must be provided** to me. Also **port 8080** on the **Production** server should be accessible to all the other servers mentioned above, but otherwise **not available** to the **public** or other servers. The **Production** should be setup with Logical Volume Management (**LVM**), such that the repository can be grown when needed. Also a **separate volume** should be set aside for the repository data mounted at root (for example **/data**), this volume will *eventually* have 2.0 TB of space allocated to it, 32 GB should be set aside for the root partition and 4GB for swap.

Provisioning & Deployment

We'll combine a number of tools to install, configure and update boxes. Ansible will be used for provisioning Docker Images (and the servers), which can then be deployed to all environments as well as configuring and orchestrating the servers. This will allow us to essentially configure once, and deploy identical stack to all the developers involved and to the servers provided by University of Maryland. In addition we'll be able to manage rolling updates and deployments, via Ansible to ensure smooth updates, as features progress from development to stage and finally to production.
The approach is outlined roughly here:

1. Write Ansible playbooks for creating Docker Images
   (Layered for each component of the Application: Drupal, Fedora, Islandora, etc).
2. Write Ansible playbooks that customize each environment
   (Different code on the development environment vs. production)
3. Generate Docker images via playbooks in step 1.
4. Push Docker images up into registry.
5. Write Ansible playbooks to pull Docker images down to remote hosts and start up Docker containers on remote hosts, passing in configuration information.
6. Run Ansible playbooks to start up the containers.

This gets us to the point where we can have a fully operational development, stage, and production environment, but this does not yet cover automated deployment to any of the environments. For that we'll have to build another system, which is described in Automated Deployment.

We'll have a number of layered images and a smaller number of containers that will run those images. The benefit of layered images decreases build and deployment times, and containers are separated on an application basis. Containers & the images they are running:

- MySQL Container
  - MySQL Image
- Fedora Container
  - Tomcat Image
  - Fedora Image (Build on-top of Tomcat)
- Drupal Container
  - Drupal Image
  - Islandora Dependencies Image (Build on-top of Drupal)
  - Islandora Image (Build on-top of Islandora Dependencies)
  - David Livingstone Image (Build on-top of Islandora Image)
    - Three flavors of this image will be build on for each environment

Automated Deployment

Once we have a framework in place for provisioning Docker Images, which we can then deploy those images manually to development, stage, and production environment. If we need to change any aspect of our system, we would change an Ansible playbook manually and rebuild the Docker images and deploy them to one or more of the environments.

Ideally we would like to automate this process though, such that it doesn't require in depth knowledge of Ansible and Docker to update each environment. To achieve this we would require a Continuous Integration server. Either we can run our own Jenkins server out of AWS (low end cheap server, around $6-10 dollars a month) or perhaps we can get away with a free version of Travis CI which would not require us to host our own server (it might be too limited, although initial research suggests it will do just fine).

Development Environment

Since development is in constant flux it would be ideal if it would alway be running at HEAD (aka the very latest code). To achieve this we would have our CI server monitor any Github repositories of interest, and whenever a change is made to one of them, it would kick off a process to build new images for the Development environment. This doesn't alleviate us of all manual changes though, if a new tool / module is needed in the development environment someone would have to update the Ansible playbook, but the generation of new Docker images and their deployment would be automated.

Staging & Production Environments

We want to be a bit more selective in what we deploy to stage and production, but at the same time we want to make it as easy and automated as is possible, so in this case we could do the same as with development except monitor a separate branch for features approved for stage / production.

Installation Components

Fedora

This will include the latest stable release of **Fedora 3.8.1** (released on June 11th, 2015), along with the latest stable **GSearch 2.8** and **Apache Solr 4.6.1** (not the latest but the latest supported by GSearch), **Djatoka / Kakadu 1.1** and **Java 7** (Djatoka requires Sun/Oracle Java), the Basic Solr-Config (HEAD) and **Drupal Filter 7.x-1.6**.

Islandora

UCLA used several github repositories as a form of automated deployment. Such that collaborators could commit new modules and code to a git repository and have it deployed automatically. We will instead monitor changes to github repositories of interest and auto-deploy when changes are made on their development, stage, and production branches respectively.

Core Modules

This will include relevant parts of the latest stable release of Islandora 7.x-1.6 (released on November 3rd, 2015), which includes the following Solution Packages & Modules:

- Islandora Core
- Tuque
- Solution Pack Basic Collection
- Solution Pack Basic Image
- Solution Pack Large Image
- Solution Pack Audio
- Solution Pack Video
- Solution Pack Book
- Solution Pack Newspaper
- Islandora Paged Content
- Islandora FITS
- Islandora Internet Archive Book Viewer
- Islandora OpenSeadragon
- Islandora JWPlayer
- Islandora Checksum
- Islandora Checksum Checker
- Islandora OCR
- Islandora XACML Editor
- Islandora Usage Stats
- Islandora Batch
- Islandora XML Forms
- Objective Forms
- PHP Lib
- Islandora Solr Metadata
- Islandora Solr Search

UCLA / Other Custom Modules

In addition modules that were developed by 3rd parties such as UCLA in which can be useful will be installed as well:

- Solution Pack Manuscript (Manuscript Content Model)
- Islandora Paged TEI Seadragon (Viewer for TEI and Manuscripts)
- Custom Nexus Theme (Developed by Kathy Chavez)

Contrib Modules

The following Drupal Contrib Modules seem to be in use, but may get dropped at a later time if deemed unnecessary (Discussion with Kathy is needed):

- addthis
- admin_views
- devel (will not be installed on *Production*, just *Development*)
- devel_themer (will not be installed on *Production*, just *Development*)
- dragndrop_upload
- filefield_sources
- filefield_sources_plupload
- fpa
- internal_nodes
- jcarousel
- media
- menu_attributes
- metatag
- multiupload_filefield_widget
- nodequeue_pager
- private_files_download_permission
- simplehtmldom
- Although it may be simpler to just include all the modules from here, rather than work out the dependencies now, though not ideal.

Libraries

The following Drupal Libraries seem to be in use, but may get dropped at a later time if deemed unnecessary (Discussion with Kathy is needed):

- ckeditor
- html_encoder
- jquery.blockui
- Others may be needed (access to current http://livingstoneonline.org required to verify)

Features

The following *Drupal Features* seem to be in use, but may get dropped at later time if deemed unnecessary (Discussion with Kathy is needed):

- Basic page content type
- Level 2 images view
- level 2 and 3 stage
- Pages Panels
- Repository content type & view
- TempModsDisplay

Additional Dependencies

As well as the following additional software:

- Drupal 7.41
- PHP >= 5.3.3 <= 5.5
- Apache >= 2
- Fits
- Image Magick
- FFmpeg
- Tesseract
- Lame
- poppler-utils

Quality Assurance Testing

Each of the installed components will be tested manually to confirm they are working.

Users (with appropriate permissions will be able to):

- Upload / create (Audio, Video, Images, Books, Newspapers, Manuscripts, Collections)
- View (Audio, Video, Images, Books, Newspapers, Manuscripts, Collections)
- Modify existing metadata
- Search for objects, based on MODS metadata

In addition all the automated tests provided by islandora (known as simple tests) will be run to confirm the installation is working as expected. The purpose here is to verify that the installation for all environments is fully functional.

Documentation

Since the Ansible playbooks will describe exactly how the systems will be configured (what commands are issued and what files are modified to install the software), the playbooks act as documentation as well as a functional system for creating each environment. It would be redundant to include documentation on how to install the software manually, by entering commands at the command line, and editing files to configure the server.

Instead of providing documentation on how to configure the systems directly, documentation will be provided at a higher level. It will describe how to Ansible and Docker are used to configure the servers (as well as how to deploy servers locally or remotely to develop / test features). As well as how the Continuous Integration, Automated Deployment systems are setup and how they can be modified. It will serve as a reference should changes need to be made to the Ansible playbooks or the Continuous Integration, Automated Deployment systems, by System Administrators at University of Maryland or others.

This documentation will be provided in Github wiki linked to the deployment code so it can easily be updated and kept in sync as time goes on.

Documentation for Islandora and the software include in the install will not be provided. It is publicly available at the Islandora Wiki.

Services / Features Not Included

What's **not** included:

- Installation of additional modules / software not noted above (can be added at a later time)
- Ongoing support of the installation
- Monitoring services (such as monit, nagios)
- Enhancements such as caching services, or other performance related add ons
- Backup system for the repository / MySQL databases

Only the initial installation is covered by this quote; any issues with any of the installed software (such as known bugs or newly discovered bugs) that is not related to it's proper installation is not covered by this quote.

Deliverables

Here is short summary of deliverables for reference.

1. Reproducible system for setting up software in each environment (Dev, Stage, Prod)
    a. Ansible playbooks for provisioning Docker Images
    b. Ansible playbooks for deploying Docker Containers
    c. Ansible playbooks for orchestrating Docker Containers
2. Automated Deployment System
    a. Configure CI system to update playbooks when changes occur
    b. Configure CI system to automatically build Docker Images when playbook changes
    c. Configure CI system to deploy Docker Containers when Docker Images change
3. Working Islandora installation on Stage / Production servers Provided by UMD.
4. Documentation
    a. Documentation to setup a local environment (Dev, Stage, Prod) using Docker
    b. Documentation on how to use playbooks to provision / deploy / orchestrate containers
    c. Documentation on how to update playbooks to include new Drupal modules, Libraries, and Drupal Features.
    d. Documentation on how to update running containers
    e. Documentation on how to configure the CI system
        i. How it's structured
        ii. How to setup monitoring of additional git repositories

All code and documentation deliverables will be distributed in the form of Github repositories, in which the client will have full access.