Introduction

In Islandora's Github documentation, the Manual Installation section includes the following disclaimer: "[this] documentation is in need of attention. We are aware that some components no longer work as documented here." This document is my attempt at providing some of this much-needed attention. After quite a bit of trial-and-error, I've been able to get a manual installation of Islandora (for Drupal 10) successfully working. To do so, I had to diverge quite a bit from what's currently posted online.

Below is a copy of Islandora's manual instructions (as of **June 21, 2024**). My recommended changes appear throughout this document in orange boxes like this one. Hopefully, after vetting my suggestions, they can be incorporated into the online documentation, making the manual process that much easier / less painful for others to do in the future.

Below are the main software differences between my environment and what's defined in the documentation:

- **Ubuntu 20.04** (vs. Debian in the documentation)
- **MySQL** (vs. PostgreSQL in the documentation)

If you have questions or need anything clarified, feel free to contact me.
(Stefan Langer: https://github.com/slangerx ; Issue # 2326)

Preparing a LAPP Server¶

In this section, we will install:¶

- Java
- cURL
- Apache 2, the webserver that will deliver webpages to end users
- PHP 8, the runtime code interpreter that Drupal will use to generate webpages and other services via apache, as well as that Drush and Composer will use to run tasks from the command line
- Several modules for PHP 8 which are required to run the PHP code that Drupal and other applications will be executing
- MySQL, the database that Drupal will use for storage (as well as other applications down the line)

Account tracker

It might be nice to include a consolidated list of all the user accounts that get created throughout the installation process, so they're easier to keep track of and maintain in the future.

MySQL

MySQL *root* account

username = `root`
password = `[mysql_root_password]`

MySQL account for Drupal database access

username = `[mysql_drupal]`
password = `[mysql_drupal_password]`

MySQL account for Fedora access

username = `[mysql_fedora]`
password = `[mysql_fedora_password]`

Tomcat

username = `tomcat`
password = `[tomcat_user_password]`

Fedora

fedoraAdmin account

username = `fedoraAdmin`
password = `[fedora_admin_password]`

fedoraUser account

username = `fedoraUser`
password = `[fedora_user_password]`

ActiveMQ

username = `[activemq_user]`
password = `[activemq_user_password]`

Ports

I initially had some port conflicts on my server that caused a bunch of problems. I wonder if it would a good idea to have users check whether the following necessary ports are currently available?

```
sudo lsof -i:61613
sudo lsof -i:61616
sudo lsof -i:61623
sudo lsof -i:61626
```

Java

Since so many applications depend on Java, maybe it makes sense to install it much earlier in this process?

Install latest version of Java:

```
sudo apt-get update
sudo apt install default-jdk
java -version     # (11.0.22)
```

OR

```
sudo apt-get -y install openjdk-11-jdk openjdk-11-jre
```

cURL

Also, there are no instructions for installing cURL (required to install Composer and Blazegraph later on).

Resource: https://www.cyberciti.biz/faq/how-to-install-curl-command-on-a-ubuntu-linux/

```
sudo apt update
sudo apt upgrade
sudo apt install curl
curl --version
```

Apache 2¶

Install Apache 2¶

Apache can typically be installed and configured outright by your operating system's package manager:

```
sudo apt-get -y install apache2 apache2-utils
```

This will install:

- A `systemd` service that will ensure Apache can be stopped and started, and will run when the machine is powered on
- A set of Apache configurations in `/etc/apache2`, including the basic configuration, ports configuration, enabled mods, and enabled sites
- An Apache webroot in `/var/www/html`, configured to be the provided server on port `:80` in `/etc/apache2/sites-enabled/000-default.conf`; we'll make changes and additions to this file later
- A user and group, `www-data`, which we will use to read/write web documents.

Enable Apache Mods¶

We're going to enable a couple of Apache mods that Drupal highly recommends installing, and which are de-facto considered required by Islandora:

```
sudo a2enmod ssl
sudo a2enmod rewrite
sudo systemctl restart apache2
```

Add the Current User to the `www-data` Group¶

Since the user we are currently logged in as is going to work quite a bit inside the Drupal directory, we want to give it group permissions to anything the `www-data` group has access to. When we run `composer`, `www-data` will also be caching data in our own home directory, so we want this group modification to go in both directions.

N.B. This code block uses **backticks**, not single quotes; this is an important distinction as backticks have special meaning in `bash`.

Note If doing this in the terminal, replace "whoami" with your username and remove the backticks

```
sudo usermod -a -G www-data `whoami`
sudo usermod -a -G `whoami` www-data
# Immediately log back in to apply the new group.
sudo su `whoami`
```

PHP 8.3

Install PHP 8.3 *(updated from: PHP 7.4)*

Maybe because I'm using Ubuntu instead of Debian, the PHP installation instructions currently online didn't work for me. The following steps did.
Resource: https://www.cherryservers.com/blog/how-to-install-php-ubuntu

```
sudo apt update

sudo apt install php

php --version

sudo add-apt-repository ppa:ondrej/php
# Press enter when prompted.

sudo apt update

sudo apt install php8.3 -y
```

Additional PHP modules / extensions

```
php -m  # see a current list of installed PHP modules

sudo apt install php8.3-
{cli,common,curl,gd,imap,intl,mysql,opcache,redis,xdebug,
xml,yaml,zip}

sudo apt install libapache2-mod-php unzip
```

Update PHP settings so the Drupal site can upload large files:

```
sudo nano /etc/php/8.3/apache2/php.ini

     upload_max_filesize = [recommended_filesize]

     post_max_size = [recommended_max_size]

     max_input_time = [recommended_input_time]
```

Restart Apache so these changes are active:

```
sudo systemctl restart apache2
```

Installation directories created:

```
/etc/php/8.3

/usr/bin/php8.3
```

MySQL

Install MySQL *(instead of PostgreSQL 11)*

I'm using MySQL as my database server rather than PostgreSQL, which will be reflected throughout this document.
Resource: https://ubuntu.com/server/docs/install-and-configure-a-mysql-server

Install:

```
sudo apt install mysql-server

sudo service mysql status  # press "q" to exit

sudo ss -tap | grep mysql

sudo service mysql restart

sudo journalctl -u mysql   #troubleshooting
```

Update (missing) root password:

```
sudo mysql -u root
#login for first time (password should be blank)

ALTER USER 'root'@'localhost' IDENTIFIED WITH
mysql_native_password BY '[new-password]';
```

Installing Composer, Drush, and Drupal¶

In this section, we will install:¶

- Composer at its current latest version, the package manager that will allow us to install PHP applications
- Either the Islandora Starter Site, or the Drupal recommended-project, which will install, among other things:
  - Drush 10 at its latest version, the command-line PHP application for running tasks in Drupal
  - Drupal 10 at its latest version, the content management system Islandora uses for content modelling and front-end display

Install Composer¶

Download and install Composer 2.x¶

Composer provides PHP code that we can use to install it. After downloading and running the installer, we're going to move the generated executable to a place in `$PATH`, removing its extension:

```
curl "https://getcomposer.org/installer" > composer-install.php
chmod +x composer-install.php
php composer-install.php
sudo mv composer.phar /usr/local/bin/composer
```

Download and Scaffold Drupal¶

At this point, you have the option of using the Islandora Starter Site, with its pre-selected modules and configurations which function "out of the box," or build a clean stock Drupal via the Drupal Recommended Project and install Islandora modules as you desire.

Option 1: Create a project using the Islandora Starter Site¶

Navigate to the folder where you want to put your Islandora project (in our case `/var/www/html`), and create the Islandora Starter Site:

This is where I set up my dev Drupal installation, which can obviously change. **Note** that in the online instructions, the path specified here and the one specified in the Solr section should match; but don't.

Give the user the option to name their Drupal site directory whatever they want.

```
cd /var/www/html

sudo mkdir [DRUPAL_DIRECTORY_NAME]

cd [DRUPAL_DIRECTORY_NAME]

sudo composer create-project islandora/islandora-starter-
site .
#keep period (.) at the end to install in current directory
```

This will install all PHP dependencies, including Drush, and scaffold the site.

Drush is not accessible via `$PATH`, but is available using the command `composer exec -- drush`

~~Option 2: Create a basic Drupal Recommended Project¶~~

~~Navigate to the folder where you want to put your Drupal project (in our case /var/www), and create the Drupal Recommended Project:~~

```
~~cd /var/www~~
~~composer create-project drupal/recommended-project my-project~~
```

NOTE: I didn’t try out or test Option 2.

Make the new webroot accessible in Apache¶

Before we can proceed with the actual site installation, we’re going to need to make our new Drupal installation the default web-accessible location Apache serves up. This will include an appropriate `ports.conf` file, and replacing the default enabled site.

Notice

Out of the box, these files will contain support for SSL, which we will not be setting up in this guide (and therefore removing with these overwritten configurations), but which are **absolutely indispensable** to a production site. This guide does not recommend any particular SSL certificate authority or installation method, but you may find DigitalOcean's tutorial helpful.

/etc/apache2/ports.conf | root:root/644

```
Listen 80
```

Remove everything but the "Listen 80" line. You can leave the comments in if you want.

/etc/apache2/sites-enabled/000-default.conf | root:root/777

Create a unique site virtual host (rather than updating the default: “000-default.conf”):

```
sudo nano /etc/apache2/sites-
available/[DRUPAL_SITE_IP_ADDRESS].conf
```

```
<VirtualHost *:80>
  ServerName [DRUPAL_SITE_IP_ADDRESS]
  DocumentRoot "/var/www/[DRUPAL_DIRECTORY_NAME]/web"
  <Directory "/var/www/[DRUPAL_DIRECTORY_NAME]/web">
    Options Indexes FollowSymLinks MultiViews
    AllowOverride all
    Require all granted
  </Directory>
  # Ensure some logging is in place.
  ErrorLog "/var/log/apache2/[DRUPAL_SITE_IP_ADDRESS]_error.log"
  CustomLog "/var/log/apache2/[DRUPAL_SITE_IP_ADDRESS]_access.log"
combined
</VirtualHost>
```

Set permissions and enable the virtual host
(*NOTE:* These steps are **missing** from the current documentation.)

```
sudo chown -R root:root /etc/apache2/sites-
available/[DRUPAL_SITE_IP_ADDRESS].conf
```

```
sudo chmod -R ??? /etc/apache2/sites-
available/[DRUPAL_SITE_IP_ADDRESS].conf
# what permission should be applied to this file?
```

```
sudo systemctl restart apache2
```

```
sudo a2ensite [DRUPAL_SITE_IP_ADDRESS].conf
```

```
sudo systemctl reload apache2
```

Restart the Apache 2 service to apply these changes:
sudo systemctl restart apache2

Prepare the MySQL database¶

We're going to create a user in MySQL for this Drupal site. Then create a database that we can use to install Drupal.

Give the user the option to name their Drupal database whatever they want.

```
mysql -u root -p

CREATE DATABASE [DRUPAL_DATABASE_NAME];

CREATE USER '[MYSQL_USER_FOR_DRUPAL]'@'localhost' IDENTIFIED
BY '[MYSQL_PASSWORD_FOR_DRUPAL]';

GRANT ALL PRIVILEGES ON [DRUPAL_DATABASE_NAME].* TO
'[MYSQL_USER_FOR_DRUPAL]'@'localhost';

exit
```

Install Drupal using Drush¶

The Drupal installation process can be done through the GUI in a series of form steps, or can be done quickly using Drush's `site-install` command. It can be invoked with the full list of parameters (such as `--db-url` and `--site-name`), but if parameters are missing, they will be asked of you interactively.

Option 1: Site install the Starter Site with existing configs¶

Follow the instructions in the README of the Islandora Starter Site. The steps are not reproduced here to remove redundancy. When this installation is done, you'll have a starter site ready-to-go. Once you set up the external services in the next sections, you'll need to configure Drupal to know where they are.

Create a basic Drupal recommended project
Resource: https://github.com/Islandora-Devops/islandora-starter-site

```
sudo nano
/var/www/html/[DRUPAL_DIRECTORY_NAME]/web/sites/default/sett
ings.php
```

Add the following to the very bottom of this file:

```
$settings['flysystem'] = [
  'fedora' => [
    'driver' => 'fedora',
    'config' => [
      'root' => 'http://127.0.0.1:8080/fcrepo/rest/',
    ],
  ],
];
```

I believe I initially ran into some permission problems when I first tried to install the site. Are there any permissions you'd recommend applying beforehand?

```
sudo chown ??? -R /var/www/html/[DRUPAL_DIRECTORY_NAME]

sudo chmod ??? -R /var/www/html/[DRUPAL_DIRECTORY_NAME]
```

Install the site:
Resource: https://github.com/Islandora-Devops/islandora-starter-site

```
cd /var/www/html/[DRUPAL_DIRECTORY_NAME]

sudo composer exec -- drush site:install --existing-config
```

Response:

```
[success] Installation complete.  User name: admin  User
password: [make note of this password]
```

I believe I ran into some permission problems later on with these directories. Are there any permissions you'd recommend applying afterwards?

```
sudo chown www-data:www-data -R
/var/www/html/[DRUPAL_DIRECTORY_NAME]/web/sites/default/files

sudo chown www-data:www-data -R /tmp

sudo chown www-data:www-data -R
/var/www/html/[DRUPAL_DIRECTORY_NAME]/web/modules/custom

sudo composer exec -- drush user:role:add fedoraadmin admin
```

After ActiveMQ is installed and working, do the following:

```
composer exec -- drush migrate:import --userid=1 --
tag=islandora
```

~~Option 2: Site install the basic Drupal Recommended Project~~¶

```
~~cd /var/www/drupal~~
~~drush -y site-install standard --db-url="pgsql://DRUPAL_DB_USER:DRUPAL_DB_PASSWORD@127.0.0.1:5432/DRUPAL_DB" --site-name="SITE_NAME" --account-name=DRUPAL_LOGIN --account-pass=DRUPAL_PASS~~
```

~~This uses the same parameters from the above step, as well as:~~

- ~~SITE_NAME: Islandora 2.0~~
    - ~~This is arbitrary, and is simply used to title the site on the home page~~
- ~~DRUPAL_LOGIN: islandora~~
    - ~~The Drupal administrative username to use~~
- ~~DRUPAL_PASS: islandora~~
    - ~~The password to use for the Drupal administrative user~~

NOTE: I didn't try out or test Option 2.

Congratulations, you have a Drupal site! It currently isn't really configured to do anything, but we'll get those portions set up in the coming sections.

Installing Tomcat and Cantaloupe¶

In this section, we will install:¶

- Tomcat 9, the Java servlet container that will serve up some Java applications on various endpoints, including, importantly, Fedora
- Cantaloupe 5, the image tileserver - running in Tomcat - that will be used to serve up large images in a web-accessible fashion

Tomcat 9¶

Installing OpenJDK 11¶

Tomcat runs in a Java runtime environment, so we'll need one to continue. In our case, OpenJDK 11 is open-source, free to use, and can fairly simply be installed using `apt-get`:

```
sudo apt-get -y install openjdk-11-jdk openjdk-11-jre
```

The installation of OpenJDK via `apt-get` establishes it as the de-facto Java runtime environment to be used on the system, so no further configuration is required.

NOTE: Java was already installed at the very beginning of these revised instructions.

The resultant location of the java JRE binary (and therefore, the correct value of `JAVA_HOME` when it's referenced) will vary based on the specifics of the machine it's being installed on; that being said, you can find its exact location using `update-alternatives`:

```
update-alternatives --list java
```

Take a note of this path as we will need it later.

Creating a `tomcat` User¶

Apache Tomcat, and all its processes, will be owned and managed by a specific user for the purposes of keeping parts of the stack segregated and accountable.

```
sudo addgroup tomcat
sudo adduser tomcat --ingroup tomcat --home /opt/tomcat --shell /usr/bin
```

You will be prompted to create a password for the `tomcat` user; all the other information as part of the `adduser` command can be ignored.

Downloading and Placing Tomcat 9¶

Tomcat 9 itself can be installed in several different ways; while it's possible to install via apt-get, this doesn't give us a great deal of control over exactly how we're going to run and manage it; as a critical part of the stack, it is beneficial for our purposes to have a good frame of reference for the inner workings of Tomcat.

We're going to download the latest version of Tomcat to /opt and set it up so that it runs automatically. Bear in mind that with the following commands, this is going to be entirely relative to the current version of Tomcat 9, which we'll try to mitigate as we go.

```
cd /opt
sudo wget -O tomcat.tar.gz TOMCAT_TARBALL_LINK
sudo tar -zxvf tomcat.tar.gz
sudo mv /opt/TOMCAT_DIRECTORY/* /opt/tomcat
sudo chown -R tomcat:tomcat /opt/tomcat
```

- TOMCAT_TARBALL_LINK: No default can be provided here; you should navigate to the Tomcat 9 downloads page and grab the link to the latest .tar.gz file under the "Core" section of "Binary Distributions". It is highly recommended to grab the latest version of Tomcat 9, as it will come with associated security patches and fixes.
- TOMCAT_DIRECTORY: This will also depend entirely on the exact version of tomcat downloaded - for example, apache-tomcat-9.0.50. Again, ls /opt can be used to find this.

Creating a setenv.sh Script¶

When Tomcat runs, some configuration needs to be pre-established as a series of environment variables that will be used by the script that runs it.

/opt/tomcat/bin/setenv.sh | tomcat:tomcat/755

```
export CATALINA_HOME="/opt/tomcat"
export JAVA_HOME="PATH_TO_JAVA_HOME"
export JAVA_OPTS="-Djava.awt.headless=true -server -Xmx1500m -Xms1000m"
```

- PATH_TO_JAVA_HOME: This will vary a bit depending on the environment, but will likely live in /usr/lib/jvm somewhere (e.g., /usr/lib/jvm/java-11-openjdk-amd64); again, in an Ubunutu environment you can check a part of this using update-alternatives --list java, which will give you the path to the JRE binary within the Java home. Note that update-alternatives --list java will give you the path to the binary, so for PATH_TO_JAVA_HOME delete the /bin/java at the end to get the Java home directory, so it should look something like this:

```
export JAVA_HOME="/usr/lib/jvm/java-11-openjdk-amd64"
```

Creating the Tomcat Service¶

Tomcat includes two shell scripts we're going to make use of - startup.sh and shutdown.sh - which are light wrappers on top of a third script, catalina.sh, which manages spinning up and shutting down the Tomcat server.

Debian and Ubuntu use `systemctl` to manage services; we’re going to create a .service file that can run these shell scripts.

/etc/systemd/system/tomcat.service | root:root/755

```
[Unit]
Description=Tomcat

[Service]
Type=forking
ExecStart=/opt/tomcat/bin/startup.sh
ExecStop=/opt/tomcat/bin/shutdown.sh
SyslogIdentifier=tomcat

[Install]
WantedBy=multi-user.target
```

Enabling and Starting Tomcat¶

We’re going to both `enable` and `start` Tomcat. Enabling Tomcat will ensure that it starts on boot, the timing of which is defined by the `[Install]` section’s `WantedBy` statement, which specifies what it should start after. This is separate from starting it, which we need to do now in order to get Tomcat up and running without requiring a reboot.

```
sudo systemctl enable tomcat
sudo systemctl start tomcat
```

We can check that Tomcat has started by running `sudo systemctl status tomcat | grep Active`; we should see that Tomcat is `active (running)`, which is the correct result of startup.sh finishing its run successfully.

Installing Cantaloupe 5¶

Since version 5, Cantaloupe is released as a standalone Java application and is no longer deployed in Tomcat via a .war file. Even so, we can still fine-tune how it runs and even install it as a service.

Downloading Cantaloupe¶

Releases of Cantaloupe live on the Cantaloupe release page; the latest version can be found here as a `.zip` file.

```
sudo wget -O /opt/cantaloupe.zip CANTALOUPE_RELEASE_URL
sudo unzip /opt/cantaloupe.zip
```

- `CANTALOUPE_RELEASE_URL`: It’s recommended we grab the latest version of Cantaloupe 5. This can be found on the above-linked release page, as the `.zip` version; for example, https://github.com/cantaloupe-project/cantaloupe/releases/download/v5.0.3/cantaloupe-5.0.3.zip - make sure **not** to download the source code zip file as that isn't compiled for running out-of-the-box.

```
sudo rm -R /opt/cantaloupe.zip
sudo mv /opt/[CANTALOUPE_VER] /opt/cantaloupe
```

Creating a Cantaloupe Configuration¶

Cantaloupe pulls its configuration from a file called cantaloupe.properties; there are also some other files that can contain instructions for Cantaloupe while it's running; specifically, we're going to copy over the delegates.rb file, which can also contain custom configuration. We won't make use of this file; we're just copying it over for demonstration purposes.

Creating these files from scratch is *not* recommended; rather, we're going to take the default cantaloupe configurations and plop them into their own folder so we can work with them.

sudo mkdir /opt/cantaloupe_config

~~sudo cp CANTALOUPE_VER/cantaloupe.properties.sample /opt/cantaloupe_config/cantaloupe.properties~~
~~sudo cp CANTALOUPE_VER/delegates.rb.sample /opt/cantaloupe_config/delegates.rb~~
~~- CANTALOUPE_VER: This will depend on the exact version of Cantaloupe downloaded; in the above example release, this would be cantaloupe-5.0.3~~

```
sudo cp /opt/cantaloupe/cantaloupe.properties.sample
/opt/cantaloupe_config/cantaloupe.properties
sudo cp /opt/cantaloupe/delegates.rb.sample
/opt/cantaloupe_config/delegates.rb
```

The out-of-the-box configuration will work fine for our purposes ← (I didn't find this to be true; see next page), but it's highly recommended that you take a look through the cantaloupe.properties and see what changes can be made; specifically, logging to actual logfiles isn't set up by default, so you may want to take a peek at the log.application.SyslogAppender or log.application.RollingFileAppender, as well as changing the logging level.

Customizing the Cantaloupe config file is missing from the online documentation. It's important to at least have users update `source.static` (see below):

```
sudo nano /opt/cantaloupe_config/cantaloupe.properties
```

```
    source.static = HttpSource

    HttpSource.BasicLookupStrategy.url_prefix =

    log.application.FileAppender.pathname =
    /var/log/islandora/cantaloupe-application.log

    log.application.RollingFileAppender.enabled = true

    log.application.RollingFileAppender.pathname =
    /var/log/islandora/cantaloupe-application.log

    log.application.RollingFileAppender.TimeBasedRollingPolicy.fil
    ename_pattern = /var/log/islandora/cantaloupe-application-
    %d{yyyy-MM-dd}.log

    log.error.FileAppender.pathname =
    /var/log/islandora/cantaloupe-error.log

    log.error.RollingFileAppender.enabled = true

    log.error.RollingFileAppender.pathname =
    /var/log/islandora/cantaloupe-error.log

    log.error.RollingFileAppender.TimeBasedRollingPolicy.filename_
    pattern = /var/log/islandora/cantaloupe-error-%d{yyyy-MM-
    dd}.log

    log.access.FileAppender.pathname =
    /var/log/islandora/cantaloupe-access.log

    log.access.RollingFileAppender.enabled = true

    log.access.RollingFileAppender.pathname =
    /var/log/islandora/cantaloupe-access.log

    log.access.RollingFileAppender.TimeBasedRollingPolicy.filename
    _pattern = /var/log/islandora/cantaloupe-access-%d{yyyy-MM-
    dd}.log
```

Installing and configuring Cantaloupe as a service¶

Since it is a standalone application, we can configure Cantaloupe as a systemd service like we did with Tomcat, so it can start on boot:

/etc/systemd/system/cantaloupe.service | root:root/755

```
[Unit]
Description=Cantaloupe

[Service]
ExecStart=java -cp /opt/ cantaloupe /CANTALOUPE_VER.jar -
Dcantaloupe.config=/opt/cantaloupe_config/cantaloupe.properties -Xmx1500m -Xms1000m
edu.illinois.library.cantaloupe.StandaloneEntry
SyslogIdentifier=cantaloupe

[Install]
WantedBy=multi-user.target
```

- CANTALOUPE_VER: This will depend on the exact version of Cantaloupe downloaded; in the above example release, this would be cantaloupe-5.0.3

We can now enable the service and run it:

```
sudo systemctl enable cantaloupe
sudo systemctl start cantaloupe
```

We can check the service status with `sudo systemctl status cantaloupe | grep Active` and the splash screen of Cantaloupe should be available at http://localhost:8182 and http://localhost:8182/iiif/2

In case you need to open port 8182:

```
sudo ufw allow 8182/tcp
sudo ufw status verbose  #check port status
```

Installing Fedora, Syn, and Blazegraph¶

In this section, we will install:¶

- Fedora 6, the back-end repository that Islandora will use
- Syn, the authentication broker that will manage communication with Fedora
- Blazegraph, the resource index layer on top of Fedora for managing discoverability via RDF

Fedora 6¶

Stop the Tomcat Service¶

We're going to stop the Tomcat service while working on setting up Fedora to prevent any autodeploy misconfigurations.

```
sudo systemctl stop tomcat
```

Creating a Working Space for Fedora¶

Fedora's configuration and data won't live with Tomcat itself; rather, we're going to prepare a space for them to make them easier to manage.

```
sudo mkdir -p /opt/fcrepo/data/objects
sudo mkdir /opt/fcrepo/config
sudo chown -R tomcat:tomcat /opt/fcrepo
```

Creating a Database for Fedora¶

The method for creating the database here will closely mimic the method we used to create our database for Drupal.

```
sudo -u postgres psql
create database FEDORA_DB encoding 'UTF8' LC_COLLATE = 'en_US.UTF-8' LC_CTYPE = 'en_US.UTF-8' TEMPLATE template0;
create user FEDORA_DB_USER with encrypted password 'FEDORA_DB_PASSWORD';
grant all privileges on database FEDORA_DB to FEDORA_DB_USER;
\q
```

- FEDORA_DB: fcrepo
  - This will be used as the database Fedora will store the repository in.
- FEDORA_DB_USER: fedora
- FEDORA_DB_PASSWORD: fedora

- Again, this should be a secure password of some kind; leaving it as `fedora` is not recommended.

Create MySQL user and database for Fedora:

```
sudo mysql -u root -p

CREATE DATABASE fcrepo;

CREATE USER '[MYSQL_USER_FOR_FEDORA]'@'localhost' IDENTIFIED BY
'[MYSQL_PASSWORD_FOR_FEDORA]';

GRANT ALL PRIVILEGES ON fcrepo.* TO '[MYSQL_USER_FOR_FEDORA]'@'localhost';
```

Adding a Fedora Configuration¶

The Fedora configuration is going to come in a few different chunks that need to be in place before Fedora will be functional. We're going to place several files outright, with mildly modified parameters according to our configuration.

The basics of these configuration files have been pulled largely from the templates in Islandora-Devops/islandora-playbook internal Fedora role; you may consider referencing the playbook's templates directory for more details.

Namespace prefixes¶

`i8_namespaces.yml` is a list of namespaces used by Islandora that may not necessarily be present in Fedora; we add them here to ensure we can use them in queries.

/opt/fcrepo/config/i8_namespaces.yml | tomcat:tomcat/644

```
# Islandora 8/Fedora namespaces
#
# This file contains ALL the prefix mappings, if a URI
# does not appear in this file it will be displayed as
# the full URI in Fedora.
acl: http://www.w3.org/ns/auth/acl#
bf: http://id.loc.gov/ontologies/bibframe/
cc: http://creativecommons.org/ns#
dc: http://purl.org/dc/elements/1.1/
dcterms: http://purl.org/dc/terms/
dwc: http://rs.tdwg.org/dwc/terms/
ebucore: http://www.ebu.ch/metadata/ontologies/ebucore/ebucore#
exif: http://www.w3.org/2003/12/exif/ns#
fedoraconfig: http://fedora.info/definitions/v4/config#
fedoramodel: info:fedora/fedora-system:def/model#
foaf: http://xmlns.com/foaf/0.1/
geo: http://www.w3.org/2003/01/geo/wgs84_pos#
gn: http://www.geonames.org/ontology#
iana: http://www.iana.org/assignments/relation/
islandorarelsext: http://islandora.ca/ontology/relsext#
islandorarelsint: http://islandora.ca/ontology/relsint#
```

```
ldp: http://www.w3.org/ns/ldp#
memento: http://mementoweb.org/ns#
nfo: http://www.semanticdesktop.org/ontologies/2007/03/22/nfo#
ore: http://www.openarchives.org/ore/terms/
owl: http://www.w3.org/2002/07/owl#
premis: http://www.loc.gov/premis/rdf/v1#
prov: http://www.w3.org/ns/prov#
rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs: http://www.w3.org/2000/01/rdf-schema#
rel: http://id.loc.gov/vocabulary/relators/
schema: http://schema.org/
skos: http://www.w3.org/2004/02/skos/core#
test: info:fedora/test/
vcard: http://www.w3.org/2006/vcard/ns#
webac: http://fedora.info/definitions/v4/webac#
xml: http://www.w3.org/XML/1998/namespace
xmlns: http://www.w3.org/2000/xmlns/
xs: http://www.w3.org/2001/XMLSchema
xsi: http://www.w3.org/2001/XMLSchema-instance
```

Allowed External Content Hosts¶

We have Fedora provide metadata for some resources that are contained in Drupal. Fedora needs to know to allow access to these External Content hosts.

We create a file `/opt/fcrepo/config/allowed_external_hosts.txt | tomcat:tomcat/644`

```
http://localhost:8000/
```

Note: the trailing backslash is important here. For more information on Fedora's External Content and configuring it, see the Fedora Wiki pages

Fedora configuration properties file¶

Fedora 6 now allows you to put all your configuration properties into a single file. We use `0640` permissions as you will want to put your database credentials in here.

`/opt/fcrepo/config/fcrepo.properties | tomcat:tomcat/640`

```
fcrepo.home=FCREPO_HOME
# External content using path defined above.
fcrepo.external.content.allowed=/opt/fcrepo/config/allowed_external_hosts.txt
# Namespace registry using path defined above.
fcrepo.namespace.registry=/opt/fcrepo/config/i8_namespaces.yml
fcrepo.auth.principal.header.enabled=true
# The principal header is the syn-setting.xml "config" element's "header" attribute
fcrepo.auth.principal.header.name=X-Islandora
# false to use manual versioning, true to create a version on each change
fcrepo.autoversioning.enabled=true
fcrepo.db.url=FCREPO_DB_URL
fcrepo.db.user=FCREPO_DB_USERNAME
fcrepo.db.password=FCREPO_DB_PASSWORD
fcrepo.ocfl.root=FCREPO_OCFL_ROOT
fcrepo.ocfl.temp=FCREPO_TEMP_ROOT
fcrepo.ocfl.staging=FCREPO_STAGING_ROOT
# Can be sha512 or sha256
fcrepo.persistence.defaultDigestAlgorithm=sha512
```

```
# Jms moved from 61616 to allow external ActiveMQ to use that port
fcrepo.dynamic.jms.port=61626
# Same as above
fcrepo.dynamic.stomp.port=61623
fcrepo.velocity.runtime.log=FCREPO_VELOCITY_LOG
fcrepo.jms.baseUrl=FCREPO_JMS_BASE
```

- `FCREPO_HOME` - The home directory for all Fedora generated output and state. Unless otherwise specified, all logs, metadata, binaries, and internally generated indexes, etc. It would default to the Tomcat starting directory. A good default would be `/opt/fcrepo`
- `FCREPO_DB_URL` - This parameter allows you to set the database connection url. In general the format is as follows:

  `jdbc:<database_type>://<database_host>:<database_port>/<database_name>`

  Fedora currently supports H2, PostgresQL 12.3, MariaDB 10.5.3, and MySQL 8.0

  So using the default ports for the supported databases here are the values we typically use:

  - PostgresQL: `jdbc:postgresql://localhost:5432/fcrepo`
  - MariaDB: `jdbc:mariadb://localhost:3306/fcrepo`
  - MySQL: `jdbc:mysql://localhost:3306/fcrepo`
- `FCREPO_DB_USERNAME` - The database username
- `FCREPO_DB_PASSWORD` - The database password
- `FCREPO_OCFL_ROOT` - Sets the root directory of the OCFL. Defaults to `FCREPO_HOME/data/ocfl-root` if not set.
- `FCREPO_TEMP_ROOT` - Sets the temp directory used by OCFL. Defaults to `FCREPO_HOME/data/temp` if not set.
- `FCREPO_STAGING_ROOT` - Sets the staging directory used by OCFL. Defaults to `FCREPO_HOME/data/staging` if not set.
- `FCREPO_VELOCITY_LOG` - The Fedora HTML template code uses Apache Velocity, which generates a runtime log called velocity.log. Defaults to `FCREPO_HOME/logs/velocity`. A good choice might be /opt/tomcat/logs/velocity.log
- `FCREPO_JMS_BASE` - This specifies the baseUrl to use when generating JMS messages. You can specify the hostname with or without port and with or without path. If your system is behind a NAT firewall you may need this to avoid your message consumers trying to access the system on an invalid port. If this system property is not set, the host, port and context from the user's request will be used in the emitted JMS messages. If your Alpaca is

on the same machine as your Fedora and you use the `islandora-indexing-fcrepo`, you could use http://localhost:8080/fcrepo/rest.

Check the Lyrasis Wiki to find all of [Fedora's properties]

Adding the Fedora Variables to `JAVA_OPTS`¶

We need our Tomcat `JAVA_OPTS` to include references to our repository configuration.

`/opt/tomcat/bin/setenv.sh`

Before:

3 | export JAVA_OPTS="-Djava.awt.headless=true -server -Xmx1500m -Xms1000m"

After:

3 | export JAVA_OPTS="-Djava.awt.headless=true -Dfcrepo.config.file=/opt/fcrepo/config/fcrepo.properties -DconnectionTimeout=-1 -server -Xmx1500m -Xms1000m"

Ensuring Tomcat Users Are In Place¶

While not strictly necessary, we can use the `tomcat-users.xml` file to give us direct access to the Fedora endpoint. Fedora defines, out of the box, a `fedoraAdmin` and `fedoraUser` role that can be reflected in the users list for access. The following file will also include the base `tomcat` user. As always, these default passwords should likely not stay as the defaults.

`/opt/tomcat/conf/tomcat-users.xml | tomcat:tomcat/600`

```
<?xml version="1.0" encoding="UTF-8"?>
<tomcat-users xmlns="http://tomcat.apache.org/xml"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://tomcat.apache.org/xml tomcat-users.xsd"
        version="1.0">
  <role rolename="tomcat"/>
  <role rolename="fedoraAdmin"/>
  <role rolename="fedoraUser"/>
  <user username="tomcat" password="TOMCAT_PASSWORD" roles="tomcat"/>
  <user username="fedoraAdmin" password="FEDORA_ADMIN_PASSWORD" roles="fedoraAdmin"/>
  <user username="fedoraUser" password="FEDORA_USER_PASSWORD" roles="fedoraUser"/>
</tomcat-users>
```

- `TOMCAT_PASSWORD`: `tomcat`
- `FEDORA_ADMIN_PASSWORD`: `islandora`
- `FEDORA_USER_PASSWORD`: `islandora`

Downloading and Placing the Latest Release¶

Fedora .war files are packaged up as releases on the official GitHub repository. You should download the most recent stable release.

```
sudo wget -O fcrepo.war FCREPO_WAR_URL
sudo mv fcrepo.war /opt/tomcat/webapps
sudo chown tomcat:tomcat /opt/tomcat/webapps/fcrepo.war
```

- FCREPO_WAR_URL: This can be found at the fcrepo downloads page; the file you're looking for is:
  - Tagged in green as the 'Latest release'
  - Named "fcrepo-webapp-VERSION.war"

Start the Tomcat Service¶

As before, start the Tomcat service to get Fedora up and running.

```
sudo systemctl start tomcat
```

Note: sometimes it takes a while for Fedora and Tomcat to start up, usually it shouldn't take longer than 5 minutes.

Once it starts up, Fedora REST API should be available at http://localhost:8080/fcrepo/rest. The username is fedoraAdmin and we defined the password before as FEDORA_ADMIN_PASSWORD (default: "islandora").

Syn¶

Downloading the Syn JAR File¶

A compiled JAR of Syn can be found on the Syn releases page. We're going to add this to the list libraries accessible to Tomcat.

```
sudo wget -P /opt/tomcat/lib SYN_JAR_URL
# Ensure the library has the correct permissions.
sudo chown -R tomcat:tomcat /opt/tomcat/lib
sudo chmod -R 640 /opt/tomcat/lib
```

- SYN_JAR_URL: The latest stable release of the Syn JAR from the releases page. Specifically, the JAR compiled as -all.jar is required.

Generating an SSL Key for Syn¶

For Islandora and Fedora to talk to each other, an SSL key needs to be generated for use with Syn. We're going to make a spot where such keys can live, and generate one.

```
sudo mkdir /opt/keys
sudo openssl genrsa -out "/opt/keys/syn_private.key" 2048
sudo openssl rsa -pubout -in "/opt/keys/syn_private.key" -out "/opt/keys/syn_public.key"
sudo chown www-data:www-data /opt/keys/syn*
```

Placing the Syn Settings¶

Syn sites and tokens belong in a settings file that we're going to reference in Tomcat.

`/opt/fcrepo/config/syn-settings.xml | tomcat:tomcat/600`

```
<config version='1' header='X-Islandora'>
  <site algorithm='RS256' encoding='PEM' anonymous='true' default='true' path='/opt/keys/syn_public.key'/>
  <token user='islandora' roles='fedoraAdmin'>ISLANDORA_SYN_TOKEN</token>
</config>
```

- `ISLANDORA_SYN_TOKEN`: `islandora`
    - This should be a secure generated token rather than this default; it will be configured on the Drupal side later.

Adding the Syn Valve to Tomcat¶

Referencing the valve we've created in our `syn-settings.xml` involves creating a `<Valve>` entry in Tomcat's `context.xml`:

There are two options here:

1. Enable the Syn Valve for all of Tomcat.¶

`/opt/tomcat/conf/context.xml`

Before:

29 | `-->`

30 | `</Context>`

After:

29 | `-->`

30 | <Valve className="ca.islandora.syn.valve.SynValve" pathname="/opt/fcrepo/config/syn-settings.xml"/>

31 | </Context>

2. Enable the Syn Valve for only Fedora.¶

Create a new file at

/opt/tomcat/conf/Catalina/localhost/fcrepo.xml

```
<Context>
  <Valve className="ca.islandora.syn.valve.SynValve" pathname="/opt/fcrepo/config/syn-settings.xml"/>
</Context>
```

Your Fedora web application needs to be deployed in Tomcat with the name `fcrepo.war`. Otherwise, change the name of the above XML file to match the deployed web application's name.

Restarting Tomcat¶

Finally, restart tomcat to apply the new configurations.

```
sudo systemctl restart tomcat
```

Note: sometimes it takes a while for Fedora and Tomcat to start up, usually it shouldn't take longer than 5 minutes.

Note: after installing the Syn valve, you'll no longer be able to manually create/edit or delete objects via Fedora Web UI. All communication with Fedora will now be handled from the Islandora module in Drupal.

Redhat logging¶

Redhat systems have stopped generating an all inclusive `catalina.out`, the `catalina.<date>.log` does not include web application's log statements. To get Fedora log statements flowing, you can create your own LogBack configuration file and point to it.

/opt/fcrepo/config/fcrepo-logback.xml | tomcat:tomcat/644

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration>
<configuration>
 <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
  <encoder>
   <pattern>%p %d{HH:mm:ss.SSS} [%thread] \(%c{0}\) %m%n</pattern>
  </encoder>
```

```
</appender>

<appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
  <file>${catalina.base}/logs/fcrepo.log</file>
  <append>true</append>
  <rollingPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
    <fileNamePattern>${catalina.base}/logs/fcrepo.%d{yyyy-MM-dd}.log.%i</fileNamePattern>
    <maxFileSize>10MB</maxFileSize>
    <maxHistory>30</maxHistory>
    <totalSizeCap>2GB</totalSizeCap>
  </rollingPolicy>
  <encoder>
    <pattern>%p %d{HH:mm:ss.SSS} [%thread] \(%c{0}\) %m%n</pattern>
  </encoder>
</appender>

<logger name="org.fcrepo.auth" additivity="false" level="${fcrepo.log.auth:-null}">
  <appender-ref ref="STDOUT"/>
  <appender-ref ref="FILE"/>
</logger>
<logger name="org.fcrepo.config" additivity="false" level="${fcrepo.log.config:-null}">
  <appender-ref ref="STDOUT"/>
  <appender-ref ref="FILE"/>
</logger>
<logger name="org.fcrepo.event" additivity="false" level="${fcrepo.log.event:-null}">
  <appender-ref ref="STDOUT"/>
  <appender-ref ref="FILE"/>
</logger>
<logger name="org.fcrepo.http.api" additivity="false" level="${fcrepo.log.http.api:-null}">
  <appender-ref ref="STDOUT"/>
  <appender-ref ref="FILE"/>
</logger>
<logger name="org.fcrepo.http.commons" additivity="false" level="${fcrepo.log.http.commons:-null}">
  <appender-ref ref="STDOUT"/>
  <appender-ref ref="FILE"/>
</logger>
<logger name="org.fcrepo.jms" additivity="false" level="${fcrepo.log.jms:-null}">
  <appender-ref ref="STDOUT"/>
  <appender-ref ref="FILE"/>
</logger>
<logger name="org.fcrepo.kernel" additivity="false" level="${fcrepo.log.kernel:-null}">
  <appender-ref ref="STDOUT"/>
  <appender-ref ref="FILE"/>
</logger>
<logger name="org.fcrepo.persistence" additivity="false" level="${fcrepo.log.persistence:-null}">
  <appender-ref ref="STDOUT"/>
  <appender-ref ref="FILE"/>
</logger>
<logger name="org.fcrepo.search" additivity="false" level="${fcrepo.log.search:-null}">
  <appender-ref ref="STDOUT"/>
  <appender-ref ref="FILE"/>
</logger>
<logger name="org.fcrepo.storage" additivity="false" level="${fcrepo.log.storage:-null}">
  <appender-ref ref="STDOUT"/>
  <appender-ref ref="FILE"/>
</logger>
<logger name="org.fcrepo" additivity="false" level="${fcrepo.log:-INFO}">
  <appender-ref ref="STDOUT"/>
  <appender-ref ref="FILE"/>
</logger>

<root level="${fcrepo.log.root:-WARN}">
  <appender-ref ref="STDOUT"/>
  <appender-ref ref="FILE"/>
</root>
```

```
</configuration>
```

Then alter your $JAVA_OPTS like [above](#) to include

```
-Dlogback.configurationFile=/opt/fcrepo/config/fcrepo-logback.xml
```

This will generate a log file at ${catalina.base}/logs/fcrepo.log and will rotate each day or if the logs reaches 10MB. It will maintain 30 days of old logs, or 2GB whichever comes first.

Blazegraph 2¶

Creating a Working Space for Blazegraph¶

Blazegraph needs a space for configurations and data; we're going to create this space in /opt.

```
sudo mkdir -p /opt/blazegraph/data
sudo mkdir /opt/blazegraph/conf
sudo chown -R tomcat:tomcat /opt/blazegraph
```

Downloading and Placing the Blazegraph WAR¶

The Blazegraph .war file can be found in a few different places, but to ensure we're able to easily wget it, we're going to use the maven.org repository link to grab it.

```
cd /opt
sudo wget -O blazegraph.war BLAZEGRAPH_WARFILE_LINK
sudo mv blazegraph.war /opt/tomcat/webapps
sudo chown tomcat:tomcat /opt/tomcat/webapps/blazegraph.war
```

- BLAZEGRAPH_WAR_URL: You can find a link to this at the Maven repository for Blazegraph; you'll want to click the link for the latest version of Blazegraph 2.1.x, then get the link to the .war file within that version folder.

Once this is downloaded, give it a moment to expand before moving on to the next step.

Configuring Logging¶

We would like to have an appropriate logging configuration for Blazegraph, which can be useful for looking at incoming traffic and determining if anything has gone wrong with Blazegraph. Our logger isn't going to be much different than the default logger; it can be made more or less verbose by changing the default WARN levels. There are several other loggers that can be enabled, like a SPARQL query trace or summary query evaluation log; if these are desired they should be added in. Consult the Blazegraph documentation for more details.

/opt/blazegraph/conf/log4j.properties | tomcat:tomcat/644

```
log4j.rootCategory=WARN, dest1

# Loggers.
log4j.logger.com.bigdata=WARN
log4j.logger.com.bigdata.btree=WARN

# Normal data loader (single threaded).
#log4j.logger.com.bigdata.rdf.store.DataLoader=INFO

# dest1
log4j.appender.dest1=org.apache.log4j.ConsoleAppender
log4j.appender.dest1.layout=org.apache.log4j.PatternLayout
log4j.appender.dest1.layout.ConversionPattern=%-5p: %F:%L: %m%n
#log4j.appender.dest1.layout.ConversionPattern=%-5p: %r %l: %m%n
#log4j.appender.dest1.layout.ConversionPattern=%-5p: %m%n
#log4j.appender.dest1.layout.ConversionPattern=%-4r [%t] %-5p %c %x - %m%n
#log4j.appender.dest1.layout.ConversionPattern=%-4r(%d) [%t] %-5p %c(%l:%M) %x - %m%n

# Rule execution log. This is a formatted log file (comma delimited).
log4j.logger.com.bigdata.relation.rule.eval.RuleLog=INFO,ruleLog
log4j.additivity.com.bigdata.relation.rule.eval.RuleLog=false
log4j.appender.ruleLog=org.apache.log4j.FileAppender
log4j.appender.ruleLog.Threshold=ALL
log4j.appender.ruleLog.File=/var/log/blazegraph/rules.log
log4j.appender.ruleLog.Append=true
log4j.appender.ruleLog.BufferedIO=false
log4j.appender.ruleLog.layout=org.apache.log4j.PatternLayout
log4j.appender.ruleLog.layout.ConversionPattern=%m
```

Adding a Blazegraph Configuration¶

Our configuration will be built from a few different files that we will eventually reference in `JAVA_OPTS` and directly apply to Blazegraph; these include most of the functional pieces Blazegraph requires, as well as a generalized configuration for the `islandora` namespace it will use. As with most large configurations like this, these should likely be tuned to your preferences, and the following files only represent sensible defaults.

/opt/blazegraph/conf/RWStore.properties | tomcat:tomcat/644

```
com.bigdata.journal.AbstractJournal.file=/opt/blazegraph/data/blazegraph.jnl
com.bigdata.journal.AbstractJournal.bufferMode=DiskRW
com.bigdata.service.AbstractTransactionService.minReleaseAge=1
com.bigdata.journal.Journal.groupCommit=false
com.bigdata.btree.writeRetentionQueue.capacity=4000
com.bigdata.btree.BTree.branchingFactor=128
com.bigdata.journal.AbstractJournal.initialExtent=209715200
com.bigdata.journal.AbstractJournal.maximumExtent=209715200
com.bigdata.rdf.sail.truthMaintenance=false
com.bigdata.rdf.store.AbstractTripleStore.quads=true
com.bigdata.rdf.store.AbstractTripleStore.statementIdentifiers=false
com.bigdata.rdf.store.AbstractTripleStore.textIndex=false
com.bigdata.rdf.store.AbstractTripleStore.axiomsClass=com.bigdata.rdf.axioms.NoAxioms
com.bigdata.namespace.kb.lex.com.bigdata.btree.BTree.branchingFactor=400
com.bigdata.namespace.kb.spo.com.bigdata.btree.BTree.branchingFactor=1024
com.bigdata.journal.Journal.collectPlatformStatistics=false
```

/opt/blazegraph/conf/blazegraph.properties | tomcat:tomcat/644

```
com.bigdata.rdf.store.AbstractTripleStore.textIndex=false
com.bigdata.rdf.store.AbstractTripleStore.axiomsClass=com.bigdata.rdf.axioms.OwlAxioms
com.bigdata.rdf.sail.isolatableIndices=false
com.bigdata.rdf.store.AbstractTripleStore.justify=true
com.bigdata.rdf.sail.truthMaintenance=true
com.bigdata.rdf.sail.namespace=islandora
com.bigdata.rdf.store.AbstractTripleStore.quads=false
com.bigdata.namespace.islandora.lex.com.bigdata.btree.BTree.branchingFactor=400
com.bigdata.journal.Journal.groupCommit=false
com.bigdata.namespace.islandora.spo.com.bigdata.btree.BTree.branchingFactor=1024
com.bigdata.rdf.store.AbstractTripleStore.geoSpatial=false
com.bigdata.rdf.store.AbstractTripleStore.statementIdentifiers=false
```

/opt/blazegraph/conf/inference.nt | tomcat:tomcat/644

```
<http://pcdm.org/models#memberOf> <http://www.w3.org/2002/07/owl#inverseOf>
<http://pcdm.org/models#hasMember> .
<http://pcdm.org/models#fileOf> <http://www.w3.org/2002/07/owl#inverseOf> <http://pcdm.org/models#hasFile> .
```

Specifying the `RWStore.properties` in `JAVA_OPTS`¶

In order to enable our configuration when Tomcat starts, we need to reference the location of `RWStore.properties` in the `JAVA_OPTS` environment variable that Tomcat uses.

/opt/tomcat/bin/setenv.sh

Before:

3 | export JAVA_OPTS="-Djava.awt.headless=true -Dfcrepo.config.file=/opt/fcrepo/config/fcrepo.properties -DconnectionTimeout=-1 -server -Xmx1500m -Xms1000m"

After:

3 | export JAVA_OPTS="-Djava.awt.headless=true -Dfcrepo.config.file=/opt/fcrepo/config/fcrepo.properties -DconnectionTimeout=-1 -Dcom.bigdata.rdf.sail.webapp.ConfigParams.propertyFile=/opt/blazegraph/conf/RWStore.properties -Dlog4j.configuration=file:/opt/blazegraph/conf/log4j.properties -server -Xmx1500m -Xms1000m"

OR...

instead of fully replacing JAVA_OPTS (e.g. the *before/after* on previous page), the following could just be added to the end of JAVA_OPTS, so no user-specific settings get unintentionally lost or overwritten:

```
-Dcom.bigdata.rdf.sail.webapp.ConfigParams.propertyFile=/opt/blazegraph/conf/RWStore.properties -Dlog4j.configuration=file:/opt/blazegraph/conf/log4j.properties
```

Restarting Tomcat¶

Finally, restart Tomcat to pick up the changes we've made.

```
sudo systemctl restart tomcat
```

Installing Blazegraph Namespaces and Inference¶

The two other files we created, blazegraph.properties and inference.nt, contain information that Blazegraph requires in order to establish and correctly use the datasets Islandora will send to it. First, we need to create a dataset - contained in blazegraph.properties - and then we need to inform that dataset of the inference set we have contained in inference.nt.

```
curl -X POST -H "Content-Type: text/plain" --data-binary @/opt/blazegraph/conf/blazegraph.properties
http://localhost:8080/blazegraph/namespace
```

If this worked correctly, Blazegraph should respond with "CREATED: islandora" to let us know it created the islandora namespace.

```
curl -X POST -H "Content-Type: text/plain" --data-binary @/opt/blazegraph/conf/inference.nt
http://localhost:8080/blazegraph/namespace/islandora/sparql
```

If this worked correctly, Blazegraph should respond with some XML letting us know it added the 2 entries from inference.nt to the namespace.

In case you need to open port 8080:

```
sudo ufw allow 8080/tcp

sudo ufw status verbose  #check port status
```

Installing Solr¶

In this section, we will install:¶

- Apache Solr 8, the search engine used to index and find Drupal content
- search_api_solr, the Solr implementation of Drupal's search API

Solr 8¶

Downloading and Placing Solr¶

The Solr binaries can be found at the Solr downloads page; the most recent stable release of Solr 8 should be used.

```
# While generally we download tarballs as .tar.gz files without version
# information, the Solr installer is a bit particular in that it expects a .tgz
# file with the same name as the extracted folder it contains. It's odd, and we
# can't really get around it.
cd /opt
wget SOLR_DOWNLOAD_LINK
sudo mv '[SOLR_TARBALL]?action=download' [SOLR_TARBALL]
sudo tar -xzvf SOLR_TARBALL
```

- SOLR_DOWNLOAD_LINK: **NOTICE**: This will depend on a few different things, not least of all the current version of Solr. The link to the .tgz for the binary on the downloads page will take you to a list of mirrors that Solr can be downloaded from, and provide you with a preferred mirror at the top. This preferred mirror should be used as the SOLR_DOWNLOAD_LINK. - SOLR_TARBALL: The filename that was downloaded, e.g., solr-8.9.0.tgz

Running the Solr Installer¶

Solr includes an installer that does most of the heavy lifting of ensuring we have a Solr user, a location where Solr lives, and configurations in place to ensure it's running on boot.

```
sudo UNTARRED_SOLR_FOLDER/bin/install_solr_service.sh SOLR_TARBALL
```

- UNTARRED_SOLR_FOLDER: This will likely simply be solr-VERSION, where VERSION is the version number that was downloaded.

The port that Solr runs on can potentially be configured at this point, but we'll expect it to be running on 8983.

Wait until the command output reaches:

```
Started Solr server on port 8983 (pid=****). Happy searching!
systemd[1]: Started LSB: Controls Apache Solr as a Service.
```

After which you can press `q` to quit the output (this won't kill Solr so it's safe).

You can check if Solr is running correctly by going to http://localhost:8983/solr

Increasing the Open File Limit (Optional)¶

Solr's installation guide recommends that you increase the open file limit so that operations aren't disrupted while Solr is trying to access things in its index. This limit can be increased while the system is running, but doing so won't persist after a reboot. You can hard-increase this limit using your system's `sysctl` file:

```
/etc/sysctl.conf
```

Add the following line to the end of the file:

```
fs.file-max = 65535
```

Then apply your new configuration.

```
sudo sysctl -p
```

Creating a New Solr Core¶

Initially, our new Solr core will contain a configuration copied from the example included with the installation, so that we have something to work with when we configure this on the Drupal side. We'll later update this with generated configurations we create in Drupal.

```
cd /opt/solr
sudo mkdir -p /var/solr/data/SOLR_CORE/conf
sudo cp -r example/files/conf/* /var/solr/data/SOLR_CORE/conf
sudo chown -R solr:solr /var/solr
sudo -u solr bin/solr create -c SOLR_CORE -p 8983
```

- SOLR_CORE: islandora8

```
sudo /opt/solr/bin/solr restart
```

You should see an output similar to this:

```
WARNING: Using _default configset with data driven schema functionality. NOT RECOMMENDED for production use.
    To turn off: bin/solr config -c islandora8 -p 8983 -action set-user-property -property update.autoCreateFields -value false

Created new core 'islandora8'
```

Installing search_api_solr¶

Rather than use an out-of-the-box configuration that won't be suitable for our purposes, we're going to use the Drupal search_api_solr module to generate one for us. This will also require us to install the module so we can create these configurations using Drush.

~~cd /opt/drupal~~
~~sudo -u www-data composer require drupal/search_api_solr:^4.2~~
~~drush -y en search_api_solr~~

Note that in the online documentation, the path to the Drupal site defined above **doesn't match** the path to the Drupal site specified earlier (**`/opt` vs. `/var/www`**) :

```
cd /var/www/html/[DRUPAL_DIRECTORY_NAME]

/var/www/html/[DRUPAL_DIRECTORY_NAME]/vendor/drush/drush/drush
solr-gsc default_solr_server
/var/www/html/[DRUPAL_DIRECTORY_NAME]/solrconfig.zip

sudo unzip -d ~/solrconfig solrconfig.zip

sudo cp ~/solrconfig/* /var/solr/data/[SOLR_CORE]/conf

sudo rm -R /var/www/html/[DRUPAL_DIRECTORY_NAME]/solrconfig.zip

sudo systemctl restart solr
```

You should see an output similar to this:

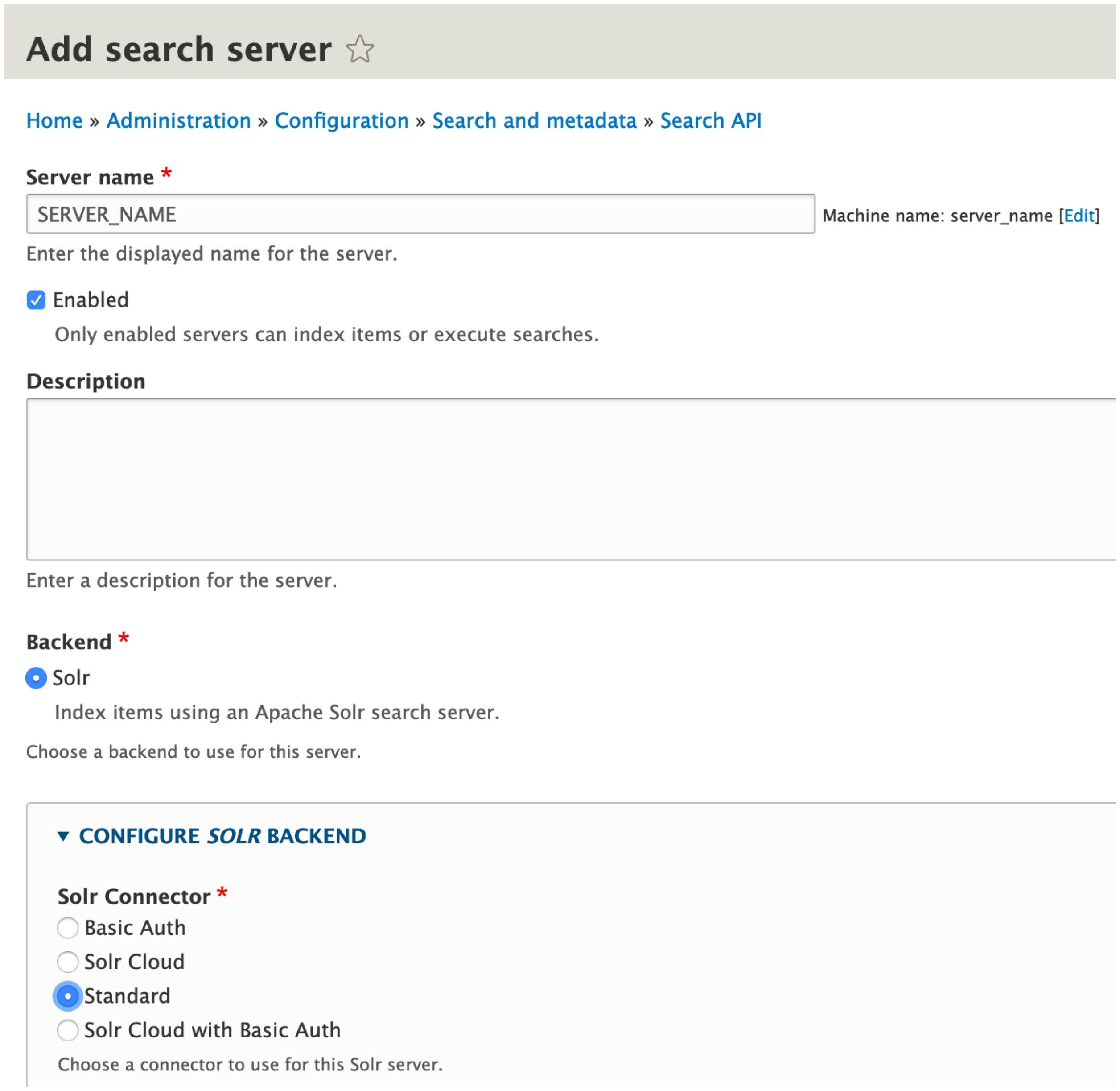
```
The following module(s) will be enabled: search_api_solr, language, search_api

 // Do you want to continue?: yes.

 [success] Successfully enabled: search_api_solr, language, search_api
```

Configuring search_api_solr¶

Before we can create configurations to use with Solr, the core we created earlier needs to be referenced in Drupal.

Log in to the Drupal site at /user using the sitewide administrator username and password (if using defaults from previous chapters this should be islandora and islandora), then navigate to /admin/config/search/search-api/add-server.

Fill out the server addition form using the following options:



Add search server

Home » Administration » Configuration » Search and metadata » Search API

Server name *

SERVER_NAME

Machine name: server_name [Edit]

Enter the displayed name for the server.

[✓] Enabled

Only enabled servers can index items or execute searches.

Description

Enter a description for the server.

Backend *

[● Solr]

Index items using an Apache Solr search server.

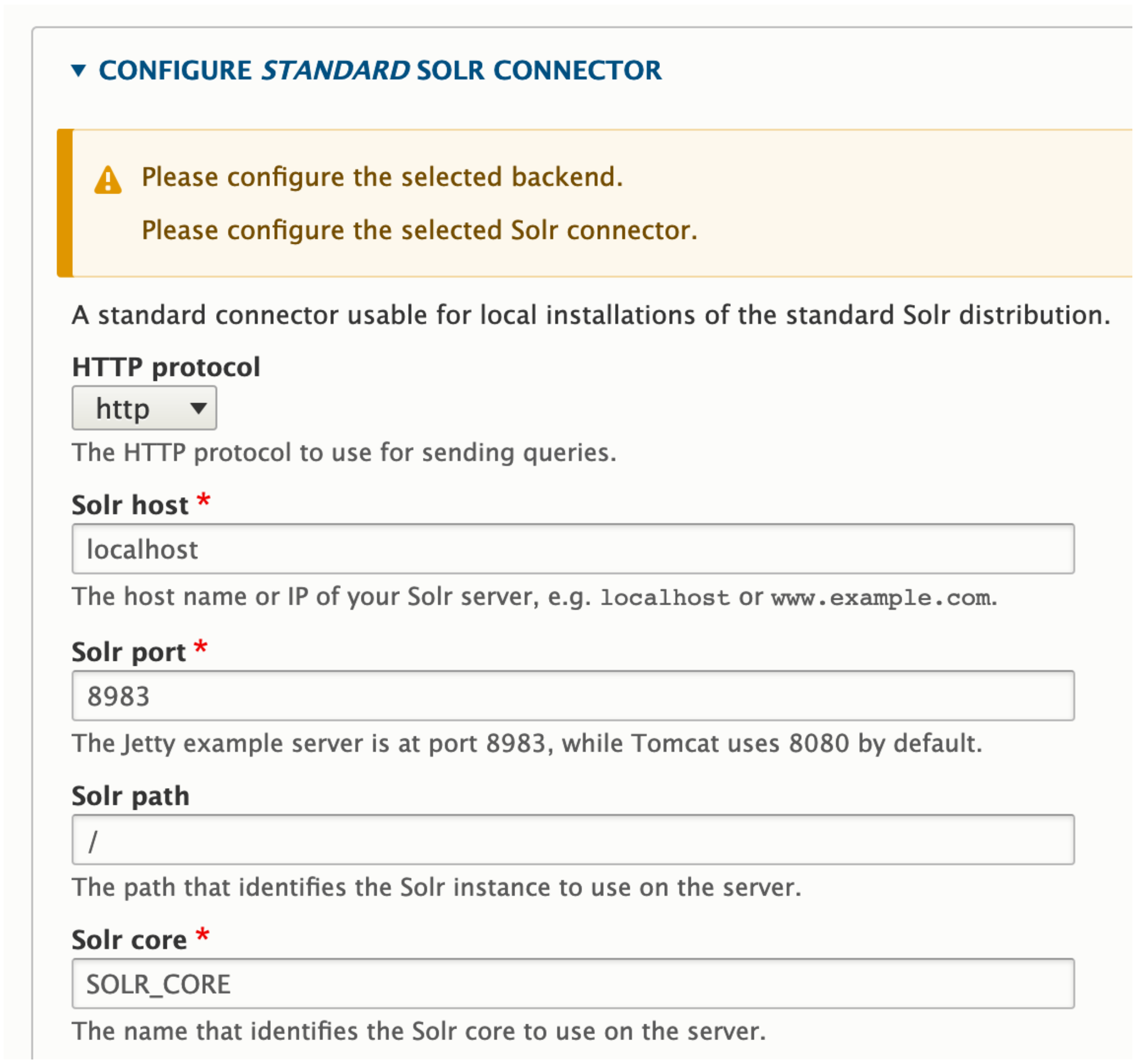
Choose a backend to use for this server.

▼ CONFIGURE *SOLR* BACKEND

Solr Connector *

Basic Auth
Solr Cloud
[● Standard]
Solr Cloud with Basic Auth

Choose a connector to use for this Solr server.


▼ CONFIGURE *STANDARD* SOLR CONNECTOR

Please configure the selected backend.

Please configure the selected Solr connector.

A standard connector usable for local installations of the standard Solr distribution.

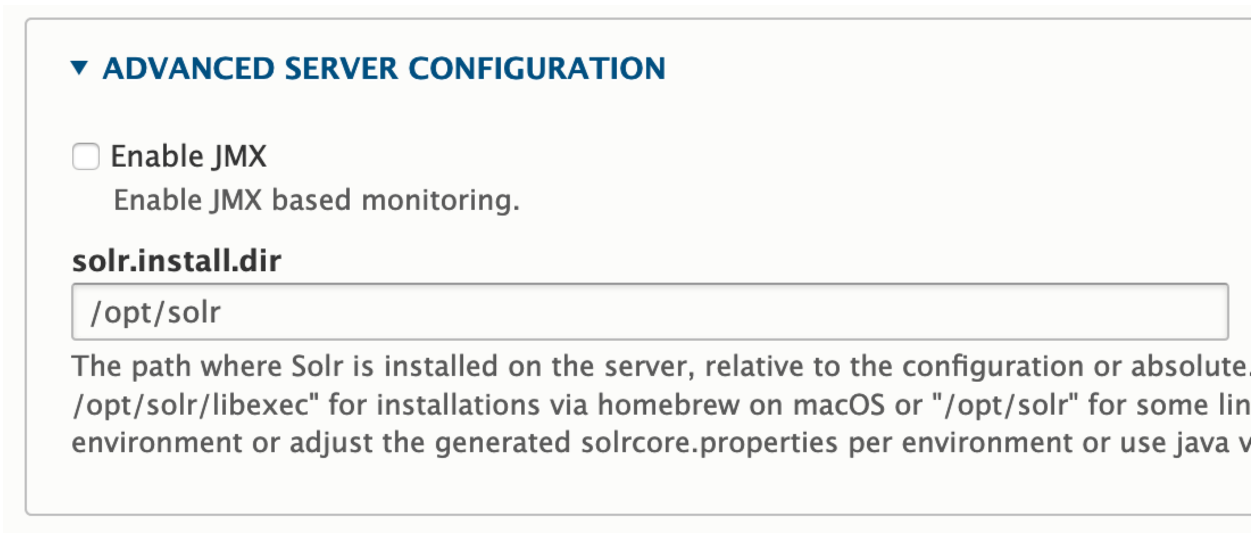
HTTP protocol
http ▼
The HTTP protocol to use for sending queries.

Solr host *
localhost
The host name or IP of your Solr server, e.g. `localhost` or `www.example.com`.

Solr port *
8983
The Jetty example server is at port 8983, while Tomcat uses 8080 by default.

Solr path
/
The path that identifies the Solr instance to use on the server.

Solr core *
SOLR_CORE
The name that identifies the Solr core to use on the server.



Enable JMX

Enable JMX based monitoring.

solr.install.dir

/opt/solr

The path where Solr is installed on the server, relative to the configuration or absolute.
/opt/solr/libexec" for installations via homebrew on macOS or "/opt/solr" for some lin
environment or adjust the generated solrcore.properties per environment or use java v

- SERVER_NAME: islandora8
    - This is completely arbitrary, and is simply used to differentiate this search server configuration from all others. **Write down** or otherwise pay attention to the machine_name generated next to the server name you type in; this will be used in the next step.

As a recap for this configuration:

- **Server name** should be an arbitrary identifier for this server
- **Enabled** should be checked
- **Backend** should be set to **Solr**
- Under **CONFIGURE SOLR BACKEND**, **Solr Connector** should be set to **Standard**
- Under **CONFIGURE STANDARD SOLR CONNECTOR**:
    - **HTTP protocol** is simply set to **http** since we've set this up on the same machine Drupal lives on. On a production installation, Solr should likely be installed behind an HTTPS connection.
    - **Solr host** can be set to **localhost** since, again, this is set up on the same machine Drupal lives on. On a production installation, this may vary, especially if parts of the installation live on different severs
    - **Solr port** should be set to the port Solr was installed on, which is **8983** by default
    - **Solr path** should be set to the configured path to the instance of Solr; in a default installation, there is only one Solr instance, and it lives at **/**

    - **Solr core** should be the name of the Solr core you created earlier, which is why it's listed as **SOLR_CORE** here
- Under **ADVANCED SERVER CONFIGURATION**, **solr.install.dir** should be set to the path where we installed Solr, which this guide has established at **/opt/solr**

Click **Save** to create the server configuration.

NOTICE You can ignore the error about an incompatible Solr schema; we're going to set this up in the next step. In fact, if you refresh the page after restarting Solr in the next step, you should see the error disappear.

Generating and Applying Solr Configurations¶

Now that our core is in place and our Drupal-side configurations exist, we're ready to generate Solr configuration files to connect this site to our search engine.

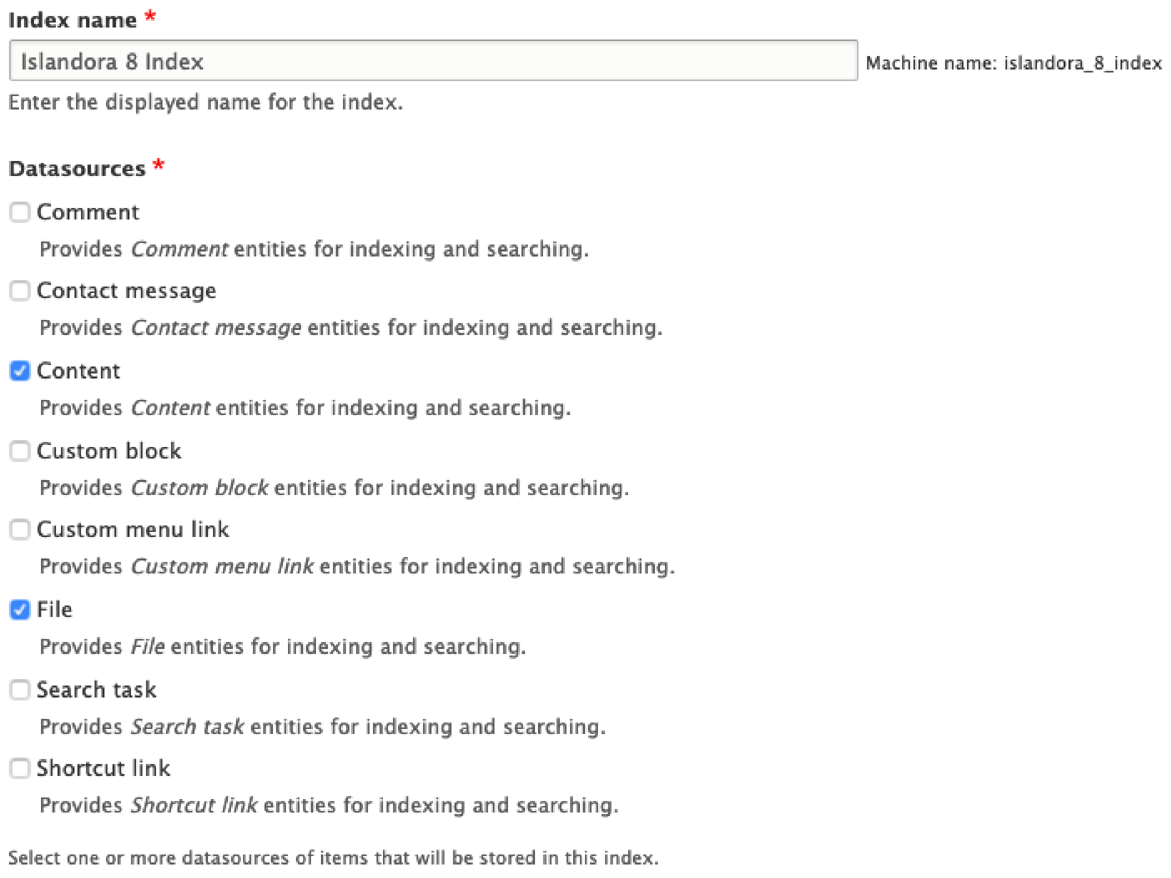
```
cd /opt/drupal
drush solr-gsc SERVER_MACHINE_NAME /opt/drupal/solrconfig.zip
unzip -d ~/solrconfig solrconfig.zip
sudo cp ~/solrconfig/* /var/solr/data/SOLR_CORE/conf
sudo systemctl restart solr
```

- `SERVER_MACHINE_NAME`: This should be the `machine_name` that was automatically generated when creating the configuration in the above step.

Adding an Index¶

In order for content to be indexed back into Solr, a search index needs to be added to our server. Navigate to `/admin/config/search/search-api/add-index` and check off the things you'd like to be indexed.

NOTICE You should come back here later and reconfigure this after completing the last step in this guide. The default indexing configuration is pretty permissive, and you may want to restrict, for example, indexed content to just Islandora-centric bundles. This guide doesn't set up the index's fields either, which are going to be almost wholly dependent on the needs of your installation. Once you complete that configuration later on, re-index Solr from the configuration page of the index we're creating here.


Index name *

Islandora 8 Index

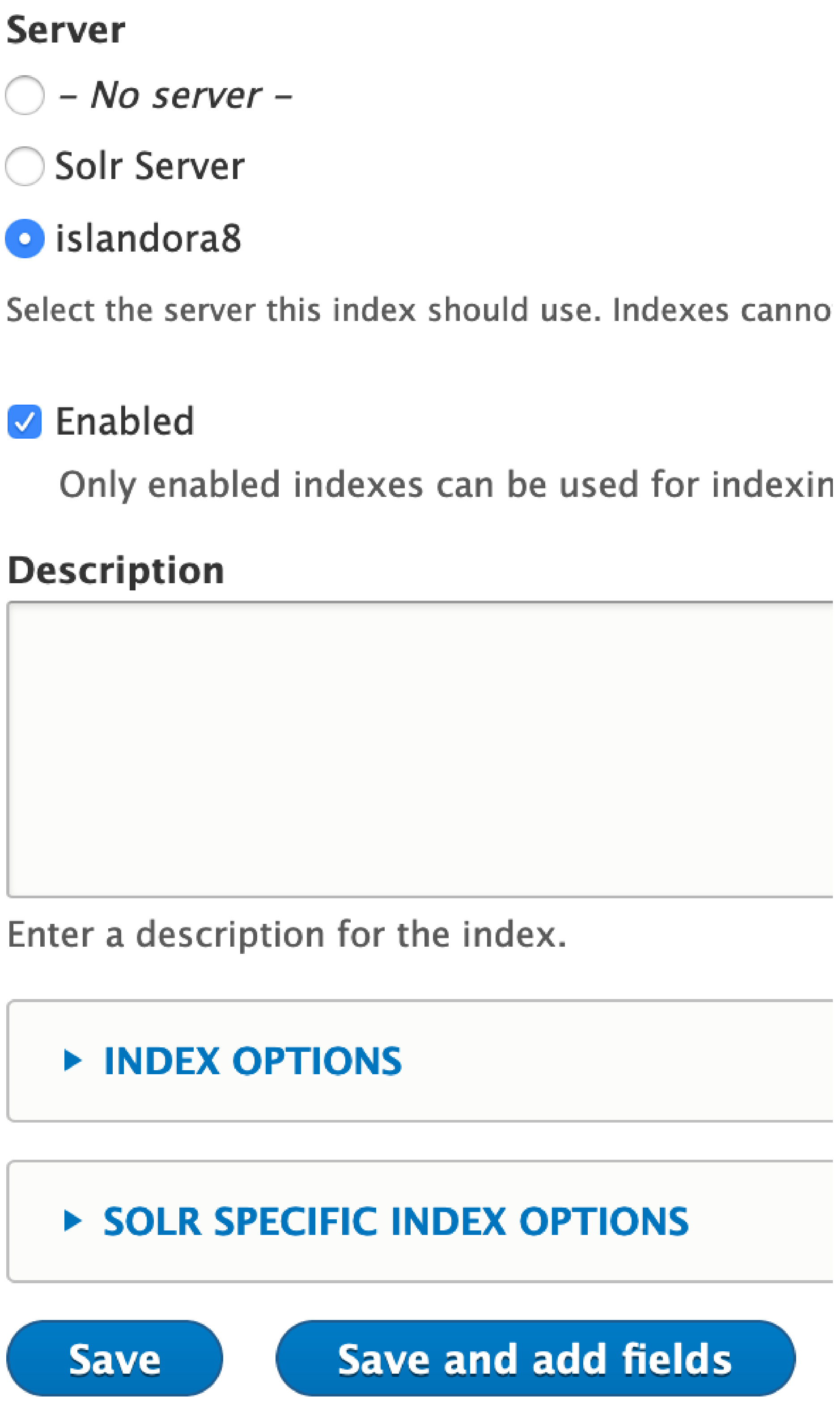
Machine name: islandora_8_index

Enter the displayed name for the index.

Datasources *

- [ ] Comment
  Provides *Comment* entities for indexing and searching.
- [ ] Contact message
  Provides *Contact message* entities for indexing and searching.
- [✓] Content
  Provides *Content* entities for indexing and searching.
- [ ] Custom block
  Provides *Custom block* entities for indexing and searching.
- [ ] Custom menu link
  Provides *Custom menu link* entities for indexing and searching.
- [✓] File
  Provides *File* entities for indexing and searching.
- [ ] Search task
  Provides *Search task* entities for indexing and searching.
- [ ] Shortcut link
  Provides *Shortcut link* entities for indexing and searching.

Select one or more datasources of items that will be stored in this index.


Server

( ) *– No server –*

( ) Solr Server

(●) islandora8

Select the server this index should use. Indexes canno

[✓] Enabled

Only enabled indexes can be used for indexin

Description

Enter a description for the index.

Click **Save** to add your index and kick off indexing of existing items.

Installing Crayfish¶

Needs Maintenance

The manual installation documentation is in need of attention. We are aware that some components no longer work as documented here. If you are interested in helping us improve the documentation, please see Contributing.

In this section, we will install:¶

- **FITS Web Service** (*missing* *from current documentation*)
- Islandora/Crayfish, the suite of microservices that power the backend of Islandora 2.0
- **Individual** ***(misspelled)*** microservices underneath Crayfish

FITS Web Service

Resources: https://projects.iq.harvard.edu/fits/downloads
https://github.com/Islandora/Crayfish/tree/4.x/CrayFits
Install:

```
cd /opt

sudo wget https://github.com/harvard-
lts/fits/releases/download/[FITS_VERSION_NUMBER]/fits-
[FITS_VERSION_NUMBER].zip

sudo unzip /opt/fits-[FITS_VERSION_NUMBER].zip -d /opt/fits

sudo wget -O fits.war
https://projects.iq.harvard.edu/sites/projects.iq.harvard.edu/files
/fits/files/fits-[FITS_WAR_VERSION_NUMBER].war

sudo mv fits.war /opt/tomcat/webapps

sudo chown tomcat:tomcat /opt/tomcat/webapps/fits.war

sudo systemctl restart tomcat

sudo nano /opt/tomcat/conf/catalina.properties
```

Add to bottom of file:

```
fits.home=/opt/fits
shared.loader=/opt/fits/lib/*.jar
```

```
sudo systemctl restart tomcat
```

Test: http://localhost:8080/fits/

Crayfish 2.0[¶](#)

Installing Prerequisites[¶](#)

This section contains the most inaccuracies. There are incorrect file extensions and paths; customized settings that end up getting completely ignored and/or are redundant; plus, certain key software components are **missing**. My guess is that this section is still optimized to work with Karaf, which has since been removed.

Some packages need to be installed before we can proceed with installing Crayfish; these packages are used by the microservices within Crayfish. These include:

- Imagemagick, which will be used for image processing. We'll be using the LYRASIS build of imagemagick here, which supports JP2 files.
- Tesseract, which will be used for optical character recognition; note that by default Tesseract can only understand English; several other individual Tesseract language packs can be installed using apt-get, and a list of available packs can be procured with sudo apt-cache search tesseract-ocr
- FFMPEG, which will be used for video processing
- Poppler, which will be used for generating PDFs

```
sudo apt-get install software-properties-common
```

```
sudo add-apt-repository -y ppa:lyrasis/imagemagick-jp2
sudo apt-get update
sudo apt-get -y install imagemagick tesseract-ocr ffmpeg poppler-utils
```

NOTICE: If you get the sudo: apt-add-repository: command not found, run sudo apt-get install software-properties-common in order to make the command available.

Cloning and Installing Crayfish[¶](#)

We're going to clone Crayfish to /opt, and individually run composer install against each of the microservice subdirectories.

```
cd /opt
sudo git clone https://github.com/Islandora/Crayfish.git crayfish
sudo chown -R www-data:www-data crayfish
sudo -u www-data composer install -d crayfish/Homarus
sudo -u www-data composer install -d crayfish/Houdini
sudo -u www-data composer install -d crayfish/Hypercube
sudo -u www-data composer install -d crayfish/Milliner
sudo -u www-data composer install -d crayfish/Recast
```

Add **(missing) CrayFits microservice**, which exists in the "Desktop Docker Portainer Demo":

```
sudo -u www-data composer install -d crayfish/CrayFits
```

Preparing Logging¶

Not much needs to happen here; Crayfish opts for a simple logging approach, with one `.log` file for each component. We'll create a folder where each logfile can live.

```
sudo mkdir /var/log/islandora
sudo chown www-data:www-data /var/log/islandora
```

Configuring Crayfish Components¶

Each Crayfish component requires one or more `.yaml` file(s) to ensure everything is wired up correctly.

NOTICE

The following configuration files represent somewhat sensible defaults; you should take consideration of the logging levels in use, as this can vary in desirability from installation to installation. Also note that in all cases, `http` URLs are being used, as this guide does not deal with setting up https support. In a production installation, this should not be the case. These files also assume a connection to a PostgreSQL database; use a `pdo_mysql` driver and the appropriate `3306` port if using MySQL.

Homarus (Audio/Video derivatives)¶

/opt/crayfish/ rus/**cfg/**config.yaml | www-data data/644

```
---
homarus
  execu
  mi
    - 
    - au
    - audio
    - image/j
    - image/png
   default: video/
  mime_to_format:
   valid:
    - video/mp4_mp4
    - video/x-msvideo_avi
    - video/ogg_ogg
    - audio/x-wav_w
    - audio/mpeg
    - audio/aac
    - image/i
    - imag
   defa
fedo
sym
  enab
  config: /                    gs.xml
```

Configuration files located in the /cfg/ directory are ignored.

The **/cfg/** directory may have been used by Karaf; but that's no longer a part of the Islandora environment.

Avoid the following error (when accessing this service through a browser):

- `The authenticator manager no longer has "anonymous" security. Please remove this option under the "main" firewall.`
- `sudo nano /opt/crayfish/Homarus/config/packages/security.yaml`
    - under "firewalls" => "main", set "`anonymous`" to "`false`"

Add/Update log path:

```
sudo nano /opt/crayfish/Homarus/config/packages/monolog.yaml

        homarus:
            type: rotating_file
            path: /var/log/islandora/Homarus.log
            level: DEBUG
            max_files: 1
            channels: ["!event", "!console"]
```

```
sudo nano
/opt/crayfish/Homarus/config/packages/crayfish_commons.yaml

     crayfish_commons:
       fedora_base_uri: 'http://localhost:8080/fcrepo/rest'
       apix_middleware_enabled: true
```

Houdini (Image derivatives)¶

Currently the Houdini microservice uses a different system (Symfony) than the other microservices, this requires different configuration.

/opt/crayfish/Houdini/config/services.yaml | www-data:www-data/644

```
# This file is the entry point to configure your own services.
# Files in the packages/ subdirectory configure your dependencies.
# Put parameters here that don't need to change on each machine where the app is deployed
# https://symfony.com/doc/current/best_practices/configuration.html#application-related-configuration
parameters:
  app.executable: ~~/usr/local/bin/convert~~
  app.formats.valid:
    - image/jpeg
    - image/png
    - image/tiff
    - image/jp2
  app.formats.default: image/jpeg

services:
  # default configuration for services in *this* file
  _defaults:
    autowire: true      # Automatically injects dependencies in your services.
    autoconfigure: true # Automatically registers your services as commands, event subscribers, etc.

  # makes classes in src/ available to be used as services
  # this creates a service per class whose id is the fully-qualified class name
  App\Islandora\Houdini\:
    resource: '../src/*'
    exclude: '../src/{DependencyInjection,Entity,Migrations,Tests,Kernel.php}'

  # controllers are imported separately to make sure services can be injected
  # as action arguments even if you don't extend any base controller class
  App\Islandora\Houdini\Controller\HoudiniController:
    public: false
    bind:
      $formats: '%app.formats.valid%'
      $default_format: '%app.formats.default%'
      $executable: '%app.executable%'
    tags: ['controller.service_arguments']

  # add more service definitions when explicit configuration is needed
  # please note that last definitions always *replace* previous ones
```

```
/usr/bin/convert
# this is where my executable exists
```

/opt/crayfish/Houdini/config/packages/~~crayfish_commons.yml~~ | www-data:www-data/644

Filename should be "crayfish_commons.**yaml**" (not **.yml**, which gets ignored)

```
crayfish_commons:
  fedora_base_uri: 'http://localhost:8080/fcrepo/rest'
  ~~syn_config: '/opt/fcrepo/config/syn-settings.xml'~~
```

`syn_config` as a setting no longer works and results in an error.
Maybe replace with: `apix_middleware_enabled: true`

/opt/crayfish/Houdini/config/packages/~~monolog.yml~~ | www-data:www-data/644

Filename should be "monolog.**yaml**" (not **.yml**, which gets ignored)

```
monolog:

  handlers:

    houdini:
      type: rotating_file
      path: /var/log/islandora/Houdini.log
      level: DEBUG
      max_files: 1
```

The below files are two versions of the same file to enable or disable JWT token authentication.

/opt/crayfish/Houdini/config/packages/~~security.yml~~ | www-data:www-data/644

Filename should be "security.**yaml**" (not **.yml**, which gets ignored)

Enabled JWT token authentication:

```
security:

  # https://symfony.com/doc/current/security.html#where-do-users-come-from-user-providers
  providers:
    jwt_user_provider:
      id: Islandora\Crayfish\Commons\Syn\JwtUserProvider

  firewalls:
    dev:
      pattern: ^/(_(profiler|wdt)|css|images|js)/
      security: false
    main:
      anonymous: false
      # Need stateless or it reloads the User based on a token.
      stateless: true

      provider: jwt_user_provider
      guard:
        authenticators:
          - Islandora\Crayfish\Commons\Syn\JwtAuthenticator
```

The "Enabled JWT token authentication" settings didn't work for me and resulted in error messages.

The "Disabled JWT token authentication" settings worked fine.

```
            # activate different ways to authenticate
            # https://symfony.com/doc/current/security.html#firewalls-authentication

            # https://symfony.com/doc/current/security/impersonating_user.html
            # switch_user: true


    # Easy way to control access for large sections of your site
    # Note: Only the *first* access control that matches will be used
    access_control:
        # - { path: ^/admin, roles: ROLE_ADMIN }
        # - { path: ^/profile, roles: ROLE_USER }
```

Disabled JWT token authentication:

```
security:

    # https://symfony.com/doc/current/security.html#where-do-users-come-from-user-providers
    providers:
        jwt_user_provider:
            id: Islandora\Crayfish\Commons\Syn\JwtUserProvider

    firewalls:
        dev:
            pattern: ^/(_(profiler|wdt)|css|images|js)/
            security: false
        main:
            anonymous: true
            # Need stateless or it reloads the User based on a token.
            stateless: true
```

Hypercube (OCR)¶

/opt/crayfish/Hypercube/**cfg**/config.yaml | www-data:www-data/644

```
---
hyp
  t            le: tesseract
               dftote

  b
log:
  level: N
  file: /var/l
syn:
  enab
  c
```

All config files located in the **/cfg/** directory are ignored.

Avoid the following error (when accessing this service through a browser):

- `The authenticator manager no longer has "anonymous" security. Please remove this option under the "main" firewall.`
- `sudo nano /opt/crayfish/Hypercube/config/packages/security.yaml`
  - under "firewalls" => "main", set "`anonymous`" to "`false`"

Add/Update log path:

```
sudo nano /opt/crayfish/Hypercube/config/packages/monolog.yaml

        hypercube:
            type: rotating_file
            path: /var/log/islandora/Hypercube.log
            level: DEBUG
            max_files: 1
            channels: ["!event", "!console"]
```

Milliner (Fedora indexing)¶

/opt/crayfish/Milliner/**cfg/**config.yaml | www-data:www-data/644

```
---
fed                 ://localhost:808
                    /localhost
                     http:/

db:
  drive
  host: 12
  port: 5432
  dbname:
  user:
  pas
l
                       illi

  co              po/config/syn-setting
```

All config files located in the **/cfg/** directory are ignored.

Avoid the following error (when accessing this service through a browser):

- `The authenticator manager no longer has "anonymous" security. Please remove this option under the "main" firewall.`
- `sudo nano /opt/crayfish/Milliner/config/packages/security.yaml`
  - under "firewalls" => "main", set "`anonymous`" to "`false`"

Add/Update log path:

```
sudo nano /opt/crayfish/Milliner/config/packages/monolog.yaml

        milliner:
            type: rotating_file
            path: /var/log/islandora/Milliner.log
            level: DEBUG
            max_files: 1
            channels: ["!event", "!console"]
```

Recast (Drupal to Fedora URI re-writing)¶

Is Recast still necessary? It doesn't exist in the "Desktop Docker Portainer Demo" and it has never generated a log file for me, even after ingesting items into my environment.

/opt/cra[illegible]h/Recast/**cfg/**config.yaml [illegible]ww-data:www-data/644

```
---
fe[illegible]
[illegible]8080
de[illegible]
log:
  level: l[illegible]
  file: /var/[illegible]
syn:
  enable: tr[illegible]
  config: [illegible]
name[illegible]
- [illegible]
[illegible]de[illegible]
[illegible]g/ns/ldp#[illegible]
  me[illegible]mentoweb.org/[illegible]
  pcdm[illegible]m.org/models#"
  pcdmus[illegible]//pcdm.org/use#"
  webac: "ht[illegible]://fedora.info/definitions/v4/we[illegible]c#"
  vcard: "http://www.w3.org/2006/vcard/ns#"
```

All config files located in the **/cfg/** directory are ignored.

Avoid the following error (when accessing this service through a browser):

- `The authenticator manager no longer has "anonymous" security. Please remove this option under the "main" firewall.`
- `sudo nano /opt/crayfish/Recast/config/packages/security.yaml`
    - under "firewalls" => "main", set "`anonymous`" to "`false`"

Add/Update log path:

`sudo nano /opt/crayfish/Recast/config/packages/monolog.yaml`

```
        recast:
            type: rotating_file
            path: /var/log/islandora/Recast.log
            level: DEBUG
            max_files: 1
            channels: ["!event", "!console"]
```

Creating Apache Configurations for Crayfish Components¶

Finally, we need appropriate Apache configurations for Crayfish; these will allow other services to connect to Crayfish components via their HTTP endpoints.

Each endpoint we need to be able to connect to will get its own .conf file, which we will then enable.

NOTICE

These configurations would potentially have collisions with Drupal routes, if any are created in Drupal with the same name. If this is a concern, it would likely be better to reserve a subdomain or another port specifically for Crayfish. For the purposes of this installation guide, these endpoints will suffice.

/etc/apache2/conf-available/Homarus.conf | root:root/644

```
Alias "/homarus" "~~/opt/crayfish/Homarus/src~~"
<Directory "~~/opt/crayfish/Homarus/src~~">
  FallbackResource /homarus/index.php
  Require all granted
  DirectoryIndex index.php
  SetEnvIf Authorization "(.*)" HTTP_AUTHORIZATION=$1
</Directory>
```

`/opt/crayfish/Homarus`**`/public`**

/etc/apache2/conf-available/Houdini.conf | root:root/644

```
Alias "/houdini" "/opt/crayfish/Houdini/public"
<Directory "/opt/crayfish/Houdini/public">
  FallbackResource /houdini/index.php
  Require all granted
  DirectoryIndex index.php
  SetEnvIf Authorization "(.*)" HTTP_AUTHORIZATION=$1
</Directory>
```

/etc/apache2/conf-available/Hypercube.conf | root:root/644

```
Alias "/hypercube" "~~/opt/crayfish/Hypercube/src~~"
<Directory "~~/opt/crayfish/Hypercube/src~~">
  FallbackResource /hypercube/index.php
  Require all granted
  DirectoryIndex index.php
  SetEnvIf Authorization "(.*)" HTTP_AUTHORIZATION=$1
</Directory>
```

`/opt/crayfish/ Hypercube`**`/public`**

/etc/apache2/conf-available/Milliner.conf | root:root/644

```
Alias "/milliner" "~~/opt/crayfish/Milliner/src~~"
<Directory "~~/opt/crayfish/Milliner/src~~">
  FallbackResource /milliner/index.php
  Require all granted
  DirectoryIndex index.php
  SetEnvIf Authorization "(.*)" HTTP_AUTHORIZATION=$1
</Directory>
```

`/opt/crayfish/ Milliner`**`/public`**

/etc/apache2/conf-available/Recast.conf | root:root/644

```
Alias "/recast" "~~/opt/crayfish/Recast/src~~"
<Directory "~~/opt/crayfish/Recast/src~~">
  FallbackResource /recast/index.php
  Require all granted
  DirectoryIndex index.php
  SetEnvIf Authorization "(.*)" HTTP_AUTHORIZATION=$1
</Directory>
```

`/opt/crayfish/Recast/public`

Add *missing* Apache configuration for **CrayFits**:

```
sudo nano /etc/apache2/conf-available/CrayFits.conf

     Alias "/crayfits" "/opt/crayfish/CrayFits/public"
     <Directory "/opt/crayfish/CrayFits/public">
       FallbackResource /crayfits/index.php
       Require all granted
       DirectoryIndex index.php
       SetEnvIf Authorization "(.*)" HTTP_AUTHORIZATION=$1
     </Directory>

sudo chown -R root:root /etc/apache2/conf-available/CrayFits.conf

sudo chmod -R 644 /etc/apache2/conf-available/CrayFits.conf
```

Enabling Each Crayfish Component Apache Configuration¶

Enabling each of these configurations involves creating a symlink to them in the `conf-enabled` directory; the standardized method of doing this in Apache is with `a2enconf`.

```
sudo a2enconf Homarus Houdini Hypercube Milliner Recast
```

`CrayFits`

Restarting the Apache Service¶

Finally, to get these new endpoints up and running, we need to restart the Apache service.

```
sudo systemctl reload apache2
```

```
sudo systemctl restart apache2
```

Installing ActiveMQ and Alpaca¶

Karaf no longer needed

You no longer need to install Karaf. We no longer do this, we just deploy the java apps.

In this section, we will install:¶

- Apache ActiveMQ, a messaging server that will be used to handle communication between Alpaca and other components
- Islandora/Alpaca, Java middleware that handle communication between various components of Islandora.

Installing ActiveMQ¶

In our c the default installation method fo tiveMQ via apt-get will suffice.

```
sudo                    ivemq
```

/data

- /share/activemq
- A on boot
- of the ActiveMQ service

N the JMS setting in the alpaca

Wr ersion listed under Insta

These ActiveMQ installation instructions didn't work for me. The steps on the following page did.

Install ActiveMQ

Resource: https://docs.vultr.com/install-apache-activemq-on-ubuntu-20-04

```
cd /opt

sudo wget
http://archive.apache.org/dist/activemq/[ACTIVEMQ_VERSION_NUMBER]/ap
ache-activemq-[ACTIVEMQ_VERSION_NUMBER]-bin.tar.gz

sudo tar -xvzf apache-activemq-[ACTIVEMQ_VERSION_NUMBER]-bin.tar.gz

sudo mkdir /opt/activemq

sudo mv apache-activemq-[ACTIVEMQ_VERSION_NUMBER]/* /opt/activemq

sudo addgroup --quiet --system activemq

sudo adduser --quiet --system --ingroup activemq --no-create-home --
disabled-password activemq

sudo chown -R activemq:activemq /opt/activemq

sudo rm -R apache-activemq-[ACTIVEMQ_VERSION_NUMBER]-bin.tar.gz

sudo rm -R apache-activemq-[ACTIVEMQ_VERSION_NUMBER]

sudo nano /etc/systemd/system/activemq.service
      [Unit]
      Description=Apache ActiveMQ
      After=network.target

      [Service]
      Type=forking
      User=activemq
      Group=activemq

      ExecStart=/opt/activemq/bin/activemq start
      ExecStop=/opt/activemq/bin/activemq stop

      [Install]
      WantedBy=multi-user.target

sudo nano /opt/activemq/conf/jetty.xml
```

- Change: `<property name="host" value="127.0.0.1"/>`
- To: `<property name="host" value="0.0.0.0"/>`

```
sudo nano /opt/activemq/conf/users.properties
```

- Define ActiveMQ *`username=password`*

```
sudo systemctl daemon-reload
sudo systemctl start activemq
sudo systemctl enable activemq
sudo systemctl status activemq
sudo systemctl restart activemq
sudo apt-cache policy activemq   # note version number
```

Test: http://localhost:8161/

Update ActiveMQ Web Console credentials

Resource: https://activemq.apache.org/components/classic/documentation/getting-started#:~:text=The%20default%20username%20and%20password,properties%20file.

```
sudo nano /opt/activemq/conf/jetty-realm.properties
```

Replace the following credentials:

Before:

```
admin: admin, admin
user: user, user
```

After:

```
# admin: admin, admin
# user: user, user
[activemq_user]: [activemq_user_password], admin
```

```
sudo systemctl restart activemq
```

Test: http://localhost:8161/admin/

- Username: `[activemq_user]`
- Password: `[activemq_user_password]`

After ActiveMQ is installed and working, run the following command on your Drupal site:

```
cd /var/www/html/[DRUPAL_DIRECTORY_NAME]
composer exec -- drush migrate:import --userid=1 --tag=islandora
```

Installing Alpaca¶

Install Java 11+ if you haven't already.

Make a directory for Alpaca and download the latest version of Alpaca from the Maven repository. E.g.

```
mkdir /opt/alpaca
cd /opt/alpaca
sudo curl -L https://repo1.maven.org/maven2/ca/islandora/alpaca/islandora-alpaca-app/2.2.0/islandora-alpaca-app-2.2.0-all.jar -o alpaca.jar
```

Configuration¶

Alpaca is made up of several services, each of these can be enabled or disabled individually.

Alpaca takes an external file to configure its behaviour.

Look at the example.properties file to see some example settings.

The properties are… *(see next page)*

Entire configuration file (alpaca.properties):

```
sudo nano /opt/alpaca/alpaca.properties
        # Common options
        error.maxRedeliveries=5
        jms.brokerUrl=tcp://localhost:61616
        jms.username=[activemq_user]
        jms.password=[activemq_user_password]
        jms.connections=10

        # Custom Http client options
        # All timeouts in milliseconds
        request.configurer.enabled=false
        request.timeout=-1
        connection.timeout=-1
        socket.timeout=-1

        # Additional HTTP endpoint options, these can be for Camel or to be sent to the baseUrl or
        service.url
        http.additional_options=

        # Fedora indexer options
        fcrepo.indexer.enabled=true
        fcrepo.indexer.node=queue:islandora-indexing-fcrepo-content
        fcrepo.indexer.delete=queue:islandora-indexing-fcrepo-delete
        fcrepo.indexer.media=queue:islandora-indexing-fcrepo-media
        fcrepo.indexer.external=queue:islandora-indexing-fcrepo-file-external
        fcrepo.indexer.milliner.baseUrl=http://127.0.0.1/milliner/
        fcrepo.indexer.concurrent-consumers=-1
        fcrepo.indexer.max-concurrent-consumers=-1
        fcrepo.indexer.async-consumer=false

        # Triplestore indexer options
        triplestore.indexer.enabled=true
        triplestore.baseUrl=http://localhost:8080/blazegraph/namespace/islandora/sparql
        triplestore.index.stream=queue:islandora-indexing-triplestore-index
        triplestore.delete.stream=queue:islandora-indexing-triplestore-delete
        triplestore.indexer.concurrent-consumers=-1
        triplestore.indexer.max-concurrent-consumers=-1
        triplestore.indexer.async-consumer=false

        # Derivative services
        derivative.systems.installed=fits,homarus,houdini,ocr

        derivative.fits.enabled=true
        derivative.fits.in.stream=queue:islandora-connector-fits
        derivative.fits.service.url=http://localhost/crayfits
        derivative.fits.concurrent-consumers=-1
        derivative.fits.max-concurrent-consumers=-1
        derivative.fits.async-consumer=false

        derivative.homarus.enabled=true
        derivative.homarus.in.stream=queue:islandora-connector-homarus
        derivative.homarus.service.url=http://127.0.0.1/homarus/convert
        derivative.homarus.concurrent-consumers=-1
        derivative.homarus.max-concurrent-consumers=-1
        derivative.homarus.async-consumer=false

        derivative.houdini.enabled=true
        derivative.houdini.in.stream=queue:islandora-connector-houdini
        derivative.houdini.service.url=http://127.0.0.1/houdini/convert
        derivative.houdini.concurrent-consumers=-1
        derivative.houdini.max-concurrent-consumers=-1
        derivative.houdini.async-consumer=false

        derivative.ocr.enabled=true
        derivative.ocr.in.stream=queue:islandora-connector-ocr
        derivative.ocr.service.url=http://localhost/hypercube
        derivative.ocr.concurrent-consumers=-1
        derivative.ocr.max-concurrent-consumers=-1
        derivative.ocr.async-consumer=false
```

```
# Common options
error.maxRedeliveries= ~~4~~ 5
```

This defines how many times to retry a message before failing completely.

There are also common ActiveMQ properties to setup the connection.

```
# ActiveMQ options
jms.brokerUrl=tcp://localhost:61616
```

This defines the url to the ActiveMQ broker which you installed earlier.

```
jms.username= [activemq_user] (defined in /opt/activemq/conf/users.properties)
jms.password= [activemq_user_password]
```

This defines the login credentials (if required)

```
jms.connections=10
```

This defines the pool of connections to the ActiveMQ instance.

~~jms.concurrent-consumers=1~~

This defines how many messages to process simultaneously.

islandora-indexing-fcrepo¶

This service manages a Drupal node into a corresponding Fedora resource.

It's properties are:

```
# Fcrepo indexer options
fcrepo.indexer.enabled=true
```

This defines whether the Fedora indexer is enabled or not.

```
fcrepo.indexer.node=queue:islandora-indexing-fcrepo-content
fcrepo.indexer.delete=queue:islandora-indexing-fcrepo-delete
fcrepo.indexer.media=queue:islandora-indexing-fcrepo-media
fcrepo.indexer.external=queue:islandora-indexing-fcrepo-file-external
```

These define the various queues to listen on for the indexing/deletion messages. The part after `queue:` should match your Islandora instance "Actions".

```
fcrepo.indexer.milliner.baseUrl=http://localhost~~:8000~~/milliner   http://127.0.0.1/milliner/
```

This defines the location of your Milliner microservice.

```
fcrepo.indexer.concurrent-consumers= ~~1~~ -1
fcrepo.indexer.max-concurrent-consumers=~~1~~ -1
```

These define the default number of concurrent consumers and maximum number of concurrent consumers working off your ActiveMQ instance. A value of `-1` means no setting is applied.

```
fcrepo.indexer.async-consumer=~~true~~ false
```

This property allows the concurrent consumers to process concurrently; otherwise, the consumers will wait to the previous message has been processed before executing.

islandora-indexing-triplestore¶

This service indexes the Drupal node into the configured triplestore

It's properties are:

```
# Triplestore indexer options
triplestore.indexer.enabled=~~false~~ true
```

This defines whether the Triplestore indexer is enabled or not.

```
triplestore.index.stream=queue:islandora-indexing-triplestore-index
triplestore.delete.stream=queue:islandora-indexing-triplestore-delete
```

These define the various queues to listen on for the indexing/deletion messages. The part after `queue:` should match your Islandora instance "Actions".

```
triplestore.baseUrl=http://localhost~~:8080~~/bigdata/namespace/kb/sparql
http://localhost/bigdata/namespace/kb/sparql
```

This defines the location of your triplestore's SPARQL update endpoint.

```
triplestore.indexer.concurrent-consumers= ~~1~~ -1
triplestore.indexer.max-concurrent-consumers= ~~1~~ -1
```

These define the default number of concurrent consumers and maximum number of concurrent consumers working off your ActiveMQ instance. A value of `-1` means no setting is applied.

```
triplestore.indexer.async-consumer=~~true~~ false
```

This property allows the concurrent consumers to process concurrently; otherwise, the consumers will wait to the previous message has been processed before executing.

islandora-connector-derivative¶

This service is used to configure an external microservice. This service will deploy multiple copies of its routes with different configured inputs and outputs based on properties.

The routes to be configured are defined with the property derivative.systems.installed which expects a comma separated list. Each item in the list defines a new route and must also define 3 additional properties.

derivative.<item>.enabled=true

This defines if the item service is enabled.

derivative.<item>.in.stream=queue:islandora-item-connector.index

```
derivative.<item>.in.stream=queue:islandora-connector-<item>
```

This is the input queue for the derivative microservice. The part after queue: should match your Islandora instance "Actions".

derivative.<item>.service.url=http://example.org/derivative/convert

This is the microservice URL to process the request.

derivative.<item>.concurrent-consumers= ~~1~~ `-1`

derivative.<item>.max-concurrent-consumers= ~~1~~ `-1`

These define the default number of concurrent consumers and maximum number of concurrent consumers working off your ActiveMQ instance. A value of -1 means no setting is applied.

derivative.<item>.async-consumer=~~true~~ `false`

This property allows the concurrent consumers to process concurrently; otherwise, the consumers will wait to the previous message has been processed before executing.

For example, with two services defined (houdini and crayfits) my configuration would have

derivative.systems.installed=~~houdini,fits~~ `fits,homarus,houdini,ocr`

derivative.houdini.enabled=true
derivative.houdini.in.stream=queue:islandora-connector-houdini
derivative.houdini.service.url=http://127.0.0.1~~:8000~~/houdini/convert `http://127.0.0.1/houdini/convert`

derivative.houdini.concurrent-consumers=~~1~~ `-1`

derivative.houdini.max-concurrent-consumers=~~4~~ `-1`

derivative.houdini.async-consumer=~~true~~ `false`

derivative.fits.enabled=true
derivative.fits.in.stream=queue:islandora-connector-fits

derivative.fits.service.url=http://127.0.0.1~~:8000~~/crayfits `http://localhost/crayfits`

derivative.fits.concurrent-consumers=~~2~~ `-1`

derivative.fits.max-concurrent-consumers=~~2~~ `-1`

derivative.fits.async-consumer=false

Customizing HTTP client timeouts¶

You can alter the HTTP client from the defaults for its request, connection and socket timeouts. To do this you want to enable the request configurer.

request.configurer.enabled=~~true~~ `false`

Then set the next 3 timeouts (measured in milliseconds) to the desired timeout.

```
request.timeout=-1
connection.timeout=-1
socket.timeout=-1
```

The default for all three is `-1` which indicates no timeout.

Alter HTTP options¶

By default, Alpaca uses two settings for the HTTP component, these are * disableStreamCache=true * connectionClose=true

If you want to send additional configuration parameters or alter the existing defaults. You can add them as a comma separated list of key=value pairs.

For example

```
http.additional_options=authMethod=Basic,authUsername=Jim,authPassword=1234
```

These will be added to ALL http endpoint requests.

Note: We are currently running Camel 3.7.6, some configuration parameters on the above linked page might not be supported.

Deploying/Running¶

You can see the options by passing the `-h|--help` flag

```
> java -jar /opt/alpaca/alpaca.jar -h
Usage: alpaca [-hV] [-c=<configurationFilePath>]
  -h, --help      Show this help message and exit.
  -V, --version   Print version information and exit.
  -c, --config=<configurationFilePath>
                  The path to the configuration file
```

Using the -V|--version flag will just return the current version of the application.

```
> java -jar /opt/alpaca/alpaca.jar -v
2.0.0
```

To start Alpaca you would pass the external property file with the -c|--config flag.

For example if you are using an external properties file located at /opt/alpaca/alpaca.properties, you would run:

```
java -jar alpaca.jar -c /opt/alpaca/alpaca.properties
```

To use systemd to start and stop the service create the file /etc/systemd/system/alpaca.service:

```
[Unit]
Description=Alpaca service
After=network.target

[Service]
Type=forking
ExecStart=java -jar /opt/alpaca/alpaca.jar -c /opt/alpaca/alpaca.properties
ExecStop=/bin/kill -15 $MAINPID
SuccessExitStatus=143
Restart=always

[Install]
WantedBy=default.target
```

Now you can start the service by running systemctl start alpaca and set it to come up when the system reboots with systemctl enable alpaca.

```
sudo systemctl start alpaca

sudo systemctl enable alpaca

sudo systemctl status alpaca

sudo systemctl restart alpaca
```