Building DuraCloud from Source

A more current version of this guide is now available. If you are deploying DuraCloud for development, see this page: https://github.com/duracloud/deployment-docs/blob/master/dev-guide-getting-started.md. If you are deploying DuraCloud in production, see the pages here: https://github.com/duracloud/deployment-docs. The following document should be considered deprecated and will be removed in the near future.

Introduction

If you would prefer to install DuraCloud from a binary distribution, you can find instructions for that process here.

DuraCloud application software is composed of many parts. A breakdown of the primary pieces is as follows:

- DuraStore - this web application provides the access to and management of storage resources, which includes handling the storage portion of the DuraCloud REST API
- StorageProviders - this set is made up of the StorageProvider interfaces and the implementations which connect to distinct cloud stores; the storage providers which are currently supported are listed here.
- DurAdmin - this web application is the UI front end of DuraCloud, it provides users with a view into the information available from the other applications. DurAdmin uses the REST APIs of the other applications to communicate with them.
- Security - handles security for the DuraCloud applications
- Common - a set of projects which provide utilities for other portions of the codebase to reuse

The DuraCloud software, by its very nature, is designed to be integrated with underlying cloud storage providers. As may be expected, these integrations are necessary for the system to be properly exercised. In order for DuraCloud to connect to these underlying providers, appropriate credentials must be provided as part of the application configuration step. It is recommended that you acquire the necessary storage provider credentials prior to attempting to set up DuraCloud. Only one storage provider is required to run DuraCloud.

This guide lays out the steps necessary to begin using DuraCloud:

1. Configure DuraCloud database
2. Build and deploy the DuraCloud web applications
3. Test your installation

Although this document is written from a Linux environment perspective, analogous builds/installations have been tested in Windows (but may have limitations, as noted below). Any comments or feedback are welcomed.

Prerequisites

Software that must be installed on your system prior to building/using DuraCloud

1. Java: Version 11 required
    a. OpenJDK is recommended for building DuraCloud, as this is the JDK used for DuraCloud testing.
2. Maven 3.x
3. Tomcat 8.5+
4. Git
5. DuraCloud Management Console
6. DuraCloud Mill

Setting up DuraCloud

Any portions of the configuration below for which you need to include a replacement value will be written in all capital letters and included in brackets: [LIKE-THIS]

Build and deploy the DuraCloud web applications

1. Check out latest stable release from GitHub repository. Determine the release number of the latest stable release by going to https://github.com/duracloud/duracloud/releases and making a note of the tag name of the most current release.

```
git clone https://github.com/duracloud/duracloud.git
cd duracloud
git checkout <latest-release-tag>
```

2. Set environment variables

```
export JAVA_OPTS="-XX:MaxPermSize=256m"
export MAVEN_OPTS="-Xmx1024m
```

3. Configure Tomcat
    a. Add to $CATALINA_HOME/conf/tomcat-users.xml

```
<tomcat-users>
  <role rolename="manager-gui"/>
  <role rolename="manager-script"/>
  <role rolename="admin"/>
  <user username="[ANY-USERNAME]" password="[ANY-PASSWORD]" roles="admin,manager-gui,manager-
script"/>
</tomcat-users>
```

    b. Add to $CATALINA_HOME/conf/server.xml

    Add the config attribute "URIEncoding" with value "UTF-8" to your Tomcat Connector. Your connector may look like the following:

```
<Connector port="8080" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443"
               URIEncoding="UTF-8" />
```

4. Configure DuraCloud
    a. Create a duracloud properties file (/any/path/to/duracloud.properties) that contains the following keys:

```
mill.db.name=<mill db name>
mill.db.port=<mysql port>
mill.db.host=<mysql host>
mill.db.user=<username>
mill.db.pass=<password>

db.name=<ama db name>
db.host=<mysql ama host>
db.port=<mysql ama port>
db.user=<username>
db.pass=<password>
```

    b. Add a system property (duracloud.config.file) to point to your duracloud configuration file.
        i. This can be done by adding the property directly to your JAVA_OPTS or CATALINA_OPTS environment variables. You can also set these values in $CATALINA_HOME/bin/setenv.sh (linux,osx) or $CATALINA_HOME/bin/setenv.bat (windows), for example:

```
JAVA_OPTS="${JAVA_OPTS} -Dduracloud.config.file=file:c:/duracloud/duracloud-config.
properties"
```

        ii. The duracloud.config.file system property can also refer to an Amazon S3 address using the s3://<bucket>/<path to file> syntax provided your tomcat instance is running on an instance with the appropriate AWS credentials. More information on AWS credentials management.
5. Start Tomcat

```
$CATALINA_HOME/bin/startup.sh
```

6. Configure Maven
    a. Add tomcat user to $M2_HOME/conf/settings.xml. Make sure that the username and password used here match those included in the tomcat-users.xml file.

```
<servers>
  <server>
    <id>tomcat-server</id>
    <username>[ANY-USERNAME]</username>
    <password>[ANY-PASSWORD]</password>
  </server>
</servers>
```

7. Build
    a. From top of the source tree, execute (note that this build runs unit tests, but not integration tests):

```
mvn clean install -DskipIntTests
```

8. Bonus, running Integration Tests
    a. Grab a copy of the file in the codebase under common-json/src/main/resources/test-config.json and place it somewhere on your system
    b. You'll need an account for each of the providers to be tested. Update the JSON file to include your credentials. These tests actually communicate with each storage provider and verify that the calls being made work properly
    c. Add an environment variable called DURACLOUD-TEST-CONFIG with the value being the full path to your updated credentials JSON file. A sample credential configuration file can be found in the baseline at <project root>/integration/src/test/resources/test-config.json
    d. From the top of the source tree, execute

```
mvn clean install -pl integration
```

Test your installation

1. Once all of the above steps have been completed, your DuraCloud should be ready to test.
    a. Go to http://localhost:8080/duradmin (change host or port as necessary).
    b. Log in using valid DuraCloud Management Console credentials.
    c. You should be able to view, add, update, and delete spaces and content in Spaces tab
2. Congratulations! You now have a functional DuraCloud installation.

Optional items

SyncTool installers

1. The installers for the Sync Tool (Windows, Mac, Linux) are built using BitRock InstallBuilder Enterprise, so this tool must be installed locally.
    a. Download and install BitRock InstallBuilder Enterprise version 15.1.0 or above
    b. Place the license file for InstallBuilder in the InstallBuilder installation directory
2. Update the maven settngs.xml file ($M2_HOME/conf/settings.xml) to include the path to the installbuilder executable
    a. Add a duracloud profile to include the InstallBuilder path, and include the duracloud profile in the set of active profiles

```
<profiles>
  <profile>
    <id>duracloud</id>
    <properties>
      <installbuilder.executable.path>[FULL-PATH-TO-BITROCK-INSTALLBUILDER-EXECUTABLE]<
/installbuilder.executable.path>
    </properties>
  </profile>
</profiles>

...

<activeProfiles>
  <activeProfile>duracloud</activeProfile>
</activeProfiles>
```

    b. From the //synctoolui directory in the DuraCloud source tree, execute

```
mvn clean install -Pinstallers
```

c. The installers can be found under the synctoolui/target directory

Logging

1. DuraCloud uses the SLF4J logging framework backed by the LogBack implementation
2. By adding either a logback.xml or logback-test.xml file on the classpath, logging configuration can be customized

DuraCloud internal tools

Building Java client packages

1. To create a distributable zip of the storeclient or reportclient which includes their dependencies, from within the project directory (//storeclient, //reportclient) run

```
mvn install -Ppackage-client
```

2. The packaged zip will be found under the project's target directory