BBRB Biorepositories and Biospecimen Research Branch

leidos
Leidos Biomedical Research, Inc.

IT-9015 Building and Deploying CDR-Lite

Version 1.01

11/7/2016

Version History

Version	Author	Effective Date	Description of Change
1.00	William Lander	11/4/2016	Initial Version
1.01	William Lander	11/7/2016	Administrative Update

Table of Contents

1 Introduction

The National Cancer Institute's (NCI's) Comprehensive Data Resource Lite (CDR-Lite) is a Web-based application, custom built to support specimen collection, clinical data entry, specimen logistics, and curation and aggregation of study data. CDR-Lite is based on an earlier effort, CDR which is a distributed web-based system that manages and maintains multidimensional data models of biospecimens. The CDR collects both biospecimen annotation and clinical data associated with those biospecimens from cancer patient donors and post-mortem donors for the NCI Biospecimen Pre-Analytical Variables Program and the National Institutes of Health Genotype-Tissue Expression program.

Given the success of the CDR, it became apparent that a comprehensive data resource would be beneficial for many general applications. For that reason, tasking for generalizing and simplifying the code went to the development team. This work resulted in the CDR-Lite. As part of the simplifying process, there is no more reliance on commercial products; the underlying database for CDR-Lite changed from Oracle to Postgres.

Both CDR and CDR-Lite provide secure data access based on a user's roles and privileges. Through dynamic content redaction, it protects private information in compliance with Health Insurance Portability and Accountability Act (HIPAA) regulations. Their graphic user interfaces use standard operating procedures (SOPs) for sample collection and processing, streamlining data entry. This includes automated data checks and validations that simultaneously confirm data integrity and SOP adherence.

Visit http://biospecimens.cancer.gov/newsevents/news/07172015.asp for more information.

Both CDR and CDR-Lite, are Grails applications. Follow the steps below in getting this standalone version of the CDR-Lite up and running. Knowledge of Groovy, Java, and Git is assumed.

2 Preliminary Steps

The CDR-Lite was developed using the NetBeans IDE and uses PostgreSQL 9.3. A knowledge of NetBeans, Java, Git, and basic Unix commands is assumed.

1. Install the latest version of JDK 7 per instructions for your environment. JDK 8 is not supported.

2. Install PostgreSQL 9.3.

3. In PostgreSQL, create a database called `cdr`. Create a user and password. By default, the CDR-Lite will try to connect to the database as "cdr/admin." The `cdr` user account will not need any special privileges. Through hibernate, it will create tables and sequences only. If other than the default user name and password are created, the alternative username and password must synchronize with the database, user, and password you create. The file `DataSource.groovy`, which is in the `conf` folder under `grails-app`, should contain the user and password information.

4. Install Grails 2.4.4 per instructions for your environment. **Make sure to use this version;** later versions of Grails are not supported, and earlier versions may not be compatible. Visit http://grails.org/doc/latest/guide/gettingStarted.html for more Grails installation information.

5. Install the Git client.

6. Obtain the source code for CDR-Lite by cloning from GitHub: https://github.com/NCIP/CDR-Lite. After cloning from GitHub, open a console window, navigate to the `CDR-Lite` folder, and run the following command:

   ```
   set JAVA_OPTS=-XX:MaxPermSize=128m -XX:PermSize=512m -Xms1024m -Xmx2048m
   ```

   (This is a generous estimate. Default memory settings are not enough.)

7. In the local environment, open the file cdrlite/grails-app/conf/DataSource.groovy. Update the entries at

   ```
   environments {
       development {
           dataSource {
                 username = "Postgres SQL UserName"
                 password = "Postgres SQL Password"
                 …
   ```

   Postgres SQL UserName and Postgres SQL Password must match the username and password you created in PostgreSQL.

8. Run the command

   ```
   grails RunApp
   ```

   This downloads all project plug-ins, resolve dependencies, and determine whether your environment is set up properly. If all goes well, you should see the following message:

   ```
   Server running. Browse to http://localhost:8080/cdrlite
   ```

   Open a browser of your choice and enter http://localhost:8080/cdrlite

9. Log in with the following test account:

   ```
   admin/admin
   ```

If you run into errors, try the following:

a. `grails clean`

b. `grails upgrade`

   (Answer 'yes' to any questions.)

c. `grails RunApp`

3 Build the WAR File

Note: Make sure Tomcat is not running within the NetBeans IDE before you start.

The **wars** batch script is in the main project folder. Run it from the command line prompt. The script executes a Grails command and builds the WAR file for a requested environment, which could be any one of the following: Production, Test, QA, DEV. Depending on the script options, the WAR file can be built for **all environments** or for **any single environment.** The environment definitions are in `cdrlite/conf/DataSource.groovy`.

3.1 Run the Script

At the command line prompt, change the directory to the project folder:

`C:\dev\svn\cdrlite>wars test` (builds WAR file for the test environment)

`C:\dev\svn\cdrlite>wars` (builds multiple WAR files for every environment)

If you see an error message reading "applicationContext.xml not found," run `grails upgrade`.

Please note: The above command may take a long time to complete (average time is about 3–7 minutes per environment).

3.2 Naming Convention for the WAR File

The WAR file builds with the environment name and the current version appended to its name. The WAR file(s) built from the above step can be found in the `target` directory within the main project folder. The version number is read from the `application.properties` file found in the main project directory.

The WAR file in this example is built as version 3.5 and targeted for the ec2cdr environment:

`cdrlite.war.ec2cdr-3.5`

4 Update the WAR File on the Server

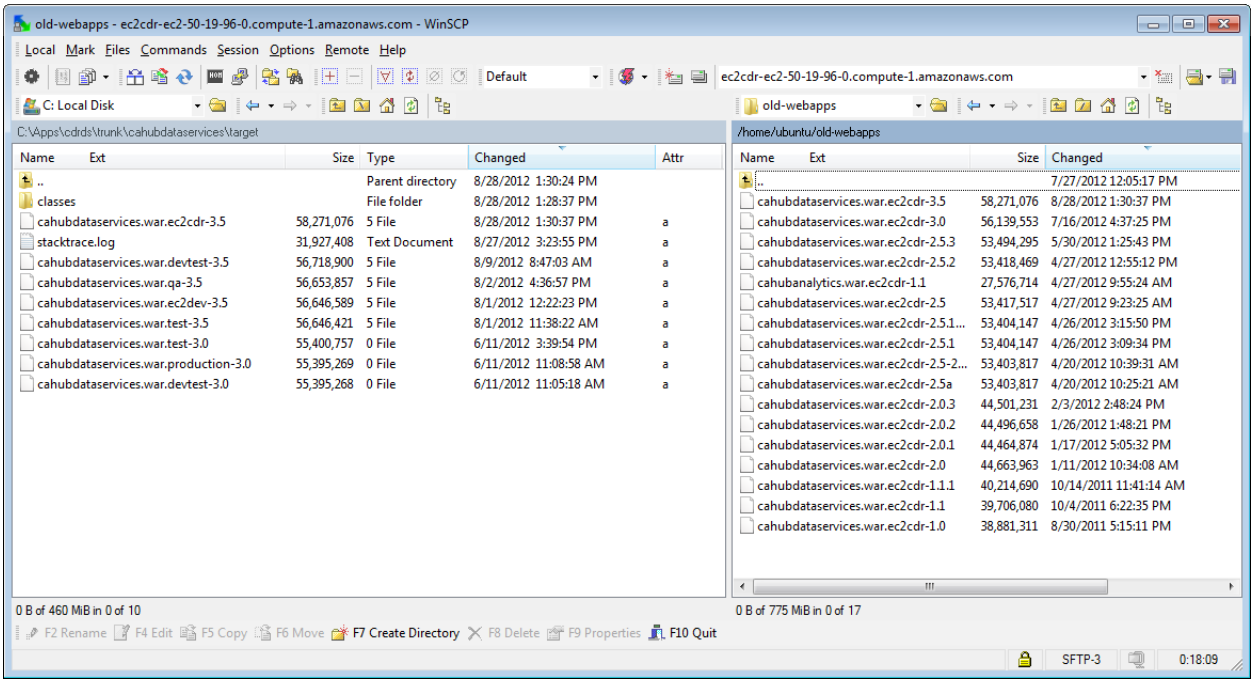
Using WinSCP, remote copy the WAR file on the server for which it was built.[1]

Example: In the case of deploying on the Test environment, log in as "jenkins" and copy the WAR file from the target directory to the `/var/storage/old-webapps` directory.

[1] This manual process has been replaced by an automated deployment tool, Jenkins, but the manual process will work, and documentation of the deployment tool is outside the scope of this document.

The `Config.groovy` and `DataSource.groovy` files for the correct environment will be embedded in the WAR file.

Table 1- Example WebApps Directory



4.1 Update the Application on the Server with the Latest Version

Using PuTTY, log in via ssh to the server for which the WAR file was built.

Open two sessions of PuTTY so that you can view the logs on one while you stop and restart the Tomcat server on the other.

On the first session (A), become root (`sudo su -`) and, upon logging in, change directory to `/var/log/tomcat7`. Tail the log file: `tail -200f catalina.out`.

On the second session (B), become root (`sudo su -`) and change directory to `/var/lib/tomcat7/webapps`. Set your session to root by typing `sudo su –` and entering the appropriate password. You need to be root to stop and start Tomcat.

At this point, alert anyone who may be impacted that the server is being stopped while you deploy a new version.

Run the following commands on session B:

1. Make a copy of the current WAR file for fallback and recovery purposes:

```
/var/lib/tomcat7/webapps$ cp –p cdrlite.war
cdrlite.war.backup.PEH.28-Aug-2012
```

2. Copy the WAR file from the {upload} directory (in this case, `/home/ubuntu/old-webapps`) to the Tomcat webapps directory. For example:

```
/var/lib/tomcat7/webapps$ cp –p /home/ubuntu/old-
webapps/cdrlite.war.ec2cdr-3.5
```

3. Check whether Tomcat is running:

```
ps –aef | grep java
```

You should see something like this:

```
tomcat7   3059     1  0 Jul08 ?        00:10:23
/usr/lib/jvm/java-7-oracle/bin/java -
Djava.util.logging.config.file=/var/lib/tomcat7/conf/logging.prop
erties -Xms2048m -Xmx2048m -XX:PermSize=512m -XX:MaxPermSize=512m
-
Djavax.sql.DataSource.Factory=org.apache.commons.dbcp.BasicDataSo
urceFactory -
Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager
-Djava.endorsed.dirs=/usr/share/tomcat7/endorsed -classpath
/usr/share/tomcat7/bin/bootstrap.jar:/usr/share/tomcat7/bin/tomca
t-juli.jar -Dcatalina.base=/var/lib/tomcat7 -
Dcatalina.home=/usr/share/tomcat7 -Djava.io.tmpdir=/tmp/tomcat7-
tomcat7-tmp org.apache.catalina.startup.Bootstrap start
```

4. Stop Tomcat:

```
/etc/init.d/tomcat7 stop
```

At this point, view the logs scrolling on the screen in session A.

5. Remove the old WAR file and the (exploded) project dir:

```
/var/lib/tomcat7/webapps$ rm –rf cdrlite.war

/var/lib/tomcat7/webapps$ rm –rf cdrlite
```

6. Copy the uploaded WAR file from its environment and version-specific name to something Tomcat will recognize and deploy upon startup:

```
/var/lib/tomcat7/webapps$ cp –p cdrlite.war.ec2cdr-3.5
cdrlite.war
```

7. Start Tomcat again:

```
/etc/init.d/tomcat7 start
```

This will deploy the WAR file and re-create the project directory in `tomcat7/webapps/`.

Watch the scrolling logfile in session A and the output from session B. Scan for any stack traces in the logs, and keep an eye on the Java process in top until the application server has completed startup. You should see the following message at the bottom of the log file in session A:

```
Configuring Spring Security ...
Configuring Spring Security LDAP ...
Configuring Spring Security UI ...
Aug 28, 2012 5:58:52 PM org.apache.coyote.http11.Http11Protocol start
INFO: Starting Coyote HTTP/1.1 on http-8080
Aug 28, 2012 5:58:52 PM org.apache.jk.common.ChannelSocket init
INFO: JK: ajp13 listening on /0.0.0.0:8009
Aug 28, 2012 5:58:52 PM org.apache.jk.server.JkMain start
INFO: Jk running ID=0 time=0/21  config=null
Aug 28, 2012 5:58:52 PM org.apache.catalina.startup.Catalina start
INFO: Server startup in 136410 ms
```

4.2 Check that Application Is Running and Changes Have Been Made

Open a browser and navigate to the application URL on the server where the WAR file has been deployed. Check that the application is running as expected and that all the changes for this version are present.

Note: The NCI FNLRC environment has an Apache proxy, a redirect, and an SSL certificate in place, so that http://cdrlite.ncifcrf.gov automatically takes the user to https://cdrlite.ncifcrf.gov/cdrlite. The scope of this document does not cover server configuration and administration, but you may wish to consider something similar in your installation environment. Such configuration is highly dependent on the environment where the application will be running.