LIFERAY. Enterprise. Open Source. For Life.

JBoss EAP Domain Mode

Deploying Liferay Portal v6.2 to JBoss EAP 6.1.0 in Domain mode

Version: 1.1 (jun 2013)

Author: igor.spasic@liferay.com

JBoss Application Server EAP (and 7) can be booted in two different modes. A *managed domain* allows you to run and manage a multi-server topology. Alternatively, you can run a *standalone* server instance.

If more than one standalone instance is launched and multi-server management is desired, it is the user's responsibility to coordinate management across the servers. For example, to deploy an application across all of the standalone servers, the user would need to individually deploy the application on each server. It is perfectly possible to launch multiple standalone server instances and have them form an HA cluster.

One of the primary new features of JBoss Application Server is the ability to manage multiple JBoss Application Server instances from a single control point. A collection of such servers is referred to as the members of a “domain” with a single Domain Controller process acting as the central management control point. All of the JBoss Application Server instances in the domain share a common management policy, with the Domain Controller acting to ensure that each server is configured according to that policy. The Host Controller on each host interacts with the Domain Controller to control the lifecycle of the application server instances running on its host and to assist the Domain Controller in managing them.

Deploying Liferay

In the time of writing this document, the latest community version of JBoss AS7 is **v7.1.1** (http://www.jboss.org/jbossas/downloads/). Unfortunately, this version of JBoss has some bugs that makes deployment impossible – some JBoss exceptions are thrown on WAR scanning before actual deployment.

For that reason, this document describes deployment procedure on JBoss **EAP 6.1.0**. However, this version is available only through Red Hat as **Application Platform 6.1.0**. Alternatively, JBoss can be built from its GitHub source.

Here are the steps how to deploy Liferay war.

JBoss Modules

This step is identical to *standalone* configuration. Create the folder: **$JBOSS_HOME/modules/com/liferay/portal/main** and add portal dependencies:

- hsql.jar, postresql.jar or mysql.jar (i.e. database driver)
- jtds.jar
- portal-service.jar
- portlet.jar

Also, add file **module.xml** with the following content:

```
<?xml version="1.0"?>
<module xmlns="urn:jboss:module:1.0" name="com.liferay.portal">
  <resources>
        <resource-root path="hsql.jar" />
        <resource-root path="jtds.jar" />
        <resource-root path="mysql.jar" />
        <resource-root path="portal-service.jar" />
        <resource-root path="portlet.jar" />
        <resource-root path="postgresql.jar" />
  </resources>
  <dependencies>
        <module name="javax.api" />
        <module name="javax.mail.api" />
        <module name="javax.servlet.api" />
        <module name="javax.servlet.jsp.api" />
        <module name="javax.transaction.api" />
  </dependencies>
</module>
```

JDK module

Edit file: **$JBOSS_HOME/modules/system/layers/base/sun/jdk/main/module.xml** and add following lines on appropriate place:

```
<path name="com/sun/crypto"/>
<path name="com/sun/crypto/provider"/>
```

Portal Authentication

Edit **$JBOSS_HOME/domain/configuration/domain.xml** and add following content to <security-domains /> container within the **jboss:domain:security:1.2** subsystem:

```
<security-domain name="PortalRealm">
    <authentication>
        <login-module code="com.liferay.portal.security.jaas.PortalLoginModule"
flag="required"/>
    </authentication>
</security-domain>
```

There are several (4) places in the file where to add this; you can add it everywhere (to be safe :) or just to the subsystems that belongs to "**default**" profile.

JBoss Web System

We need to modify the JBoss web subsystem to allow Liferay Portal to serve as the root context for the server.

Edit file **$JBOSS_HOME/domain/configuration/domain.xml** and locate the **jboss:domain:web:1.4** subsystems. Modify its virtual server definition to disable welcome root, like this:

```
<virtual-server name="default-host" enable-welcome-root="false">
    <alias name="localhost" />
    <alias name="example.com" />
</virtual-server>
```

(Also put correct alias names). Again, this should be changed on 4 places in the configuration file.

JVM configuration

Edit **$JBOSS_HOME/bin/domain.conf** and add the following line to the end of the file:

```
JAVA_OPTS="$JAVA_OPTS -Dfile.encoding=UTF-8 -Djava.net.preferIPv4Stack=true -
Duser.timezone=GMT"
```

Open the **$JBOSS_HOME/domain/configuration/domain.xml** file and locate the **<server-groups />** container. Comment or remove the default server-group entries. Add the following content to define a server group for portal instance(s):

```
<server-group name="liferay-server-group" profile="default">
    <jvm name="default">
        <heap size="1024m" max-size="2048m"/>
        <permgen size="384m" max-size="512m" />
    </jvm>
    <socket-binding-group ref="full-sockets[1]"/>
</server-group>
```

Open the **$JBOSS_HOME/domain/configuration/host.xml** file and locate the **<servers />** container. Comment or remove all default server entries. Add the following to define a server for Liferay instance:

```
<server name="liferay-01" group="liferay-server-group">
    <jvm name="default" />
</server>
```

Deploying portal war

Deploying can be done using command line interface (CLI) or JBoss Administration console. Either way, start JBoss in domain mode by executing the following command from the **$JBOSS_HOME/bin** folder:

```
./domain.sh
```

CLI

Start the JBoss administration CLI by issuing the following command from the **$JBOSS_HOME/bin** folder:

```
./jboss-cli --connect
```

Deploy portal war by issuing the following command:

```
deploy /…/portal/dist/liferay-portal-6.2.0.war --server-groups=liferay-server-group
```

This will start deployment on all servers in Liferay server group.

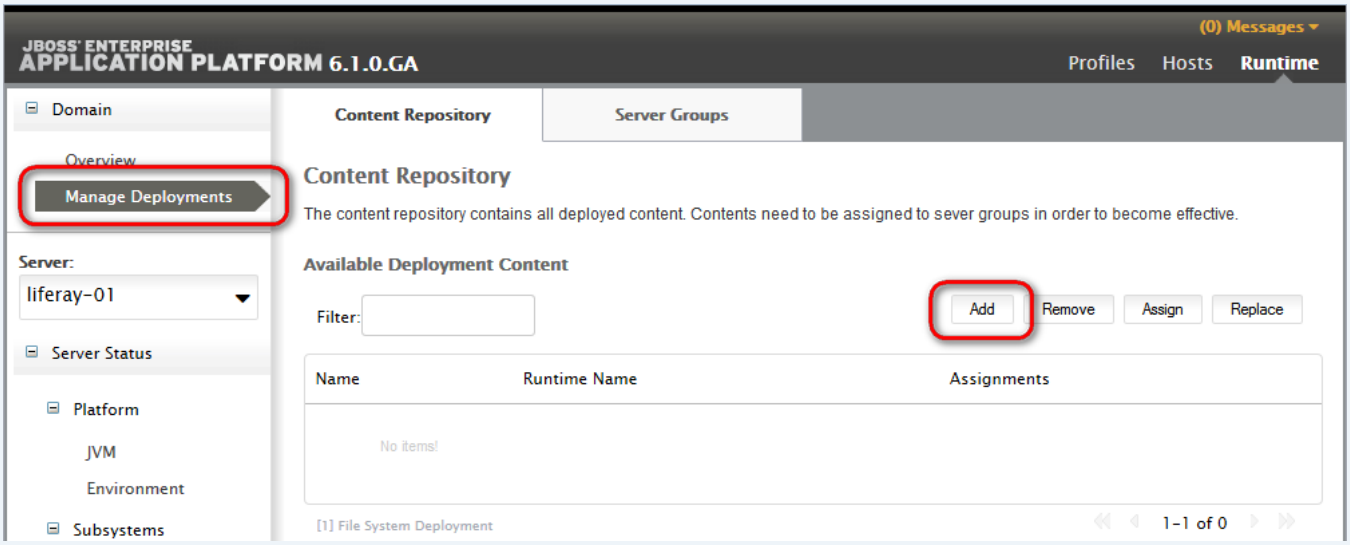
[1] The "full-sockets" socket binding group should be used for a regular standalone Liferay instance. To set up a JBoss cluster, you should use the "full-ha-sockets" socket binding group – and some additional configuration.

Administration Console

First we need to create a management user. Go to **$JBOSS_HOME/bin** folder and invoke **add-user**. Choose to add *Management User* of *ManagementRealm*.

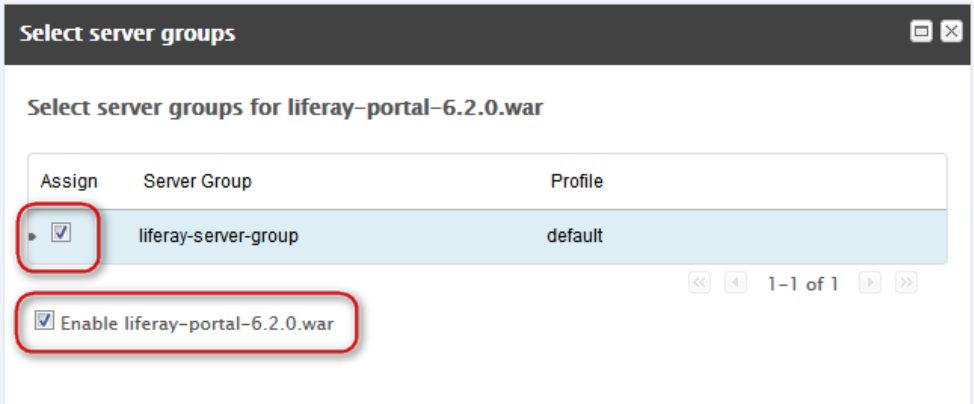
Go to http://localhost:9990/console in your browser.

Click on '*Manage Deployments*', then '*Add*' and select portals war.



Click '*Next*' and then '*Save*'. The content will be added. WAR name appears in the selection list above.

Now click on '*Assign*' to assign it to the server group. Choose liferay server group, and leave '*Enable…*' checkbox checked, so the deployment starts right away:



Clicking on '*Save*' will start the deployment process.

One of the benefits of the UI console is being able to see deployment errors easily: they appear in the 'Messages' box, top-right corner.

Log is located here: **$JBOSS_HOME/domain/servers/liferay-01/log/server.log**.

After the deployment, open: http://localhost:8080 to verify if the portal is deployed.

Standalone vs Domain mode

- In domain mode there is no Deployment Scanner that monitors a directory for new files and deploy those files. All deployments are only possible in CLI or Console.
  This means that portal hot-deployment feature is not available.
- WAR file in Domain mode is NOT extracted on deployment and it is located under **domain/data/content** folder and long hash as a single file.
  This means that all kind of Liferay patching and modification is not working.
  This also means that none of JSON WebServices will be registered, as no jar is scanned.

The solution for hot deploy is to have such folder and to mirror the server logic; using the CLI.

The solution for patching is to prepare standalone bundle, apply patches and then zip it and deploy everything as one WAR.

The solution for JSON WebServices is not trivial – idea is to drop jar scanning and add some kind of spring hook on bean registration (some containers provides this).

Choosing mode

It is very important to emphasize the following:

It's important to understand that the choice between a managed domain and standalone servers is all about how your servers are managed, not what capabilities they have to service end user requests. This distinction is particularly important when it comes to high availability clusters. It's important to understand that HA functionality is orthogonal to running standalone servers or a managed domain. That is, a group of standalone servers can be configured to form an HA cluster. The domain and standalone modes determine how the servers are managed, not what capabilities they provide.

Therefore:

A single server installation gains nothing from running in a managed domain, so running a standalone server is a better choice.

For multi-server production environments, the choice of running a managed domain versus standalone servers comes down to whether the user wants to use the centralized management capabilities a managed domain provides.

Running a standalone server is better suited for most development scenarios. Any individual server configuration that can be achieved in a managed domain can also be achieved in a standalone server, so even if the application being developed will eventually run in production on a managed domain installation, much (probably most) development can be done using a standalone server.

Running a managed domain mode can be helpful in some advanced development scenarios; i.e. those involving interaction between multiple JBoss instances. Developers may find that setting up various servers as members of a domain is an efficient way to launch a multi-server cluster.