JBoss Remote Deployer

As of CARGO 1.0.3, the way CARGO supports remote deployments on the JBoss Application Server has drastically evolved. This document explains how to configure this support.

You can use the quick links to directly fo to the chapter regarding the JBoss version you're targeting:

- JBoss 4.0.x and 4.2.x
- JBoss 5.0.x, 5.1.x, 6.0.x and 6.1.x
- JBoss 7.0.x and 7.1.x

JBoss 4.0.x and 4.2.x

In JBoss 4.x, the JMX-based remote deployer can accept URLs that point to another machine. For example:

> *`http://10.156.216.147:8080/jmx-console/HtmlAdaptor?action=invokeOpByName&name=jboss.system:service%3DMainDeployer&methodName=deploy&argType=java.net.URL&arg0=http%3A%2F%2F10.156.220.90%3A18080%2Fparuemas-1.0-SNAPSHOT.war`*

In this case, our JBoss server on `10.156.216.127` will connect to `10.156.220.90` via HTTP and download the `paruemas-1.0-SNAPSHOT.war` file. Once the file is downloaded, JBoss will automatically deploy it.

The CARGO JBoss container uses this principle to expose your Java EE application to JBoss using an HTTP server that is started on the machine where CARGO is currently running. That HTTP server is controlled using two parameters:

- **`JBossPropertySet.REMOTEDEPLOY_HOSTNAME`** (i.e., `cargo.jboss.remotedeploy.hostname`): sets the hostname that the JBoss server will attempt to connect to. By default, the CARGO JBoss container will automatically resolve the current machine's network name or IP address and fill this accordingly.
- **`JBossPropertySet.REMOTEDEPLOY_PORT`** (i.e., `cargo.jboss.remotedeploy.port`): sets the port number on which the HTTP server will be started on the machine running CARGO. By default, that one is set to *1 + the HTTP port for JBoss*. For example, if JBoss is running on HTTP port `8080`, then the default for `JBossPropertySet.REMOTEDEPLOY_PORT` will be `18080`.

The obvious limitation you should be aware of is that JBoss needs to have direct access to the machine running CARGO.

JBoss 5.0.x, 5.1.x, 6.0.x and 6.1.x

Starting from JBoss 5.x, JBoss has a Deployment Manager that can be used for deploying things on the JBoss server. If you really want to know how it works under the hood, you can read about this feature on the JBoss Website. If you only want to remotely deploy applications using CARGO to your JBoss server, you can of course ignore that document.

To connect to the JBoss Deployment Manager, CARGO uses JBoss' JMX RMI port; and that's what makes things get a bit complicated. For the connection to succeed, the following JARs need to be in the container classpath or in the current Java Thread's context classloader:

- JBoss deployment manager JARs
- JBoss remoting client JARs

Here is an example code for the users of the Java API:

```
List<URL> urls = new ArrayList<URL>();

// Add many libraries from JBOSS_HOME
for (File jar : new File(this.localContainer.getHome(),
"lib").listFiles())
{
    if (jar.isFile())
    {
        urls.add(jar.toURI().toURL());
    }
}
for (File jar : new File(this.localContainer.getHome(),
"common/lib").listFiles())
{
    if (jar.isFile())
    {
        urls.add(jar.toURI().toURL());
    }
}

// Create a ClassLoader contaning all these JARs
URL[] urlsArray = new URL[urls.size()];
urlsArray = urls.toArray(urlsArray);
URLClassLoader classLoader = new URLClassLoader(urlsArray,
this.getClass().getClassLoader());
Thread.currentThread().setContextClassLoader(classLoader);

// Now, create the JBoss Remote container
...
```

Here is an example Maven2 plugin configuration:

```
<!--
  Some transitive dependencies of JBoss artifacts, for example
apache-xerces:xml-apis, are
  only available on the JBoss third party repository.
  -->
<pluginRepositories>
  <pluginRepository>
    <id>repository.jboss.org_thirdparty-releases</id>
    <name>JBoss.org third party releases repository</name>

<url>https://repository.jboss.org/nexus/content/repositories/thirdpa
rty-releases</url>
    <releases>
      <enabled>true</enabled>
```

```
    </releases>
    <snapshots>
      <enabled>false</enabled>
    </snapshots>
  </pluginRepository>
  <pluginRepository>
    <id>repository.jboss.org_thirdparty-uploads</id>
    <name>JBoss.org third party uploads repository</name>

<url>https://repository.jboss.org/nexus/content/repositories/thirdpa
rty-uploads</url>
    <releases>
      <enabled>true</enabled>
    </releases>
    <snapshots>
      <enabled>false</enabled>
    </snapshots>
  </pluginRepository>
</pluginRepositories>

<repositories>
  <repository>
    <id>repository.jboss.org_thirdparty-releases</id>
    <name>JBoss.org third party releases repository</name>

<url>https://repository.jboss.org/nexus/content/repositories/thirdpa
rty-releases</url>
    <releases>
      <enabled>true</enabled>
    </releases>
    <snapshots>
      <enabled>false</enabled>
    </snapshots>
  </repository>
  <repository>
    <id>repository.jboss.org_thirdparty-uploads</id>
    <name>JBoss.org third party uploads repository</name>

<url>https://repository.jboss.org/nexus/content/repositories/thirdpa
rty-uploads</url>
    <releases>
      <enabled>true</enabled>
    </releases>
    <snapshots>
      <enabled>false</enabled>
    </snapshots>
  </repository>
</repositories>

 ...

<plugin>
  <groupId>org.codehaus.cargo</groupId>
  <artifactId>cargo-maven2-plugin</artifactId>
  <version>${cargo.plugin.version}</version>
  <configuration>
```

```
    <container>
      <containerId>jboss51x</containerId>
      <type>remote</type>
    </container>
    <configuration>
      <type>runtime</type>
      <properties>
        <cargo.hostname>production27</cargo.hostname>
        <cargo.rmi.port>11099</cargo.rmi.port>
      </properties>
    </configuration>
  </configuration>
  <dependencies>
    <dependency>
      <groupId>org.jboss.integration</groupId>
      <artifactId>jboss-profileservice-spi</artifactId>
      <version>5.1.0.GA</version>
    </dependency>
    <dependency>
      <groupId>org.jboss.jbossas</groupId>
      <artifactId>jboss-as-client</artifactId>
      <version>5.1.0.GA</version>
      <type>pom</type>
```

```
    </dependency>
  </dependencies>
</plugin>
```

You can also use the CARGO JBoss remote deployer to remotely deploy to JBoss farms versions 5.x and newer. To do so, use these two properties:

- `cargo.jboss.clustered`: if `true`, deployment is done in the `farm` directory
- `cargo.jboss.configuration`: JBoss profile name, default name is `default`

JBoss 7.0.x and JBoss 7.1.x

The same instructions for JBoss 5.x, 5.1.x and 6.x also apply for JBoss 7.x, with some differences:

- The JAR files to include are different
- JBoss 7.x uses the `cargo.jboss.management.port` port.

The following examples detail these differences.

Here is an example code for the users of the Java API:

```
/**
 * Add all JARs in a folder in the list of files (recursive).
 * @param folder Folder to recursively scan.
 * @param files List containing all files.
 */
public static void addAllJars(File folder, List<URL> files) throws
Exception
{
    if (folder.isDirectory())
    {
        for (File file : folder.listFiles())
        {
            if (file.isFile() && file.getName().endsWith(".jar"))
            {
                files.add(file.toURI().toURL());
            }
            else if (file.isDirectory())
            {
                addAllJars(file, files);
            }
        }
    }
}
```

```
List<URL> urls = new ArrayList<URL>();

// Add many libraries from JBOSS_HOME/modules
addAllJars(new File(this.localContainer.getHome(), "modules"),
urls);

// Create a ClassLoader contaning all these JARs
URL[] urlsArray = new URL[urls.size()];
urlsArray = urls.toArray(urlsArray);
URLClassLoader classLoader = new URLClassLoader(urlsArray,
this.getClass().getClassLoader());
Thread.currentThread().setContextClassLoader(classLoader);

// Now, create the JBoss Remote container
...
```

Here is an example Maven2 plugin configuration:

```
<plugin>
  <groupId>org.codehaus.cargo</groupId>
  <artifactId>cargo-maven2-plugin</artifactId>
  <version>${cargo.plugin.version}</version>
  <configuration>
    <container>
      <containerId>jboss7x</containerId>
      <type>remote</type>
    </container>
    <configuration>
      <type>runtime</type>
      <properties>
        <cargo.hostname>production27</cargo.hostname>

<cargo.jboss.management.port>19999</cargo.jboss.management.port>
      </properties>
    </configuration>
  </configuration>
  <dependencies>
    <dependency>
      <groupId>org.jboss.as</groupId>
      <artifactId>jboss-as-controller-client</artifactId>
      <version>7.0.2.Final</version>
    </dependency>
  </dependencies>
</plugin>
```

JBoss socket connection bugs

On some Linux distributions, remote deployment may fail with an exception like:

```
Caused by: java.io.IOException: Can not get connection to
server.
           Problem establishing socket connection for
InvokerLocator
           [socket://host:32342/]
```

That is a known bug, documented in http://community.jboss.org/wiki/WhydoIgetasocketconnectionerrorwhenusingremoteJBossAS and the solution presented on that document is to use a cron job as the root user to fix the file when it gets broken.

First, create the correct version of `/etc/hosts` and save it somewhere, such as `/etc/hosts.fixed`, and populate it with the following contents (replace `myhost` with the hostname of your computer, as reported by the `hostname` command, and the correct IP assignment, as reported by `ipconfig`):

/etc/hosts.fixed

```
127.0.0.1 myhost localhost localhost.localdomain
::1 myhost localhost6 localhost6.localdomain6
127.0.1.1 myhost
## (optional entry); update if assigned a new address from DHCP
192.168.1.5 myhost

## The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost  ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

Important: If possible, enter your hostname with both the "basic" hostname and the fully-qualified domain name.

Next, create a script named `/etc/restore-etc-hosts.sh` to replace the `/etc/hosts` file if Network Manager breaks it:

/etc/restore-etc-hosts.sh

```
#!/bin/sh

if [ `grep -c NetworkManager /etc/hosts` -eq 1 ]; then
   cp /etc/hosts.fixed /etc/hosts
fi
```

Finally, setup a cron job to run this script as often as you like. We recommend every couple of minutes.

```
*/3 * * * * /etc/restore-etc-hosts.sh
```

Once the `/etc/hosts` file is corrected, restart JBoss AS and run the tests again. Your socket connection error should be gone!