JBoss Data Grid lab guides

Lab 6

Thomas Qvarnström

v1.1 2015-12-06

Table of Contents

This guide explains the steps for running lab 6, either follow the steps in the step-by-step section or if you feel adventurous try to accomplish goals without the step-by-step guide.

1. Background

When the security department review the new solution with client server mode, they expressed worries about the fact that clients are not authenticated. To go live with Client server mode we need to implement authentication using simple username password.

2. Use-case

Increase the security by adding authentication for the Data Grid running in Client/Server mode.

3. These are the main steps of lab 6

1. Setup security for the JDG lab
2. Implement a simple call back login handler, used by the HR client.

3.1. Setup the lab environment

To assist with setting up the lab environment we have provided a shell script that does this.

NOTE *If you previously setup up lab 5 using this script there is no need to do this for lab 6*

1. Run the shell script by standing in the jdg lab root directory (~/jdg-labs) execute a command like this

```
$ sh init-lab.sh --lab=6
```

NOTE *If the EAP and JDG servers are running stop them*

4. Step-by-Step

1. Open `target/jboss-datagrid-6.3.0-server/standalone/configuration/standalone.xml` using vi or text editor of choice
2. Add authentification to hotrod endpoint in subsystem `urn:infinispan:server:endpoint:...` like this:

```
<subsystem xmlns="urn:infinispan:server:endpoint:6.1">
    <hotrod-connector socket-binding="hotrod" cache-container="local">
        <topology-state-transfer lazy-retrieval="false" lock-timeout="1000"
replication-timeout="5000"/>
        <authentication security-realm="ApplicationRealm">
            <sasl server-name="tasks" mechanisms="DIGEST-MD5" qop="auth">
                <policy>
                    <no-anonymous value="true"/>
                </policy>
                <property name="com.sun.security.sasl.digest.utf8">true</property>
            </sasl>
        </authentication>
    </hotrod-connector>
    <memcached-connector socket-binding="memcached" cache-container="local"/>
    <rest-connector virtual-server="default-host" cache-container="local" security-
domain="other" auth-method="BASIC"/>
</subsystem>
```

3. Add security to the `urn:infinispan:server:core:...` subsystem, like this:

```
<subsystem xmlns="urn:infinispan:server:core:6.1" default-cache-container="local">
    <cache-container name="local" default-cache="default" statistics="true">
        <security>
          <authorization>
              <identity-role-mapper/>
              <role name="taskusers" permissions="READ WRITE BULK_READ"/>
          </authorization>
        </security>
    ...
    </cache-container>
</subsystem>
```

4. Later in the same subsystem configuraiton add the following:

```
<local-cache name="tasks" start="EAGER">
    <locking acquire-timeout="30000" concurrency-level="1000" striping="false"/>
    <transaction mode="NONE"/>
    <security>
      <authorization roles="taskusers"/>
    </security>
</local-cache>
```

5. Create an application user (replace with username of your choice)

```
./target/jboss-datagrid-6.3.0-server/bin/add-user.sh -a -g taskusers -u thomas -p
thomas-123 -r ApplicationRealm
```

6. Start the servers runing the following commands from different console windows.

   EAP Server:

   ```
   $ ./target/jboss-eap-6.3/bin/standalone.sh
   ```

   JDG Server:

   ```
   $ ./target/jboss-datagrid-6.3.0-server/bin/standalone.sh -Djboss.socket.binding.port
   -offset=100
   ```

7. In JBoss Developer Studio create a new Java Class under `org.jboss.infinispan.demo` called `LoginHandler` implemeted implement it like this:

```
package org.jboss.infinispan.demo;

import java.io.IOException;

import javax.security.auth.callback.Callback;
import javax.security.auth.callback.CallbackHandler;
import javax.security.auth.callback.NameCallback;
import javax.security.auth.callback.PasswordCallback;
import javax.security.auth.callback.UnsupportedCallbackException;
import javax.security.sasl.RealmCallback;

public class LoginHandler implements CallbackHandler {

    final private String login;
    final private char[] password;
    final private String realm;

    public LoginHandler(String login, char[] password, String realm) {
        this.login = login;
        this.password = password;
        this.realm = realm;
    }

    @Override
    public void handle(Callback[] callbacks) throws IOException,
UnsupportedCallbackException {
        for (Callback callback : callbacks) {
            if (callback instanceof NameCallback) {
                ((NameCallback) callback).setName(login);
            } else if (callback instanceof PasswordCallback) {
                ((PasswordCallback) callback).setPassword(password);
            } else if (callback instanceof RealmCallback) {
                ((RealmCallback) callback).setText(realm);
            } else {
                throw new UnsupportedCallbackException(callback);
            }
        }
    }

}
```

8. Open Config.java and add the LoginHandler as a callbackHandler to together with the other security configuration like this.

```
security()
    .authentication()
        .enable()
        .serverName("tasks")
        .saslMechanism("DIGEST-MD5")
        .callbackHandler(new LoginHandler("thomas", "thomas-123".toCharArray(),
"ApplicationRealm"));
```

NOTE	*If you do changed the username and password previously when creating the application users you need to update them here*

The final Config.java should look like this:

```
package org.jboss.infinispan.demo;

import javax.enterprise.inject.Produces;

import org.infinispan.client.hotrod.RemoteCache;
import org.infinispan.client.hotrod.RemoteCacheManager;
import org.infinispan.client.hotrod.configuration.ConfigurationBuilder;
import org.jboss.infinispan.demo.model.Task;

/**
 * This class produces configured cache objects via CDI
 *
 * @author tqvarnst
 *
 */
public class Config {

    @Produces
    public RemoteCache<Long, Task> getRemoteCache() {
        ConfigurationBuilder builder = new ConfigurationBuilder();
        builder.addServer()
            .host("localhost").port(11322)
            .security()
            .authentication()
                .enable()
                .serverName("tasks")
                .saslMechanism("DIGEST-MD5")
                .callbackHandler(new LoginHandler("thomas", "thomas-123".toCharArray(
), "ApplicationRealm"));
        return new RemoteCacheManager(builder.build(), true).getCache("tasks");
    }
}
```

9. Save and run the Arquillian test
10. Open `TaskServerTest.java` and add LoginHandler to to the ShrinkWrap test

```
    .addClass(LoginHandler.class)
```

11. Deploy the application using the following command from lab6 dir

```
$ mvn clean package jboss-as:deploy
```

12. Congratulations you are done with lab 6.