**Install and configure Nexus Repository**

Adding custom metadata to artifacts in the Nexus Repository requires Nexus Professional. Another Open Source Repository that offers the same functionality is Artifactory from JFrog. Both offer a REST API and a Professional Version. In both cases you need the Pro Version to add metadata to artifacts.

**Setup Nexus**

To setup Nexus follow these steps:

* Download Nexus Pro from Sonatype, http://www.sonatype.com/Products/Nexus-Professional/Purchase/Free-Trial
* Extract the archive
* The nexus-custom-metadata-plugin is not enabled by default. To enable it copy it to the plugin-repository folder.

|  |
| --- |
| **cp** -r nexus-professional-trial-2.0.6**/**nexus**/**WEB-INF**/**optional-plugins**/**nexus-custom-metadata-plugin-2.0.6**/**  nexus-professional-trial-2.0.6**/**nexus**/**WEB-INF**/**plugin-repository**/** |

* Start Nexus

|  |
| --- |
| nexus-professional-trial-2.0.6/bin$ .**/**nexus start |

* Open Nexus in your browser and register for the 14 day trial version, http://localhost:8081/nexus
* You will receive an Email with the Trial registration code via Email within a minute
* Login with admin/admin123

To verify that the **nexus-custom-metadata-plugin** was installed successful go to **Administration -> Plugin Console**. There you should see the following information. If you click the links you should be forwarded to the REST API of the Plugin.

* Nexus Professional Plugin :: Custom Metadata
* <http://localhost:8081/nexus/nexus-custom-metadata-plugin/m2/docs/index.html>

Under the same section you will find the Core Documentation as well. That one contains the Core REST API calls that you can use by default. Unfortunately that API does not allow you to store metadata for artifacts.

* Nexus : Core Plugins : Core Documentation
* <http://localhost:8081/nexus/nexus-core-documentation-plugin/core/docs/index.html>

**Setup Maven Project**

Under the REST API documentation you will find a client.jar that provides all the REST Models we need for calling the REST API. Therefore download the JAR and upload it to the ThirdParty repository in Nexus.

* Download from: <http://localhost:8081/nexus/nexus-custom-metadata-plugin/m2/docs/nexus-custom-metadata-plugin-client.jar>
* Next, you need to configure your Maven **$HOME/.m2/settings.xml** to use Nexus for resolving Maven artifacts. Add the following lines to the **settings.xml**.

|  |
| --- |
| **<mirrors>**  **<mirror>**  *<!--This sends everything else to /public -->*  **<id>**nexus**</id>**  **<mirrorOf>**\***</mirrorOf>**  **<url>**http://localhost:8081/nexus/content/groups/public**</url>**  **</mirror>**  **</mirrors>**  **<profiles>**  **<profile>**  **<id>**nexus**</id>**  *<!--Enable snapshots for the built in central repo to direct -->*  *<!--all requests to nexus via the mirror -->*  **<repositories>**  **<repository>**  **<id>**central**</id>**  **<url>**http://central**</url>**  **<releases><enabled>**true**</enabled></releases>**  **<snapshots><enabled>**true**</enabled></snapshots>**  **</repository>**  **</repositories>**  **<pluginRepositories>**  **<pluginRepository>**  **<id>**central**</id>**  **<url>**http://central**</url>**  **<releases><enabled>**true**</enabled></releases>**  **<snapshots><enabled>**true**</enabled></snapshots>**  **</pluginRepository>**  **</pluginRepositories>**  **</profile>**  **</profiles>**  **<activeProfiles>**  *<!--make the profile active all the time -->*  **<activeProfile>**nexus**</activeProfile>**  **</activeProfiles>** |

* After that we are ready to start developing the plugin.

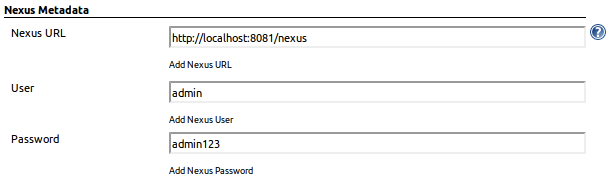
**Integrate Jenkins and Nexus with a custom Jenkins Plugin**

* Now that we have Jenkins and Nexus up and running we can go back to do some coding. I suggest you clone the code from my github repository. I will explain all changes I made step-by-step.
* **Add Dependencies**
* For calling the Nexus REST API’s I decided to use the Jersey Framework. Simply add the following dependencies to your **pom.xml**.

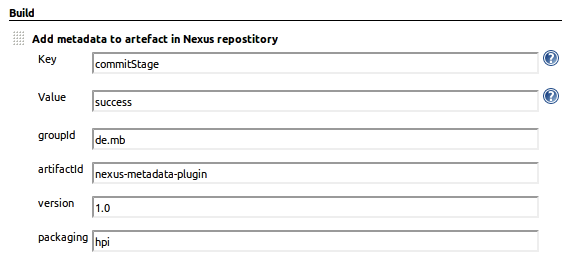
|  |
| --- |
| **<dependency>**  **<groupId>**com.sun.jersey**</groupId>**  **<artifactId>**jersey-client**</artifactId>**  **<version>**1.12**</version>**  **</dependency>**  **<dependency>**  **<groupId>**org.sonatype.nexus**</groupId>**  **<artifactId>**nexus-rest-api-model**</artifactId>**  **<version>**2.0.6**</version>**  **</dependency>**  **<dependency>**  **<groupId>**org.sonatype.nexus**</groupId>**  **<artifactId>**nexus-custom-metadata-plugin-client**</artifactId>**  **<version>**1.0**</version>**  **</dependency>** |

* **Configure Metadata Plugin**
* Rename the **HelloWorldBuilder** packages under **src/main/java** and **src/main/resources** to **NexusMetadataBuilder**.
* To make the Jenkins Plugin configurable add the following lines to the **global.jelly**. This allows us to specify the Nexus URL and credentials in the Jenkins configuration.

|  |
| --- |
| **<f:section** title="Nexus Metadata"**>**  **<f:entry** title="Nexus URL" field="nexusUrl"  description="Add Nexus URL"**>**  **<f:textbox** **/>**  **</f:entry>**  **<f:entry** title="User" field="nexusUser"  description="Add Nexus User"**>**  **<f:textbox** **/>**  **</f:entry>**  **<f:entry** title="Password" field="nexusPassword"  description="Add Nexus Password"**>**  **<f:textbox** **/>**  **</f:entry>**  **</f:section>** |

* [](http://blog.codecentric.de/files/2012/08/configurePlugin.png)
* In the next step we add some fields that we can configure on the Job. We need to be able to tell the Plugin the Key/Value we want to store with the artifact and the location of the artifact in the Nexus repository (groupId, artifactId, version, packaging).

|  |
| --- |
| **<f:entry** title="Key" field="key"**>**  **<f:textbox** **/>**  **</f:entry>**  **<f:entry** title="Value" field="value"**>**  **<f:textbox** **/>**  **</f:entry>**  **<f:entry** title="groupId" field="groupId"**>**  **<f:textbox** **/>**  **</f:entry>**  **<f:entry** title="artifactId" field="artifactId"**>**  **<f:textbox** **/>**  **</f:entry>**  **<f:entry** title="version" field="version"**>**  **<f:textbox** **/>**  **</f:entry>**  **<f:entry** title="packaging" field="packaging"**>**  **<f:textbox** **/>**  **</f:entry>** |

* [](http://blog.codecentric.de/files/2012/08/custom-job-config.png)
* Now we can create the **NexusMetadataBuilder.java** [[5]](https://blog.codecentric.de/en/2012/08/tutorial-create-a-jenkins-plugin-to-integrate-jenkins-and-nexus-repository/#5) class that will put all the pieces together. This class takes care of reading the plugin configuration (NexusUrl & Credentials) as well as the configuration from the Build Job (groupId, artifactId, version, packaging). In perform Method we create the REST Client that calls out to the Nexus REST API. The first call uses the Nexus Core API and checks the status of Nexus. It should return “200 OK” when its up and running. The REST Mount Point is: **/service/local/status**

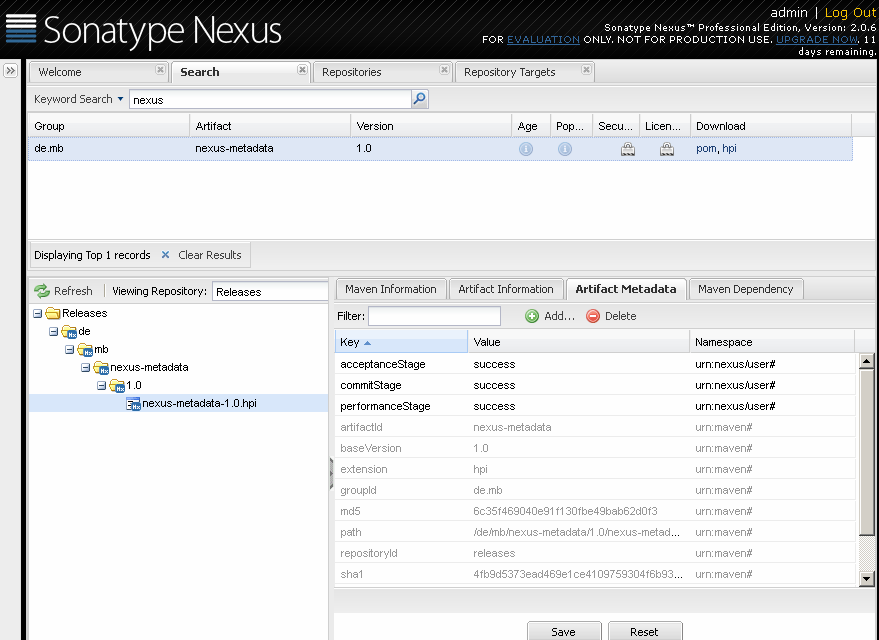
|  |
| --- |
| *// setup REST-Client*  ClientConfig config = **new** DefaultClientConfig();  Client client = Client.create(config);  client.addFilter( **new** HTTPBasicAuthFilter(user, password) );  WebResource service = client.resource( url );    listener.getLogger().println("Check that Nexus is running");  String nexusStatus = service.path("service").path("local").path("status").accept(MediaType.APPLICATION\_JSON).get(ClientResponse.**class**).toString();  listener.getLogger().println(nexusStatus + "**\n**"); |

* For the custom metadata plugin we need to encode the “subject” (location of the artifact) with Base64, which is not really RESTful in my opinion. The REST Mount Point is: **/service/local/index/custom\_metadata/{repository}/{subject}**

|  |
| --- |
| String artefact = "urn:maven/artifact#"+getGroupId()+":"+getArtifactId()+":"+getVersion()+"::"+getPackaging()+"";  listener.getLogger().println("GET metadata for artefact " + artefact);  String encodedString = **new** String( Base64.encode( artefact.getBytes() ) );    listener.getLogger().println("POST: add new metadata to artefact " + artefact);  CustomMetadataRequest customRequest = getCustomMetadataRequest( getKey(), getValue() );    service.path("service").path("local").path("index").path("custom\_metadata").path("releases")  .path(encodedString).accept( MediaType.APPLICATION\_JSON ).post( customRequest ); |

* **Build Plugin from Github**
* If you want to build the project from Github follow these steps:

|  |
| --- |
| **git@**github.com:marcelbirkner**/**nexus-metadata-plugin.git  **cd** nexus-metadata-plugin  mvn clean package |

* Afterwards you will find the **nexus-metadata-plugin.hpi** plugin in the **/target** folder. Once you deploy the plugin and configure each build step of the continuous delivery build pipeline the deployed artifact in Nexus will have the metadata attached, see diagram.
* [](http://blog.codecentric.de/files/2012/08/screenshot-nexus-metadata.png)