

Maven Integration with Jfrog

By

Keshav Kummari

Click on *libs-release-local* to generate maven security files i.e. *settings.xml* & *settings-security.xml*

The screenshot displays the JFrog Artifactory web interface. At the top, a green header bar contains the JFrog Artifactory logo, a search icon, and a user greeting "Welcome, admin". Below the header, a main banner states "Artifactory is happily serving 0 artifacts" and "Artifactory Version 5.4.5 (Uptime is 0d 0h 23m 36s)".

The interface is divided into several sections:

- Quick Search:** A search bar with the text "Quick Search" and "Or go to >". Below it are links for "Package Search", "Archive Search", "Property Search", "Checksum Search", and "JCenter Search".
- Set Me Up:** A section titled "Set Me Up" with a "Filter by Repository Key" dropdown. It lists repository keys: "m libs-release", "m libs-snapshot", "m **libs-release-local**", "m libs-snapshot-local", and "m jcenter". The "libs-release-local" key is highlighted in yellow.
- Last Deployed Builds:** A section titled "Last Deployed Builds" with a timestamp "(27/07/17 21:39:23)". It states "No builds to display." and includes a link "Learn how to integrate your build information with Artifactory." A "Transcript" button is visible on the right.
- Most Downloaded Artifacts:** A section titled "Most Downloaded Artifacts" with a timestamp "(27/07/17 21:39:23)". It states "No artifacts to display." and includes a link "Learn how to deploy your artifacts to Artifactory."

At the bottom, there is a navigation bar with icons and labels for various features: "User Guide", "Webinar Signup", "Support Portal", "Stackoverflow", "Blog", "Rest API", "High Availability", "JFrog Bintray Distribution", "JFrog Xray", "JFrog CLI", "Build Integration", and "Docker".

Click on *libs-release-local* & Click on *Generate Maven Settings*

Set Me Up

Tool: **m Maven** **Generate Maven Settings**

Repository: **libs-release-local**

Type password to insert your credentials to the code snippets
Type Password

General
Click on "Generate Maven Settings" in order to resolve artifacts through Virtual or Remote repositories.

Deploy
To deploy build artifacts through Artifactory you need to add a deployment element with the URL of a target local repository to which you want to deploy your artifacts. For example:

```
1 <distributionManagement>
2   <repository>
3     <id>central</id>
4     <name>Nands-MacBook-Pro-15.local-releases</name>
5     <url>http://localhost:8081/artifactory/libs-release-local</url>
6   </repository>
7 </distributionManagement>
```

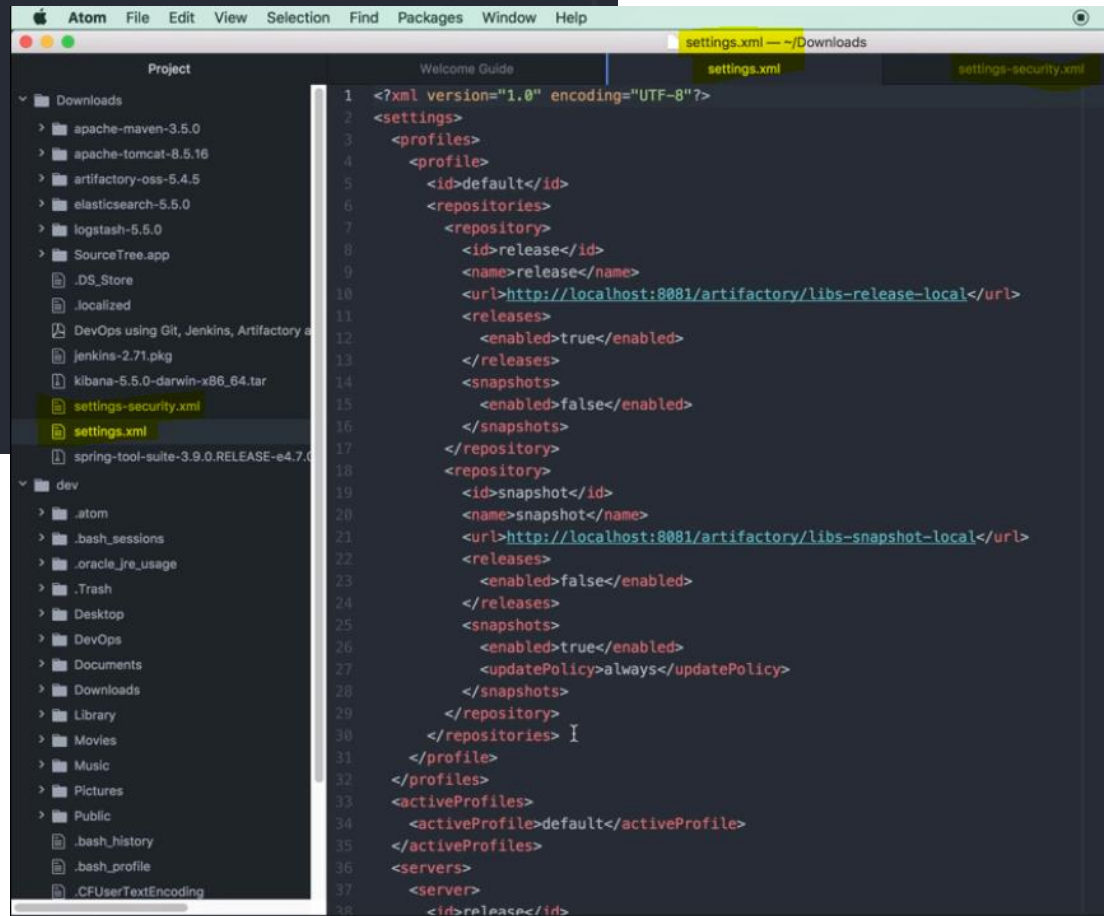
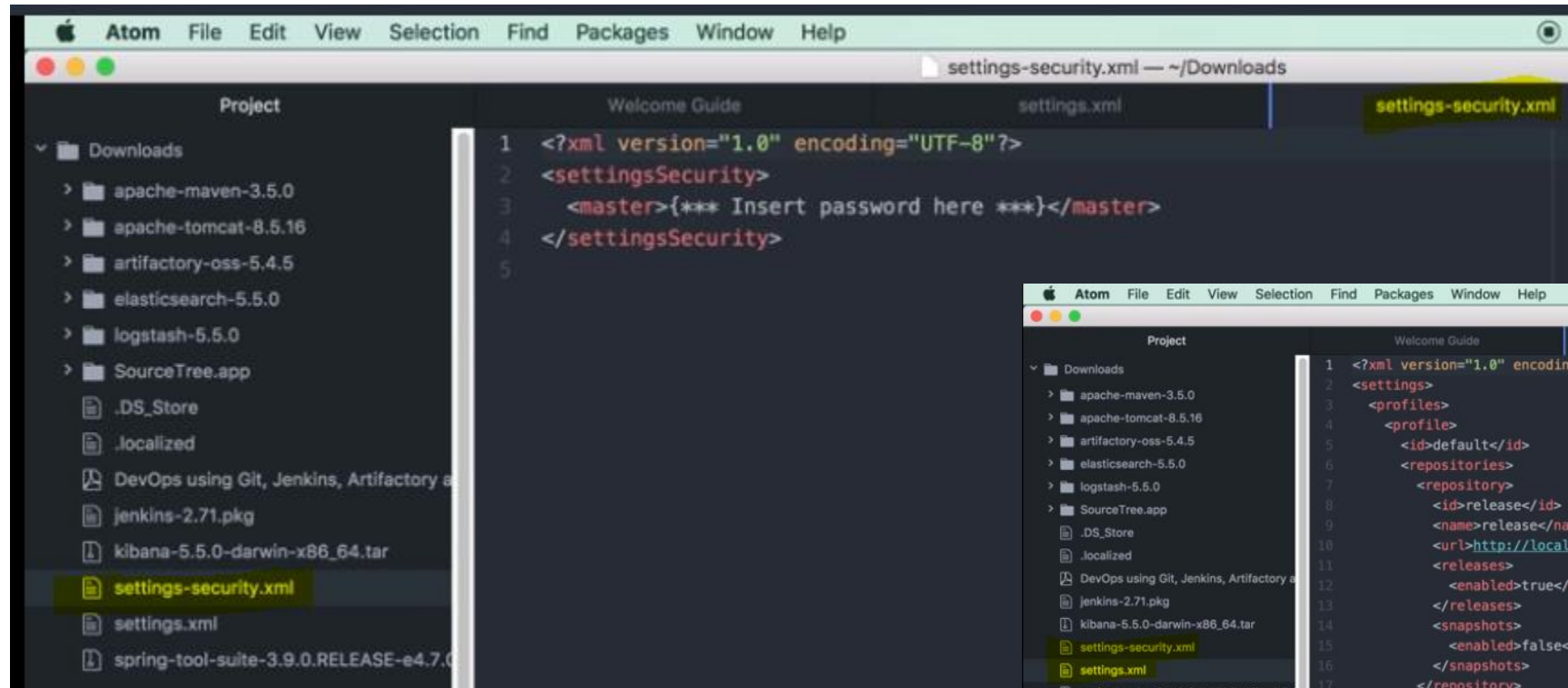
Click on “Generate Settings” & Download

The screenshot shows the JFrog Artifactory web interface. A modal window titled "Set Me Up" is open, displaying three buttons: "Generate Settings" (highlighted in yellow), "Download Snippet" (highlighted in yellow), and "Deploy Settings". Below the buttons is a text area containing an XML configuration for Maven settings. The XML defines two servers: "central" and "snapshots", both using placeholder values for username and password. The background interface includes a sidebar with navigation icons, a search bar, and a main content area with the message "Artifactory is happily serving 0 artifacts".

Set Me Up

[Generate Settings](#) [Download Snippet](#) [Deploy Settings](#)

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <settings xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.1.0
  http://maven.apache.org/xsd/settings-1.1.0.xsd" xmlns="http://maven.apache.org/SETTINGS/1.1.0"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
4   <servers>
5     <server>
6       <username>${security.getCurrentUsername()}</username>
7       <password>${security.getEscapedEncryptedPassword(!"*** Insert encrypted password here ***")}
8     </password>
9     <id>central</id>
10    </server>
11    <server>
12      <username>${security.getCurrentUsername()}</username>
13      <password>${security.getEscapedEncryptedPassword(!"*** Insert encrypted password here ***")}
14    </password>
15    <id>snapshots</id>
16  </server>
17  </servers>
18  <profiles>
19    <profile>
```



Go to /.m2 dir & copy **settings.xml** & **settings-security.xml** files & generate password & update them in those two files

```
drwxr-xr-x  3 dev  staff   176 Jul 22 23:41 PUBLIC
[Nands-MacBook-Pro-15:~ dev$ mkdir .m2
[Nands-MacBook-Pro-15:~ dev$ cp Downloads/*.xml .m2/
[Nands-MacBook-Pro-15:~ dev$ cd .m2
[Nands-MacBook-Pro-15:.m2 dev$ ls -l
total 16
-rw-r--r--@ 1 dev  staff   128 Jul 27 22:03 settings-security.xml
-rw-r--r--@ 1 dev  staff  1297 Jul 27 22:03 settings.xml
[Nands-MacBook-Pro-15:.m2 dev$ mvn -emp password
{mhtxRufZ5fgHVDgG9ypg7i3ByD4uiovVwTDCGj52h5E=}
[Nands-MacBook-Pro-15:.m2 dev$ vi settings-security.xml
[Nands-MacBook-Pro-15:.m2 dev$ mvn -ep password
{AEnpFTB12F0H2chBZebXFf3UG018pPCQu+/Eq/HMzwA=}
[Nands-MacBook-Pro-15:.m2 dev$
```


- **# vi settings-security.xml**

```
settings-security.xml — ~/Downloads
Welcome Guide  settings.xml  settings-security.xml
1  <?xml version="1.0" encoding="UTF-8"?>
2  <settingsSecurity>
3    <master>{*** Insert password here ***}</master>
4  </settingsSecurity>
5
```

- **# vi settings.xml**

```
settings.xml
. .oracle_jre_usage
. .Trash
. Desktop
. DevOps
. Documents
. Downloads
. Library
. Movies
. Music
. Pictures
. Public

36  <servers>
37    <server>
38      <id>release</id>
39      <username>admin</username>
40      <password>{AEnpFTBi2F0H2chBZebXFf3UG0l8pPCQu+/Eq/HMzWA=}</password>
41    </server>
42    <server>
43      <id>snapshot</id>
44      <username>admin</username>
45      <password>{AEnpFTBi2F0H2chBZebXFf3UG0l8pPCQu+/Eq/HMzWA=}</password>
46    </server>
47  </servers>
48 </settings>
```

Do maven clean now!

```
[Nands-MacBook-Pro-15:.m2 dev$ ls -al .m2
ls: .m2: No such file or directory
[Nands-MacBook-Pro-15:.m2 dev$ ls -al
total 16
drwxr-xr-x  4 dev  staff   136 Jul 27 22:05 .
drwxr-xr-x+ 24 dev  staff   816 Jul 27 22:05 ..
-rw-r--r--@ 1 dev  staff   144 Jul 27 22:05 settings-security.xml
-rw-r--r--@ 1 dev  staff  1329 Jul 27 22:05 settings.xml
[Nands-MacBook-Pro-15:.m2 dev$ mvn clean
[INFO] Scanning for projects...
[INFO] -----
[INFO] BUILD FAILURE
[INFO] -----
[INFO] Total time: 0.614 s
[INFO] Finished at: 2017-07-27T22:06:38-05:00
[INFO] Final Memory: 8M/245M
[INFO] -----
[ERROR] The goal you specified requires a project to execute but there is no POM in this directory (/Users/dev/.m2).
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MissingProjectException
```


Execute # mvn help:describe

```
[Nands-MacBook-Pro-15:~]$ mvn help:describe
```

- Build may fail, but it will generate required dependencies.
- Now, you can see it has created repository directory, PFS below:

```
[Nands-MacBook-Pro-15:~]$ ls -l
total 16
drwxr-xr-x  14 dev  staff   476 Jul 27 22:06 repository
-rw-r--r--@  1 dev  staff   144 Jul 27 22:05 settings-security.xml
-rw-r--r--@  1 dev  staff  1329 Jul 27 22:05 settings.xml
[Nands-MacBook-Pro-15:~]$ ls -l repository/
total 0
drwxr-xr-x  3 dev  staff  102 Jul 27 22:06 backport-util-concurrent
drwxr-xr-x  3 dev  staff  102 Jul 27 22:06 classworlds
drwxr-xr-x  3 dev  staff  102 Jul 27 22:06 com
drwxr-xr-x  3 dev  staff  102 Jul 27 22:06 commons-cli
drwxr-xr-x  3 dev  staff  102 Jul 27 22:06 commons-lang
drwxr-xr-x  3 dev  staff  102 Jul 27 22:06 jdom
drwxr-xr-x  3 dev  staff  102 Jul 27 22:06 jtidy
drwxr-xr-x  3 dev  staff  102 Jul 27 22:06 junit
drwxr-xr-x  6 dev  staff  204 Jul 27 22:06 org
drwxr-xr-x  4 dev  staff  136 Jul 27 22:06 plexus
drwxr-xr-x  3 dev  staff  102 Jul 27 22:06 xmlpull
drwxr-xr-x  3 dev  staff  102 Jul 27 22:06 xpp3
[Nands-MacBook-Pro-15:~]$
```