Project Idea

A Jenkins plugin to wrap the https://github.com/cdancy/artifactory-rest library, allowing Pipeline scripts to interact easily with Artifactory repository instances via the Artifactory REST API, and eliminating the need to parse responses, making pipeline scripts more concise.

As explained in the mailing list, it is essential for build steps to return only simple data types, like string, integers, lists of simple data types, maps of simple data types. Steps cannot return methods, nor objects with behaviors. Also, using the GlobalVariable is not recommended, however it might be inevitable. The student is expected to study the use of GlobalVariable in plugins that use it and ask for guidance on the Pipeline Authoring SIG gitter chat on this matter. The student should study the docker pipeline plugin source code and the pipeline loader plugin to understand how they use the GlobalVariable.

In the following, we show how the finished plugin would look like from the user point of view in a Jenkins Pipeline DSL program. This is not a specification, it is only an example. The student is expected to study the the artifactory-rest library, the Jenkins Plugin tutorials and the Scripted Pipeline syntax, and propose a proper Jenkins Pipeline DSL syntax for this project. The examples below should not be taken literally and are not necessarily prescriptive.

To create an artifactory client use the artifactoryClient step:

```
def artifactoryClient = artifactoryClient url: “arturl”, username: “user”, password: “secret-or-api-token”
```

The artifactory-rest client can set a verbosity level (see this wiki for a groovy example of the underlying implementation of the logger framework supporting the verbosity):

```
artifactoryClient.verbosity = “level”
```

The artifactory-rest client can set System properties to configure the underlying JClouds library:

```
artifactoryClient.setProperty("jclouds.so-timeout", "60000")
```

To query an artifact property:

```
def resp = artifactoryStorageApi.getProperties client: artifactoryClient, repoKey: “repo-key”, itemPath: “path/to/item”
```

The response is automatically parsed and the user can simply read the properties:

```
echo resp.errors
echo resp.errors[0].context
echo resp.errors[0].message
echo resp.get(“key”)
echo resp. … (etc.)
```

Alternatively, it has been proposed that the REST API could be generalized:

```
restApiClient.withServer(url: “foo://artifactory”, project: project, repo: repo, credentialsId: “myartifactory”) {
  def response = restApiClient.get “api/storage/...”
  echo response....
}
```

Existing solutions

There is already the Artifactory plugin which also integrates Artifactory to Jenkins Pipeline DSL, see Working With Pipeline Jobs in Jenkins. That plugin offers a limited set of operations which are coded directly in the plugin. This proposal provides an alternative. This plugin allows the Pipeline Script author to access the Artifactory REST API directly. The student is expected to do their own comparison of the two libraries (artifactory-rest and the jenkins-artifactory-plugin).

Quick start

There are many technologies to use together to form this plugin. The student who wishes to get started will need to:

- Try out Artifactory REST APIs using Curl or Groovy
- Try writing a small client using one of REST API Client libraries
- study plugin tutorials on how to write a Pipeline Step plugin
    - Tutorials listed on the student information page
    - Writing Pipeline compatible plugins
    - Writing Pipeline steps
    - Updating plugin for Pipeline
    - looking at existing pipeline compatible plugins will be very useful. Example:
        - External Workspace Manager (look at the steps folder, the steps themselves, and their execution classes)
- study the artifactory-rest library. The student can also learn from a similar library about adding logger facilities to the library by reading this wiki.
- create a basic custom pipeline compatible plugin and load it in Jenkins (see the plugin tutorials)

Open questions

- GlobalVariables have been debated on the mailing list, and their use is controversial. However they do work. Example of global variables usage in working plugins: docker, pipeline loader plugin.
- This approach has to be compared with the existing artifactory plugin.

Skills to study/improve

- Java
- REST API
- Artifactory
- Jenkins Pipeline

Links

- https://github.com/cdancy/artifactory-rest
- https://github.com/cdancy/jenkins-rest
- https://github.com/cdancy/bitbucket-rest
- https://www.jfrog.com/confluence/display/RTF/Artifactory+REST+API

- https://www.jfrog.com/confluence/display/RTF/Working+With+Pipeline+Jobs+in+Jenkins