Project Idea

A Jenkins plugin to wrap the https://github.com/cdancy/bitbucket-rest library, allowing Pipeline scripts to interact easily with Bitbucket instances via the Bitbucket 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 .

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 bitbucket-rest library, the Jenkins Plugin tutorials and the Scripted Pipeline syntax, and propose a proper Jenkins Pipeline DSL syntax for this project.

Examples

To create a bitbucket client use the bitbucketClient step:

```
def bbClient = bitbucketClient url: “bburl”, username: “user”, password: “secret”
```

You can use a Bitbucket Personal Token (bearer):

```
def bbClient = bitbucketClient url: “url”, token: “secret”
```

The Bitbucket-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):

```
bbClient.verbosity = “level”
```

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

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

To query a pull-request, use the bitbucket pull-request step:

```
def resp = bitbucketPullRequest client: bbClient, prNumber: 4, project: “BBPROJ”, repo: “GIT_REPO”
```

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.fromRef
echo resp.toRef
echo resp. … (etc.)
```

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

```
restApiClient.withServer(url: “foo://bitbucket”, project: project, repo: repo, credentialsId: “mybitbucket”)
{
  def response = restApiClient.get “api/pullRequests/changes”
```

```
  echo response.commits[0].message
}
```

Expectations

The student is expected to come up with a prototype to demonstrate the capability of the plugin, for example by implementing a single REST API method and a single response type, before proceeding with a complete implementation. This will help the student create proper step method calls and proper response objects.

Many interactions with a Bitbucket server are possible, see the full Javadoc for the Bitbucket REST library. Having low level access to the Bitbucket REST API allows users to interact with Bitbucket in ways that are not provided by existing higher level plugins.

Advanced concept: it would be interesting to generate this plugin automatically simply by reading the bitbucket-rest library. This would enable automatic updates to this project each time the underlying library is updated, and it would also enable the automatic maintenance of the Jenkins-rest plugin and the Artifactory-rest plugin. This could be done with OpenAPI (Swagger).

Quick start

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

- 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 bitbucket-rest library, try the examples in the wiki
- create a basic custom pipeline compatible plugin and load it in Jenkins (see the plugin tutorials)

Links

- https://github.com/cdancy/artifactory-rest
- https://github.com/cdancy/jenkins-rest
- https://github.com/cdancy/bitbucket-rest
- Discussion on returning simple data from pipeline steps.
- Example of a response content supplier for a Jenkins Pipeline Step
- OpenAPI (Swagger)

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.

Skills to improve/study

- Java
- REST API
- Bitbucket
- Jenkins Pipeline