gitlab-jenkins-sonar-api

gitlab-jenkins-sonar集成过程中，自动化操作所需要的关键API和操作方法
（1）jenkins-api
（2）gitlab-api
（3）sonar-api

1、jenkins-api

gem install jenkins_api_client
gem: https://rubygems.org/gems/jenkins_api_client
doc: http://www.rubydoc.info/gems/jenkins_api_client/1.4.3/

关键代码：

```
require 'jenkins_api_client'
@client = JenkinsApi::Client.new(:server_url => 'http://192.168.8.158:8890',
                                 :username => 'anonymous',
                                 :password => '')
ok=@client.job.create('test_create',File.read('config.xml'))
```

红色的是关键api方法，使用@client.job.create创建jenkins job任务
这个方法支持传入一个xml模板文件作为参数
这个后面需要要给jenkins加用户权限控制，使用username，password，现在先用anonymous

模板文件已经提供：

config.xml

注意这个模板文件用于pulic仓库的，如果是私有仓库，还需要额外加入验证的tag

不同job对应不同的repo，需要修改模板里的内容，主要修改的内容包括：remote git的地址、要扫描的分支名称、Sonar-Scanner的扫描参数等

使用nokogiri去修改xml的解析和修改
gem:https://rubygems.org/gems/nokogiri
http://www.nokogiri.org/tutorials/

关键代码：

```
@doc = Nokogiri::XML(File.open("config.xml"))
@doc.at_xpath("//hudson.plugins.git.UserRemoteConfig/url").content='http://192.168.8.158:8889/root/test.git'
sonar_properties=@doc.xpath("//hudson.plugins.sonar.SonarRunnerBuilder/properties").text
```

at_xpath方法使用xpath路径语言定义到要修改的标签，.content方法可以取出标签内的内容，然后可以赋值修改（注意.text不可以修改）

```ruby
require 'jenkins_api_client'
require 'nokogiri'

@client = JenkinsApi::Client.new(:server_url => 'http://192.168.8.158:8890',
                                 :username => 'anonymous',
                                 :password => '')

# modify config
@doc = Nokogiri::XML(File.open("config.xml"))
@doc.at_xpath("//hudson.plugins.git.UserRemoteConfig/url").content='http://192.168.8.158:8889/root/test.git'
@doc.at_xpath("//hudson.plugins.git.BranchSpec/name").content="*/master"
sonar_properties=@doc.xpath("//hudson.plugins.sonar.SonarRunnerBuilder/properties").text #sonar-properties


p File.read('config.xml')
ok=@client.job.create('test_create',File.read('config.xml'))
p ok
```

@client.job.create后，在远程jenkins，就创建了job

其它说明：

注意：（1）
xml模板文件中sonar-scanner的参数，projectKey建议使用 username:repo_name的形式（如 root:repo1）
sonar.projectKey、 sonar.projectName、sonar.projectVersion、sonar.sources、 sonar.language这些都是必要参数
如果还需要其他参数，去查文档

```
<builders>
  <hudson.plugins.sonar.SonarRunnerBuilder plugin="sonar@2.4.2">
    <project></project>
    <properties>
      sonar.projectKey=my:project985
      sonar.projectName=My project985
      sonar.projectVersion=1.11
      sonar.sources=src
      sonar.language=java
      sonar.sourceEncoding=utf-8
    </properties>
    <javaOpts></javaOpts>
    <additionalArguments></additionalArguments>
    <jdk>(Inherit From Job)</jdk>
    <task></task>
  </hudson.plugins.sonar.SonarRunnerBuilder>
</builders>
```

（2）
public仓库在使用sonar做CI时候，是下面的配置

```
<userRemoteConfigs>
  <hudson.plugins.git.UserRemoteConfig>
    <url>http://192.168.8.158:8889/root/badboy.git</url>
  </hudson.plugins.git.UserRemoteConfig>
</userRemoteConfigs>
```

如果请求private仓库，需要多一个credentials验证

```
<scm class="hudson.plugins.git.GitSCM" plugin="git@2.4.4">
  <configVersion>2</configVersion>
  <userRemoteConfigs>
    <hudson.plugins.git.UserRemoteConfig>
      <url>http://192.168.8.158:8889/root/test.git</url>
      <credentialsId>8aeebcf7-336f-4a57-b968-605e703f0102</credentialsId>
    </hudson.plugins.git.UserRemoteConfig>
  </userRemoteConfigs>
```

这个id对应 jenkins_home下的credential.xml文件内的配置（/var/lib/jenkins/credentials.xml）

```
<com.cloudbees.plugins.credentials.SystemCredentialsProvider plugin="credentials@2.0.7">
  <domainCredentialsMap class="hudson.util.CopyOnWriteMap$Hash">
    <entry>
      <com.cloudbees.plugins.credentials.domains.Domain>
        <specifications/>
      </com.cloudbees.plugins.credentials.domains.Domain>
      <java.util.concurrent.CopyOnWriteArrayList>
        <org.jenkinsci.plugins.plaincredentials.impl.StringCredentialsImpl plugin="plain-creder
          <scope>SYSTEM</scope>
          <id>0b852a32-5afe-442a-83e5-3e7d80375295</id>
          <description>GitLab API Token</description>
          <secret>RFnJcVKY2WRPLnEmAwQQgg==</secret>
        </org.jenkinsci.plugins.plaincredentials.impl.StringCredentialsImpl>
        <com.cloudbees.plugins.credentials.impl.UsernamePasswordCredentialsImpl>
          <id>8aeebcf7-336f-4a57-b968-605e703f0102</id>
          <description>hehe</description>
          <username>root</username>
          <password>kzWu9LmeSzoJi0Huo2+YucbPn0x2SC7rQ2r/6URQ5P8=</password>
        </com.cloudbees.plugins.credentials.impl.UsernamePasswordCredentialsImpl>
      </java.util.concurrent.CopyOnWriteArrayList>
    </entry>
  </domainCredentialsMap>
</com.cloudbees.plugins.credentials.SystemCredentialsProvider>
```

当目标仓库为private时候，需要repo的username和pwd（需要用户指明），
jenkins的api中并没有这个生成credential的api方法（已确定没有）
这个过程采用模拟post请求的方式完成，通过分析网络请求，使用下面的代码模拟提交

关键代码：

```
require 'net/http'
uri = URI.parse('http://192.168.8.158:8889/descriptor/com.cloudbees.plugins.credentials.CredentialsSelectHelper/resolver/com.cloudbees.plugins.credentials.CredentialsSelectHelper$SystemContextResolver/provider/com.cloudbees.plugins.credentials.SystemCredentialsProvider$ProviderImpl/context/jenkins/addCredentials')

params = {"json":'{"domain": "_","": "0","credentials": {"username": "av","password": "abcdfv","id": "123-321-456-789-987","description": "dsbdsbdsb","stapler-class": "com.cloudbees.plugins.credentials.impl.UsernamePasswordCredentialsImpl","$class": "com.cloudbees.plugins.credentials.impl.UsernamePasswordCredentialsImpl"}}'}

res = Net::HTTP.post_form(uri, params)
puts res.body
```

颜色部分是需要修改的， id自己可以定义，可以直接用于job模板中的credentialsId，用于git clone /fetch private私有仓库时候的验证
下面这个xml是带密码验证的模板，比上面那个多了一个credentialsId

config (1).xml

2、gitlab-api

上面job创建好后，模板文件中Build Triggers是 Build when a change is pushed to GitLab.
需要将hook地址，http://jekins_server_url/project/job_name ，（固定格式）
这个url需要加入到对应代码仓库的webhook列表里

`gem install gitlab`
https://rubygems.org/gems/gitlab
http://www.rubydoc.info/gems/gitlab/3.6.1

使用add_project_hook方法:

```
#add_project_hook(project, url, options = {}) ⇒ Gitlab::ObjectifiedHash
```

Adds a new hook to the project.

Examples:

```
Gitlab.add_project_hook(42, 'https://api.example.net/v1/webhooks/ci')
```

Parameters:

- **project** (Integer, String) — The ID or name of a project.
- **url** (String) — The hook URL.
- **options** (Hash) *(defaults to: {})* — A customizable set of options.
- **option** (Boolean) — :push_events Trigger hook on push events (0 = false, 1 = true)
- **option** (Boolean) — :issues_events Trigger hook on issues events (0 = false, 1 = true)
- **option** (Boolean) — :merge_requests_events Trigger hook on merge_requests events (0 = false, 1 = true)
- **option** (Boolean) — :tag_push_events Trigger hook on push_tag events (0 = false, 1 = true)

Returns:

- (Gitlab::ObjectifiedHash) — Information about added hook.

关键代码：

```
require 'gitlab'
g = Gitlab.client(endpoint: 'http://192.168.8.158:8889/api/v3',
                  private_token: 'YTyCv4978MXmdL2B9C62')
g.add_project_hook(project_id,'http://jekins_server_url/project/job_name')
```

到此，job任务建立了，webhook配置好了
像仓库push时候，会触发push event，gitlab通知webhook的地址，jenkins收到通知，触发器被触发，将远程版本库的代码拉到jenkins的工作区间，
再后进行启动sonarqube扫描

3、sonar-api

job任务build完成后，sonar-scanner结束扫描，数据可以在sonarqube的web ui界面看到
其Web Service API在 http://123.59.135.93:8891/api_documentation

项目扫描后的指标，通过api/resources这个接口获取
官方只提供了java的client，ruby下没有好用的gem，
用open-uri去取，不用curl，那个取的有问题

http://192.168.8.158:8891/api_documentation/api/resources
使用depth=-1，可以度量项目中的每一个文件

关键代码：

```
require 'open-uri'
require 'json'
require 'pp'

data=open('http://192.168.8.158:8891/api/resources/index?resource=my:project985&depth=-1&metrics=lines,blocker_violations').read
cc=JSON.parse(data)
pp data
```

注意，这个api在5.2下可以用,5.4以上被废弃，要修改（目前使用的5.2）
http://docs.sonarqube.org/display/DEV/API+Changes