Auto-Release Branch Creation

Pre-requisites for auto release branch creation

1. Repository (Gitlab/Bitbucket)
2. Jenkins Environment.

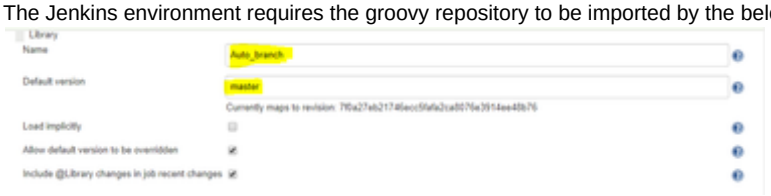
The Repository setup for using multiple groovy scripts to be added to the library.

The screenshot provides the details of Jenkins Global library and its content

1. Folder ./vars
2. Scripts auto branch, auto version,PushECR
3. The scripts are with the extensions .groovy as the libraries are written in groovy scripting.

Jenkins Environment

The Jenkins environment requires the groovy repository to be imported by the below way.



The repository details are mentioned in the Library which are part of Jenkins tool configuration.

Importing the Library in the Jenkins file.

The library mentioned in Jenkins is importing the list of libraries mentioned in Jenkins.

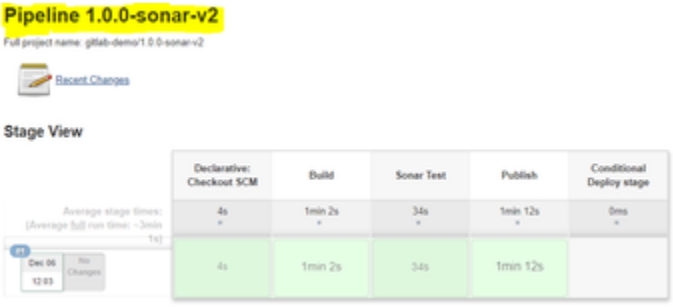
```
@Library(['my_sample_demo@scala_support','Auto_branch']) _

autobranch()
```

Jenkins output after generating the auto branch after effecting the Library

Groovy script for auto-branching based on tag:

```
pipeline {
    agent { label 'scala'}
    stages {
        stage('Build') {
            steps {
                checkout scm
                echo 'Start Compiling'
                build 'Scala'
            }
        }
        stage('Sonar Test') {
            steps {
                echo 'Start Sonar Test'
                sonar 'Scala'
            }
        }
        stage('Publish') {
            agent { label 'slave'}
            steps {
```

```
#!/usr/bin/groovy

import hudson.EnvVars;
import hudson.slaves.EnvironmentVariablesNodeProperty;
import hudson.slaves.NodeProperty;
import hudson.slaves.NodePropertyDescriptor;
import hudson.util.DescribableList;
import jenkins.model.Jenkins;
import groovy.transform.Field

@Field newversion

def call(Map config = [:]) {
nextversion = nextPackageVersion()
buildversion = buildnumber()
newversion = "${nextversion}"
print "Next Version"
print newversion
} //Map the configuration to Jenkins.

def nextPackageVersion(String latestVersion = "0.0.0") {
latestVersion = latestVersion.replaceAll("""", "");
def (major, minor, patch) = latestVersion.tokenize('.').collect { it.toInteger() }
def nextVersion
switch (env.BRANCH_NAME) {
case 'master':
nextVersion = "${major + 1}.${minor}.${patch}"
break
case ~/.develop./:
nextVersion = "${major}.${minor + 1}.${patch}"
break
case ~/.feature./:
nextVersion = "${major}.${minor}.${patch + 1}"
break

}

} //To fix the details of the next package version for the branch given

def buildnumber() {
def name = "${env.JOB_NAME}"
def job_data = Jenkins.instance.getItemByFullName(name)
def done_job
if ( "${env.BUILD_NUMBER}" == "1" ) {
done_job = 0
} else {
done_job = job_data.getLastSuccessfulBuild().getNumber()
}

print "job done"
done_job
} //To get the latest build number and add it to the branch.
```



The new multi-job pipeline environment has created a new job based on the creation of new branch in the environment.

The details of the Jenkins file where the tags are created.

```
        stage('Conditional Deploy stage') {
             when {
                    branch 'master'
    }
     steps {
           script {

                       sh  "git remote set-url origin http://root:password@novartis.devops.altimetrik.io:7080/altimetrik-poc/demo-sonar.git"
                            sh "git fetch"
                            sh "git tag >> tag.txt"
                            sh "cat tag.txt"
                            tags = branchtag()
                            sh "git checkout -b ${version}-${tags}"
                       sh  "git push --force origin ${version}-${tags}"


     }
  }
```

The Jenkins file contains the details of the scripts to create a new branch based on the tag from the master branch only.

Note: If the details of the conditional statement is not provided in the above file it abruptly create new branches unlimited.

The details of the tags are stored in a new function.

```
def branchtag() {
    def tag = sh returnStdout: true, script: "head -1 tag.txt "
        print tag
    tag
}
```

The above function would get the information of the top most tag available on the the repository . which is called in the Jenkins file.

The END.