

Verizon

DevOps Setup Guide

Devops Setup steps for Jenkins, JIRA and Stash

Subramanian, Raja
[Pick the date]

JIRA

JIRA acts as our Test case management and Issue tracking tool.

Add-ons:

The following add-ons are required in JIRA for our Framework.

- Zephyr
- ZAPI
- Bob Swift CLI

Note : Admin access is required in JIRA for installing these add-ons.

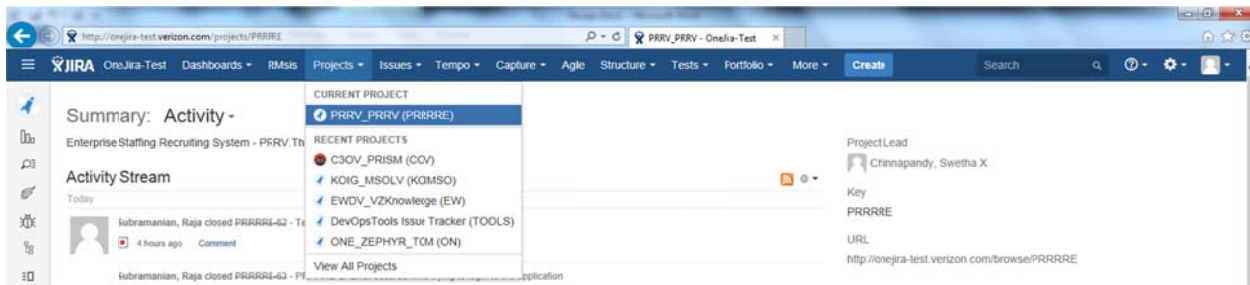
Test Cases in JIRA

Once Zephyr plugin is installed in JIRA we can create new test cases in JIRA. There are two options to add new test cases to JIRA as described below:

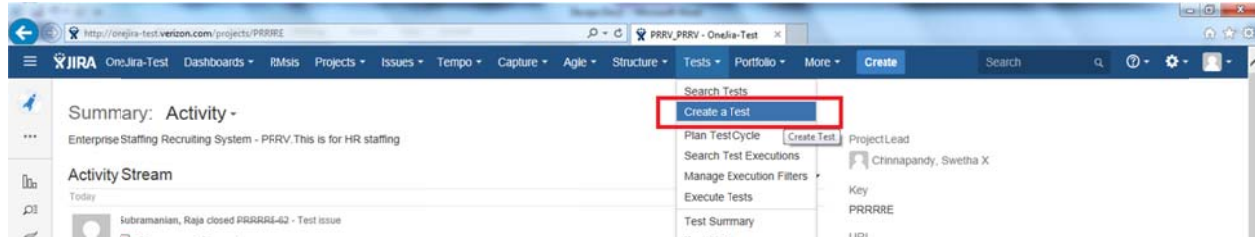
Option 1:

Test cases can be created in JIRA manually using JIRA UI.

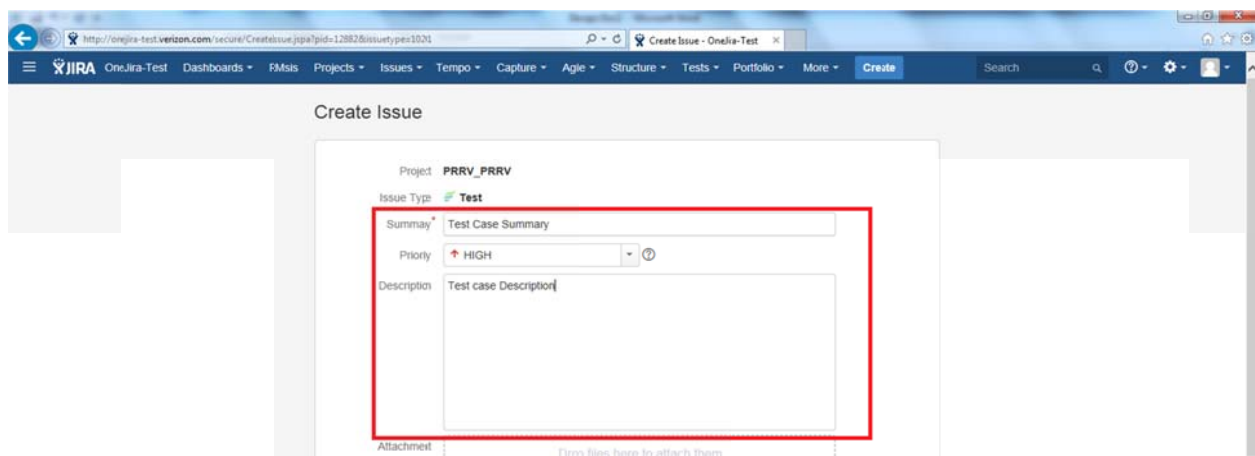
- Login to onejira-test and navigate to the Specific project.



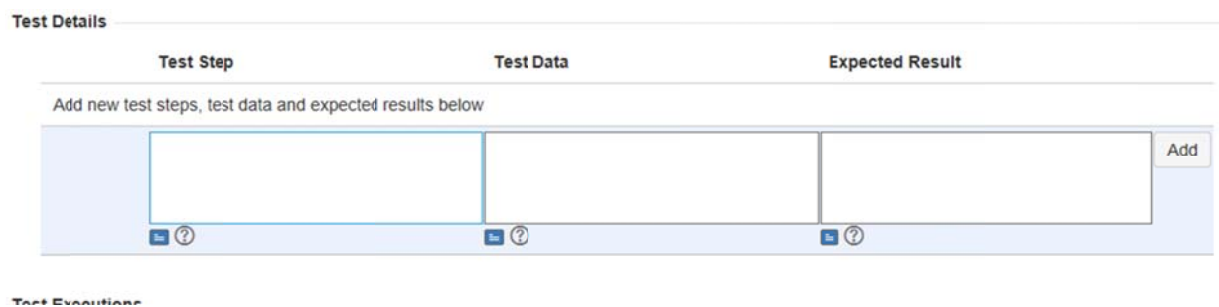
- Click on “Tests” tab and select “Create a Test”.



- Provide Summary, Priority and Description for the test case and click create.



- After the test case is create add test steps to the created test case.



This process of creating test case is manual and can be used only if we have few number of test cases to add.

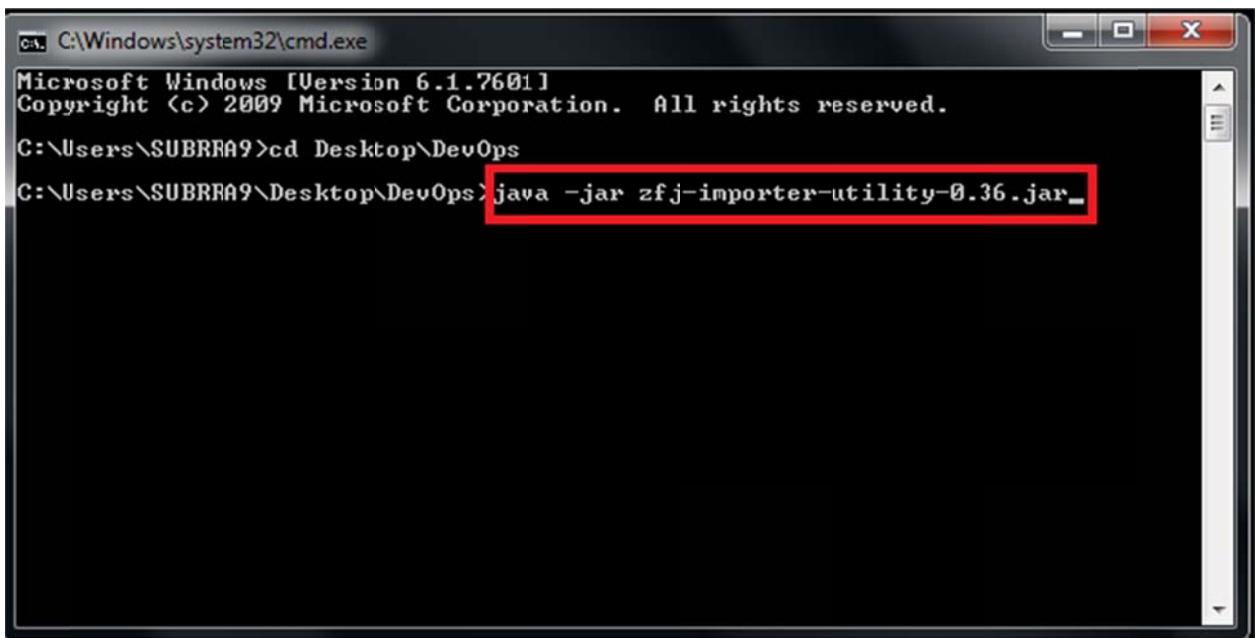
Note 1: Labels are provided in JIRA test cases to group the test cases. For example Msolv test cases KOIMSO_1 to KOIMSO_11 are grouped inside a folder GUI_TestCases. So the label for all these test cases are “GUI_TestCases”.

Note 2: The class name of the test cases in your selenium scripts should be the same as the JIRA test case id. JIRA test case id's has a “-” in it and it cannot be used as class name in java. So replace the “-” with “_” in the class name. Example: Class name of the test case with id “PRRRRE-71” will be “PRRRRE_71”

Option 2:

Test cases can be imported to JIRA from excel sheet using zfy-importer.

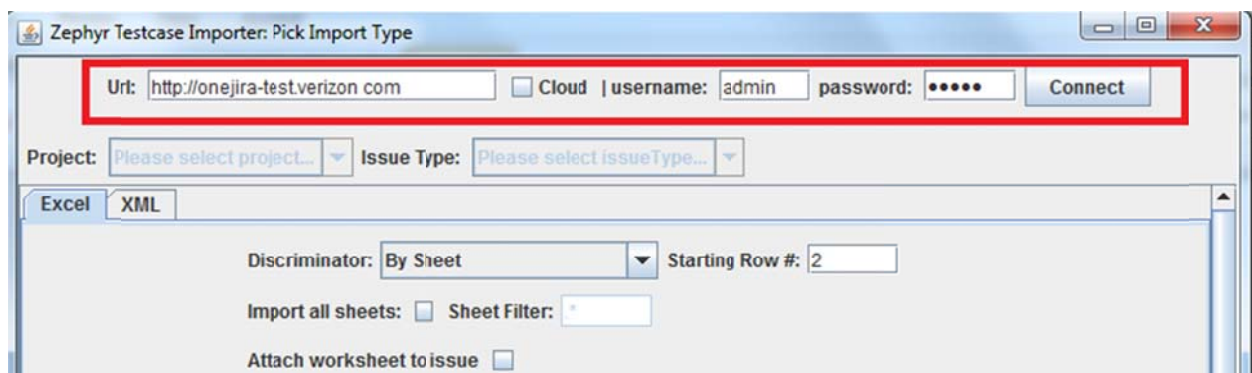
- Download the zfy-importer jar file.
- Execute the jar file using java on a command prompt.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\SUBBRA9>cd Desktop\DevOps
C:\Users\SUBBRA9\Desktop\DevOps>java -jar zfy-importer-utility-0.36.jar_
```

- Fill in JIRA url, username and password on the importer console and click connect.



Zephyr Testcase Importer: Pick Import Type

Url: ☐ Cloud | username: password:

Project: Issue Type:

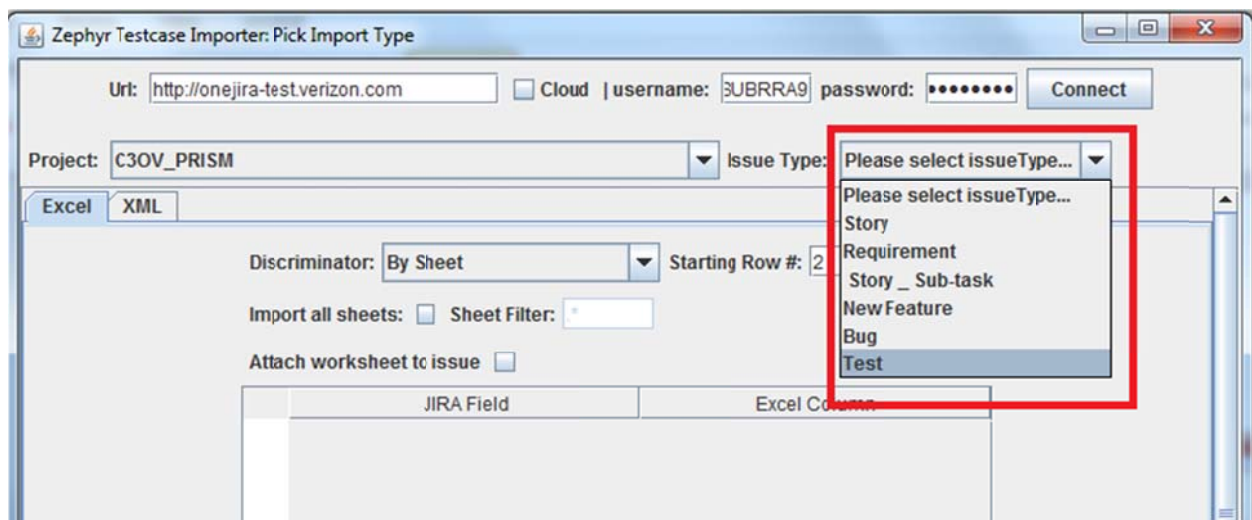
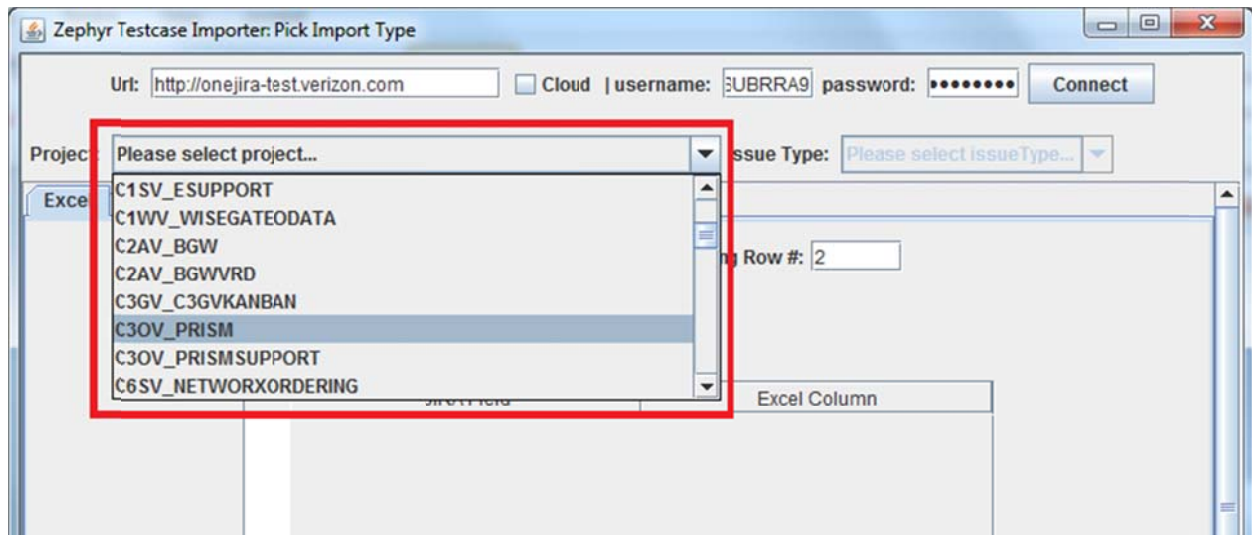
Excel XML

Discriminator: Starting Row #:

Import all sheets: ☐ Sheet Filter:

Attach worksheet to issue ☐

- After the connection is established select the project from drop down and Issue Type as Test.



- Then prepare an excel sheet with the test case summary, description and test data.

	A	B	C	D	E	F	G
1	S.NO	Summary	Description	Test Steps	Test Data	Expected Results	
2	1	Test 1	This is Test 1	Step1	Data1	Result1	
3				Step2	Data2	Result2	
4				Step3	Data3	Result3	
5							
6	2	Test 2	This is Test 2	Step1	Data1	Result1	
7				Step2	Data2	Result2	
8				Step3	Data3	Result3	
9							

- Enter the appropriate Excel Columns as JIRA Fields in the importer console.

Excel XML

Discriminator: By Empty Row Starting Row #: 2

Import all sheets: ☐ Sheet Filter: *

Attach worksheet to issue ☐

JIRA Field	Excel Column
Name *	B
Step *	D
Result *	F
Testdata *	E
ExternalId *	A
Labels	
Comments	
Versions	
Components	
Priority	
Assignee	
Description	C
Due Date	

- Now click on Pick import file button and upload the excel sheet. Then click Start Import.

Assignee

Description C

Due Date

sm\Test Case Prism\Consumption_ONT.xls

Pick Import File Pick Import Folder | ☒ Start Import

Once the import completes all the test cases in excel sheet will be available in JIRA.

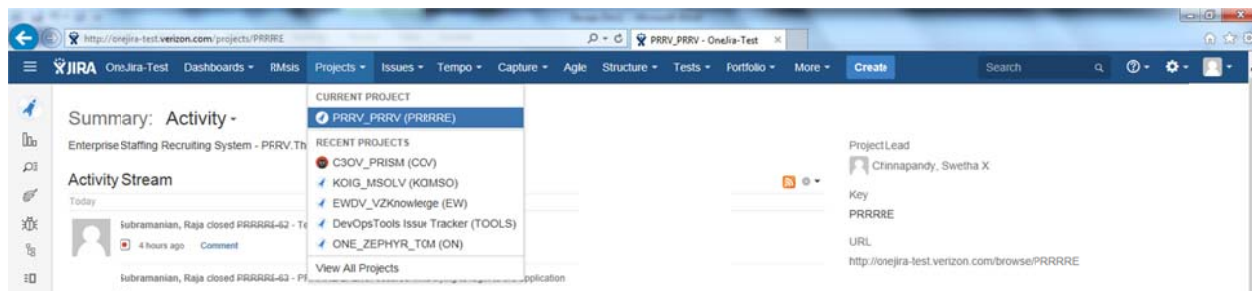
Note 1: Labels are provided in JIRA test cases to group the test cases. For example Msolv test cases KOIMSO_1 to KOIMSO_11 are grouped inside a folder GUI_TestCases. So the label for all these test cases are "GUI_TestCases".

Note 2: The class name of the test cases in your selenium scripts should be the same as the JIRA test case id. JIRA test case id's has a "-" in it and it cannot be used as class name in java. So replace the "-" with "_" in the class name. Example: Class name of the test case with id "PRRRRE-71" will be "PRRRRE_71"

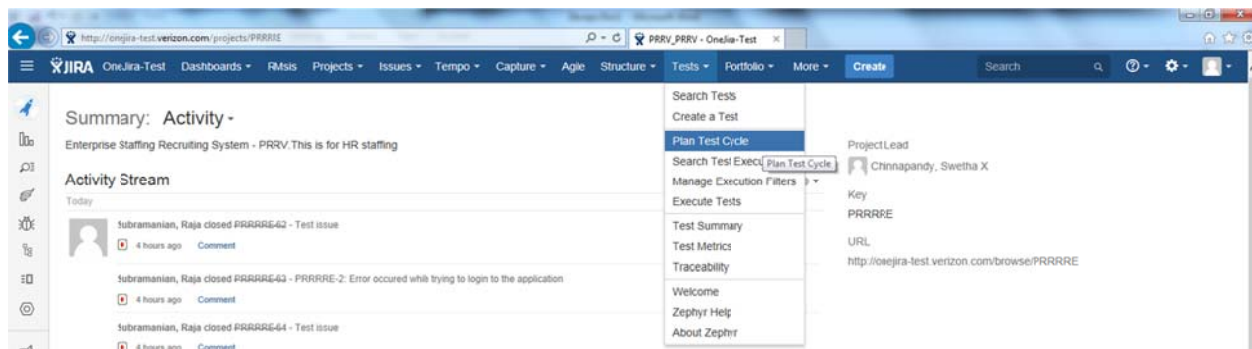
Test Cycles in JIRA

Test cycles can be created in JIRA by following the below steps.

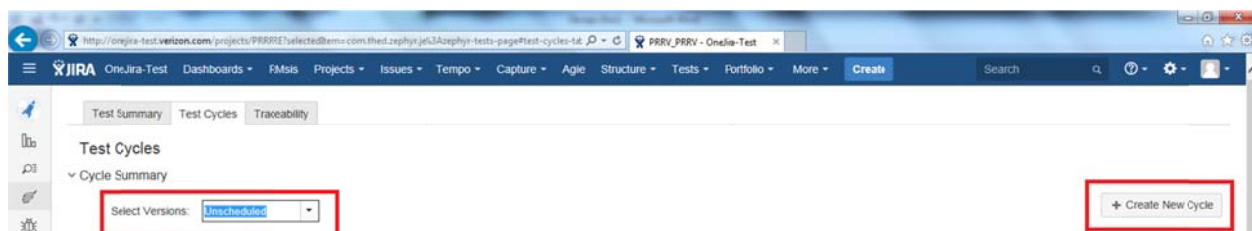
- Login to onejira-test and navigate to the Specific project.



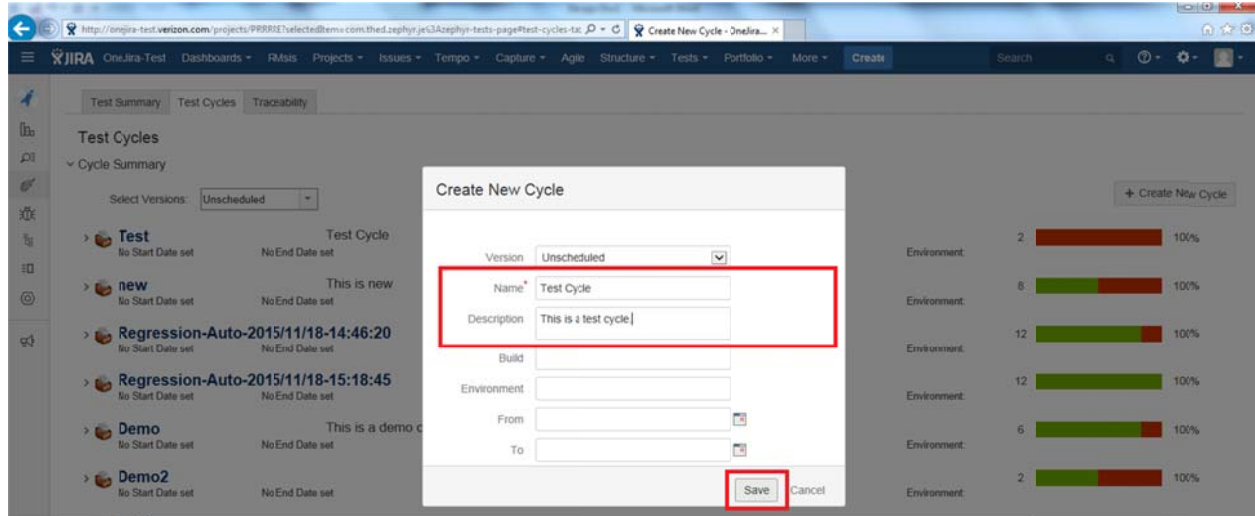
- Click on "Tests" tab and select "Plan Test Cycle".



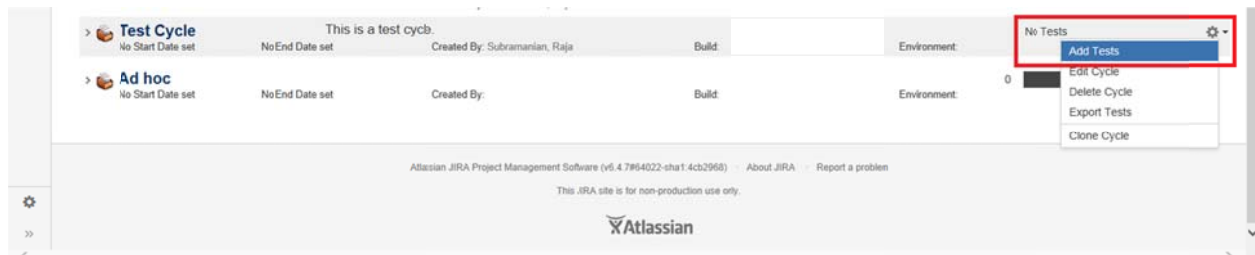
- In Test Cycles page choose "Select Versions" as "Unscheduled" from the drop down and click "Create New Cycle".



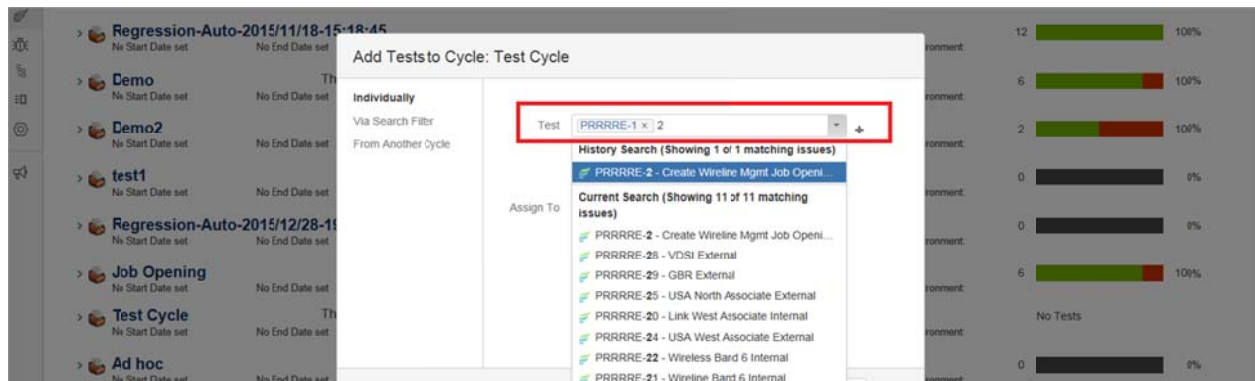
- Provide Name, description and click save.



- After the test cycle is created click on settings button near the test case name and select “Add Tests”



- On the Add tests pop up type the test case id or description to add test case to the cycle.



- After adding test cases to the test cycle it will look like below.

Test Cycle

No Start Date set

No End Date set

Created By: Subramanian, Raja

Build:

Environment:

00%

This is a test cycle.

ID

Status

See [Click to view test execution in Test Cycle](#)

Defect

Component

Label

Executed By

Executed On

PRRRRE-1

UN-EXECUTED

Create Wireless Job Opening - New Headcount

Create_Job_Opening

PRRRRE-2

UN-EXECUTED

Create Wireline Mgm Job Opening - New Headcount

Create_Job_Opening

PRRRRE-3

UN-EXECUTED

Create Associate JobOpening - New Headcount

Create_Job_Opening

PRRRRE-4

UN-EXECUTED

Create International Job Opening - New Headcount

Create_Job_Opening

Ad hoc

No Start Date set

No End Date set

Created By:

Build:

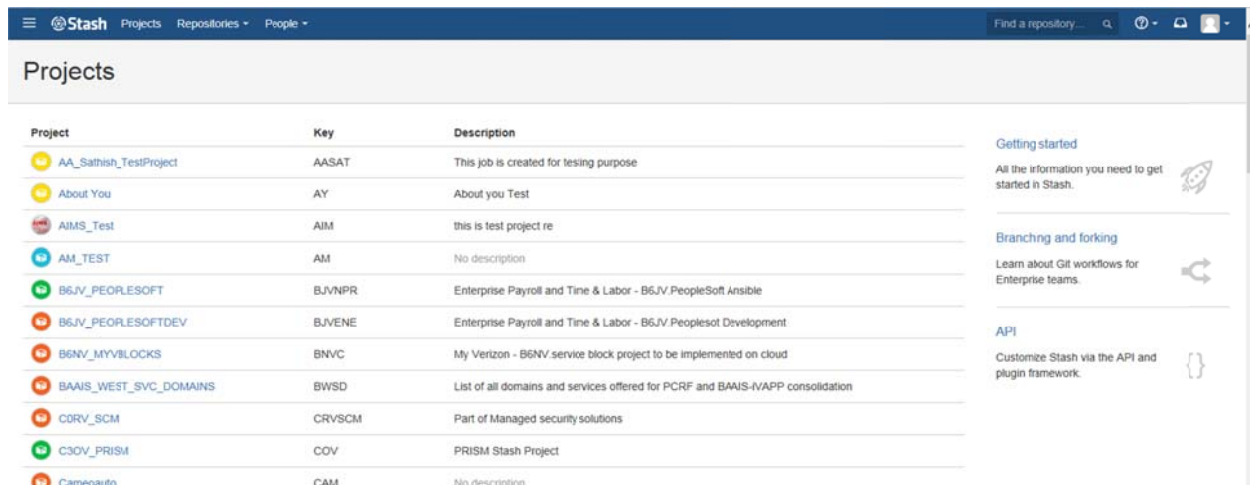
Environment:

00%

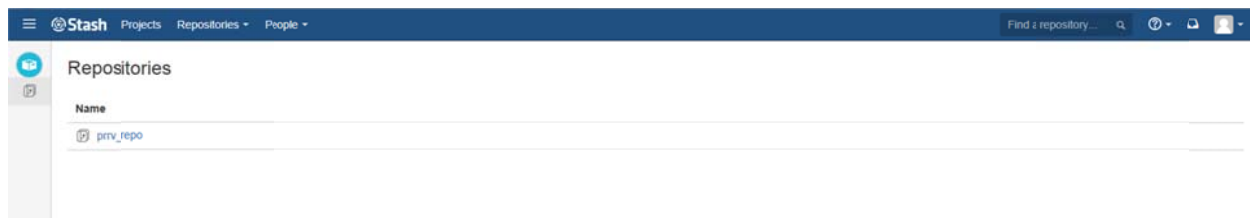
Stash

Atlassian Stash is used as our repository to store Selenium automation code. Stash is [Git](#) repository software. It provides a web interface.

- The starting page shows all the projects, on which the current user has access.



- The project page lists all the repositories belonging to the project.



Git client can be used to upload code to stash repository.

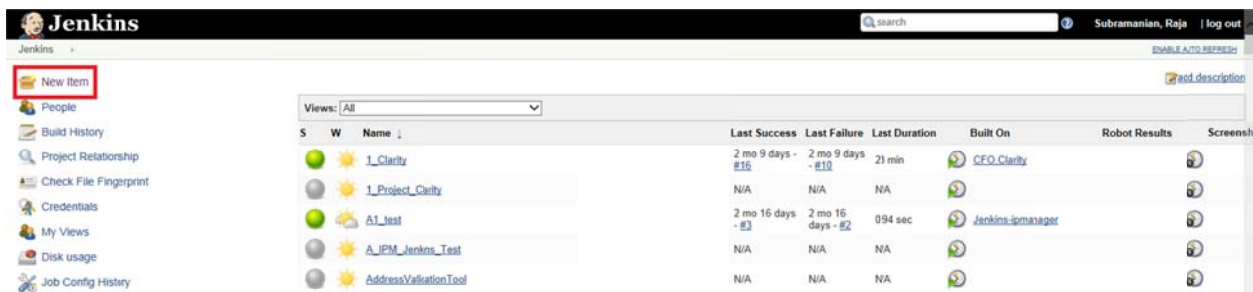
Jenkins

Jenkins is our Continuous Integration tool. We need to create two jobs in Jenkins for our Devops Framework.

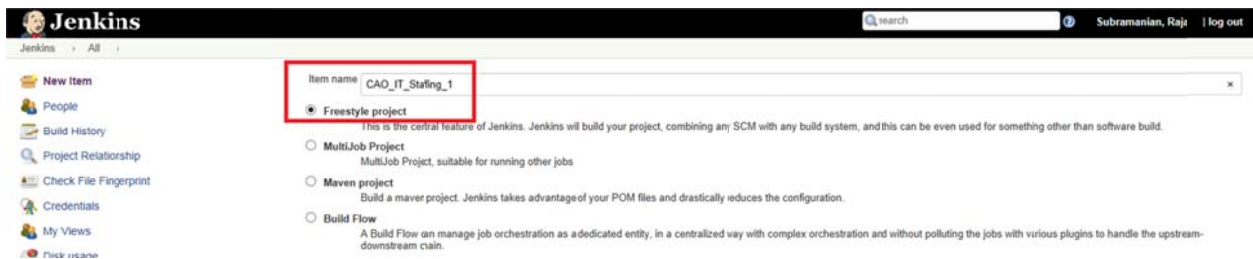
Job 1:

This job is responsible for creating test cycle in JIRA(If required) and generate testng xml file with the test cases to be executed.

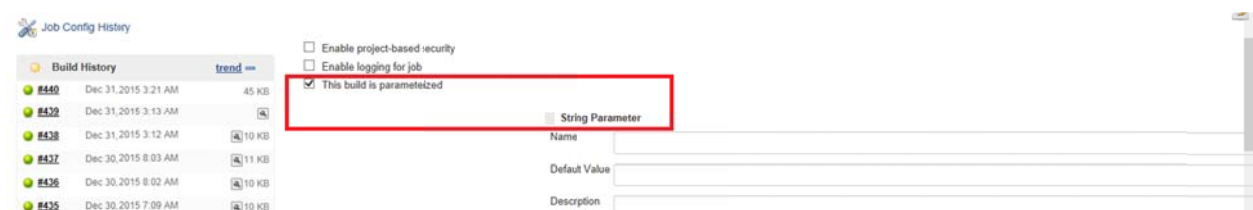
- Login to onejenkins test and click on “New item”.



- Then provide a name for the job, select “Freestyle project” radio button and click ok.



- Once the job is created click on “This build is parameterized” and select String parameter from dropdown.



- As this is a parameterized build, type in the string parameter Name, Default Value and Description.

The screenshot shows the Jenkins configuration page. On the left, there is a 'Build History' table with columns for build number, time, and size. The main area is for configuring a 'String Parameter'. The 'Name' field is 'CycleName', the 'Default Value' is 'None', and the 'Description' is 'Please enter the cycle name created in JIRA. Will be ignored if build triggered by deployment job.' The 'Enable logging for job' checkbox is checked, and 'This build is parameterized' is also checked. There are 'Plain text' and 'Preview' tabs, and a 'Delete' button at the bottom right.

Build History	trend
#88	Dec 30, 10:15 5:09 AM 10 KB
#87	Dec 29, 10:15 11:27 AM 10 KB
#86	Dec 29, 10:15 8:38 AM 10 KB
#85	Dec 29, 10:15 7:32 AM 17 KB
#84	Dec 29, 10:15 7:15 AM 10 KB
#83	Dec 29, 10:15 6:47 AM 10 KB
#82	Dec 29, 10:15 6:42 AM 10 KB
#81	Dec 29, 10:15 6:38 AM 10 KB
#80	Dec 28, 10:15 10:41 AM 17 KB
#79	Dec 28, 10:15 9:43 AM 17 KB
#78	Dec 28, 10:15 9:08 AM 17 KB

- Then scroll down to select the slave on which the job is to be executed.

The screenshot shows the Jenkins configuration page. The 'Restrict where this project can be run' checkbox is checked. The 'Label Expression' field is 'CAO.VSC.ILD.01'. The 'Slaves in label' field is '1'. There are 'Advanced Project Options' and 'Source Code Management' sections. An 'Advanced...' button is visible on the right.

- On "Add build Step" drop down select "Execute Shell" and provide the python command to execute xml_generator.py script.

The screenshot shows the Jenkins configuration page. The 'Build' section is expanded. The 'Execute shell' option is selected. The 'Command' field contains the command 'python xml_generator.py "\$CycleName"'. There is a link to 'See the list of available environment variables'.

- Next is to configure post build action to trigger Job 2 automatically if this job executes without any issue.

The screenshot shows the Jenkins configuration page. The 'Post-build Actions' section is expanded. The 'Trigger parameterized build on other projects' checkbox is checked. The 'Build Triggers' section shows 'Projects to build' as 'CAO.IT.Staffing_2', 'Trigger when build is' as 'Stable', and 'Trigger build without parameters' as unchecked. The 'Parameters from properties file' section shows 'Use properties from file' as 'env.properties' and 'Don't trigger if any files are missing' as unchecked. The 'Predefined parameters' section shows 'Parameters' as 'TESTNG_XML_FILE=\$(WORKSPACE)testing.xml'. There are 'Advanced...' and 'Delete' buttons on the right.

Here env.properties is a property file which stores the cycle_name parameter which can be used by the next job. TESTNG_XML_FILE is the environment variable to pass the testng xml file generated in this job to the next job.

- Once the configurations are done click on save button



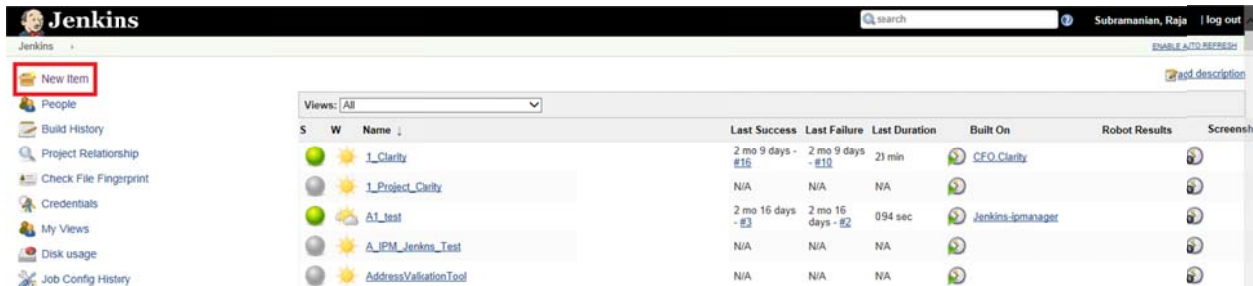
This completes the created of first Jenkins job.

Note: Python script xml_generator.py should be placed inside the workspace of this build before executing the build.

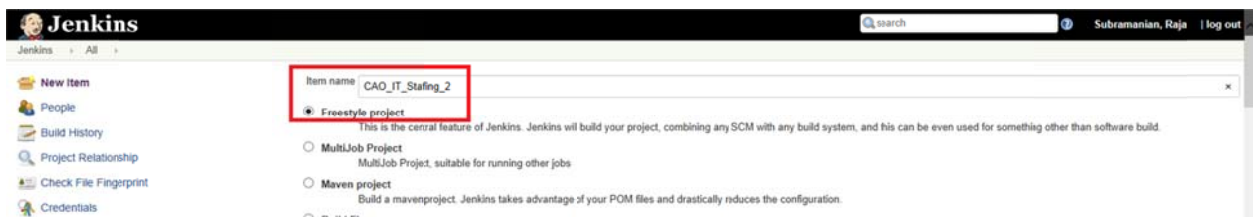
Job 2:

This job is responsible for pulling the code from stash, executing the automation tests and reporting back the results to JIRA. It also creates issues in JIRA in case of test case failure.

- Login to onejenkins test and click on “New item”.



- Then provide a name for the job, select “Freestyle project” radio button and click ok.



- Once the job is created, select the slave on which the job is to be executed.



- Under Source Code Management select git and provide repository url & credentials for downloading the selenium automation source code.



- On “Add build Step” drop down select “Execute Shell” and provide the shell commands to clean previous execution results, copy the testng xml file from previous job, selenium script execution through ant and finally python command to execute Update_result.py script.



Build

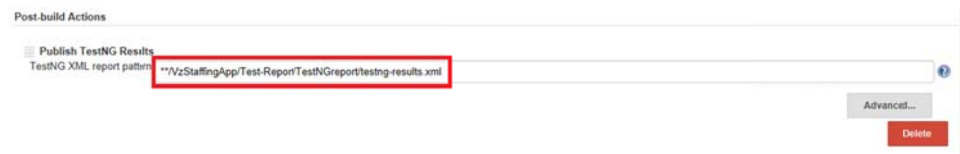
☐ Execute shell

Command

```
rm -rf VzStaffingApp/reports/Screenshots/*
rm -rf VzStaffingApp/reports/*.html
cp $(TESTING_XML_FILE) VzStaffingApp/.
cd VzStaffingApp
export ANT_HOME=/apps/opt/mw/apache-ant-1.9.5
path=/apps/opt/mw/apache-ant-1.9.5/bin:$PATH
/apps/opt/mw/apache-ant-1.9.5/bin/ant -d testng-execution
cd ..
cp VzStaffingApp/Test-Report/TestReport/testng-results.xml .
python Update_result.py "$cycle_name"
```

[See the list of available environment variables](#)

- Then on “Add post-build Action” drop down select “Publish TestNG result” and provide the testng-result xml file path.



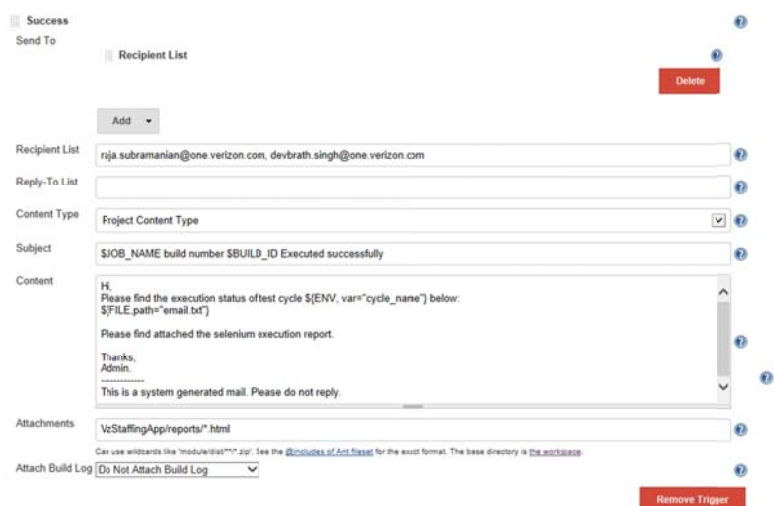
Post-build Actions

☐ Publish TestNG Results

TestNG XML report pattern

****/VzStaffingApp/Test-Report/TestReport/testng-results.xml**

Also select “Editable Email Notification” from the drop down and configure email notification for Success and unstable builds by adding trigger. The email should have content and attachments.



☐ Success

Send To

☐ Recipient List

Recipient List

rja.subramanian@one.verizon.com, devbrath.singh@one.verizon.com

Reply-To list

Content Type

Project Content Type

Subject

\$JOB_NAME build number \$BUILD_ID Executed successfully

Content

H,

Please find the execution status of test cycle \$(ENV, var="cycle_name") below:

\$FILE_PATH"email.txt"

Please find attached the selenium execution report.

Thanks,

Admin.

This is a system generated mail. Please do not reply.

Attachments

VzStaffingApp/reports/*.html

Can use wildcards like "module/dist/*". See the [@includes of Ant Regex](#) for the exact format. The base directory is the workspace.

Attach Build Log

☐ Unstable (Test Failures)

Send To ☐ Recipient List

Recipient List:

Reply-To List:

Content Type:

Subject:

Content:

Attachments:

Attach Build Log:

Note : The content for the email is obtained from a text file (email.txt), which is generated by the Update_result.py python script.

JIRA CLI Client:

For the Update_result.py script to create and update issues in JIRA, we need JIRA CLI Client to be configured in Jenkins Workspace of Job 2. The steps to configure JIRA CLI is as follows.

- Download the client software for Bob Swift CLI add on in JIRA from the following location <https://bobswift.atlassian.net/wiki/display/ACLI/Downloads>.

fisheye-cli-5.0.0-distribution.zip	2015-10-31 22:40:03
hipchat-cli-5.0.0-distribution.zip	2015-10-31 22:39:54
jira-cli-5.0.0-distribution.zip	2015-10-31 22:39:46
atlassian-cli-4.5.0-distribution.zip	2015-09-16 17:52:05
bamboo-cli-4.5.0-distribution.zip	2015-09-16 17:51:38
confluence-cli-4.5.0-distribution.zip	2015-09-16 17:51:26
crowd-cli-4.5.0-distribution.zip	2015-09-16 17:51:10

- Extract the zip file and copy atlassian-cli-4.5.0 folder to the Workspace of Jenkins Job 2.

- Navigate inside the folder and Edit jira.sh file. Note: Modify jira.bat file if you are using the CLI on a Windows machine.
- Add the Base URL of JIRA along with username and password inside jira.sh file as shown below.

```
#!/bin/bash
# Comments
# - Customize for your installation, for instance you might want to add default parameters like the following:
# java -jar "dirname $0"/lib/jira-cli-4.5.0.jar --server https://my.example.com --user automation --password automation "##"
java -jar "dirname $0"/lib/jira-cli-4.5.0.jar --server https://onmlira-test-version.com --user Username --password Password "##"
```

This completes the JIRA CLI Client configuration.

Note: JIRA CLI requires java 1.7 or above to execute.

Python Scripts

There are two python scripts involved in our framework. Both the scripts works with python version 2.6.

Xml_generator.py

This script is responsible for generating testng xml file based on the input. It accepts one string parameter for test cycle name.

- If the cycle name is provided (Ex: python xml_generator.py "Demo Cycle"), then the python script will find out the test cases available in the "Demo Cycle" and generates a testng xml file.
- If the cycle name is not provided (Ex: python xml_generator.py "None"), then the python script will create a test cycle in JIRA (Ex: [Regression-Auto-2015/12/23-23:01:09](#)) and adds all the test cases in the project to that test cycle. Then a testng xml file is generated for the newly created test cycle.

The script is made as generic as possible. Please find the lines that are to be changed if used for another project.

- Project name variable
project_name = "KOIG_MSOLV"
- En_user variable which is used for API authentication
en_user = "U1VCUIJBOTpNYXRyaXhAODU="

Note : The user name and password is base64 encrypted. Can be done using the following link. <https://www.base64encode.org/>

Encode to Base64 format

Simply use the form below

username:password

> ENCODE <

UTF-8



(You may also select output charset.)

dXNlcm5hbWU6cGFzc3dvcmQ=

- Base url parameter for jira url.
baseUrl = 'http://onejira-test.verizon.com'
 - Parallel execution of tests in selenium hub will require increase in the threadcount parameter number.
root = Element('suite', name=cycle_name, parallel="tests", threadcount="1")
 - Class path of the selenium test cases
node3 = SubElement(node2, 'class',
name="msol."+test_labels[test_names[i]][0]+"."+class_names[i])
- Ex :

```
<classes>
```

```
<class name="msol.GUI_TestCases.KOIMSO_1"/>
```

```
</classes>
```

Note : Labels are provided in JIRA test cases to group the test cases. For example Msolv test cases KOIMSO_1 to KOIMSO_11 are grouped inside a folder GUI_TestCases. So the label for all these test cases are "GUI_TestCases". This label field is also substituted in the python script above to get the proper class path "msol.GUI_TestCases.KOIMSO_1".

Update_result.py

This script is responsible for reading the testng-result xml file and updating the test case results back to test cycle in JIRA. It also creates issue in JIRA in case of test case failure. It uses a combination of API and CLI to create and update issues.

Issue Creation Logic:

- ✓ If a test case passes and an open issue linked to the test case is already available. The issue will be closed.
- ✓ If a test case fails and an open issue linked to the test case is already available. The same issue will be updated with the latest failure.
- ✓ If a test case fails and no issue is available for the ticket. A new issue will be created and linked to this test case.

The script is also made as generic as possible. Please find the lines that are to be changed if used for another project.

- Project name variable
project_name = "KOIG_MSOLV"
 - En_user variable which is used for API authentication
en_user = "U1VCUIJBOTpNYXRyaXhAODU="
- Note : The user name and password is base64 encrypted. Can be done using the following link. <https://www.base64encode.org/>

Encode to Base64 format

Simply use the form below

username:password

> ENCODE <

UTF-8



(You may also select output charset.)

dXNlcm5hbWU6cGFzc3dvcmQ=

- Base url parameter for jira url.
baseUrl = 'http://onejira-test.verizon.com'
- Screen shot location from where the image file is to be uploaded
file_out = os.popen("ls Test_Report/ScreenShots/*"+i+"*.png")