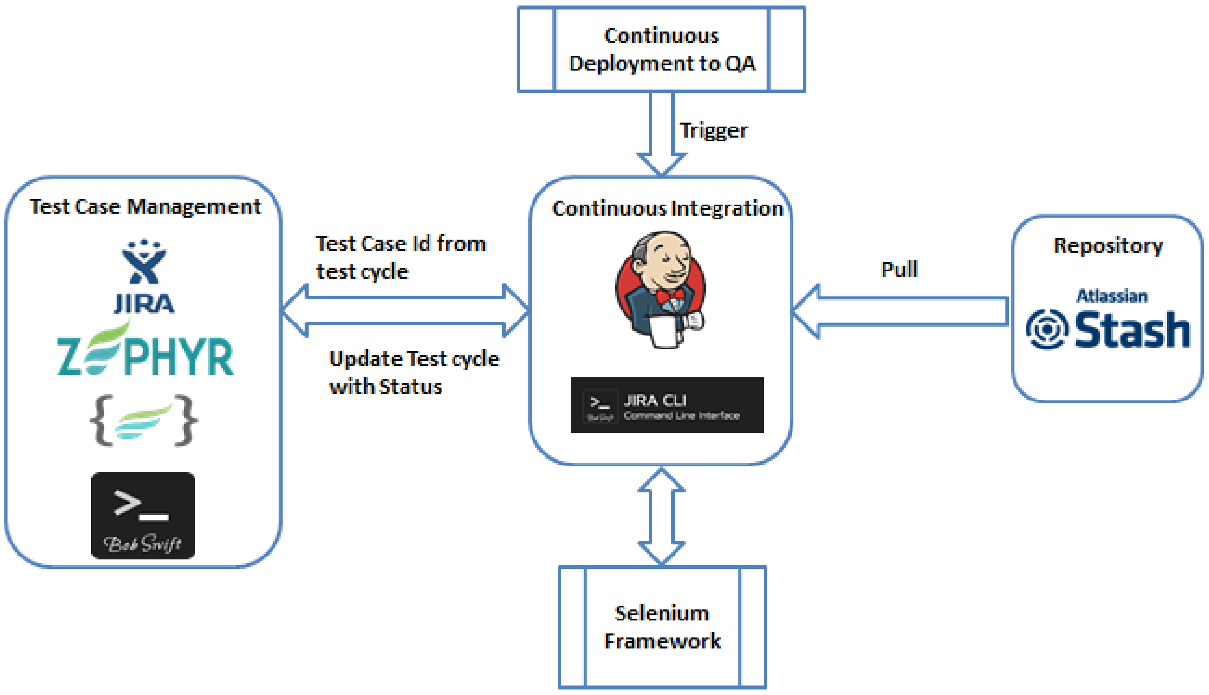
# ***QA Automation Framework***

gquotient

## Use Case:

End-to-end automation of QA process, where once the testcase and corresponding test scripts has been written  the automation framework takes care of running the scripts, updating the test cases with the status and create issues when a bug has been found.



## Toolsets:

***Testing Framework***: **Selenium *Grid*** along with **TestNG** has been used.

***Test Case management suite***: **JIRA** along with the below plugins has been used.

**Zephyr**:  Converts JIRA to a Test case management tool

**ZAPI**:  Provides API interface for Zephyr plugin

**Bob Swift**:  JIRA CLI plugin

***Source Code Repository***:  **Stash** has been used as central repo for storing all the automation scripts

***Continuous Integration***:  **Jenkins** along with the below plugins has been used

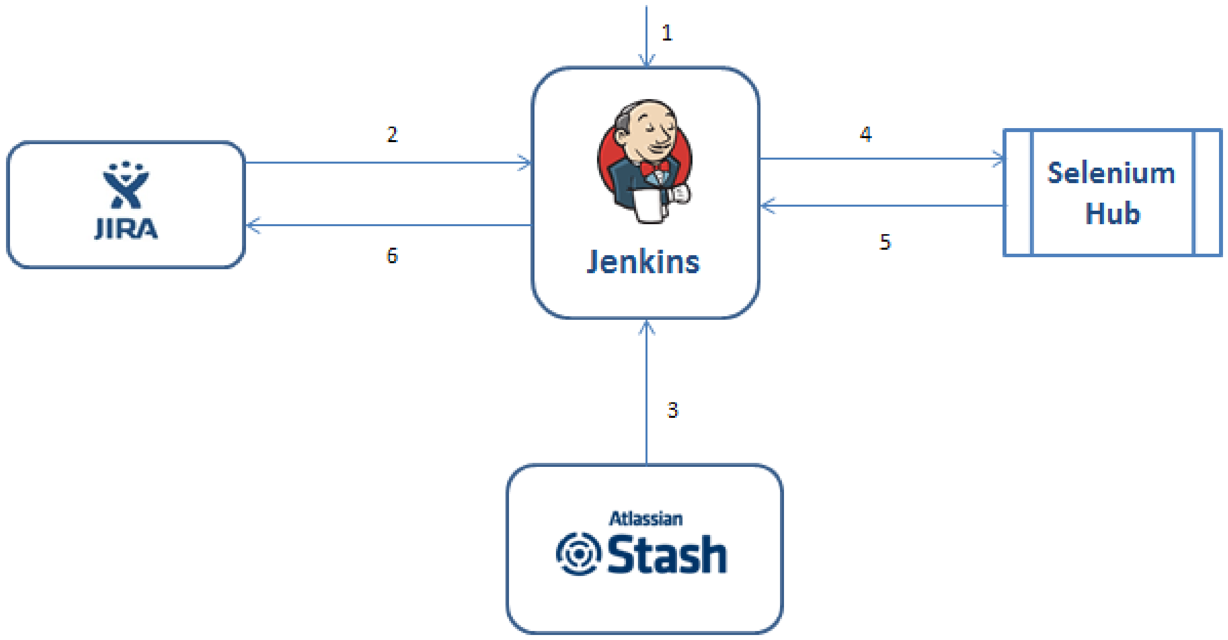
**JIRA CLI Client**: Client application to interact with Bob Swift plugin in JIRA

**GIT**: Used to checkout automation script from Stash

**TestNG**: Publishes results in Graphical format

**Email Notification**: Send execution status and report from Jenkins via email

## Process Flow:



1. Jenkins job for the corresponding project is triggered with cycle name as parameter. This invokes a python script xml_generator.py.
   **Note:** If no cycle ID is provided then the Python script will create a new test cycle for the corresponding project in JIRA.
2. This python script will communicate with JIRA through API calls and find out the test cases added to the test cycle provided. Then a testng xml file is generated with the test cases available in the test cycle.
   **Note:** If no cycle ID then the python script will add all the test case's in that project to newly created cycle id and generate the testng xml file.
3. After the generation of xml file the Jenkins build job will automatically trigger the next Build job. This job will first clone the Selenium Automation code for the specific project from Stash to Jenkins Workspace.

4. Using the testng xml file generated in the previous job the automation scripts will be executed on a Selenium Hub environment from Jenkins.
5. After the execution of the selenium scripts the result xml file is parsed by a python script update_result.py.
6. This script will update the execution status of all the test cases back to JIRA test cycle and will create issues in JIRA for the failed test cases.
   Finally an e-mail will be triggered with consolidated report of the execution.

## JIRA API & CLI Calls

The API and CLI calls used in the above process flow are listed below.

***ZAPI:***

*Project API GET Call* - To find the project ID.

*Cycle API GET Call* -To find the Cycle ID of the test cycle.

*Cycle API POST Call* - To create new Test Cycle.

*Execution API GET Call* – To find test cases added to the Test Cycle.

*Execution API POST Call* – To add test cases to Test cycle

*Search API POST Call* – To find the test cases available in a Project.

*QuickExecute API POST Call* – To update results of test cases in Test Cycle.

***REST API:***

*Issue API GET Call* – To get labels and components of a Test Case.

***CLI:***

*createIssue* – CLI to create new issue.

*AddAttachment* – CLI to add screenshot to the issue.

*LinkIssue* – CLI to link issue with test case.

*GetIssueList* – CLI to search available issues linked to the test case.

*AddComment* – CLI to update comment to an existing issue.

*TransitionIssue* – CLI to close an open issue.

gquotient