**Swagger POC**

**What is this all about?**

* This POC demonstrates the usage of swagger-diff module in Jenkins to check the breaking api changes.
* This also describes a solution to combine bitbucket and Jenkins for automated builds of pull requests and reporting back the status to Stash.

**Requirement**

* The requirement was to compare the swagger specification files from pull request branch with the target branch, whenever there is a pull request.
* If the swagger specification differs, then the build should fail along with the declination of the pull request in bitbucket.
* If the swagger specifications are same, then the build should be successful along with the automatic merging.

**Workflow**

* Following components were used to build this flow

1. Pull request notification plugin in bitbucket.
2. Git plugin in Jenkins.
3. Jenkins text finder plugin in Jenkins.
4. PostBuildScript plugin in Jenkins.
5. Bitbucket cli.

* The idea was to trigger the job and pass the parameters from bitbucket to Jenkins job using pull request notification plugin, whenever there is a pull request.
* The moment, Jenkins job is triggered after the opening and re-opening of pull request, parameters will be passed to the Jenkins job, after which a groovy system script will run before the scm runs.
* This groovy script will help to set the build description with the pull request id and link back to bitbucket pull request.
* After this, the source of the pull request shall be automatically merged with the target branch (This will help us to have both swagger specification files from source and target in same workspace so that we can run swagger-diff module on them).
* Jenkins text finder plugin will help to fail the build, if there is a keyword found, “unmatched difference” , which implies that the swagger specification file differs.
* In order to approve a pull request we require at least one successful Jenkins build. Thereby we would like to get not only the build result of the code checked in for the pull request but get the build status after the code has been merged with the target branch.
* Only if the build was successful and the number of successful builds configured in Stash is reached the pull request shall be approved and merged.

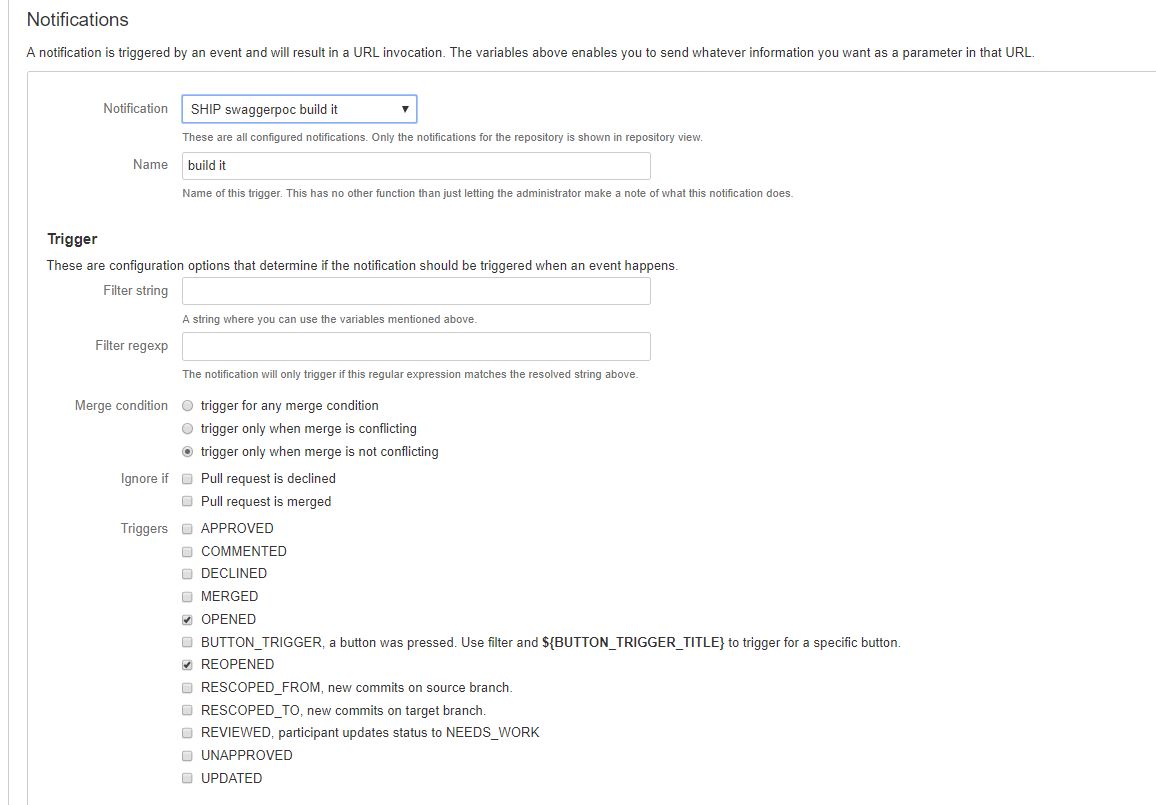
**Step-by-Step Configuration**

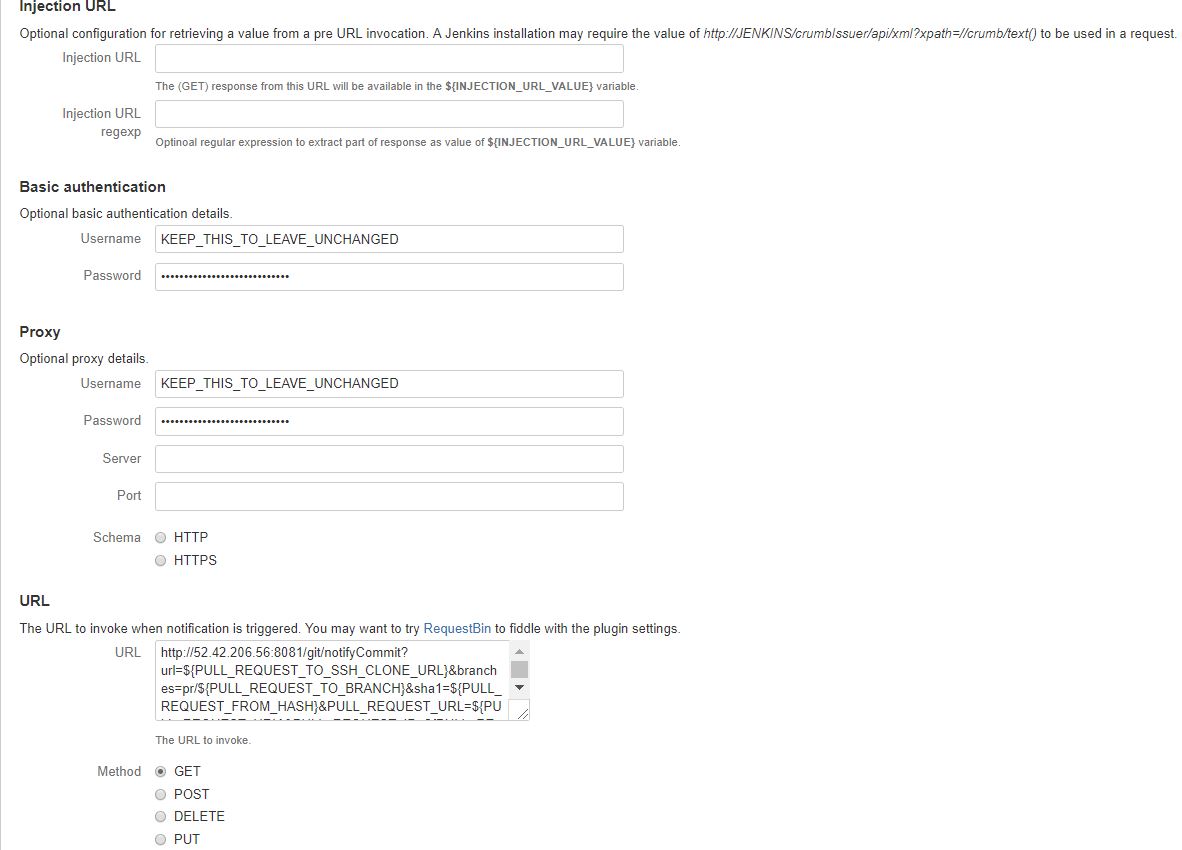
Here are the steps necessary to achieve the above mentioned workflow.

* Install and configure Pull Request Notifier for bitbucket.
* Install bitbucket cli on Jenkins server.
* Configure Pull Request Configuration in bitbucket.
* Install and configure Jenkins Git Plugin.
* Install and configure Jenkins Stash Notifier Plugin.
* Create/Update Jenkins job

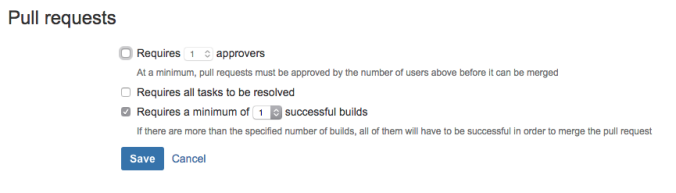
**Install and configure Pull Request Notifier for Stash**

* Install the Pull Request Notifier for Stash add-on via the Universal Plugin Manager or manually by downloading from the Atlassian Marketplace.
* Choose Administration > Manage Add-ons > Pull Request Notifier > Configure to configure the plugin.
* Select the following triggers. They will trigger a new build when a pull request is opened, reopened.
* OPENED
* REOPENED
* Enter Jenkins and proxy credentials as required
* Choose GET as HTTP method
* Enter [http://34.217.72.1:8081/git/notifyCommit?url=${PULL\_REQUEST\_TO\_SSH\_CLONE\_URL}&branches=pr/${PULL\_REQUEST\_TO\_BRANCH}&sha1=${PULL\_REQUEST\_FROM\_HASH}&PULL\_REQUEST\_URL=${PULL\_REQUEST\_URL}&PULL\_REQUEST\_ID=${PULL\_REQUEST\_ID}&PULL\_REQUEST\_FROM\_REPO\_PROJECT\_KEY=${PULL\_REQUEST\_FROM\_REPO\_PROJECT\_KEY}&PULL\_REQUEST\_FROM\_REPO\_SLUG=${PULL\_REQUEST\_FROM\_REPO\_SLUG}](http://34.217.72.1:8081/git/notifyCommit?url=$%7bPULL_REQUEST_TO_SSH_CLONE_URL%7d&branches=pr/$%7bPULL_REQUEST_TO_BRANCH%7d&sha1=$%7bPULL_REQUEST_FROM_HASH%7d&PULL_REQUEST_URL=$%7bPULL_REQUEST_URL%7d&PULL_REQUEST_ID=$%7bPULL_REQUEST_ID%7d&PULL_REQUEST_FROM_REPO_PROJECT_KEY=$%7bPULL_REQUEST_FROM_REPO_PROJECT_KEY%7d&PULL_REQUEST_FROM_REPO_SLUG=$%7bPULL_REQUEST_FROM_REPO_SLUG%7d) as the URL to invoke. The trick here is the pr prefix in the branch name which is then used also in the branch name in the Jenkins job configuration. By this we can differentiate between regular commits to the target branch and the pull requests and can have two jobs in parallel. This notification will trigger automatically the correct builds using the Push Notification Support of the Jenkins Git Plugin based on the target branch and automatically merges the pull request with the target branch. Please adjust the Jenkins URL (http://34.217.72.1:8081) to match with your environment.





**Configure Pull Request Configuration in Bitbucket**

* Choose in your Stash repository Settings > Pull requests and select the Requires a minimum of successful builds and enter the number of builds. [](https://christiangalsterer.files.wordpress.com/2015/04/jenkins_stash_pr_stash_pull_request_configuration.png)

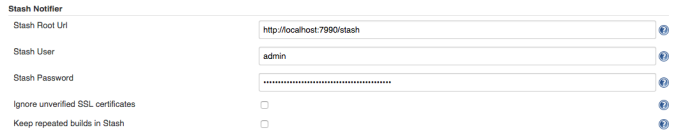
**Install and configure Jenkins Git Plugin**

* Install the Jenkins Git Plugin via Jenkins > Manage Jenkins > Manage Plugins.
* Perform all standard configuration so that you can access Stash via SSH and can pull repositories from Stash.
* Using git repository url as ssh, was necessary to pass the parameters from bitbucket to Jenkins.

#### Install and configure Jenkins Pre SCM Buildstep Plugin

* Install the [Pre SCM Buildstep Plugin](https://wiki.jenkins-ci.org/display/JENKINS/pre-scm-buildstep) via **Jenkins** > **Manage Jenkins** > **Manage Plugins**
* Perform all standard configuration so that you can access Stash via SSH and can pull repositories from Stash.

#### Install and configure Jenkins Stash Notifier Plugin

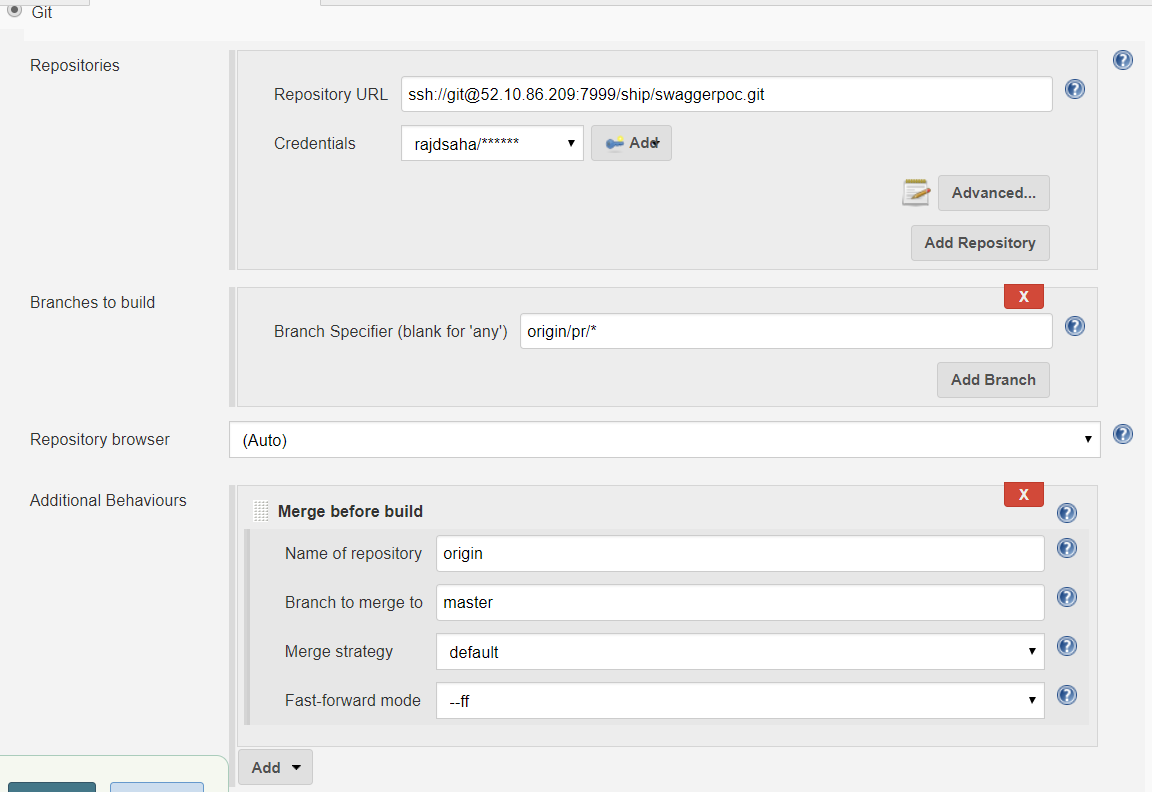
1. Install the [Jenkins Stash Notifier Plugin](https://wiki.jenkins-ci.org/display/JENKINS/StashNotifier+Plugin) via **Jenkins** > **Manage Jenkins** > **Manage Plugin**
2. Choose **Jenkins** > **Configure System** > **Stash Notifier** and enter the data for your Stash installation. [](https://christiangalsterer.files.wordpress.com/2015/04/jenkins_stash_pr_jenkins_stash_notifier_configuration1.png)

**Install and configure Jenkins Groovy Plugin**

* Install the Jenkins Groovy Plugin via Jenkins > Manage Jenkins > Manage Plugins

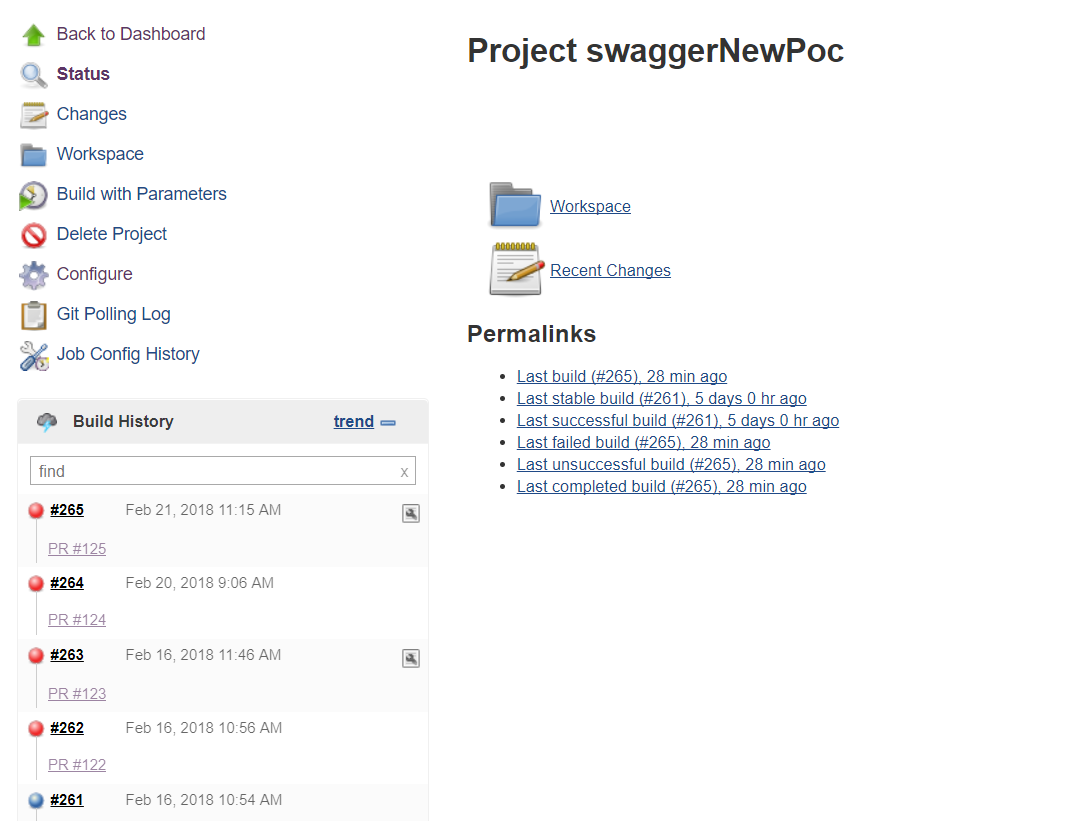
**Create Jenkins job to Build the Pull Request**

* Add the Git configuration for your Jenkins job as follows.

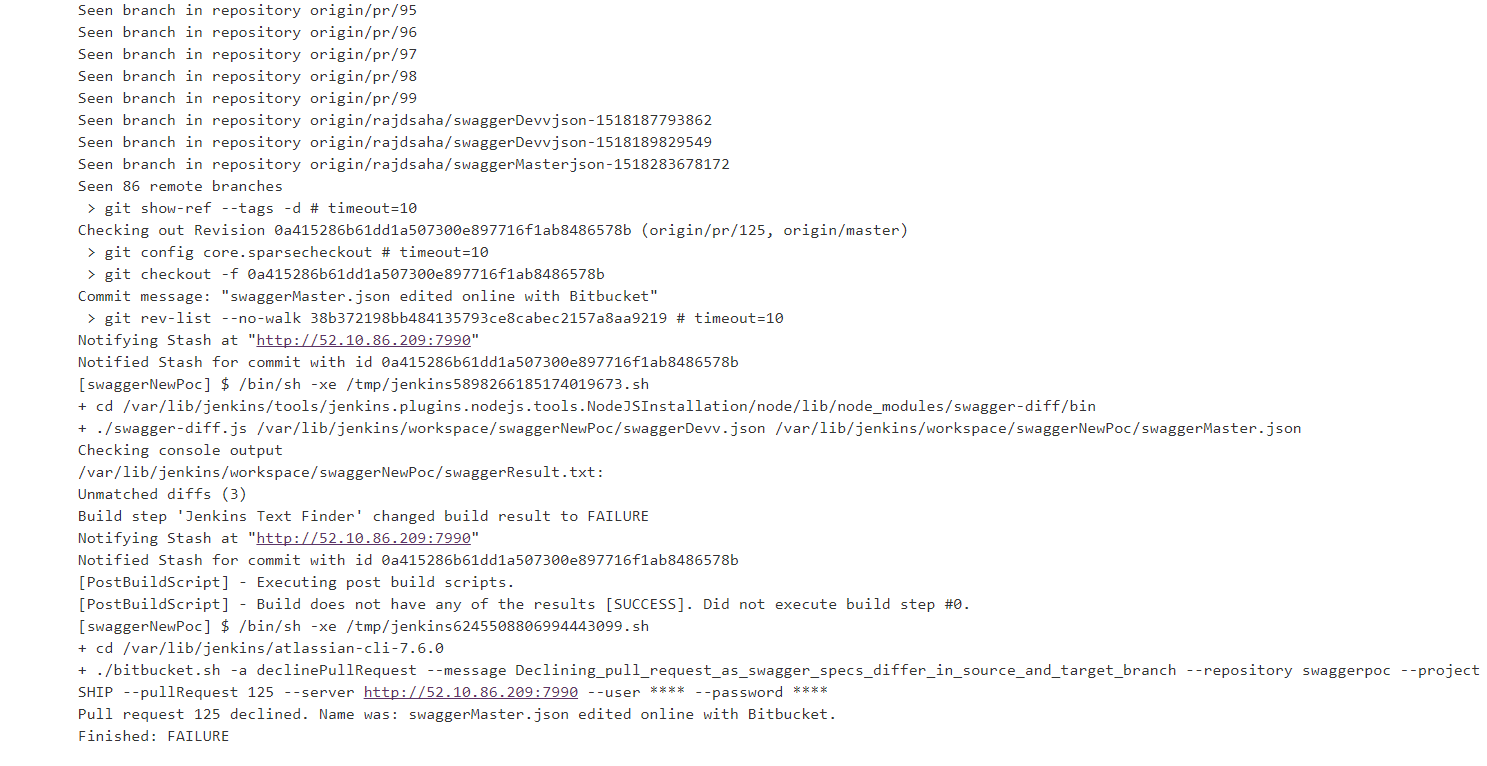


* Select Git in the Source Code Management section
* Update the Repository URL to point to your Stash installation.
* Update Branch Specifier (blank for ‘any’) and Branch to merge to to the branch name which should be build and act as the target branch. If the target branch is master, then the values should be entered in the format as shown in the following picture. Please note the pr prefix in the Branch Specifier (blank for ‘any’) field, which we have used in the URL in the Pull Request Notification Plugin configuration in a previous step jenkins\_stash\_pr\_jenkins\_job\_git\_configuration
* Set the build description with the pull request ID and a link back to the Stash pull request.
* Select in the Build Environment section the Run buildstep before SCM runs and add an Execute system Groovy script step using the following script:
* def currentBuild = Thread.currentThread().executable
* def PULL\_REQUEST\_URL = build.buildVariableResolver.resolve('PULL\_REQUEST\_URL')
* def PULL\_REQUEST\_ID = build.buildVariableResolver.resolve('PULL\_REQUEST\_ID')
* def description = "<a href='$PULL\_REQUEST\_URL'>PR #$PULL\_REQUEST\_ID</a>"
* currentBuild.setDescription(description)
* This will set add pull request ID in format PR #ID, e.g. PR #1 with a link back to the Stash pull request. Here we use the new capabilities of the Jenkins Git Plugin to pass additional parameters in the commit notification, here PULL\_REQUEST\_ID and PULL\_REQUEST\_URL.
* Create/update the Stash notification configuration of your Jenkins job(s) as follows
* Add in the Post-build Actions section the Notify Stash Instance post build action and configure. No further configuration is necessary as we configured the Stash parameters as system wide settings already in step Install and configure Jenkins Stash Notifier Plugin.jenkins\_stash\_pr\_jenkins\_job\_stash\_notifier\_configuration
* Execute shell script was performed during the build to run the swagger-diff module on swagger specification files.
* Post build shell script was used to run the bitbucket cli, so that we can automatically merge or decline the pull request on the basis of success or failure of build.
* Save job configuration.

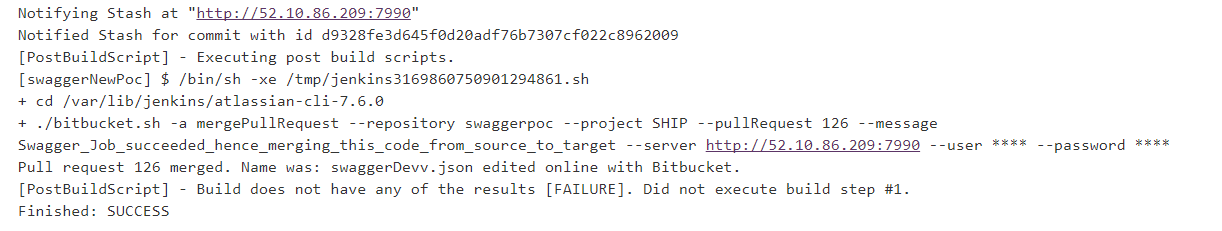
When everything is setup correctly and you have created your first pull request in Stash you should see a new build triggered in Jenkins along with the ID of the pull request in the build description.



**Console output of Jenkins job when build is failing and pull request is being declined**



**Console output of Jenkins job when build is success and pull request is being merged**



**Important note**

* Due to security reasons, parameters need to be white listed at the Jenkins Master.
* Special startup parameters:
* To make the parameters passable from the commit notify to the jobs the parameters need to be added to the start-up parameters of Jenkins this is to do with security changes in Jenkins.
* “-Dhudson.plugins.git.GitStatus.safeParameters= PULL\_REQUEST\_URL, PULL\_REQUEST\_ID, PULL\_REQUEST\_FROM\_HASH, PULL\_REQUEST\_TO\_BRANCH”
* https://wiki.jenkins-ci.org/display/JENKINS/Features+controlled+by+system+properties
* For windows this can be done using the Jenkins.xml in the Jenkins home DIR. These need to be added before the -jar term as parameters after this are ignored.
* Ubuntu:
* File: /etc/default/Jenkins
* Property: JAVA\_ARGS
* RedHat:
* File: /etc/sysconfig/Jenkins
* Property: JENKINS\_JAVA\_OPTIONS