|  |  |
| --- | --- |
| Payments SECURITY - 3DS 1.0  CONTINUOUS INTEGRATION & CONTINUOUS DEPLOYMENT PROCESS | Version 2.0  Date: June 2018 |

Contents

# 1.Introduction

# 2.Prerequisites

## 3.Download Redis cache

## 4.Installation of Redis Cache

## *5.Configure Master and Slave Nodes*

## *6.Start Server*

## *7.Create Cluster using built-in Ruby script*

## *8.Add Slaves*

# Introduction

* Redis is an in-memory key-value store known for its flexibility, performance, and wide language support
* Redis Cache – Redis Caching is a good way to reduce database server’s load and improve its performance.

# 2.0 Prerequisites

* **apt-get update** updates the list of available packages and their versions, but it does not install or upgrade any packages.
* **apt-get upgrade** actually installs newer versions of the packages on the system.

## **3.0 Download Redis cache**

## Automated Deployment Process adopted by 3DS 1.0

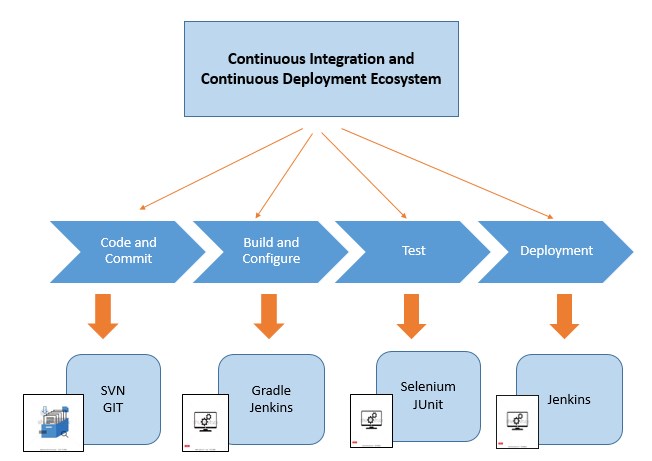


Fig.1 CI-CD system

The above illustration shows the progressions of CI-CD system and the tools involved.

## Jenkins Build process – Orchestration Layer

Fig.2 Build Cycle

* Jenkin job is created, and codebase is checked out from centralized repository using version control tools i.e. SVN/GIT. (RUPAY codebase is maintained under GIT Hub and Visa/ Master codebase is maintained under SVN repository).

Sequential steps for new Jenkin Job configuration are shown in the below Jenkin’s snapshots.

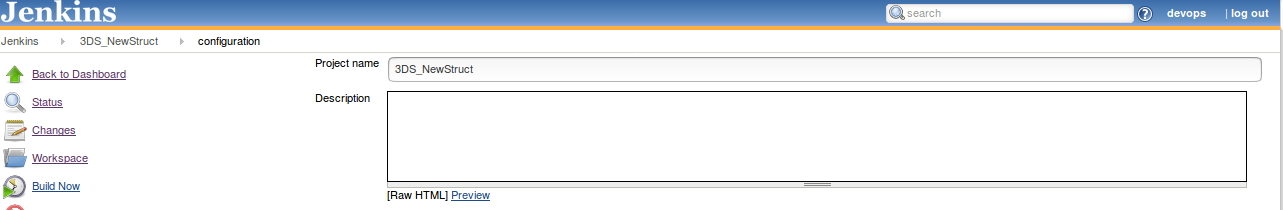


Fig.3 Project name for Build Jenkin Job.

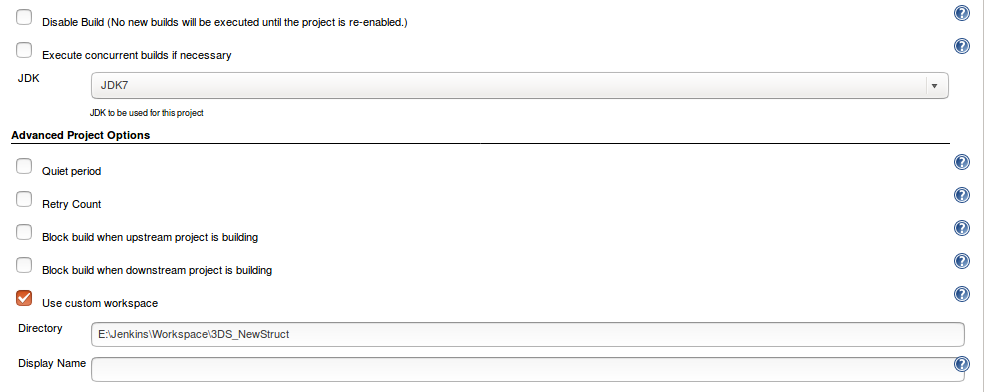


Fig.4 Configuration of Java version and Jenkin Directory.

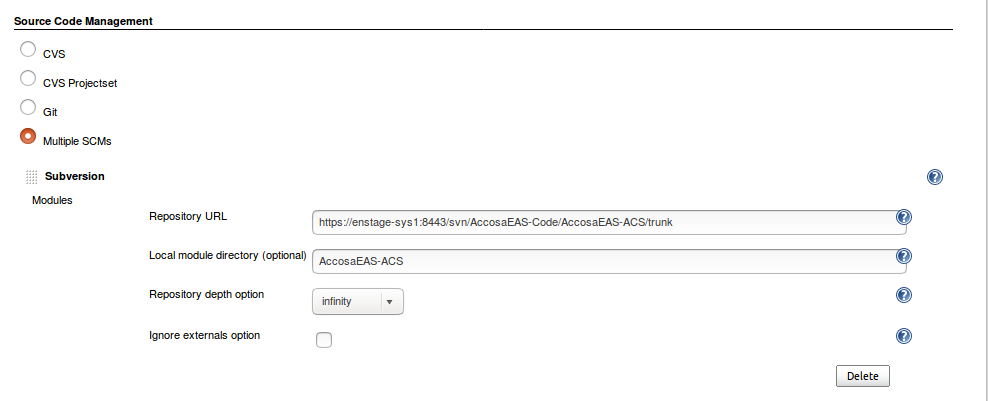


Fig.5 Configuration of the Version Control Repository – SVN.

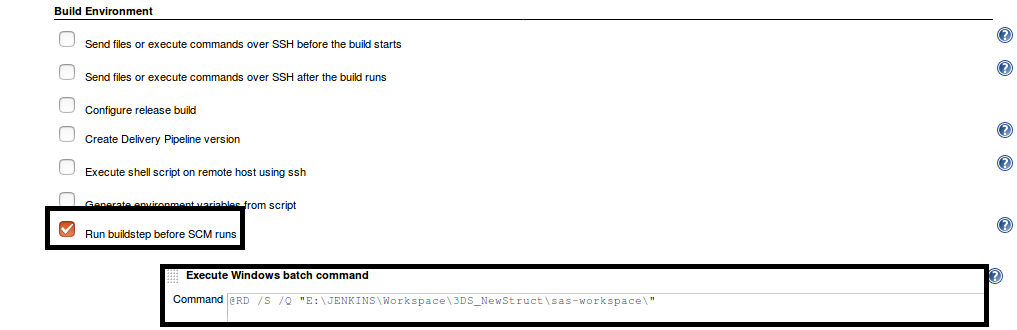


Fig.6 Command for removal of old SAS workspace.

* Gradle script is invoked.
* The SAS-Workspace cleanup operation is performed i.e. the existing SAS-Workspace is deleted.
* Symlink command is run and all the new codebase modules are linked to newly created SAS-Work space.
* Gradle is used to Build the codebase and version accurately. Example sas-2018-01-22-16-12.4.0.zip.

Snapshots of Gradle invoke are sequentially shown below

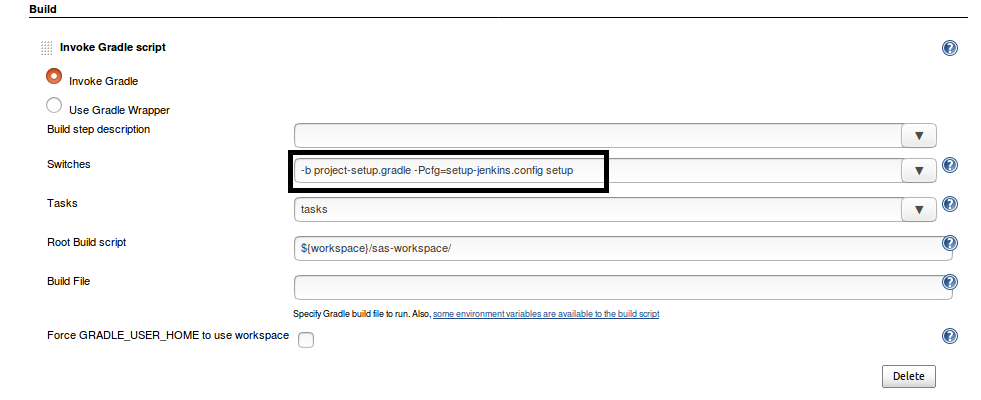


Fig.7 Configuration of Symlink Command.

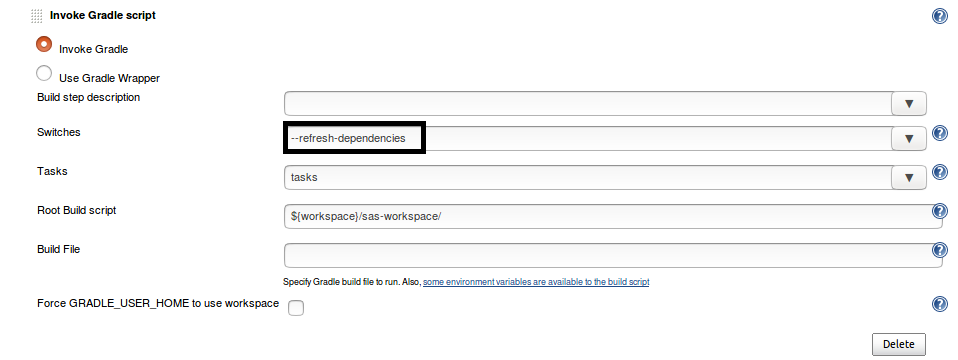


Fig.8 Refreshing of Build dependencies.

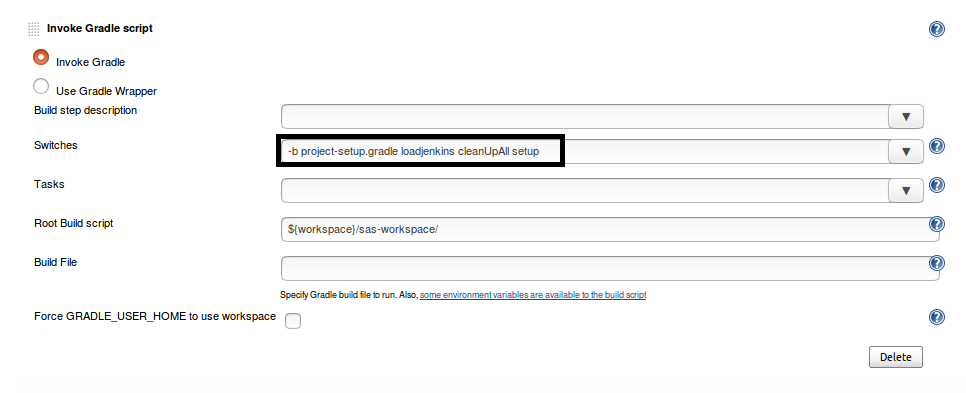


Fig.9 Cleanup of old SAS-workspace.

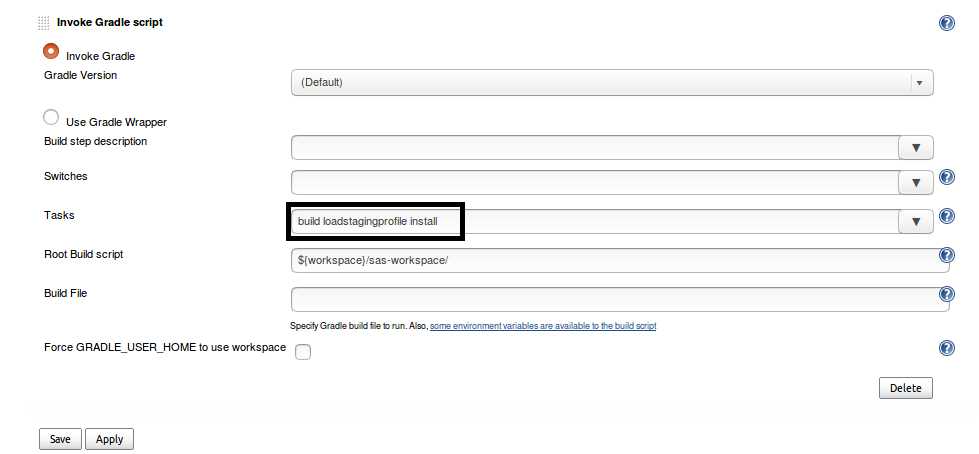


Fig.10 Build invoke.

* The SAS-zip file created during the Build process is moved to the Artifacts location from the SAS-Workspace and this is automated using Perl Scripting as shown below:



Fig.11 Perl script

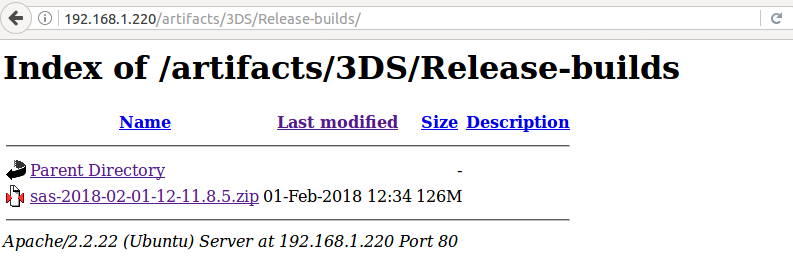


Fig.12 SAS-zip file of release build in the Artifacts location.

**Below are the Snapshots for the configuration of GIT Hub Repository**

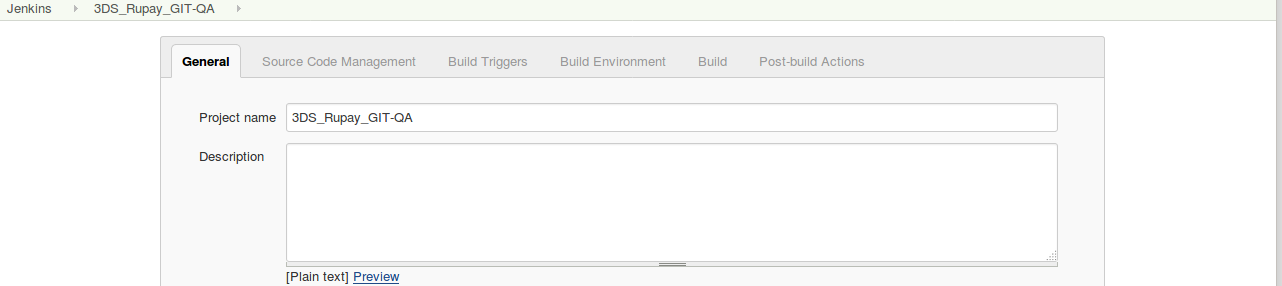


Fig.13 Project Name for Build Jenkin Job.

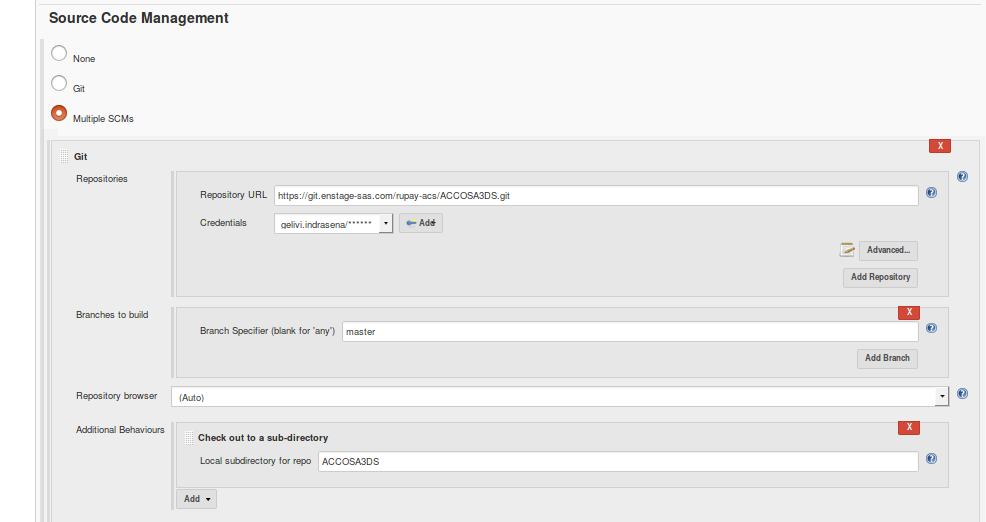


Fig.14 Configuration of the Version Control Repository – GIT Hub.

## Deployment process – Shell Script (Execution Layer)

Fig. 15 Deployment Cycle

* Deployment process is automated using Shell scripting.
* New Jenkin Job is created for the Deployment Process.
* The Shell script is executed, and the following tasks take place:
  + Configures the bank specific application servers.
  + From the Artifacts location SAS-zip file is downloaded into the configured application. server Extracts the SAS-zip file and Stops the Jboss service (Application server).
  + Takes the Backup of the existing Jboss modules and deployment data.
  + Deletes the existing Temp, log and Data folders.
  + Copies the downloaded modules into the configured application server.
  + Starts the application services with newly downloaded codebase modules and deployment data.

Configuration of Shell Script into a Jenkin job is shown in below snapshots

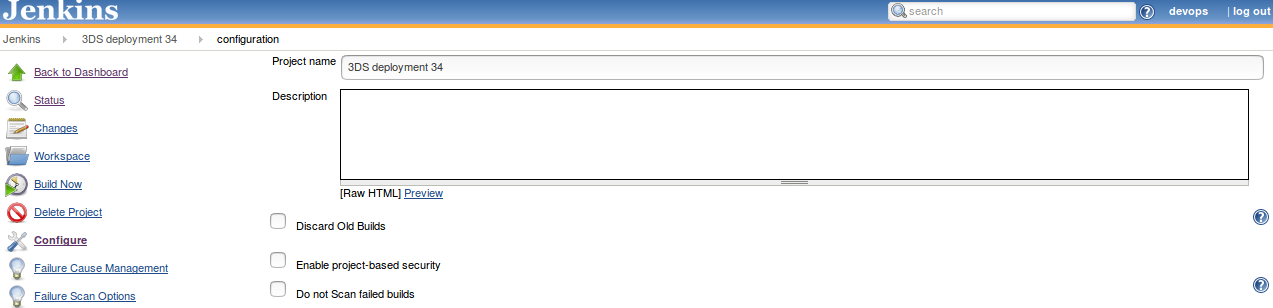


Fig.16 Project Name for Jenkin Job of Deployment.

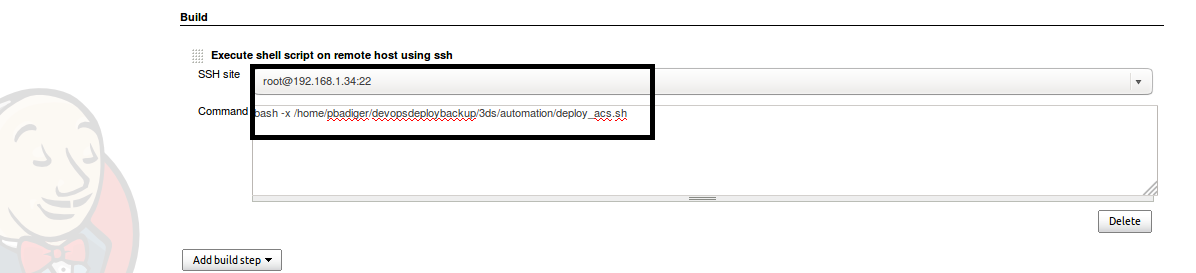


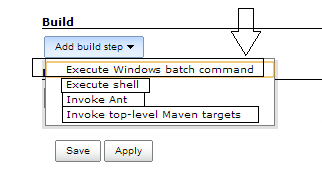
Fig.17 Configuration of Shell script path and server.

# Post Build Actions

* The status of the Build and Deployment is noted.
* All the concerned teams are notified.

# Execute Selenium build using Jenkins

We can execute test cases in Jenkins using 4 ways refer the below screenshot

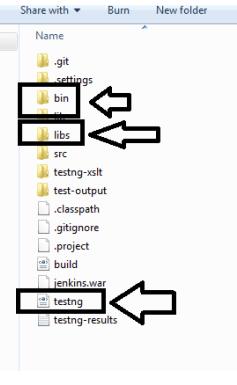


3DS 1.0 will execute using Window batch command

## Step 1- Create a batch file first then we will add the same batch file to Jenkins

*a-To create the batch file we need to set classpath of TestNG so that we can execute testng.xml files*

our project structure should look like



#### b- Open command prompt and set the classpath-

While setting classpath we will set the path of bin folder and libs folder (inside libs we have all libraries)

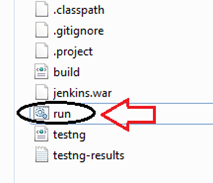
Home directory > set classpath=<Path>\bin;<Path>\libs\\*;

Note- Please make the Path changes as per destination system

#### c- Open notepad and type the below command and save as .bat file –

In 3DS 1.0 case, it has been saved as run.bat

java -cp bin;libs/\* org.testng.TestNG testng.xml



## Step 2-Create a job in Jenkins which will execute our build

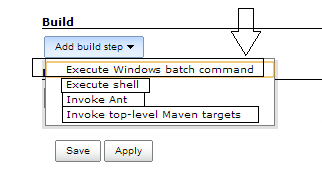
Open Jenkins on browser ( http://192.168.105.99:7777/jenkins)

a- Click on the new item

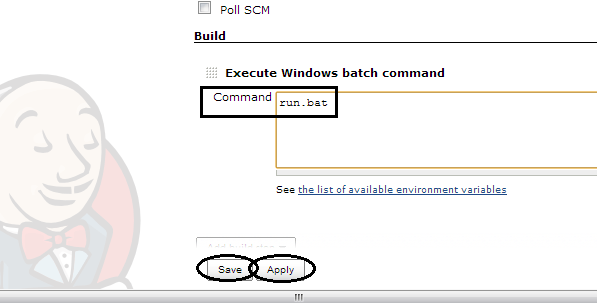
b- Give the Job-Name, select Build a free-style software project and Click on OK  button

c- Navigate to Advanced Project Options > Check the use custom workspace > in directory we will specify the project home directory

d- Important part now specify the Add Build step >Click on Execute Windows batch command

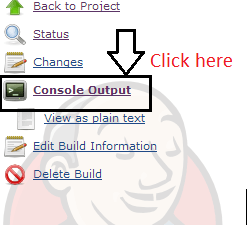


e-In the section please specify the batch file which we created and click on Apply and save



## Step 3- you can finally run the Build > Click on Build now option

## Step 4- Check Build history and Console output and verify the output



# Schedule your build in Jenkins for periodic execution

Jenkins comes with very good functionality in which we can schedule jobs which we created

You can schedule build for existing jobs which already created and while creating new project also we can specify the same.

Let’s schedule the job. Refer the below screenshot

### Step 1- Schedule Configuration

Open job which we created now and Click on configure > select the check box build periodically

### Step 2- Time Slot Configuration

Specify the time here we need to careful about the syntax

Jenkins will accept 5 parameters.

1. Specify minute and range will vary from 0-59

2. Specify hours and range will vary from 0-11

3. Specify day and range will vary from 0-7 here 0 and 7 will be Sunday

4. Specify month and range will vary from 1-12

5. Specify year so here you can specify \*

Example 1- if you specify    00 22 \* \* \* it means your build will run daily @ 10 PM

Example 2- if you specify    50 \* \* \* \* it means your build will run daily  50 min  after every hour

Example 3- if you specify    00 22 1 \* \* it means your build will run every Monday @ 10 PM

# Conclusion

To improve the efficiency and effectiveness of the automation build, deployment and testing process, ACS payment security LOB adapted ways to improve their Devops and test automation. As a solution to help address these issues, we designed CICD process using Jenkins. Using CICD framework, we can earn considerable savings with respect to the people, time, and hardware necessary to perform Devops and QA.