

Changing Cosmic Build System to Bamboo

CERN Summer Student Program

Created by:

Robert Varga

2013 August

Supervisor:

Pablo Saiz

IT-SDC

Contents

Abstract	2
Introduction.....	3
Specification of the work project	3
Used tools in the project	4
Cosmic Build System.....	4
Bamboo CI System.....	5
Selenium.....	6
Implementation.....	6
Created plans in Bamboo	6
Package Creation with Bamboo	9
Package deployment and test running for modules in Bamboo.....	11
Selenium Tests.....	13
Acknowledgement.....	16
References.....	16

Abstract

The aim of this project is to change Cosmic custom build system to an Automated build system used Bamboo CI System services. The goal is when a developer performs some changes on the source code, the system builds installation packages for different architectures and runs tests automatically on the software modules as soon as possible.

The Bamboo build system polls the git repository which is a commonly used source code repository by the developers of the IT department. Bamboo CI System is a widely used system by the department. Thus the project uses widely accepted tools by the department which makes the Cosmic project even more standardized.

Project also aims to create packages for every versions of Cosmic modules for different architectures (SLC5/SLC6) which can be accessed by different package repositories on AFS file system.

The created package repositories can be used for automated deploy environment such as puppet.

Introduction

The Cosmic project for LHC experiments aims to provide a single entry point to the monitoring data collected from the distributed computing systems of the LHC virtual organization.

The Cosmic collects information from multiple sources. It covers various activities of the LHC experiments like job processing, data management, transfer tests, monitoring of the reconstruction at Tier-0, or monitoring of site efficiency. So, the Cosmic monitoring tool provides information from the Grid from different aspects.

The Cosmic is currently working on different Grid flavors (OSG, LCG, gLite). It is deployed for the four LHC experiments ATLAS, ALICE, CMS and LHCb.

Because of the diversity of systems and the widespread use of the Cosmic monitoring tool it is necessary to create a build system which can easily create installation packages for different architectures when a software developer release new versions of Cosmic modules.

Instead of a software developer manually create installation packages using a custom build system it would be more convenient if a build system automatically performs tasks which are create installation packages for each module when the source code has been changed.

To perform automatically build and deploy software modules the IT group uses Bamboo Continuous Integration System as a common solution which is provide a lot of features to customize our jobs or tasks. An advantage of Bamboo it can collaborate with source code repositories (git, svn, etc.) (check for changes of the source code) which are commonly used by the IT group. One other benefit of Bamboo build system that can automatically run tests for the software module and provides information about the status of the tasks and the test results.

Due the highlighted advantages above it is desirable to migrate from the custom build system to the Bamboo CI System.

Specification of the work project

Cosmic monitoring Tool consists of software modules. The developer group have already created a custom build system for the modules based on python distutils and Koji build system to create rpm packages for the software modules. The goal of the project to create rpm packages automatically in a shared file repository and run tests,

when a software developer makes changes on the source code. To perform the package creation for Cosmic modules the work project uses the Bamboo CI System. The Bamboo System polls the git repository for the source code changes periodically. When the changes committed in the source code repository Bamboo creates packages of the latest version of the software module for SLC5 and SLC6 architecture and it copies to a shared package repository on the AFS file system. When the changes committed in the source code repository Bamboo creates packages of the earlier tagged versions of the software module for SLC5 and SLC6 architecture which are not exists already on the AFS package repository (creates rpm packages only for the newly created tags). Jobs in the Bamboo plan creates separate package repositories on AFS for stable and unstable (rc release candidate) tagged versions of the software modules. A Bamboo job deploys the newly created packages one by one: After the package creation, a job deploys Cosmic modules to a Bamboo machine used by the AFS package repositories. After the deployment a Bamboo job run tests for Cosmic modules which are contain Selenium tests. After the module deployment on Bamboo machine the job removes the software modules with the recently installed packages (remove cosmic package with its dependencies) to obtain clean Bamboo machine for every modules.

Used tools in the project

This chapter summarize the main tools which are used in my work to meet the requirements of the project.

Cosmic Build System

The developed custom build system for Cosmic modules uses python distutils and the Koji build system which is operate with the git source code repository.

The python distutils services makes very simple the installation process or the package creation process for Cosmic modules. These processes can customize with setup.cfg file which is describe the installation folder and the required packages for the modules and the module.cfg describes some metadata (URL, Author, e-mail, version number of the module, etc.). The setup.py code contains a setup method, which is copies resource files to the installation folder. With python distutils, bdist_rpm command creates rpm package for the Cosmic module.

Koji build system used for release a new version of Cosmic module which is operate with the git repository (Checks the working branch and version number, add release notes for the module).

However, This custom made build system makes very easy to create packages and it is working properly, this system is not provide test execution and automated build options for the modules

Bamboo CI System

Bamboo is a continuous integration (CI) server that can be used to automate the release management for a software application, creating a continuous delivery pipeline. Bamboo schedules and coordinates the work involved in building and testing an application. Therefore, to use Bamboo, you will need to already have the following set up [1]:

- a code repository that contains the complete source code for the project.
- build scripts
- test suites

Bamboo uses the concept of a **plan** with **jobs** and **tasks** to configure and order the actions in the workflow. These concepts may deserve more explanation, because these concepts appear later in this document. I used these "Bamboo building blocks" to solve my task. The main properties of these concepts summarized by table 1, and the hierarchy of these "Building blocks" can be seen on figure 1. In this project, I used the middle structure on figure 1 for the stages, so each stage within a plan contains only one job which is executes several tasks on a Bamboo machine.

Plan	Stage	Job	Task
Processes a series of one or more stages that are run sequentially using the same repository.	Processes its jobs in parallel , on multiple agents.	Processes a series of one or more tasks that are run sequentially on the same agent.	Run sequentially within a job on a Bamboo working directory
Specifies the default repository.	Must successfully complete all its jobs before the next stage in the plan can be processed.	Controls the order in which tasks are performed.	Is a small discrete unit of work, such as source code checkout, running a script
Specifies how the build is triggered.	May produce artifacts that can be made available for use by a subsequent stage.	Collects the requirements of individual tasks in the job so that these requirements can be matched with agent capabilities	
Specifies notifications of build results.		Defines the artifact that the build will produce	
Provides for the definition of plan variables .			

Table 1. Properties of the building blocks of Bamboo CI System

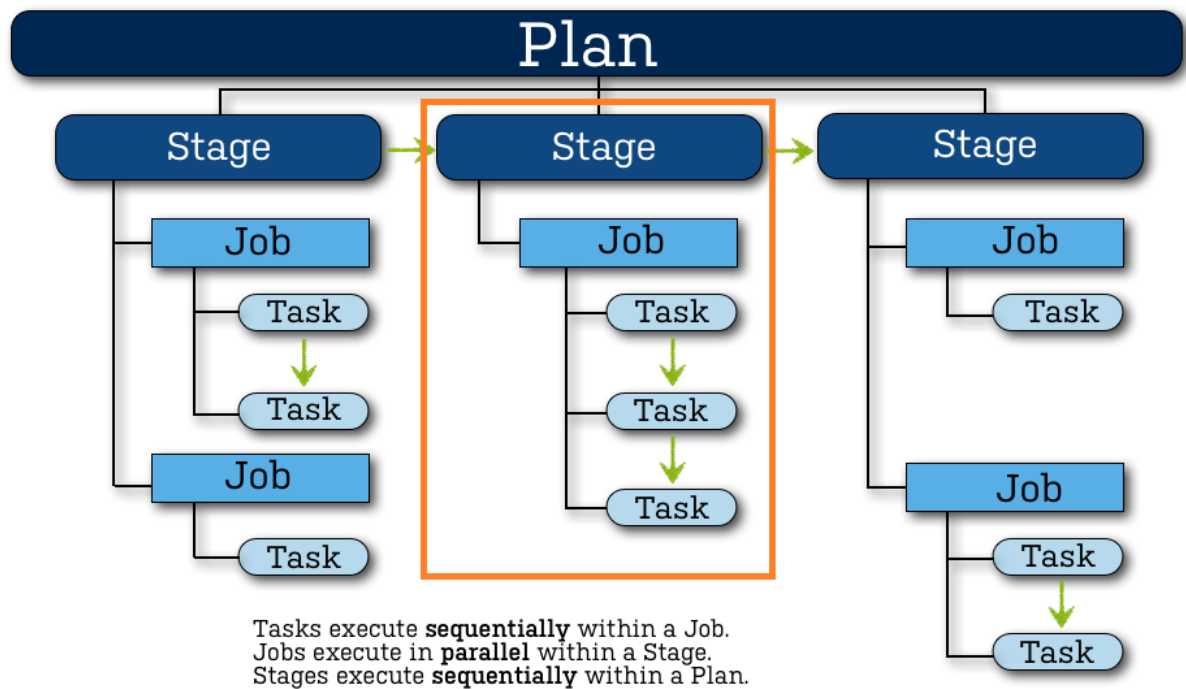


Fig. 1 Bamboo plan anatomy.

Selenium

Selenium is a test automation tool for web applications. Some modules of Cosmic have a web interface so they are running in Internet browser. Selenium IDE is a Firefox plugin which provides an easy-to-use interface for developing automated tests. It can record user interactions with the web browser and save it as a test suite (html format) that can be later executed to test Cosmic modules. These test suites can check the appearance of specific texts and other elements such as buttons or pop-up windows. After the execution of the test suite Selenium generates test results in html format.

Implementation

This chapter describes the details of the created Bamboo plans for Cosmic and shows the workflow of the process. To go into the details of the Bamboo jobs in this chapter also discuss the working mechanism of the created tasks and the creation of Selenium tests.

Created plans in Bamboo

I created different plans for SLC5/SLC6 architectures. The jobs of these plans have different requirements (see: configure jobs -> requirements). To build earlier versions of

software modules I define another plans to manage tasks with the tagged versions (in git) of the Cosmic modules. Thus, a failure of the package creation or installation of an earlier module version will not hinder the creation of the latest versions of the software modules. All of these plans consists of two stages. A stage can run when the previous stages have successfully finished (see: Table 1.).

The first stage executes a job which is performs a source code checkout and builds the rpm packages for the modules. After the package creation this job is responsible for the creation of the package repositories on AFS file system which are available for public. These processes are implemented in separated tasks within the job. These tasks uses predefined Bamboo local variables which are contains password and frequently used file paths (see: configure plan -> variables). The use of the stored variable makes more convenient to follow changes on the AFS file system (eg. changes the path of the repository.)

The second stage deploys the software modules on clean Bamboo machine one-by-one. The job runs Selenium tests for the module if the developers created test suite for the module.

The Bamboo plans sends notification via e-mail about the job status changes. Bamboo sends notification to cosmic-bamboo-notification group mail list. To make not so obtrusive these notifications, Bamboo will sends notifications only, when the status of the job has been changed (failed-succeed, succeed-failed transitions) (see: configure plans -> notifications).

In accordance with the described above, the created plans can be seen on figure 2. The name of the jobs within the stages are highlighted blue on figure 2. The workflow of the created plan can be seen on figure 3. As we can see on figure 3, the deployment stages are use the shared repository to deploy cosmic packages on the Bamboo machine. The job in stage 2 generates Selenium test results for the modules which have a test suite. These results are available on Bamboo server as a job artifact.

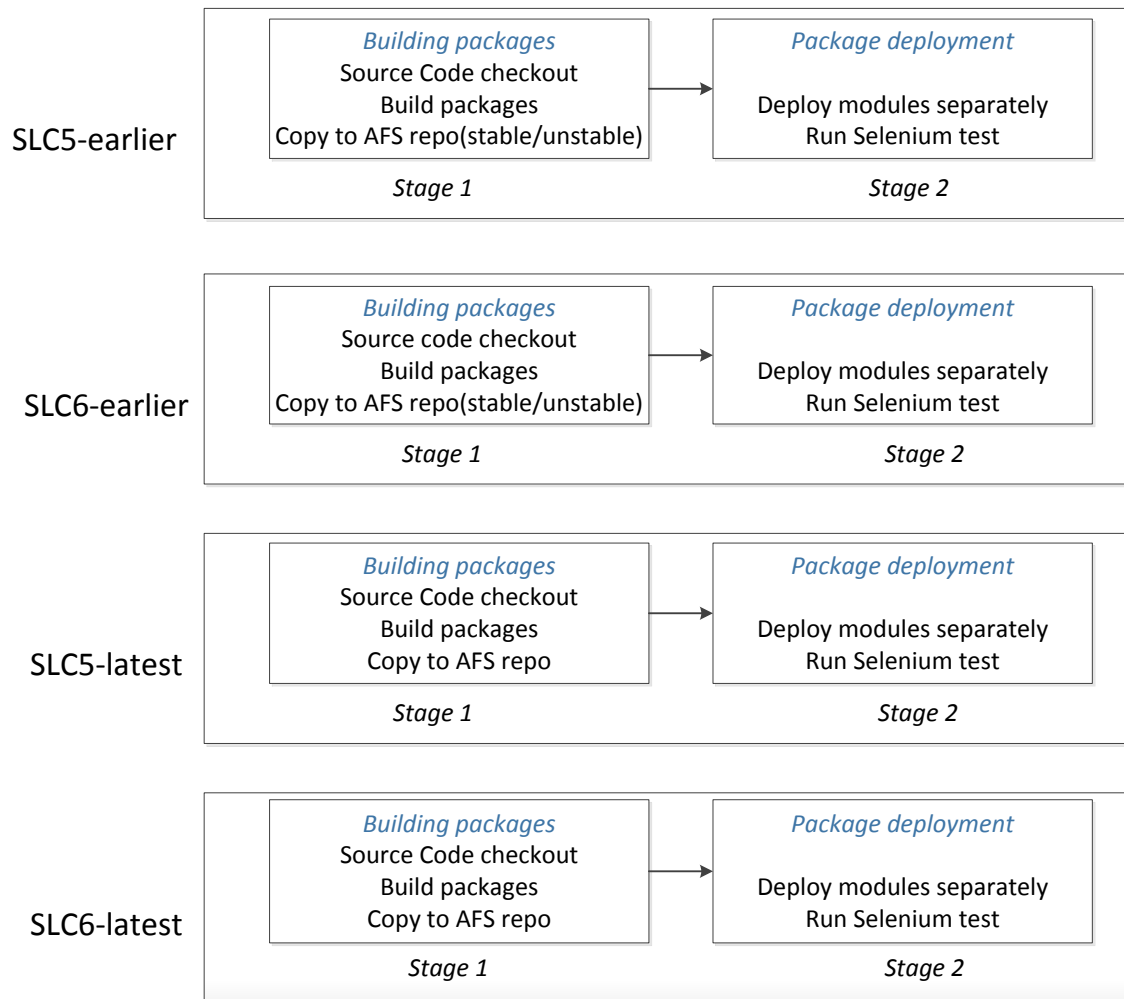


Fig. 2 Created Bamboo plans for Cosmic.

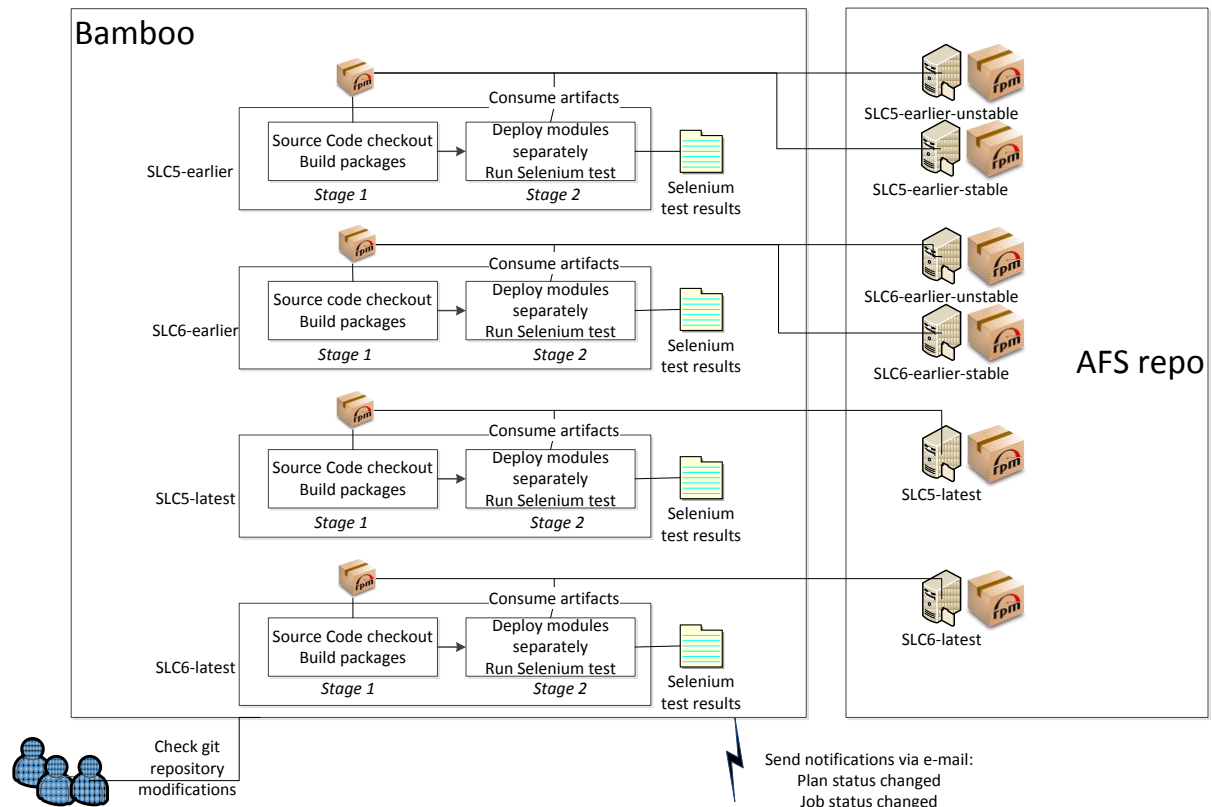


Fig. 3 Created workflow for Cosmic.

Package Creation with Bamboo

To create packages for the modules, I write bash scripts within a job in stage 1 (see Fig. 2). The scripts executed by the Bamboo agent. Job will perform a source code checkout as a first step.

After the source code checkout, the script will gather the name of the modules (or git tag names for earlier versions of Cosmic modules) to know for which modules have to create rpm packages. Because of the building dependency, Firstly the script install the cosmic main module to build any other module packages successfully (see Fig 4 and Fig 5). After package creation and installation of the main Cosmic module the script will build packages for other Cosmic modules sequentially. The flowchart of the script for the package creation for the latest version can be seen on figure 4, and for the previous versions can be seen on figure 5. On figure 5. k represents the number of the earlier versions of Cosmic modules and n represents the number of the tagged versions of the other Cosmic modules (so, there are the total number of the git tags is $n+k$). As we can see on the flowcharts, the scripts check the creation of the packages on Bamboo agent and they printed on the log file, when a package module does not created.

For the previous versions (see Fig. 5) the script executes a source code checkout with the corresponding git tag before to create tagged module. The script also checks the

existence of the tagged module on shared file repository to eliminate unnecessary work with the package creation. The task will create packages only for the "newest" earlier modules. Thus, with the proliferation of the git tags will not increase script run-time.

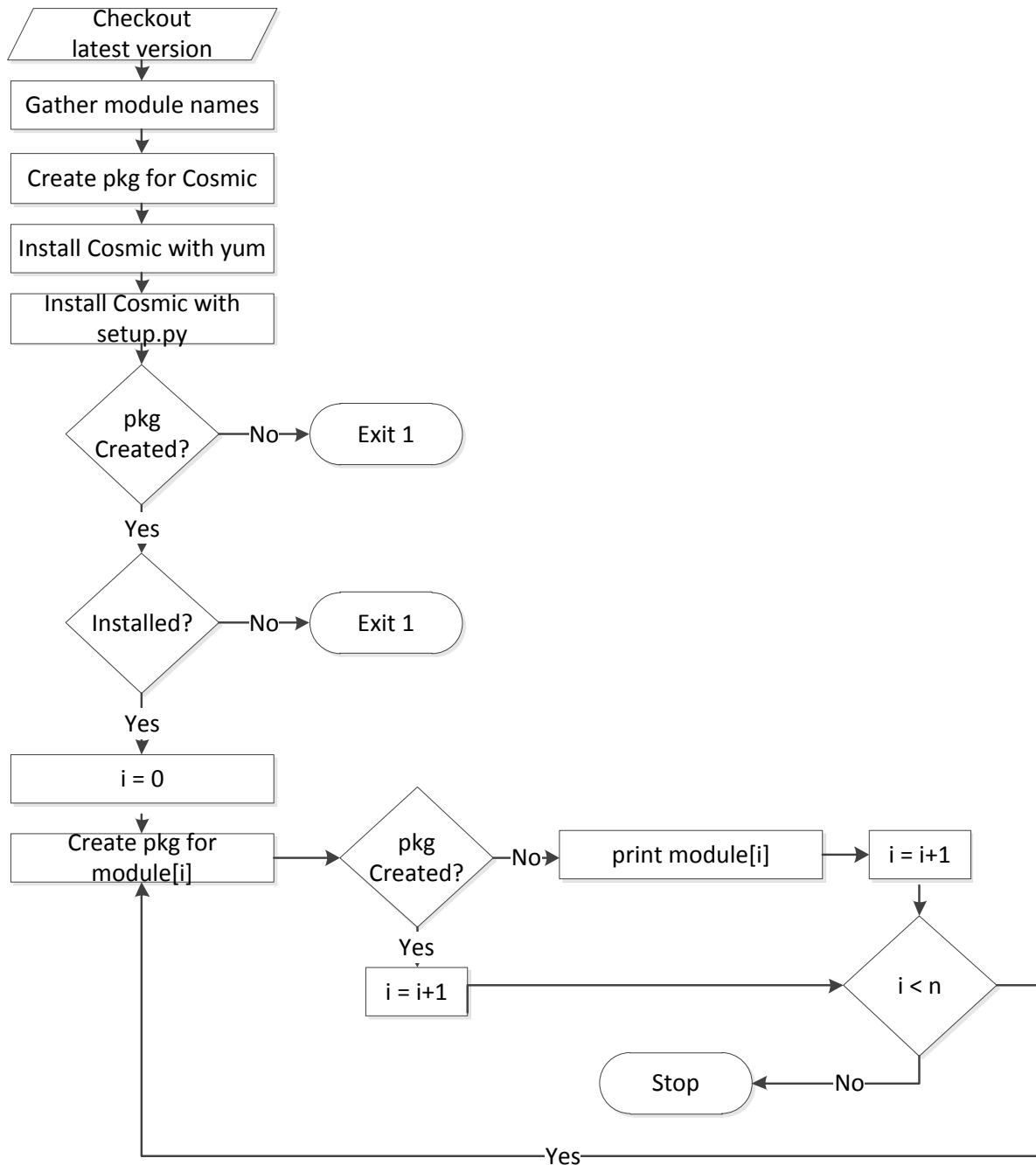


Fig. 4 Flowchart for creating latest rpm packages.

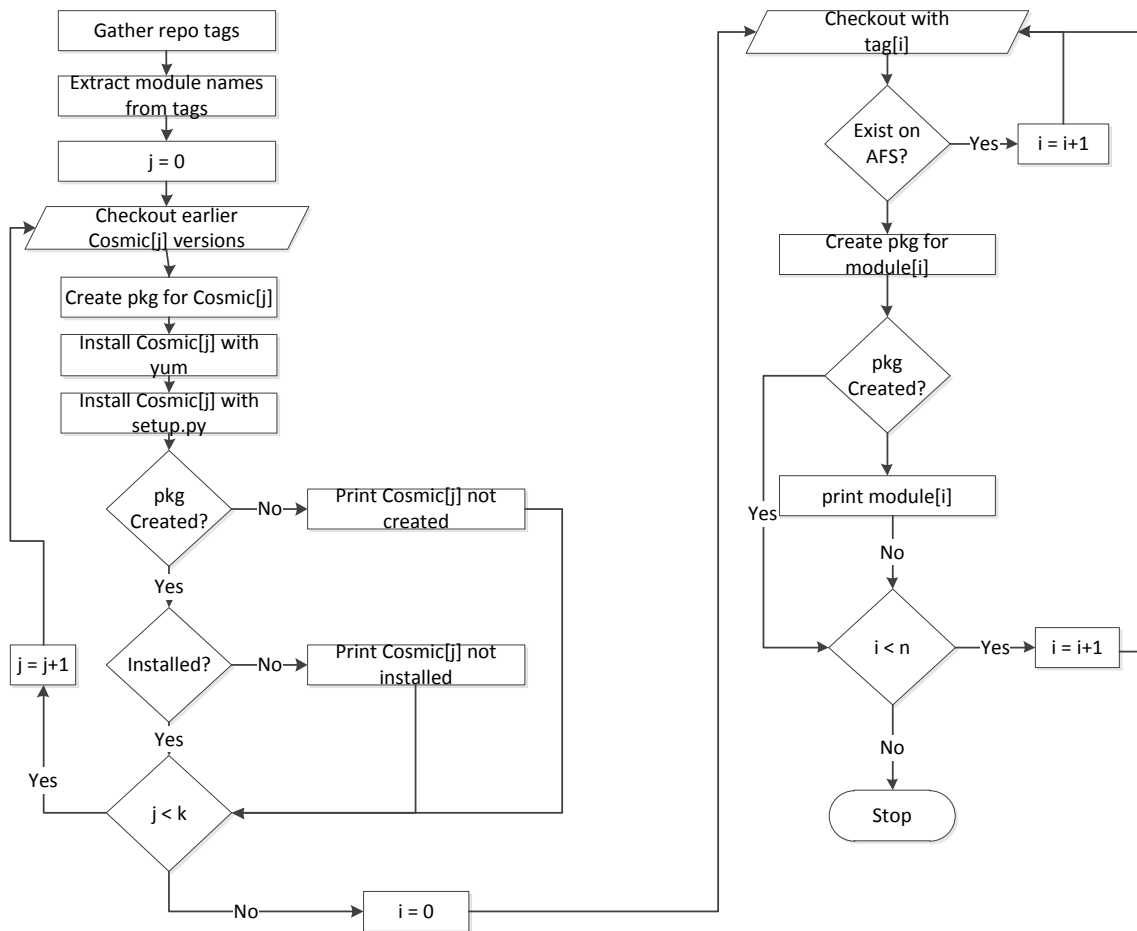


Fig. 5 Flowchart for creating earlier Cosmic packages

Package deployment and test running for modules in Bamboo

In stage 2 (see Fig. 2) a script within the job performs the package deployment using the shared file repository and runs Selenium tests. To run Selenium tests, the script installs Firefox and a display for the Bamboo agent. The Selenium tests (contains in cosmic/<module_name>/selenium/testSuite.html) are executed by Selenium-standalone.jar (jar file available on AFS file system). On figure 7. we can see a Selenium test for Cosmic-wlcc module which is checks some elements of the index page.

The flowchart of the script can be seen on figure 6. The script deploys packages one-by-one on clean Bamboo machine. After module deployment and Selenium tests, the script will remove all previously installed packages to keep clean the Bamboo machine for the next Cosmic module to see all required package install properly during the deployment. As we can see on figure 6. the script collects packages which are not deployed for some reason and prints out the undeployed modules. In this case the job will fail to inform the developers about the building or the deployment went wrong during the plan execution.

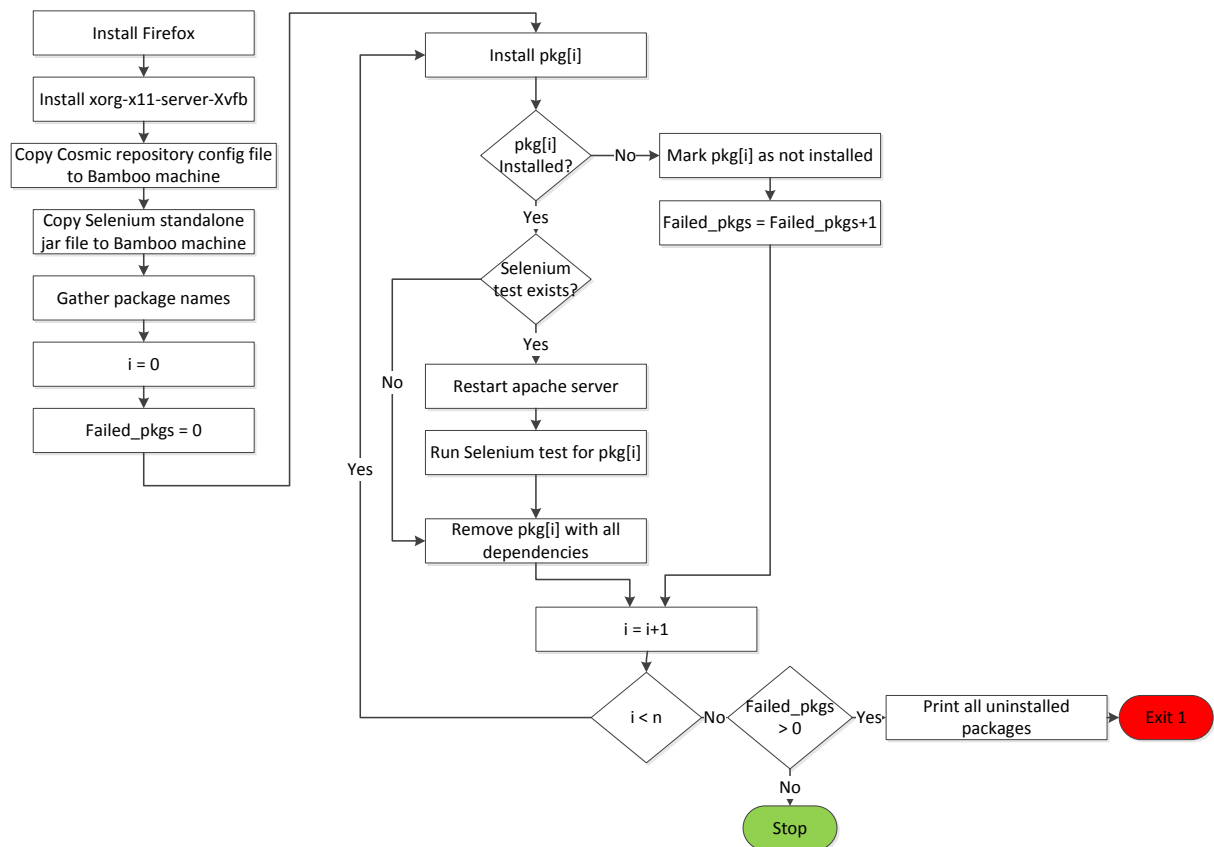


Fig 6. Flowchart for package deployment and test execution.

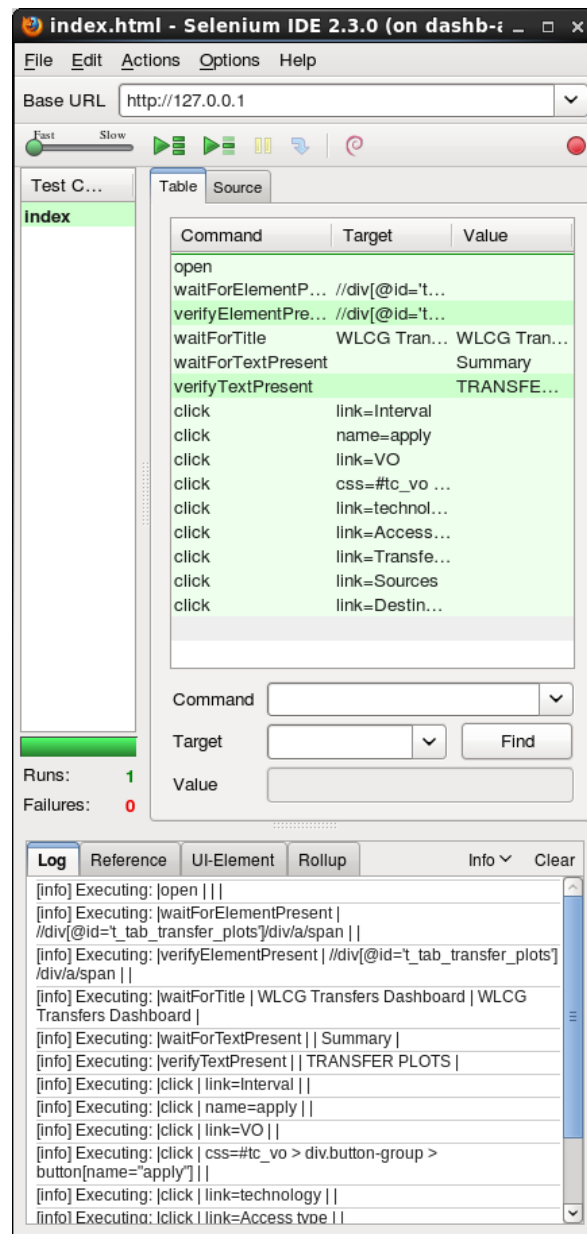


Fig. 7 Created Selenium test for Cosmic-wlcg module with Firefox plugin.

Selenium Tests

I created Selenium tests for the cosmic-django and cosmic-wlcg module to check the index pages of the applications after the deployment. The created test suites check the existence of the elements on the index pages. The created tests and their commands can be seen on figure 8. In the first step, the Selenium tests are navigate to 127.0.0.1 (localhost) to access to the index page of the Cosmic application on the Bamboo agent. On the figure 8. the first column contains the name of the Selenium commands which are

perform in Firefox browser, the second contains the targeted element of the command and the third contains the expected value of the element.

index			index		
open	http://127.0.0.1/cosmic/		open	http://127.0.0.1/	
verifyTextPresent		Page not found	waitForElementPresent	//div[@id=t_tab_transfer_plots]/div/a/span	
verifyTextPresent		^admin/	verifyElementPresent	//div[@id=t_tab_transfer_plots]/div/a/span	
open	http://127.0.0.1/cosmic/admin		waitForTitle		WLCG Transfers Dashboard
verifyTextPresent		unable to open database file	waitForTextPresent		Summary
cosmic-django			verifyTextPresent		TRANSFER PLOTS
			click	link=Interval	
			click	name=apply	
			click	link=VO	
			click	css=#tc_vo > div.button-group > button[name="apply"]	
			click	link=technology	
			click	link=Access type	
			click	link=Transfer VS Reading	
			click	link=Sources	
			click	link=Destinations	
			cosmic-wlcg		

Fig 8. Created Selenium tests for the modules.

The Selenium test suite for cosmic-django module contains two test cases. The first test case is created for the index page and the second for the admin page. The developers which are more familiar with the functionalities of the module can add more complex test cases for the test suite without to break Selenium test execution mechanism in Bamboo. The created test suite with the test cases for cosmic-django can be seen on figure 9.

After the running of the Selenium tests, the task makes the test results available on Bamboo framework as a job artifact. An example of the Selenium test result for cosmic-wlcg module can be seen on figure 10.

Test Suite			index		
index			open	http://127.0.0.1/cosmic/	
admin			verifyTextPresent		Page not found
			verifyTextPresent		^admin
admin					
open	http://127.0.0.1/cosmic/admin				
verifyTextPresent		unable to open database file			

Fig 9. Test Suite for cosmic-django module with two test cases.

Test suite results

result:	failed
totalTime:	30
numTestTotal:	1
numTestPasses:	0
numTestFailures:	1
numCommandPasses:	1
numCommandFailures:	2
numCommandErrors:	0
Selenium Version:	2.31
Selenium Revision:	.0
Test Suite	
index	

index.html		
index		
open		
waitForElementPresent	//div[@id='t_tab_transfer_plots']/div/a/span	Timed out after 30000ms
verifyElementPresent	//div[@id='t_tab_transfer_plots']/div/a/span	false
waitForTitle	WLCG Transfers Dashboard	WLCG Transfers Dashboard
waitForTextPresent		Summary
verifyTextPresent		TRANSFER PLOTS
click	link=Interval	
click	name=apply	
click	link=VO	
click	css=#tc_vo > div.button-group > button[name="apply"]	
click	link=technology	
click	link=Access type	
click	link=Transfer VS Reading	
click	link=Sources	
click	link=Destinations	

Fig 10. Selenium Test result in Bamboo framework for cosmic-wlcg module.

Acknowledgement

I want to thank Andres Abad Rodriguez, who helped me to understand how Bamboo CI System works in IT department.

I also want to say a big thank you to Alexandre Beche and Ivan Antoniev Dzhunov who are helped me to get familiar with the Cosmic source code, git repository commands, linux commands and to configure apache server to run installed Cosmic modules on virtual machine.

Finally, a special thanks to my supervisor Pablo Saiz who gave me useful advices for bash scripting, makes the tasks more effectively and for his help, when I stuck with the project.

I am very grateful for their helps and their encouragements.

References

[1] Atlassian documentation for Bamboo 4.4

<http://downloads.atlassian.com/software/bamboo/downloads/documentation/BAMBOO-4-4-20130130-PDF.pdf>