CERN SUMMER STUDENT REPORT 2015
PH-SFT
Evangelos Chatziafratis
National Technical University of Athens
CERN Summer Student 2015
Main Supervisor: Patricia Mendez Lorenzo
2nd Supervisor: Benedikt Hegner
PROJECT: Connecting Jenkins to Cdash
**JENKINS:** Jenkins is the infrastructure for continuous building, testing and distribution of software for LHC experimental community. All projects-like LCG-Externals (experimental, dev2, dev3, dev4), ROOT, Geant etc-are monitored with the graphical interface of Jenkins. After every build, we can see the results of the build using the Console Output. Furthermore, we can use some of the Jenkins Plugins to set a description for our builds.

**CDASH:** Cdash is the monitoring toolkit that collects the results of the builds/tests executed by Jenkins and publishes them in detailed and structured way. It helps a lot because it creates graphs and sketches for the errors happened during the build, the compiler warnings etc and it can be helpful to quickly review the result of the build in a human friendly way. Because it is a separate infrastructure than Jenkins, all files need to be uploaded by the Jenkins so that afterwards can be displayed by Cdash. However, the link created inside the Jenkins was now working as we will see explicitly in the next page of the current report.

**PROJECT:** My job was to achieve the integration of Cdash into the Jenkins that was missing. This integration will provide Jenkins with the monitoring potential provided by Cdash. The goal was to have at each build a URL that would redirect us directly to the results of Cdash for the particular build that we are interested in. This will improve the way one can go from Jenkins jobs’ results to the Cdash results and can save a lot of time, because after this URL is created, there is no need for searching inside the Cdash for the build we want; we can go there directly clicking on the link of Jenkins.

**DESCRIPTION SETTER PLUGIN:** For the main part of the project we had to use a plug in for Jenkins infrastructure that was already available in Jenkins Plugins. The name of the plug in is DESCRIPTION SETTER and is used for setting the description of the build according to a regular expression provided to the plug in that is set by us, the users. The regular expression is used to parse the log output (console output-log file) and using the regular expression we can have a capturing group that will then be used for the job description. We can have multiple capturing groups as well and so with this plug in we can output many useful facts for the build that can be deduced from the log output.

We will now show some pictures describing the situation before the project was done and after I finished the project. Below, we have two pictures depicting the situation before the project. The URL was not working and
Figure 1: The situation before the Project was started. Here is the link that Jenkins created but was not working. With blue is the link we clicked on.

Figure 2: The message we were receiving when we clicked on the link provided by Jenkins that supposedly would redirect us to Cdash results of the build.

we were redirected to a rubbish site. Below, in FIGURE 3, we have the final link that the Description Setter plug in was creating for us based on the regular expression we provided. This would redirect us to the correct Cdash job that corresponds to the Jenkins job. To show how the Description setter works we need to go to the Jenkins Configure section and choose SET BUILD DESCRIPTION. Then, the box of FIGURE 4 appears. We select the Regular expression to be HERE IS THE CDASH URL: ‘(.*)’ because then the plug in will parse the log output and when it finds the exact phrase HERE IS THE CDASH URL: then it will capture whatever follows. We need to highlight that the regular expressions rules tell us that the .* is used to match anything until the next line and the parenthesis are used to denote the capturing group. So what we need to do is to output in the log output of the Jenkins job the phrase HERE IS THE CDASH URL: followed by the correct URL that will redirect us to the Cdash job results. After we have captured the URL, we can set the description to be the URL but using a more friendly message; in our case the message was LOOK: CLICK HERE FOR THE CDASH RESULTS OF THE BUILD and the \<a href\> is used to display it as an active link. The \1 is used so that the plug in knows where
Figure 3: The link that was created using the Description Setter Plugin.

Figure 4: The box shows what the regular expression is and how we want to use the capturing group for the Description in case the build fails or not.

the capturing group needs to go at the final Description. We can also use multiple groups and we can refer to them using \1, \2 (or even \3) accordingly in our description. We will describe afterwards how I was able to create this link correctly.

**CREATING THE URL TO OUTPUT:** Jenkins creates many environment variables that can be used by our scripts to perform certain tasks. What I particularly used, were the $SLOTNAME, $MODE, $BUILDTYPE, $COMPILER, $LABEL. They contain information about the machine that the build was run, the type of the project (experimental, dev2, dev3, dev4

Figure 5: The script we used for the timestamp and the echo of the URL to the log output.
etc.), the use of compiler (gcc48, gcc49 etc.). They are very useful because they are created by its job dynamically and so we can achieve portability of our script. The command I used was the "find" command. The $TIMESTAMP variable was created based on the "find" command. The timestamp is used in the Cdash filtering section and it is the time of the build. A folder which has the timestamp as the name is located inside the folder "Testing" that the Jenkins creates and uploads during the build of the job. So, using "find" we can take the name of the folder and then use it and put it inside the URL we create and we extract to the log output.

At the beginning of the job, we used a 3 valued filter for Cdash. The filters were the BuildName, the NodeName and the Timestamp. Afterwards we thought we should try only the Timestamp as a filtering choice and we implemented the script using only the parts that had to do with the Timestamp. In conclusion, we managed to connect Jenkins with Cdash and have a URL link from a Jenkins job directly to the correct data in Cdash so that easily one can monitor the status of the build!

SPECIAL THANKS: Here I would like to thank my main supervisor Patricia Mendez Lorenzo for her help and kindness and my second supervisor Benedikt Hegner for his time that he spent with me discussing parts of the project :) Finally, I would like to thank CERN for the opportunity of coming here as a Summer Student :) It changed my life :)

Figure 6: An example of how regular expressions can be used to capture information.