Improvements to the EMI Build and Test Tools

Andres Abad Rodriguez
CERN
CH-1211. Genève 23, Switzerland
E-mail: andres.abad.rodriguez@cern.ch

Duarte Bacelar de Begonha De Meneses
CERN
CH-1211. Genève 23, Switzerland
E-mail: duarte.meneses@cern.ch

Abstract

One of the major goals of the EMI is the integration of the components of the pre-existing middleware into a single consistent set of packages with uniform distributions and repositories. The EMI build and release is based on ETICS. During the last year, a number of changes have been implemented in the development tools and to ETICS in particular: such as new platforms, infrastructure changes, new QA reports, etc, in order to provide the features needed for the EMI 2 and 3 releases.

EGI Community Forum 2012 / EMI Second Technical Conference,
Munich, Germany
26-30 March, 2012

1. Introduction

One of the main goals of the European Middleware Initiative (EMI) [1] project was the integration of the pre-existing middlewares (ARC, gLite, UNICORE and dCache) into a single consistent set of packages with uniform distribution and repositories. The primary tool used to achieve this goal during the first two years of the project was ETICS [2], together with some other tools used to generate charts, metrics and reports.

The main task during the first year of the project was mainly the integration of all the middlewares products inside ETICS to get a single set of packages; but during this second year the tools had to be updated to implement new requirements. Those changes affected all the components of the system: infrastructure, clients, etc.

This document summarizes the changes made within the ETICS system and other tools during the second year of EMI.

2. How ETICS system works

To understand the modifications made, it is necessary to keep in mind how the ETICS system works.

ETICS is the integration, build, packaging and testing tool [7]. The definition of the repository for source code, how to build it, and so forth is specified using ETICS configurations. These configurations can be edited using a web interface or a command line client. Using these tools is also possible to do the remaining necessary tasks, such as build and test your configurations, access the build repositories or check the status of your current and previous jobs.

To do the desire build or test of configurations, there is a pool of nodes containing the various supported platforms. These nodes are virtual machines containing a clean installation of a minimal image created in advance by the tools team. The minimal images contain the minimal set of packages installed to allow the system to build. All the dependencies needed are installed on demand. When a build or test is submitted, the job information is sent to one of the nodes according to the user requirements. The source is checked out, the build commands executed, etc. At the end of the build, the artefacts are sent to a repository, the reports stored and the node deleted.

All the artefacts created during the build or test, including the job reports, are stored in repositories that ETICS creates automatically immediately after each build.

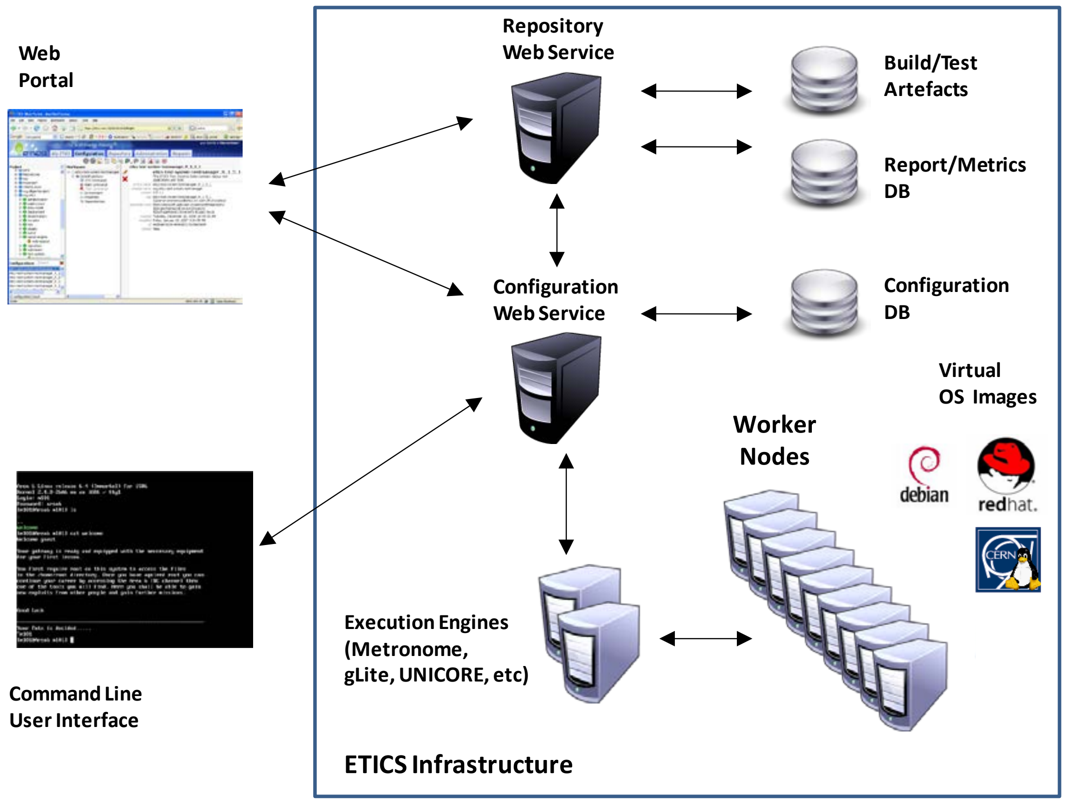

Figure 1General picture of how the ETICS system works

3.Improvements and modifications done during the second year of the EMI project

3.1 New platform images

An important requirement in the second year has been the addition to the current list of the platforms Scientific Linux 6 (SL6) 64 bits [3] and Debian 6 64 bits [4].

3.1.1 Scientific Linux 6

SL6 was a quite simple task. The ETICS system already supported Scientific Linux 5 (SL5) 32 and 64 bits for the first year. The new SL6 platform was added with only few modifications needed in the system. As it was SL5 before, SL6 was also supported by HyperV, the virtualization tool at CERN and the used in ETICS.

3.1.2 Debian 6

Debian 6 was a more challenging task due to the lack of experience of the Debian environment within the team. The packages installed to create the minimal image were slightly different due to the fundamental difference in how Debian splits core libraries, clients, development and server side packages which conforming to the Debian Policy document. In addition, some problems relating to Debian not being officially supported in HyperV needed to be solved.

3.2 Changes in the ETICS command line client

Due to the new platforms, the ETICS command line client has been modified. Improvements were needed in how the client is installed, its dependencies and how it is executed. For SL6 only small adjustments were needed, such as modifications to adapt the system the differences in packaging between SL5 and SL6. For Debian, the knowledge of this platform inside the EMI tools team was not enough. Consequently, a task force consisting of Debian experts from the different middlewares was created as a starting point to collect all the information needed for the modifications.

At the end of the build phase, a new repackage phase is triggered to obtain a binary RPM or DEB according to the Fedora or Debian packaging policies respectively. Mock [5] is used for the RPM generation and PBuilder [6] for the creation of DEB packages. Two modifications were needed in the repackage. The first one was to modify the repackage behaviour. Previously, it was triggered once all modules had finished their build phase. The new behaviour is triggering the repackage at the end of each module build phase. This was needed because in this second year of EMI the recommended way of building is to provide everything and only use ETICS for the repackage phase; but some developers are still building using ETICS. To support this hybrid system, the repackage phase works had to be modified. This was initially a simple modification, but some performance problems obstructed these changes from reaching production some weeks from the expected date. The second modification needed was to add PBuilder into the system. The advice given by the Debian task force regarding this point were very important to reach an initial starting point and to know what were the steps to be followed to successfully use this tool.

3.3 Changes in the repositories

The repositories had to be also updated to support the new platforms. For SL6, only modifications to support new compressors were necessary. For Debian, the generation of APT repositories had to be developed from scratch. The functionalities in APT repositories are the same as those in YUM. All the new file types (.dsc, .deb) are correctly handled and it is possible to browser their content in the web interface as it was already possible with all the other types (.rpm, .tar.gz, etc…) used in the YUM repositories.

3.4 Infrastructure modification to improve job performance and resource usage

With the new platforms in place, there was the possibility that the existing resources were not enough to manage the demand. A study of the current situation at that moment was done to discover which parts of the system could be tuned to decrease the job build/test time. The final modifications made in the hypervisors and in the nodes decreased the total time between 10% and 15%. These improvements related to:

- A new file system being adopted in the worker nodes. The existing EXT2 file system was replaced by the newer and quicker EXT3.
- The disk type was changed in the worker nodes: from IDE to SCSI.
- The RAID level was modified in the hypervisors: from RAID-1 (mirroring) to RAID-0 (striping).

- Separate virtual disks in the nodes to allow easy disk resizing for specific needs.

The second improvement was in the usage of the resources. The previous paradigm used to control the pool was a static one. There was always the same amount of nodes started of each platform. This was very inefficient as the load not always being the same and there are moments where some platforms are in more demand than others. The solution adopted was to move from a static to an elastic paradigm. With the new system, the amounts of nodes that are started for each platform are exactly the amount of nodes waiting in the queue. The system tries also to have always one free node of each type running to decrease waiting times. This new paradigm of starting on demand allows the system to handle high demands in specific types. The usage of resources with this paradigm is more optimal than before as only the nodes that are going to be used are started.

All these changes have improved the handling of the node demand, including the new platforms, without increasing the hardware resources.

3.5 Reliability

Another of the key aspects for this period was the reliability of the system. To improve this situation, the monitoring system was updated with new scripts and revisions of the existing ones. Some examples of those scripts are the verification of the backup files, AFS spaces usage, or broken node detection. The system was also improved to send SMS alerts in case of urgent matters. Checks for some know recurrent failures were also implemented, which where possible will also fix them. Another addition to the tools for monitoring the system is *collectd*. This tool has been installed on every node. It collects information about several system components and sends it to a central server. All that information can be aggregated afterwards to analyse the resource usage or to detect possible bottlenecks.

3.6 Improvements for developers

Several developers in the project use Maven as an integration tool and as a collection point for their external libraries. For faster access and reliability reasons, a Maven mirror has been installed at CERN. It forms a mirror of the official Maven repository and others demanded by the developers.

The minimal image used in the ETICS nodes is where all builds take place. To facilitate the work with these images, several actions were taken. The first was to make available all the images to be downloaded, together with tools to convert them to other virtualization system formats. Documented instructions about how to create one of these nodes was also made available.

The last modification done for the developers was the possibility to access to snapshots of the node where their builds were running. At the end of each build, the script that controls the pool creates a snapshot of the node before it is destroyed. Using these snapshots, the developers can access the nodes and check their logs to discover possible errors in their jobs.

3.7 Quality assurance (QA) reports improvements

One feature of the ETICS system is its extensibility, using plugins to do extra actions during some of its phases. This feature has been used to provide the data for the QA task. During the first year of EMI a set of plugins had already been added to run static code analysis and other tests [8] (SLOCCount, FindBugs, Checkstyle, PMD, etc...), but new ones have been added in this second year: RPMlint, CPPcheck, PyLint and IPv6. RPMlint deserves some comments due to its importance inside the project. It is a tool adapted as an ETICS plugin that checks whether a RPM is following the official packaging guidelines. It was really important to have this plugin in place as one of the objectives for this second year was to generate packages that could be released in the EPEL repositories.

A new chart generator framework has been created to simplify as much as possible the generation of charts for all the QA existing and new reports. Figure 2 describes how it works:

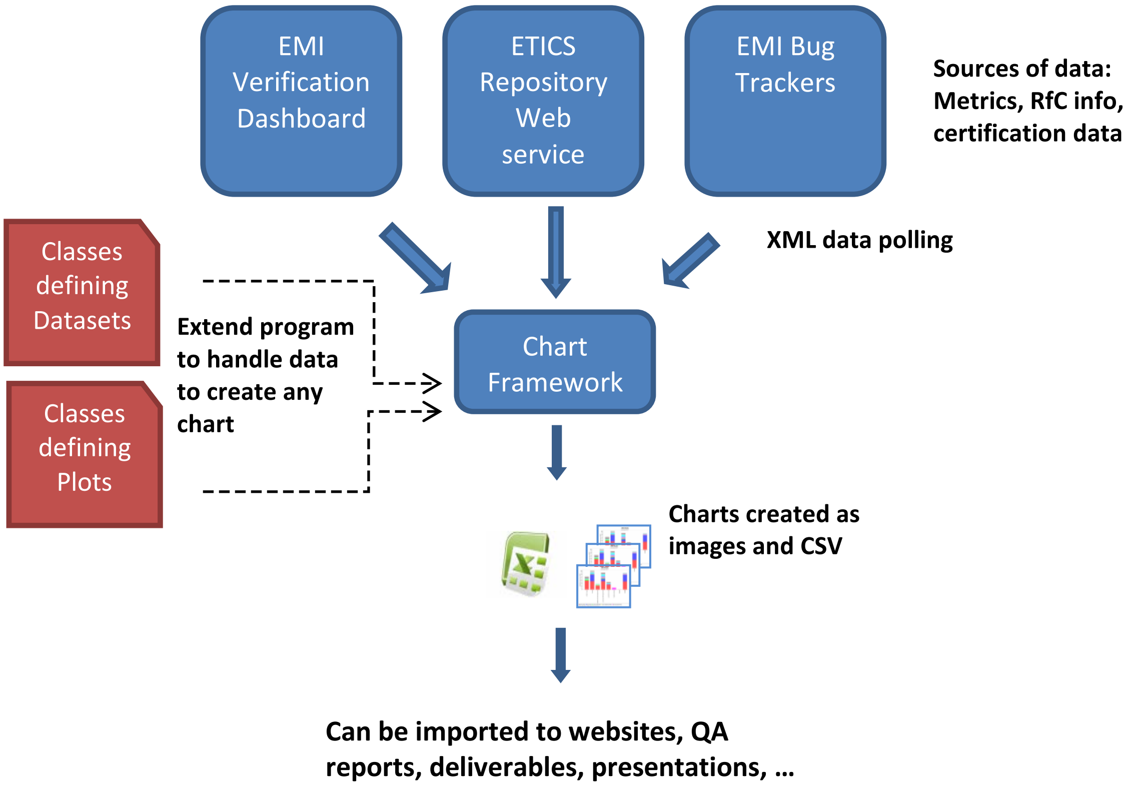
EMI Verification Dashboard
ETICS Repository Web service
EMI Bug Trackers
Sources of data: Metrics, RfC info, certification data
XML data polling
Classes defining Datasets
Classes defining Plots
Extend program to handle data to create any chart
Chart Framework
Charts created as images and CSV
Can be imported to websites, QA reports, deliverables, presentations, …

Figure 2 How chart generator framework works

It collects the information from the different bug trackers, the EMI dashboard and ETICS to generate, according to some definitions, the necessary charts for each report. One of the major advantages of this framework is that is easily expandable to new types of charts.

4.Conclusions

The new two platforms required for the second year of EMI brought together modifications in almost all parts of the system, mainly due to the new Debian platform. That platform was successfully integrated despite it was initially unfamiliar to the team. The system has been adapted to use standard tools for the generation of the final binary DEB and RPM packages. These tools are Mock (for RPMs) and PBuilder (for DEBs). Standard repositories (APT and YUM) are now used and should improve the future sustainability.

One major achievement was the improvements of the infrastructure. With those changes in the configuration of the hypervisors, the nodes and the scripts that control the queue; the system was able to absorb the new demand generated with the addition of the new platforms. The increase in demand is more than doubled compared with the amount of jobs sent in the previous year.

The QA reports are more complete now with the addition of new types of charts. The report output was improved by creating the chart generator framework.

References

[1] European Middleware Initiative http://www.eu-emi.eu/

[2] ETICS http://etics.cern.ch/eticsPortal

[3] Scientific Linux https://www.scientificlinux.org/

[4] Debian http://www.debian.org/

[5] Mock http://fedoraproject.org/wiki/Projects/Mock

[6] PBuilder http://pbuilder.alioth.debian.org/

[7] Software Build Systems: Principles and Experience (ISBN-10: 0321717287)

[8] Measuring Performance : Using the new metrics to deploy strategy and improve performance (ISBN-10: 0970247117)