Journal of Physics: Conference Series **513** (2014) 062018 doi:10.1088/1742-6596/513/6/062018

An integrated infrastructure in support of software development

S Antonelli[1], C Aiftimiei[2], M Bencivenni[1], C Bisegni[3], L Chiarelli[4], D De Girolamo[1], F Giacomini[1], S Longo[1], M Manzali[1], R Veraldi[1] and S Zani[1]

[1] INFN-CNAF, Bologna, Italy
[2] INFN, Padova, Italy
[3] INFN-LNF, Frascati, Italy
[4] GARR, Roma, Italy

E-mail: stefano.antonelli@cnaf.infn.it, cristina.aiftimiei@pd.infn.it, marco.bencivenni@cnaf.infn.it, claudio.bisegni@lnf.infn.it, lorenzo.chiarelli@garr.it, donato.degirolamo@cnaf.infn.it, francesco.giacomini@cnaf.infn.it, stefano.longo@cnaf.infn.it, matteo.manzali@cnaf.infn.it, riccardo.veraldi@cnaf.infn.it, stefano.zani@cnaf.infn.it

Abstract. This paper describes the design and the current state of implementation of an infrastructure made available to software developers within the Italian National Institute for Nuclear Physics (INFN) to support and facilitate their daily activity. The infrastructure integrates several tools, each providing a well-identified function: project management, version control system, continuous integration, dynamic provisioning of virtual machines, efficiency improvement, knowledge base. When applicable, access to the services is based on the INFN-wide Authentication and Authorization Infrastructure. The system is being installed and progressively made available to INFN users belonging to tens of sites and laboratories and will represent a solid foundation for the software development efforts of the many experiments and projects that see the involvement of the Institute. The infrastructure will be beneficial especially for small- and medium-size collaborations, which often cannot afford the resources, in particular in terms of know-how, needed to set up such services.

1. Introduction

The success of a scientific endeavor depends, often significantly, on the ability to collect and later process large amounts of data in an efficient and effective way. Despite the enormous technological progress in areas such as electronics, networking and storage, the cost of the computing factor remains high. Moreover the limits reached by some historical directions of hardware development, such as the saturation of the CPU clock speed with the consequent strong shift towards hardware parallelization, has made the role of software more and more important.

This paper describes the design and the current status of implementation of an integrated infrastructure of services aimed at supporting and facilitating the daily activities of INFN researchers and technologists who work with software in roles as diverse as developers,

maintainers, reviewers, supporters or project managers. The activity is carried out as part of the ISSS project [1], with the purpose to augment the existing INFN-wide IT services.

Once completed, the infrastructure will represent a one-stop shop offering a variety of tools already configured to support established best practices, so that the quality of the produced software and of related services (notably support) could be continuously improved at a decreasing cost. The generic term *quality* refers to characteristics such as low presence of defects, runtime performance efficiency, maintainability, easy portability to new platforms. Similarly, the generic term *cost* covers many aspects, such as time devoted to development, test, support and maintenance, money spent in hardware resources and electrical power, low service reliability.

The system is being installed and progressively made available to users belonging to tens of INFN sites and laboratories and will represent a solid foundation for the software development efforts of the many experiments and projects that see the involvement of INFN. The infrastructure will be beneficial especially for small- and medium-size collaborations, which often cannot afford the resources, in particular in terms of know-how, needed to set up such services.

This paper is organized as follows: section 2 lists the functions that have been considered in the initial design of the infrastructure; section 3 describes the status of the implementation of those functions at the time of writing; section 4 describes the foreseen evolution of the infrastructure both for the existing tools and for the introduction of new services that have been identified in the meantime; in section 5 we present some concluding remarks.

2. Functions

The infrastructure is designed to comprise several services, each providing a function. The functions so far identified include: project management, version control system, continuous integration, availability of development platforms, efficiency improvement, knowledge base.

When applicable, access to the services is based on the INFN-wide Authentication and Authorization Infrastructure (AAI). For web applications the AAI offers a SAML-based Identity Provider (IdP) that allows a user to authenticate via either a personal X.509 certificate or a username/password pair. The AAI is available also to non-INFN users via a simple registration procedure, permitting also to projects not entirely formed by INFN users to access this infrastructure.

2.1. Project management

Any software project that goes beyond a limited-duration single-person exercise would benefit from a project management system. The more complex the project is (more developers, more users, larger code base), the larger the benefit. The main functionality offered by a project management system is the possibility to track a variety of issues, such as: requests of support submitted by users; requests to apply changes to the code (be they due to defects or additional improvements); internal tasks to manage releases and reviews. Additional useful features include support for reporting project metrics and statistics and for linking code change requests to actual changes in the code base. This latter aspect, if properly applied, enables the full control of the whole chain from user requirements to software package deployment.

2.2. Version control

A *Version Control System* (VCS) allows to store permanently any change applied to a code base (source code, tests, documentation, build and packaging instructions, etc.) along with metadata to keep track of the history of the changes. Versioning information is essential in a variety of situations: to keep alive and actively maintain multiple releases of the same software component; to undo changes that have introduced defects; to properly support a development model that allows multiple people to further develop the same software component in parallel.

2.3. Continuous integration

In order for the developers to keep their confidence high that changes applied to the code base, by themselves or by their peers, do not cause regressions, it is a recommended practice to verify continuously (e.g. periodically or even at every change) that all changes introduced in the software at least build correctly and pass basic tests. If portability is a relevant characteristic of the software, the job executing the build and the tests should run on multiple platforms. If the software provides some network services, the tests may require the deployment of the software on multiple nodes and their co-ordination during the test run.

As an additional benefit, the automatic build and test phases are an excellent occasion to run other quality checks. Several static and dynamic analysis tools exist that are able to expose actual or potential defects in the code before they reach production.

2.4. Availability of development platforms

The software development activity, especially for software that has to be available on multiple platforms, often requires the availability of multiple nodes: for actual coding, for building, for deployment and testing. To allow flexibility in the allocation of such resources from the point of view of both the user and the provider, a solution based on a system capable of instantiating virtual machines on demand stands as an obvious choice.

2.5. Efficiency improvement

Given the magnitude of the computing needs of a typical current experiment, even a slight improvement in the efficiency of the software brings significant savings in terms of resources, be they processors, memory, storage, network or, more and more relevant, electrical power.

Moreover the current trend towards increasing hardware parallelism, either as multi-core CPUs or as highly-parallel accelerators like GPUs or MICs, requires that the full exploitation of the potentially available performance be addressed at application level, with possibly large modifications to existing code to make it more akin to parallel execution. Multiple sources of parallelism and optimization are exploitable: overall computing model, algorithms and data structures, programming languages, compilation options, implementation techniques.

In most, if not all, of these cases it is of paramount importance to be able to profile the program execution with adequate tools so that the changes are guided by objective measurements and not by guesses.

2.6. Knowledge base

Even the availability of the best tools on the market would not be sufficient to achieve the stated goal of producing good-quality software. More important is the sharing of knowledge and experience already acquired by others. Different means are employable to spread knowledge, such as training courses, discussion forums, consultancy by experts, collaborative platforms.

3. Current status

At the core of the implementation of many of the functions described in the previous section lies a virtualization infrastructure based on a number of *KVM* [2] hypervisors sitting inside a private, non-routed, B-class network, accessing external resources via NAT. The network services (DHCP, DNS, NTP, NAT, etc.) are provided via *pfSense* [3]. The storage needed by the virtual machines is provided with *FreeNAS* [4], configured with the ZFS file system, which guarantees both performance and resilience to disk failures. The orchestration of the infrastructure is based on *oVirt* [5], the community edition of Red Hat Enterprise Virtualization Manager, offering a flexible set of instantiation options. The abstraction layer of the whole system is based on *δ-Cloud* [6], which offers both an EC2 [7] and a proprietary interface. The overall layout is shown in figure 1.

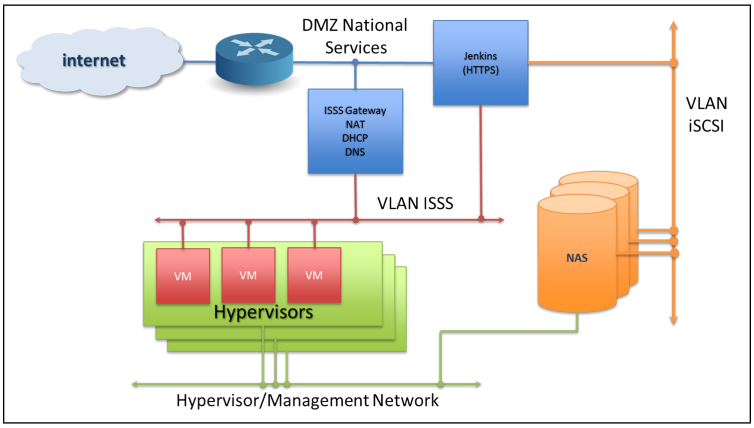

Figure 1. The overall layout of the underlying virtualized infrastructure.

For project management we have chosen Atlassian *JIRA* [8], a widely-used tool that offers excellent support for large and complex development projects involving many users. It covers many aspects of the software project lifecycle, including: interaction with users for requirements and support; organization of issues, tasks and activities; integration with code repositories; reporting. Authentication to the service is based on the Single Sign-on made available by the INFN-wide AAI.

Jenkins [9] provides the framework for continuous integration and represents the foundation for any process that aims at producing high-quality software components in a reproducible way, as shown for example in figure 2. A few slaves are attached to the system, covering the most common Linux distributions for 32- and 64-bit platforms. Authentication to the service is based on the INFN-wide AAI. Since the service is accessible by a potentially large number of users, belonging to many distinct communities, particular attention has been devoted to the configuration of the system so that jobs run by different users are well isolated from one another.

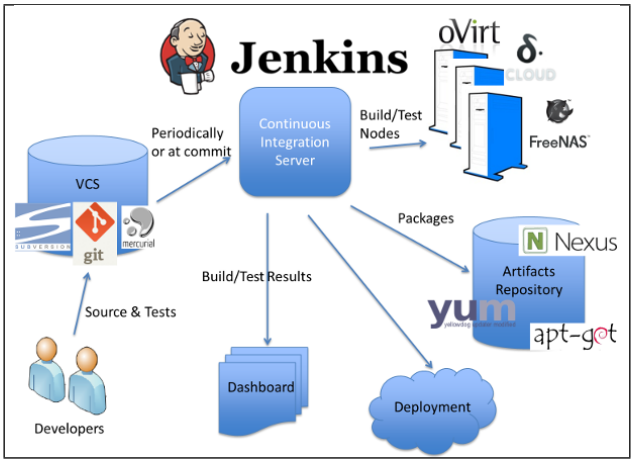

Figure 2. A typical software process using a continuous integration approach.

Subversion [10] has been chosen as version control system to cover existing needs. The authentication to the system is based on SSH public keys. Beside Subversion, solutions for *git* [11] and *mercurial* [12] repositories hosted by third-parties, such as *GitHub* [13] and *Bitbucket* [14], are encouraged and well supported by the other tools, notably JIRA and Jenkins.

4. Future evolution

The future evolution of the infrastructure depends mainly on the needs and priorities expressed by the community of users.

One immediate action is the integration of static analysis tools into Jenkins.

Following that, the underlying virtualization infrastructure will be opened to any user, so that anyone can demand a host for personal use, e.g. for development or testing. There are no significant technical issues to make this happen; however the security implications have to be better understood and addressed, also from the legal point of view.

Another already identified need is the availability of a collaborative platform, which will be addressed shortly, firstly with a chat service and later with a Q&A system. The latter in particular would greatly help share the large accumulated knowledge on computing matters in the community.

In the longer term we plan to investigate the possibility to submit the continuous integration jobs directly on the INFN Tier-1 hosted at CNAF, leveraging the sharing facilities natively present in a modern batch system.

In parallel, the availability of the service needs to be properly promoted in an environment—INFN—that is very distributed geographically and where communication is not always easy.

5. Conclusions

The success of a scientific experiment relies more and more on sound computing practices, notably in the development of scientific software.

The infrastructure presented in this paper aims at providing a one-stop shop for software developers, especially members of small- and medium-size experiments, where they can find state-of-the-art tools and services that help them deliver software of increasing quality at reducing costs and on time, through the adoption of established best practices.

References

[1] The ISSS Project *Infrastructure in Support of Software Development* `https://web.infn.it/isss`
[2] KVM *Kernel Based Virtual Machine* `http://www.linux-kvm.org`
[3] pfSense `http://www.pfsense.org/`
[4] FreeNAS `http://www.freenas.org/`
[5] oVirt `http://www.ovirt.org/`
[6] δ-Cloud `https://deltacloud.apache.org/`
[7] Amazon Elastic Compute Cloud `https://aws.amazon.com/ec2/`
[8] Atlassian JIRA `https://www.atlassian.com/software/jira`
[9] Jenkins `https://jenkins-ci.org/`
[10] Subversion `https://subversion.apache.org/`
[11] git `http://git-scm.com/`
[12] mercurial `http://mercurial.selenic.com/`
[13] GitHub `https://github.com/`
[14] Bitbucket `https://bitbucket.org/`