

# Common software lifecycle management in external projects

C. Duma<sup>1</sup>, A. Costantini<sup>1</sup>, D. Michelotto<sup>1</sup>, P. Orviz<sup>2</sup> and D. Salomoni<sup>1</sup>

<sup>1</sup>INFN Division CNAF, Bologna, Italy

<sup>2</sup>IFCA, Consejo Superior de Investigaciones Cientificas-CSIC, Santander, Spain

E-mail: [ds@cnafe.infn.it](mailto:ds@cnafe.infn.it)

## Abstract.

This paper describes the common procedure defined and adopted in the field of software lifecycle management, the Continuous Integration and Delivery systems setup to manage the new releases, as a first step to ensure the quality of the provided solutions, services or components, while strengthening the collaboration between developers and operations teams among different external projects. In particular, the paper analyses the common software lifecycle management procedure adopted in two EC funded

project: eXtreme DataCloud and DEEP Hybrid DataCloud.

## 1. Introduction

A relevant activity in the software-oriented projects is the definition and implementation of the entire Software Lifecycle Management process. As the software components envisaged by the project have a history of development in previous successful European projects implementing different types of modern software development techniques, the natural choice was to complement the previous, individual, Continuous Development and Integration services with a Continuous Testing, Deployment and Monitoring as part of a DevOps approach:

- Continuous Testing - the activity of continuously testing the developed software in order to identify issues in the early phases of the development. For Continuous testing, automation tools will be used. These tools enable the QA's for testing multiple code-bases and in parallel, to ensure that there are no flaws in the functionality. In this activity the use of Docker containers for simulating testing environments on the fly, is also a preferred choice. Once the code is tested, it is continuously integrated with the existing code.
- Continuous Deployment - the activity of continuously updating production environment once new code is made available. Here we ensure that the code is correctly deployed on all the servers. If there is any addition of functionality or a new feature is introduced, then one should be ready to add resources according to needs. Therefore, it is also the responsibility of the SysAdmin to scale up the servers. Since the new code is deployed on a continuous basis, automation tools play an important role for executing tasks quickly and frequently. Puppet, Chef, SaltStack and Ansible are some popular tools that could be used at this step. This activity represents the Configuration Management - the process of standardising the resources configurations and enforcing their state across infrastructures

in an automated manner. The extensive use of Containerisation techniques will provide an entire runtime environment: application/service, all its dependencies, libraries and binaries, and configuration files needed to run it, bundled in one package - container. T3.1 will also manage the scalability testing, being able to manage the configurations and do the deployments of any number of nodes automatically.

- Continuous Monitoring - very crucial activity in the DevOps model of managing software lifecycle, which is aimed at improving the quality of the software by monitoring its performance. This practice involves the participation of the Operations team who will monitor the users' activity to discover bugs or improper behaviour of the system. This can also be achieved by making use of dedicated monitoring tools, which will continuously monitor the application performance and highlight issues. Some popular tools useful in this step are Nagios, NewRelic and Sensu. These tools help to monitor the health of the system proactively and improve productivity and increase the reliability of the systems, reducing IT support costs. Any major issues found could be reported to the Development team to be fixed in the continuous development phase.

These DevOps activities are carried out on loop continuously until the desired product quality is achieved. Automation will play a central role in all the activities in order to achieve a complete release automation, moving the software from the developers through build and quality assurance checks, to deployment into integration testbeds and finally to production sites part of the Pilot Infrastructures.

## **2. Software Quality Assurance and Control**

Software Quality Assurance (SQA) covers the set of software engineering processes that foster the quality and reliability in the software produced. The activities involved in this task are mainly focused on:

- Defining and maintaining a common SQA procedure to guide the software development efforts throughout its life cycle.
- Formulating a representative set of metrics for the software quality control to follow up on the behaviour of the software produced, aiming to detect and fix early deviations in the software produced.
- Enabling a continuous integration process, eventually complemented by a continuous delivery scenario, promoting the automation adoption for the testing, building, deployment and release activities.

In order to define the SQA process, the specific context of the software developed in the project has to be taken into account. The following particularities characterise the corresponding development teams:

- Heterogeneous developer profiles: different backgrounds and different degrees of expertise.
- Geographically distributed.
- Different home institutes which implies different cultures, different development technologies, process and methods.
- High turnover due to the limited duration of the projects where the grid software has been developed so far.
- More focus on development activities, with limited resources, if any, available for quality assurance activities.

The Quality Assurance process has to take all these factors into account to define the Software Quality Assurance Plan (SQAP). A set of "QA Policies" have also to be defined to guide the

development teams towards uniform practices and processes. These QA Policies define the main activities of the software lifecycle, such as releasing, tracking, packaging and documenting the software carried out by the project. This is done in collaboration with development teams, making sure they are flexible enough to co-exist as much as possible with current development methods. The SQA activities have to be monitored and controlled to track their evolution and put in place corrective countermeasures in case of deviations.

Finally, in order to measure and evaluate the quality of the software, a quality model have to be defined to help in evaluating the software products and process quality. It helps to set quality goals for software products and processes. The Quality Model has to follow the ISO/IEC 25010:2011 “Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models” [R18] to identify a set of characteristics that need to be present in software products and processes to be able to meet the quality requirements.

### **3. Software Maintenance and Support**

Regarding the software maintenance and support area of the software lifecycle management the main objectives covered by the project are:

- To increase the quality levels of the software by contributing to the implementation and automation of the Quality Assurance (QA) and Control procedures defined by the project.
- To boost the software delivery process, relying on automation.
- To emphasize the communication and feedback with/from end users, in order to guarantee adequate requirements gathering and support.
- To guarantee the stability of services already deployed in production and the increase of their readiness levels, where needed.

Moreover the common practices deal with the definition of all processes and procedures regarding the software maintenance and support, and their continuous execution:

- Software Maintenance - regarding software preparation & transition from the developers to production repositories and final users.
- Problem Management - providing the analysis & documentation of problems.
- Change Management - control code, configuration changes, retirement calendars.
- Coordination the provisioning of adequate support to released software.
- Responsible for the release management and coordination and the maintenance of the artefacts repositories, defining policies and release cycles.

The plan regarding the software maintenance and support management have to follow the guidelines of the ISO/IEC 14764:2006 standard [R30], and includes a set of organizational roles and administrative roles to handle maintenance implementation, change management and validation, software release, migration and retirement, support and helpdesk activities. Component releases are classified in major, minor, revision and emergency, based on the impact of the changes on the component interface and behaviour. Requests for Change (RfC) are managed adopting a priority-driven approach, so that the risk of compromising the stability of the software deployed in a production environment is minimized. The User Support activity deals with the coordination of the support, to users of the software components developed within the project and included in the main project software distributions.

#### 4. Services for continuous integration and SQA

A set of tools and services are needed to support the PTs, the Software Quality Assurance, the Continuous Integration and the software release and maintenance. The choice of using publicly available cloud services has three main reasons:

- Higher public visibility and in line with project objectives for open source software
- Provides a path to further development, support and exploitation beyond the end of the project.
- Smaller effort needed inside the project to operate and manage those services.

The list of services needed is given below with a small description for each service and the Web link.

Service	Product	Description
Project Issue Dashboard	<a href="https://www.atlassian.com/software/jira">https://www.atlassian.com/software/jira</a>	<b>Jira</b> as issue tracking product and project management tool. It provides bug tracking, issue tracking, and project management functions.
Project Management/Wiki	<a href="https://www.atlassian.com/software/confluence">https://www.atlassian.com/software/confluence</a>	<b>Confluence</b> as collaborative tool. It provides collaboration workspaces for knowledge exchange, social networking, personal information management and project management.
Project Documentation Repository	<a href="https://nextcloud.com/">https://nextcloud.com/</a>	Repository for easy document sharing
Events Management	<a href="https://getindico.io/">https://getindico.io/</a>	Open source tool for event organization
Source code repository and version control	<a href="https://github.com">https://github.com</a>	GitHub is a Version Control System.
Continuous Integration	<a href="https://jenkins-ci.org">https://jenkins-ci.org</a>	<b>Jenkins</b> for the Continuous Integration service. It will deploy, manage and support a Jenkins server to execute automatically most of the SQA tests. The code review and the documentation checking items are not automated.
Artefacts repository for docker images	<a href="https://hub.docker.com/">https://hub.docker.com/</a>	DockerHub organizations and repositories
Automatic deployment and configuration	<a href="https://galaxy.ansible.com/">https://galaxy.ansible.com/</a>	Deployment and configuration management tool
GitHub Pull Requests (PR)	<a href="https://help.github.com/articles/about-pull-requests/">https://help.github.com/articles/about-pull-requests/</a>	Service for source code review

#### 5. DevOps adoption from user communities

#### 6. Conclusions

The experience gathered throughout the project with regards to the adoption of different DevOps practices is not only useful and suitable for the software related to the core services in the INDIGO-DataCloud solution, but also applicable to the development and distribution of the applications coming from the user communities.

#### Acknowledgments

DEEP-HybridDataCloud has been funded by the European Commission H2020 research and innovation program under grant agreement RIA XXXXXXXX. eXtreme DataCloud has been funded by the European Commission H2020 research and innovation program under grant agreement RIA XXXXXXXX.