Using git for configuration management and secured centralized deployment

Daniel Biella[1], Jörg Mathieu[2] , Sven Radermacher[3]

[1] University of Duisburg-Essen, Forsthausweg 2, 47048 Duisburg, daniel.biella@uni-due.de
[2] University of Duisburg-Essen, Forsthausweg 2, 47048 Duisburg, joerg.mathieu@uni-due.de
[3] University of Duisburg-Essen, Forsthausweg 2, 47048 Duisburg, sven.radermacher@uni-due.de

Keywords
Configuration management, Information security, git, HISinOne

1. ABSTRACT

This article explains a best practice approach to configure, deploy and operate the web-based campus IT system HISinOne in a heterogeneous server environment by means of distributed version and configuration control, and a centralized software repository, using common tools, such as git and ssh.

2. THE CAMPUS IT SYSTEM HISINONE

HISinOne is a web-based student lifecycle management IT system with many installations in Germany (www.hisinone.de). The system covers the complete student life cycle. It is mainly programmed in Java with parts written in Ruby and runs as a Tomcat-based Java8 web application. The development of HISinOne is centrally governed by HIS eG, a registered cooperative society, in which a majority of German HEIs are represented as members. Each HEI operating HISinOne must configure and scale the IT system to its particular needs. While some part of the configuration settings are exclusively stored in a central database, other configuration parameters are stored in configuration files. In addition, the software supports so called extensions, which allows customers to create their own java extensions using an API. In larger HEIs, the software is usually operated in a server cluster for reasons of load-balancing and to implement dedicated subclusters with limited role-based access by configuring a minimized set of roles with access rights in these logical subclusters.

In small scaled environments (i.e. without logical subclustering), the software can be operated mostly with a set of built-in configuration management tools. However, more individualized, complex and security-aware operation scenarios require specialized machine-related configuration settings in the configuration files. Furthermore, changes in a server cluster exceeding a dozen of servers cannot be applied manually for reasons of system integrity.

The University of Duisburg-Essen was one of the first HEIs to rollout HISinOne for student management as described by Biella et al. (2012, 2013). The software is operated in a testing, staging and production environment. In terms of a software configuration management workflow, we want to highlight a best practice approach focusing on version and configuration control, collaboration support, branching, and deployment.

3. VERSION AND CONFIGURATION CONTROL

Software version control is usually applied in software development to uniquely identify the state of a software product within a development cycle. If applied to a software that already went through a vendor's QA development cycle and only lacks customer-specific configurations, the process is rather called configuration control. As HISinOne is a web application, we use git as the software version control system with a centralized server-based repository. There are four layers of data that can be identified and that need to be clearly separated and organized in this customer-specific configuration process: the software distributed "as-is" by the vendor (layer 1), file-based changes that overwrite this data (layer 2), files that extend the existing code base, i.e. customer-specific

configurations or java extensions (layer3), and individual server-related files (layer 4). All files from layers 1 to 4 define a configured clustered versioned state of HISinOne for the repository. All states, i.e. commits, are kept in the central repository, by which fast and tracable rollbacks are facilitated, especially when handling a large number of changed files.

4. COLLABORATION

Due to the layered structure, modifications by multiple experts in a distributed software version system require documentation and communication skills to keep merging efforts and issue handling at a minimum. It is crucial to establish an awareness for documentation. Commenting commits has helped to improve the change documentation and allows the system administrators to easily track the changes committed by co-workers in a collaborative environment.

5. BRANCHING

Branching is a mechanism for creating a new version instance within the git repository. The creation of a separate branch in configuration control is usually triggered by a new major software release or a software patch distributed by the vendor (i.e. changes in layer 1), changes in the HEI's configuration or extensions (layers 2 or 3), or changes in the server-related configuration (layer 4). In all cases, the code integrity of the layered structure of layers 1 to 4 must be re-established accordingly with all customer-specific requirements, in a process called "merging". Merging can be challenging in layer 2 as there may exist up to three mutually different versions of a file that have to be merged into one new file ("cherry-picking process"). In layers 3 and 4, customer-specific settings have to be merged and cross-checked with regard to the latest configuration specifications by the vendor. Finally, the "new" branch can be used to deploy a new testing environment and an iterative test scenario can be initiated. Once tested, approved and staged, the branch becomes the new "master" or production branch and can be deployed to the production environment.

6. DEPLOYMENT

The deployment of a web application running in a Tomcat server cluster implies that a pre-defined set of files (i.e. the approved commit) is copied to a dedicated folder on each application server. Although git is a versioning tool following a distributed development paradigm, we use a centralised git repository server for deployment. The repository contains a very small set of files, which is individual to each application server and, hence, excluded from the automated distribution. These server-specific files do not change very often and are copied manually, if required. Logical subclustering is basically implemented using these individual configuration files by shaping a servers "task" within the cluster using a server-specific HISinOne module configuration and their corresponding role-based access restrictions only.

The general deployment workflow for a production server environment starts with a local git checkout on the central git repo server using the master branch. When finished, ssh connections are opened to each web application target server, which trigger a remote script execution on each server. The remote script synchronizes the contents in the "webapps" folder using rsync via a secured connection, with minimal set of local rights and with regard to the exclusion list specified in order to keep the individual layer 4 files unchanged.

7. SUMMARY AND OUTLOOK

The configuration management and deployment best practice described here, has been successfully used for more than one year. It is planned to add more parameters to the deployment shell scripts, such as an individual "exclude file list" and the branch name. We also consider the use of gitlab.

8. REFERENCES

Azhdari, S., Biella, D. & Blotevogel, U. (2012). Exam management and curriculum implementation with HISinOne – A status report. *Proceedings EUNIS 2012*

Biella, D., Blotevogel, U., Ney, M. & Radermacher, S. (2013). Student management with HISinOne – A status report. *Proceedings EUNIS 2013*

9. AUTHORS' BIOGRAPHIES

Dr Daniel Biella is working for the Centre of Information and Media services at the University of Duisburg-Essen (UDE). He is responsible for the IT operation of the campus and resource management systems (ZIM-CR), including HISinOne and SAP among others. He also works in various projects that involve web-based information systems and XML-based metadata standards. In addition, he has been working as lecturer in information visualization and web-based 3D learning museums.

M.Sc. Sven Radermacher is working for the Centre of Information and Media services at the University of Duisburg-Essen (UDE) as team leader campus management (CM) in the ZIM-CR department. He has been working with IT systems for 13 years and was one of UDE's IT system administrators piloting the software HISinOne in 2009.

B.Sc. Jörg Mathieu is working as IT system administrator for the Centre of Information and Media services at the University of Duisburg-Essen (UDE), in the ZIM-CR department. In addition to system configuration, his focus is on the security infrastructure of the CM system HISinOne, including topics like reverse proxying, secure deployment techniques, and firewalls.