CONFIGURATION MANAGEMENT OF THE CONTROL SYSTEM

Vincent Hardion, Darren Paul Spruce, Mirjam Lindberg, Antonio Milan Otero, Julio Lidon-Simon, Jerzy Jan Jamroz, Andreas Persson, MAXIV Laboratory, Lund, Sweden

Abstract

The control system of big research facilities like synchrotron involves a lot of work to keep hardware and software synchronised to each other to have a good coherence. Modern Control System middleware Infrastructures like Tango use a database to store all values necessary to communicate with the devices. Nevertheless it is necessary to configure the driver of a PowerSupply or a Motor controller before being able to communicate with any software of the control system. This is part of the configuration management which involves keeping track of thousands of equipments and their properties. In recent years, several DevOps tools like Chef, Puppet, Ansible or SpaceMaster have been developed by the OSS community. They are now mandatory for the configuration of thousands of servers to build clusters or cloud servers. Define a set of coherent components, enable Continuous Deployment in synergy with Continuous Integration, reproduce a control system for simulation, rebuild and track changes even in the hardware configuration are among the use cases. We will explain the strategy of MaxIV on this subject, regarding the configuration management.

INTRODUCTION

The term configuration refers to the actions that need to be taken for a software to run in a coherent state with its environment without any intrinsic changes involving development.

Device Re-partition

The MAX IV control system is based on Tango for the middleware layer and Sardana for the application layer, with it's Taurus Framework as the graphical user interface and additionally spock as the command line interface.

Each beamline and the Machine will have their own control system on different separated networks.

The linac will be composed by around 1500 Tango devices to control 75 different types of device. A majority of the physical values will be monitored with PLCs and also, most of the controllers will be driven by TCP/IP communication.

Configuration of the Control System

The different types of software in the MAX IV Control System consist of a set of operating systems, network services and base frameworks such as Tango, device drivers, libraries, Tango devices and applications such as GUIs, Sardana macros and controllers which all need to be deployed and configured before usage. The configuration is not only necessary for the installation but also during the lifetime of each component when a new version update needs to activate new functionalities.

Any difference of configuration between computers is a potential risk the consequence being to change the behaviour of a piece of software. To avoid these differences, a best practice in the Continuous Integration process is to compile and test in the environment closest possible to the target platform[1]. This can be done by cloning a production server but a preferable method is to track the minimal dependencies and configuration.

EXPERTISE DEVELOPMENT AS STRATEGY, AUTOMATION AS A TOOL

The first principle for the long term strategy of the Kontrol and IT Support (KITS) group at MAX IV is to develop the expertise of the group. This is inspired by the Toyota way[2]. One action involves eliminating repetitive manual interventions which in the end bring little value.

Defining the necessary entry points of configuration is one of the tasks involved in building a piece of software. During the development, the configuration of the test instance is done manually. But on a deployment-wide scale each manual operation is (time consuming). This is a common pattern when the information must pass through different levels of the control system.

For example a Tango device is developed to communicate with a PLC, a field bus controller, but each instance is configured differently. The MAX IV Linac vacuum system is managed by several Allen Bradley PLC controllers. Each Tango device needs to be configured with the IP address of the PLC but also with a list of tags to expose around 1000 parameters. The automation of this configuration is going to dramatically reduce a repetitive manual task.

Conditions

There are several conditions to fulfill in order to achieve the automatic deployment of the configuration:

- the information needs to be centralised
- the medium has to be independent from the system that is to be configured.

Pros

Several advantages to have automation of the configuration are:

- to greatly reduce the time to deploy
- to keep track of the essential configuration version if a content versioning system is used (GIT, SVN, ...).
- to keep coherence between configuration data and the process using it by considering the version of a software component as part of the configuration management.
- apply an additional configuration entry while at the same time being able to upgrade to the last version of the software which is able to process it (e.g. new property for a Tango device server, ...)

ISBN 978-3-95450-139-7

- to make it easy to refactor the data organisation applying change incrementally, reducing the risk of errors.

For the Vacuum system of the MAX IV linac, we use a generic Tango device, PyAttributeProcessor, that exposes any PLC parameter as an attribute. We can first deploy one Tango Device per section including pressures and temperatures and finally after the commissioning decide to have one Tango device for the entire Linac with all of the pressure values and another Tango device for the temperature values.

CONFIGURATION MANAGEMENT STRATEGY

"Software Configuration Management (SCM)[1] is the task of tracking and controlling changes in the software, part of the larger cross-discipline field of configuration management."[3]. From the OS to the Tango Device all layers have to be managed before it is possible to truly control the configuration of the Control System.

Operating System

The Operating System was one of the first software layer employing SCM tools to control third party application installation and to ensure coherence. Besides package managers like RPM or APT for Linux which control the version of software components and their initial configuration, some independent OS tools were developed to manage the configuration of distributed software during its entire lifecycle. CFEngine[4] was the first modern open source tool released in 1993 and is part of the top 3 used in the OSS sphere together with Puppet[5] and Chef[6].

For the MAX IV Control System several criteria have been defined for the choice of the configuration management software:

- Open Source: with an established community and support from a consultant company
- Portability between Linux distributions: the standard OS of the MAX IV control system is CentOs 6 but some equipment comes with other flavours of embedded Linux.
- Integration: as much as possible with Python as it will represent ¾ of the future developments of the CS team. (But in any case the language used by the software itself could not be a deal breaker).
- Evolving: to be able to cover the configuration of the entire control system layer.
- Developer friendly: to reduce the amount of system administration for the control engineer.

For the functionality:

- Idempotent: the capacity to check if a system is compliant with the reference without needing to modify it. The same operation applied several times has the same consequence as if applied once.
- Stateless: the application should not leave tracks in the operating system to avoid any memory effects. The configuration is held in one place.
- Small Footprint: the deployment is reduced to a minimum of servers to avoid spending much time managing the configuration management system. The system should come with a minimal dependency set.

As the development of the control system software at MAX IV is based on a Lean[2] strategy, the preference is for the tools with minimal functionality corresponding to the specification. The complexity only grows with tools that add a lot of additional, non-relevant functionality.

Tools Choice

Spacewalk[7] supported by RedHat has been the first tool studied to understand the benefits of the configuration management for the control system. Its functionality extends beyond the configuration management scope by also providing monitoring, and OS installation. In parallel, Chef and Puppet were those tested from the list of possible tools[8] as they are supported by a huge community. These alternatives focused only on the configuration, without any database requirement and both are portable (Ruby based). Both can work in push only mode (in contrast to client/server mode) with a small footprint on the managed servers. They provide a complete set of functionality which provide the checking of targeted systems with the referential configuration. The installation and use of these tools requires a solid knowledge of the Ruby platform, thus increasing the learning curve. So, the Python based tools have been investigated to see if they could provide the same functionality. SaltStake[9] and Ansible[10] are 2 possible solutions but only the last has been tested in real conditions. The quick start and the simplicity differs from the previous tools: Ansible needs only SSH and Python, installed by default in recent Linux distributions.

ANSIBLE

With Ansible the configuration is written in plain text and only a command line is necessary to run the application in parallel on all controlled computers.

Inventory

A text file so-called "Inventory" keeps the list of computers and their roles inside the control system.

ISBN 978-3-95450-139-7

Figure 1: Ansible Inventory file of MAX II.

The Figure 1 shows how we have managed to configure a Machine network and the different roles in a Tango control system which are shared with the beamline networks:

- the Tango database,
- the server which runs the Tango devices,
- the client computer which runs GUIs and CLIs.

The inventory variables are used to distinguish the configuration between the control systems. In the Figure 1, "[machine:vars]" contains the variable "tango_host" which will be applied to any control system computers of the Machine.

Ad-hoc Command

The *ad hoc* command is useful to complete an action on several computers in the same time. Ansible uses the inventory file to include the computers in the execution list. Ansible comes with a predefined list of action to execute a shell command, to install a package, to start a service, etc. There are also third party modules for managing specific applications such as MySQL.

The Figure 2 shows an example of how to check the Red Hat version of the control system computers. Ansible continues the execution of this command, even if one computer is unreachable and until each computer has returned an answer.

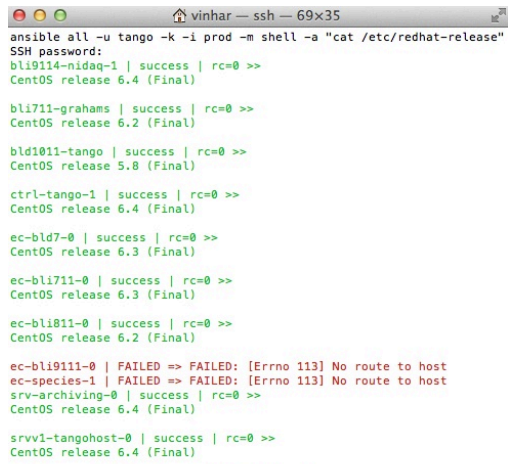

Figure 2: Ad hoc Ansible command to check the CentOS version.

Playbook

An Ansible playbook is used to keep the different actions needed to reach a configuration state instead of repeating them manually with an *ad hoc* command. The playbook is used with the inventory file during the execution of Ansible. The playbooks are written in yaml format.

The Figure 3 shows the definition of one playbook to check or to set the common configuration of Tango for a database, a server or a client. Here the "tango-common" RPM package and the TANGO_HOST environment variable represent the minimal configuration.

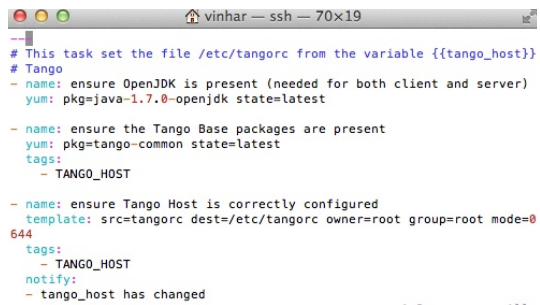

Figure 3: Playbook for checking the TANGO_HOST value.

ISBN 978-3-95450-139-7

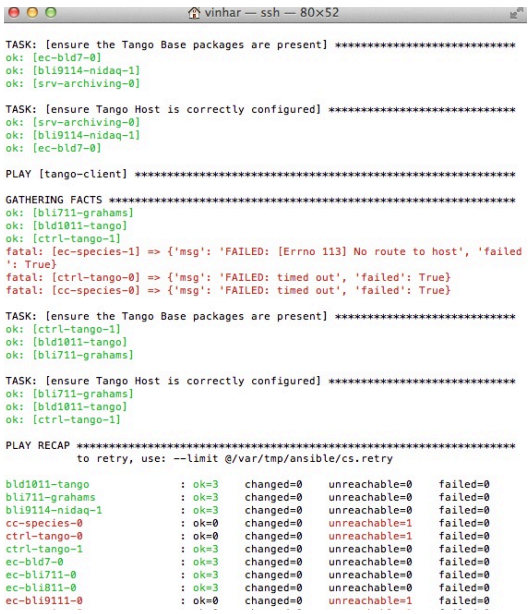

Figure 4: execution of the tango common playbook.

A complete report of what has been changed to reach the expected configuration is displayed at the end of the playbook execution. In the Figure 4 the green line indicates that no action has been done as the current configuration of the control system computer is correct.

FUTURE DEVELOPMENT

The decision to continue the test in depth has been taken after installing Ansible and successfully completing the configuration of 3 laptops with all the CS client applications in only 1 day of development. The next days we were able to control most of the MAX II Machine and beamlines.

Automatic Deployment

The ability of Ansible to work with the application layer gives the possibility to deploy any software items. By setting the continuous Integration server to execute an Ansible playbook any in-house developments can be deployed automatically after their release.

Tango Configuration through Ansible

The next step will be to use the Ansible model of configuration management to set up the Control System. Thanks to the plugin system of Ansible, which can be in Python or other languages, we will be able to connect with the Tango tools. The idempotent behaviour will be the main task to adapt the existing components.

This could lead to a perfect coherence between the version of a software, the location where it is running and the correct configuration related to the local hardware.

CONCLUSION

The first installation of the MAX IV Control System, planned for Q4 2013, will involve 1500 Tango devices to configure within a short time frame. For this the first priority was to establish a process of gathering the configuration information and then how to use them to deploy the Tango representation of the hardware equipments. After knowing only how to proceed manually the software engineers are currently developing an automatic way to configure an entire Tango Database.

We found some obstacles to the full automation: some equipment comes with proprietary and graphical utilities, mandatory for changing the IP address but difficult to integrate without API in Ansible.

Ansible has already been executed in the production server to change the MAX II's Machine Tango database computer and by consequence to change the TANGO_HOST every where. Also it helped a lot for the migration of Tango to the version 8.

Because ~10% of our time is allocated to infrastructure improvement, Ansible allows us to iterate step by step adding new computers or new services when needed. In this context we will try to fully integrate the Tango configuration to Ansible to keep all configuration in one single point.

REFERENCES

[1] V. Hardion et al, Assessing Software Quality at Each Step of its Lifecycle to Enhance Reliability of Control Systems, SOLEIL, Gif-sur-Yvette France, ICALEPCS'11, Grenoble (2011)
[2] The Toyota Way, Jeffrey Liker, McGraw-Hill Education (India) Pvt Limited, Mar 1, 2004
[3] http://en.wikipedia.org/wiki/Software_configuration_management
[4] http://cfengine.com/
[5] http://puppetlabs.com/
[6] http://www.opscode.com/chef/
[7] http://spacewalk.redhat.com/
[8] http://en.wikipedia.org/wiki/Comparison_of_open_source_configuration_management_software
[9] http://saltstack.com/
[10] http://www.ansibleworks.com/

ISBN 978-3-95450-139-7