ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEPHA040

IRFU EPICS ENVIRONMENT

J.F. Denis, A. Gaget, F. Gohier, F. Gougnaud, T.J. Joannem, Y. Lussignol
IRFU, CEA, Université Paris-Saclay, Gif-sur-Yvette, France

Abstract

The 3 years collaboration with ESS* at Lund (Sweden) has given us the opportunity to use new COTS hardware and new tools. Based on that experience, we have developed the IEE (IRFU** EPICS Environment) by retaining relevant and scalable ESS solutions. This platform centralized several functionalities, fully installed by scripting, on a server that is running on a virtual machine. The functionalities are an EPICS environment and the root file system with the kernel for each embedded systems. In order to provide homogeneous EPICS modules between all collaborators, a template was designed and used as containers for new developments. Furthermore, a development and a production workflow is also proposed and strongly recommended. Due to the current responsibility of CEA IRFU to provide an EPICS platform for SARAF** at Tel Aviv (Israel), IEE was chosen as the standard platform for the whole accelerator. This paper will present the new standard IRFU EPICS Environment based on MTCA and virtual machines.

INTRODUCTION

Based on our experiences of the standard platform conception for SPIRAL2 (France) and IFMIF[1] (Japan) project, and contributing to the ESS[2] (Sweden), CEA decided to combine state of the art technologies as ESS and our expertise acquired on IFMIF and SPIRAL2[3] to create the new standard platform "IEE": Irfu EPICS Environment. The main idea of this platform is to propose a generic methodology for future projects in keeping an EPICS environment centralized on a server (IEE server) shared it with all the clients fully parametrized and automatically installed. A light and standalone version of the IEE server can also be provided for development on standalone PCs. This platform is used for the SARAF[3] (Israel) project.

HARDWARE CONTEXT

Hardware

The current hardware standard for our EPICS projects are the following (see Table 1.), based on mTCA.The standardized crates are the NATIVE-R2/NATIVE-R5 from NAT with the MCH-PHYS80 and the Intel CPU COMex3.

Fast (Up to 250 MS/s) and semi-fast (Up to 5 KS/s) acquisitions for beam diagnostics and RF signals acquisition are assured respectively by the FMC boards IOxOS ADC_3110/3111 and ADC_3117. These boards are plugged on a IOxOS IFC1410 motherboard which integrates an internal FPGA with a customizable area for user specific functionalities.

The timing system is based on MRF boards, the mTCA-EVM300 for the Event master and the mTCA-EVM300U for the Event receiver with delay compensation.

For remote I/Os, our choice remained the cheap and convenient Beckhoff EtherCAT modules with communication based on Modbus/Tcp.

This hardware has been integrated in IEE, but more hardware can be easily added.

Table 1: Current Hardware Used With IEE

Designation	Reference	Tender
mTCA Crate	NATIVE-R2/ NATIVE-R5	NAT
MCH	MCH-PHYS80	NAT
CPU Intel	RTM COMex3	NAT
CPU carried board	IFC1410	IOxOS
Fast Acquisition (8ch., 250 MS/s)	ADC311	IOxOS
Slow Acquisition (20 ch., up to 5 MS/s)	ADC3117	IOxOS
Timing System Master	mTCA-EVG300	MRF
Timing System Receiver	mTCA-EVR300U	MRF

IEE SOFTWARE DESCRIPTION

EPICS

The "EPICS Software Distribution" is a selection of "Base", "IOC support modules" and "extensions" exists on official EPICS websites. The currently used EPICS Base is R3.15.4. The IOC support modules are categorized as either Software Support or Hardware Support. The extensions are host tools and Channel Access clients such as Control System Studio (CSS), Archive Appliance, CA libraries (Java, Python…).

The development environment fully relies on the standard "EPICS Build Facility". EPICS software can be divided into multiple <top> areas. An example of a <top> area is the EPICS Base itself. Each <top> may be maintained separately. A <top> directory structure essentially contains the build configuration files in a configure folder and a Makefile. The GUIs are made using CSS BOY.

The Irfu EPICS Environment provides a standard development model that all the developers involved in our control system software team have to follow. These

*ESS, https://europeanspallationsource.se/
**IRFU, https://irfu.cea.fr/en/
***SARAF, http://soreq.gov.il/mmg/eng/Pages/SARAF-Facility.aspx

standards give the necessary homogeneity to the software modules produced.

The IEE environment is installed on the server inside the directory /iee/, and contains the subfolders listed in Table 2.

Table 2: Main Directories Provided by the IEE Server

Designation	Reference
Base	The central core of the EPICS control system toolkit
Toolchains	Cross compilation toolchains for all embedded system supported
Extensions	Java and Python CA libraries
Iocs	All stables IOCs used in production
Modules	All stables modules used in production
Data	Writable directory to share data between server and embedded system,.
Startup	Boot scripts to configure embedded system during the boot. Also contains startup scripts to start IOCs after boot.
Support	Support modules from the EPICS community (Modbus, StreamDevice, S7PLC…)
Tops	Directory shared between development PCs and embedded systems. Inside, each developer has a private directory and can test his developments directly on the embedded system.

Linux

By default, Linux is the operating system installed on each type of hardware. Except the mTCA system, the last version of Centos is used. The PowerPC of the mTCA board involves to build a specific Linux with a cross compilation toolchain. This distribution is based on FSL-Qoriq from NXP and is built with the tool Yocto.

VIRTUALIZATION

The IEE server is the main part of the infrastructure. Its main goal is to provide an EPICS environment, to distribute all sort of necessary files like root file systems or kernel binaries, and run some essential services such as DHCP, NFS... needed by different kinds of clients. It can run directly on a physical machine or can be virtualized. Running the server on a virtual environment offers some machine is full virtualized (Operating System & drivers), while for the second one the Operating System is the same as the host machine and only the applications are full isolated from the host, this is called a containerized application (see Fig. 1). Containers consume much less machine resources and it is more powerful, so we decided to use this technology through the software called Docker.

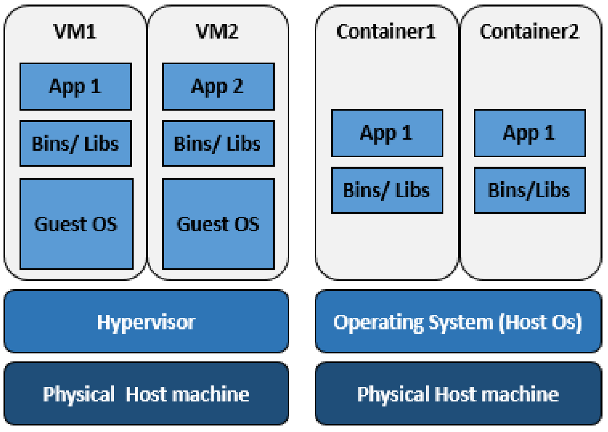

Figure 1: Illustration of the difference between virtual machines and containers.

Virtualization is also interesting in hardware failure. To limit this issue, we decided to create a pool of physical machines, called a cluster. Setting up a cluster enables to use another host machine in case of hardware failure. The main container "IEE server" running into the master server can be duplicated on two slaves servers. In case of physical failure of the master server, one of the slave servers can take the hand automatically.

Hence to make the best of all these virtualization advantages, the IEE server runs inside a Docker container, on a cluster of physical machines. This provides high disponibility of the service and protects against most hardware failures.

A Proxmox test stand was elaborated in order to try this tool. It is composed of three physical machines on which Proxmox was installed (see Fig. 2). These three machines are linked in order to create an IEE cluster. The IEE server runs on the physical master server, and is duplicated every 15s on the two physical slave servers. In case of the master server crashing, the IEE server can restart shortly on one of the slave. Furthermore, a container can also be easily migrated on another physical machine of the cluster, for hardware maintenance or upgrade.

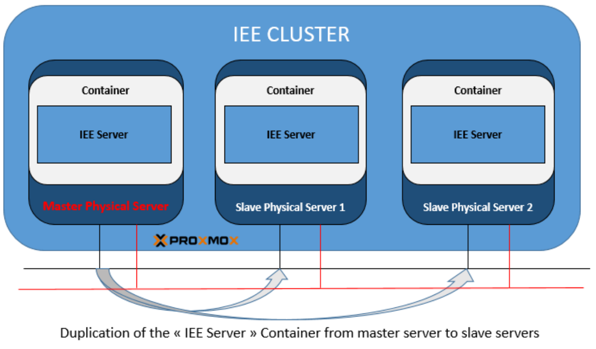

Figure 2: Illustration of the IEE cluster.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEPHA040

AUTOMATISATION

Ansible

Ansible is a tool which enables to automatically configure a machine using scripts and a set of parameters. These scripts are called playbooks and roles. A role is dedicated to set up a specific functionality such as the configuration of a dhcp server. It is developed independently from the playbook. A playbook is composed of one or several roles to be called on an inventory of target machines. In order to be confident about any installation of the IEE server, these roles and playbooks are thoroughly tested using Molecule and versioned on Gitlab. The playbooks and roles are then used to "provision" the various containers and virtual machines available for IEE user and administrators (see Fig. 3).

Cookiecutter

Cookiecutter is a tool to create file and folder tree based on a project template. In our case, we use it to create a role template that includes all necessary files and folder, with for instance a preset test section.

Molecule

Molecule is designed to test Ansible roles. It first creates a working environment via drivers (Docker, Vagrant, etc…), then download and execute the playbook and its roles, and test the result of the execution with tools such as testinfra, Inspec.

Packer

Packer automates the creation of many types of machine image (Ovf, box…). In our case, two types of images are created. Based on the last version of Centos, one is dedicated to run in a Container and another one to run inside a hypervisor such as Virtualbox or VmWare. In both case, the image integrates every package needed for the IEE server which allows to provision the box without internet access. It also enables to ensure homogeneity of the CentOs version installed on each system.

Kickstart

This tool enables to automate the installation of an Operating System on machines. It requires a single file form, "Eq.", if in the text. The equation number is placed in parentheses [e.g., Eq. (1)].

Containing the answers to all the questions that would normally be answered through a graphical interface during the installation. We use it in combination with Packer.

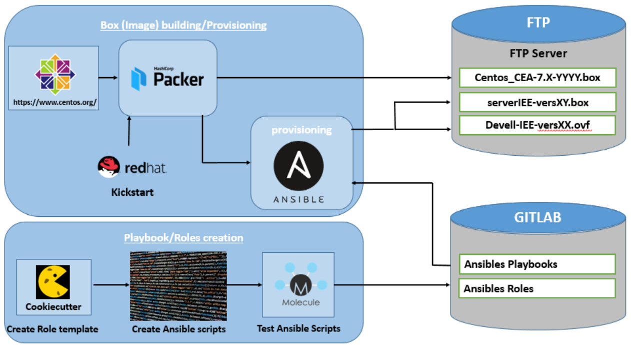

Figure 3: Flow production of machine images.

ENVIRONMENT

In order to avoid any perturbation in production, the development environment is completely distinct. Another IEE server is dedicated to the development (see Fig. 4).

Development Environment

By default, IOCs first load EPICS modules in development, and if nothing is found, they load stable EPICS modules. When the developer is confident about his new feature or patch of the EPICS module, he creates a tag and push his development to Gitlab server.

Production Environment

Only stable EPICS modules, tagged modules puished on the Gitlab server, are used by IOCs running on embedded system. Currently the update process of EPICS modules is done manually, and requires to stop the experiment.

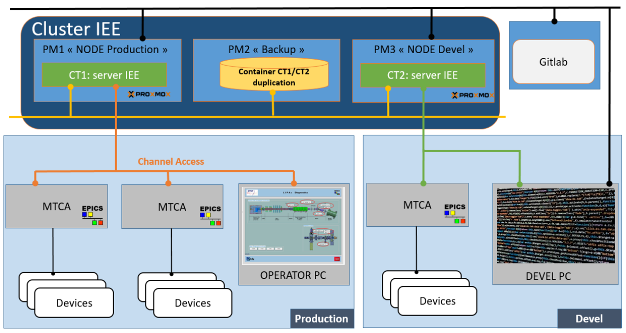

Figure 4: Development and production environment.

REFERENCES

[1] J. Marroncle *et al.*, "IFMIF-LIPAc Diagnostics and its Challenges," in *Proc. IBIC'12*, Tsukuba, Japan, Oct. 2012, paper WECC01, pp. 557-565.

[2] F. Gougnaud *et al.*, "The Implementation of the Spiral2 Injector Control System," in *Proc. ICALEPCS'11*, Grenoble, France, Oct. 2011, paper MOPMU025, pp. 491-493.

[3] F. Gougnaud *et al.*, "Evolution Based On MicroTCA And MRF Timing System," presented at the ICALEPCS'19, New York, NY, USA, Oct. 2019, paper MOPHA052, this conference.