EPJ Web of Conferences **295**, 07028 (2024) https://doi.org/10.1051/epjconf/202429507028
CHEP 2023

Lifecycle Management, Business Continuity and Disaster Recovery Planning for the LHCb Experiment Control System Infrastructure

Pierfrancesco Cifra[1,2,*], *Francesco* Sborzacchi[1,**], *Niko* Neufeld[1,***], and *Luis Granado* Cardoso[1,****]

[1]CERN, Meyrin, 1211 Geneva, Switzerland
[2]Nikhef National Institute for Subatomic Physics, 1098 XG Amsterdam, The Netherlands

Abstract. LHCb (Large Hadron Collider beauty) is one of the four large particle physics experiments aimed at studying differences between particles and anti-particles and very rare decays in the charm and beauty sector of the standard model at the LHC. The Experiment Control System (ECS) is in charge of the configuration, control, and monitoring of the various subdetectors as well as all areas of the online system, and it is built on top of hundreds of Linux virtual machines (VM) running on a Red Hat Enterprise Virtualisation cluster. For such a mission-critical project, it is essential to keep the system operational; it is not possible to run the LHCb's Data Acquisition without the ECS, and a failure would likely mean the loss of valuable data. In the event of a disruptive fault, it is important to recover as quickly as possible in order to restore normal operations. In addition, the VM's lifecycle management is a complex task that needs to be simplified, automated, and validated in all of its aspects, with a particular focus on deployment, provisioning, and monitoring. The paper describes the LHCb's approach to this challenge, including the methods, solutions, technology, and architecture adopted. We also show limitations and problems encountered, and we present the results of tests performed.

1 Introduction

The LHCb online system [1] comprises all aspects of online computing, experiment controls, as well as the Timing and Fast Control [2]. It provides the infrastructure for the data taking for the High-Level Triggers, as well as the control, configuration, and monitoring of the entire experiment, and consists of approximately 4000 physical servers and embedded systems interconnected through high-density routers and distribution switches. Central to the functionality of the LHCb online system is the ECS, which operates autonomously and in complete isolation from the CERN networks. The ECS will manage the configuration, monitoring, and operation of all experimental equipment involved in the experiment's various activities, such as data acquisition (frontend electronics, readout network, Event Filter Farm,

*e-mail: pierfrancesco.cifra@cern.ch
**e-mail: francesco.sborzacchi@cern.ch
***e-mail: niko.neufeld@cern.ch
****e-mail: luis.granado@cern.ch

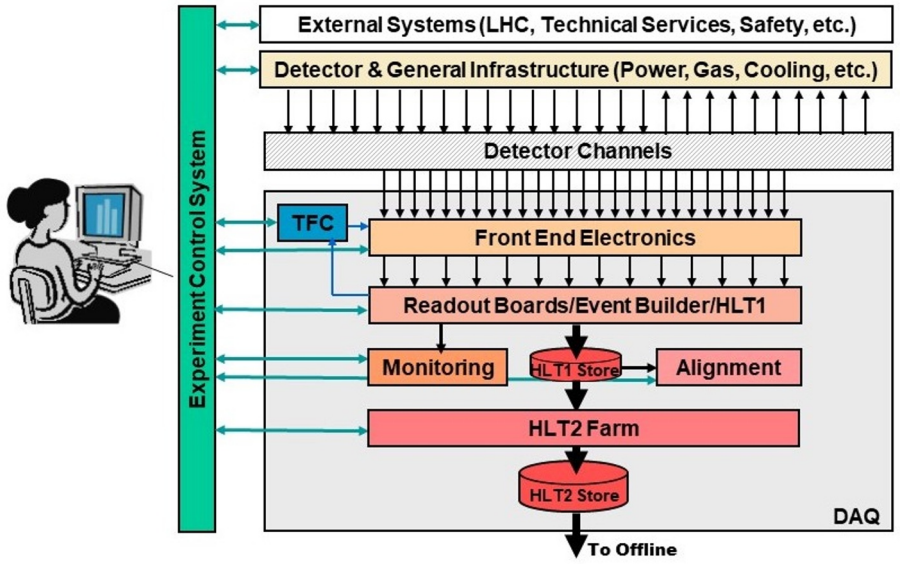

Figure 1. Scope of the Experiment Control System

etc.), detector operations (high and low voltages, temperatures, etc.), experimental infrastructure (magnet, cooling, electricity, detector safety, etc.), and interaction with the outside world (LHC Accelerator, CERN safety system, CERN technical services, etc.). As shown in Fig. 1, the ECS provides a single interface between the users and all experimental equipment. The whole control system runs the standard LHC Supervisory Control and Data Acquisition (SCADA) system, WinCC-OA, a product of ETM/Siemens, on top of Linux VMs. Given its central role, it is important to effectively manage the underling infrastructure, including several aspects of the VM's life-cycle, from provisioning to configuration, with a great focus on monitoring, in order to spot any issues and ensure smooth operations of the ECS. Also, the establishment of a robust disaster recovery procedure should not be underestimated; even if this can be wrongly seen as a very unlikely event, a hardware fault or any human error can easily have an impact on the ECS operations, and in our experience, it is always critical to be prepared and have a well-defined plan to recover from any fault with minimal downtime.

2 The Experiment Control System infrastructure

Virtualisation provides several advantages, such as better resource utilisation, improved management through centralised control, increased availability, and easy adaptation to changing resource needs (e.g. adding CPU or memory to a running system without any disturbance). It can also save time on IT projects by eliminating the need for dedicated hardware and providing standard software configurations. At LHCb, virtualisation, in combination with commercial storage systems, serves as the basic infrastructure for the ECS. This section gives a brief overview of the virtualisation cluster and the storage system of the ECS infrastructure.

2.1 Virtualisation Infrastructure

The virtualisation cluster is based on Red Hat Enterprise Virtualisation (RHEV) [3], which is a management software built on top of the open-source Kernel Virtual Machine (KVM) hypervisor provided by Red Hat. The system is composed of 8 servers, whose hardware specifications are summarised in Table 1. The choice to adopt RHEV as a virtualisation platform was driven by its support and some of the features it gives us, such as redundancy, high availability, and live migration of the VMs. The last one is crucial: in case of a failure on a node, the cluster automatically and transparently migrates the VMs from a faulty node to a healthy one without requiring any manual intervention and with almost no downtime for the ECS.

Table 1. A Virtualiztion Node

Server	Gigabyte MZ92-FS0
CPU	2 x AMD EPYC 7502
Memory	16 x 64 GB DDR4-3200 ECC
OS	RHEL 8.6, Linux 4.18
Network	Mellanox MT28800 ConnectX-5 Ex, PCIeGen4 x16

2.2 Storage Infrastructure

The storage infrastructure is based on a NetApp [4] appliance. The hardware specifications are listed in Table 2. The system serves both as a storage VM and as an NFS share for the WinCC-OA projects. The second one, the most important, is regularly backed up to a remote redundant site. NetApp, in fact, natively supports automatic backup using IBM Tivoli Storage Manager [5]. Another main feature provided by NetApp is inline deduplication: copies of data are eliminated, significantly decreasing storage capacity requirements. For instance, the logical space used by the disks of the VMs is 11.4 TiB, but only 4.85 TiB of physical space is allocated. For the WinCC-OA projects, we have a logical space of 1.55 TiB used with only 1.09 TiB physical space. This comes to us with almost no extra effort and without any impact on the performance, and it helps to minimise the costs for the storage disks. NetApp also provides the capability of having local snapshots; while this is not helpful in case of a fault, it is a useful feature for the ECS developers, who can easily access old or deleted files.

Table 2. Storage node

Controller	AFF-A400
Storage Array	NS224
Disks	12 x X4016A NetApp 3.84TB NVMe SSD Drive
NetApp OS	ONTAP 9.11

The disks used are high quality SSDs, meeting the requirement of the high number of IOPs needed by the volume used to store the WinCC-OA projects, on which we measured an average of 40 kIOPS.

3 Lifecycle Management

Efficiently managing the life cycle of such a mission-critical system is not a trivial task. The complexity arises from the system's scale, complexity, and criticality, where any disruption

or inefficiency can have a significant impact on the experiment's operations. That is why highly organised life-cycle management is required. In this section, the design choice and tools used for the provisioning, configuration, and monitoring of the ECS VMs are shown.

3.1 VM deployment and configuration

Each VM is deployed using Foreman [6] and connected to the RHEV cluster. This gives us an easy way to deploy it, just by setting the name and selecting the hostgroup through the foreman GUI. An important point is the integration between foreman and the DHCP. In our infrastructure, we use the KEA DHCP [7], but there is no official integration between foreman and KEA. To overcome this issue, we developed our internal foreman smart proxy plug-in [8] in order to handle the DHCP registration. VMs are organised per-subdetector, which are mapped to a foreman hostgroup. This allows us to have a different configuration according to the sub-detector to which the VM belongs. The VM is configured using Puppet [9], a software configuration management tool which uses its own declarative language. All the puppet code is stored in a git repository, and it presents a hierarchical structure that can be summarised in three main levels:

1. General LHCb machine configuration (Authentication, logs...)
2. ECS VM configuration (WinCC-OA, packages, NFS mounts...)
3. Sub-detector specific configuration (WinCC-OA Projects, permissions...)

Each VM is associated with a WinCC-OA project, which is not stored on the disk of the VM but on a different volume of the storage system and mounted on the VM through NFS. This choice allows us to separate at the infrastructure level the WinCC-OA project and the VM, and this is useful for several reasons: the WinCC-OA project represents a very important component that requires months of development from the ECS operators, while this is not true for the VM's disk which contains only the operating system (OS) and RPMs packages that can be easily reinstalled with foreman and puppet; it also decouples the high IOPs requirements of the project from the low IOPs requirements of the OS. Each WinCC-OA project is managed by a corresponding systemd unit that we use in order to start and stop the project. The use of this mechanism allows us to have a high-level way of controlling the project and, more importantly, to make a backup before the project is started. This way, we avoid making a live backup of the project, which can lead the corruption of the WinCC-OA database.

3.2 Monitoring

For the monitoring part, we adopt a very common setup: VM metrics are exported using Telegraf [10], collected with Prometheus [11], and displayed through Grafana [12]. To monitor the status of the systemd unit for the corresponding WinCC-OA project, we use a custom telegraf script. From the infrastructure point of view, we are only concerned about its general state, and we don't need to manage any internal ECS error or warning. To ensure a quick intervention by an expert, e-mail alerts are configured to be sent by the Prometheus module Alertmanager every time there is an unexpected behaviour of the VM, such as high CPU usage, high memory usage, or high system load. Alerts are also triggered in the event of an unresponsive VM or an error state of the systemd unit that controls the WinCC-OA project. All the monitorings use open-source software that runs as a containerized application in a Kubernetes cluster.

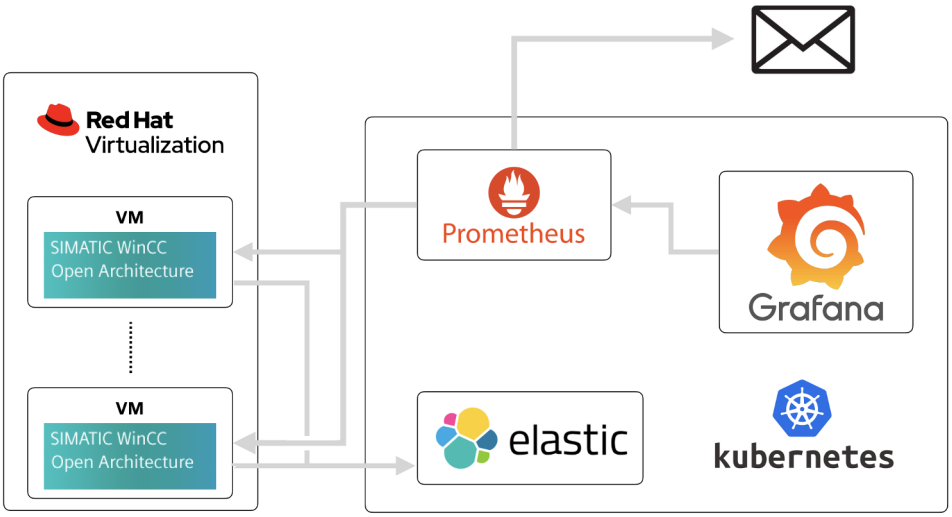

Figure 2. Schematic representation of the monitoring schema.

4 Disaster Recovery

Given the critical role of the ECS, we have a well-defined and automated procedure to recover from a major failure. The goal is to have the VMs and their WinCC-OA-OA projects up and running as quickly as possible. The basic idea is to re-deploy the whole set of control VMs with their SCADA projects. A critical point in the procedure is the WinCC-OA licence linked to the VM; these are machine bound-licenses created by an external licensing tool and identify a machine by its hardware (Ethernet MAC) address. To avoid creating a new license during the recovery, the VMs are deployed with the same identity; in this way, the old licence can be reused. To do this, the VM is directly deployed on the virtualisation cluster with that specific Ethernet MAC address and later registered in foreman. We need to keep track of all the production VMs currently used in the data acquisition. To do that, we rely on an external database in the CERN IT department, using Database On Demand Service (DBOD) [13], which we can consider safe. The information stored in the database is the name of the VM, the subdetector to which it belongs, its MAC address, and the resources allocated to that specific VM, such as the number of CPUs and the amount of memory. The procedure can be summarized in three simple steps:

1. Deploy the VM in the virtualisation cluster with the defined MAC address
2. Register the VM in foreman
3. Start VM, OS provisioning and configuration

It is important to highlight that with this procedure, we are only aiming at the recreation of the control cluster. We are assuming that an RHEV cluster and a foreman instance are up and running and that the basic services such as network, DNS, and DHCP are fully operational.

4.1 Testing the Procedure

We had the opportunity to test the procedure in the production cluster. The main goal of the test was to have a better overview of what is really needed in an emergency situation, what are

the critical steps in the procedure, and what could be improved in order to further minimise the downtime. One of the bottlenecks was the installation of the various RPMs packages needed by the control cluster; this is not a critical issue during the ECS commissioning, but it can significantly slow down the recovery after a major fault while on data acquisition. Our first approach was to use puppet to distribute the packages on the VMs, but we can save a significant portion of time by installing them at OS provisioning time using custom kickstart installation. [14]. One of the main non-critical issues was that many modifications were made by hand by the operators of the ECS. The test was a good opportunity to spot all the changes and add them to the puppet code. That is also why we intend to repeat this test on a regular basis. The whole procedure required 3 hours to reinstall 300 virtual machines, configure them, and have all the WinCC-OA projects running.

5 Conclusion

Foreman in combination with puppet works very well, allowing for easy and very smooth deployment of the VM without requiring any manual configuration other than the VM registration. The monitoring tools suite, along with the alerting system, allows for quick identification of any issue at the infrastructure level. The disaster recovery procedure gives us a backup plan in order to minimise downtime from any disruptive fault caused by hardware, software, or human error. The combination of all the design choices and tools used allowed us to successfully and smoothly commission 300 virtual machines used for LHC Run 3.

References

[1] T Colombo, et al. The LHCb Online system in 2020: trigger-free read-out with (almost exclusively) off-the-shelf hardware. *J. Phys.: Conf. Ser.*. **1085**, 032041 (2018), http://cds.cern.ch/record/2665498

[2] Feo, M., Alessio, F., Durante, P., Cardoso, L. & Vouters, G. The Real-Time System for Distribution of Clock, Control and Monitoring Commands with Fixed Latency of the LHCb experiment at CERN. *IEEE Transactions On Nuclear Science*. (2023)

[3] Red Hat Enterprise Virtualization https://www.redhat.com/en/technologies/virtualization/enterprise-virtualization

[4] NetApp https://www.netapp.com/

[5] IBM Tivoli Storage Manager https://www.ibm.com/docs/en/tsm

[6] Foreman https://www.theforeman.org/

[7] KEA DHCP https://www.isc.org/kea/

[8] Foreman Smart-Proxy Plug-in https://github.com/theforeman/smart-proxy

[9] Puppet https://www.puppet.com

[10] Telegraf https://www.influxdata.com/time-series-platform/telegraf/

[11] Prometheus https://prometheus.io/

[12] Grafana https://grafana.com/

[13] Aparicio, R. & Coz, I. Database on Demand: insight how to build your own DBaaS. *Journal Of Physics: Conference Series*. **664**, 042021 (2015)

[14] Kickstart Installation https://access.redhat.com/documentation/it-it/red_hat_enterprise_linux/7/html/installation_guide/sect-kickstart-howto