MANAGEMENT OF SERVER AND NETWORK INFRASTRUCTURE AT SuperKEKB

H. Sugimura*, F. Ito, A. Morita, T. T. Nakamura, S. Sasaki,
High Energy Accelerator Research Organization (KEK), Tsukuba, Japan

Abstract

The SuperKEKB accelerator employs an EPICS-based control system for its operation. This paper presents the design and implementation of the core server infrastructure and network environment that support the accelerator control system. The servers are provisioned and managed using Ansible, enabling infrastructure-as-code practices that ensure idempotency, reproducibility, and maintainability across the system.

We describe the configuration of physical and virtual servers, the monitoring infrastructure based on Zabbix, iDRAC, and the Elastic Stack, as well as our operational experiences, including the smooth OS migration from CentOS 7 to AlmaLinux 9. The lessons learned and the benefits of code-based infrastructure management are also discussed.

INTRODUCTION

At the SuperKEKB accelerator facility, documentation related to computing and networking infrastructure has traditionally been maintained in the form of internal technical notes. These records, inherited from the preceding KEKB accelerator project, have supported the continuity of technical knowledge despite personnel changes over time. In particular, the dedicated accelerator control network required systematic management of configuration files for machines running core services such as DNS, DHCP, and NTP. These configuration files have been carefully preserved across operating system upgrades, allowing seamless continuity in system administration.

In addition to these core service servers, various user-facing machines have been provided to support the development of accelerator-related software tools. These systems required the installation and maintenance of application environments, which have been updated as needed in accordance with changes in the underlying operating systems.

In recent years, however, modern technologies have emerged that dramatically improve the maintainability and reproducibility of infrastructure. One such advancement is Infrastructure as Code (IaC), which enables system configurations to be described programmatically. By adopting this approach, server setup and configuration tasks can be performed consistently by any administrator, thereby reducing reliance on specific individuals. Ansible, a representative tool for IaC, has been introduced in our environment to automate and accelerate server provisioning processes [1]. Its adoption has also proven beneficial in disaster recovery scenarios, allowing rapid reinstallation and configuration of servers from scratch following hardware failures. At SuperKEKB, Ansible-based automation has been employed to deploy and manage a variety of servers essential to the control system.

In the past, server monitoring was often performed by manually saving and searching command-line output logs to determine who had executed which operations on which machines. This method has since been replaced by a more modern and robust monitoring framework using Zabbix, which stores monitoring data in a structured database and provides intuitive visualization [2]. Zabbix is used primarily for OS-level monitoring, while hardware-level monitoring is achieved using tools provided by the server vendor. In our system, all servers are standardized on Dell hardware, and monitoring is performed using iDRAC (Integrated Dell Remote Access Controller), which enables centralized management and hardware status tracking.

For network monitoring, we have adopted the Elastic Stack to collect, store, and analyze communication logs [3]. These data are visualized using Grafana and Kibana, enabling comprehensive visibility into network activity and facilitating efficient troubleshooting [4, 5].

SYSTEM OVERVIEW

The server infrastructure of the SuperKEKB accelerator can be broadly categorized into three types: servers for core infrastructure services, user-facing servers, and administrator-exclusive servers. Each of these three server types is deployed across two network layers—the accelerator control network and the higher-level institutional network. In particular, the core service servers are configured with redundancy by deploying two servers per network layer, resulting in a total of four core servers in operation. These servers host essential services directly (i.e., without containerization or virtualization layers). The operating system selected for these servers is AlmaLinux 9 [6]. Previously, CentOS 7 had been used [7]; however, with its end of life (EOL), AlmaLinux was chosen as a binary-compatible alternative to RHEL.

User-facing servers and administrator-exclusive servers are operated as virtual machines (VMs). The physical host machines also run AlmaLinux 9 and utilize KVM (Kernel-based Virtual Machine) for virtualization, hosting multiple VMs on each server. This virtualization approach was adopted in response to recent advances in server hardware capabilities, which now allow a single physical server to reliably support multiple VMs. Currently, over twenty VMs are in operation across the system. VM management was performed using both the virsh command-line interface and

* hitoshi.sugimura@kek.jp

Cockpit [8], a web-based server administration interface developed by Red Hat. The current system consists of seven host machines, with computational loads distributed among them to balance performance and resource utilization.

Regarding network configuration, all servers within the accelerator control network are interconnected via a 10 GbE network. In contrast, servers on the KEK internal network are connected via a 1 GbE network. The accelerator control system is built on EPICS, which operates reliably over the 10 GbE infrastructure [9]. Most user access, however, occurs from office spaces via the internal network, where only 1 GbE connectivity is available. Nevertheless, because user activities are primarily limited to data retrieval, this bandwidth is generally sufficient and does not pose a risk to accelerator operations.

An overview of the current system architecture, including the network layout and server classifications, is illustrated in Fig. 1.

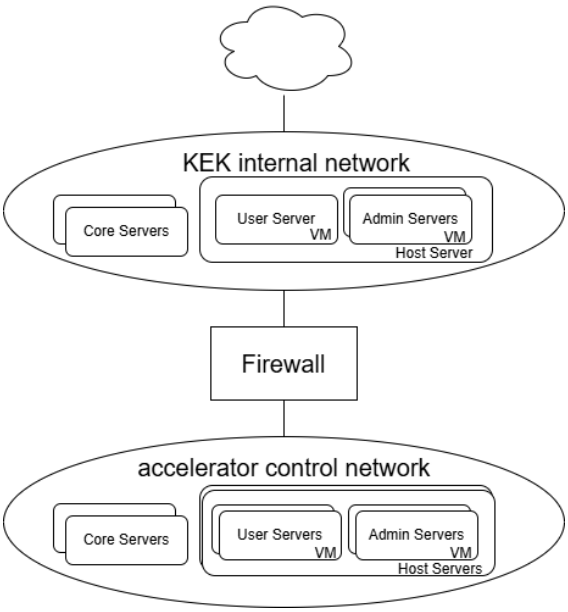

Figure 1: Architecture of server and network.

SERVER PROVISIONING WITH ANSIBLE

A dedicated control server was prepared to execute Ansible playbooks, from which the provisioning of all servers—including VMs, physical host machines, and core infrastructure servers—was carried out. In order for the control server to connect to target machines via SSH, all servers were preconfigured to allow access through the appropriate SSH ports. Additionally, since public key authentication was used, it was necessary to pre-distribute the corresponding public keys to each target server. As Ansible is implemented in Python, the necessary Python environment was also set up on the control server.

Each server type has a corresponding Ansible playbook that defines the provisioning steps. These playbooks, along with their associated inventory files, are placed directly within the top-level directories corresponding to each server group, namely vm_server_setup, core_server_setup, and user_server_setup. This structure allows administrators to execute group-specific provisioning in a modular and self-contained manner, using the appropriate combination of playbooks, inventory, and roles. Within each directory, roles are further categorized by service type (e.g., LDAP, DHCP, proxy, etc.). The directory structure used to organize the Ansible playbooks, roles, and inventory files is shown in Fig. 2. This modular structure allows administrators to selectively apply or update specific roles without reapplying the entire configuration. In addition to server-side configuration, Ansible also provisions basic network settings such as hostname, IP address assignment, and DNS resolution, contributing to the consistent integration of the servers into the accelerator control network.

The Ansible configuration codebase (including playbooks, roles, and inventory files) is maintained in a GitLab repository. Each administrator pulls the latest version from the repository into their local working directory when executing a playbook. Since administrators also serve as developers, any updates or newly created roles are committed and pushed back to the shared repository, ensuring collaborative development and version control.

```
.
├── commons
│   ├── roles
│   └── vars
├── core_server_setup
│   ├── group_vars
│   ├── host_vars
│   ├── roles
│   ├── inventory.ini
│   └── playbook.yml
├── user_server_setup
│   ├── group_vars
│   ├── host_vars
│   ├── inventory.ini
│   └── playbook.yml
└── vm_server_setup
    ├── group_vars
    ├── host_vars
    ├── roles
    ├── inventory.ini
    └── playbook.yml
```

Figure 2: Directory structure of the Ansible-based provisioning system. Common roles and shared variables are maintained under `commons/`, while server-type-specific playbooks, inventories, and roles are placed in `vm_server_setup/`, `core_server_setup/`, and `user_server_setup/`. The file `playbook.yml` represents a typical playbook, although in practice multiple playbooks are defined for individual servers within each directory.

MONITORING INFRASTRUCTURE

Multiple tools are employed to monitor the server infrastructure at different levels.

- OS-level monitoring using Zabbix
- Hardware-level monitoring via iDRAC
- Network traffic monitoring and visualization using Elastic Stack and Grafana/Kibana

Zabbix

A dedicated VM has been provisioned to host the Zabbix server, which is responsible for operating system–level monitoring. The Zabbix agent is installed on each VM and physical host to periodically collect various system metrics. When abnormal conditions are detected—such as resource exhaustion or service failure—email alerts are automatically triggered. The collected time-series data are also visualized using Grafana for easier interpretation and trend analysis.

iDRAC

iDRAC is a built-in hardware management interface provided with Dell servers. It enables remote monitoring of critical hardware components such as power supply units, memory modules, CPUs, and storage devices. In the event of a hardware fault, iDRAC sends alerts via email or SNMP traps, allowing for prompt detection and response.

Elastic Stack

For network-level monitoring, the Elastic Stack is employed in combination with Grafana and Kibana for visualization. A dedicated VM is set up to host the Elastic Stack components, which collect and index network communication logs. These logs are then visualized to assist in performance monitoring, troubleshooting, and usage auditing.

OPERATIONAL EXPERIENCES

In the summer of 2023, the operating system across our server infrastructure was upgraded from CentOS 7 to AlmaLinux 9. Although the overall system architecture remained unchanged, some packages that had previously been available in the CentOS 7 repositories were no longer included in AlmaLinux 9. As a result, additional effort was required to identify and locate alternative sources for these packages.

A representative example was OpenLDAP, which had been deprecated in RHEL 9 and its derivatives. To continue using it, we obtained the necessary packages from the EPEL (Extra Packages for Enterprise Linux) repository. While this repository is relatively well-known and easy to find, the availability of similar resources in the future remains uncertain.

Despite such issues, the upgrade process was carried out smoothly, largely due to the use of Ansible for the majority of system configuration tasks.

One of the remaining operational challenges concerns the eventual EOL of specific applications. While a smooth migration path is ideal, it may initially be necessary to perform manual installations via the command line during early evaluation and testing phases. Once a stable configuration is confirmed, the installation and configuration procedures can be incorporated into Ansible playbooks to ensure reproducibility and consistency.

CONCLUSION

By adopting Ansible for server provisioning, we have developed and maintained a shared codebase within a multi-person team. This approach has ensured idempotency, allowing any administrator to perform consistent and reproducible system setups. As a result, even in cases where provisioning fails or new servers are introduced, the environment can be rebuilt quickly and reliably.

Furthermore, the codification of system configurations facilitated a relatively smooth upgrade process during the transition from CentOS 7 to AlmaLinux 9. For infrastructure monitoring, we have deployed tools such as Zabbix and iDRAC, enabling prompt detection of system abnormalities and contributing to the overall stability and maintainability of the control environment.

REFERENCES

[1] Ansible Documentation, https://docs.ansible.com/

[2] Zabbix Documentation, https://www.zabbix.com/documentation

[3] Elastic Stack Documentation, https://www.elastic.co/guide/index.html

[4] Grafana Documentation, https://grafana.com/docs/

[5] Kibana, https://www.elastic.co/kibana

[6] AlmaLinux OS Foundation, https://almalinux.org

[7] CentOS Project, https://www.centos.org

[8] Cockpit Project, https://cockpit-project.org

[9] EPICS: Experimental Physics and Industrial Control System, https://epics-controls.org