Implementing Lenovo Confluent Management Software

Last update: **29 July 2025**

Introduces the Lenovo Confluent software

Provides an overview of the Confluent design principles

Describes the Confluent software features

Demonstrates the Confluent collective mode for the HA implementation

Anil Thapa
Aurelien Ortiz
Jarrod Johnson

Table of Contents

1 Abstract

This document provides a comprehensive overview of the Lenovo Confluent Management software. Its purpose is to introduce the key features, design principles, and capabilities of the Confluent tool. We begin by outlining the Confluent design principles and describing the major components of the system, followed by installation recommendations for a minimal setup. Next, we explore the security design and REST API capabilities with examples. We then delve into the Confluent High Availability (HA) architecture, presenting various deployment scenarios. Subsequent sections highlight core features such as device onboarding, hardware and firmware management, and operating system deployment methods – both diskless and disk-based. We also explain how Confluent integrates with Ansible to enhance cluster management capabilities. Finally, the last chapter offers an overview of the Confluent graphical user interface (GUI) and command-line interface (CLI), showcasing key tools with examples and screenshots.

2 Introduction

The Lenovo EveryScale HPC/AI Software Stack combines open-source with proprietary best-of-breed Supercomputing software to provide the most accessible and widely adopted open-source HPC/AI stack among Lenovo customers. This paper highlights one of its key components: the Lenovo Confluent tool, which is designed to streamline the provisioning and management of large-scale clusters.

Lenovo Confluent is a powerful open-source provisioning tool developed by Lenovo to deploy and manage the system configuration of computing clusters. From automated hardware detection on the network to hardware setup, bare-metal booting, and operating system installation, Confluent significantly eases the lives of system administrators.

It is designed with scalability in mind, Confluent incorporates a comprehensive set of modern features that streamline the management of scale-out infrastructures. It supports the entire Lenovo ThinkSystem server portfolio and is continuously updated to align with evolving server architecture and requirements.

Lenovo Confluent is widely adopted across Lenovo's largest HPC/AI clusters worldwide, it has been proven to be a reliable and indispensable asset. Its intuitive interface and robust capabilities make it a go-to solution for professionals managing complex, high-performance environments.

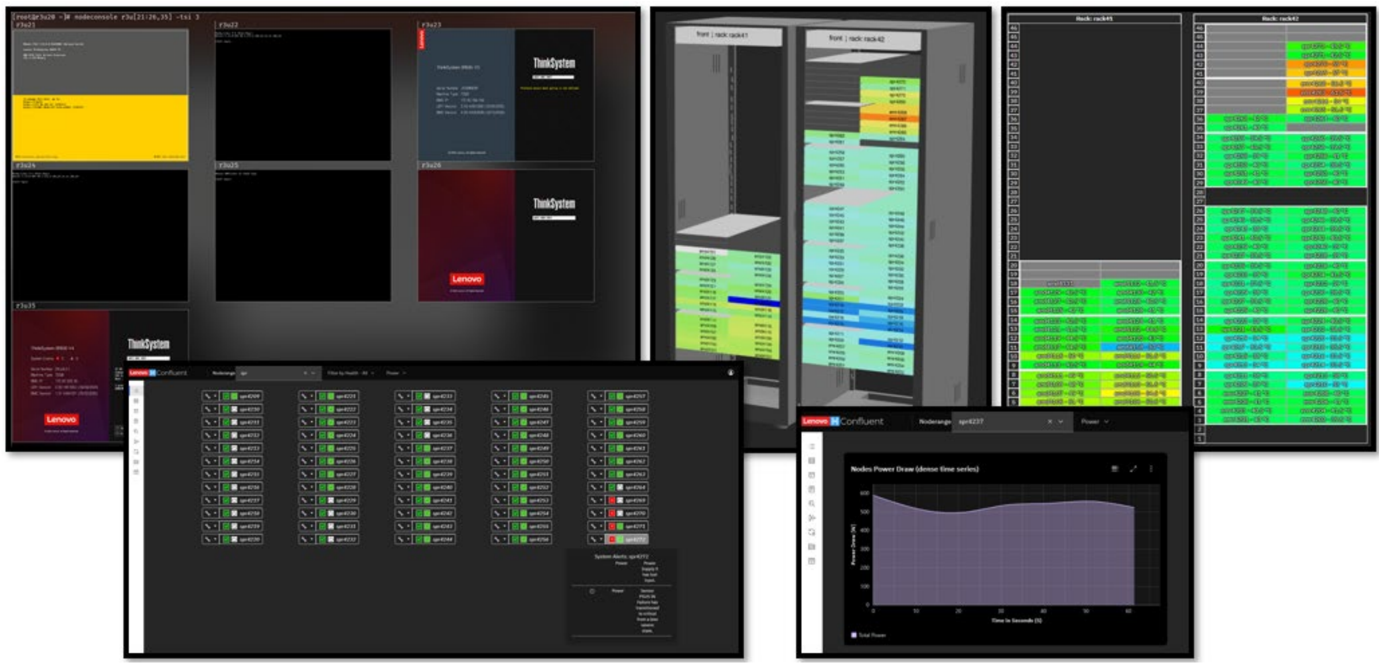


Figure 1: Lenovo Confluent GUI

3 Confluent Design principles

Confluent is built on a robust client-server architecture, where the Confluent management nodes run the confluentd service, acting as the central control point. Clients interact with the service through various interfaces, including command-line tools (CLI), a user-friendly web UI, and REST API endpoints. This flexible design enables administrators to manage clusters efficiently using their preferred interface.

The Confluent service can be deployed as a standalone server in environments where high availability is not a priority. For larger or mission-critical deployments, multiple Confluent instances can be configured to ensure both scalability and fault tolerance.

The diagram below illustrates the architectural design of Confluent.

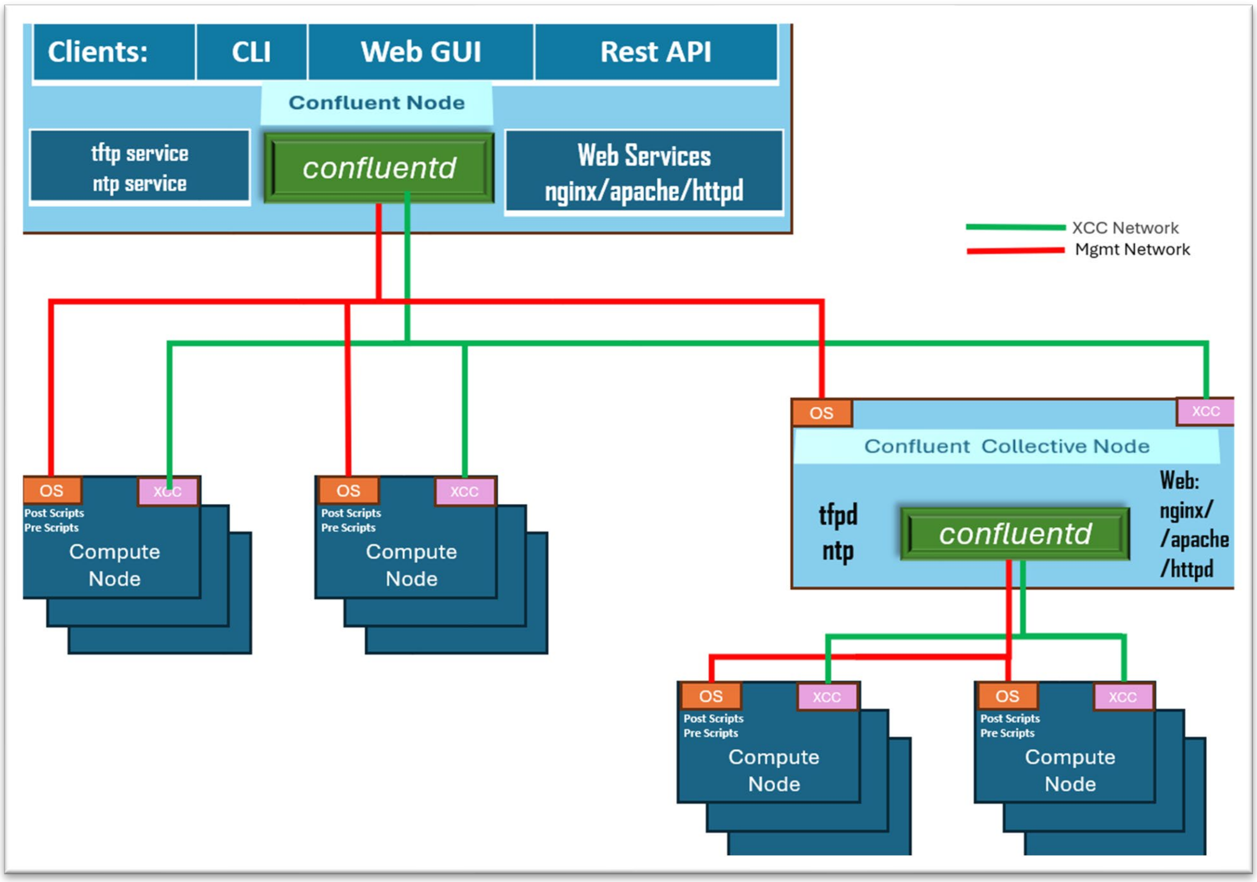


Figure 2 - Confluent Design Architecture

Confluent consists of four fundamental components:

1. Confluent Node
2. Confluent Collective Node
3. Compute Node
4. Networks

3.1 Confluent Components

3.1.1 Confluent Node

In any Confluent deployment – whether standalone or in multi-instance mode – there must be one initial server designated as the Confluent node, which is the first to run the confluentd service. This node serves as the foundation of the management infrastructure and can be either a physical or virtual server where the Confluent software is installed. Once operational, it plays a key role in orchestrating the provisioning and configuration of the entire cluster.

The Confluent node stores and maintains a comprehensive database that includes information about all managed nodes, groups, operating system definitions, and their associated attributes. To support operating system installations and other provisioning tasks, it also enables and runs essential network services such as tftpd, httpd and chronyd.

In multi-instance mode (i.e. Collective Mode), additional Confluent instances can be added to enhance scalability and availability.

3.1.2 Confluent Collective Node(s)

A multi-instance setup with Confluent is managed through collective mode, which enables multiple Confluent servers to work together as peers, providing a scalable, resilient, and highly available management infrastructure for large-scale clusters.

Traditionally, in very large clusters, there is a head node and one or more service nodes. These service nodes are subordinate to the head node and help offload tasks to reduce CPU and network load. However, a major drawback of this architecture is that if the head node fails, the service nodes typically become non-functional as well.

Confluent takes a different approach. In a Collective deployment, all nodes are treated as equals – there is no strict hierarchy between “head” node and “service” nodes. That said, administrators can still organize deployments hierarchically for practical reasons, designating one node as the primary system and others as collective members responsible for managing subsets of the infrastructure based on network topology or workload distribution.

A Confluent collective node is installed and configured similarly to a standalone Confluent node. Its primary role is to share the management workload, reducing pressure on any single node. Beyond load balancing, the collective architecture also provides high availability and failover capabilities. If one node in the collective group becomes unavailable, the remaining nodes continue to operate seamlessly, ensuring uninterrupted service and improved system resilience.

3.1.3 Compute node

In a Confluent-managed cluster, compute nodes are fully orchestrated by the Confluent provisioning system – from initial hardware discovery and configuration to bare-metal OS installation and ongoing lifecycle management.

The term “Compute Node” is used generically within Confluent to refer to any node managed by the system. This includes a wide variety of logical server roles such as CPU nodes, GPU nodes, login nodes, storage node, and more. Regardless of their specific function, all these nodes are provisioned and maintained through the same unified Confluent framework.

3.1.4 Networks

In general, there are at least two networks in a large cluster environment: the XClarity Controller (XCC) network – also known as the Baseboard Management Controller (BMC) network – and the Management Provisioning network.

3.1.4.1 xClarity Controller Network

The Lenovo XClarity Controller (XCC) is the next generation management controller that replaces the baseboard management controller (BMC) for Lenovo ThinkSystem servers. It serves as the integrated service processor, consolidating multiple functions – including system monitoring, Super I/O, video control, and remote presence – into a single chip embedded on the server system board.

The XCC network is used by Confluent to control nodes using out-of-band management via the service processor. In some configurations, the XCC is set to “shared mode”, allowing it to use the same physical interface as the management and provisioning network. In this setup, both out-of-band management traffic (via XCC) and in-band system management traffic share a single physical network connection. This method simplifies cabling and reduces additional hardware requirements.

3.1.4.2 Management and Provisioning Network

The management and provisioning network is used by Confluent to install operating systems and manage compute nodes. This includes tasks such as modifying OS settings, copying files, installing additional packages, and applying security configurations. Both the Confluent node and the in-band network interfaces of the compute nodes are connected to this network. In large cluster environments, Confluent collective nodes can be configured to distribute the workload, reducing the load on a single Confluent node and enhancing scalability and reliability.

3.2 Installation Recommendations

Confluent is highly flexible and can be installed on either a physical or virtual server. It operates efficiently without requiring high-end hardware specifications. A single Confluent server – typically equipped with a 960 GB local drive, 10G network interfaces, and an 8- to 12-core CPU – can manage a mid-sized cluster of approximately 1,500 nodes without the need for additional collective nodes.

From a networking standpoint, 10G or even 25G Ethernet has become the standard in most server configurations. The Confluent node can be equipped with two such interfaces configured in a bonded setup, providing both increased bandwidth and network redundancy.

For larger environments – or even smaller clusters where redundancy and fault tolerance are critical – a High Availability (HA) configuration is recommended. This ensures continuous operation and service availability in the event of a node failure.

3.3 Security Design

Confluent is designed with a strong focus on security, incorporating secure default behaviours with the option to reduce security if needed. It employs fully validated TLS to protect collective, deployment and hardware management processes. The use of TPM2 enhances security by protecting boot volumes for supported profiles and persisting node trust across reboots in stateless environments. Confluent offers flexible node authentication options, balancing convenience with the need to protect sensitive data, such as encrypted root passwords. Its SSH PKI strategy ensures secure and convenient SSH access without requiring users to self-create SSH keys or update known hosts. Additionally, Confluent supports Secure Boot for both media and HTTPS boot methods.

Extending these capabilities, when deploying OS, Confluent also ensures that its SSH management approach maintains strict security standards throughout the deployment workflow. No ssh user or host private keys are ever transmitted anywhere. Instead, when the SSH management feature is enabled, hosts generate their host private key, and automation makes a call over HTTPS to submit the public key for signing and if correctly authenticated, receives a signed certificate for the host. It further configures nodes for host-based authentication using SSH certificate authorities, enabling node to node ssh by all users without user private key management required based on this SSH PKI.

Below is brief overview of how Confluent ensures security during deployment:

- Nodes have no credentials by default; granting one is optional and restricted.
- The nodedeploy command arms a one-time, weakly authenticated api token grant (higher security with lower compatibility is available through XCC grant rather than network grant).
- For a token grant to succeed:
  - Must be armed by an administrator (i.e. nodedeploy or API).
  - Request must originate from a local, non-routed address and privileged port.
  - Token is generated and hashed securely; only the hash is sent.
  - Granting a token disables future grants.

A valid token allows access to sensitive deployment data (e.g. root password hash, SSH key signing).

3.4 REST API

Confluent provides a REST API that enables users and developers to interact with the system programmatically. This means that, in addition to using the command-line or web interface, you can also send structured requests to Confluent over the network to perform tasks like monitoring, configuring, or controlling nodes.

The REST API is designed to mirror a hierarchical, filesystem-like structure, making it intuitive to navigate and use. This approach simplifies integration with other tools and is especially valuable for automation. Whether you're managing a handful of servers or thousands, the REST API offers a flexible and scalable way to interact with Confluent. An example of turning ON a single node “pmx3” using REST API shown below.

```
[root@r3u20 ~]# nodepower pmx3
pmx3: off
```

```
[root@r3u20 ~]# curl -H 'Accept: application/json' -k -H 'Content-Type: application/json' -u
"jjohnson2:$APIPASS" https://localhost/confluent-api/nodes/pmx3
/power/state -X PUT -d '{"state": "on"}'
{
    "_links": {
        "collection": {
            "href": "./"
        },
        "self": {
            "href": "./state"
        }
    },
    "state": {
        "value": "on"
    }
}
```

```
[root@r3u20 ~]# nodepower pmx3
pmx3: on
```

A full description of the API is available here:
https://hpc.lenovo.com/users/documentation/developer/api.html

3.5 Confluent Collective Mode

3.5.1 High Availability Architecture

Confluent's scalability and high availability (HA) are significantly enhanced through its collective mode. This mode allows a single Confluent interface to be scaled across multiple servers or virtual machines, ensuring HA and the capability to manage thousands of systems efficiently. The collective mode expects a minimum of 3 servers for redundancy.

Confluent collective relies on a straightforward majority-based quorum mechanism, defined as:

$$Quorum = \frac{N}{2} + 1$$

N is the total number of collective members. This means that if half or more of the quorum nodes become unreachable, Confluent locks down functionality to maintain consistency and prevent split-brain scenarios. It is recommended to use an odd number of nodes in the collective. As an example, a 4-nodes collective requires 3 healthy nodes to reach quorum, offering fault tolerance for only 1 node. In contrast, a 3-nodes collective also tolerates the failure of 1 node, but with fewer resources and simpler coordination. Therefore, a 4-node setup does not provide any additional redundancy over a 3-node setup.

The node attribute database is natively and automatically replicated across all members of the Confluent collective group. There is no need for any external database – each active node maintains a complete local copy of the database, stored at /etc/confluent/cfg. This internal replication ensures high availability and consistency. When a collective member goes offline and later rejoins the cluster, the entire node attribute database is automatically synchronized from the active members to the reconnecting member, ensuring it is fully up to date without manual intervention.

While the node attribute database is automatically replicated among Confluent collective members, the storage located at /var/lib/confluent – used by Confluent for OS deployment – must be synchronized or shared at the user's discretion. There is no strict requirement on the synchronization mechanism, as long as it ensures consistency across all collective members. This means that several storage synchronization or remote/clustered filesystem approaches are acceptable.

3.5.2 Collective Example: 2 Active node / 1 tiebreaker

One of the minimal viable configurations for operating in collective mode is illustrated in the following diagram. This architecture tolerates the failure of one server within the quorum.

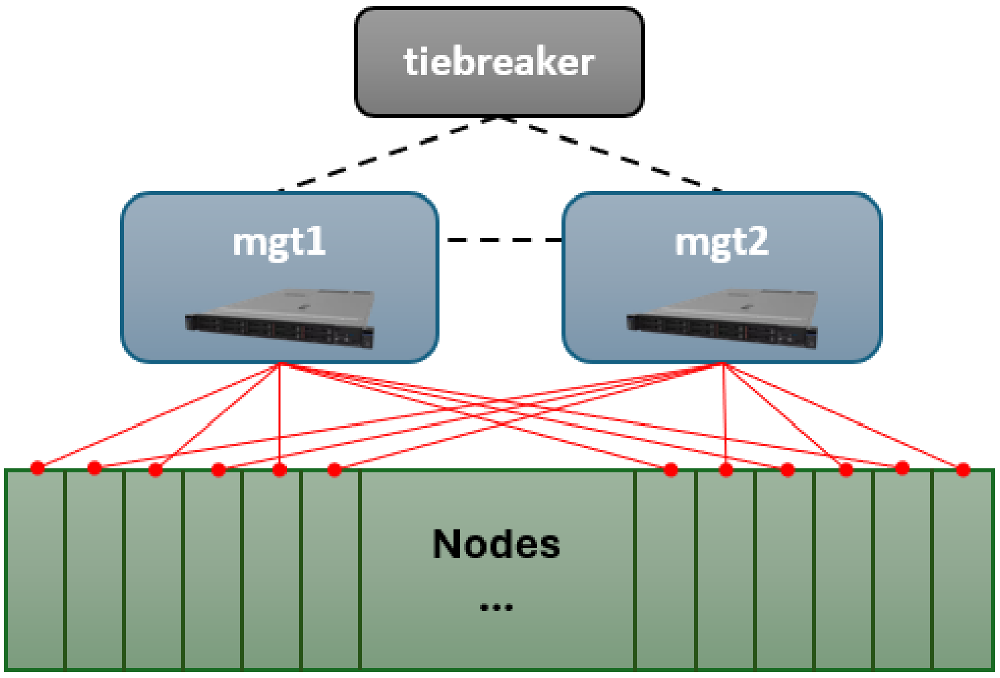


Figure 3 – Example of three collective members with two actives and one tiebreaker

The configuration provides:

- **2 actives collective members** – These nodes share the responsibility for managing compute nodes within the cluster.
- **1 tiebreaker node** – This node only acts as a collective member to participate in quorum-related decisions. It can be hosted on a login node or a lightweight virtual machine, provided it maintains reliable network communication with other active collective members.

3.5.3 Collective Example: 3 Active Nodes

In the diagram below, the cluster is configured with three Confluent servers that share the responsibility of deploying all compute nodes. This architecture is designed to tolerate the failure of one server within the quorum:

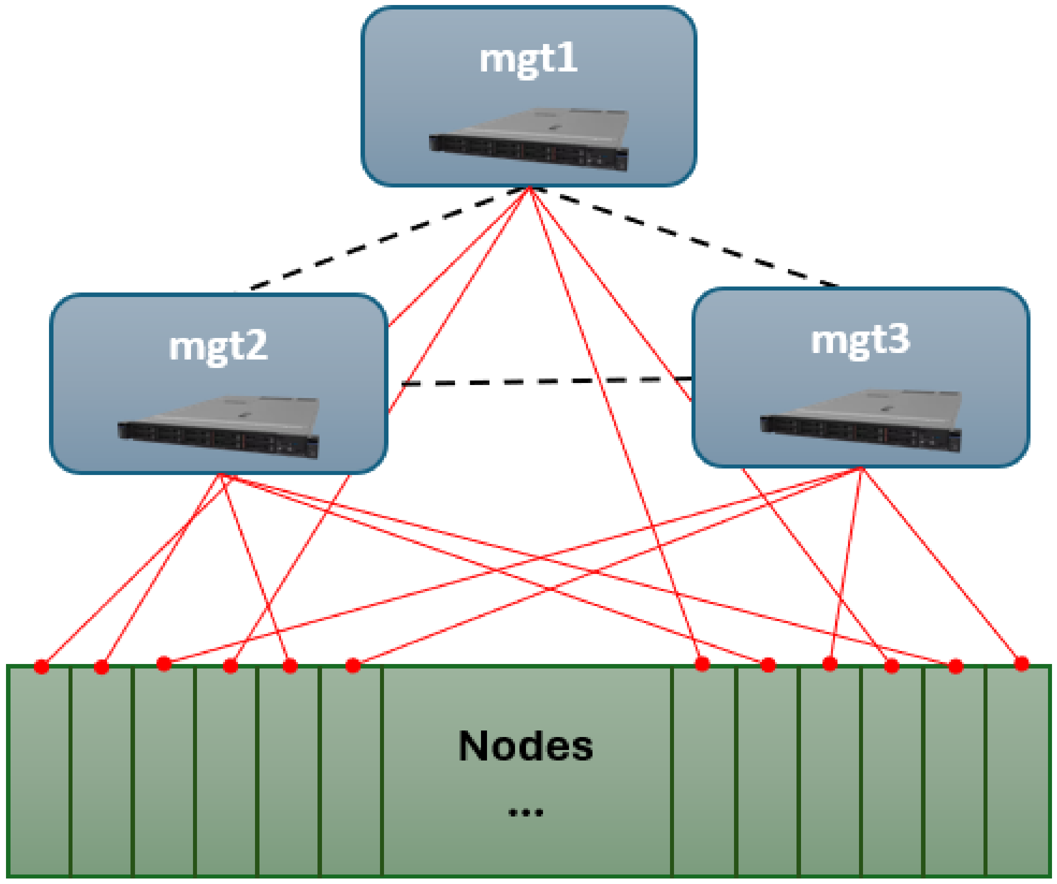


Figure 4 – Example of three collective members

3.5.4 Collective Example: 5 Active nodes

In the example below, the cluster is configured with five Confluent servers, enabling it to efficiently manage large-scale environments consisting of thousands of compute nodes. This architecture is designed to tolerate the failure of two servers within the quorum:

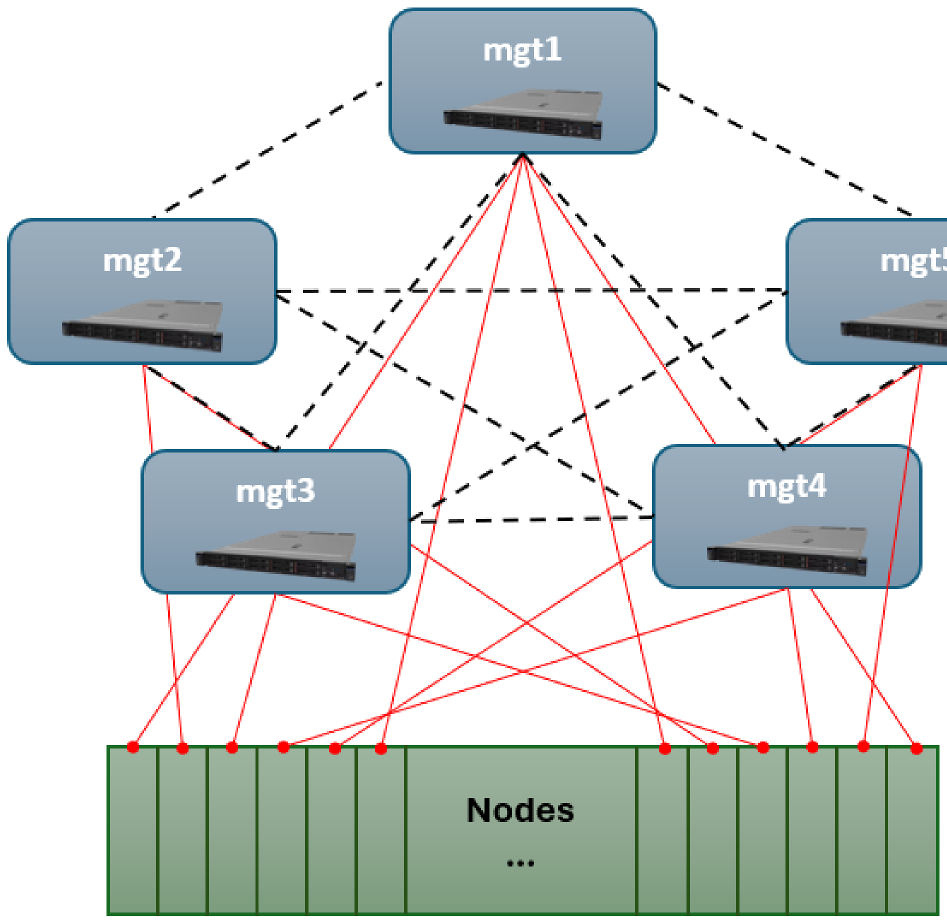


Figure 5 – Example of five collective members

4 Confluent Feature Highlights

4.1 Device Onboarding

From factory default to fully configured, Confluent excels at swiftly binding devices to logically well-defined nodes. This process can utilize various pieces of information, such as the serial number or MAC address of the server, as well as the physical location of the node within the chassis or the network switches it is connected, enabling a more automated device onboarding workflow.

A key component of this automation is the nodediscover command, which initiates a discovery process that scans the environment for new or unconfigured hardware. Confluent can automatically detect Lenovo devices on the network and apply the appropriate configurations with minimal manual input, streamlining the onboarding process and accelerating deployment.

Confluent also supports the detection and management of:

- Network switches – including platforms such as NVIDIA Cumulus, CISCO and Nokia.
- Power Distribution Units (PDUs) – including outlet status monitoring and control for brands like Delta, Eaton, and Vertiv.
- Cooling Distribution Units (CDUs) – specifically from the Cooltera brand (i.e. Vertiv), which is often integrated into Lenovo's Neptune Direct Water-Cooling solutions.

Whether deploying a single server or orchestrating a full rack-scale rollout, Confluent provides the intelligence and automation needed to simplify infrastructure management and reduce operational overhead.

4.2 Hardware Management

Hardware management is essential for administrating HPC/AI infrastructures. Confluent handles essential operations using IPMI or Redfish protocols, allowing it to remotely control any node in the cluster. This level of control and configuration includes:

- Powering on/off the nodes
- Setting the next boot device (e.g. force network boot)
- Configuring BIOS/UEFI/XCC settings
- Configuring hardware storage controllers (e.g. creating/deleting raid arrays / setting drive usage)
- Checking the health status of servers, switches, pdu and cdu
- Gathering telemetry information (e.g. temperature, voltages, power, energy, etc.)
- Managing virtual USB device mount
- Gathering "service data" for use by Lenovo's support center

One particularly useful and convenient feature of Confluent is its ability to monitor switches, PDUs, and CDUs. Once defined, administrators can easily monitor the health status of these components using the nodesensors command, providing quick and centralized visibility into system conditions. The figure below illustrates an example of a Cooltera CDU:

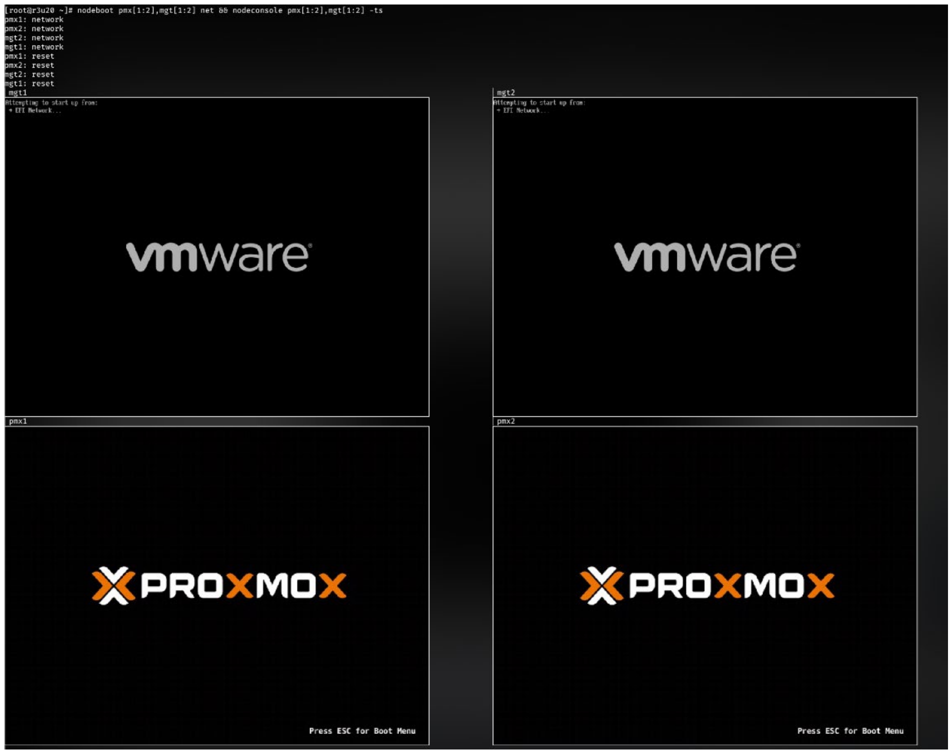
```
# nodesensors cdu1
cdu1: Primary loop supply temperature: 8.5 °C
cdu1: Secondary loop supply temperature: 19.7 °C
cdu1: Secondary loop return temperature: 20.7 °C
cdu1: Ambient air temperature: 20.2 °C
cdu1: Primary loop return temperature: 21.8 °C
cdu1: Relative Humidity: 52.7 %
cdu1: Dewpoint: 10.3 °C
cdu1: Pump 1 Speed: 73 %
cdu1: Pump 2 Speed: 0 %
cdu1: Number of active alarms: 1
cdu1: Input flow rate: 10 l/m
cdu1: Output flow rate: 22 l/m
cdu1: Secondary loop return pressure: 1.03 bar
```

Similarly, Confluent can provide a detailed overview of the PDU usage:

```
# nodesensors vtv1
vtv1: Circuit 1 Current: 0.27 A
vtv1: Circuit 2 Current: 0.3 A
vtv1: Circuit 3 Current: 0.39 A
vtv1: Circuit 4 Current: 0.72 A
vtv1: Circuit 5 Current: 1.6 A
vtv1: Circuit 6 Current: 1.69 A
vtv1: Line A Current: 2.59 A
vtv1: Line B Current: 2.51 A
vtv1: Line C Current: 3.39 A
vtv1: Phase AB Energy: 3978.65 kWh
vtv1: Phase AB Current: 0.99 A
vtv1: Phase AB Real Power: 199.0 W
vtv1: Phase AB Voltage: 214.7 Vrms
vtv1: Phase AB Apparent Power: 212.0 VA
vtv1: Phase BC Energy: 5737.511 kWh
vtv1: Phase BC Current: 1.86 A
vtv1: Phase BC Real Power: 376.0 W
vtv1: Phase BC Voltage: 215.0 Vrms
vtv1: Phase BC Apparent Power: 400.0 VA
vtv1: Phase CA Energy: 7990.681 kWh
vtv1: Phase CA Current: 2.01 A
```

4.3 Virtual Machine Management

Confluent supports the management of virtualized environments, including integration with platforms such as VMware vCenter and Proxmox. These additional plugins provide a text and graphics console, power control, inventory visibility, and set boot device support for virtual machines. This integration allows administrators to manage virtual environments alongside physical infrastructure. These virtual environments are commonly used for lab setups, testing, and development purposes. The figure below is a screenshot showing Confluent support for VMware and Proxmox.



```
[root@r3u20 ~]# nodeboot pmx[1:2],mgt[1:2] net && nodeconsole pmx[1:2],mgt[1:2] -ts
pmx1: network
pmx2: network
mgt2: network
mgt1: network
pmx1: reset
pmx2: reset
mgt2: reset
mgt1: reset
```

Figure 6: Confluent VMware and Proxmox support Screenshot

4.4 Firmware Management

Confluent can also be used to report firmware information on nodes and perform firmware updates on various components, including the XCC (BMC), BIOS/UEFI, network adapters, RAID controllers, and more. This feature, combined with Lenovo's EveryScale best recipes, ensures that all hardware components are up-to-date and functioning optimally, contributing to the overall stability and performance of the system, and maintaining a homogeneous cluster.

> Lenovo EveryScale provides a comprehensive matrix of validated firmware, operating systems, drivers, and management tools that are tested together to ensure compatibility, stability, and optimal performance across Lenovo server platforms. These curated combinations – referred to as Best Recipes – help cluster administrators maintain consistent and reliable infrastructure.
>
> https://support.lenovo.com/us/en/solutions/ht510136-lenovo-scalable-infrastructure-best-recipes

4.5 Operating Systems and CPU Architecture

Confluent can deploy servers over PXE, HTTP(s) boot or removable media (both real and virtual). For instance, when a PXE request from a booting node is detected, Confluent assigns the node its IP address and initiates the installation workflow, which can be either diskful or diskless.

Unlike other solutions like xCAT2, Confluent does not rely on a DHCP server. This eliminates any conflicts with pre-existing DHCP servers and remove the need for a dedicated network for HPC/AI cluster deployment.

Additionally, Confluent allows for extensive customization during various phases of deployment (e.g. post, firstboot, onboot) through the use of local commands or automatically triggered remote Ansible playbooks.

The following figure illustrates the currently supported operating systems in Confluent, highlighting that Red Hat Enterprise Linux, Rocky Linux, and Ubuntu are the three primary distributions that benefit from comprehensive support through both Lenovo Confluent and EveryScale.

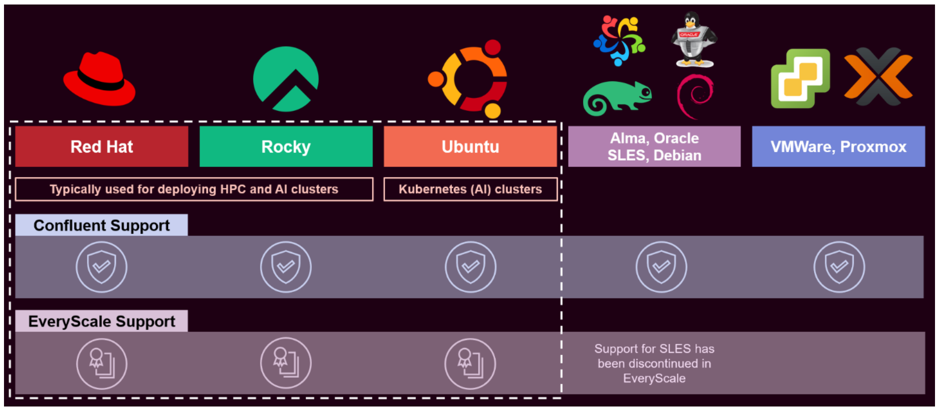

Red Hat
Rocky
Ubuntu
Alma, Oracle SLES, Debian
VMWare, Proxmox
Typically used for deploying HPC and AI clusters
Kubernetes (AI) clusters
Confluent Support
EveryScale Support
Support for SLES has been discontinued in EveryScale

Figure 7 - OS support in Confluent and EveryScale

The table below highlights the CPU architectures currently supported by Confluent across all supported operating systems:

	x86	ARM
Red Hat	✓	✓
Rocky Linux	✓	✓
Ubuntu	✓	✓
Alma Linux	✓	✓
Oracle Linux	✓	✓
SLES	✓	✗

Debian	✓	✗
VMware ESXi	✓	✗
Proxmox	✓	✗

4.6 OS Deployment Methods

Confluent’s diskless feature streamlines the deployment and management of operating systems with a straightforward scripted build process. The portable images can be easily modified, ensuring seamless updates. Users can create images from scratch or capture an installed system image, offering flexibility.

Deployment options for diskless installations include:

- **Tethered diskless** – In this mode, OS is retrieved block by block using a highly available multipath HTTPS access filesystem on-demand. It provides lower memory consumption on the compute nodes and faster boot times.
- **Untethered diskless** – In this mode, the root filesystem is downloaded to the compute node’s memory, reducing sensitivity to network conditions but increasing both memory consumption and boot time.

Confluent also supports **diskful installations**, automating the workflow through a configuration file, such as a kickstart file for RHEL-like operating systems. The kickstart file includes predefined configuration settings, such as partitioning schemes and package selections, which streamline the installation process. This method is particularly advantageous for nodes with dedicated OS drives, as it significantly reduces boot time and eliminates the need for in-memory OS or remote filesystem access.

Additionally, Confluent provides advanced configuration capabilities such as: comprehensive network setup for both Ethernet and Infiniband (including interface bonding, DOCA driver support), configuration file synchronization via the syncfiles tool, software RAID and Intel VROC support, and more.

4.7 Ansible

Confluent offers full compatibility with Ansible, enabling seamless automation both during the initial installation process and throughout the ongoing management of infrastructure. This integration empowers administrators to leverage Ansible playbooks to automate the provisioning, configuration, and maintenance of nodes, ensuring a consistent, repeatable, and efficient setup across the entire environment.

Administrators can maintain a separate Ansible configuration, independent of Confluent itself. They can build and manage their own Ansible directory structure and playbooks, tailored to their specific operational needs. During the Confluent deployment process, specific playbooks from the Ansible tree can be triggered at defined stages, allowing custom configurations to be applied automatically.

This approach is especially valuable in diskful installations, where nodes are not frequently reinstalled. Instead of relying on full redeployments to apply changes, administrators can use Ansible to continuously manage and update node configurations over time. However, this capability is also applicable to diskless environments, offering the same flexibility and control for building node images.

Additionally, Confluent includes the confluent2ansible command, which automatically generates an Ansible Inventory based on the data stored in Confluent’s internal database

5 GUI and CLI: Capabilities Overview

5.1 Real time visualization of telemetry

Through its intuitive WebUI, Confluent provides a comprehensive visualization of the physical infrastructure, enabling administrators to monitor and manage their systems with precision and ease. One of the standout features is the RackView with sensor overlay, which provides a real-time graphical representation of the data center layout, enriched with telemetry data displayed in distinct colors for quick interpretation.

This visualization includes critical metrics such as:

- Inlet Temperature
- Processor Temperature
- Power Consumption

By overlaying this telemetry data directly onto the physical rack layout, administrators can instantly identify hot spots, power anomalies, or underperforming nodes – enabling faster diagnostics and more informed decision-making.

In addition, Confluent includes built-in tools for consistency checking and statistical analysis. These tools help identify inconsistent nodes (e.g. mismatched firmware versions or configuration drift), or outlier nodes based on performance, temperature, power usage, or any other numerical metric.

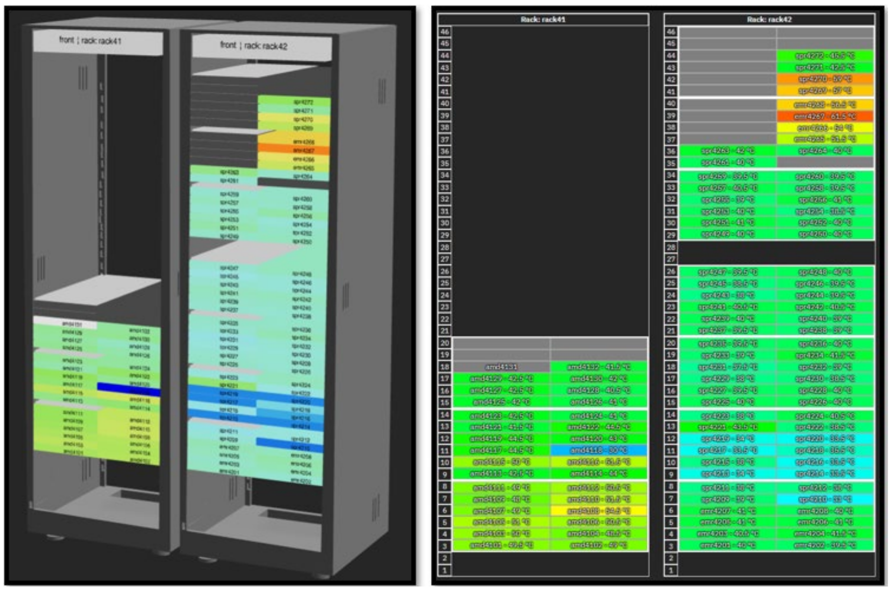


Figure 8: Confluent 3D Rack View (Left) & Flat Rack View (Right)

5.2 Console Access Management and Logging

Confluent provides robust remote console management and logging capabilities with configurable retention policies. The system supports full logging with fine-grained timestamps, ensuring precise tracking of events. This allows administrators to maintain detailed records of console activities for auditing and troubleshooting purposes. Additionally, administrators can interact with multiple consoles simultaneously within a single window.

Confluent consoles are available through both CLI and WebUI, providing flexibility and convenience for administrators to manage and monitor systems using their preferred interface. The WebUI offers a powerful multi-console view, enabling administrators to monitor and interact with multiple nodes simultaneously within a single browser window. One of its unique features is the “type in all terminals” functionality, which allows keystrokes - including arrow keys and escape keys - to be broadcast to all selected nodes at once.

Support for select graphical consoles being managed is also available.

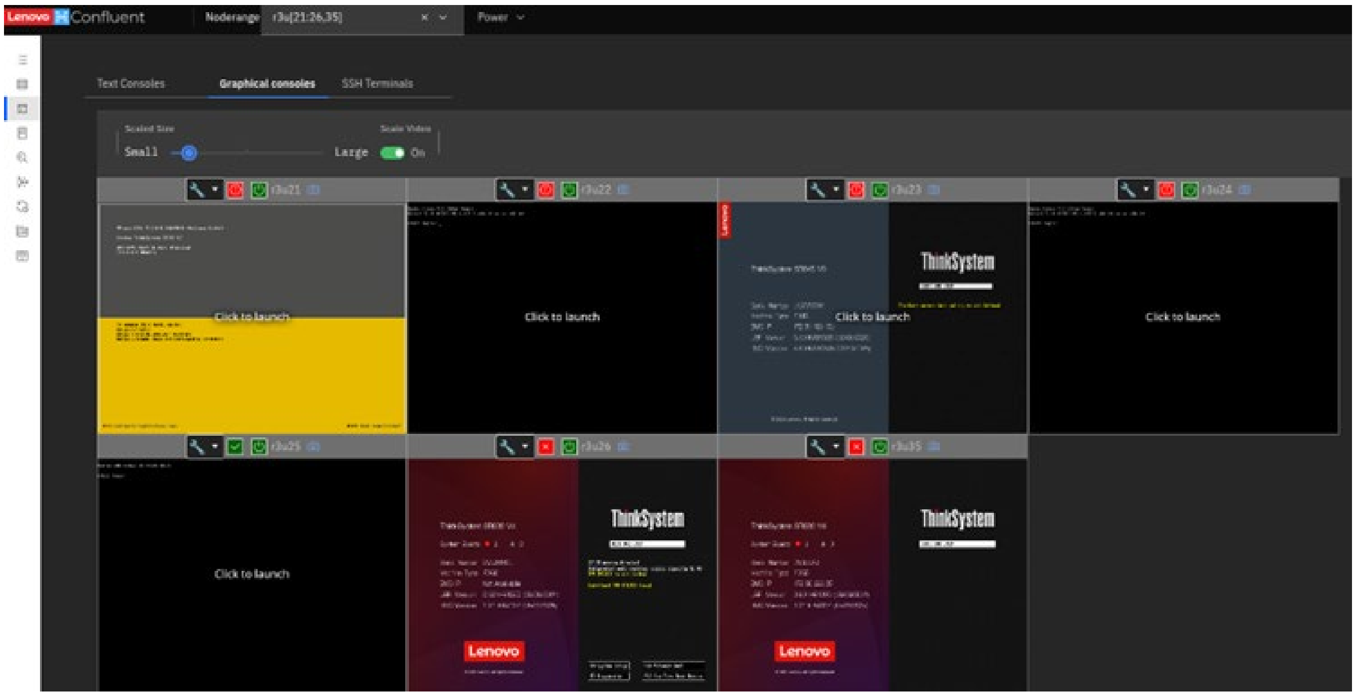


Figure 9 - Confluent Multi-Consoles

5.3 Admin Tools

Confluent provides a comprehensive set of command-line tools designed to simplify the daily administration of large-scale clusters. Below are a few example commands (note: this is not an exhaustive list). Full list of confluent command can be found in this link https://hpc.lenovo.com/users/documentation/man/

- nodepower – Check or change power state of confluent nodes (i.e. on, off, boot, shutdown, reset, status, pdu_status, pdu_on, pdu_off).

- nodeshell – Enables administrators to execute commands across multiple nodes within a specified node range via SSH. It allows execution of a command on many nodes in parallel and provide a powerful attribute expressions syntax.
- nodeconsole – Opens an interactive console session to a given node (i.e. text or serial console of the system). The nodeconsole command recently introduced a new capability for visualizing the state of multiple nodes at the same time, by capturing and displaying screenshots of their remote consoles in the terminal.
- nodereseat – Request a reseat of a node (note: applicable only to chassis designs where compute nodes are housed in trays). When executed, it instructs the enclosure manager to reseat the node in its current slot – functionally equivalent to physically removing the node from the chassis and reinserting it.
- nodeinventory – Pulls information about hardware of a node, including adapters, serial numbers, processors, and memory modules.
- nodefirmware – Reports and updates various firmware on nodes.

5.4 Diagnostic Tools

Troubleshooting hardware errors/anomalies can be difficult, time consuming and sometime challenging. Confluent provides number of diagnostic tools that enables administrators identify the root cause and resolve the issue, saving significant time during the cluster operations. Few key confluent diagnostic tools, nodehealth, nodeeventlog, and confluent_selfcheck are described below.

Confluent provides the capability to monitor the health parameters of each component via at least the IPMI 2.0 standard and is integrated in Confluent. Confluent command or API interface is available to get system health information including disks, DIMMs, motherboards, network adapters and GPU issues. In addition, a wide range of temperature related issues such as voltages issues, ecc errors, various failed disks and so forth are available for alert. A “nodehealth” command is available in the confluent which can be plugged directly into Nagios or Icinga to get an alert. Nodehealth command is very handy when troubleshooting. It provides the current health assessment on a confluent node. Example below showing nodehealth output for a given compute group.

```
[root@confluent~]# nodehealth compute
node001: ok
node002: ok
node003: ok
node004: ok
```

Confluent provides nodeeventlog command to catch a wide variety of hardware faults and pull detail eventlog from confluent node that are well characterized, consistency check for arbitrary discrete data (e.g. firmware progress, OS boot status and so on). These eventlogs can be configured to notify administrators through via email. Example below showing nodeeventlogs output showing status and the PSU failure in node001.

```
[root@confluent~]# nodeeventlog node001
node001: 06/24/2025 19:13:16 System Firmware - Progress - Unspecified
node001: 06/24/2025 19:15:19 System Firmware - Progress - Starting OS boot
node001: 06/25/2025 10:47:43 Power Supply - PSU 2 - Power input lost
node001: 06/25/2025 10:47:45 Power Unit - Power Resource - Not redundant
node001: 06/25/2025 10:47:47 Power Supply - PSU 2 - Power input out of range
node001: 06/25/2025 10:47:55 Power Supply - PSU 2 IN Failure - Critical
node001: 06/25/2025 10:59:49 Power Supply - PSU 2 - Power input restored
node001: 06/25/2025 10:59:50 Power Unit - Power Resource - Redundancy restored
```

Confluent provides the confluent_selfcheck command, which performs configuration checks on a specific node. This tool is particularly useful for diagnosing issues such as nodes failing to deploy correctly:

```
# confluent_selfcheck -an r3u25
OS Deployment: Initialized
Confluent UUID: Consistent
Web Server: Running
Web Certificate: OK
Checking web download: OK
Checking web API access: OK
IP neighbor table issue check:OK
TFTP Status: OK
SSH root user public key: OK
Checking SSH Certificate authority: OK
Checking confluent SSH automation key: OK
Checking for blocked insecure boot: OK
Checking IPv6 enablement: OK
Performing node checks for 'r3u25'
Checking node attributes in confluent...
r3u25 is not currently set to deploy any profile, network boot attempts will be ignored
Checking network configuration for r3u25
r3u25 appears to have network configuration suitable for IPv4 deployment via: bridge0
No issues detected with attributes of r3u25
Checking name resolution: OK
Checking confluent automation access to r3u25...
Confluent automation access to 172.30.193.25 seems OK
```

6 Conclusion

In this document, we introduced Lenovo Confluent -a powerful solution designed to simplify the deployment and management of large-scale HPC and AI clusters. We explored its architecture and design principles, with a particular focus on the Collective Mode, which enhances high availability and load balancing across the cluster.

We then examined the core features of Confluent, highlighting the different deployment methods as well as its intuitive GUI and CLI interfaces that streamline cluster operations. Additionally, we demonstrated how Confluent supports day-to-day administration through features such as Ansible integration, diagnostic tools and few commands, making it a valuable tool for system administrators managing complex environments.

This document establishes Confluent as a robust and efficient platform for cluster management. In future work, we will delve deeper into the technical aspects of installing, deploying, and operating the Confluent software, providing hands-on guidance for real-world implementation.

7 References

https://lenovopress.lenovo.com/lp0900-lenovo-everyscale-lesi

https://hpc.lenovo.com/users/documentation/whatisconfluent.html

https://hpc.lenovo.com/users/documentation/developer/api.html

https://hpc.lenovo.com/users/documentation/confluentquickstart_el8.html

https://lenovopress.lenovo.com/lp0880-xcc-support-on-thinksystem-servers

https://xcat.org/index.html

https://lenovopress.lenovo.com/servers/dense#sort=relevance

https://support.lenovo.com/us/en/solutions/ht510136-lenovo-scalable-infrastructure-best-recipes

8 Authors

Anil Thapa is the Senior High-Performance Computing (HPC) Architect on the EMEA team at Lenovo within the Infrastructure Solution group. He has over 20+ years experiences in high performance computing in HPC system design, implementation and deployment of large and complex environments. Anil comes with deep technical HPC skills, prior to joining Lenovo in 2016, he led Nordic Supercomputing (a joint consortium from Nordic countries), project where he was the key HPC technical leader for Nordics technical experts. Anil Thapa holds, master's degree in computing system and network engineering and master's in research (MRes) in System engineering from London South Bank University.

Aurelien Ortiz is the HPC and AI Software Architect on the Worldwide Team at Lenovo, within the Infrastructure Solutions Group. In this role, he leads the design and support of advanced software architectures for high performance computing and artificial intelligence solutions. Aurelien brings over 15 years of experience. Throughout his career, he has contributed to a wide range of projects involving HPC clusters, AI infrastructure, and large-scale storage systems. Prior to joining Lenovo in 2025, Aurelien held various engineering roles where he applied his deep technical expertise to advance performance and scalability in complex computing environments. Aurelien holds a PhD in Distributed Systems from the University of Paul Sabatier in Toulouse, France, awarded in 2009.

Jarrod Johnson is a Senior Software Engineer at Lenovo with ISG SW & Solutions development. He is the creator of Lenovo Confluent cluster management software. He has been working with scale out management software since 2003 while working with IBM where he was a lead architect for the xCAT 2 project. He came to Lenovo in 2014 as he began the work to incorporate lessons learned in developing xCAT to create confluent as a successor project. Jarrod holds BS in Computer Science from North Carolina State University

Trademarks and special notices