ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-THPDP047

ELK STACK DEPLOYMENT WITH ANSIBLE

T. Gatsi, X. P. Baloyi, J. L. Lekganyane, R. L. Schwartz
South African Radio Astronomy Observatory, Cape Town, South Africa

Abstract

The 64-dish MeerKAT radio telescope, constructed in South Africa, became the largest and most sensitive radio telescope in the Southern Hemisphere and will eventually be integrated with the Square Kilometre Array (SKA).

A Control and Monitoring (CAM) system for a Radio Astronomy Project, such as MeerKAT, produces vast amounts of sensor data and event logs. Viewing and analysing this data to trace system issues require engineers and maintainers to spend significant time searching for the related events.

The ELK (Elasticsearch, Logstash, Kibana) software stack, deployed using Ansible, was implemented for the MeerKAT radio telescope in order to have the capability to aggregate system process logs and save search time.

A cluster deployment was used to ensure load balancing, high availability and fault tolerance within the MeetKAT CAM environment. This was deployed using Linux Containers (LXC) running inside a Proxmox virtual environment, with Ansible as the software deployment tool. Each container in the cluster performs cluster duties, such as deciding where to place index shards, and when to move them. Each container is also configured to be a data node, which makes up the heart of the cluster.

Logstash is used to ingest, transform and send data to Elasticsearch for indexing. The data is then made available for visualisation to the MeerKAT Operations Team and other interested users via the Kibana Graphical User Interface (GUI).

INTRODUCTION

The MeerKAT radio telescope generates large amounts of logs daily. The ability to effectively manage, analyse, and extract insights from this data is paramount. One key solution that has emerged to address this need is the ELK cluster, comprising three core components – Elasticsearch [1], Logstash [2], and Kibana [3]. The ELK stack [4] provides a robust framework for ingesting, processing, storing, and visualising data.

In the context of the MeerKAT CAM system environment, an ELK cluster is deployed on three LXC nodes within a Proxmox virtual environment. This enables efficient display and analysis of control and monitoring system logs through Kibana, as shown in Fig. 1.

The process of deployment of this ELK cluster can be complex, requiring numerous resources and time. To make the deployment easier and faster, the automation platform from Red Hat, Ansible [5], was used within the MeerKAT CAM environment. This paper outlines the process of how this was accomplished.

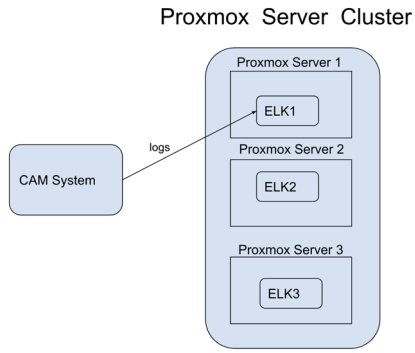

Figure 1: ELK stack deployment in CAM.

ELK SOFTWARE STACK

Elasticsearch

At the heart of the ELK cluster lies Elasticsearch. This powerful and scalable search and analytics engine serves as the backbone of the entire setup [6]. Its primary role is to efficiently index and store data, making it searchable and retrievable in real-time.

Complementing Elasticsearch is Logstash, a versatile data processing pipeline. Logstash serves as the bridge between data sources and Elasticsearch, facilitating the ingestion of data, its transformation, and potential enrichment.

In the MeerKAT CAM environment, Elasticsearch's distributed nature allows it to be seamlessly spread across our three Ubuntu LXC container nodes. As CAM system logs are generated, they are fed into Elasticsearch via the configured Logstash server, where they are indexed and organised, forming the foundation for comprehensive log analysis.

Indexing Indexing entails importing data from external sources into the elasticsearch cluster. In the CAM software application, data is ingested as logs from the production system. ElasticSearch functions as a textual indexer, exclusively analysing plain text data. However, a plugin allows data storage in base64 format.

During data indexing, fields are defined, where either a built-in analyser is used, or a custom analyser is created. Specific fields are also selected to be made available in search results. Notably, indexing operations differs from typical CRUD (Create, Read, Update and Delete) operations – instead of updating or deleting data directly, Elasticsearch generates a new version of the index while marking the old version as deleted.

This point is crucial, since improper configurations can lead to indefinite data expansion due to accumulating "deleted" versions. Properly configured purging involves segmenting shards and periodically merging segments to

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-THPDP047

physically remove outdated documents. All operations, including indexing, can be executed through Elasticsearch's REST API, detailed later in this article.

Searching Searching stands as one of the most vital actions in ElasticSearch. Similar to indexing, Elasticsearch offers a REST API for search operations. The API presents a broad spectrum of search possibilities, ranging from basic term searches to intricate searches involving hierarchies, synonyms, language detection, and more.

Searches within Elasticsearch are score-based, employing formulas to assess document relevance against the query. This scoring mechanism can be customised to match specific requirements. Typically, cluster searches occur in two phases:

1. The primary node transmits the query to both nodes and shards, subsequently retrieving document identifiers and their corresponding scores. The master subsequently selects meaningful documents based on scores and a "size" parameter, limiting the maximum results.
2. The master sends requests to nodes to retrieve the selected documents from the previous phase. After receiving these documents, the master compiles the final result for the client. Alternative search modes, such as "query_and_fetch," conduct simultaneous searches across all shards, returning not only document identifiers and scores but also data itself. Here, the maximum results are determined by the size parameter plus the shard count.

A notable ElasticSearch configuration feature is the ability to allocate specific nodes exclusively for query operations and others for data storage, known as data nodes. This setup streamlines queries by eliminating the need to search across the entire cluster, resulting in faster search performance.

Data Management As part of the CAM system software environment, Elasticsearch Curator was installed. Curator is a tool that helps manage and maintain Elasticsearch indices. It is used for tasks like index lifecycle management, data retention, and optimising cluster performance. Curator is particularly helpful when dealing with large amounts of data and indices, as it automates the process of handling these tasks.

As the Elasticsearch cluster accumulates data, the number of indices can grow substantially. Curator can be used to manage the lifecycle of indices, such as closing or deleting old indices to free up resources.

For the CAM cluster, curator is configured to delete logs after 90 days so they are limited in a controlled manner.

Through data retention policies (e.g. keeping data for a certain period and then removing it), Curator can help automate the process. It can identify indices that have reached a certain age and perform actions to delete or archive them.

Elasticsearch indices can become inefficient over time due to factors like segment fragmentation. Curators can optimise indices by triggering force merges, which improves search performance and reduce resource consumption.

Curator can also facilitate the process of creating and managing data backup for disaster recovery purposes. This involves taking periodic snapshots of indices and storing them in a repository.

Elasticsearch aliases provide a way to point to one or more indices as a single entity. Curators can help manage aliases by adding or removing them from indices, allowing control of which indices are actively queried.

Logstash

Included in the Elastic stack, Logstash is a data processing and enrichment tool that is often used in conjunction with Elasticsearch to manage and process large volumes of data, especially logs, and then index it into Elasticsearch for efficient searching and analysis.

Logstash can ingest data from various sources, including log files, message queues, databases, APIs, and more. It provides input plugins that allow users to configure Logstash to fetch data from these sources.

In the CAM software, data is ingested as log files into Elasticsearch for fault finding and troubleshooting purposes. After data is ingested, Logstash can perform various processing tasks on it. This includes parsing raw log data into structured fields, transforming data, enriching it with additional information, and performing filtering or conditional actions.

Logstash can enhance the data with context by adding additional information. For example, it can perform IP address lookups to determine geolocation or correlate data with external data sources. Logstash can transform data formats, such as converting logs into a standardised format or enriching them with calculated fields.

Logstash allows users to filter and exclude unwanted data, reducing the amount of irrelevant information that's indexed into Elasticsearch. Once data is processed, Logstash can send the transformed and enriched data to Elasticsearch for indexing. It uses the Elasticsearch output plugin to accomplish this.

ELK STACK DEPLOYMENT IN CAM

ELK Stack

All LXC nodes host a Logstash instance. As CAM system logs flow into Logstash, it employs configurable filters and parsers to extract valuable information and structure the data in a standardised format. This enriched data is then forwarded to Elasticsearch for storage and further analysis.

The visualisation and user interaction layer of the ELK stack is embodied by Kibana. This dynamic tool empowers users to create interactive dashboards, visualisations, and reports that unveil insights from the stored data. Within the CAM environment, Kibana instances are set up in all of the three LXC container nodes. These instances establish a connection with the Elasticsearch cluster via the proxmox servers, as shown in Fig. 1, retrieving the structured CAM system logs. Through Kibana's intuitive interface, custom dashboards can be craft, that visually represent the trends, anomalies, and performance metrics present within the CAM system logs.

This fusion of Elasticsearch, Logstash, and Kibana forms a symbiotic relationship, allowing the CAM system

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-THPDP047

users to seamlessly navigate, comprehend and search through the logs to find instances where and when issues occurred on the system using the log data.

As the CAM system logs flow through this intricate architecture, a series of orchestrated events occur. The logs are captured and transmitted to Logstash, where they undergo transformation and enrichment, and are then stored within the distributed Elasticsearch cluster.

Kibana instances tap into this repository, rendering the data into compelling visualisations that provide insights into the intricacies of the CAM system's behaviour. Whether it's tracking performance metrics, detecting anomalies, or uncovering patterns, Kibana's dashboarding capabilities empower users to derive actionable insights.

Figure 2 shows the pipeline of CAM logs.

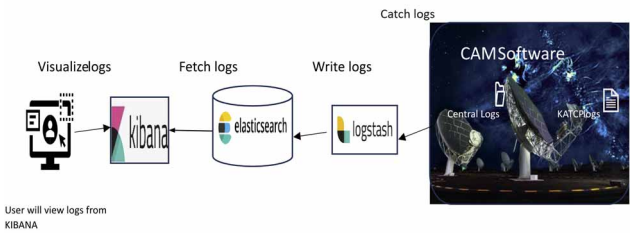

Figure 2: How Logs move from CAM to the User.

Proxmox

The orchestration of this ELK cluster within the CAM proxmox virtual environment introduces another layer of complexity and flexibility. Proxmox's virtualization capabilities enable the abstraction of hardware resources, providing a scalable and manageable infrastructure. Distributing the ELK cluster across our three proxmox hosts enhances resilience and ensures that the cluster remains operational even in the face of node failures. Moreover, the container-based virtualization offered by LXC further optimises resource utilisation and isolation, ensuring that each component of the ELK stack operates smoothly without interference.

Cluster Capabilities

Within the CAM system, the cluster benefits extend beyond scalability and availability. Elasticsearch facilitates shard replication, safeguarding against data loss by creating duplicate copies across the 3 nodes in the cluster. If a replica is lost, the cluster automatically clones and redistributes a new one.

Furthermore, Elasticsearch clusters can autonomously discover other nodes. By default, upon node startup, Elasticsearch uses Zen discovery mode, utilising unicast and multicast to locate instances across all operating system ports. This mode is specified in the Elasticsearch configuration. If a compatible instance is found, and the cluster name is matched, communication is established, and the new node joins the running cluster. ElasticSearch offers various modes for this feature, including cross-server node discovery.

USING ANSIBLE TO DEPLOY ELASTICSEARCH

Deploying the ELK stack using Ansible can help simplify deployment processes, reduce errors and ensure consistency across an infrastructure. Ansible is a powerful automation tool that uses YAML files to define configurations and tasks.

Ansible is used for the ELK stack deployment in CAM because of the below reasons. Automation with Ansible streamlines the entire ELK deployment process, effectively reducing the need for manual labour and minimising the potential for errors. Furthermore, it ensures a consistent configuration across all ELK components, promoting reliable performance and eliminating discrepancies.

In terms of scalability, Ansible simplifies the process of expanding the ELK stack by allowing the definition of new instances in Ansible playbooks. The reusability of ansible playbooks is a valuable advantage, as they can be employed for future deployments, ultimately saving time and effort in the long run. Moreover, Ansible's idempotence feature guarantees that running playbooks multiple times will consistently yield the same desired outcome, preventing unintended changes to the system. The modularity of Ansible breaks down the deployment into manageable roles and tasks, making maintenance tasks significantly easier to manage.

When it comes to security, Ansible facilitates the consistent implementation of security measures across the entire ELK stack. In addition, Ansible playbooks serve as documentation, making the setup process clear and understandable for all team members involved.

Version control is another benefit, as infrastructure code can be effectively version controlled, enhancing traceability and accountability throughout the deployment process. Lastly, Ansible's flexibility is evident in its ability to work across various platforms and environments, making it a versatile tool for deployments in diverse scenarios.

ELK STACK DEPLOYMENT WITH ANSIBLE IN CAM

Setup of an ELK stack using Ansible to manage the deployment process across 3 LXC containers can greatly streamline the task and ensure consistency across the environment. This step-by-step guide outlines how to achieve this deployment while allocating the same computing resources to each container and utilising the Ansible tool.

Inventory Setup

Begin by preparing an Ansible inventory file. This file lists the hostnames of the 3 LXC containers that will be deployed for the ELK stack. The inventory file might look like plaintext.

```
[elk_nodes]
container1 ansible_host=IP_ADDRESS_1
container2 ansible_host=IP_ADDRESS_2
container3 ansible_host=IP_ADDRESS_3
```

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-THPDP047

ELK Ansible Playbook

```yaml
- name: Deploy ELK Stack
  hosts: elk_nodes
  become: true

  tasks:
    - name: Install Java
      apt:
        name: openjdk-8-jre
        state: present

    - name: Download and Install Elasticsearch
      command: >
        wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -
        && sudo sh -c 'echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" > /etc/apt/sources.list.d/elastic-7.x.list'
        && sudo apt update && sudo apt install -y elasticsearch
      notify:
        - Start Elasticsearch

    - name: Configure Elasticsearch
      template:
        src: templates/elasticsearch.yml.j2
        dest: /etc/elasticsearch/elasticsearch.yml
      notify:
        - Restart Elasticsearch

    - name: Start Elasticsearch
      systemd:
        name: elasticsearch
        state: started
        enabled: yes

- name: Download and Install Logstash
      command: >
        wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -
        && sudo sh -c 'echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" > /etc/apt/sources.list.d/elastic-7.x.list'
        && sudo apt update && sudo apt install -y logstash
      notify:
        - Start Kibana

    - name: Configure logstash

      template:
        src: templates/logstash.yml.j2
        dest: /etc/logstash/logstah.yml
      notify:
        - Restart logstash

    - name: Start Logstash
      systemd:
        name: logstash
        state: logstash
        enabled: yes

    - name: Download and Install Kibana
      command: >
        wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -
        && sudo sh -c 'echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" > /etc/apt/sources.list.d/elastic-7.x.list'
        && sudo apt update && sudo apt install -y kibana
      notify:
        - start Kibana

    - name: Configure Kibana
      template:
        src: templates/kibana.yml.j2
        dest: /etc/kibana/kibana.yml
      notify:
        - Restart Kibana

    - name: Start Kibana
      systemd:
        name: kibana
        state: started
        enabled: yes
```

Figure 3: ELK Ansible Playboo

Following the inventory setup, generate an Ansible Playbook, like the one shown in Fig. 3.

Organisation of the playbook is done by utilising Ansible roles, for example, the ELK role is created and used.

Configuration management is done using Github, which improves modularity and reusability. Roles can also include templates for Elasticsearch and Kibana configurations. For instance, the template for elasticsearch.yml.j2 might contain configuration settings like the cluster name, memory settings, and node roles.

The same computing resources are allocated to each container, and therefore it is easy to use Ansible to configure resource settings. Ansible provides modules for managing resources, like the lxc_container module for LXC.

Execution of the playbook is done using the command below, where the inventory.ini is the name of the inventory file and elk_deploy.yml with the name of the playbook file:

$ ansible-playbook -i inventory.ini elk_deploy.yml

CONCLUSION

By following the above steps, the ELK stack has been efficiently deployed using Ansible across 3 LXC containers in the CAM infrastructure system, ensuring consistent configuration and resource allocation. This setup will enable effective management and analysis of CAM system logs using the ELK stack, promoting operational insights and informed decision-making.

In conclusion, the ELK cluster represents a sophisticated interaction between Elasticsearch, Logstash, and Kibana within the CAM system environment. The dynamic cluster of LXC container nodes within the Proxmox virtual environment forms the backbone for analysing and visualising CAM system logs. This amalgamation of technologies not only streamlines the handling of vast data streams but also empowers user to uncover invaluable insights hidden within the logs.

By utilising Ansible for deploying ELK containers across multiple Proxmox virtual environments, the deployment process is streamlined, improving consistency, and increasing the efficiency of managing the infrastructure.

Consideration will soon be given to using Docker in the deployment of the ELK stack, as this has become a standard approach in industry that provides flexibility and simplicity in management.

Using Ansible alongside Docker can further enhance the deployment process by automating the setup and configuration of a Dockerised ELK stack.

REFERENCES

[1] Elasticsearch, https://github.com/elastic/elasticsearch

[2] Kibana, https://github.com/elastic/kibana

[3] Logstash, https://github.com/elastic/logstash

[4] The Complete Guide to the ELK Stack, https://logz.io/learn/complete-guide-elk-stack/

[5] Ansible, https://www.ansible.com

[6] Elastic, https://www.elastic.co