Bachelor's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

Karl-Henrik Horve
Marcus Eugen Brockstedt Mathisen
Fredrik Leonard Stenersen
Bjørn Kristian Strand

Creating a Scalable Log Analytics Pipeline with GitOps

Bachelor's thesis in Bachelor in Digital Infrastructure and Cyber Security
Supervisor: Erik Hjelmås
Co-supervisor: Jørn Skjerven
May 2024

NTNU
Norwegian University of Science and Technology

Karl-Henrik Horve
Marcus Eugen Brockstedt Mathisen
Fredrik Leonard Stenersen
Bjørn Kristian Strand

Creating a Scalable Log Analytics Pipeline with GitOps

Bachelor's thesis in Bachelor in Digital Infrastructure and Cyber Security
Supervisor: Erik Hjelmås
Co-supervisor: Jørn Skjerven
May 2024

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

NTNU
Norwegian University of Science and Technology

NTNU
Kunnskap for en bedre verden

Department of Information Security and Communication Technology

DCSG2900 Bachelor Thesis

Creating a Scalable Log Analytics Pipeline with GitOps

Author:
Karl-Henrik Horve
Marcus Eugen Brockstedt Mathisen
Fredrik Leonard Stenersen
Bjørn Kristian Strand

May, 2024

Abstract

The increasing importance of observability and log analytics in incident response and operational monitoring has driven the need for scalable and efficient logging solutions, particularly in cloud-native environments. The Norwegian University of Science and Technology's Security Operations Center (NTNU SOC) recognizes this need and seeks to enhance its capabilities in managing the increasing volume and diversity of log data.

This project focuses on the design, implementation, and evaluation of such a pipeline, utilizing open-source components like Apache Kafka, Vector, and Opensearch. Throughout the project's lifecycle, we adhered to Infrastructure as Code (IaC) and GitOps principles, and methodologies. The finished log analytics pipeline, designed and implemented in accordance with the NTNU SOC's requirements, showcases a viable, scalable, and open-source solution for effective log management within cloud environments. Future work will address the identified bottlenecks, expand the pipeline functionality, and incorporate further security measures to create a production-ready solution.

This POC log analytics pipeline not only demonstrates the feasibility of an open-source, cloud-native solution for log management but also provides a reference architecture for the NTNU SOC and other organizations seeking to enhance their observability capabilities.

Sammendrag

Norges Teknisk-Naturvitenskapelige Universitet (NTNU) sin SOC ser økende behov for observasjonsevne og logganalyse for å effektivt håndtere hendelser og overvåke driften av infrastrukturen de beskytter. Dette er spesielt viktig i kontainermiljøer, hvor en kontainer kan være avsluttet når en hendelse oppdages, noe som gjør logganalyse essensielt for å forstå hva som har skjedd. For å forbedre sin sin støtte for cloud-native logging, foreslo NTNU SOC å bygge en proof-of-concept (POC) logganalyse-pipeline. Prosjektet fokuserer på å designe, implementere og evaluere en slik pipeline, med bruk av åpen kildekode-verktøy som OpenSearch, Apache Kafka og Vector. I tillegg skulle prosjektet bygge på IaC-verktøy som Terraform og Ansible for automatisert provisjonering av infrastrukturen i SkyHiGh, NTNU sin implementasjon av OpenStack-skyen.

Projektet har resultert i en fungerende logganalyse-pipeline, som demonstrerer hvordan man kan håndtere store mengder data fra ulike kilder i et cloud-native miljø. Pipelinen er definert som kode, noe som sikrer sporbarhet og reproduserbarhet, og er implementert ved hjelp av GitOps-metodikk.

Gjennom testing av pipelinen ble dens funksjonalitet og skalerbarhet evaluert. Vi oppnådde en fungerende referanse arkitektur for en skalerbar logg innhenting og prosesserings-infrastruktur, som vil fungere som et utgangspunkt for videre arbeid. Imidlertidig avdekket testingen også ytelsesutfordringer med OpenSearch under høy belastning. Dette indikerer behov for videre optimalisering av OpenSearch-konfigurasjonen for å håndtere store datamengder effektivt.

Preface

The Norwegian University of Science and Technology's Security Operations Center (NTNU SOC) recognizes the growing importance of observability and log analytics. In an ever-evolving threat landscape, robust log analytics capabilities are essential for threat detection, efficient incident response, and overall resilience of digital infrastructure. The ability to collect, process, and analyze vast amounts of log data in real-time is fundamental for information security teams to identify anomalies and potential threats and to respond quickly to security incidents.

This thesis, titled "Creating a Scalable Log Analytics Pipeline with GitOps", marks the completion of our studies in the bachelor's course "Digital Infrastructure and Cyber Security" at the NTNU in Gjøvik. We are grateful to the people at the NTNU SOC for providing valuable insight and guidance throughout the duration of the project. We would also like to extend our thanks to our academic supervisors at NTNU for providing guidance and constructive feedback both in, and outside of meetings. We hope this thesis provides a comprehensive overview of our journey in designing and implementing a proof of concept log analytics pipeline.

Table of Contents

List of Figures

List of Tables

Acronyms

2FA Two-Factor Authentication. 67

API Application Programming Interface. 7–9

CIS Center for Internet Security. 63

CISA Cybersecurity and Infrastructure Security Agency. 63, 64, 66, 67

CRD Custom Resource Definition. 60, 74

FOSS Free Open Source Software. 2, 3, 5, 76, 77, 79, 80

GUI Graphical User Interface. 77

HPA Horizontal Pod Autoscaler. 43

IaC Infrastructure as Code. i, ii, 1–3, 5, 6, 8, 11, 17, 18, 78, 79

KMS Key Manager Service. 67

NSA National Security Agency. 63, 64, 66, 67

NTNU Norwegian University of Science and Technology. ii, iii, 1, 43, 63

NTNU SOC NTNU Security Operations Centre. i–iii, 1–3, 5, 8, 12–14, 16, 18, 76, 77, 79, 80

POC Proof Of Concept. i, ii, 2, 3, 30, 44, 63, 64, 66, 67, 76, 78–80

RBAC Role Based Access Control. 66, 67

SOC Security Operations Centre. ii, 1, 43, 63

SSH Secure Shell. 32, 36

VRL Vector Remap Language. 43, 45

YAML Yet Another Markup Language. x, 19, 24, 78

Glossary

Ansible An open-source IT automation tool used for configuration management, application deployment, orchestration, and provisioning . 17–21

Broker A server in an Apache Kafka cluster responsible for storing and managing log data within topics and partitions. Brokers handle requests from producers to write messages and from consumers to read messages. 37, 39, 41, 42

Cruise Control Is a tool designed to manage several operational aspects of Kafka clusters, such as partition rebalancing, Broker failure detection, or Replica distribution for topics. 74

FluxCD A GitOps toolkit for Kubernetes that automates the deployment and management of applications based on configurations stored in Git repositories. .. 8, 9, 30–33, 35

GitOps An operational framework that applies DevOps best practices to infrastructure automation, using Git as the single source of truth for infrastructure and application configuration. i, 10, 17, 18, 34

Grafana An open-source visualization and analytics platform that allows you to query, visualize, and alert based on metrics. 52, 57, 58

Kafka A distributed event streaming platform for building real-time data pipelines. 16, 39, 42, 52, 54, 55, 74–76

Kubernetes An open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications. 1–3, 7–9, 15, 16, 52, 54–56, 74–77

Observability Pipeline A system that collects, processes, and routes telemetry data (logs, metrics, and traces) from various sources.. 77

OpenStack An open-source cloud computing platform that provides Infrastructure as a Service (IaaS). It allows users to provision and manage virtual machines, networks, and storage resources in a cloud environment. ii, 6, 29, 30, 75

Orchestration The automated configuration, coordination, and management of computer systems, middleware, and services. 7, 18, 19

Partition A partition is a segmentation of a topic into several logs, where each partition is an ordered, immutable sequence of records that are continually appended in seperate kafka brokers. 41

Prometheus An open-source monitoring and alerting toolkit. It collects time-series data, stores it in a time-series database, and provides a query language and visualization tools to analyze and create alerts based on that data. 26, 28, 52, 55–57

Provisioning The process of allocating and preparing IT infrastructure resources, like virtual machines, networks, and storage, to support the deployment of applications and services. x, 18, 19, 21, 32, 33

Terraform An open-source Infrastructure as Code (IaC) tool used for building, changing, and versioning infrastructure. It lets you define and provision infrastructure resources declaratively across various providers. ii, 8, 21, 77, 80

Topic A Kafka topic is a logical channel for organizing and storing a stream of records to brokers, allowing producers to send data and consumers to read it. 41, 42, 48, 51

Vector An open-source observability data pipeline tool used to collect, transform, and route logs, metrics, and traces.. 8, 74, 76, 77

Chapter 1

Introduction

1.1 Background

NTNU Security Operations Centre, from now referred to as NTNU SOC, is an operations center affiliated with the digital security section at NTNU's IT division, and is responsible for coordination of the operational digital security at NTNU, and is NTNU's official point of contact for digital security incidents. NTNU SOC provides services like Intrusion Detection, Technical Security Analysis, and Incident Management for all NTNU campuses [22]. NTNU is Norway's largest university measured in the number of students, with 43 194 students in 2023 [29]. As development in the digital sector continues, the infrastructure NTNU SOC is monitoring grows larger and more complex. To monitor and protect NTNU's assets across all campuses, NTNU SOC needs a log collection pipeline capable of handling peak load generated by their dynamic workload.

A log collection pipeline is a system tasked with collecting logs from many sources and aggregating them before indexing them in a log management system like OpenSearch or Splunk. Security Operations Centres can search through logs for debugging, threat-hunting, and other operational tasks. Complete and accurate log data is a prerequisite for reliable performance analysis, threat-hunting, and infrastructure troubleshooting within log management systems; therefore, the log collection pipeline must ensure that all logs written to the pipeline are aggregated correctly and indexed in the log management system.

1.1.1 Academic Background

The members of the group are all in the last year of their bachelor's degree in Digital Infrastructure and Cybersecurity at NTNU Gjøvik and have, through their studies, acquired knowledge in topics of hosting infrastructure, securing services, and designing scalable applications. This serves as the basis of knowledge the group builds upon in this thesis.

1.1.2 Knowledge Gaps

Since Kubernetes is a technology not covered by the study program, the group had to learn what Kubernetes does and how to run both stateless and stateful applications in a Kubernetes cluster. In addition, the group members that have not had Infrastructure as Code had to acquire knowledge about the concept of Infrastructure as Code and the technical use and implementation of CI/CD pipelines.

1.1.3 Thesis Rationale

The group chose this task based on several factors. The high degree of technical work in log collection and Infrastructure as Code was a big motivator as this is of high interest amongst the group members, both during the degree through subjects and part-time jobs and for further work after the degree. In addition, Kubernetes is the leading tool for orchestrating containerized applications and is in high demand within the industry. Since we have little to no experience with Kubernetes from the degree, we chose this task to gain hands-on experience and improve our competency with Kubernetes and Infrastructure as Code.

1.2 Problem Area

As the cybersecurity landscape constantly develops and the number of services running in the cloud continues to grow, the need for a consistent and scalable log collection and management system increases. Logging important infrastructure is a key element in ensuring desired performance and detecting and taking action against any threat actors attempting to conduct malicious activity against the infrastructure. Therefore, the NTNU SOC wants us to develop a Proof Of Concept (POC) infrastructure for a log collection pipeline that is both scalable and can be easily redeployed. The pipeline covers all stages, from extracting logs generated by each service or machine to being indexed into a log management system for further analysis.

1.3 Scope and Framework

The thesis covers the implementation of a POC infrastructure for log collection and processing pipeline through IaC on Kubernetes. Since the product is a POC, some features that a production-ready infrastructure requires have been scoped out.

Firstly, since the product is a POC, most of the security features will not be configured to be production-ready.

Additionally, the NTNU SOC had already decided to use OpenSearch and created Ansible playbooks to configure this to their needs. Configuring OpenSearch has been scoped out, as the group would get access to these playbooks to configure OpenSearch.

1.3.1 Timeframe

The project duration is from January 8th, 2024, to May 21st, 2024. We agreed to have a first draft ready by May 6th, giving us three weeks to finalize the report.

1.3.2 Software

The client requires that all software used in the POC infrastructure is Free Open Source Software (FOSS) and that the source code of the finished product is licensed under a FOSS license. Defining what licenses comply with the definition of FOSS licenses is further discussed in section 2.1.1 Licensing.

1.4 Project Goals

1.4.1 Learning Goals

- **L1:** Gain practical experience in implementing and configuring cloud-based infrastructure.
- **L2:** Acquire skills in leveraging GitOps practices to optimize and automate infrastructure workflows.
- **L3:** Develop the team's understanding of Kubernetes principles, particularly in relation to scaling and managing cloud-native applications.
- **L4:** Learn to design and implement multi-component software solutions as Infrastructure as Code.

1.4.2 Effect Goals

- **E1:** Simplify data-flow management by implementing flow-based programming tools.
- **E2:** Automate and streamline infrastructure management with GitOps methodologies.
- **E3:** Reduce licensing costs for NTNU SOC.

1.4.3 Result Goals

- **R1:** Have a POC solution that can collect a log from the source, process it, and index it for long-term storage in OpenSearch.
- **R2:** Have the POC solution be defined and deploy-able through IaC, and adheres to GitOps principles.
- **R3:** Present a log analytics pipeline capable of dynamically scaling in response to fluctuating traffic volumes.

1.5 Target Group

The main target group of the thesis is NTNU SOC as they intend to use the POC infrastructure as a reference for improving their current infrastructure. Additionally, since the client has required that only FOSS is used in the product, and the source code of our product must be licensed under a FOSS license, additional target groups include any organization or individual that wishes to deploy a log collection and management system through IaC on Kubernetes.

1.6 Thesis Structure

The thesis consists of 7 sections.

Requirements Spesifications details the framework the group had to work with, what methods and principles the group chose for the project, and lastly, what choices the group made regarding the software included in the solution.

Methodology goes over what methods the group has chosen to develop the product, and provides a high-level overview of the solution.

Implementation & Challenges details how the group has implemented the solution, what configuration choices have been made, and outlines the challenges the group has faced in the technical implementation.

Security details how the group has created the infrastructure regarding security, what measures have been taken, and what parts have been left out.

Results details the performance testing the group has done on the infrastructure.

Discussion details the parts of the process where the group has made important decisions, and the rationale behind them.

Conclusion discusses if the project objectives have been achieved and what work can be expanded upon.

Chapter 2

Requirement Specifications

2.1 Solution Requirements

2.1.1 Licensing

The NTNU SOC required that all the software used in the final product must be licensed under a FOSS license, this is to reduce their current licensing costs.

To determine what software is eligible for use in the product, the group had to define what licenses permit the use of the software in a production setting without cost. The GNU Project[1] is a collaborative initiative to create free and open-source software and maintains a strict definition of free software principles. The initiative has a list of licenses that comply with its definition of free software [2], which has served as a foundation for the group to determine what software is eligible for use in the project.

The group has decided to uphold the requirements of the NTNU SOC by only using software in the finished product with an approved license from the GNU Project. Additionally, the final product created by the group is to be licensed with an approved license from the GNU Project.

2.1.2 Design

For the product's design, the NTNU SOC required that the product and underlying infrastructure be configured and deployed through IaC technologies. Additionally, since the group decided to deploy the pipeline on Kubernetes, the NTNU SOC required that the cluster automatically scale to meet the demand of the current workload without operator interference.

[1] https://www.gnu.org/
[2] https://www.gnu.org/licenses/license-list.html

2.2 Chosen Methods and Principles For The Pipeline

2.2.1 Infrastructure As Code

The practice of managing infrastructure as code (IAC) involves using code to define and manage IT infrastructure, such as servers, networks, or storage. This approach simplifies and streamlines infrastructure management by allowing administrators to describe the desired state of their infrastructure in code, which is then automatically executed to create, configure, and manage the infrastructure's resources.

The three core practices as defined by the cloud infrastructure engineer Kief Morris [20, p.9] are as follows:

- "Define everything as code"
- "Continuously test and deliver all work in progress"
- "Build small, simple pieces that you can change independently"

2.2.2 CI/CD

Continuous Integration: Continuous Integration is a software development practice where a team of developers integrates one or more changes daily [26, p. 5]. A common tool used for CI is Git, which is the most widely used software for version control [26, p. 5]. This practice allows developers working on the same project to store their code in a central repository, ensuring version control. Code pushed to a Git repository is merged with other developers' code, and automated testing steps ensure the code can be merged successfully.

Continuous Delivery/Deployment: Continuous Deployment is a software engineering approach in which software changes are delivered frequently through automated deployment processes [26, p. 8]. Tools such as GitLab Runners can pull code from a Git repository and perform the necessary configurations to make the application deployable automatically. Continuous Delivery, while involving many of the same principles, requires manual intervention by a developer before deployment.

2.2.3 GitOps

Triggered by a talk at the Agile 2008 conference [20, preface. xix] by Andrew Clay-Shafer and Patrick Debois, the DevOps movement has grown with the IaC concept. DevOps is a set of practices that aims to reduce resistance between developing and operating software applications.

The process of GitOps emphasizes communication and collaboration between the developers and the people operating the applications, focusing on automated deployment via Continuous Integration/Continuous Delivery (CI/CD). GitOps methodology, which is an extension of DevOps, encourages not just the applications to be managed through Git but also the infrastructure. [26, p. 26]. By managing all the infrastructure deployments in a version-controlled and structured manner, we establish a single source of truth, ensuring that all changes made to the pipeline are tracked, easier collaboration, and much simpler recovery from misconfiguration. Additionally, having all infrastructure defined in Git allows for streamlined deployment with CI/CD pipelines.

Gitlab runners

A Gitlab runner is a machine that runs the jobs defined in our Gitlab CI/CD pipelines. The runners can be registered in various environments, either on physical machines, virtual machines locally, or as virtual machines in the cloud, as we do in OpenStack. [11] When a pipeline triggers,

Gitlab assigns the job to a runner, who then carries out the instructions, which could be deploying virtual machines, as is described in 4.4.

2.3 Chosen Technologies For The Pipeline

This section covers the different technologies and tools implemented in the solution and aims to lay a technical foundation for the rest of the thesis.

2.3.1 Kubernetes

Kubernetes is an open-source container Orchestration platform introduced by Google in 2014[18] and builds upon their decade-long previous experience with clustering management through Google Borg.[31]. The platform lets developers define containerized application deployments declaratively while providing robust automation and scaling capabilities for modern distributed systems. Kubernetes runs deployments in a cluster comprising two main components: the control plane and the worker nodes.

Control Plane:

The control plane detects changes to the cluster and cluster configuration and performs operations to ensure the cluster is running in the desired state. It performs tasks like detecting and restarting failing nodes, assigning newly created pods to available nodes, and connecting the cluster to the cloud provider API[18].

Worker Node:

The worker nodes are configured with the container runtime and are responsible for hosting the containerized applications. They can be virtual or physical machines, and can be configured to run single-application pods or multi-application pods [18].

Helm Charts:

Helm Charts are an abstraction of application deployment on Kubernetes. The charts are a collection of configuration files that define a set of required Kubernetes resources, and the application's configuration can typically be customized by modifying a values.yaml file [5].

2.3.2 Grafana K6

Grafana K6 is a load-testing tool used to simulate traffic to different components in the infrastructure [19]. In this project, the group has used Grafana K6 to emulate log collection and forwarding software to test the performance and auto-scaling capabilities of the pipeline.

2.3.3 Kafka

Apache Kafka is an open-source distributed event-streaming platform from the Apache Software Foundation[15]. It works as an endpoint for incoming data streams and sorts the different streams into a set of queues (topics). These queues can be configured to store data until certain criteria are met, working like a buffer for the incoming events before they are further processed.

Strimzi Project:

Strimzi is an open-source Apache Kafka operator that facilitates the deployment of Apache Kafka on Kubernetes. The Strimzi Project provides images, operators, and custom resource definitions for Kafka designed to run on Kubernetes. Additionally, Strimzi supports monitoring and metrics tools

like Prometheus for extracting metrics from the different components that the Strimzi operator deploys[28].

2.3.4 Vector

Vector, an open-source project maintained by Datadog, is a versatile tool for building observability pipelines. Its primary function is collecting, processing, and routing data, [34] including logs and metrics within distributed systems. The tool uses a component-based approach, where each component performs a specific function, like collecting data from various sources, transforming, enriching, or routing the data to different destinations. Furthermore, Vector's extensive list of pre-built sources, sinks, and transformations, along with its vendor-neutral nature, makes it compatible with a wide range of systems and technologies.

2.3.5 OpenSearch

OpenSearch, a distributed search and analytics engine derived from Elasticsearch, is the chosen solution for log data indexing at NTNU SOC. This decision follows Amazon AWS's fork and continued maintenance of OpenSearch after Elastic NV transitioned Elasticsearch and Kibana to non-open-source licenses[8]. Leveraging the OpenSearch Dashboard, the NTNU SOC analytics team gains a comprehensive infrastructure monitoring suite. This includes custom dashboard creation, alerting based on query triggers, and active threat-hunting capabilities within log data.

The extensive range of first-party, free features and plugins available within OpenSearch further enhances the SOC's ability to visualize and analyze the state of its monitored infrastructure[13].

2.3.6 Terraform

HashiCorp Terraform is an IaC tool that allows you to define computer and network infrastructure as declarative, human-readable code. Terraform serves as a middle layer between the systems administrators and the cloud platforms where the infrastructure is hosted, takes the declarative configuration files for the infrastructure, and makes requests to the cloud provider API endpoint to provision the infrastructure as shown in 4.3.1. Terraform has thousands of providers, which enables Terraform to work with any service with an accessible API. [33]. This allows for consistent infrastructure deployments and integration with other IaC technologies like CI/CD pipelines for automatic deployments and reconfigurations as detailed in 4.4. Terraform is also idempotent, meaning it will only execute commands to apply changes to the components not currently in the desired state.

2.3.7 Ansible

Ansible is an automation engine capable of handling provisioning, configuration management, and application deployment through a set of playbooks written by a systems architect or administrator. The playbooks define what jobs should be carried out and what systems the plays should be applied to. Ansible is a decentralized solution, meaning that the Ansible program/binary is not required to be installed on the hosts it configures. Instead, Ansible is installed on a machine called the Control Node, and this node uses Ansible playbooks to perform a set of plays on the defined target devices[16].

2.3.8 FluxCD

FluxCD is a continuous delivery solution for Kubernetes that integrates with Git providers to enable a GitOps workflow. FluxCD can be configured to observe an application repository for

configuration changes and apply these changes to the Kubernetes cluster through the Kubernetes API extensions. This means that the group can configure applications by making changes to the Kubernetes manifest and push the changes to Git, and FluxCD will handle the deployment, eliminating the need for an operator to run kubectl commands[9].

Chapter 3

Methodology

3.1 Project And Development Process

The project plan for this thesis is outlined in Appendix A. This chapter will discuss the methods utilized by the group for developing the solution and writing the thesis. Additionally, the chapter details the methods used to decide what software was chosen for the project, and how the group designed the solution.

The group has chosen a combined process of Kanban and GitOps (2.2.3) for this thesis. Since certain group members have external commitments that render them unavailable for a certain amount of time, the group opted not to adopt a Scrum-based method, as the short timescale of the sprints and the daily scrum meetings would be challenging to enforce. The use of Kanban and GitOps allows the group to have a structured view of the process for the project as a whole, and by utilizing the GitLabs integrated issue and issue board features, the group was able to work with a GitOps workflow to document the progress on the project better.

3.1.1 GitLab Issues And Kanban Board

The group used the GitLab integration issues and issue board to structure the issues and Kanban board. This was chosen to minimize the use of third-party apps the group had to keep track of during the project and to allow the group to integrate the Kanban and GitOps workflow closer.

Having the issues in GitLab allowed the group to organize them in their related repositories instead of having a monolithic issue board with all issues related to the project. For bigger sections of the project where issues would span multiple repositories, the group also linked the issues to epics or milestones to better visualize progress on a larger scale.

Since most of the context and progress for the issues were documented inside the issues themselves with tagged commits and issue history, the group designed the Kanban boards with a minimal approach. For the kanban structure, the group organized the tasks into four states: open, doing, blocked / waiting, and closed. The open state represents issues that any group member can start on. To work on the issue, one or more group members assign the issues to them and move them to the doing queue. Issues in the doing queue are issues that are currently being worked on, and no conflicts are hindering further progress on the issue. When a conflict arises, halting progress to an issue, or the need for input from the client or the supervisors is required, the issue is moved to the blocked / waiting queue until the conflict is resolved. When an issue has been resolved, the issue is moved to the done queue, and marked as resolved.

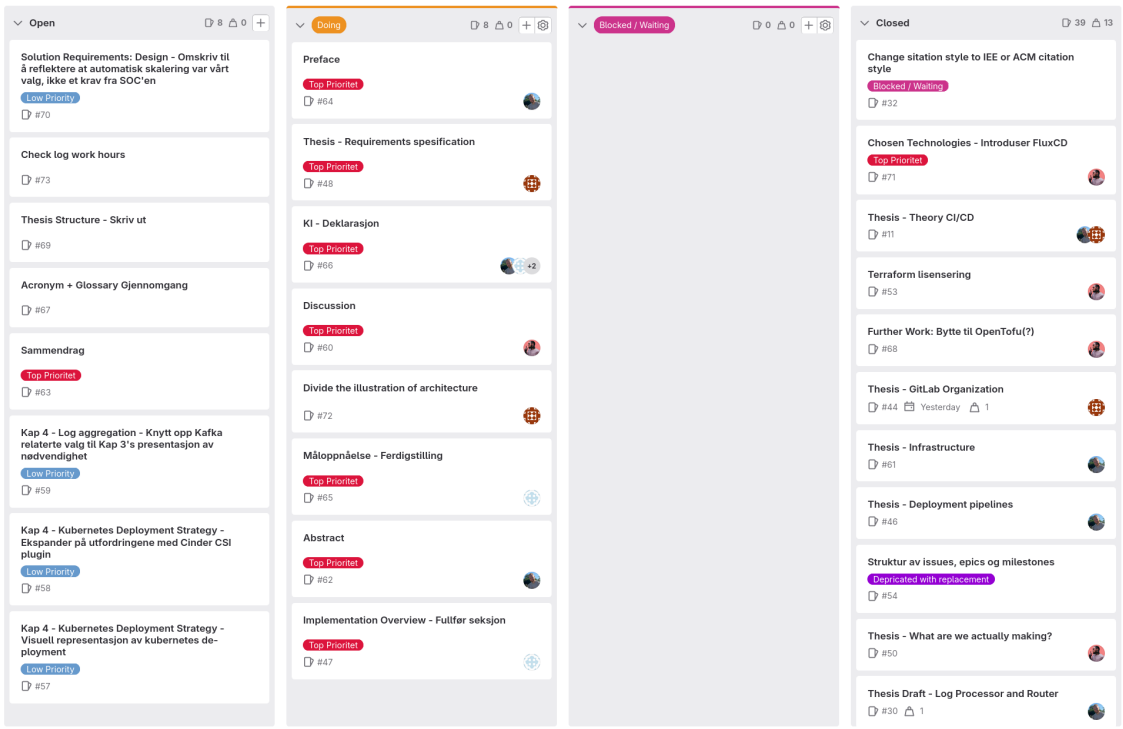


Figure 3.1: Snippet of the Kanban board for the Thesis repository

3.1.2 GitOps Workflow

It was established early on in the project that the group would focus on developing the product using IaC practices. Therefore, the group chose to implement a GitOps workflow, using CI/CD solutions to automate the deployment and configuration of the solution. This allowed the team to redeploy the infrastructure without manual configuration of the components and streamlined the development and subsequent troubleshooting steps by applying changes to the infrastructure by committing changes to the source-code repositories.

The group tagged each commit to the source-code repositories with the corresponding issue to integrate the GitOps and Kanban methods. This ensured better documentation on the issue's progress, as the issue will contain a history of all the commits regarding that specific issue.

3.1.3 Timeline

As mentioned above, the group would collect issues into bigger epics and milestone collections. By doing this, GitLab would give us a timeline of the project's different parts and visualize the progress for each part.

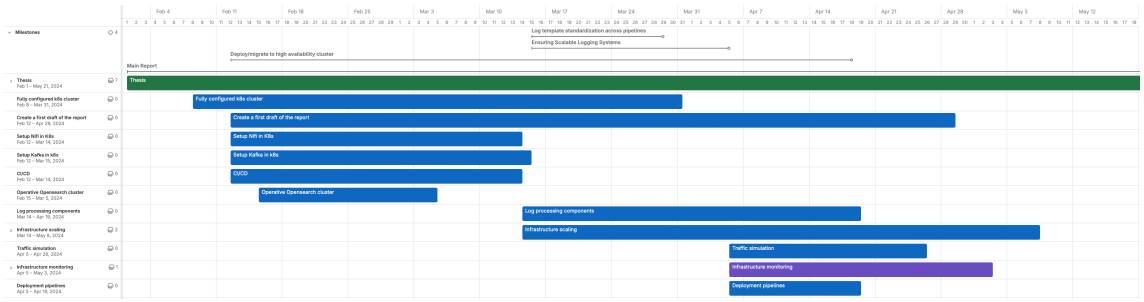


Figure 3.2: Project timeline

3.1.4 Meetings

Meetings with the client

An important part of setting the scope for the task, and for defining requirements for both the solution and its components were meetings with the client, NTNU SOC. The group decided in coordination with the NTNU SOC to arrange meetings as needed during the project's duration, and would instead use a collective Teams workspace for communication outside of the meetings. Additionally, the group was granted access to work on the premises of NTNU SOC, which allowed the group to clear up any questions about the project in a more informal conversation.

Meetings with supervisors

The group was assigned two supervisors for the project and started by arranging meetings as needed. This was later changed to weekly meetings to display progress and discuss further work on both the solution and the thesis. The group would send the thesis to the supervisors two days before the weekly meetings to give the supervisors enough time to revise the progress of the thesis for further discussion with the group during the weekly meetings.

Group meetings/work sessions

Since the group had multiple members with obligations requiring them to take extended periods of time away from campus [1], and since one member of the group was a remote student, the group elected not to have weekly or daily meetings, but would instead meet one day a week for a collective working session. These working sessions were planned so that all members of the group could participate, and they were mandatory to participate in.

3.2 Architecture

3.2.1 Components

To create the solution, the group has divided the architecture into three main components: a log forwarder tasked with collecting, formatting, and forwarding logs, a log processor responsible for transforming and routing logs, and a storage system for indexing logs for further analytic work.

[1] Extended: potentially two or more days a week, two or three times a month

Figure 3.3: Solution components overview

Log Forwarder

The first component of the solution is the log forwarder. This component is tasked with collecting logs from the system or services that the NTNU SOC wish to monitor and perform basic formatting on the log data before forwarding the logs to the log processor component of the pipeline.

The log forwarder sits on the same server as the service or server that the NTNU SOC wishes to collect logs from. This means that the log forwarder is decoupled from the rest of the solutions infrastructure, and must be configured on a per-host basis, as visualized in figure 3.4.

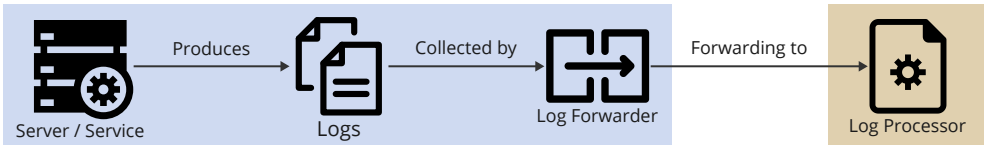


Figure 3.4: Visualization of Log Forwarder responsability

Log Processor and Router

The log processor and router component of the solution is further subdivided into two smaller components: an event streaming platform and a log transformation and routing component. Together these components are tasked with ingesting logs from the log forwarders and routing them through a set of processors before forwarding them to the storage solution.

The first sub-component of the Log Processor component is the event streaming platform 3.5. This component collects the logs sent by the log forwarders and organizes them into a set of N queues. These queues determine how the next step of the component handles the logs. Sorting the logs into queues through an event streaming platform allows for scaling either the log forwarders or the log transformers independently of each other, and works as a buffer for the log processor during activity spikes.

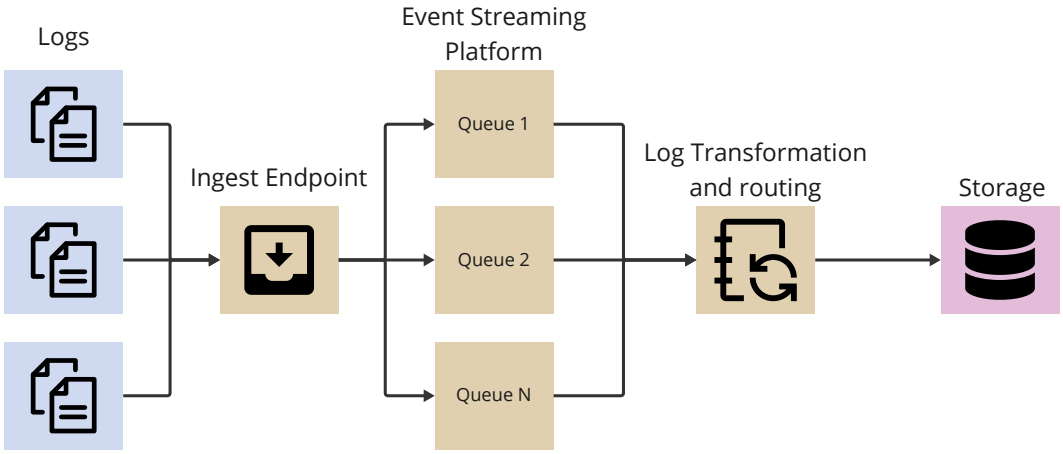


Figure 3.5: Overview of the event streamer functionality

The second sub-component is the log transformation and routing solution. This component retrieves logs from the queues of the event streaming platform and performs transformations on the log data by routing them to and from an internal or external processor. Internal processors are processors built into the solution for the log router that the group has chosen. In contrast, external processors can be microservices hosted by the NTNU SOC independent of the log router 3.6.

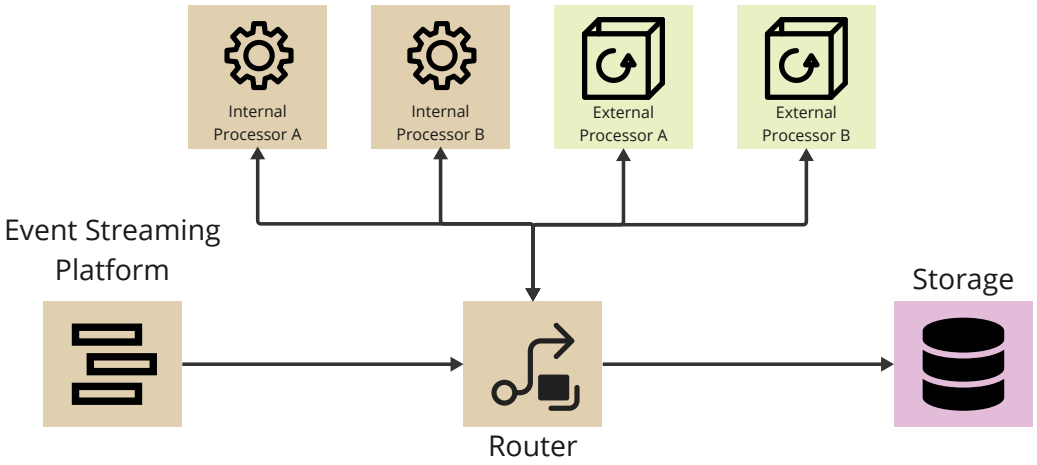


Figure 3.6: Overview of the processor and router architecture

Log Storage

The final component of the solution is the log storage component. When the logs have been processed, the logs are indexed into one of the N indexes in the log management system for further work by the analyst team at NTNU SOC. Additionally, the router has the capability to forward logs to an API endpoint to share log data with cooperating partners, illustrated in 3.7.

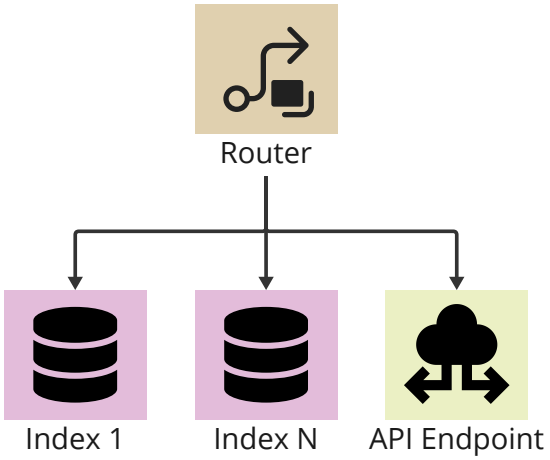


Figure 3.7: Overview of the storage architecture

3.3 Implementation Overview

This section provides a high-level overview of how the project components are implemented, how they interact, and what choices have been made to implement the solution. This section will not detail the technical implementation and the challenges associated with the software development process as this is covered in Chapter 4.

3.3.1 Infrastructure

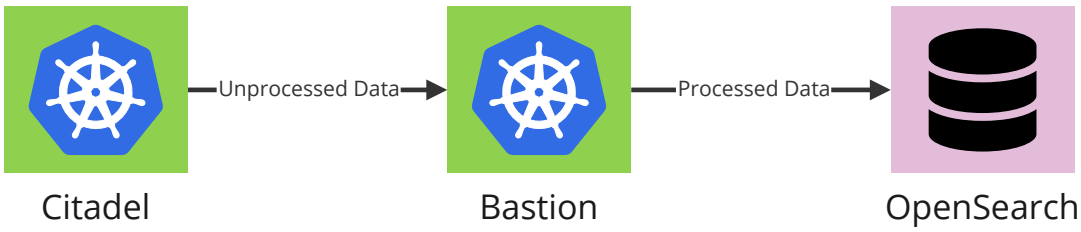


Figure 3.8: Cluster traffic overview

The team has elected to use two solutions to host the project's infrastructure. Specifically, it has elected to run some services in a Kubernetes cluster while others run directly on virtual machines.

Because the solution's components have different use cases and needs, the team has chosen the underlying infrastructure to fit each component's needs. The team has used two other hosting solutions for the components: running applications on Kubernetes or running applications directly on virtual machines.

Log forwarder component

In our implementation, the group has used Grafana K6 as a log forwarder to simulate log traffic. In a production setting, this would be exchanged for a solution like FluentBit to collect the logs from the applications or servers, before forwarding them to the log processing and routing component.

Log Processing and Routing Component

For various reasons, the team decided to host the applications inside a Kubernetes cluster for the

solution's log processing and routing component.

Firstly, the NTNU SOC wanted the team to integrate a solution for implementing containerized microservices into the solution.

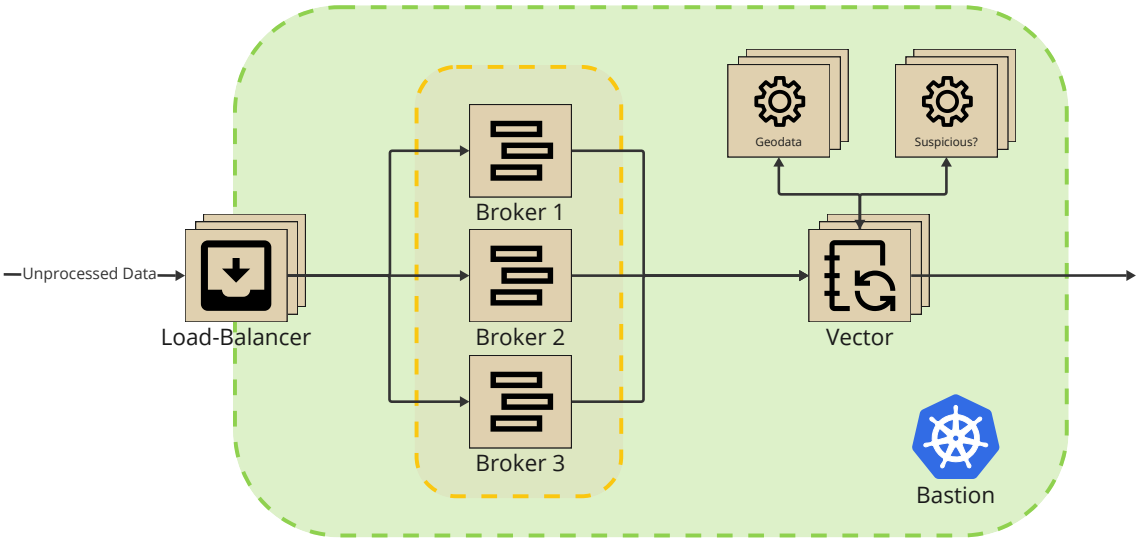


Figure 3.9: Pipeline components in the Bastion cluster

Secondly, the team envisions this as part of the solution where automatic scaling to meet the demand of a higher workload is most critical. In the events where there are no more available resources to process incoming logs, the log processor component will drop log data it cannot process, causing incomplete log history. We want to avoid this, as having complete log data is an important factor for the NTNU SOC. By running this component in a Kubernetes cluster, the operator can define threshold values for automatic cluster scaling, meaning that the cluster will provision new nodes without operator input. Likewise, predefined threshold values will cause the cluster to decommission excess worker nodes when the component is under lower workloads.

Log Storage Component

In our solution, the log storage component runs directly on virtual machines, partly because of the need for persistent data storage and dedicated resources to handle the workload from an analytics team. Still, the team chose to run the log storage component on virtual machines because the team would get access to the NTNU SOC's playbooks for configuring this with Ansible.

3.3.2 Kafka Redundancy And Robustness

Due to the retention requirements for logs in Kafka, it was necessary to configure the storage to be more robust and available than the standard setup in Strimzi. The team adjusted the Kafka cluster configuration to reduce the risk of data loss in case of failure. It increased the replication factor of topics to maintain data integrity if a Kubernetes pod fails. The full configuration can be read in section 4.8.4, "Configuring Data-replication."

Chapter 4

Implementation & Challenges

4.1 Environment

This project is deployed on NTNU's OpenStack implementation called "SkyHiGh", an infrastructure-as-a-service (IaaS) platform. This choice closely mirrors the SOCSTACK environment the NTNU-SOC is currently developing, ensuring that our implementation aligns with their future infrastructure. A GitOps workflow is implemented using NTNU's git.gvk.idi.ntnu.no GitLab instance, establishing a single source of truth for infrastructure and configuration management.

4.2 Tooling

This project leverages a combination of IaC and GitOps to streamline and automate infrastructure provisioning, system configuration, and application deployment. Terraform and Ansible are tools that we use to collectively offer a robust solution for configuring infrastructure and system-level settings. Terraform defines the infrastructure components such as routers, networks, and servers, while Ansible specializes in managing system-level configurations, encompassing tasks like software installation and further system setup after Terraform has made basic settings. FluxCD, on the other hand, is employed to automate the deployment and synchronization of Kubernetes resources, ensuring that the desired state defined in Git is consistently maintained in the cluster.

4.2.1 Terraform

Terraform serves as one of the two primary IaC tools due to, but not limited to the following reasons:

- **Provider Ecosystem:** Extensive support for cloud providers [1] (like SkyHiGh's OpenStack), version control systems (GitHub, Gitlab, ...), and Kubernetes simplifies resource declaration and management within our mixed environment.
- **Ansible Integration:** The Ansible provider's [2] ability to dynamically generate Ansible inventories fosters seamless interaction between infrastructure provisioning and configuration steps.
- **GitOps Alignment:** Remote state storage in GitLab reinforces our GitOps practice, maintaining a comprehensive audit trail of changes.

[1] https://registry.terraform.io/browse/providers
[2] https://registry.terraform.io/providers/ansible/ansible/latest/docs

- **Documentation:** The Terraform Registry's [3] detailed documentation aids in rapid learning and troubleshooting.

4.2.2 Ansible

Ansible provides agentless configuration management for virtual machines outside of the Kubernetes clusters. Its key benefits include [16]:

- **Simplicity:** Ansible's YAML-based playbooks are human-readable, promoting maintainability.
- **Flexibility:** Ansible can manage various configuration tasks, from software installation and package updates to system-level settings.
- **Idempotency:** Ansible's declarative language makes it easy to write playbooks that ensure systems converge to the desired state, regardless of their starting point. This minimizes unexpected changes and simplifies re-runs of playbooks.

4.2.3 FluxCD

FluxCD enables a GitOps-based continuous delivery approach within Kubernetes clusters. We adopted FluxCD for:

- **GitOps Integration:** FluxCD aligns with our Git-centric methodology, treating Git repositories as the source of truth for cluster configurations and deployments.
- **Terraform Compatibility:** Easy bootstrapping of FluxCD using Terraform ensures streamlined cluster initialization.
- **Automation:** FluxCD continuously reconciles desired states from Git with the live cluster, automating updates and reducing drift.

4.3 Automated Infrastructure Management With IaC

To automate and streamline infrastructure management for our log analytics pipeline, we have adopted a GitOps-based approach. This methodology, which prioritizes Git as the single source of truth for infrastructure configuration, has guided the development of our automated system for managing IaC. While this section focuses on the Provisioning and, later, the Orchestration of OpenSearch nodes as a representative example, it is important to note that the entire log analytics pipeline, including networking, storage, and other components, is also managed using IaC principles. This aids in consistency, reproducibility, and adherence to the project's requirements.

4.3.1 Provisioning

The log analytics pipeline relies on a cluster of OpenSearch nodes to store and index log data for efficient search and analysis by the NTNU SOC. To automate the Provisioning of these nodes, we have utilized Terraform, as shown in the following code example (4.1); it showcases the declarative configuration used to provision an OpenSearch dashboard node within our OpenStack environment. It is written in a declarative style, which means that the code defines the desired state of the instance without specifying the exact steps to create it. This Terraform block configures the initial features of an OpenStack node that is used to host the OpenSearch dashboard. The following

[3] https://registry.terraform.io

figure and table provide an overview of the parameters used and their significance in Provisioning the OpenSearch nodes.

```
resource "openstack_compute_instance_v2" "dashboard_node" {
  count           = var.dashboard_node_count
  name            = "opensearch-dashboard-${count.index}"
  image_name      = "CentOS 7.6 x86_64"
  flavor_name     = var.dashboard_flavor_name
  key_pair        = var.keypair_name
  security_groups = ["default", openstack_compute_secgroup_v2.sc_os_dashboard.name,
                     openstack_compute_secgroup_v2.sc_os_1.name]

  network { uuid = openstack_networking_network_v2.network.id }
  depends_on = [openstack_networking_subnet_v2.subnet]
}
```

Parameter	Description
resource type	openstack_compute_instance_v2 is used to specify that we want to create a virtual machine instance in OpenStack.
resource name	dashboard_node is used as the name for this resource.
count	Dynamically creates multiple instances based on a variable.
name	Provides a unique name for each instance.
image_name	Specifies the OS image to use.
flavor_name	Determines the CPU and RAM resources for the instance.
security_groups	Attaches wanted security groups to control network traffic.
network	Specifies the OpenStack network to which the instance is connected.
depends_on	Ensures the subnet exists before creating the instance.

Figure 4.1: Terraform - Provisioning an OpenSearch node

4.3.2 Orchestration

With the instances provisioned by Terraform, the next step in creating and managing instances is to configure these into fully functioning OpenSearch nodes. The following YAML code block in figure 4.2 is the Ansible task [1] where Ansible is used to configure an OpenStack instance once it has been provisioned with Terraform. These tasks automate the installation and setup of Opensearch, it is important to note that this is not all the Ansible code, it is just a snippet that is used to provide the general image of how Orchestration is done on a technological level. To clarify, the code in this example is from the official OpenSearch Ansible playbook and is not code written by the team. It is demonstrated here as it provides a good insight into how the OpenSearch nodes are configured.

[1] https://github.com/opensearch-project/ansible-playbook/blob/main/roles/linux/opensearch/tasks/opensearch.yml

```yaml
- name: OpenSearch Install | Download opensearch {{ os_version }}
  ansible.builtin.get_url:
    url: "{{ os_download_url }}/{{ os_version }}/opensearch-{{ os_version }}-linux-x64.tar.gz"
    dest: "/tmp/opensearch.tar.gz"
  register: download

- name: OpenSearch Install | Create opensearch user
  ansible.builtin.user:
    name: "{{ os_user }}"
    state: present
    shell: /bin/false
    create_home: true
    home: "{{ os_home }}"
  when: download.changed or iac_enable

- name: OpenSearch Install | Create home directory
  ansible.builtin.file:
    path: "{{ os_home }}"
    state: directory
    owner: "{{ os_user }}"
    group: "{{ os_user }}"
  when: download.changed or iac_enable
```

Task	Purpose and Key Parameters
Task 1: Download OpenSearch	Idempotently downloads the specified OpenSearch version if needed. **get_url**: Ansible module for file downloads. **url**: URL for the OpenSearch archive. **dest**: Destination path on the target system.
Task 2: Create OpenSearch User	Creates a dedicated system user (non-interactive) for running OpenSearch. **user**: Ansible module for user management. **name**: Specifies the username. **state: present**: Ensures the user exists. **shell: /bin/false**: Disables shell access for security. **create_home: true**: Creates a home directory.
Task 3: Create Home Directory	Creates the home directory for the OpenSearch user with correct ownership and permissions. **file**: Ansible module for file/directory operations. **path**: Specifies the directory path. **state: directory**: Ensures it's a directory. **owner**: Owner of the directory (OpenSearch user). **group**: Group ownership for the directory.

Figure 4.2: Ansible - Configuring OpenSearch nodes with Ansible

4.3.3 Bridging Terraform and Ansible

These tools can also facilitate the creation of dynamic inventories, ensuring that Ansible targets the correct machines. This synchronization is achieved by using the Terraform output data (IP addresses, hostnames, etc.) as input for Ansible's inventory file.

The cloud.terraform plugin for Ansible

The cloud.terraform [2] Ansible plugin automates this process by pulling the information straight from the statefiles created by Terraform. We include it in the hosts.yaml inventory file as shown in figure 4.3.

[2]https://github.com/ansible-collections/cloud.terraform

```yaml
---
plugin: cloud.terraform.terraform_provider
```

Figure 4.3: Ansible - Configuring dynamic inventories with the cloud.terraform

Ansible provider for Terraform

The Ansible provider for Terraform, as shown in figure 4.4, is a vital component for integrating Terraforms infrastructure Provisioning capabilities with Ansible configuration management. By defining the Ansible provider within the terraform configuration, we create a connection that allows direct interaction between these tools. This enables us to use outputs from terraform (like IP addresses of instances that are created) as input variables for Ansible playbooks, ensuring a very streamlined configuration of the provisioned infrastructure. [1]

```
terraform {
  required_providers {
    ansible = {
      source = "ansible/ansible"
      version = "1.2.0"
    }
  }
}
```

Figure 4.4: Terraform - Including the ansible provider in Terraform configuration

In short, these two components work together in the following way.

- Terraform (With the ansible provider):

"Hey, Ansible, here is the infrastructure I just created. Please configure it."

- Ansible (With the cloud.terraform plugin):

"Thank you, Terraform! Let me check your state file to see what I need to work on. "

[1] https://registry.terraform.io/providers/ansible/ansible/latest/docs

4.4 The Deployment Pipeline

Deployment pipelines are commonly employed to automate the deployment of infrastructure as code to the cloud. These pipelines consist of predefined jobs that can be implemented within various CI/CD tools. While GitHub, Jenkins, and other platforms offer similar capabilities, for our project, we specifically chose GitLab as our CI/CD platform, and all of our pipelines are defined within GitLab and executed by Gitlab runners. We structured our deployment pipeline in the following way.

4.4.1 Pipeline Structure

The deployment pipeline for our log analytics solution is divided into five stages: initialize, plan, deploy, populate, and optionally destroy. These stages are further divided into jobs, which are instructions given to the runner to execute. Completion of one stage will trigger the execution of the next. Meanwhile, the destroyed stage is manually executed when needed. The design of our deployment pipeline has been crafted to "only include stages that add value," in line with Kief Morris's recommendation to avoid unnecessary overhead in the delivery process [20, p. 112].

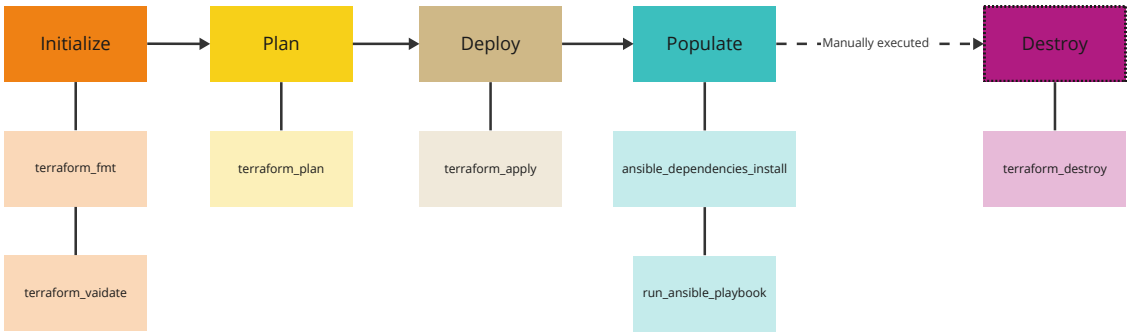


Figure 4.5: Deployment pipeline in Gitlab

- **Initialize:** Validate and format Terraform code.
- **Plan:** Creates an execution plan,
- **Deploy:** Executes the generated plan
- **Populate:** Configure the instances with ansible
- *Destroy*: Safely deprovision all resources created by terraform

4.4.2 Terraform Pipeline

The following section will break down what is happening on the Gitlab Runners to give a better understanding of how the Terraform code transforms into actual virtual machines running in the cloud. As demonstrated in figure 4.6, we can automate an otherwise manual job, which is then executed by the CI/CD operator, in this case, the GitLab runner. Without automated jobs like this, you would have to type in the terminal manually; now it runs as a job in the pipeline, which would then run when the previous job succeeded.

```yaml
terraform_apply:
  stage: Deploy
  image:
    name: hashicorp/terraform:latest
  dependencies:
    - terraform_plan
  script:
    - terraform apply -auto-approve tfplan
  when: on_success
```

Figure 4.6: Terreform plan as a CI/CD job in YAML

Gitops in practice

The terraform_plan job, shown in figure 4.7, highlights the benefits of GitOps in practice. Triggered by a commit to the Git repository, this job automatically executes the Terraform plan, making the desired infrastructure changes a reality in the OpenStack environment.

In this specific example, the output of the terraform_plan command indicates the successful creation of a security group rule, directly corresponding to a change made in the Git commit *"Added security rule to allow port 9100..."*. This demonstrates the core principle of GitOps: using Git as the single source of truth for infrastructure definitions. [26, p. 26] Any modifications to the configuration files in GitLab trigger the pipeline, ensuring that the live infrastructure always aligns with the desired state defined in the repository.

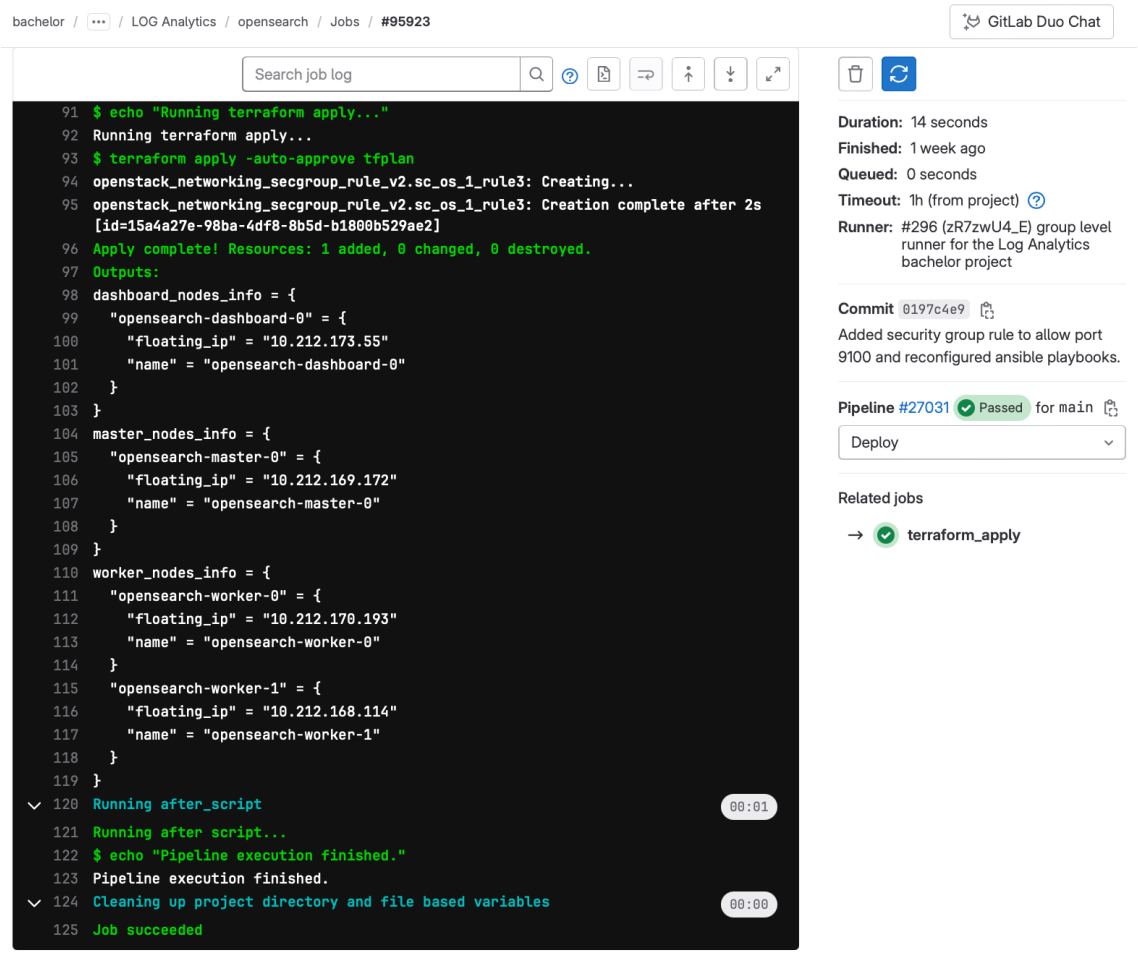


Figure 4.7: GitLab Runner Output from terraform_apply CI/CD job in figure 4.6

This automation streamlines the deployment process and enhances reliability and traceability. By relying on Git's version control system, we can easily track changes, roll back to previous configurations if needed, and collaborate effectively on infrastructure updates. The terraform_apply job, therefore, exemplifies how GitOps principles can be effectively implemented to manage and deploy cloud infrastructure.

4.4.3 Ansible Pipeline

Once the OpenSearch master and worker nodes are created, they are empty machines that need to be configured with the required software. This includes downloading and installing OpenSearch and configuring it thereafter. The code in figure 4.8 contains the most important parts of the job that runs the ansible playbook to configure the OpenSearch nodes automatically from the GitLab runners. This consists of executing the playbook to configure the hosts defined in the hosts.yaml file. This is the same file as mentioned in 4.3. Further explanation of configuring OpenSearch nodes are provided in section 4.5.

```yaml
run_ansible_playbook:
  stage: Populate
  script:
    - >
      ANSIBLE_SCP_IF_SSH=True
      ansible-playbook
      -i ../ansible/inventories/opensearch/hosts.yaml
      ../ansible/main.yaml
      -u centos --become
      --private-key=/home/gitlab-runner/.ssh/opensearch.pem
      --extra-vars "admin_password=$OPENSEARCH_ADMIN_PASSWORD"
      --extra-vars "kibanaserver_password=$OPENSEARCH_KIBANA_PASSWORD"
      --extra-vars "logstash_password=$OPENSEARCH_LOGSTASH_PASSWORD"
  artifacts:
    paths:
      - ansible/
  cache:
    paths:
      - ~/.ansible/
  dependencies:
    - ansible_dependencies_install
    - terraform_apply
```

Figure 4.8: Running ansible playbooks as a CI/CD job in YAML

While this YAML code is responsible for running, the code in 4.2 is an example of the actual changes being made. Everything that is configured here is stored and managed in the OpenSearch repository in GitLab, ensuring that a single source of truth is used. i.e., having all configurations in one place.

4.4.4 Cross Repository Integration

To facilitate the reusability of the CI/CD pipelines, the YAML code was worked on in its own repository, "CICD-Log-analytics". Relevant pipelines that might need them are then included in the repository. The following YAML code in 4.9 is the entire content of the OpenSearch repositories .gitlab-ci.yml file; this demonstrates how smoothly the jobs are integrated where needed.

```yaml
# Includes the terraform deployemt job from the CICD repository
include:
  - project: 'bachelor/2024/log-analytics/cicd-log-analytics'
    ref: main
    file: 'deployment_pipeline/deploy_terraform/.terraform_deployment_pipeline.yml'

# Includes the ansible deployemt job from the CICD repository.
  - project: 'bachelor/2024/log-analytics/cicd-log-analytics'
    ref: main
    file: 'deployment_pipeline/deploy_ansible/.ansible_deployment_pipeline.yml'
```

Figure 4.9: Including jobs from CI/CD repository

There are a few advantages to separating them into their own repository. Some of the advantages are:

- **Modularity:** Reduces the need for multiple identical pieces of code across the repositories
- **Scalability:** CI/CD pipelines are efficiently integrated as needed
- **Clear versioning:** Separate version control and a secluded environment support troubleshooting or rollbacks if needed.

4.4.5 Proof of Concept

There are a couple of reasons the deployment pipelines were not integrated across all the infrastructure. While this would mean just a few clicks would deploy and configure both the OpenSarch nodes and all the other infrastructure stored in the other repositories as well, it would also take considerable amounts of time to implement, as some of the Terraform code would need to be tailored to take Gitlab secrets as environment variables and not locally stored variables in .tfvar files and such.

Although the project initially aimed to automate deployment for all infrastructure components, time constraints and the complexity of adapting existing Terraform code to handle GitLab secrets securely led us to prioritize implementing deployment pipelines only for the OpenSearch nodes. This focused approach allowed us to demonstrate the core principles and benefits of automated deployment while establishing a working proof-of-concept for future expansion of the remaining infrastructure.

4.5 Log Indexing And Storage With OpenSearch

In this section, we detail the implementation and challenges encountered while integrating OpenSearch (as introduced in 2.3.5) into our infrastructure stack. Due to OpenSearch's stateful nature and persistence requirements, as well as not being within the project's scope, we opted to deploy it directly on virtual machines rather than within a containerized Kubernetes environment.

4.5.1 Deploying OpenSearch Via Terraform

Terraform was employed to automate the provisioning of the OpenSearch cluster within SkyHigh. To maintain a clear separation of concerns and enhance security, a dedicated network was established for the OpenSearch nodes. This network segmentation ensures the OpenSearch cluster's communication remains isolated from other components of the infrastructure.

Two distinct security groups were created to govern traffic flow: one for worker and master nodes, permitting ingress traffic only on essential ports (22 for SSH, 9100 for Prometheus scraping, and 9200 for HTTP/HTTPS API traffic), and another for dashboard nodes, allowing ingress on ports 22 and 5601 (for the dashboard interface). Internal cluster communication occurs over port 9300, which did not require explicit exposure due to the nodes residing within the same isolated network.

```
...
The OpenSearch role should be seen as input for your own playbook. I've had to remove a lot due to
the way we do things, which wouldn't give you any more understanding or meaning.
Mostly, it's installing packages that are needed before Ansible copies config files to the correct
locations, local firewall and system files to the host. This also depends a bit on which
distribution you use, etc.
...
The moment you start fiddling with TLS and OpenSearch, the difficulty level increases
significantly.
```

Figure 4.10: Translated excerpts from stakeholder communication

During the project's planning phase, it was agreed that pre-configured scripts would be provided to streamline the OpenSearch cluster deployment. However, as demonstrated in Figure 4.10, the scripts shared by the stakeholders would require additional work before they could be implemented. We spent a substantial amount of time trying to adapt the provided scripts but ultimately opted to instead adapt the official OpenSearch Ansible playbooks directly. This decision allowed us greater control and customization of the deployment process. Modifications were made to the playbooks to align with the project's specific requirements. Consult Appendix B for a comparison between the original and modified playbook.

4.5.2 Creating and Distributing Certificates

A core requirement from the stakeholders was that all traffic within the infrastructure be encrypted using TLS. This necessitated the creation and distribution of certificates trusted by all components of the infrastructure stack. Terraform was used to facilitate the retrieval of the root CA's PEM file and signing key from the OpenStack keymanager[4]. These resources were then used to generate and distribute certificates to the OpenSearch cluster.

```
1 data "openstack_keymanager_secret_v1" "ca_cert_pem" {
2   name = "bastion-ca-cert-pem"
3 }
4
5 data "openstack_keymanager_secret_v1" "ca_cert_signing_key" {
6   name = "bastion-ca-cert-signing-key-pem-pkcs8"
7 }
```

Figure 4.11: Importing certificates stored in OpenStack

[4]The Root CA is generated and provided to the key manager via the 'gitlab-infra' repository, which is used to deploy the baseline components necessary for automation (GitLab Runners, etc.)

```
resource "tls_private_key" "opensearch_master_http" {
  count     = length(openstack_compute_instance_v2.master_node)
  algorithm = "RSA"
}

resource "tls_cert_request" "opensearch_master_http" {
  count           = length(tls_private_key.opensearch_master_http)
  private_key_pem = tls_private_key.opensearch_master_http[count.index].private_key_pem
  dns_names = ["${openstack_compute_instance_v2.master_node[count.index].name}.bastion.com"]
  subject {
    country             = "NO"
    common_name         =
    ↪  "${openstack_compute_instance_v2.master_node[count.index].name}.bastion.com"
    organizational_unit = "CA"
    organization        = "bastion.com, Inc."
  }
}

resource "tls_locally_signed_cert" "opensearch_master_http" {
  count                 = length(tls_private_key.opensearch_master_http)
  cert_request_pem      = tls_cert_request.opensearch_master_http[count.index].cert_request_pem
  ca_private_key_pem    = data.openstack_keymanager_secret_v1.ca_cert_signing_key.payload
  ca_cert_pem           = data.openstack_keymanager_secret_v1.ca_cert_pem.payload
  validity_period_hours = 43800
  allowed_uses = [
    "digital_signature",
    "key_encipherment",
    "server_auth",
    "client_auth",
  ]
}
```

Figure 4.12: Creating HTTP/HTTPS certificates for OpenSearch Master nodes

Figures 4.11 and 4.12 illustrate the Terraform configuration for importing the root CA certificate and signing key from OpenStack and creating HTTP/HTTPS certificates for the OpenSearch master nodes. To streamline the distribution process, the raw certificate data was directly embedded into the dynamically generated Ansible inventories. Considering the OpenSearch cluster was not initially within the project's scope, this approach was deemed a suitable compromise for the proof-of-concept nature of this project.

Figure 4.13 demonstrates how Ansible hosts were defined within Terraform, incorporating the necessary certificate information as inventory variables. These variables were then utilized by the modified Ansible playbooks to configure the OpenSearch nodes with the appropriate certificates, allowing infrastructure-wide TLS.

```
resource "ansible_host" "os_master_node" {
  count  = length(openstack_compute_instance_v2.master_node)
  name   = openstack_compute_instance_v2.master_node[count.index].name
  groups = ["os-cluster", "master"]
  variables = {
    ansible_host = openstack_networking_floatingip_v2.master_fip[count.index].address
    ansible_user = "root"
    ip           = openstack_compute_instance_v2.master_node[count.index].network.0.fixed_ip_v4
    roles        = "data,master"

    root_ca_pem       = data.openstack_keymanager_secret_v1.ca_cert_pem.payload
    root_ca_key       = data.openstack_keymanager_secret_v1.ca_cert_signing_key.payload
    internal_cert_key =
    ↪  tls_private_key.opensearch_master_internal[count.index].private_key_pem_pkcs8
    internal_cert_pem = tls_locally_signed_cert.opensearch_master_internal[count.index].cert_pem
    http_cert_key     = tls_private_key.opensearch_master_http[count.index].private_key_pem_pkcs8
    http_cert_pem     = tls_locally_signed_cert.opensearch_master_http[count.index].cert_pem
    admin_cert_key    = tls_private_key.opensearch_admin.private_key_pem_pkcs8
    admin_cert_pem    = tls_locally_signed_cert.opensearch_admin.cert_pem

    additional_ssh_keys = var.opensearch_authorized_key
  }
}
```

Figure 4.13: "ansible_host" resource for OpenSearch master nodes

4.5.3 Ansible Playbooks

After provisioning the OpenSearch nodes with Terraform, Ansible was employed to automate their configuration. Figure 4.14 depicts the main playbook, which orchestrates the execution of playbooks to allow root SSH access, install and configure OpenSearch, and installs both the Prometheus exporter plugin[5] and node-exporter[6].

A notable modification made to the Ansible playbooks was the removal of the automatic certificate generation functionality. Instead, the playbooks were adapted to utilize the certificates created and distributed by Terraform, as detailed in the previous subsection.

In addition to the modifications related to certificate management, adjustments were made to the Ansible inventory to dynamically incorporate the IP addresses and hostnames of the provisioned OpenSearch nodes as explained in 4.3.3.

```
---
- name: Allow root ssh
  import_playbook: allow_root_ssh.yaml

- name: Install and configure Opensearch
  import_playbook: opensearch.yaml

- name: Install prometheus exporter plugin
  import_playbook: prometheus_exporters.yaml
```

Figure 4.14: Opensearch main playbook

[5] https://github.com/Aiven-Open/prometheus-exporter-plugin-for-opensearch
[6] https://github.com/prometheus/node_exporter

4.5.4 Deploying OpenSearch Via GitLab Pipelines

Figure 4.9 illustrates our OpenSearch repository's GitLab CI file, which triggers the deployment pipelines outlined in section 4.4, aligning with the GitOps methodology we decided on.

4.6 Kubernetes Deployment Strategy

The decision to utilize Kubernetes as the foundation for our logging infrastructure was driven by the advantages it offers in terms of scalability, containerization benefits, and orchestration capabilities. Key considerations behind this choice include:

Scalability: Kubernetes is well known for its inherent ability to manage resources dynamically. As the volume of logs generated by our infrastructure increases, Kubernetes enables us to scale our computing resources seamlessly to accommodate the rising demand[4].

Containerization: A logging system naturally lends itself to modular components like data collection agents, specialized parsers and processors, and robust storage solutions. Containerization provides a good foundation for packaging and deploying these elements independently, simplifying management and promoting streamlined updates.

Orchestration: Kubernetes excels in managing complex workloads, automating tasks like the scheduling of containers, monitoring their health, and executing seamless updates with minimal disruption. In the context of our implementation, this translates to increased reliability and reduced administrative overhead[4].

Overall, Kubernetes was the logical choice for this project due to its alignment with the anticipated needs for scalability, streamlined deployment, and efficient management of our logging infrastructure.

4.6.1 Deployment Choices

When considering Kubernetes deployment, we evaluated two primary approaches: a self-deployed cluster and the OpenStack template. By consulting 'Kubernetes Up and Running'[4] and examining the Kubernetes Templates available in Skyhigh, we created a simple pros and cons list to narrow down the decision:

Self-Deployed Cluster
Pros: Offers greater flexibility in configuring the cluster's components and tailoring it specifically to our needs. Provides granular control of customization options.
Cons: Requires a significantly higher level of in-house Kubernetes expertise. Increases complexity related to the initial setup and ongoing management of the cluster.

OpenStack Template
Pros: Provides a simplified deployment path and reduces the administrative overhead compared to a self-deployed approach. May include pre-configured integrations with the existing OpenStack environment.
Cons: Limits customization possibilities. New features and updates to the Kubernetes cluster may be dependent on updates to the OpenStack template itself.

Ultimately, we opted for the OpenStack template approach. The appeal of a faster deployment process and the potential synergy with our existing OpenStack environment outweighed the limitations in customization. While some adjustments were required to address specific component issues, as we'll discuss later, this approach aligned well with our project's timeline and available resources.

```
module "k8s_bastion" {
  source = "./modules/k8s_with_fluxcd"

  k8s_cluster_name = "bastion"
  k8s_cluster_master_count = 1
  k8s_cluster_node_count = 3
  keypair = data.openstack_compute_keypair_v2.jumpbox.name
  k8s_cluster_delete_csi_cinder_controllerplugin = true
  flux_bootstrap_repo = var.flux_bastion_bootstrap_repo
  flux_bootstrap_folder_path = "flux/bastion/bootstrap"
  gitlab_repository_ssh_url =
  ↪  "ssh://git@${var.gitlab_host}/${var.gitlab_group}/${var.flux_bastion_bootstrap_repo}.git"
  gitlab_repository_path = "${var.gitlab_group}/${var.flux_bastion_bootstrap_repo}"
}
```

Figure 4.15: Kubernetes - Deploying Kubernetes cluster via the module block

4.6.2 Deploying With Terraform

To avoid the monolithic stack antipattern, as described by Kief Morris[20, Patterns and Anti-patterns for Structuring Stacks], we adopted a modular and logically organized approach within Terraform. Frequently used blocks of code were encapsulated into distinct modules, simplifying the configuration of multiple clusters (e.g., Bastion and Citadel). Additionally, we grouped Terraform resources based on their function, improving the readability and structure of our infrastructure configuration.

Figure 4.15 demonstrates the use of a Terraform module to provision our 'Bastion' Kubernetes cluster. This module defines the cluster's core configuration in OpenStack. It specifies the cluster name, the OpenStack template to use, the number of master and worker nodes, their flavors (compute resources), and any necessary network settings. This module also handles the bootstrapping of FluxCD for configuration management and the removal of the CSI Cinder plugin.

Deployment Challenges

Despite the benefits of the OpenStack template, we faced several challenges that required resolution:

Mismatched Drivers:
The CSI Cinder plugin version bundled within the OpenStack template was incompatible, resulting in issues with attaching provisioned volumes to pods. We had to troubleshoot the specific component issue (the CSI-Attacher version) and either consider a patch if possible or update the images used by the CSI Cinder Controllerplugin

Non-Functional Scaling:
While we were unable to directly deploy a functional auto-scaling cluster from the "kubernetes-v1.23.16-ha" OpenStack template, we successfully managed to tailor the cluster labels during our Terraform deployment to enable node auto-scaling and self-healing for the 'non-ha' template (see Figure 4.16). We did not encounter any issues with clusters deployed with this configuration, but for our POC we ultimately opted for the default, non-auto-scaling version.

Addressing Challenges:
Resolving these issues often involved in-depth analysis and the implementation of customized solutions. For instance, we had to delete the default CSI Cinder statefulset(see Figure 4.17) and redeploy it using FluxCD with the updated container images.

```
resource "openstack_containerinfra_cluster_v1" "bastion" {
  ## Non-relevant configuration excluded from example ##

  merge_labels = true # Merges the custom labels with the ones provided by the cluster template
  labels = {
    auto_scaling_enabled: true
    auto_healing_enabled: true
    min_node_count: 2
    max_node_count: 5
  }
}
```

Figure 4.16: Kubernetes - Merging labels to enable auto-scaling

```
resource "null_resource" "delete_statefulset" {
  count = var.k8s_cluster_delete_csi_cinder_controllerplugin? 1 : 0
  depends_on = [openstack_containerinfra_cluster_v1.k8s]
  triggers = {
    cluster_id = openstack_containerinfra_cluster_v1.k8s.id
  }

  provisioner "local-exec" {
    command     = #<<EOT
      TMP_KUBECONFIG=$(mktemp)
      echo "${openstack_containerinfra_cluster_v1.k8s.kubeconfig.raw_config}" > $TMP_KUBECONFIG
      kubectl --kubeconfig=$TMP_KUBECONFIG delete statefulset csi-cinder-controllerplugin
      ↪  --namespace kube-system
      rm -f $TMP_KUBECONFIG
    EOT
    interpreter = ["/bin/sh", "-c"]
  }
}
```

Figure 4.17: Kubernetes - Deleting the CSI Cinder Controllerplugin via Terraform

4.6.3 Provisioning Additional Credentials To FluxCD

Our infrastructure components are distributed across multiple GitLab repositories for better organization. FluxCD requires additional credentials to access these repositories, which are created and provided as Kubernetes secrets through the 'flux_git_repo_credentials' module - Figure 4.18 shows how this is used when provisioning the 'Bastion' cluster.

```
module "flux_git_repo_credentials_bastion" {
  source   = "./modules/flux_git_repo_credentials"
  for_each = { for repo in var.flux_bastion_additional_repos : repo => repo }
  providers = {
    kubernetes = kubernetes.bastion
  }

  repository_path = "${var.gitlab_group}/${each.key}"
  gitlab_host_key = var.gitlab_host_key

  depends_on = [ module.k8s_bastion ]
}
```

Figure 4.18: Kubernetes - Providing credentials to FluxCD for additional repositories

4.7 Gitops And Continuous Delivery In Kubernetes

Our infrastructure Provisioning process leverages Terraform to orchestrate the initial deployment of FluxCD within our Kubernetes clusters. This establishes the foundation for our GitOps workflow and continuous delivery pipeline. The process outline is summarized in the following steps:

FluxCD Provider
Our Terraform configuration includes a dedicated FluxCD provider, configured to interact with our Kubernetes clusters. The Git integration within the flux provider specifies the SSH credentials FluxCD will use to authenticate with our GitLab instance.

SSH Key Generation
To ensure secure Git access, we use Terraform to generate SSH key pairs for FluxCD. These keys are subsequently added as deploy keys to our GitLab projects.

FluxCD Bootstrap
Flux's bootstrap Terraform resource deploys the FluxCD components into the Kubernetes cluster and configures it to synchronize with our bootstrap Git repository.

Repository Secrets
We create Kubernetes Secrets containing the FluxCD SSH identities for our application repositories (Figure 4.18). These secrets are made available in the flux-system namespace and will be used by FluxCD to interact with the repositories.

4.7.1 Orchestration Strategy

Kubernetes Kustomizations (kustomization.yaml files) play a central role in how we group and deploy related resources. Kustomizations define logical collections of Kubernetes manifests, ensuring they are applied together in a consistent manner.

Our "cluster-configuration" directory contains the kustomization.yaml file presented in Figure 4.19, which defines the initial resources to be deployed as a logical collection within our cluster.

```yaml
---
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
resources:
  - namespace.yaml
  - kustomizations/cluster-roles.yaml
  - kustomizations/storage-classes.yaml
```

Resource	Description
namespace.yaml	Deploys the namespace for our cluster configuration resources.
kustomizations/cluster-roles.yaml	Contains the Kustomization for deploying cluster-wide RBAC roles.
kustomizations/storage-classes.yaml	Contains the Kustomization for Provisioning persistent storage classes.

Figure 4.19: Kubernetes Kustomization example

Kustomizations like this serve as a stepping stone for implementing dependency-driven workflows with FluxCD's custom resources, which we'll explore in the next subsection.

Dependency-Driven Deployments

In complex deployments involving multiple components, we leverage FluxCD's Kustomization CRDs[7] to orchestrate deployments based on explicit dependencies. This ensures that resources are deployed in the correct order, preventing failures and streamlining the overall process.

Why dependencies matter: If an application relies on the existence of specific namespaces, ConfigMaps, or other Kubernetes resources, deploying it prematurely can lead to errors. Dependency-driven deployments mitigate these issues.

FluxCD Kustomizations: Flux Kustomizations allow us to specify these dependencies, instructing FluxCD on the correct deployment sequence. Consider the Flux Kustomization file presented in Figure 4.20:

[7] https://fluxcd.io/flux/components/kustomize/kustomizations/

```yaml
---
apiVersion: kustomize.toolkit.fluxcd.io/v1
kind: Kustomization
metadata:
  name: storage-classes
  namespace: cluster-config
spec:
  interval: 10m
  path: "./bootstrap/cluster-config/storage-classes"
  prune: true
  dependsOn:
    - name: csi-cinder-controllerplugin-rbac
      namespace: kube-system
  sourceRef:
    kind: GitRepository
    name: flux-system
    namespace: flux-system
```

Parameter	Description
dependsOn	This section defines that deployment of resources in the ./bootstrap/cluster-config/storage-classes directory should occur only after the csi-cinder-controllerplugin-rbac resources in the kube-system namespace are successfully deployed.
interval	FluxCD will check for dependency status changes every 10 minutes.
prune	If the dependency is removed, FluxCD will also remove the resources defined within this Kustomization.

Figure 4.20: FluxCD Kustomization example

4.7.2 Repository Structuring

Careful repository organization is key to avoiding the monolithic antipattern [20] and maintaining a scalable and well-structured GitOps implementation. We approach this in the following manner:

Normalizing the repository structure

To manage resource deployment effectively, we adhere to a specific directory structure within our Git repositories. Each directory defines a logical group of Kubernetes manifests that FluxCD will synchronize:

```
[sync-dir]
├── namespace.yaml
├── kustomization.yaml
├── kustomize
│   ├── example-resource-dir-1.yaml
│   └── example-resource-dir-2.yaml
├── example-resource-dir-1
│   └── ...  Resource collection.
└── example-resource-dir-2
    └── ...  Resource collection.
```

Directory/File	Description
[sync-dir]	The root directory that FluxCD synchronizes.
namespace.yaml	Creates the namespace for all subsequent resources.
kustomization.yaml	The top-level Kubernetes Kustomization file, outlining which resources to initially deploy.
kustomize	Holds FluxCD Kustomizations for dependency-driven deployments within this directory.
example-resource-dir-1/2	Directories containing YAML manifests for specific applications or infrastructure components.

Figure 4.21: Repository structure example

Bootstrap repository

Our "bootstrap" repository contains the Kubernetes manifests initially deployed by FluxCD during the bootstrapping process. Figure 4.22 details the directory structure and a short summary of the manifests within.

```
bootstrap
├── cert-manager
├── cluster-config
├── flux-sources
├── flux-system
├── kube-system
└── nginx-ingress
```

Directory	Description
cert-manager	Manages deployment and configuration of cert-manager for TLS certificate issuance.
cluster-config	Deploys low-level resources like cluster roles and storage classes.
flux-sources	Declares other Git repositories and directories that FluxCD should subsequently synchronize (enabling nested GitOps workflows). These resources are only deployed when all other configuration from this repository are succesfully deployed.
flux-system	Holds FluxCD's own configuration, entirely managed by FluxCD itself.
kube-system	Contains necessary modifications and replacements to kube-system resources deployed by Sky-High Kubernetes templates.
nginx-ingress	Deploys and configures the nginx-ingress-controller.

Figure 4.22: Bootstrap Repository

Additional repositories

While our primary 'applications' repository houses the majority of our application deployments, certain applications have been separated into their own repositories to demonstrate the viability of managing components independently and showcase how this is achieved within our GitOps framework. For instance, the Citadel cluster's traffic simulation (detailed in 4.12) is deployed from its own dedicated repository.

In a production environment, we recommended adopting a multi-repository approach, where each

application is maintained in its own repository. This adoption can introduce several benefits, two of them being:

Reduced Blast Radius
Isolating applications into separate repositories limits the impact of a potential security breach or misconfiguration. If one repository is compromised, the others remain unaffected, enhancing the overall resilience of the system.[20, Pattern: Service Stack]

Granular Access Control
Managing permissions on a per-repository basis allows for finer-grained control over who can modify specific application configurations. This minimizes the risk of unauthorized changes and improves security.

4.7.3 FluxCD And Git Integration

Our GitOps implementation relies on secure and streamlined interaction between FluxCD and our Git repositories. In this section, we'll delve into authentication methods and the workflow of how changes propagate from Git to our Kubernetes cluster.

Authentication and Authorization

We employ SSH key-based authentication to ensure secure communication between FluxCD and our Git repositories. Here's how it works:

- **Dedicated SSH Keys:** We generate separate SSH key pairs for each repository that FluxCD needs to access.
- **GitLab Deploy Keys:** These SSH keys are added as project-level deploy keys in GitLab, granting both read and write access. Write access is required for FluxCD to update certain resources related to its own state tracking within the repository.
- **Flux Source Configuration:** Within each Flux source (GitRepository custom resource), we specify the repository URL, the branch to track, and reference the corresponding SSH key (identity) stored as a Kubernetes Secret.

Change Propagation Workflow

1. **Developer Change:** A developer modifies a YAML manifest (deployment, configuration, etc.) within a Git repository.
2. **Git Push:** The developer commits and pushes the change to the repository's tracked branch.
3. **FluxCD Detection:** FluxCD, according to its configured interval, detects the change in the Git repository.
4. **Manifest Retrieval:** FluxCD fetches the updated manifests from the Git repository.
5. **Kubernetes Reconciliation:** FluxCD compares the desired state (from Git) with the live state of the Kubernetes cluster. It then applies necessary changes (creations, updates, or deletions) to bring the cluster in line with the Git repository.

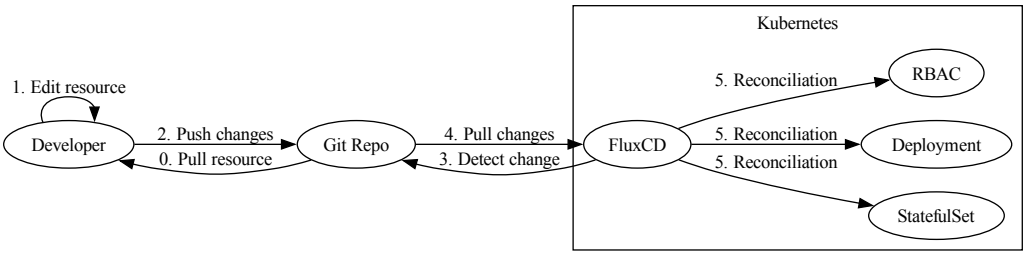


Figure 4.23: Change Propagation Workflow

4.8 Log Aggregation

This section covers the aggregation layer responsible for collecting, buffering, and routing high volumes of diverse log data from various sources. Apache Kafka provides a scalable and fault-tolerant message Broker, while Strimzi simplifies the deployment and management of Kafka within the Kubernetes environment.

4.8.1 GitOps Implementation

Our GitOps approach leverages FluxCD and a carefully designed folder structure to manage the lifecycle of Strimzi and its subsequent Kafka resources within our Kubernetes environment. The following outlines our approach:

Directory structure

A dedicated Strimzi directory within our primary application Git repository houses all Strimzi-related manifests and configuration.

```
strimzi
├── kafka-cluster
│   └── bastion
│       ├── kafka-cluster.yaml
│       ├── kafka-nodepool.yaml
│       └── metrics-config.yaml
├── kafka-topic
│   └── k6-traffic-sim.yaml
├── kafka-ui
│   ├── helm-deploy.yaml
│   └── ingress.yaml
├── kustomizations ...  Kustomization files for the fluxCD API. These are
│                       used to ensure that the resources are deployed in
│                       the correct order and only when it's dependencies
│                       are ready.
│   ├── kafka-cluster.yaml
│   ├── kafka-topic.yaml
│   ├── kafka-ui.yaml
│   ├── rbac.yaml
│   └── strimzi-operator.yaml
├── rbac
│   └── kafka-cluster-bastion-rbac.yaml
├── strimzi-operator
│   └── helm-deploy.yaml
├── helm-repo.yaml
├── kustomization.yaml
├── namespace.yaml
└── pod-monitor.yaml
```

Figure 4.24: Strimzi Directory structure

4.8.2 Deploying The Operator

We employed the Helm deployment approach outlined in the Strimzi documentation[8], adapting it for integration with our FluxCD-based GitOps workflow. The 'rbac.create:true' setting was enabled to grant the Strimzi operator the ability to manage necessary Role-Based Access Control (RBAC) resources. Finally, we enabled Strimzi's built-in Grafana dashboards ('dashboards.enabled:true' in Figure 4.25) for enhanced observability. These dashboards will be deployed to our Prometheus namespace, offering pre-configured visualizations of Kafka metrics.

```
1 strimzi/strimzi-operator/helm-deploy.yaml
2 ---
3 values:
4   replicas: 1
5   rbac:
6     create: true
7   dashboards:
8     enabled: true
9     namespace: prometheus
```

Figure 4.25: Select Strimzi operator settings

[8]https://github.com/strimzi/strimzi-kafka-operator/blob/main/helm-charts/helm3/strimzi-kafka-operator/README.md

4.8.3 Deploying The Kafka Cluster

In light of ZooKeeper's deprecation within Apache Kafka[9], we've adopted Kafka's Raft mode (KRaft) for cluster metadata management. Raft mode is well-suited for Kubernetes due to its reduced dependencies and alignment with operator-based management. With Strimzi version 0.40.0 and above, Raft is deployed by default when using Node Pools[10], further simplifying our configuration. We've assigned both controller and Broker roles to each node. This ensures that in the event of a node failure, the cluster maintains its quorum through the seamless re-election of a controller. Furthermore, persistent-claim volumes provide data persistence for each Kafka Broker. If a node or the entire cluster experiences issues, the data is preserved. Upon recovery, Kafka can re-synchronize from these persistent volumes.

```
strimzi/kafka-cluster/bastion/kafka-nodepool.yaml
---
spec:
  replicas: 3
  roles:
    - controller
    - broker
  storage:
    type: jbod
    volumes:
      - id: 0
        type: persistent-claim
        size: 20Gi
        class: strimzi-cluster-storage
        deleteClaim: false
```

Figure 4.26: Kafka Nodepool configuration

Distributing Brokers across multiple nodes in case of node-failure
We utilize the affinity rules as detailed in Figure 4.27 to guide the deployment of our Kafka cluster. The 'nodeAffinity' setting, with the 'requiredDuringSchedulingIgnoredDuringExecution' rule, ensures that Kafka pods are scheduled on Kubernetes nodes labelled 'magnum.openstack.org/nodegroup: Kafka-nodegroup'. These nodes are provisioned with resources suitable for Kafka workloads.

The 'podAntiAffinity' rule distributes Kafka Brokers across different physical nodes within the designated node group. This strategy enhances resilience by preventing a single node failure from disrupting the entire Kafka cluster. Additionally, the "kubernetes.io/hostname" topology key ensures that the anti-affinity rule considers individual physical nodes for optimal placement.

[9] https://kafka.apache.org/documentation/#zk_depr
[10] https://github.com/strimzi/strimzi-kafka-operator/releases/tag/0.40.0

```
strimzi/kafka-cluster/bastion/kafka-cluster.yaml
---
spec.kafka.template.pod.affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
        - matchExpressions:
          - key: magnum.openstack.org/nodegroup
            operator: In
            values:
            - kafka-nodegroup
  podAntiAffinity:
    preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 100
        podAffinityTerm:
          labelSelector:
            matchExpressions:
              - key: strimzi.io/cluster
                operator: In
                values:
                - bastion
          topologyKey: "kubernetes.io/hostname"
```

Figure 4.27: Configuring Kafka pod affinity rules

4.8.4 Configuring Data-Replication

In Apache Kafka, each topic is divided into partitions to enhance scalability and parallelism. Partition replication (visualized in Figure 4.28) is the process of maintaining multiple copies (replicas) of each partition across different brokers in the cluster. One broker acts as the leader for a partition, handling all reads and writes to that partition. The other brokers act as followers, replicating the data from the leader and maintaining an identical copy. This replication mechanism ensures data redundancy and high availability, as if the leader broker fails, one of the follower brokers can be elected as the new leader, allowing the topic to continue functioning without data loss.

Configuring data replication is crucial for maintaining resilience and high availability. As seen in Figure 4.29, we've configured several replication-related parameters to achieve this.

We decided to set the default replication factor and the minimum in-sync replicas to three to match the amount number of brokers in our cluster. With three replicas, we can tolerate the failure of up to two brokers without losing data. Additionally, by requiring acknowledgments from all in-sync replicas for each write, we guarantee that data is durably stored across multiple brokers, and it allows the configuration of a minimum amount of in-sync replicas. This instructs the Kafka brokers to only consider writes to a topic successful when it has been confirmed written by a quorum of replicas.

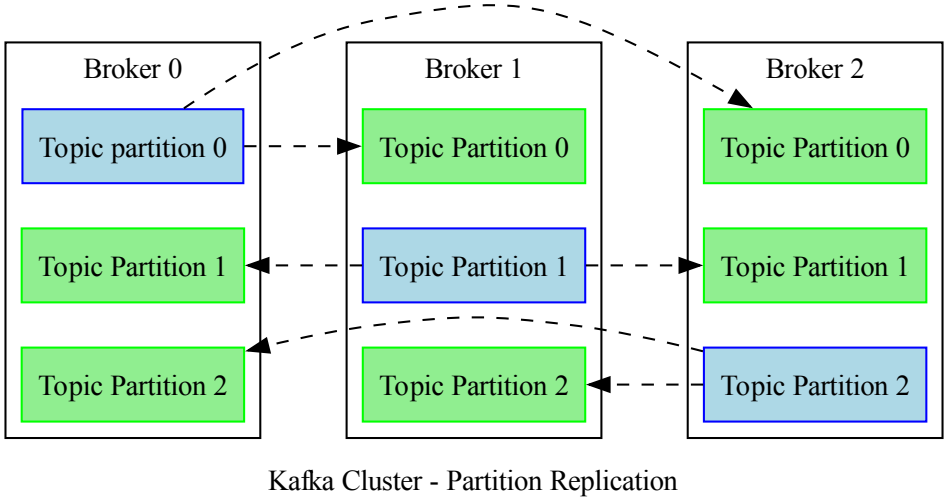


Figure 4.28: Kafka partition replication
Blue nodes represent Partition leaders, and green nodes represent partition followers - I.E. the replicated partitions.

```
strimzi/kafka-cluster/bastion/kafka-cluster.yaml
---
spec.kafka.config:
  acks: all
  offsets.topic.replication.factor: 3
  transaction.state.log.replication.factor: 3
  transaction.state.log.min.isr: 3
  default.replication.factor: 3
  min.insync.replicas: 3
  num.partitions: 10
```

Parameter	Description
acks	The 'all' value mandates that the leader Broker waits for acknowledgments from all in-sync replicas before confirming a write as successful.
offsets.topic.replication.factor	This setting controls the replication of the internal offsets Topic, which stores the progress of consumers within each topic Partition. A replication factor of 3 means that offset information is maintained on three distinct Brokers.
transaction.state.log.replication.factor	This parameter governs the replication of the transaction state log, another internal Topic crucial for managing the state of transactions, guaranteeing that transactions are either fully committed or aborted.
transaction.state.log.min.isr	This setting dictates the minimum number of replicas that must acknowledge a transaction-related write for it to be considered successful.
default.replication.factor	This establishes the default replication factor for automatically created topics within the cluster.
min.insync.replicas	In conjunction with the 'acks: all' setting, this parameter requires a minimum of three replicas to acknowledge a write before it's considered committed.
num.partitions	This defines the default number of Partitions for Topics with no specified partition count.

Figure 4.29: Kafka Cluster - Default replication configurations for Kafka Topics

4.8.5 Exposing Metrics

To enable monitoring of our Kafka cluster's health and performance, we've configured metrics export. Kafka exposes internal metrics in JMX (Java Management Extensions) format. The Strimzi operator facilitates the use of the Prometheus JMX Exporter[11] to bridge these metrics into the Prometheus ecosystem.

Our configuration instructs the Strimzi operator to deploy the JMX Exporter alongside each Kafka Broker. This exporter translates Kafka's JMX metrics into a format that Prometheus can understand and collect. We've adapted the example provided by Strimzi[12] to align with our specific monitoring requirements. Upon deployment, said configuration is fetched from the defined configMap.

```
strimzi/kafka-cluster/bastion/kafka-cluster.yaml
---
spec.kafka.metricsConfig:
  type: jmxPrometheusExporter
  valueFrom:
    configMapKeyRef:
      name: kafka-metrics
      key: kafka-metrics-config.yml
```

Figure 4.30: Configuring prometheus exporter

4.8.6 Deploying The Kafka Topic

To facilitate testing and validation of our Kafka infrastructure, we've created a Kafka Topic named 'k6', as described in Figure 4.31.

```
strimzi/kafka-topic/k6-traffic-sim.yaml
---
apiVersion: kafka.strimzi.io/v1beta1
kind: KafkaTopic
metadata:
  name: k6
  namespace: strimzi
  labels:
    strimzi.io/cluster: bastion
spec:
  partitions: 10
  replicas: 3
  config:
    retention.ms: 604800000
    segment.bytes: 1073741824
```

Parameter	Description
strimzi.io/cluster	This value of this label specifies which Kafka Cluster the topic should be assigned to.
retention.ms	This parameter controls how long a message is retained before cleanup. In this case, it will be retained for 604800000ms, ie 7 days.
segment.bytes	This controls the segment file size for the message. Larger file sizes will give us a lower file count but less fine-tuned retention control since retention and cleanup are done one file at a time.

Figure 4.31: Configuring the 'K6' Kafka Topic

[11] https://github.com/prometheus/jmx_exporter
[12] https://github.com/strimzi/strimzi-kafka-operator/blob/main/examples/metrics/kafka-metrics.yaml

4.8.7 Autoscaling

While the concept of autoscaling is attractive for adapting to dynamic workloads, implementing it effectively for Kafka clusters presents several challenges.

Horizontally scaling the Kafka cluster by adding brokers through a Horizontal Pod Autoscaler (HPA)[13] is technically possible. However, the benefits would be limited. Newly added brokers would only be able to handle partitions for newly created topics. Existing topics with pre-allocated partitions would remain on the original brokers, leading to an imbalanced cluster (See Figure 4.32).

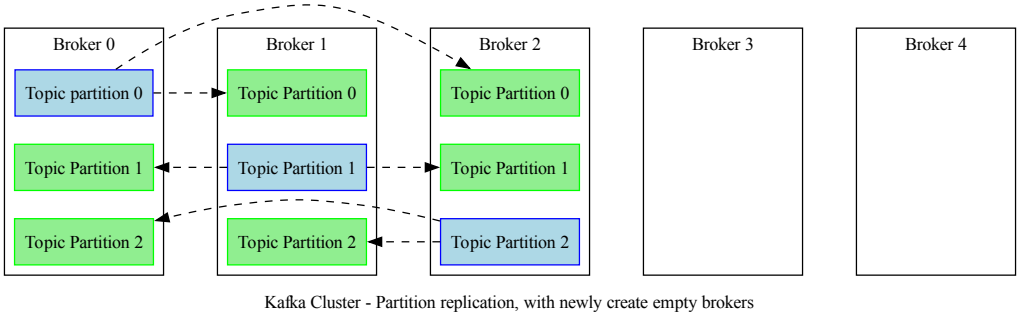


Figure 4.32: Kafka partition replication after cluster scaling

Addressing this imbalance would necessitate automatic cluster rebalancing, a process that involves migrating vast amounts of data across brokers, resulting in a surge of read/write operations and potentially impacting overall cluster performance[17]. Achieving true autoscaling would, therefore, require orchestrating two intertwined processes: broker scaling and cluster rebalancing.

Moreover, if the cluster scales reactively in response to a sudden load spike, initiating rebalancing during this period could further strain the system, hindering its ability to handle the increased traffic. Proactive scaling using machine learning algorithms to anticipate load fluctuations could be explored, but wouldn't account for unpredictable events like cyber attacks, which could significantly increase traffic to the NTNU SOC.

4.9 Log Processing

To enhance log messages with geodata, we elected to leverage Vector's built-in message transformation abilities using Vector Remap Language (VRL) and enrichment tables. This integration offers efficiency gains by minimizing data handoffs between separate external processing components. For this proof of concept, we've limited the scope of log processing to geodata enrichment and a basic suspicious origin check. Further processing, such as threat detection or anomaly identification, was deemed unnecessary for the scope of this project.

[13] https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/

```json
{
    "method": "PUT",
    "source_ipv4": "106.220.83.40",
    "source_mac": "84:62:98:f3:a4:a3",
    "timestamp": "2024-05-10T10:31:18.071Z",
    "url": "https://www.centralbleeding-edge.name/strategic/seamless/channels/utilize"
}
```

Figure 4.33: Log contents before processing

4.9.1 The GeoLite2 Geodata Database

For this POC, the GeoLite2 database is used to provide geodata enrichment. This freely available database offers a straightforward solution for demonstrating the concept of geodata augmentation. To simplify the POC setup, we utilize a version of the GeoLite2 database accessible directly from GitHub([14]). It's important to note that this specific version might have limitations in terms of accuracy or update frequency and introduces a security risk compared to the versions available from MaxMind([15]). In a production environment, integrating the most up-to-date Geodata source directly from MaxMind should be done to ensure data precision and security.

To optimize geodata retrieval performance, the GeoLite2 database is deployed within a RAMdisk. This technique leverages the significantly faster read/write speeds of RAM compared to traditional disk storage. In a high-volume log processing scenario, the RAMdisk approach minimizes latency during frequent geodata lookups. This configuration, as illustrated in Figure 4.34, involves using an initContainer to fetch and place the GeoLite2 database within a Kubernetes emptyDir volume backed by the in-memory storage medium. Then the GeoLite2 database is used to create a corresponding enrichment table.

```yaml
k8s-applications/clusters/bastion/vector/helm-deploy.yaml
---
values:
  initContainers:
      - name: fetch-geolite2-mmdb
        image: curlimages/curl:7.78.0
        command: ['sh', '-c', 'curl -L https://git.io/GeoLite2-City.mmdb -o
        ↪  /geolite2/GeoLite2-City.mmdb']
        volumeMounts:
        - name: geolite2
          mountPath: /geolite2
    extraVolumes:
      - name: geolite2
        emptyDir:
          medium: Memory
    extraVolumeMounts:
      - name: geolite2
        mountPath: /geolite2
  customConfig:
    enrichment_tables:
      geoip:
        locale: "en"
        path: "/geolite2/GeoLite2-City.mmdb"
        type: "geoip"
```

Figure 4.34: Geodata - GeoLite2 database configuration

[14]https://github.com/P3TERX/GeoLite.mmdb
[15]https://dev.maxmind.com/geoip/geolite2-free-geolocation-data

While RAMdisks typically face challenges related to data persistence, our implementation mitigates these concerns through the use of an initContainer. This container downloads the database before the main Vector container starts, guaranteeing [16] that even upon pod restarts, the database remains accessible, rendering the RAMdisk's volatility and lack of persistence a non-issue.

4.9.2 Vector Transformations

```
k8s-applications/clusters/bastion/vector/helm-deploy.yaml
---
values.customConfig.transforms.geodata:
  type: remap
  inputs:
    - kafka_cluster_bastion_k6
  drop_on_abort: false
  source: |-
    . = parse_json!(.message)
    .geodata = find_enrichment_table_records!("geoip", {"ip": .source_ipv4})
    .suspicious = includes(["RU", "CN"], .geodata[0].country_code)
```

Figure 4.35: Geodata - Transforming the logs

The geodata transformation logic within Vector is configured as a 'transform', as can be seen in Figure 4.35. The 'source' field contains an inline VRL script that is responsible for the message transformations. First, the contents of the 'message' field from the incoming event are extracted and set as the root datastructure[17](Figure 4.33). Second, the IP address (source_ipv4) is parsed from the message. Vector then queries the GeoLite2 enrichment table to associate geographical information with that IP address and appends this under the 'geodata' key. Finally, a rule-based evaluation determines if the log message should be flagged as suspicious based on the retrieved country code (e.g., "RU", "CN"). This process is visualized in Figure 4.36, and Figure 4.37 details the message contents after processing.

[16] Given that the GeoLite2 source is available

[17] This removes other metadata appended by the Kafka brokers. Should this metadata be required, simply extract the incoming event in its entirety.

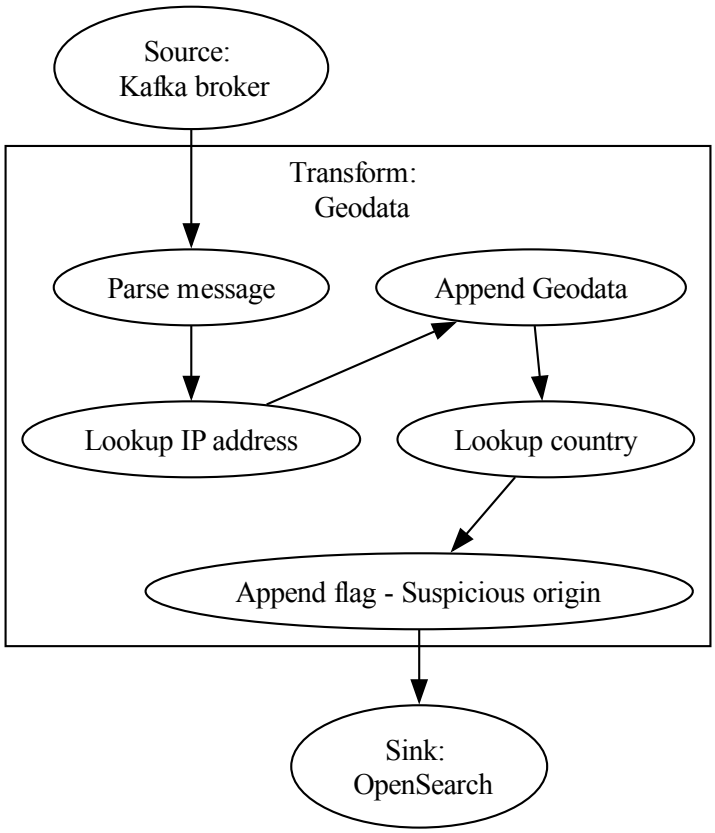


Figure 4.36: Vector - Geodata enrichment process
Actions taken by the 'Geodata' transformation

4.10 Log Routing Strategies

In complex data processing systems, it's sometimes necessary to route data between different components for specialized transformations, enrichments, or other processing steps that cannot be performed by a single tool or application. This section focuses on the various routing strategies that can be employed in such scenarios. It's important to note that, whenever possible, it's more efficient to consolidate processing steps into a single component. This helps minimize latency and computational overhead introduced by transferring data between different locations.

4.10.1 Route Handling - What Are The Options?

As we see it, routing data through a pipeline essentially boils down to 2 essential components;

Pipeline-definition:
This component serves as the blueprint or configuration that details all the parts and operations of a pipeline. It specifies how data should be routed through various components within the system.

Message-handler:
This dynamic component actively processes and routes data according to the rules defined in the

```
{
    "geodata": [
      {
        "city_name": "Pune",
        "continent_code": "AS",
        "country_code": "IN",
        "country_name": "India",
        "latitude": 18.6161,
        "longitude": 73.7286,
        "metro_code": null,
        "postal_code": "411001",
        "region_code": "MH",
        "region_name": "Maharashtra",
        "timezone": "Asia/Kolkata"
      }
    ],
    "method": "PUT",
    "source_ipv4": "106.220.83.40",
    "source_mac": "84:62:98:f3:a4:a3",
    "suspicious": false,
    "timestamp": "2024-05-10T10:31:18.071Z",
    "url": "https://www.centralbleeding-edge.name/strategic/seamless/channels/utilize"
}
```


Figure 4.37: Log contents after processing

pipeline definition. It manages data transfer from one stage to another or from one component to another within the pipeline.

If we consider that each component can come in two states, centralized or decentralized, we can design a system, or system components, according to the following 4 principles:

Centralized pipeline-definition and centralized message-handling
In this system, both the pipeline definition and message handling are controlled from a single central point. A central authority determines how data should move through the network and also directly manages the message transfer.

Centralized pipeline-definition and decentralized message-handling
Here, while the pipeline definition is centrally defined, the actual handling of messages is distributed among various components. This means that multiple decentralized components execute the centrally planned routes.

Decentralized pipeline-definition and centralized message-handling
In this model, different components within a larger system can have independent pipeline definitions. However, all messages are forwarded through a central system that handles the data transfer between the individual components.

Decentralized pipeline-definition and decentralized message-handling
In this approach, each component in the pipeline is responsible for both the pipeline definition and the message handling. There is no central oversight; instead, each component independently handles the message routing according to its part of the pipeline definition.

These approaches are further explained in the following subsections.

Centralized Log-Routing

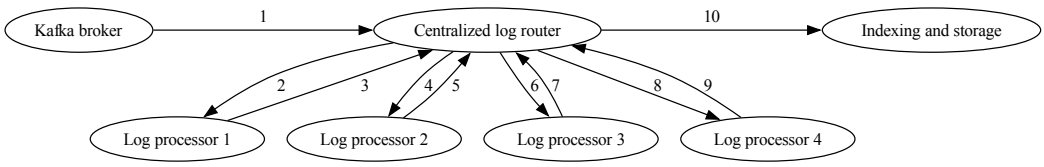


Figure 4.38: Centralized Log-Routing

This system reflects the centralized pipeline definition and centralized message handling. This approach establishes a core service that acts as the central hub for all message traffic. This service holds the pipeline definition, and once established, the message transfer is exclusively handled by this service. The operational steps taken for a select pipeline by the service, as depicted in Figure 4.38, include:

1. **Message Consumption:** The service consumes a message from the defined Kafka topic.
2. **Forwarding:** The message is forwarded to the next designated service for processing.
3. **Return and Re-forwarding:** Once processing is complete, the message is returned to the centralized service, which determines the next step in the pipeline and forwards the message accordingly.
4. **Final Delivery:** After all processing stages are complete, the log is sent to its final storage destination.

Self-contained log-routing

This system reflects the centralized pipeline definition and decentralized message handling. In this approach, we shift the message-handling away from the centralized service. Since the pipeline definition is central, we have an initial routing service embedded into the message itself. Subsequent components can then read the embedded pipeline definition and forward the message accordingly. Here's a general overview, exemplified in Figure 4.39:

- **Message consumption:** The routing service consumes a message from the Kafka Topic defined in the pipeline definition.
- **Embedding Routing-logic:** The pipeline definition is embedded into the message.
- **Forwarding:** The routing service forwards the message to the first component in the pipeline.
- **Processing and Re-forwarding:** When a component receives a message, it -
  1. Executes its designated task.
  2. Reads the instructions within the message to determine the next component in the sequence.
  3. Forwards the message to that service.
- **Final Delivery:** Eventually, all messages will be forwarded to the final storage destination.

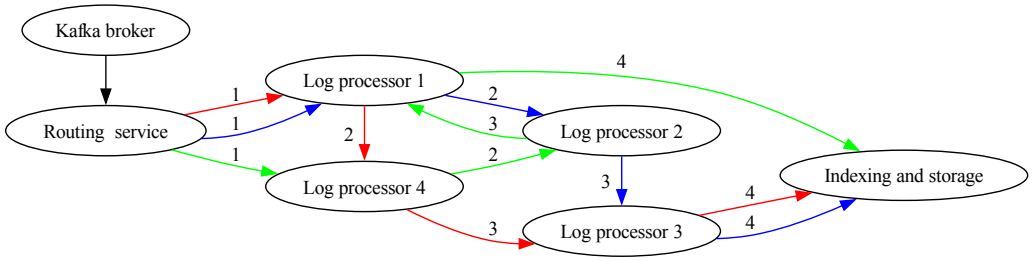


Figure 4.39: Self-Contained Log Routing

Component-To-Component Routing

This system reflects the definition of a decentralized pipeline and the decentralized message-handling approach. To build a pipeline with this method, you effectively need to configure the routing logic in each pipeline component. One way to achieve this is to forward messages according to the ingress port. For example, imagine we have a processing service involved in 3 unique pipelines. We need to configure the service to listen on a unique port for each pipeline it's a component in. The message is then forwarded according to the port from which it was retrieved.

As these pipelines are built per component, anytime the remaining components of a pipeline converge with another, said pipeline can be routed into that point in the existing pipeline. See the green pipeline in Figure 4.40

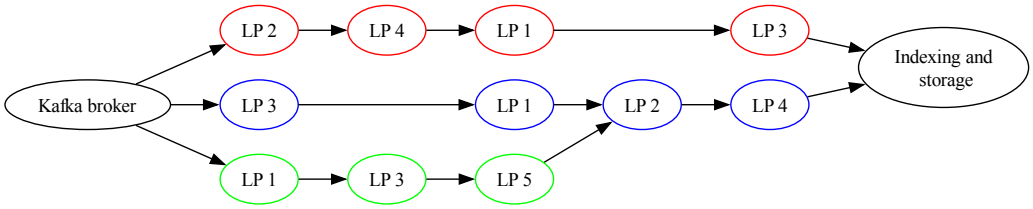


Figure 4.40: Component-To-Component Routing
LP = Log Processor

4.10.2 Route-handling - Why Does It Matter?

The choice between centralization and decentralization impacts both the design and behavior of your system. Here are the 3 most important considerations:

Parsing Efficiency:
In a centralized model, message parsing is often streamlined. Logs pass through a single component responsible for parsing. This optimizes parsing logic and prevents the message from being parsed multiple times in different locations. Additionally, only the relevant portions of the message need to be sent to external processors, reducing unnecessary data transfer.

Decentralized routing can lead to redundant parsing. Each component in the pipeline might need to parse the entire message, even if only a small segment is relevant to its task. This increases computational overhead and can negatively impact overall system performance.

Message Size:

Centralized systems enable more control over message size. After initial parsing, specific fields or data elements can be extracted and sent to external processors independently. This reduces network traffic and the processing burden on downstream components.

Messages in a decentralized system can grow in size as they pass through the pipeline. Each processing step might add enrichments or metadata, increasing the overall message footprint. This can lead to higher bandwidth costs and slower processing times.

Message Frequency:
Since the message has to return to the central hub to determine the next step in the pipeline, the frequency of message transfers doubles in the context of a centralized system. Consequently, this pattern increases network traffic and might introduce additional latency.

In a decentralized system, messages do not need to return to the central hub. As such, the frequency should be half that of the centralized version.

4.10.3 Data Routing With Vector

Vector is both the log processor and router in our solution, and this section details how to configure Vector as a log router.

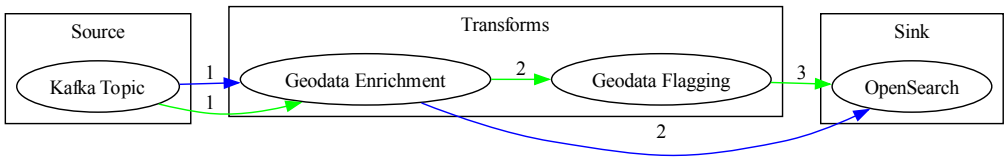


Figure 4.41: Vector configuration
Every node-connection is configured separately, arrows corresponds to the node's input source.

Configuration - Autoscaling

Our current configuration leverages CPU utilization as the primary scaling metric. Figure 4.42 showcases how we define autoscaling parameters within the helm-deploy.yaml file.

```yaml
helm-deploy.yaml
---
values:
  role: "Aggregator"
  autoscaling:
    enabled: true
    minReplicas: 3
    maxReplicas: 5
    targetCPUUtilizationPercentage: 80
    behaviour:
      scaleDown:
        stabilizationWindowSeconds: 300
```

Parameter	Description
autoscaling.enabled	Activates autoscaling for this Vector deployment.
minReplicas	Sets the minimum number of Vector instances to maintain, even during low traffic periods.
maxReplicas	Defines the upper limit for the number of Vector instances that can be spawned during peak loads.
targetCPUUtilizationPercentage	Triggers autoscaling when average CPU utilization across Vector instances reaches 80%.
stabilizationWindowSeconds	Introduces a delay of 300 seconds before scaling down to prevent Vector from oscillating between adding and removing instances due to minor fluctuations in CPU usage.

Figure 4.42: Helm values for our vector deployment

As mentioned earlier, future deployment iterations will incorporate consumer latency as a scaling metric. This will allow Vector to react to the volume of incoming logs and the processing time required for each log message. This can lead to more targeted and efficient autoscaling, ensuring optimal resource utilization.

Configuration - Data Sources

Figure 4.43 demonstrates how we configure a Kafka data source within the helm-deploy.yaml file.

```yaml
helm-deploy.yaml
---
values:
  customConfig:
    sources:
      kafka_cluster_bastion_k6:
        type: kafka
        bootstrap_servers: bastion-kafka-bootstrap.strimzi.svc.cluster.local:9092
        group_id: vector
        topics:
          - k6
```

Parameter	Description
type	Specifies the data-source type, Kafka in this case.
bootstrap_servers	This points Vector to the Kafka cluster it should connect to by specifying the hostname and port of the bootstrap server.
group_id	Assigns a unique group identifier for this Vector instance within the Kafka cluster. This is particularly important when using consumer groups for parallel processing within Kafka.
Topics	Defines a list of Kafka topics that Vector should subscribe to. In this case, Vector will listen for and ingest messages published to the ”k6” topic.

Figure 4.43: Configuring a data-source for Vector

Configuration - Data Sinks

Figure 4.44 showcases how we configure an OpenSearch data sink within the helm-deploy.yaml file.

```
helm-deploy.yaml
---
values.customConfig.sinks:
  opensearch_k6_geodata:
    type: elasticsearch
    endpoints:
      - https://10.212.170.251:9200
    inputs:
      - geodata
    api_version: v8
    auth:
      strategy: basic
      user: admin
      password: MySuperSecretPassword123!
    mode: bulk
    bulk:
      index: "k6-geodata"
      action: "create"
    tls:
      verify_certificate: false
      verify_hostname: false
```

Parameter	Description
type	Specifies that we're configuring an ElasticSearch data sink.
endpoints	Defines the URL of the OpenSearch cluster endpoint.
inputs	Identifies a specific data stream within Vector that should be routed to this sink.
api_version	Specifies the ElasticSearch API version that Vector should use for communication.
auth.strategy	Basic authentication is used in this example.
auth.user	Username for accessing OpenSearch.
auth.password	Password for accessing OpenSearch.
mode	This setting instructs Vector to send log data to OpenSearch in batches for improved efficiency.
bulk.index	Specifies the OpenSearch index where the log data should be stored.
bulk.action	Allows Vector to create the specified OpenSearch index if it doesn't already exist.
tls.verify_certificate	Disables certificate verification.
tls.verify_hostname	Disables host name verification.

Figure 4.44: Configuring data sinks in Vector

4.11 Metrics and Monitoring

To gain insights into the health and performance of our Kubernetes infrastructure and Kafka deployment, we've chosen Prometheus and Grafana as our monitoring stack.

4.11.1 Prometheus

```
prometheus/helm-deploy.yaml
---
values:
    prometheus:
      enabled: true
      ingress:
        enabled: true
        ingressClassName: nginx
        hosts:
          - prometheus.bastion.local
      prometheusSpec:
        logLevel: debug
        podMonitorSelector: {}
        podMonitorSelectorNilUsesHelmValues: true
```

Parameter	Description
ingress.enabled	Enables the creation of an ingress resource.
ingress.ingressClassName	This defines which ingress class the ingress resource should be created as.
ingress.hosts	Specifies the hostname where the Prometheus instance will be accessible.
prometheusSpec.logLevel	Log verbosity level.
prometheusSpec.podMonitorSelector	By setting this selector as empty(''), Prometheus is instructed to discover all PodMonitors for metrics scraping.
prometheusSpec.podMonitorSelectorNilUsesHelmValues	When enabled, if podMonitorSelector is set to Nil or , selectors use values from helm deployment.[25]

Figure 4.45: Prometheus Helm values

The configuration snippet presented in Figure 4.45 will deploy a Prometheus resource in our cluster. Notably, these selector settings result in the pod-selector configuration displayed in Figure 4.46. This effectively means that Prometheus will discover monitors in any namespace, so long as they have the "release: kube-prometheus-stack" label-value pair.

```
---
apiVersion: monitoring.coreos.com/v1
kind: Prometheus
spec:
  podMonitorNamespaceSelector: {}
  podMonitorSelector:
    matchLabels:
      release: kube-prometheus-stack
```

Figure 4.46: Default Prometheus pod-selector configuration

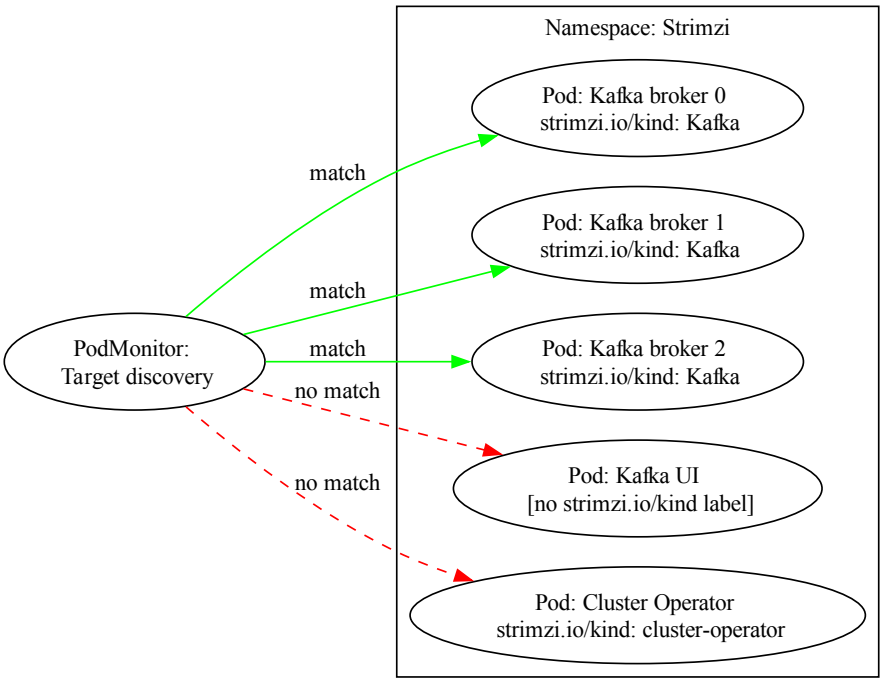


Figure 4.47: PodMonitor for the strimzi namespace
The PodMonitors target discovery is configured to match pods with the label-value pair 'strimzi.io/kind: Kafka' in the 'strimzi' namespace.

Pod Monitors

Pod Monitors are custom Kubernetes resources used by Prometheus to define how metrics should be scraped from pods. More specifically, they're used to discover the endpoints from which Prometheus should scrape metrics. In Figure 4.48, you can see the key settings we need to configure the pod-monitor we deploy for our Kafka clusters - consult also Figure 4.47:

```yaml
strimzi/pod-monitor.yaml
---
apiVersion: monitoring.coreos.com/v1
kind: PodMonitor
metadata:
  name: kafka-resources-metrics
  namespace: prometheus
  labels:
    app: strimzi
    release: kube-prometheus-stack
spec:
  selector:
    matchExpressions:
      - key: "strimzi.io/kind"
        operator: In
        values: ["Kafka", "KafkaConnect", "KafkaMirrorMaker", "KafkaMirrorMaker2"]
  namespaceSelector:
    matchNames:
      - strimzi
  podMetricsEndpoints:
  - path: /metrics
    port: tcp-prometheus
```

Parameter	Description
metadata.labels.release	Adding this label allows the podmonitor to be discovered by Prometheus.
spec.selector.matchExpressions	In this stanza we can configure fine-grained targeting for the podmonitor. Specifically, we configure the podmonitor to target pods with the label "strimzi.io/kind" and any value from the array. In simpler terms, we configure the podmonitor to target any resources that could be deployed by the operator - including our Kafka clusters.
spec.namespaceSelector.matchNames	This limits the podmonitor to only discover pods in the 'strimzi' namespace.
spec.podMetricsEndpoints	Here we configure the URL and port where the metrics are exposed.

Figure 4.48: Kafka Pod Monitor configuration

Service Monitors

Similar to Pod Monitors, Service Monitors are custom resources within the Prometheus Operator ecosystem. However, instead of targeting individual pods directly, Service Monitors instruct Prometheus to scrape the metrics via the Kubernetes service endpoints instead.

```yaml
---
apiVersion: v1
kind: Service
metadata:
  labels:
    app.kubernetes.io/component: Aggregator
    app.kubernetes.io/instance: vector
    app.kubernetes.io/name: vector
spec:
  ports:
  - name: api
    port: 8686
    protocol: TCP
    targetPort: 8686
  - name: prometheus-exporter
    port: 9598
    protocol: TCP
    targetPort: 9598
  selector:
    app.kubernetes.io/component: Aggregator
    app.kubernetes.io/instance: vector
    app.kubernetes.io/name: vector
```

Parameter	Description
metadata.labels	The label-value pairs in this stanza is what is targeted by the Service Monitor for target discovery.
spec.ports	Each entry in this list defines the ports exposed by the service, and how traffic should be directed to the associated pods.
spec.ports.name	Assignes a descriptive name to each port, which can be referenced in the ServiceMonitors 'endpoints' section.
spec.ports.port	The external port number on which the service will accept traffic.
spec.ports.protocol	Specifies the protocol used for connections to this port.
spec.ports.targetPort	The port on the pods where the incoming traffic will be forwarded.
spec.selectors	This defines the label-value pairs that pods must possess to be considered part of the service. Kubernetes uses these selectors to maintain the association between the service and its corresponding backend pods.

Figure 4.49: Vector service configuration

As can be seen from the configuration stanzas in Figure 4.49, the Kubernetes Service effectively ties all the pods into one service. We can then use a ServiceMonitor to discover the scrape-targets for Prometheus. Here we instruct the ServiceMonitor to discover services where all 3 labels match in the namespace 'vector'. Additionally, we define the endpoint as the "prometheus-exporter" port we defined in the service. Consult Figure 4.51 for a visual explanation of the configuration demonstrated in Figure 4.50.

```yaml
vector/servicemonitor.yaml
---
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: vector-metrics
  namespace: prometheus
  labels:
    app: vector
    release: kube-prometheus-stack
spec:
  selector:
    matchLabels:
      app.kubernetes.io/component: Aggregator
      app.kubernetes.io/instance: vector
      app.kubernetes.io/name: vector
  namespaceSelector:
    matchNames:
      - vector
  endpoints:
    - port: prometheus-exporter
```

Figure 4.50: Vector Service Monitor

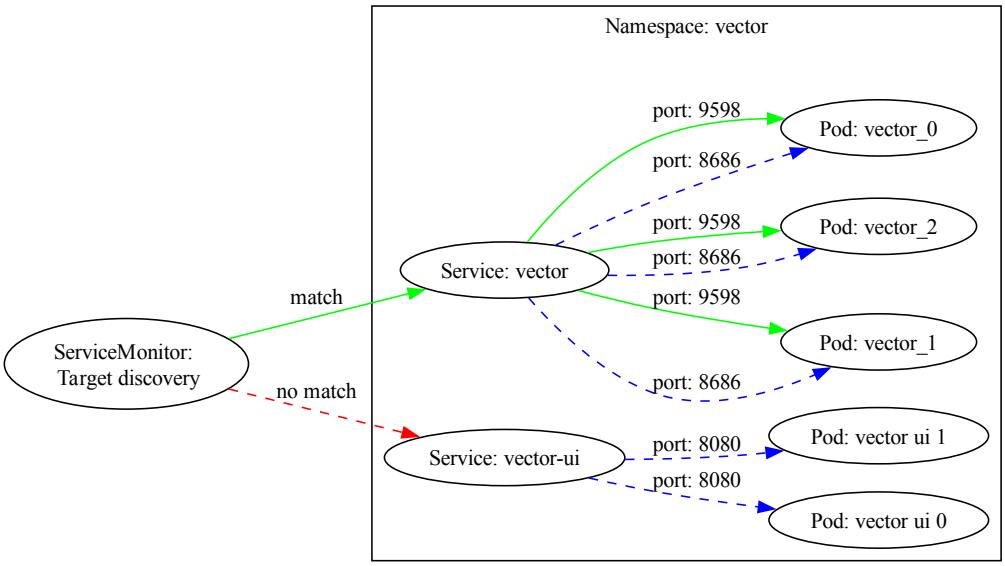


Figure 4.51: ServiceMonitor for the vector namespace
Labels are not included in the figure for visual clarity. However, The "vector" service exactly matches all 3 labels defined in the ServiceMonitor.

4.11.2 Grafana

Grafana provides a flexible dashboarding solution to visualize the vast amounts of metrics collected by Prometheus. We deploy Grafana alongside Prometheus within 'the kube-prometheus-stack' Helm chart for ease of use and configuration. Figure 4.52 highlights and explains the relevant configuration settings.

```yaml
prometheus/helm-deploy.yaml
---
values:
    grafana:
      enabled: true
      adminPassword: prom-operator
      ingress:
        enabled: true
        ingressClassName: nginx
        hosts:
          - grafana.bastion.local
```

Parameter	Description
grafana.enabled	Whether or not to deploy Grafana with the chart.
grafana.adminPassword	Sets the initial administrative password for Grafana. In a production setting, this value should be dynamically provisioned.
grafana.ingress.enabled	Whether or not to create an ingress resource for Grafana.
grafana.ingress.ingressClassName	Defines the ingress class for the created ingress resource.
grafana.ingress.hosts	specifies the hostname where Grafana can be accessed.

Figure 4.52: Grafana Helm values

The operator can deploy a sidecar container used for dashboard discovery [18]. This discovery works much like the way monitors discover their resources. In this case, it will deploy all dashboard configurations from configMaps that match the label-value pair "grafana_dashboard:1". This is what allows us to easily deploy Strimzi's bundled Grafana dashboards, as seen in Figure 4.25.

4.12 Traffic Simulation

A traffic simulation component is necessary to test the capacity and resilience of the proposed infrastructure thoroughly. This section outlines the design and implementation of a traffic simulation strategy that generates dummy log data, enabling us to evaluate the infrastructure performance under load.

4.12.1 The Citadel Cluster

To ensure realistic simulation, we deployed a separate Kubernetes cluster where we could deploy and scale the log-generation workloads according to our needs. This cluster is dedicated to running the traffic generation tools, isolating them from the main Bastion cluster where the primary infrastructure resides. This separation is done to more accurately mimic the real-world scenario where log sources would originate externally from the Bastion cluster.

4.12.2 Grafana K6

Grafana K6 was selected as the primary tool for traffic simulation due to its flexibility, extensibility, and performance testing capabilities. Additionally, the Kafka extension[19] allows direct streaming of generated logs to a Kafka topic, while the Faker extension[20] provides "realistic" data generation to enhance the simulation.

[18] 'https://github.com/prometheus-community/helm-charts/blob/main/charts/kube-prometheus-stack/values.yaml' see line 1034-1053

[19] https://github.com/grafana/xk6-output-kafka

[20] https://github.com/szkiba/xk6-faker

```
const vuCounts = [];
if (randomIntBetween(1, 10) < 10) {
    for (let i = 0; i < 5; i++) {
        vuCounts.push(randomIntBetween(1, 100));
    }
} else {
    for (let i = 0; i < 5; i++) {
        vuCounts.push(randomIntBetween(100, 500));
    }
};

export const options = {
    stages: [
        { target: vuCounts[0], duration: "1m" },
        { target: vuCounts[1], duration: "1m" },
        { target: vuCounts[2], duration: "1m" },
        { target: vuCounts[3], duration: "1m" },
        { target: vuCounts[4], duration: "1m" }
    ]
};
```

Figure 4.53: k6 log generation - traffic spikes

K6 simulation script

The K6 simulation script is the core component responsible for generating realistic log data and simulating the desired traffic patterns for testing the infrastructure. The Faker extension provides the ability to populate fields like IP addresses, HTTP methods, and URLs with realistic values, enhancing the quality of the simulation (Figure 4.54 shows a snippet of this logic). To simulate the dynamic nature of real-world traffic, the script incorporates traffic spikes with varying intensities (Figure 4.53). These spikes are designed to stress the logging infrastructure under different load conditions, helping to identify potential bottlenecks or limitations in its ability to scale effectively. It should be noted that the code snippets provided in the figures are excerpts from the full K6 script.

```
export default function () {
    const now = new Date();
    writer.produce({
        messages: [
            {
                value: schemaRegistry.serialize({
                    data:{
                        timestamp: now.toISOString(),
                        source_ipv4: faker.internet.ipv4Address(),
                        source_mac: faker.internet.macAddress(),
                        method: faker.internet.httpMethod(),
                        url: faker.internet.url(),
                    },
                    schemaType: SCHEMA_TYPE_JSON,
                }),
            },
        ],
    });
    sleep(1);
}
```

Figure 4.54: k6 log generation - log content

```
k6-log-simulator/k8s/log-simulator/kustomization.yaml
---
configMapGenerator:
  - name: k6-log-simulator-config
    files:
      - k6-log-simulator.js
  - name: k6-load-simulator-testrun
    files:
      - testrun.template.yaml
generatorOptions:
  disableNameSuffixHash: true
```

Figure 4.55: Generating configmaps for the cronjob

Preparing the necessary resources

Before deploying the simulation workloads, some resources need to be prepared and available for the testing process. Kustomize is used to generate ConfigMaps (Figure 4.55) that encapsulate the K6 simulation script and the template for the testrun CRDs.

Furthermore, since RBAC plays a vital role in managing permissions within Kubernetes, a Role (Figure 4.56) is created to define a specific set of permissions granted to the automation mechanism (in this case, Cron), such as the ability to create new testrun jobs. This Role is linked to a service account via a RoleBinding, ensuring that Cron possesses the necessary authorization to interact with the K6 testrun CRDs.

```
k6-log-simulator/k8s/log-simulator/rbac.yaml
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: k6-log-simulator
rules:
  - apiGroups:
      - k6.io
    resources:
      - testruns
    verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
```

Figure 4.56: K6 RBAC - Creating role that can interact with testrun resources

Cronjobs are leveraged to schedule the deployment of K6 workloads at regular intervals, enabling continuous and scalable traffic simulation that will thoroughly test the logging infrastructure.

Deploying Workloads

Cronjobs are utilized within the Kubernetes environment to automate the deployment of the previously mentioned workloads. A dedicated Cronjob schedules a container that executes shell com-

mands (Figure 4.57).

```
export TEST_RUN_NAME="k6-log-simulator-$(date +'%Y%m%d%H%M%S')";
envsubst < /tmp/config/testrun.template.yaml > /tmp/workdir/testrun.yaml;
kubectl apply -f /tmp/workdir/testrun.yaml;
```

Figure 4.57: Cron - Shell commands run by deployment job

These commands are responsible for:

1. Generating a unique name for the K6 testrun instance.
2. Applying substitutions to a testrun template (substituded fields not included in Figure 4.58), ensuring the testrun CRD's are uniquely named.
3. Deploying the generated testrun using kubectl apply.

```
k6-log-simulator/k8s/log-simulator/testrun.template.yaml
---
spec:
  parallelism: 5
  separate: false
  script:
    configMap:
      name: k6-log-simulator-config
      file: k6-log-simulator.js
  runner:
    image: karlhenh/k6-kafka-faker:runner-v1.0
    env:
      - name: KAFKABOOTSTRAP
        value: 10.212.170.52.9094
      - name: KAFKATOPIC
        value: k6
  starter:
    image: karlhenh/k6-kafka-faker:starter-v1.0
```

Figure 4.58: K6 testrun template - selected configuration options

Workload cleanup

To avoid accumulating thousands of stale testruns, a separate Cronjob is deployed for cleanup purposes. It executes a shellscript (Firuge 4.59) that performs the following:

- Retrieves a list of K6 testruns along with their ages.
- Parses the age, calculating the total age in seconds.
- For testruns exceeding a defined threshold (e.g., 10 minutes), it triggers a kubectl delete command to remove the testrun.

```
kubectl get testrun --namespace k6 | awk 'NR>1 {print $1, $3}' | while read name age; do
    if [[ $age =~ .*m.* ]]; then
        minutes="${age%m*}"
        seconds="${age#*m}"
        seconds="${seconds%s}"
        total_age_in_seconds=$((minutes * 60 + seconds))
    else
        total_age_in_seconds=$age
        total_age_in_seconds="${total_age_in_seconds%s}"
    fi

    if [[ $total_age_in_seconds -gt 600 ]]; then
        echo "Deleting testrun: $name (older than 10 minutes)"
        kubectl delete testrun $name --namespace k6
    fi
done
```

Figure 4.59: Cron - Shell commands run by the cleanup job

Chapter 5

Security

While security was a key consideration throughout development, the POC nature of the project means that not all security measures have been implemented to the standards required for a production environment.

This chapter serves as an overview and starting point for the security measures that needs to be changed and/or implemented in the project by the NTNU SOC before deploying it to production. The chapter is heavily influenced by the 'Kubernetes Hardening Guidance' [2] and 'Defending Continuous Integration/Continuous Delivery Environments' [1] documents published by the National Security Agency (NSA) and the Cybersecurity and Infrastructure Security Agency (CISA). In addition, some minor insights have been drawn from the Kubernetes Benchmark [14] published by the Center for Internet Security (CIS).

It's important to acknowledge that this chapter does not exhaustively cover every potential security aspect. Rather, it serves as a foundational reference, highlighting the importance of consulting established security guides and benchmarks before transitioning this POC into a production environment.

5.1 Threat Model

Based on the Kubernetes Hardening Guidance [2], the three most likely threats to a Kubernetes cluster are:

1. **Supply Chain Compromise**[1]**:** These risks stem from the interconnected nature of the components and dependencies involved in building and maintaining a Kubernetes environment. Vulnerabilities or malicious code can be introduced at various points in the supply chain, including:
    - **Container/Application Level:** Malicious actors can inject harmful code into third-party applications or containers, which, if deployed within the cluster, can compromise its security.
    - **Container Runtime:** The software responsible for running containers can have vulnerabilities that, if exploited, could lead to container breakouts or unauthorized access to the underlying host system.
    - **Infrastructure:** The infrastructure on which Kubernetes runs, including the hardware and software components, can be compromised, providing attackers with a foothold to exploit the cluster.

[1] Technically considered a technique by MITRE: https://attack.mitre.org/techniques/T1195/

2. **Malicious Threat Actors:** External attackers actively seek to exploit vulnerabilities or leverage stolen credentials to gain unauthorized access to the cluster. They may target various entry points:
    - **Control Plane:** As the control plane is responsible for managing the cluster, threat actors frequently exploit exposed control plane components with inadequate protection layers.
    - **Worker Nodes:** Worker nodes host the kubelet and kube-proxy service. Vulnerabilities or misconfigurations in these nodes allow threat actors to exploit the kubelet and kube-proxy services.
    - **Containerized Applications:** Applications accessible from outside the cluster can serve as entry points, enabling attackers to move laterally within the cluster or escalate privileges.
3. **Insider Threats:** Individuals within the organization, whether intentionally or unintentionally, can misuse their privileges or knowledge to compromise the cluster. Potential actors include:
    - **Administrators:** Administrators with broad control over containers can execute commands or access sensitive information, potentially leading to a complete cluster takeover.
    - **Users:** Users with access to containerized services can exploit vulnerabilities in applications or other cluster components.
    - **Cloud Service or Infrastructure Provider:** If compromised, these providers with access to physical systems or hypervisors can undermine the entire Kubernetes environment.

5.2 Supply Chain Security

The security of a Kubernetes environment is linked to the integrity of its supply chain. As highlighted in the NSA/CISA Kubernetes Hardening Guidance[2], supply chain attacks can introduce vulnerabilities at multiple levels, potentially compromising the entire logging infrastructure.

In the POC, container images were sourced from various locations, including Helm charts and public registries like Docker Hub. While Helm charts and official registries provide some degree of trust, the guidance emphasizes the importance of verifying the integrity and authenticity of all container images, regardless of their source. Consider the following security measures as a starting point for mitigating supply-chain related risks:

- **Image Source Verification:**
    - Limit the use of container images to trusted sources with established security practices.
    - Utilize private registries for storing and managing images, especially those modified in-house.
    - Regularly scan all container images for vulnerabilities and malware using reputable scanning tools. Integrate image scanning into the CI/CD pipeline to automate this process.
- **Image Integrity Verification:**
    - Deploy container images with name, tag, and digest(SHA256) to ensure the image hasn't been changed or tampered with in transit[2].
    - Consider using digital signatures to verify the authenticity of container images, especially those from third-party sources.

[2]https://kubernetes.io/docs/concepts/containers/images/#image-names

5.3 Secret Management

5.3.1 Certificates And SSH Keys

Currently, all certificates, most SSH keys, and some sensitive Kubernetes configuration files are created and/or distributed via Terraform (or Ansible by extension). This includes the root CA certificate and key, SSH keypairs for internal communication, GitLab deploy keys, kubeconfig files, and OpenSearch certificates.

Since we were informed that the NTNU SOC had established procedures for generating and distributing certificates, we opted for a simplified approach to certificate creation and management, as the NTNU SOC is expected to integrate our solution with their existing systems at a later time. To facilitate this, Cert-manager is deployed to streamline the integration between the Kubernetes clusters and the certificate distribution servers. To alleviate these shortcomings, we recommend starting with the following:

- **Integrate with NTNU certificate distribution:** Replace our current certificate generation and distribution with the NTNU SOC's established procedures. This will ensure consistency with organizational security standards and leverage existing infrastructure.
- **1Password as a KMS:** For secrets not already covered by NTNU SOC procedures, implement 1Password as a dedicated KMS. 1Password provides a centralized and secure way to manage secrets, reducing the risk of exposure. It integrates well with both Kubernetes through its operator[3], and with FluxCD[4].

Once the NTNU SOC's certificate distribution procedures are in place and the infrastructure is integrated with their DNS services, it's important to enable hostname and certificate verification everywhere TLS is implemented.

5.3.2 Terraform State File

The Terraform state file currently stores sensitive information, including secrets and infrastructure details. This poses a significant risk if the state file is compromised. However, since we use GitLab as a remote backend, the state file is encrypted both at rest[5], and in transit via TLS. Additionally, due to the nature of Terraform remote backends, the state file is only held in memory when in use[6], mitigating the risk of exposure. To further enhance security, implement the following recommendations:

- **Strict Access Controls:** Implement strict access controls for the Terraform state file within GitLab. Only authorized personnel should have access, and their permissions should be regularly reviewed.
- **Regular Backups:** Maintain regular backups of the state file in a secure location. This will allow for recovery in case of accidental deletion or corruption.

5.4 Kubernetes Security

5.4.1 Network Policies

The default networking policies within Kubernetes is to allow all pods within the cluster to communicate with each other. Network policies allow granular definition of rules that can control traffic

[3]https://developer.1password.com/docs/k8s/k8s-operator/
[4]https://fluxcd.io/flux/security/secrets-management/
[5]https://docs.gitlab.com/ee/administration/terraform_state
[6]https://developer.hashicorp.com/terraform/language/state/sensitive-data

flow between pods and external resources. As emphasized in the NSA/CISA Kubernetes Hardening Guidance, network policies are essential for limiting the blast radius of a potential compromise. In the event of a security breach, network policies can prevent attackers from easily moving laterally within the cluster and accessing sensitive components. Consider the following improvements before deploying the POC into production:

- **Default Deny:** Start with a default deny policy for all namespaces. This means that no traffic is allowed unless explicitly permitted by a network policy. The NSA/CISA guidance strongly recommends this approach as it significantly reduces the attack surface by isolating each namespace from all other namespaces in the cluster.
- **Least Privilege:** Each component should be granted only the necessary permissions for communication between other components. For instance, a network policy should be implemented to allow Prometheus to scrape metrics from specified endpoints in specific namespaces.
- **Network Segmentation:** Consider segmenting the network into different zones and apply network policies to control traffic between these zones. This adds another layer of security by restricting communication paths and making it harder for attackers to move laterally. For example, a network policy could be implemented that only allows traffic from the frontend zone to the backend zone, and only allows traffic from the backend zone to the data zone.

5.4.2 Resource Constraints

Resource constraints are essential for preventing resource exhaustion attacks, a type of denial of service attack where an attacker intentionally consumes excessive resources by exploiting uncontrolled resource quotas[7]. Within Kubernetes, this could involve an attacker deploying a pod that rapidly consumes all available CPU or memory resources, starving legitimate pods and disrupting the operation of critical services. For example, an attacker could create a pod that runs a computationally intensive task, such as cryptocurrency mining, or a pod that allocates large amounts of memory without releasing it.

Kubernetes provides the LimitRange[8] and ResourceQuota[9] resources to mitigate these types of events. LimitRanges allow you to set minimum and maximum resource limits for individual pods or containers within a namespace. For instance, you could define a LimitRange that prevents any single pod from using more than 500 millicores of CPU or 1GB of memory. ResourceQuotas, on the other hand, enables you to restrict the aggregate resource consumption of an entire namespace. This ensures that even if multiple pods within a namespace try to consume excessive resources, the overall resource usage remains within acceptable limits. In summary:

- **Limit Ranges:** Define limit ranges for each namespace to set minimum and maximum resource limits (CPU, memory) for pods and containers.
- **Resource Quotas:** Implement resource quotas for each namespace to limit the total resource consumption of the namespace.

These constraints prevent any single pod or namespace from monopolizing resources, ensuring the stability and availability of the Kubernetes cluster even in the face of malicious or accidental resource exhaustion attempts.

5.4.3 Role Based Access Control

Role Based Access Control (RBAC) is one of the baseline provided security mechanisms in Kubernetes, governing permissions for users and service accounts to interact with cluster resources. As

[7] https://cwe.mitre.org/data/definitions/400.html
[8] https://kubernetes.io/docs/concepts/policy/limit-range/
[9] https://kubernetes.io/docs/concepts/policy/resource-quotas/

recommended in the NSA/CISA Kubernetes Hardening Guidance, the principle of least privilege should be central to RBAC implementation. This means granting users and service accounts only the permissions necessary to perform their specific tasks, minimizing the potential damage from compromised credentials.

the POC includes some examples of RBAC implementation, but further refinement is needed for a production environment:

- **Disable anonymous access:** Ensure that kubelets are started with the '–anonymous-auth=false' flag.[10]
- **Service Account Permissions:** Since the POC is primarily operated through automated processes like GitOps, prioritize defining and restricting permissions for service accounts. Limit the scope of service account permissions to the specific resources and actions they need to perform. For example, the Prometheus service account should only have permission to scrape metrics from the specified endpoints and store them in its time-series database.

5.5 GitOps Security

Given that the Git repository serves as the single source of truth in our GitOps implementation, ensuring its security should be of high priority. Unauthorized modifications to the repository could lead to the deployment of compromised or malicious code and configurations.

As advised by NSA and CISA in their 'Defending CI/CD Environments' document[1], implementing least-privilege access controls ensures that only authorized personnel can modify infrastructure and application configurations. This can be achieved with GitLab's built-in RBAC[11], where permissions should be granted based on roles and responsibilities. Additionally, enforcing the use of strong authentication mechanisms, such as Two-Factor Authentication (2FA)[12], can further protect against unauthorized access to the pipelines.

While GitOps promotes storing configurations as code, sensitive information like passwords, API keys, and tokens should never be stored in plain text within the repository. This can be avoided by leveraging GitLab's 'CI/CD variables' feature, as demonstrated in the OpenSearch Deployment pipeline(4.4), or by integrating with a KMS like 1Password. In addition to securely storing and managing secrets, these solutions can dynamically inject the secrets into the environments during deployment, mitigating the possibility of secrets being reverse-engineered out of applications[12]. Consequently, ensure that secrets are not written to persistent storage by any of the pipeline components. In summary:

- **Least-privilege policies:** Utilize GitLab's built-in access controls to enforce the principle of least-privilege.
- **Secure secret storage and injection:** Utilize GitLab CI/CD variables or a KMS tool to handle secrets within the pipelines.
- **Ensure ephemeral secrets:** Verify that no pipelines store secrets in persistent storage.

[10] https://kubernetes.io/docs/reference/access-authn-authz/kubelet-authn-authz/#kubelet-authentication
[11] https://docs.gitlab.com/ee/user/permissions.html
[12] https://docs.gitlab.com/ee/security/two_factor_authentication.html

Chapter 6

Results

In this chapter, we present the results of our performance evaluation of the log analytics pipeline. We conducted two main tests: one with OpenSearch as the log storage and indexing solution, and another without OpenSearch, using a blackhole sink to isolate the performance of the pipeline's core components.

These tests aimed to assess the pipeline's ability to handle varying workloads, identify potential bottlenecks, and evaluate OpenSearch's impact on overall performance.

To review the jupyter notebooks and corresponding datasets used in this analysis, consult Appendix C.

6.1 Performance Evaluation With OpenSearch

Figure 6.1 provides insights into the pipeline's message-handling capabilities. During the peak load periods, Kafka reports a steady influx of 50,000 messages per second, and we can see that Vector scales the pod count both up and down as expected. However, there are no indications of an eventual convergence of message ingestion and consumption. The baseline message consumption rate does not increase as expected with the increased pod count. Instead, we see spikes in message consumption that align with said increases in pod count, suggesting high batch consumption when Vector scales up and re-coordinates pods. This indicates a potential limitation in Vector's ability to process the incoming message volume consistently.

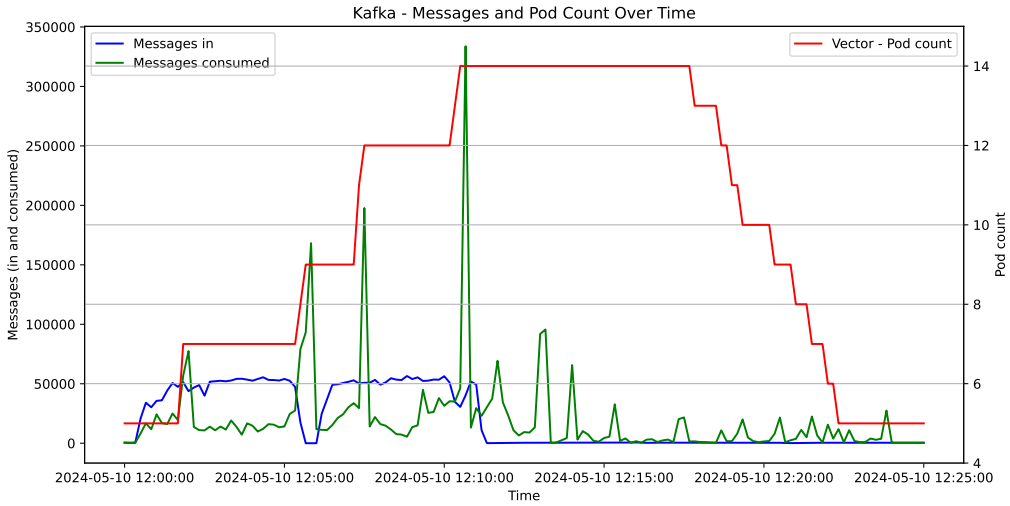


Figure 6.1: Kafka - Messages in and consumed per second over time

The impact of this processing limitation becomes evident in Figure 6.2, which depicts the latency of message consumption. As the load increases, Vector falls significantly behind, reaching a maximum of 11 million messages behind the head of the Kafka topic. The latency graph shows two distinct peaks that coincide with the end of each traffic burst, further emphasizing the pipeline's struggle to keep pace with the incoming data. This lag in message consumption is a clear indicator that the pipeline, in its current configuration, cannot adequately handle the volume of messages received during peak load.

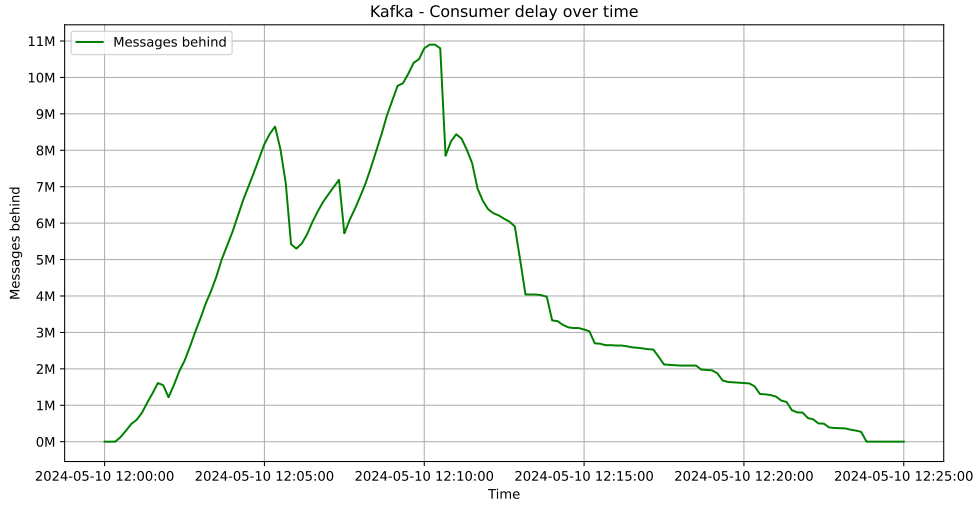


Figure 6.2: Kafka - Consumer latency

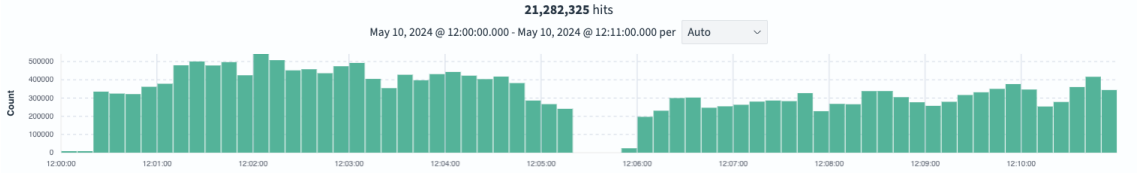


Figure 6.3: Records indexed in OpenSearch

While the pipeline successfully indexes over 21 million messages in OpenSearch (Figure 6.3), several anomalies raise concerns about the stability and efficiency of either the indexing process or the OpenSearch cluster as a whole. Figure 6.4 reveals unexpected fluctuations in the reported index size, with abrupt decreases and increases that do not correlate with the expected data influx. These fluctuations suggest potential issues with OpenSearch's internal mechanisms for managing index segments or handling incoming data.

Furthermore, Vector logs warning messages like the one shown in Figure 6.5[1], indicates that requests to OpenSearch are timing out. This could be attributed to several factors, such as network latency, resource contention within OpenSearch, or an overload of indexing requests.

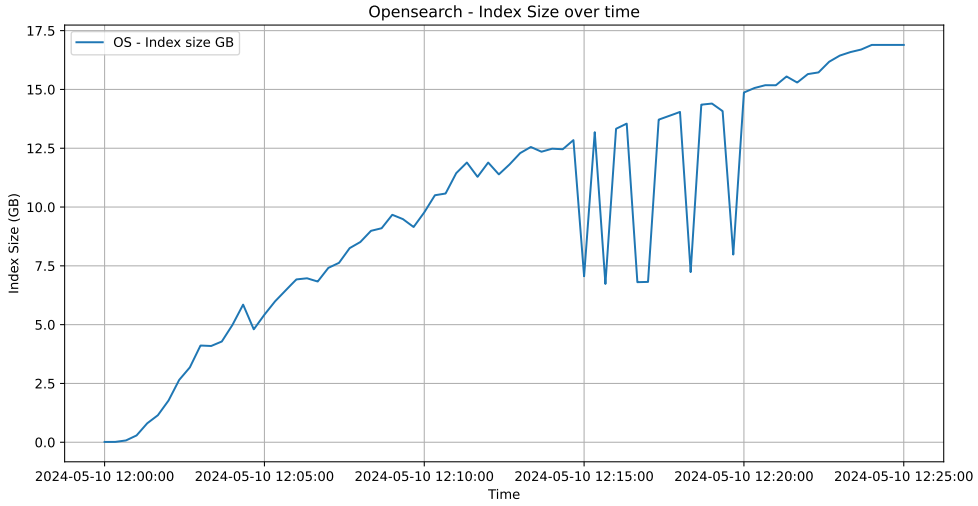


Figure 6.4: OpenSearch index size

```
vector 2024-05-08T13:44:18.631975Z  WARN sink{component_kind="sink"
↪  component_id=opensearch_k6_geodata_test component_type=elasticsearch}:request{request_id=619}:
↪  vector::sinks::util::retries: Request timed out. If this happens often while the events are
↪  actually reaching their destination, try decreasing `batch.max_bytes` and/or using
↪  `compression` if applicable. Alternatively `request.timeout_secs` can be increased.
↪  internal_log_rate_limit=true
```

Figure 6.5: Example Vector warning

A closer look at OpenSearch's resource utilization provides additional evidence of performance bottlenecks. As depicted in Figure 6.6, the cluster nodes consistently operate with minimal free memory, often hovering near critically low levels. This lack of available memory can lead to increased garbage collection activity, degraded performance, and potential instability. Moreover, the persistently high disk I/O time (Figure 6.7) suggests that OpenSearch is struggling to keep up with the rate of incoming data, resulting in substantial strain on the underlying storage system.

The combination of erratic index size fluctuations, request timeouts, and strained resource utilization strongly suggests that the indexing process in OpenSearch becomes a major bottleneck under the tested load.

[1]The timestamp in the figure is inconsistent with the presented graphs as this message is from one of the preliminary load tests where we saw the same performance patterns. We did see the same messages presented in the graphs during the test but simply forgot to record the message then.

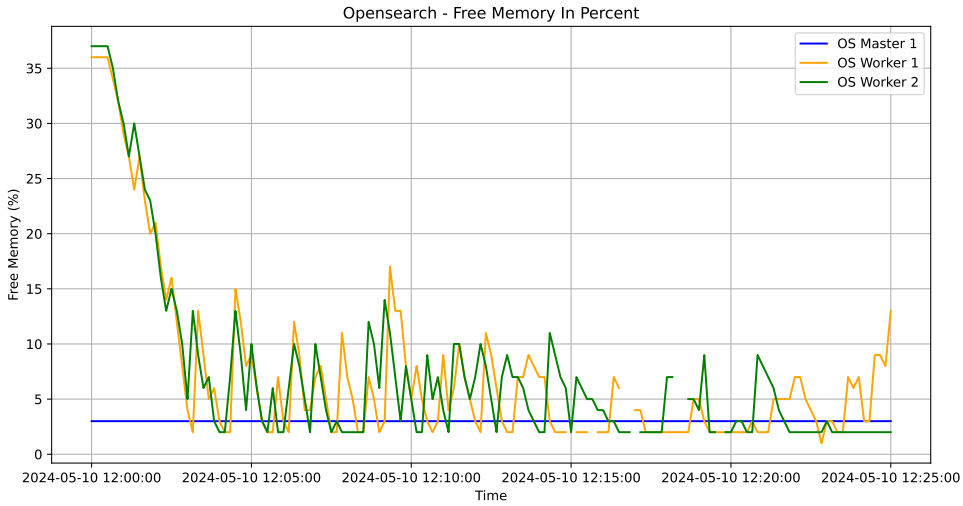


Figure 6.6: Opensearch cluster - free memory

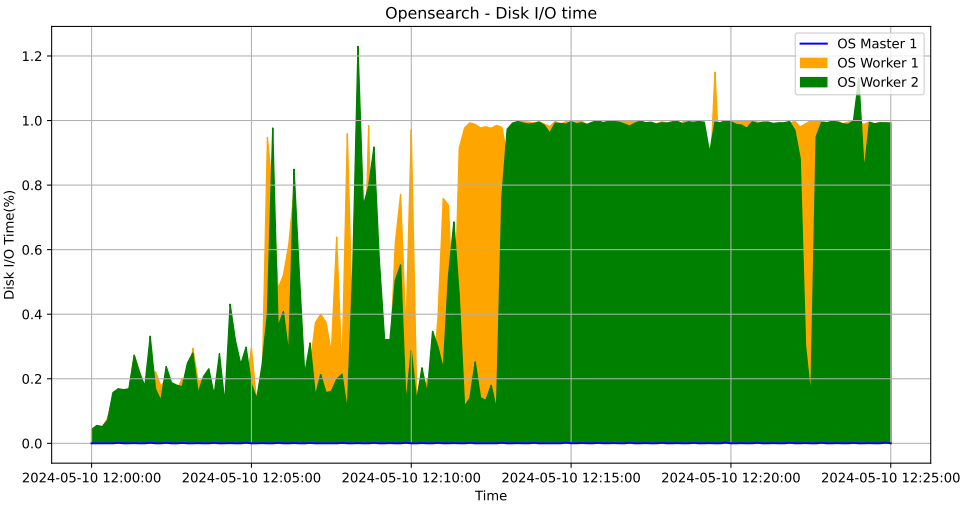


Figure 6.7: Opensearch cluster - Disk I/O Time

Based on the observations presented, it is evident that OpenSearch, in its current configuration, cannot sustain the required performance levels to handle the data volumes encountered during testing. Given the challenges encountered with OpenSearch and the fact that both configuration and optimization were outside the initial project scope, we decided to shift our focus to evaluating the pipeline's core components.

6.2 Performance Evaluation Without OpenSearch

To evalute the pipeline's core components in isolation, we replaced the OpenSearch sink with a blackhole sink, which discards incoming data after it has been processed by the pipeline. This eliminates the potential interference and bottlenecks introduced by OpenSearch and gives a clearer understanding of its capabilities and limitations when not constrained by the storage layer.

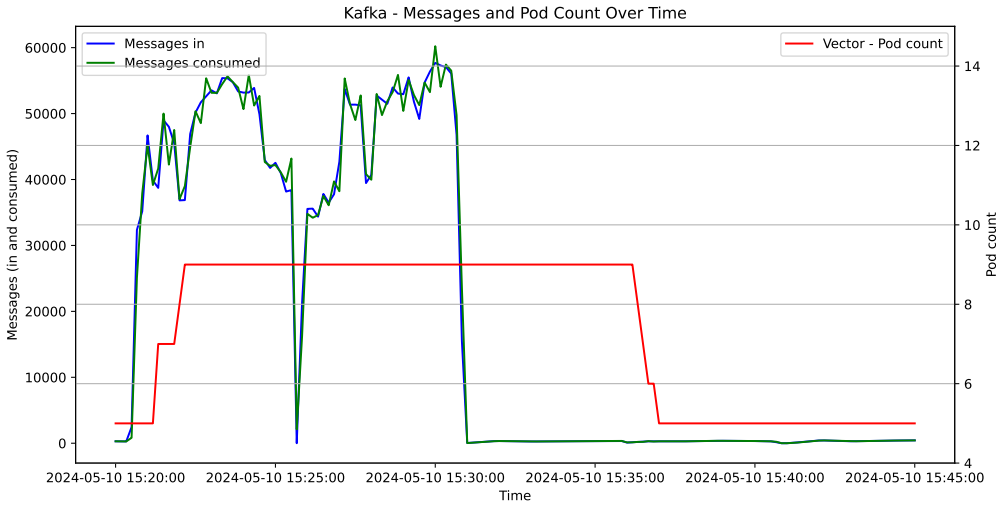


Figure 6.8: Kafka - Messages in and consumed per second over time
without OpenSearch

Figure 6.8 demonstrates a marked improvement in message handling compared to the previous test with OpenSearch. The number of messages consumed by Vector closely tracks the incoming message rate from Kafka, indicating that Vector is now able to process messages at the rate they are received. This is further supported by the consumer latency graph in Figure 6.9. While a slight delay exists, it remains consistent throughout the test and does not exhibit the escalating pattern observed previously.

The lower graph in Figure 6.9 clearly contrasts the consumer latency with and without OpenSearch. The dramatic reduction in latency when using the blackhole sink underscores the significant impact OpenSearch had on the pipeline's performance. This comparison highlights the pipeline's ability to maintain a steady pace when not constrained by the storage layer.

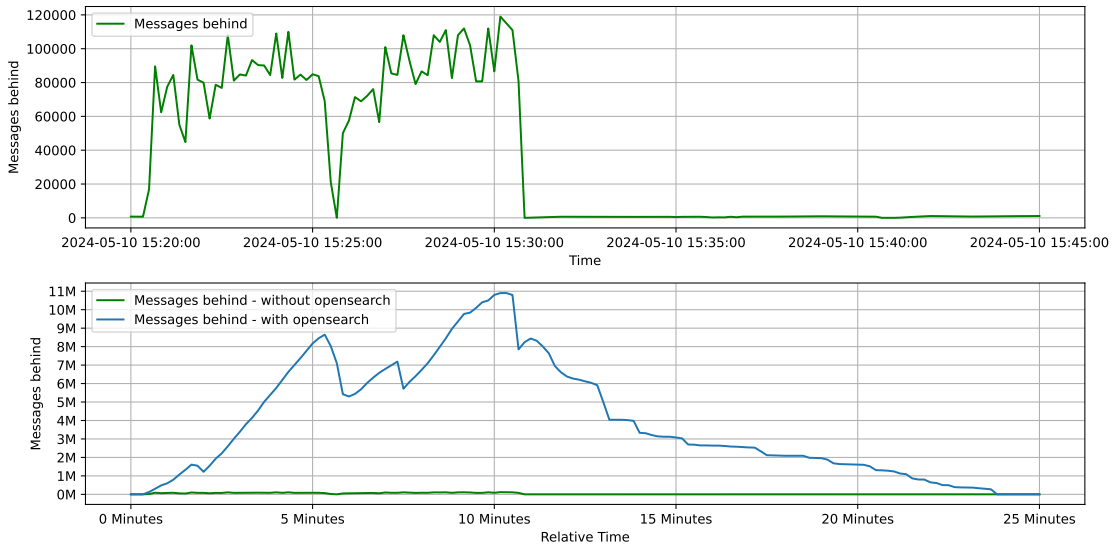


Figure 6.9: Kafka - Consumer latency
The bottom graph depicts the difference in latency with and without the OpenSearch sink

Figure 6.10 provides insights into Vector's internal event processing. The number of input events registered by Vector is roughly double the number of messages reported by Kafka. This is likely

due to Vector's internal event handling, where each message may generate multiple events as it passes through different stages of processing.[2]

The "Geodata" transform's utilization, representing the percentage of time the transform is actively processing data, peaks at around 40% during the initial surge of traffic and settles to a stable 20% during sustained load. This indicates that the transform has sufficient capacity to handle the incoming data volume.

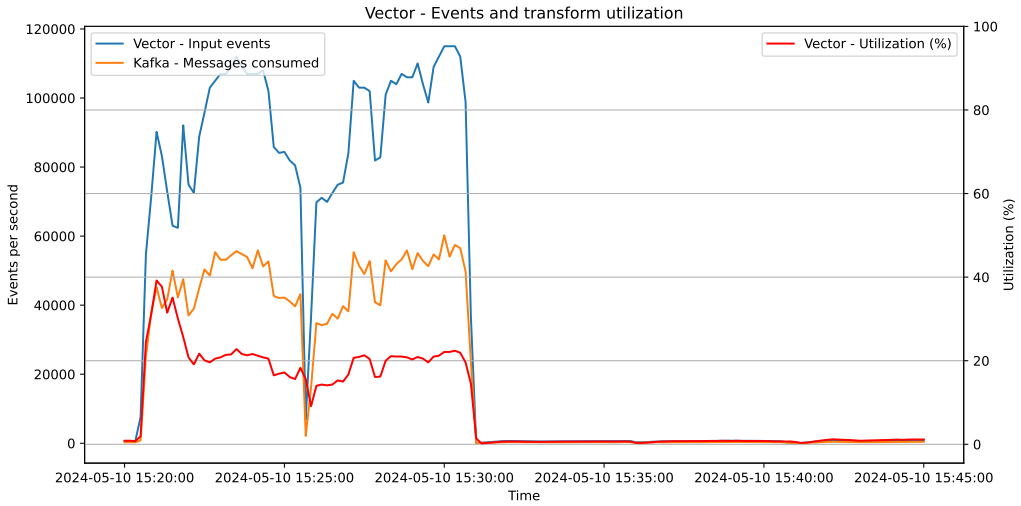


Figure 6.10: Vector - Events and transform utilization

In contrast to the results obtained with OpenSearch, the pipeline demonstrates significantly improved performance when utilizing the blackhole sink. Message consumption closely matches the incoming rate, consumer latency remains stable, and the "Geodata" transform operates well within its capacity. These findings suggest that the pipeline's core components can effectively handle the tested load when not hindered by the OpenSearch bottleneck.

[2]More specifically, we suspect that the data received from the enrichment table in use by the "Geodata" Transform is registered as an event.

Chapter 7

Discussion

7.1 Choice Of Tools and Technologies

7.1.1 Containerization vs. Virtualization

The team successfully implemented a functional Kubernetes environment but encountered a few challenges. As detailed in section 2.1.2, the decision to favor Kubernetes over virtual machines was influenced by Kubernetes' capability to horizontally auto-scale applications based on workload demands[1]. Additionally, a virtualized approach would require more overhead per instance and has an increased deployment time.

As explained in section 4.8.7, we encountered a challenge with auto-scaling the Kafka cluster. Although we could implement auto-scaling to effectively increase the number of available brokers based on usage, it would lead to an unbalanced workload across broker partitions. Consequently, auto-scaling is restricted to the Vector component. Testing revealed that this limitation does not significantly impact the pipeline, as the auto-scaling for the Vector component alone was sufficient.

Planning for scaling in Kubernetes necessitates allocating sufficient resources to accommodate internal service and pod scaling, which can potentially result in a significant amount of resources underutilized within the Kubernetes cluster. With virtualization, new VMs are created only as needed. Although VMs have a higher CPU, storage, and RAM overhead, these resources will only be taken up by deployed VMs, potentially leading to less reservation of underutilized resources.

A fully virtual-machine-based pipeline has limitations, especially for nodes dedicated to Kafka. This setup requires a "rolling restart" to implement changes, involving the manual restart of each Kafka broker sequentially.[2] Additionally, when scaling horizontally by adding new VMs, partitions need manual reassignment across the Kafka cluster, increasing management overhead.

Kubernetes operators such as Strimzi can alleviate these issues. Strimzi automates configuration changes and rolling restarts through Custom Resource Definitions (CRDs) for topics and brokers. Additionally, Cruise Control helps rebalance broker workloads, enhancing system adaptability during Kafka scaling operations[3].

Moreover, operating Vector within a virtual machine is feasible but introduces some of the issues seen in Kafka. Scaling the routing process will take longer since a new virtual machine must be provisioned and booted. This illustrates that using virtual machines for pipeline components that require autoscaling, will lead to increased scaling latency in response to fluctuating traffic volumes, as well as an ineffective use of resources.

In conclusion, considering that a log analytics pipeline experiences varying traffic throughout the

[1] https://kubernetes.io/docs/concepts/workloads/autoscaling/
[2] https://docs.confluent.io/platform/current/kafka/post-deployment.html
[3] Strimzi Documentation

day or across different seasons, integrating most parts of the pipeline within a container-based environment is advantageous. This approach facilitates timely scaling, which is crucial when receiving a large influx of logs, potentially preventing bottlenecks or at worst, crashes. Additionally, a container-based system simplifies scaling down the pipeline, thereby saving on resource-costs.

7.1.2 Managed vs. Self-deployed Kubernetes

Kubernetes has emerged as an ideal platform for pipeline implementation, leading to a decision favoring a managed Kubernetes deployment facilitated by OpenStack's Magnum component. Magnum simplifies the deployment process by allowing users to specify cluster requirements through a ClusterTemplate, negating the need for expertise in Kubernetes.[4] This contrasts with self-managed Kubernetes, which demands extensive operational efforts and technical skills for manual infrastructure management.[5]

Despite the advantages of managed Kubernetes in terms of setup ease, selecting a managed environment via OpenStack's Magnum introduced notable challenges, particularly when integrating with OpenStack services like the Cinder volume service. Issues with the Cinder CSI attacher, which failed to allocate volumes to pods properly, resulted in troubleshooting and workarounds, including manual updates to the CSI plugin within Magnum.

In contrast, deploying self-deployed Kubernetes, while initially demanding, offers greater flexibility and is not constrained by the limitations of Magnum. This flexibility enables a customized deployment of Kubernetes and utilizing OpenStack resources, tailored specifically to project requirements.

These experiences underscore a drawback of managed Kubernetes deployments: their dependency on the underlying infrastructure being up-to-date. Cloud services like Amazon Elastic Kubernetes Service (Amazon EKS) circumvent these complexities by providing managed Kubernetes without the challenges of maintaining an open-source orchestration system like OpenStack.[6]

Consequently, deploying a pipeline in a managed OpenStack environment requires regular updates to ensure compatibility with Kubernetes. Alternatively, a self-deployed Kubernetes environment may offer better control over deployment and reduce the risk of compatibility issues with OpenStack components. This comparison suggests that while managed Kubernetes deployments expedite setup, they also introduce dependencies that can complicate integration with existing infrastructure and services.

7.1.3 Event Streaming Platform

For the choice of what event streaming platform the group would implement, the group chose to go for Apache Kafka as it has an extensive list of adaptors [23] and is a part of the Apache Software Foundation, meaning that the software will always be available to use for free [7]. The group chose a Kubernetes operator for Apache Kafka instead of running the standard Apache Kafka software on Kubernetes for two main reasons.

Reason 1: Apache Kafka does not offer any official container images for Apache Kafka. This would mean the group must rely on third-party container images or build and maintain images from the Kafka source code. Neither of these options guarantees that the container image that would be used in the project would be stable and maintained on a long-term basis.

Reason 2: In addition to Strimzi being the most viable option for containerizing Kafka, the Strimzi project is also a Kubernetes operator[8], meaning that it also extends the automation capabilities of Kubernetes with its own set of custom resources. This means that the configuration

[4] https://docs.openstack.org/magnum/latest/user/index.html
[5] https://www.digitalocean.com/resources/article/unmanaged-vs-managed-kubernetes
[6] https://aws.amazon.com/eks/features/
[7] https://www.apache.org/free/#always-free
[8] https://kubernetes.io/docs/concepts/extend-kubernetes/operator/

of Kafka on Kubernetes is abstracted one layer further with the Kubernetes operator handling the application-specific configuration. Since Kafka is a complex system requiring several interconnected components to work correctly, using a Kubernetes operator drastically simplifies Kafka's deployment and configuration.

Since the group decided to use Strimzi, the group had to ensure that the project would be maintained and available as FOSS in the long term. The Strimzi Project was accepted to the Cloud Native Computing Foundation (CNCF) in August 2019 and was, in February 2024, officially a CNCF incubated project [24]. As part of the CNCF, the Strimzi Project must abide by the foundation's charter, which requires that all inbound and outbound code be licensed under the Apache 2.0 License, and all projects must be completely licensed under an OSI-approved open source license [6]. Given that the Strimzi Project is part of the CNCF and has a solid team of contributors and funding, the group has concluded that the team behind it is determined to keep the project free and open source on a long-term scale.

7.1.4 Log Processing And Routing Solution

At the beginning of the project, the NTNU SOC wanted us to consider implementing Apache NiFi as the log processing and routing solution. Apache NiFi is a widely adopted solution for this purpose, but the group opted to use Vector By Datadog for a multitude of reasons.

Since Apache NiFi's official administrators guide only has directions for installation on the Linux / Unix / macOS and Windows operating systems, the group had to find alternatives for configuring it on Kubernetes, for this, the group landed on the NiFikop Project [7]. While the NiFikop Project gave us a running configuration of Apache NiFi on Kubernetes, the group found four major issues that resulted in not going forward with NiFikop.

Reason 1: NiFikop is still in the early days of development and is therefore prone to potentially breaking updates according to their own Git repository [9]. Since our product is a POC infrastructure that the NTNU SOC intends to reference when developing their production infrastructure, the group did not want to introduce technologies that has the potential to break the pipeline after an update.

Reason 2: NiFikop has a relatively small development team currently working on the project with only 29 contributors as of May 2024 [21]. While Orange Open Source is still a major contributor to the project, the small team of contributors may result in a longer waiting time for new features and potential security updates.

Reason 3: Given the small development team and an empty list of adopters, the group is uncertain of the project's longevity. Prior attempts to make NiFi run on Kubernetes have been archived due to a lack of maintainers, notably the Cetic/NiFi helm chart [10].

Reason 4: The group had problems indexing logs into OpenSearch. Apache NiFi still recommends using the ElasticSearch sink to index logs into OpenSearch. Still, the group had trouble connecting to the OpenSearch nodes and could not index any logs. This could result from a configuration error on the group's part, but alternative log processing and routing solutions managed to index logs into OpenSearch without a problem.

Because of these reasons, the group elected to look for alternative log processing and routing solutions and chose Vector By Datadog. Vector is an alternative tool to Apache NiFi that offers many of the same features we want, and Vector addresses most of the main issues we have with Apache NiFi.

Firstly, The development team behind Vector maintains the Vector helm chart, meaning the group had fewer issues setting up and configuring Vector than NiFi on Kubernetes. For configuration, Vector supports declarative configuration natively with support for YAML, TOML, and JSON files, meaning that all the configurations for Vector can be kept in a centralized place with a single

[9] https://github.com/konpyutaika/nifikop?tab=readme-ov-file#issues-feature-requests-and-roadmap
[10] https://github.com/cetic/helm-nifi

source of truth.

Secondly, Vector is part of the DataDog software family, while still being open-sourced under the Mozilla Public License (MPL). This gives the project stability by having a reliable funding source, and the MPL license ensures that individuals and companies can use the software for any purpose. The Datadog team has stated that a big part of their focus over the last couple of years has been building up Vector at the core of their Observability Pipeline product [30]. Because of this, the group anticipates that the project will be maintained and supported in the long term.

The drawback of implementing Vector over Apache NiFi is the lack of a GUI for configuring and redirecting data flow. This ability in the log processor and router was one of the main reasons the NTNU SOC wanted to implement Apache NiFi. Still, because of the reasons listed, the group concluded that a solution with Vector would be more stable and future-proof than implementing Apache NiFi on Kubernetes. The group has chosen to leave the configuration of Apache NiFi in the source code so that in the event that the NiFikop project becomes a more viable option, the NTNU SOC has a point of reference for setup and configuration.

7.1.5 Hashicorp Transitions To Business Source License

The group has defined in section 2.1.1 that the solution must be comprised of only FOSS products, and in August 2023, Hashicorp changed its license on Terraform from the Mozilla Public License v2.0 (MPL 2.0) to the Business Source License (BSL). The Mozilla Public License v2.0 (MPL 2.0) is a license approved by the GNU Project as a FOSS license, but the new Business Source License (BSL) is not on the list of the approved licenses, but the group has chosen to implement Terraform as a core component of the solution because of the following reasons.

No licensing cost: While the new Business Source License is not an approved FOSS license by the GNU project, the current license of Terraform[11] allows for the use of Terraform in production environments like the NTNU SOC, as long as they don't offer the services to third parties. This means using Terraform for this product will not lead to additional costs for the NTNU SOC. Since one of the main motivators for redesigning the infrastructure was to reduce the current licensing costs for the NTNU SOC, the group chose to use Terraform even though it technically is not licensed under a FOSS license because it will not cost bring any licensing costs like the license change of ElasticSearch did.

Established and Robust: Given that the implementation and use of Terraform for this project will not lead to additional licensing costs, the group also decided to favor Terraform over other alternatives because Terraform is an established product and the leading infrastructure orchestration technology. While there are forks of Terraform, notably OpenTofu[12], these alternatives are relatively new and in the early days of further development.

[11] May 2024: https://github.com/hashicorp/terraform?tab=License-1-ov-file
[12] https://opentofu.org/

Chapter 8

Conclusion

8.1 Project Goal Achievements

The team aimed to achieve several learning objectives throughout this project, primarily focused on cloud-based infrastructure, GitOps methodologies, Kubernetes, and Infrastructure as Code (IaC) practices. The following section will discuss and evaluate the extent to which these goals were met, providing an overview of the project's successes and challenges.

8.1.1 Learning Goals

L1: Gain practical experience implementing and configuring cloud-based infrastructure.

This goal was successfully met. The team gained significant hands-on experience with working on OpenStack, setting up virtual machines, network components, and storage resources with IaC to support the pipeline.

L2: Acquire skills in leveraging GitOps practices to optimize and automate infrastructure workflows.

The team successfully adopted GitOps principles, using GitLab as a single source of truth for infrastructure and configuration management. The implementation of FluxCD automated the deployment and synchronization of Kubernetes resources.

L3: Develop the team's understanding of Kubernetes principles, particularly in relation to scaling and managing cloud-native applications.

The team gained a solid understanding of central Kubernetes principles, particularly in the areas of container orchestration and resource management. We demonstrated proficiency in defining Kubernetes manifests using YAML, deploying applications using Helm charts (4.8.2), and managing resources within the cluster. Although the complexities of autoscaling posed challenges within the limited project scope, we successfully explored alternative scaling strategies (7.1.1). Our experience with Kubernetes introduced the members to its power and flexibility for managing complex, cloud-native applications.

L4: Learn to design and implement multi-component software solutions as Infrastructure as Code.

Given that the group has successfully made a solution where multiple applications are interconnected and are left with a working POC, the group considers this goal to be achieved.

8.1.2 Effect Goals

E1: Simplify data-flow management by implementing flow-based programming tools.

The NTNU SOC wanted a flow-based programming solution for their data-flow pipelines to redirect dataflow through a GUI. This goal has not been met since the group decided to go for Vector over NiFikop, as outlined in section 7.1.4. The group accepts that this goal has not been met as we concluded that the drawbacks of not implementing NiFikop did not outweigh the benefits of implementing Vector. The group also kept the configuration of NiFikop in the source code that the NTNU SOC has access to, so should they opt to use NiFikop instead, they have a base config they can expand upon.

E2: Automate and streamline infrastructure management with GitOps methodologies.

The group implemented GitOps technologies and adopted GitOps methodologies early on in the project, and they have proven useful when performing tasks such as redeploying the Kubernetes cluster and performing configuration changes. The group also created deployment pipelines for OpenSearch, which can serve as a POC reference for creating deployment pipelines for the rest of the infrastructure. The group considers this goal to be achieved.

E3: Reduce licensing costs for NTNU SOC.

Since one of the primary motivators for the task by the NTNU SOC was to reduce their current licensing costs, the group has only chosen software that is either licensed under a FOSS license or is free to use in the context of the work being done by the NTNU SOC. Since there are no associated licensing costs with the new solution, as opposed to the old solution, the group considers this goal to be achieved.

8.1.3 Result Goals

R1: Have a POC solution that can collect a log from the source, process it, and index it for long-term storage in OpenSearch.

The group has demonstrated in chapter 6 that the solution is capable of ingesting logs, processing them, routing them, and indexing them into OpenSearch. The group considers this goal to be achieved.

R2: Have the POC-solution be defined and deploy-able through IaC, and adheres to GitOps principles.

The group implemented GitOps technologies and methodologies early on in the project and has actively developed the product while utilizing GitOps principles. As a result, the solution requires minimal operator involvement to redeploy the infrastructure and can be further automated with deployment pipelines. Since the entire solution is written declaratively and uses Git as a single source of truth, the group considers this goal to be achieved.

R3: Present a log analytics pipeline capable of dynamically scaling in response to fluctuating traffic volumes.

This goal was partially met. As demonstrated in 6, the processing and routing components of the pipeline dynamically scale in response to fluctuating traffic volumes. However, due to Kafka's stateful nature, the group was unable to implement a meaningful solution for automatically scaling the Kafka Cluster.

8.2 Further Work

8.2.1 Replace Terraform With OpenTofu

Since Terraform is no longer licensed under a FOSS license, if the NTNU SOC wants to transition away from using Terraform, OpenTofu looks like the most viable option at the time of writing. OpenTofu is a fork of Terraform, licensed under the GNU Project approved Mozilla Public License, version 2.0 (MPL-2.0), and is a part of the Linux foundation. They aim to make a drop-in replacement for Terraform that is backward compatible with your current Terraform code and will continue to be licensed as FOSS.

8.2.2 Product Hardening

Since the group has focused on creating a POC infrastructure and pipeline, the cluster and components have not been configured to follow best practices concerning security. Therefore, to transition from a POC to a production-ready solution, further work would include hardening of the solution and implementation of security best practices.

8.2.3 Expand Deployment Pipelines

The group created deployment pipelines for the OpenSearch cluster but did not allocate time and resources to creating pipelines for the rest of the solutions. Since the base for the deployment pipelines has been made, creating pipelines for the rest of the infrastructure is relatively straightforward.

Bibliography

[1] National Security Agency, Cybersecurity and Infrastructure Security Agency. *Defending Continuous Integration/Continuous Delivery Environments*. Tech. rep. National Security Agency, June 2023. URL: https://media.defense.gov/2023/Jun/28/2003249466/-1/-1/0/CSI_DEFENDING_CI_CD_ENVIRONMENTS.PDF.

[2] National Security Agency, Cybersecurity and Infrastructure Security Agency. *Kubernetes Hardening Guide Version 1.2*. Tech. rep. National Security Agency, Aug. 2022. URL: https://media.defense.gov/2022/Aug/29/2003066362/-1/-1/0/CTR_KUBERNETES_HARDENING_GUIDANCE_1.2_20220829.PDF.

[3] A. Arildset et al. 'Securing the Software Development Life Cycle'. Bachelor Thesis. NTNU, Norwegian University of Science and Technology, May 2023.

[4] Joe Beda Brendan Burns and Kelsey Hightower. *Kubernetes Up & Running: Dive into the Future of Infrastructure*. 2nd ed. O'Reilly, 2019.

[5] *Charts*. The Linux Foundation. URL: https://helm.sh/docs/ (visited on 15th May 2024).

[6] *Cloud Native Computing Foundation ("CNCF") Charter*. Section 11. Cloud Native Computing Foundation ("CNCF"). URL: https://github.com/cncf/foundation/blob/main/charter.md (visited on 7th May 2024).

[7] *Docs*. NiFiKop. URL: https://konpyutaika.github.io/nifikop/docs/1_concepts/1_start_here (visited on 9th Apr. 2024).

[8] Stepping up for a truly open source Elasticsearch. *Carl Meadows, Jules Graybill, Kyle Davis, Mehul Shah*. URL: https://aws.amazon.com/blogs/opensource/stepping-up-for-a-truly-open-source-elasticsearch/ (visited on 19th May 2024).

[9] *fluxcd.io*. Cloud Native Computing Foundation. URL: https://fluxcd.io/ (visited on 28th Feb. 2024).

[10] *git.gvk.idi.ntnu.no - Log Analytics Repository*. NTNU. URL: https://git.gvk.idi.ntnu.no/bachelor/2024/log-analytics.

[11] *Gitlab Runner*. Gitlab docs. URL: https://docs.gitlab.com/runner/ (visited on 9th May 2024).

[12] Michael Hill. 'Hard-coded secrets up 67% as secrets sprawl threatens software supply chain'. In: *CSO Online* (Mar. 2023). URL: https://www.csoonline.com/article/574687/hard-coded-secrets-up-67-as-secrets-sprawl-threatens-software-supply-chain.html.

[13] Amazon Web Services Inc. *What is OpenSearch?* URL: https://aws.amazon.com/what-is/opensearch/ (visited on 19th May 2024).

[14] Center for Internet Security. *CIS Kubernetes Benchmark V1.9.0*. Tech. rep. Center for Internet Security, Mar. 2024. URL: https://learn.cisecurity.org/l/799323/2024-03-28/4tkz99.

[15] *Introduction*. Apache. URL: https://kafka.apache.org/intro (visited on 19th Mar. 2024).

[16] *Introduction to Ansible*. Ansible project contributors. URL: https://docs.ansible.com/ansible/latest/getting_started/introduction.html (visited on 16th May 2024).

[17] *Kafka Rebalancing*. Redpanda. URL: https://redpanda.com/guides/kafka-performance/kafka-rebalancing (visited on 21st May 2024).

[18] *Kubernetes Documentation*. Cloud Native Computing Foundation. URL: https://kubernetes.io/docs/concepts/overview/ (visited on 23rd Feb. 2024).

[19] Grafana Labs. *Grafana k6 documentation*. URL: https://grafana.com/docs/k6/latest/ (visited on 19th May 2024).

[20] Kief Morris. *Infrastructure as Code, Dynamic Systems for the Cloud Age*. O'Reilly Media, Inc, 2021.

[21] *Nifikop*. konpyutaika. URL: https://github.com/konpyutaika/nifikop (visited on 8th May 2024).

[22] *NTNU SOC - Security Operations Centre*. NTNU. 2023. URL: https://www.ntnu.edu/web/adm-it/ntnu-soc (visited on 17th Jan. 2024).

[23] *POWERED BY*. Apache Kafka. URL: https://kafka.apache.org/powered-by (visited on 10th May 2024).

[24] *Projects, Strimzi*. Cloud Native Computing Foundation ("CNCF"). URL: https://www.cncf.io/projects/strimzi/ (visited on 7th May 2024).

[25] Prometheus-Community. *Helm-Charts*. https://github.com/prometheus-community/helm-charts/blob/main/charts/kube-prometheus-stack/values.yaml. 2024.

[26] Rohit Salecha. *Practical Gitops*. Apress Berkeley, CA, 2022.

[27] *Security Operations Center (SOC)*. IBM. 2024. URL: https://www.ibm.com/topics/security-operations-center (visited on 22nd Jan. 2024).

[28] *Strimzi*. Ckoud Native Computing Foundation. URL: https://www.cncf.io/blog/2024/02/08/strimzi-joins-the-cncf-incubator/ (visited on 30th Apr. 2024).

[29] *Studenter i universitets- og høgskoleutdanning*. Statistisk Sentralbyrå. 2023. URL: https://www.ssb.no/utdanning/hoyere-utdanning/statistikk/studenter-i-universitets-og-hogskoleutdanning (visited on 5th Apr. 2024).

[30] *Vector*. Datadog Open Source Hub. URL: https://opensource.datadoghq.com/projects/vector/ (visited on 9th May 2024).

[31] Abhishek Verma et al. 'Large-scale cluster management at Google with Borg'. In: *Proceedings of the European Conference on Computer Systems (EuroSys)*. Bordeaux, France, 2015.

[32] *What is Kanban? Here's what your Agile team needs to know*. Asana. 2022. URL: https://asana.com/resources/what-is-kanban (visited on 23rd Jan. 2024).

[33] *What is terraform?* HashiCorp. URL: https://developer.hashicorp.com/terraform/intro (visited on 9th May 2024).

[34] *What is vector?* Datadog. URL: https://vector.dev/docs/about/what-is-vector/ (visited on 16th May 2024).

[35] *Why Are Cloud-Native Applications Necessary?* SolarWinds. 2024. URL: https://www.papertrail.com/solution/tips/cloud-native-applications-and-log-management-best-practices/ (visited on 25th Jan. 2024).

Appendix

Appendix A

Project Plan

A.1 Introduction

The team has assembled a project plan as a road-map throughout the project phase. The plan elaborates on the client's reason for the project and outlines the desired result for both the client and the team. It details everything from the project's scope, to the teams structure and project methodologies.

A.2 Goals and Restrictions

A.2.1 Background

NTNU Security Operations Center, from now referred to as NTNU SOC is the digital security and emergency response center for NTNU and covers the Trondheim, Gjøvik and Aalesund campuses. NTNU SOC is the official point of contact for all security incidents and provides services like intrusion detection, technical security analysis and incident management [22]. A Security Operations Center (SOC) is a team that focuses on monitoring infrastructure and devices to detect and act upon cybersecurity events. Their main tasks are to analyze threats and threat data, respond to incidents and take necessary preventative measurements to prevent unwanted activity on the infrastructure [27].

Security Operation Centers uses logs for a variety of analysis purposes, so having an efficient and scalable log collection and processing pipeline is essential to making sure the data-set that the security analysts are working on is complete and up to date. With more and more services being created for cloud native systems using containers and container-clusters, the complexity of log gathering and analysis has increased. Some of the challenges with log gathering from cloud native systems are more logging from the different micro-services, differences in logs from e.g. a Docker container versus a Linux based operating system, and the fact that all log files are lost when a container execution is halted [35]. Demonstrating a proof of concept solution for scalable log collection & processing will serve as a foundation for the further development of NTNU SOC, and can be beneficial for other organizations looking to implement a more scalable solution.

A.2.2 Project Goals

The project goals are inspired by and partially overlapping with the bachelor-thesis of A. Arildset et al.[3]

The project goals for this project will be split into three sections: Effect goals, Result goals, and Learning goals. Effect goals are wanted improvements for the client as a result of the project. Result goals are objectives directly related to the project's outcome, defining what the team aims to accomplish by the project's end. Learning goals pertain to the skills, knowledge, and experience that the team aims to have acquired by the time the project concludes.

Effect goals

1. The new solution should be able to scale up & down to meet demands in times of higher activity, and save resources in times of lower activity.
2. Operators should be able to change data-flow and modify how the data is aggregated while the system is running.
3. The new system should reduce current licensing-costs for NTNU SOC

Result goals

1. Have a proof of concept system that is able to collect and aggregate logs before feeding it to long-term storage in OpenSearch.
2. Have a system that is easily scalable for log gathering, processing, storing and utilizing.

Learning goals

1. The team aims to have learned more about log gathering and how to process logs for real world analysis applications.

2. Acquire knowledge on how to design systems with scalability as a main focus.

A.2.3 Framework

Timeframe

- Deadline for signing and delivering the project plan: 1. February 2023

- Deadline for delivering the finished project: 21. May 2023

A.3 Scope

A.3.1 Problem Statement

NTNU SOC is moving to a cloud-native, open-source system for log collection and monitoring, aiming to reduce costs from their current Elasticsearch-based system. Notably, NTNU SOC has largely developed the Search and Indexing section, including Opensearch nodes for storage and indexing. Our project will integrate with this existing infrastructure, utilizing the code shared by NTNU SOC.

Our project's main goal is to develop and integrate the 'unified data transform and transfer bus' and 'data ingest' layers of the infrastructure. These components will be integrated to work in conjunction with the Search and Indexing infrastructure. A key task is developing and configuring the services required by the aforementioned layers, and deploying them via Kubernetes. This will include containerizing services related to aggregation, processing, normalization, and enrichment. NTNU SOC requests the use of Apache Kafka and Nifi for log- aggregation and routing. If these cannot be effectively containerized and deployed in Kubernetes, they will be handled externally.

This approach allows for scalable, flexible data handling to meet NTNU SOC's needs. By using Infrastructure as Code (IaC) for deployment, we aim for efficient, reproducible infrastructure management. Containerization is crucial for scalability and maintainability. The choice of Kubernetes is due to its capacity for managing containerized applications, essential for automatic scaling, load balancing, and recovery. In summary; the project strives to create a resilient, efficient, and adaptable log-processing pipeline for NTNU SOC.

A.4 Project Organization

A.4.1 Roles and area of responsibility

Team coordinator

The Team Coordinator ensures the smooth operation of the group. They are responsible for orchestrating the workflow among members, ensuring that tasks are evenly distributed and aligned with each member's strengths and expertise.

Key responsibilities:

- Coordinate and facilitate workloads between group members, ensuring a harmonious and efficient workflow.
- Lead meetings according to a pre-determined itinerary.
- Mediate and resolve internal issues and conflicts, acting as a neutral party to maintain group harmony.
- Act as a tiebreaker in tied internal votes, ensuring decision-making continues to move forward.
- Implement and oversee the enforcement of group rules, including the issuing of fines for broken rules.
- May sign on behalf of other group members for documents regarding the bachelor´s thesis

Quality assurance coordinator

The Quality Assurance Coordinator is tasked with safeguarding the standards of the project's output. Their focus is on ensuring the clarity, coherence, and overall quality of the final report, code-base and supporting documentation.

Key responsibilities:

- Oversee the readability and cohesion of the final report, ensuring it meets academic and professional standards.
- Guarantee high-quality project documentation, including detailed and accurate records of the project's progress and outcomes.
- Maintain an intuitive and organized file and code structure within the code-base.

Administrative coordinator

The Administrative Coordinator is the organizational hub of the team, focusing on the logistics of meeting coordination and documentation. They ensure that all administrative tasks are executed efficiently and effectively.

Key responsibilities:

- Organize and schedule meetings, ensuring all members are informed and prepared.
- Prepare and distribute meeting itineraries, contributing to focused and effective meetings.
- Record and circulate meeting minutes, providing a clear and concise record of discussions and decisions.
- Document and disseminate reports from meetings with clients and academic supervisors, ensuring transparency and clear communication.

Document coordinator

The Document Coordinator is responsible for the management and organization of all project-related documents, ensuring that important information is readily accessible and well-organized.

Key responsibilities:

- Ensure timely storage of documents in their designated locations, facilitating easy access and organization.
- Guarantee that documents are distributed as planned and are readily available to all group members.
- Oversee LaTeX templating for the final report, ensuring a professional and consistent format.

A.4.2 Routines

Collaborative work-sessions

To foster a collaborative and productive environment, weekly work-sessions are **mandatory** for all group members. These sessions aim to ensure that ideas are exchanged effectively and that the group progresses cohesively.

Client meetings

Our project involves regular, weekly meetings with our client. The primary focus of these meetings is to keep the client updated on our progress, discuss and receive feedback on our functional choices, and address any new requirements or changes that may arise.

Academic supervisor meetings

In order to maintain academic rigor in our work, we will conduct bi-weekly or weekly meetings with our academic supervisors. These meetings are intended to ensure that our project aligns with academic standards and objectives.

Code review

As our code is intended to be open-source and available to be used, quality assurance in our coding process is of utmost importance. Therefore, all commits to the production branch must undergo a thorough review. This review must be performed by a group member who did not participate in the specific update, ensuring impartiality and a fresh perspective on the work done.

Time expenditure and registration

Each group member is expected to dedicate a minimum average of 30 hours per week to the project. To track and manage this commitment, all time spent on any project-related activity must be recorded in the 'Timeføring.xlsx' sheet, available on the group's Teams channel.

Communication platforms

For effective communication, our team will utilize Microsoft Teams for formal communication and Discord for informal interactions. This dual-platform approach is designed to separate and streamline our professional and casual communications, ensuring clarity and organization.

Internal social gatherings

Recognizing the importance of a positive team dynamic and mental health of the group members, we will arrange for non-project related social activities on a weekly or bi-weekly basis.

A.4.3 Group rules

Celebration fund

To encourage adherence to the group's rules and add a positive spin to rule enforcement, any fines incurred for breaking rules are contributed to a 'Celebration Fund'. This fund will be used to finance a celebration for the team upon successful delivery of our project report.

Meetings with clients and academic supervisors

Attendance at meetings with clients and academic supervisors is mandatory. A fine of 300 NOK will be levied for failing to attend these meetings without prior notification. Additionally, tardiness exceeding 10 minutes will require the late member to offer each group member a non-alcoholic drink or snack of their choice, capped at a cost of 30 NOK, as a gesture of apology and commitment to punctuality.

Collaborative work-sessions

In-person attendance at the collaborative work-sessions is mandatory. In the case of unnotified absence, a fine of 150 NOK will be imposed. Should a group member be more than 30 minutes late, they are expected to offer each group member a non-alcoholic drink or snack, with a maximum value of 30 NOK, as a form of apology for the delay.

Task progression

Timely completion of tasks is fundamental to the project's success. If a member is unable to complete a task within the expected time-frame, they are required to inform the team and seek assistance or guidance. Failure to communicate and manage task progression in a timely manner may result in a fine of 150 NOK.

Internal conflicts

Should internal conflicts arise, they should be resolved amicably between the parties involved. If a resolution cannot be reached, the Team Coordinator will intervene as a mediator. Should the conflict remain unresolved after the Coordinator's intervention, the matter will be escalated to the appropriate authorities for further action. No fines are associated with internal conflicts, emphasizing the importance of open communication and mutual respect in resolving disagreements.

A.5 Planning, followup and reporting

A.5.1 Project Management Methodology

The NTNU-SOC has shown a great willingness to accept different technologies and solutions for their problem. They have given us the freedom to choose our methods with only a few requirements. Because of this, we have selected an agile working method. This approach allows us to experiment with various technologies and solutions, while continuously receiving feedback from our client.

When deciding on a specific methodology, we considered several factors:

1. The client's preference for us to adopt a DevOps approach, maintaining transparency in our project and the flexibility to add new features if we are ahead of schedule.
2. The fact that none of our team members have prior experience with the requested technologies.
3. The varied work schedules of our team members due to part-time jobs.

As a result, we decided against frameworks like Scrum, which require daily meetings and short-term planning, as it might not be possible with members potentially unavailable for extended periods of time. Additionally, estimating the duration of sprints was challenging due to our lack of experience in this field. This led us to choose the Kanban framework, which allows us to limit the amount of work in progress and focus on completing tasks sequentially.

Kanban

Kanban is an Agile management method built on a philosophy of continuous improvement, where work items are "pulled" from a product backlog into a steady flow of work. The framework is applied using Kanban boards—a form of visual project management.[32] Our team plans to implement this using GitLab's issue board, where each issue represents a distinct task.

Weekly in-person meetings will be conducted to review our progress, collaborate on task resolution, and plan our individual assignments for the upcoming week. which will ensure that all team members are fully informed and aligned with our goals. This approach will enable us to prioritize critical tasks and, if necessary, reassign tasks among team members to accommodate availability. Additionally, we will have weekly status meetings with our client to align with project goals and incorporate any requested changes or additional features.

Followup

- The team will convene weekly at the NTNU campus for collaborative work sessions, with the specific date and time determined at the previous meeting.
- Meetings with our client will be scheduled weekly or biweekly, as required, to ensure ongoing alignment and progress updates.
- The team coordinator will promptly organize additional meetings in response to any critical challenges or urgent issues that require immediate attention.

A.6 Organization of quality assurance

A.6.1 Documentation

The bachelor thesis will be written and produced using Overleaf, an online LaTeX editor. This choice ensures consistent formatting and professional presentation of the final document.

For project coordination, we will utilize the Microsoft Office suite to create and manage documents such as meeting minutes, time-usage logs, and resource budgets. These files will be stored in our shared team channel on Microsoft Teams, guaranteeing high availability and collaborative accessibility.

Our code-base, including the developed software and scripts, will be stored and version-controlled in the GitLab repository git.gvk.idi.ntnu.no/fredrlst/log-analytics-bachelor-project. We will employ a feature-branch strategy for version control, where new features and bug fixes are developed in separate branches and merged into the main branch upon completion and review.

In addition, this repository will house the documentation for our infrastructure stack. Adhering to the principle of "Documentation as Code," all documentation will be written in Markdown. This ensures that documentation updates are trackable and reviewable through Git. We will structure the documentation according to the following example:

Repository-root

- README.md (Project Overview and Setup Instructions)
- OpenSearch
  - OpenSearch.md (Installation, Configuration, and Usage)
- Kafka
  - kafka.md (Integration Points, Data Flows)
- fluentbit
  - fluentbit.md (Logging Conventions, Output Formats)

A.6.2 Plan for testing and inspection

Given the cloud-native nature of our project, a robust CI/CD (Continuous Integration/Continuous Deployment) workflow is crucial. In our GitLab repository, we will set up automated pipelines to run a variety of tests, ensuring the reliability and stability of our code. These tests will include:

- **Unit Tests**: To validate individual components in isolation.
- **Integration Tests**: To ensure different modules work together seamlessly.
- **System Tests**: To verify the complete and integrated software product.

We will structure our CI/CD pipeline into three primary stages:

- **Testing**: For running automated tests on new code submissions.
- **Staging**: A pre-production environment for final verification.
- **Production**: Where validated changes go live to end-users.

Each pull request to the production branch will require approval from at least one project member not involved in its development, ensuring an additional layer of quality control.

A.6.3 Risk Analysis

Effective risk management is crucial for the successful completion of our project. This subsection outlines potential risks and proposes strategies to mitigate them, ensuring the project stays on track.

Consequence						
	Catastrophic					
	Major		4			
	Moderate		5	2	1	
	Minor			6	3	
	Insignificant					
		Rare	Unlikely	Possible	Likely	Certain
Probability						

Table A.1: Risk Matrix

Risk 1: External Commitments

Description	Team members may have external commitments, such as work, renovations, or long commutes, that could limit their availability and resources.
Probability	Likely
Consequence	Moderate
Overall risk	Serious

Mitigation Strategy: Regularly schedule check-ins to assess each member's availability and redistribute tasks as needed to accommodate fluctuating schedules.

Risk 2: Scope creep

Description	Our high motivation and adaptability, coupled with big aspirations, might lead to taking on more than we can feasibly handle.
Probability	Possible
Consequence	Moderate
Overall risk	Moderate

Mitigation Strategy: Establish clear project boundaries and objectives at the outset. Regularly review project scope against set milestones to ensure alignment.

Risk 3: Nit-picking

Description	Perfectionist tendencies in some team members may lead to spending excessive time optimizing code in areas where it is not necessary.
Probability	Likely
Consequence	Minor
Overall risk	Moderate

Mitigation Strategy: Implement a policy where optimization efforts are time-boxed. For instance, allocate a specific amount of time for refining code after primary functionalities are achieved. This approach helps to limit over-optimization.

Risk 4: Unresolvable Interpersonal Conflict

Description	While unlikely, there exists a non-zero chance of interpersonal conflicts that cannot be easily resolved.
Probability	Unlikely
Consequence	Major
Overall risk	Moderate

Mitigation Strategy: Establish a conflict resolution protocol at the project's onset. Include a neutral mediator role, possibly a mentor, to facilitate resolution.

Risk 5: Long-term Illness

Description	Key project members may suffer from long-term illnesses, significantly delaying project milestones.
Probability	Unlikely
Consequence	Moderate
Overall risk	Moderate

Mitigation Strategy: Implement a system where key tasks are not solely dependent on one individual. Having at least two members familiar with each critical task can reduce the impact if one member becomes unavailable.

Risk 6: Sunk Cost Fallacy

Description	The team might continue to invest in a particular approach or component due to the significant resources already expended, even if it's no longer viable.
Probability	Possible
Consequence	Minor
Overall risk	Moderate

Mitigation Strategy: Regular project evaluations to assess the viability and effectiveness of ongoing efforts. Encourage open discussions and be ready to pivot strategies if necessary.

A.7 Plan for execution

A.7.1 Gannt

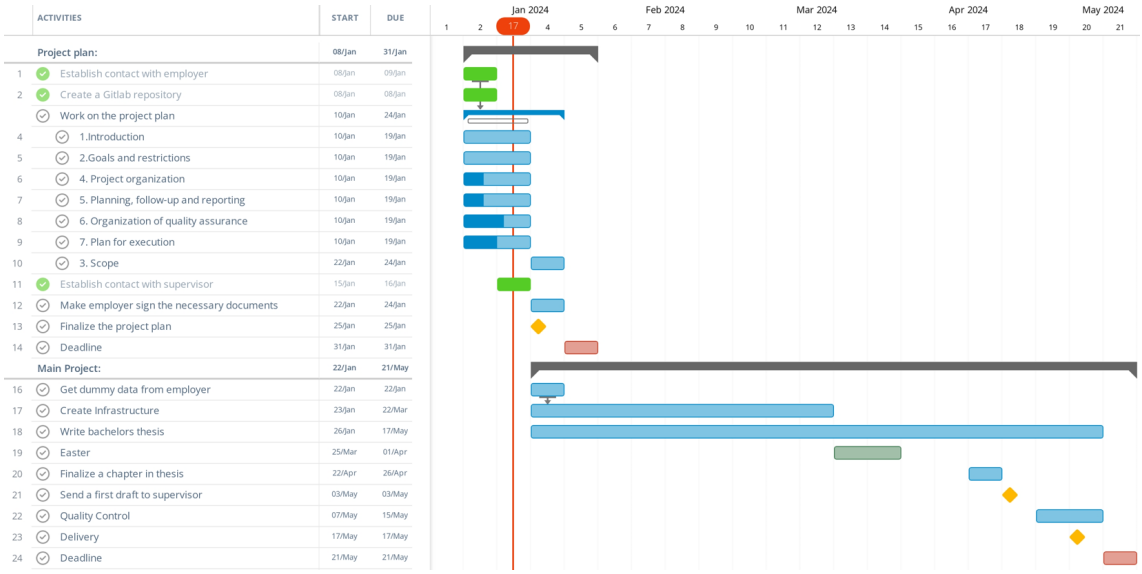


Figure A.1: Gannt chart

A.8 Signatures

I, the undersigned, hereby acknowledge and agree to the terms outlined in this document.

Karl-Henrik Horve	Date
Marcus Eugen Brockstedt Mathisen	Date
Fredrik Leonard Stenersen	Date
Bjørn Kristian Strand	Date

Appendix B

OpenSearch Ansible-Playbook modification

This appendix details the changes made to the Opensearch ansible-playbooks sourced from the https://github.com/opensearch-project/ansible-playbook/tree/main git repository.

B.1 Original opensearch security.yml

```
1  ##########################################
2  ##### PLAYBOOK MODIFICATION EXAMPLE #####
3  ##########################################
```

Figure B.1: Playbook modification example

The below yaml code is the original 'security.yml', that we modified for use in our project. Comments following the format exemplified by Figure B.1 signify sections deleted in the playbook.

```
1  ---
2  ##############################################
3  ##### DELETED PLAYBOOK SECTION 1 START #####
4  ##############################################
5
6  ## Here we are going to use self-signed certificates for Transport (Node-Node communication) & REST
   ↪  API layer
7  ## Using searchguard offline TLS tool to create node & root certificates
8  - name: Security Plugin configuration | Force remove local temporary directory for certificates
   ↪  generation
9    local_action:
10     module: file
11     path: /tmp/opensearch-nodecerts
12     state: absent
13   run_once: true
14   become: false
15   when: iac_enable
16
17 - name: Security Plugin configuration | Create local temporary directory for certificates
   ↪  generation
18   local_action:
19     module: file
```

```
20      path: /tmp/opensearch-nodecerts
21      state: directory
22    run_once: true
23    register: configuration
24    become: false
25
26  - name: Security Plugin configuration | Download certificates generation tool
27    local_action:
28      module: get_url
29      url:
        ↪  https://search.maven.org/remotecontent?filepath=com/floragunn/search-guard-tlstool/1.5/search-guard-tlstool-1.5.
30      dest: /tmp/opensearch-nodecerts/search-guard-tlstool.tar.gz
31    run_once: true
32    when: configuration.changed
33    become: false
34
35  - name: Security Plugin configuration | Extract the certificates generation tool
36    local_action: command chdir=/tmp/opensearch-nodecerts tar -xvf search-guard-tlstool.tar.gz
37    run_once: true
38    when: configuration.changed
39    become: false
40
41  - name: Security Plugin configuration | Make the executable file
42    local_action:
43      module: file
44      dest: /tmp/opensearch-nodecerts/tools/sgtlstool.sh
45      mode: a+x
46    run_once: true
47    when: configuration.changed
48    become: false
49
50  - name: Security Plugin configuration | Prepare the certificates generation template file
51    local_action:
52      module: template
53      src: tlsconfig.yml
54      dest: /tmp/opensearch-nodecerts/config/tlsconfig.yml
55    run_once: true
56    when: configuration.changed
57    become: false
58
59  - name: Security Plugin configuration | Generate the node & admin certificates in local
60    local_action:
61      module: command /tmp/opensearch-nodecerts/tools/sgtlstool.sh -c
        ↪  /tmp/opensearch-nodecerts/config/tlsconfig.yml -ca -crt -t
        ↪  /tmp/opensearch-nodecerts/config/
62    run_once: true
63    when: configuration.changed
64    become: false
65
66  ##########################################
67  ##### DELETED PLAYBOOK SECTION 1 END #####
68  ##########################################
69
70  - name: Security Plugin configuration | IaC enabled - Check certificate
71    when: iac_enable
72    block:
73      - name: Security Plugin configuration | Check cert exists
74        ansible.builtin.stat:
75          path: "{{ item }}"
76          get_attributes: false
77          get_checksum: false
78          get_mime: false
79        register: cert_stat_result
80        with_items:
81          - "{{ os_conf_dir }}/root-ca.pem"
82          - "{{ os_conf_dir }}/root-ca.key"
```

```
        - "{{ os_conf_dir }}/{{ inventory_hostname }}.key"
        - "{{ os_conf_dir }}/{{ inventory_hostname }}.pem"
        - "{{ os_conf_dir }}/{{ inventory_hostname }}_http.key"
        - "{{ os_conf_dir }}/{{ inventory_hostname }}_http.pem"
        - "{{ os_conf_dir }}/admin.key"
        - "{{ os_conf_dir }}/admin.pem"

    - name: Security Plugin configuration | Set fact. The initial value "Don't update certs"
      ansible.builtin.set_fact:
        force_update_cert: false

    - name: Security Plugin configuration | Set fact. Update certificates if at least one certificate is not found
      ansible.builtin.set_fact:
        force_update_cert: true
      with_items: "{{ cert_stat_result.results }}"
      when: item.stat.exists == False

    - name: Security Plugin configuration | Show the force_update_cert setting
      ansible.builtin.debug:
        msg: "force_update_cert: {{ force_update_cert }}"

    - name: Security Plugin configuration | Count force_update_cert nodes
      ansible.builtin.set_fact:
        force_update_cert_nodes_count: "{{ hostvars | dict2items |
        selectattr('value.force_update_cert', 'defined') |
        rejectattr('value.force_update_cert', 'equalto', false) |
        map(attribute='value.force_update_cert') | list | length }}"

    - name: Security Plugin configuration | Show the force_update_cert_nodes_count setting
      ansible.builtin.debug:
        msg: "force_update_cert_nodes_count: {{ force_update_cert_nodes_count }}"

    - name: Security Plugin configuration | Do need to update certificates
      ansible.builtin.debug:
        msg: "Need to update certificates..."
      when: force_update_cert_nodes_count | int > 0

- name: Security Plugin configuration | IaC disabled - Count force_update_cert nodes
  ansible.builtin.set_fact:
    force_update_cert_nodes_count: 0
  when: not iac_enable

- name: Security Plugin configuration | Copy the node & admin certificates to opensearch nodes if at least one certificate is not found on at least one server
  ansible.builtin.copy:
    src: "/tmp/opensearch-nodecerts/config/{{ item }}"
    dest: "{{ os_conf_dir }}"
    mode: 0600
  with_items:
    - root-ca.pem
    - root-ca.key
    - "{{ inventory_hostname }}.key"
    - "{{ inventory_hostname }}.pem"
    - "{{ inventory_hostname }}_http.key"
    - "{{ inventory_hostname }}_http.pem"
    - admin.key
    - admin.pem
  when: (configuration.changed and not iac_enable) or (iac_enable and force_update_cert_nodes_count | int > 0)

- name: Security Plugin configuration | Copy the security configuration file 1 to cluster
  ansible.builtin.blockinfile:
    block: "{{ lookup('template', 'templates/security_conf.yml') }}"
    dest: "{{ os_conf_dir }}/opensearch.yml"
    backup: true
```

```
143       insertafter: EOF
144       marker: "## {mark} OpenSearch Security common configuration ##"
145     when: configuration.changed or iac_enable
146
147   - name: Security Plugin configuration | Copy the security configuration file 2 to cluster
148     ansible.builtin.blockinfile:
149       block: "{{ lookup('file', '/tmp/opensearch-nodecerts/config/{{ inventory_hostname
        ↪  }}_elasticsearch_config_snippet.yml') }}"
150       dest: "{{ os_conf_dir }}/opensearch.yml"
151       backup: true
152       insertafter: EOF
153       marker: "## {mark} opensearch Security Node & Admin certificates configuration ##"
154     when: configuration.changed or iac_enable
155
156   - name: Security Plugin configuration | Create security plugin configuration folder
157     ansible.builtin.file:
158       dest: "{{ os_sec_plugin_conf_path }}"
159       owner: "{{ os_user }}"
160       group: "{{ os_user }}"
161       mode: 0700
162       state: directory
163     when: configuration.changed or iac_enable
164
165   - name: Security Plugin configuration | Copy the security configuration file 3 to cluster
166     ansible.builtin.template:
167       src: security_plugin_conf.yml
168       dest: "{{ os_sec_plugin_conf_path }}/config.yml"
169       backup: true
170       owner: "{{ os_user }}"
171       group: "{{ os_user }}"
172       mode: 0600
173       force: true
174     when: auth_type == 'oidc' or copy_custom_security_configs
175
176   - name: Security Plugin configuration | Prepare the opensearch security configuration file
177     ansible.builtin.command: sed -i 's/searchguard/plugins.security/g' {{ os_conf_dir
        ↪  }}/opensearch.yml
178     when: configuration.changed or iac_enable
179
180   - name: Security Plugin configuration | Set the file ownerships
181     ansible.builtin.file:
182       dest: "{{ os_home }}"
183       owner: "{{ os_user }}"
184       group: "{{ os_user }}"
185       recurse: true
186
187   - name: Security Plugin configuration | Set the folder permission
188     ansible.builtin.file:
189       dest: "{{ os_conf_dir }}"
190       owner: "{{ os_user }}"
191       group: "{{ os_user }}"
192       mode: 0700
193
194   - name: Security Plugin configuration | Restart opensearch with security configuration
195     ansible.builtin.systemd:
196       name: opensearch
197       state: restarted
198       enabled: true
199
200   - name: Wait for opensearch to startup
201     ansible.builtin.wait_for:
202       host: "{{ hostvars[inventory_hostname]['ip'] }}"
203       port: "{{ os_api_port }}"
204       delay: 5
205       connect_timeout: 1
206       timeout: 120
```

```yaml

- name: Security Plugin configuration | Copy the opensearch security internal users template
  ansible.builtin.template:
    src: internal_users.yml
    dest: "{{ os_sec_plugin_conf_path }}/internal_users.yml"
    mode: 0644
  run_once: true
  when: configuration.changed or iac_enable

- name: Security Plugin configuration | Copy custom configuration files to cluster
  ansible.builtin.template:
    src: "{{ item }}"
    dest: "{{ os_sec_plugin_conf_path }}/"
    owner: "{{ os_user }}"
    group: "{{ os_user }}"
    backup: true
    mode: 0640
    force: true
  with_items: "{{ custom_security_plugin_configs }}"
  when: copy_custom_security_configs

- name: Security Plugin configuration | Set the Admin user password
  ansible.builtin.shell: >
    sed -i '/hash: / s,{{ admin_password }},'$(bash {{ os_sec_plugin_tools_path }}/hash.sh -p {{ admin_password }} | tail -1)',' {{ os_sec_plugin_conf_path }}/internal_users.yml
  environment:
    JAVA_HOME: "{{ os_home }}/jdk"
  run_once: true
  when: configuration.changed or iac_enable

- name: Security Plugin configuration | Set the kibanaserver user pasword
  ansible.builtin.shell: >
    sed -i '/hash: / s,{{ kibanaserver_password }},'$(bash {{ os_sec_plugin_tools_path }}/hash.sh -p {{ kibanaserver_password }} | tail -1)',' {{ os_sec_plugin_conf_path }}/internal_users.yml
  environment:
    JAVA_HOME: "{{ os_home }}/jdk"
  run_once: true
  when: configuration.changed or iac_enable

- name: Security Plugin configuration | Check that the files/internal_users.yml exists
  ansible.builtin.stat:
    path: files/internal_users.yml
  register: custom_users_result
  delegate_to: localhost
  run_once: true
  become: false

- name: Security Plugin configuration | Check for a custom configuration for internal users and hash passwords for them
  when: custom_users_result.stat.exists
  block:

    - name: Security Plugin configuration | Load custom internal users configuration
      ansible.builtin.include_vars:
        file: files/internal_users.yml
        name: custom_users
      run_once: true

    # In the internal_users file.yml each user is described by the block:
    # username:
    #   hash: "{{ username_password }}"In addition to the user description blocks, there is a _meta block
    #   ...
    # In addition to the user description blocks, there is a _meta block
```

```
    # In this task, all usernames are selected from the file (excluding the _meta block), for which hashed
    # passwords will be written next
    - name: Security Plugin configuration | Filter service keys from the list of users
      ansible.builtin.set_fact:
        custom_users_filtered: '{{ custom_users | dict2items | rejectattr("key", "equalto", "_meta") | list | items2dict }}'

    # Hashed passwords are written for all users found in the previous task. Passwords are searched in variables
    # set by the user when starting the role (admin_password, kibanaserver_password, etc.).
    - name: Security Plugin configuration | Set passwords for all users from custom config
      ansible.builtin.shell: >
        sed -i '/hash: / s,{{ lookup('vars', item + '_password') }},'$(bash {{ os_sec_plugin_tools_path }}/hash.sh -p {{ lookup('vars', item + '_password') }} | tail -1)',' {{ os_sec_plugin_conf_path }}/internal_users.yml
      environment:
        JAVA_HOME: "{{ os_home }}/jdk"
      run_once: true
      when: configuration.changed or copy_custom_security_configs
      with_items: "{{ custom_users_filtered }}"

- name: Security Plugin configuration | Initialize the opensearch security index in opensearch with custom configs
  ansible.builtin.shell: >
    bash {{ os_sec_plugin_tools_path }}/securityadmin.sh
    -cacert {{ os_conf_dir }}/root-ca.pem
    -cert {{ os_conf_dir }}/admin.pem
    -key {{ os_conf_dir }}/admin.key
    -cd {{ os_sec_plugin_conf_path }}
    -nhnv -icl
    -h {{ hostvars[inventory_hostname]['ip'] }}
  environment:
    JAVA_HOME: "{{ os_home }}/jdk"
  run_once: true
  when: configuration.changed and copy_custom_security_configs

- name: Security Plugin configuration | Initialize the opensearch security index in opensearch with default configs
  ansible.builtin.shell: >
    bash {{ os_sec_plugin_tools_path }}/securityadmin.sh
    -cacert {{ os_conf_dir }}/root-ca.pem
    -cert {{ os_conf_dir }}/admin.pem
    -key {{ os_conf_dir }}/admin.key
    -f {{ os_sec_plugin_conf_path }}/internal_users.yml
    -nhnv -icl
    -h {{ hostvars[inventory_hostname]['ip'] }}
  environment:
    JAVA_HOME: "{{ os_home }}/jdk"
  run_once: true
  when: configuration.changed and not copy_custom_security_configs

########################
##### DELETED TASK #####
########################
- name: Security Plugin configuration | Cleanup local temporary directory
  local_action:
    module: file
    path: /tmp/opensearch-nodecerts
    state: absent
  run_once: true
  when: configuration.changed
  become: false
```

B.2 New opensearch security.yml

The below yaml code is the modified 'security.yml' file we employ in our project. Comments following the format exemplified by Figure B.1 signify that the task has more than just semantic alterations.

```yaml
---
## Placeholder task to set configuration to changed
## This is a workaround implemented to handle the changes to the ansible role

######################
##### ADDED TASK #####
######################
- name: Security Plugin configuration | Set configuration to changed
  ansible.builtin.set_fact:
    configuration:
      changed: true

### CONSIDER DELETING FROM HERE ###
# These tasks might not be necessary either when distributing the already made certificates to the nodes
# Consider replacing tasks with verification, if all certs are not available -> fail task.
- name: Security Plugin configuration | IaC enabled - Check certificate
  when: iac_enable
  block:
    - name: Security Plugin configuration | Check cert exists
      ansible.builtin.stat:
        path: "{{ item }}"
        get_attributes: false
        get_checksum: false
        get_mime: false
      register: cert_stat_result
      with_items:
        - "{{ os_conf_dir }}/root-ca.pem"
        - "{{ os_conf_dir }}/root-ca.key"
        - "{{ os_conf_dir }}/{{ inventory_hostname }}.key"
        - "{{ os_conf_dir }}/{{ inventory_hostname }}.pem"
        - "{{ os_conf_dir }}/{{ inventory_hostname }}_http.key"
        - "{{ os_conf_dir }}/{{ inventory_hostname }}_http.pem"
        - "{{ os_conf_dir }}/admin.key"
        - "{{ os_conf_dir }}/admin.pem"

    - name: Security Plugin configuration | Set fact. The initial value "Don't update certs"
      ansible.builtin.set_fact:
        force_update_cert: false

    - name: Security Plugin configuration | Set fact. Update certificates if at least one certificate is not found
      ansible.builtin.set_fact:
        force_update_cert: true
      with_items: "{{ cert_stat_result.results }}"
      when: item.stat.exists == False

    - name: Security Plugin configuration | Show the force_update_cert setting
      ansible.builtin.debug:
        msg: "force_update_cert: {{ force_update_cert }}"

    - name: Security Plugin configuration | Count force_update_cert nodes
      ansible.builtin.set_fact:
        force_update_cert_nodes_count: "{{ hostvars | dict2items |
        selectattr('value.force_update_cert', 'defined') |
        rejectattr('value.force_update_cert', 'equalto', false) |
        map(attribute='value.force_update_cert') | list | length }}"

```

```yaml
    - name: Security Plugin configuration | Show the force_update_cert_nodes_count setting
      ansible.builtin.debug:
        msg: "force_update_cert_nodes_count: {{ force_update_cert_nodes_count }}"

    - name: Security Plugin configuration | Do need to update certificates
      ansible.builtin.debug:
        msg: "Need to update certificates..."
      when: force_update_cert_nodes_count | int > 0

- name: Security Plugin configuration | IaC disabled - Count force_update_cert nodes
  ansible.builtin.set_fact:
    force_update_cert_nodes_count: 0
  when: not iac_enable
### TO HERE ###

#########################################################
##### CERTIFICATE DISTRIBUTION TASKS CHANGED HERE #####
#########################################################

- name: Security Plugin configuration | Create certificate files from inventory variables
  ansible.builtin.copy:
    dest: "{{ os_conf_dir }}/{{ item.filename }}"
    content: "{{ item.content }}"
    mode: 0600
  loop:
    - { filename: "root-ca.key", content: "{{ root_ca_key }}" }
    - { filename: "root-ca.pem", content: "{{ root_ca_pem }}" }
    - { filename: "{{ inventory_hostname }}.key", content: "{{ internal_cert_key }}" }
    - { filename: "{{ inventory_hostname }}.pem", content: "{{ internal_cert_pem }}" }
    - { filename: "{{ inventory_hostname }}_http.key", content: "{{ http_cert_key }}" }
    - { filename: "{{ inventory_hostname }}_http.pem", content: "{{ http_cert_pem }}" }
    - { filename: "admin.key", content: "{{ admin_cert_key }}" }
    - { filename: "admin.pem", content: "{{ admin_cert_pem }}" }
  when: (configuration.changed and not iac_enable) or (iac_enable and force_update_cert_nodes_count
  ↪  | int > 0)

- name: Security Plugin configuration | Copy the security configuration file 1 to cluster
  ansible.builtin.blockinfile:
    block: "{{ lookup('template', 'templates/security_conf.yml') }}"
    dest: "{{ os_conf_dir }}/opensearch.yml"
    backup: true
    insertafter: EOF
    marker: "## {mark} OpenSearch Security common configuration ##"
  when: configuration.changed or iac_enable

#############################################
##### THIS TASK REFERENCES THE ADDED    #####
##### 'template_security_conf.yml' FILE #####
#############################################

## Copies from a simple template instead of dynamically generating based on the search-guard tool.
- name: Security Plugin configuration | Copy the security configuration file 2 to cluster
  ansible.builtin.blockinfile:
    block: "{{ lookup('template', 'templates/security_conf_certs.yaml') }}"
    dest: "{{ os_conf_dir }}/opensearch.yml"
    backup: true
    insertafter: EOF
    marker: "## {mark} opensearch Security Node & Admin certificates configuration ##"
  when: configuration.changed or iac_enable

- name: Security Plugin configuration | Create security plugin configuration folder
  ansible.builtin.file:
    dest: "{{ os_sec_plugin_conf_path }}"
    owner: "{{ os_user }}"
    group: "{{ os_user }}"
    mode: 0700
```

```
119       state: directory
120     when: configuration.changed or iac_enable
121
122   - name: Security Plugin configuration | Copy the security configuration file 3 to cluster
123     ansible.builtin.template:
124       src: security_plugin_conf.yml
125       dest: "{{ os_sec_plugin_conf_path }}/config.yml"
126       backup: true
127       owner: "{{ os_user }}"
128       group: "{{ os_user }}"
129       mode: 0600
130       force: true
131     when: auth_type == 'oidc' or copy_custom_security_configs
132
133   ### TO HERE ###
134
135   - name: Security Plugin configuration | Prepare the opensearch security configuration file
136     ansible.builtin.command: sed -i 's/searchguard/plugins.security/g' {{ os_conf_dir
      ↪  }}/opensearch.yml
137     when: configuration.changed or iac_enable
138
139   - name: Security Plugin configuration | Set the file ownerships
140     ansible.builtin.file:
141       dest: "{{ os_home }}"
142       owner: "{{ os_user }}"
143       group: "{{ os_user }}"
144       recurse: true
145
146   - name: Security Plugin configuration | Set the folder permission
147     ansible.builtin.file:
148       dest: "{{ os_conf_dir }}"
149       owner: "{{ os_user }}"
150       group: "{{ os_user }}"
151       mode: 0700
152
153   - name: Security Plugin configuration | Restart opensearch with security configuration
154     ansible.builtin.systemd:
155       name: opensearch
156       state: restarted
157       enabled: true
158
159   - name: Wait for opensearch to startup
160     ansible.builtin.wait_for:
161       host: "{{ hostvars[inventory_hostname]['ansible_host'] }}"
162       port: "{{ os_api_port }}"
163       delay: 5
164       connect_timeout: 1
165       timeout: 180
166
167   - name: Security Plugin configuration | Copy the opensearch security internal users template
168     ansible.builtin.template:
169       src: internal_users.yml
170       dest: "{{ os_sec_plugin_conf_path }}/internal_users.yml"
171       mode: 0644
172     run_once: true
173     when: configuration.changed or iac_enable
174
175   - name: Security Plugin configuration | Copy custom configuration files to cluster
176     ansible.builtin.template:
177       src: "{{ item }}"
178       dest: "{{ os_sec_plugin_conf_path }}/"
179       owner: "{{ os_user }}"
180       group: "{{ os_user }}"
181       backup: true
182       mode: 0640
183       force: true
```

```
184     with_items: "{{ custom_security_plugin_configs }}"
185     when: copy_custom_security_configs
186
187   - name: Security Plugin configuration | Set the Admin user password
188     ansible.builtin.shell: >
189       sed -i '/hash: / s,{{ admin_password }},'$(bash {{ os_sec_plugin_tools_path }}/hash.sh -p {{
        ↪  admin_password }} | tail -1)','
190       {{ os_sec_plugin_conf_path }}/internal_users.yml
191     environment:
192       JAVA_HOME: "{{ os_home }}/jdk"
193     run_once: true
194     when: configuration.changed or iac_enable
195
196   - name: Security Plugin configuration | Set the kibanaserver user pasword
197     ansible.builtin.shell: >
198       sed -i '/hash: / s,{{ kibanaserver_password }},'$(bash {{ os_sec_plugin_tools_path }}/hash.sh
        ↪  -p {{ kibanaserver_password }} | tail -1)','
199       {{ os_sec_plugin_conf_path }}/internal_users.yml
200     environment:
201       JAVA_HOME: "{{ os_home }}/jdk"
202     run_once: true
203     when: configuration.changed or iac_enable
204
205   - name: Security Plugin configuration | Check that the files/internal_users.yml exists
206     ansible.builtin.stat:
207       path: files/internal_users.yml
208     register: custom_users_result
209     delegate_to: localhost
210     run_once: true
211     become: false
212
213   - name: Security Plugin configuration | Check for a custom configuration for internal users and
    ↪  hash passwords for them
214     when: custom_users_result.stat.exists
215     block:
216
217       - name: Security Plugin configuration | Load custom internal users configuration
218         ansible.builtin.include_vars:
219           file: files/internal_users.yml
220           name: custom_users
221         run_once: true
222
223       # In the internal_users file.yml each user is described by the block:
224       # username:
225       #   hash: "{{ username_password }}"In addition to the user description blocks, there is a _meta
        ↪  block
226       #   ...
227       # In addition to the user description blocks, there is a _meta block
228       # In this task, all usernames are selected from the file (excluding the _meta block), for which
        ↪  hashed
229       # passwords will be written next
230       - name: Security Plugin configuration | Filter service keys from the list of users
231         ansible.builtin.set_fact:
232           custom_users_filtered: '{{ custom_users | dict2items | rejectattr("key", "equalto",
            ↪  "_meta") | list | items2dict }}'
233
234       # Hashed passwords are written for all users found in the previous task. Passwords are searched
        ↪  in variables
235       # set by the user when starting the role (admin_password, kibanaserver_password, etc.).
236       - name: Security Plugin configuration | Set passwords for all users from custom config
237         ansible.builtin.shell: >
238           sed -i '/hash: / s,{{ lookup('vars', item + '_password') }},'$(bash {{
            ↪  os_sec_plugin_tools_path }}/hash.sh -p {{ lookup('vars', item + '_password') }} | tail
            ↪  -1)',' 
239           {{ os_sec_plugin_conf_path }}/internal_users.yml
240         environment:
```

```
241           JAVA_HOME: "{{ os_home }}/jdk"
242         run_once: true
243         when: configuration.changed or copy_custom_security_configs
244         with_items: "{{ custom_users_filtered }}"
245
246 - name: Security Plugin configuration | Initialize the opensearch security index in opensearch with
    ↪  custom configs
247   ansible.builtin.shell: >
248     bash {{ os_sec_plugin_tools_path }}/securityadmin.sh
249     -cacert {{ os_conf_dir }}/root-ca.pem
250     -cert {{ os_conf_dir }}/admin.pem
251     -key {{ os_conf_dir }}/admin.key
252     -cd {{ os_sec_plugin_conf_path }}
253     -nhnv -icl
254     -h {{ hostvars[inventory_hostname]['ip'] }}
255   environment:
256     JAVA_HOME: "{{ os_home }}/jdk"
257   run_once: true
258   when: configuration.changed and copy_custom_security_configs
259
260 - name: Security Plugin configuration | Initialize the opensearch security index in opensearch with
    ↪  default configs
261   ansible.builtin.shell: >
262     bash {{ os_sec_plugin_tools_path }}/securityadmin.sh
263     -cacert {{ os_conf_dir }}/root-ca.pem
264     -cert {{ os_conf_dir }}/admin.pem
265     -key {{ os_conf_dir }}/admin.key
266     -f {{ os_sec_plugin_conf_path }}/internal_users.yml
267     -nhnv -icl
268     -h {{ hostvars[inventory_hostname]['ip'] }}
269   environment:
270     JAVA_HOME: "{{ os_home }}/jdk"
271   run_once: true
272   when: configuration.changed and not copy_custom_security_configs
273
```

B.3 certificate configuration template added to role

We needed to include the following configuration template to facilitate external certificate creation and distribution.

```
1  plugins.security.ssl.transport.pemcert_filepath: {{ inventory_hostname }}.pem
2  plugins.security.ssl.transport.pemkey_filepath: {{ inventory_hostname }}.key
3  plugins.security.ssl.transport.pemtrustedcas_filepath: root-ca.pem
4  plugins.security.ssl.transport.enforce_hostname_verification: false
5  plugins.security.ssl.transport.resolve_hostname: false
6  plugins.security.ssl.http.enabled: true
7  plugins.security.ssl.http.pemcert_filepath: {{ inventory_hostname }}_http.pem
8  plugins.security.ssl.http.pemkey_filepath: {{ inventory_hostname }}_http.key
9  plugins.security.ssl.http.pemtrustedcas_filepath: root-ca.pem
10 plugins.security.nodes_dn:
11 {% for item in groups['os-cluster'] %}
12   - CN={{ item }}.{{ domain_name }},OU=CA,O={{ domain_name }}\, Inc.,C=NO
13 {% endfor %}
14 plugins.security.authcz.admin_dn:
15   - CN=admin.{{ domain_name }},OU=CA,O={{ domain_name }}\, Inc.,C=NO
```

Appendix C

Results analysis

Presented here is the dataset generated by aggregating metrics from multiple prometheus sources, followed by merging these by timestamp into one dataset. For a full overview of the individual datasets and the merging process, consult the /bachelor-thesis/results folder in our Gitlab-Project[10].

C.1 With OpenSearch

C.1.1 Dataset

Part 1

Time	Kafka - Messages behind	Kafka - Messages ingested	Kafka - Messages consumed	Vector - Pod count	OS - free memory percent: OS worker 1	OS - free memory percent: OS master 1	OS - free memory percent: OS worker 2
2024-05-10 12:00:00	465.000000	410.000000	518	5	36.000000	3	37.000000
2024-05-10 12:00:10	1400.000000	398.000000	305	5	36.000000	3	37.000000
2024-05-10 12:00:20	1360.000000	387.000000	391	5	36.000000	3	37.000000
2024-05-10 12:00:30	127000.000000	20807.000000	8194	5	36.000000	3	37.000000
2024-05-10 12:00:40	301000.000000	34103.000000	16790	5	34.000000	3	35.000000
2024-05-10 12:00:50	486000.000000	30222.000000	11692	5	32.000000	3	32.000000
2024-05-10 12:01:00	599000.000000	35635.000000	24283	5	29.000000	3	30.000000
2024-05-10 12:01:10	791000.000000	36075.000000	16877	5	27.000000	3	27.000000
2024-05-10 12:01:20	1070000.000000	44259.000000	16096	5	24.000000	3	30.000000
2024-05-10 12:01:30	1330000.000000	50649.000000	25002	5	27.000000	3	27.000000
2024-05-10 12:01:40	1610000.000000	47379.000000	19502	5	23.000000	3	24.000000
2024-05-10 12:01:50	1550000.000000	51366.000000	56877	7	20.000000	3	23.000000
2024-05-10 12:02:00	1220000.000000	43730.000000	77462	7	21.000000	3	20.000000

Continued on next page

Time	Kafka - Messages behind	Kafka - Messages ingested	Kafka - Messages consumed	Vector - Pod count	OS - free memory percent: OS worker 1	OS - free memory percent: OS master 1	OS - free memory percent: OS worker 2
2024-05-10 12:02:10	1550000.000000	46811.000000	13607	7	17.000000	3	16.000000
2024-05-10 12:02:20	1930000.000000	48922.000000	11028	7	14.000000	3	13.000000
2024-05-10 12:02:30	2220000.000000	39966.000000	10757	7	16.000000	3	15.000000
2024-05-10 12:02:40	2600000.000000	51734.000000	13950	7	12.000000	3	13.000000
2024-05-10 12:02:50	3010000.000000	52155.000000	10898	7	8.000000	3	10.000000
2024-05-10 12:03:00	3390000.000000	52509.000000	14042	7	4.000000	3	5.000000
2024-05-10 12:03:10	3800000.000000	52060.000000	11486	7	2.000000	3	13.000000
2024-05-10 12:03:20	4140000.000000	52743.000000	19148	7	13.000000	3	9.000000
2024-05-10 12:03:30	4540000.000000	54110.000000	13765	7	9.000000	3	6.000000
2024-05-10 12:03:40	5010000.000000	54223.000000	7140	7	5.000000	3	7.000000
2024-05-10 12:03:50	5380000.000000	53482.000000	16716	7	6.000000	3	3.000000
2024-05-10 12:04:00	5760000.000000	52570.000000	14737	7	3.000000	3	2.000000
2024-05-10 12:04:10	6200000.000000	53964.000000	9845	7	2.000000	3	2.000000
2024-05-10 12:04:20	6630000.000000	55457.000000	12098	7	2.000000	3	7.000000

Continued on next page

Time	Kafka - Messages behind	Kafka - Messages ingested	Kafka - Messages consumed	Vector - Pod count	OS - free memory percent: OS worker 1	OS - free memory percent: OS master 1	OS - free memory percent: OS worker 2
2024-05-10 12:04:30	7000000.000000	53232.000000	16039	7	15.000000	3	13.000000
2024-05-10 12:04:40	7380000.000000	53143.000000	15597	7	12.000000	3	9.000000
2024-05-10 12:04:50	7770000.000000	52690.000000	13410	7	8.000000	3	4.000000
2024-05-10 12:05:00	8170000.000000	54058.000000	14090	7	9.000000	3	10.000000
2024-05-10 12:05:10	8450000.000000	52501.000000	24612	7	6.000000	3	6.000000
2024-05-10 12:05:20	8650000.000000	47359.000000	27424	7	3.000000	3	3.000000
2024-05-10 12:05:30	8030000.000000	17437.000000	79141	8	2.000000	3	2.000000
2024-05-10 12:05:40	7100000.000000	0.000000	93178	9	2.000000	3	6.000000
2024-05-10 12:05:50	5420000.000000	0.000000	168253	9	7.000000	3	2.000000
2024-05-10 12:06:00	5300000.000000	0.000000	11787	9	3.000000	3	2.000000
2024-05-10 12:06:10	5440000.000000	24883.000000	11267	9	2.000000	3	6.000000
2024-05-10 12:06:20	5690000.000000	36916.000000	11044	9	12.000000	3	10.000000
2024-05-10 12:06:30	6030000.000000	49143.000000	15332	9	9.000000	3	8.000000
2024-05-10 12:06:40	6320000.000000	49565.000000	21238	9	4.000000	3	5.000000

Continued on next page

Time	Kafka - Messages behind	Kafka - Messages ingested	Kafka - Messages consumed	Vector - Pod count	OS - free memory percent: OS worker 1	OS - free memory percent: OS master 1	OS - free memory percent: OS worker 2
2024-05-10 12:06:50	6580000.000000	50608.000000	24275	9	4.000000	3	2.000000
2024-05-10 12:07:00	6790000.000000	51639.000000	30198	9	7.000000	3	10.000000
2024-05-10 12:07:10	6990000.000000	52908.000000	33680	9	8.000000	3	7.000000
2024-05-10 12:07:20	7190000.000000	50229.000000	29484	11	5.000000	3	4.000000
2024-05-10 12:07:30	5720000.000000	50827.000000	197756	12	2.000000	3	2.000000
2024-05-10 12:07:40	6090000.000000	50832.000000	14078	12	2.000000	3	3.000000
2024-05-10 12:07:50	6400000.000000	53273.000000	22039	12	11.000000	3	2.000000
2024-05-10 12:08:00	6740000.000000	49200.000000	16074	12	7.000000	3	2.000000
2024-05-10 12:08:10	7100000.000000	51161.000000	14728	12	5.000000	3	2.000000
2024-05-10 12:08:20	7530000.000000	54642.000000	11511	12	2.000000	3	2.000000
2024-05-10 12:08:30	7990000.000000	53475.000000	7654	12	2.000000	3	2.000000
2024-05-10 12:08:40	8450000.000000	53142.000000	7252	12	7.000000	3	12.000000
2024-05-10 12:08:50	8960000.000000	56529.000000	5453	12	5.000000	3	10.000000
2024-05-10 12:09:00	9360000.000000	53935.000000	13541	12	2.000000	3	6.000000

Continued on next page

Time	Kafka - Messages behind	Kafka - Messages ingested	Kafka - Messages consumed	Vector - Pod count	OS - free memory percent: OS worker 1	OS - free memory percent: OS master 1	OS - free memory percent: OS worker 2
2024-05-10 12:09:10	9770000.000000	55406.000000	14967	12	3.000000	3	14.000000
2024-05-10 12:09:20	9840000.000000	52330.000000	44924	12	17.000000	3	11.000000
2024-05-10 12:09:30	10100000.000000	52619.000000	25538	12	13.000000	3	7.000000
2024-05-10 12:09:40	10400000.000000	53524.000000	26270	12	13.000000	3	3.000000
2024-05-10 12:09:50	10500000.000000	53412.000000	37954	12	8.000000	3	8.000000
2024-05-10 12:10:00	10800000.000000	56342.000000	31426	12	5.000000	3	5.000000
2024-05-10 12:10:10	10900000.000000	51046.000000	35305	12	8.000000	3	2.000000
2024-05-10 12:10:20	10900000.000000	34800.000000	35002	13	5.000000	3	2.000000
2024-05-10 12:10:30	10800000.000000	30512.000000	46304	14	3.000000	3	9.000000
2024-05-10 12:10:40	7850000.000000	40287.000000	333827	14	2.000000	3	5.000000
2024-05-10 12:10:50	8240000.000000	51820.000000	13093	14	3.000000	3	7.000000
2024-05-10 12:11:00	8440000.000000	49320.000000	29514	14	9.000000	3	4.000000
2024-05-10 12:11:10	8320000.000000	10986.000000	23035	14	4.000000	3	2.000000
2024-05-10 12:11:20	8010000.000000	67.400000	30210	14	6.000000	3	10.000000

Continued on next page

Time	Kafka - Messages behind	Kafka - Messages ingested	Kafka - Messages consumed	Vector - Pod count	OS - free memory percent: OS worker 1	OS - free memory percent: OS master 1	OS - free memory percent: OS worker 2
2024-05-10 12:11:30	7640000.000000	129.000000	37513	14	10.000000	3	10.000000
2024-05-10 12:11:40	6950000.000000	190.000000	69317	14	7.000000	3	7.000000
2024-05-10 12:11:50	6610000.000000	252.000000	34341	14	5.000000	3	5.000000
2024-05-10 12:12:00	6380000.000000	313.000000	23277	14	3.000000	3	7.000000
2024-05-10 12:12:10	6270000.000000	367.000000	10843	14	2.000000	3	10.000000
2024-05-10 12:12:20	6210000.000000	378.000000	6504	14	11.000000	3	8.000000
2024-05-10 12:12:30	6120000.000000	385.000000	9372	14	9.000000	3	5.000000
2024-05-10 12:12:40	6040000.000000	391.000000	8953	14	6.000000	3	2.000000
2024-05-10 12:12:50	5910000.000000	399.000000	13292	14	3.000000	3	7.000000
2024-05-10 12:13:00	4990000.000000	406.000000	91870	14	2.000000	3	9.000000
2024-05-10 12:13:10	4040000.000000	415.000000	95628	14	2.000000	3	7.000000
2024-05-10 12:13:20	4040000.000000	430.000000	284	14	7.000000	3	7.000000
2024-05-10 12:13:30	4040000.000000	448.000000	569	14	7.000000	3	6.000000
2024-05-10 12:13:40	4020000.000000	465.000000	2447	14	9.000000	3	4.000000

Continued on next page

Time	Kafka - Messages behind	Kafka - Messages ingested	Kafka - Messages consumed	Vector - Pod count	OS - free memory percent: OS worker 1	OS - free memory percent: OS master 1	OS - free memory percent: OS worker 2
2024-05-10 12:13:50	3980000.000000	482.000000	4348	14	8.000000	3	3.000000
2024-05-10 12:14:00	3330000.000000	502.000000	65711	14	7.000000	3	2.000000
2024-05-10 12:14:10	3310000.000000	511.000000	3010	14	7.000000	3	2.000000
2024-05-10 12:14:20	3210000.000000	500.000000	10219	14	3.000000	3	11.000000
2024-05-10 12:14:30	3140000.000000	488.000000	7255	14	2.000000	3	9.000000
2024-05-10 12:14:40	3120000.000000	466.000000	2047	14	2.000000	3	7.000000
2024-05-10 12:14:50	3120000.000000	454.000000	1262	14	2.000000	3	6.000000
2024-05-10 12:15:00	3080000.000000	437.000000	4392	14	NaN	3	2.000000
2024-05-10 12:15:10	3030000.000000	426.000000	5509	14	2.000000	3	7.000000
2024-05-10 12:15:20	2700000.000000	438.000000	32736	14	2.000000	3	6.000000
2024-05-10 12:15:30	2690000.000000	455.000000	1875	14	2.000000	3	5.000000
2024-05-10 12:15:40	2650000.000000	481.000000	4091	14	NaN	3	5.000000
2024-05-10 12:15:50	2650000.000000	571.000000	433	14	2.000000	3	4.000000
2024-05-10 12:16:00	2640000.000000	662.000000	1695	14	2.000000	3	4.000000

Continued on next page

Time	Kafka - Messages behind	Kafka - Messages ingested	Kafka - Messages consumed	Vector - Pod count	OS - free memory percent: OS worker 1	OS - free memory percent: OS master 1	OS - free memory percent: OS worker 2
2024-05-10 12:16:10	2640000.000000	479.000000	521	14	2.000000	3	3.000000
2024-05-10 12:16:20	2620000.000000	324.000000	3014	14	7.000000	3	3.000000
2024-05-10 12:16:30	2590000.000000	402.000000	3341	14	6.000000	3	2.000000
2024-05-10 12:16:40	2580000.000000	467.000000	1034	14	NaN	3	2.000000
2024-05-10 12:16:50	2560000.000000	468.000000	2369	14	NaN	3	2.000000
2024-05-10 12:17:00	2540000.000000	463.000000	3069	14	4.000000	3	NaN
2024-05-10 12:17:10	2530000.000000	458.000000	879	14	4.000000	3	2.000000
2024-05-10 12:17:20	2330000.000000	455.000000	20275	14	2.000000	3	2.000000
2024-05-10 12:17:30	2120000.000000	447.000000	21699	14	2.000000	3	2.000000
2024-05-10 12:17:40	2110000.000000	443.000000	1450	14	2.000000	3	2.000000
2024-05-10 12:17:50	2100000.000000	431.000000	1571	13	2.000000	3	2.000000
2024-05-10 12:18:00	2090000.000000	424.000000	1043	13	2.000000	3	7.000000
2024-05-10 12:18:10	2090000.000000	409.000000	966	13	2.000000	3	7.000000
2024-05-10 12:18:20	2090000.000000	400.000000	619	13	2.000000	3	NaN

Continued on next page

Time	Kafka - Messages behind	Kafka - Messages ingested	Kafka - Messages consumed	Vector - Pod count	OS - free memory percent: OS worker 1	OS - free memory percent: OS master 1	OS - free memory percent: OS worker 2
2024-05-10 12:18:30	2090000.000000	389.000000	401	13	2.000000	3	NaN
2024-05-10 12:18:40	1980000.000000	383.000000	10793	12	2.000000	3	5.000000
2024-05-10 12:18:50	1970000.000000	393.000000	1667	12	5.000000	3	5.000000
2024-05-10 12:19:00	1960000.000000	405.000000	1780	11	5.000000	3	4.000000
2024-05-10 12:19:10	1880000.000000	417.000000	8082	11	3.000000	3	9.000000
2024-05-10 12:19:20	1680000.000000	431.000000	20010	10	2.000000	3	2.000000
2024-05-10 12:19:30	1640000.000000	442.000000	4731	10	2.000000	3	2.000000
2024-05-10 12:19:40	1630000.000000	450.000000	1743	10	2.000000	3	NaN
2024-05-10 12:19:50	1620000.000000	434.000000	930	10	2.000000	3	2.000000
2024-05-10 12:20:00	1610000.000000	417.000000	1487	10	2.000000	3	2.000000
2024-05-10 12:20:10	1600000.000000	400.000000	1898	10	2.000000	3	3.000000
2024-05-10 12:20:20	1520000.000000	383.000000	7933	9	2.000000	3	3.000000
2024-05-10 12:20:30	1310000.000000	365.000000	21601	9	2.000000	3	2.000000
2024-05-10 12:20:40	1300000.000000	176.000000	868	9	3.000000	3	2.000000

Continued on next page

Time	Kafka - Messages behind	Kafka - Messages ingested	Kafka - Messages consumed	Vector - Pod count	OS - free memory percent: OS worker 1	OS - free memory percent: OS master 1	OS - free memory percent: OS worker 2
2024-05-10 12:20:50	1280000.000000	92.400000	2489	9	2.000000	3	9.000000
2024-05-10 12:21:00	1240000.000000	161.000000	3695	8	2.000000	3	8.000000
2024-05-10 12:21:10	1130000.000000	228.000000	11215	8	2.000000	3	7.000000
2024-05-10 12:21:20	1090000.000000	296.000000	5010	8	5.000000	3	6.000000
2024-05-10 12:21:30	865000.000000	363.000000	22424	7	5.000000	3	4.000000
2024-05-10 12:21:40	804000.000000	405.000000	6518	7	5.000000	3	3.000000
2024-05-10 12:21:50	801000.000000	401.000000	669	7	5.000000	3	2.000000
2024-05-10 12:22:00	649000.000000	400.000000	15620	6	7.000000	3	2.000000
2024-05-10 12:22:10	614000.000000	395.000000	3881	6	7.000000	3	2.000000
2024-05-10 12:22:20	500000.000000	392.000000	11850	5	5.000000	3	2.000000
2024-05-10 12:22:30	496000.000000	389.000000	759	5	4.000000	3	2.000000
2024-05-10 12:22:40	390000.000000	388.000000	10935	5	3.000000	3	2.000000
2024-05-10 12:22:50	376000.000000	394.000000	1808	5	1.000000	3	2.000000
2024-05-10 12:23:00	371000.000000	399.000000	964	5	3.000000	3	3.000000

Continued on next page

Time	Kafka - Messages behind	Kafka - Messages ingested	Kafka - Messages consumed	Vector - Pod count	OS - free memory percent: OS worker 1	OS - free memory percent: OS master 1	OS - free memory percent: OS worker 2
2024-05-10 12:23:10	365000.000000	403.000000	946	5	3.000000	3	2.000000
2024-05-10 12:23:20	330000.000000	409.000000	3958	5	2.000000	3	2.000000
2024-05-10 12:23:30	305000.000000	414.000000	2901	5	2.000000	3	2.000000
2024-05-10 12:23:40	272000.000000	416.000000	3702	5	7.000000	3	2.000000
2024-05-10 12:23:50	1450.000000	414.000000	27467	5	6.000000	3	2.000000
2024-05-10 12:24:00	1440.000000	410.000000	411	5	7.000000	3	2.000000
2024-05-10 12:24:10	1420.000000	407.000000	409	5	3.000000	3	2.000000
2024-05-10 12:24:20	1420.000000	404.000000	404	5	3.000000	3	2.000000
2024-05-10 12:24:30	1400.000000	400.000000	402	5	9.000000	3	2.000000
2024-05-10 12:24:40	1410.000000	399.000000	398	5	9.000000	3	2.000000
2024-05-10 12:24:50	1410.000000	400.000000	399	5	8.000000	3	2.000000
2024-05-10 12:25:00	1420.000000	401.000000	401	5	13.000000	3	2.000000

Part 2

Time	OS - disk io time percent: OS worker 1	OS - disk io time percent: OS master 1	OS - disk io time percent: OS worker 2	OS - Index size	Vector - Input events per second	Vector - Geodata transform MB/s	Vector - Geodata transform utilization percent
2024-05-10 12:00:00	0.042100	0.000000	0.042900	13240568.000000	714.000000	0.259000	0.013900
2024-05-10 12:00:10	0.054700	0.000100	0.055300	NaN	471.000000	0.260000	0.013300
2024-05-10 12:00:20	0.050300	0.000600	0.051600	15345053.000000	782.000000	0.265000	0.012700
2024-05-10 12:00:30	0.077400	0.000000	0.070900	NaN	19300.000000	6.770000	0.068200
2024-05-10 12:00:40	0.140000	0.000100	0.157000	76236477.000000	53300.000000	18.000000	0.169000
2024-05-10 12:00:50	0.163000	0.001600	0.169000	NaN	47100.000000	16.000000	0.157000
2024-05-10 12:01:00	0.138000	0.000000	0.166000	289543128.000000	54200.000000	18.400000	0.164000
2024-05-10 12:01:10	0.159000	0.000100	0.169000	NaN	49300.000000	16.700000	0.169000
2024-05-10 12:01:20	0.273000	0.000700	0.273000	805633567.000000	60900.000000	20.600000	0.189000
2024-05-10 12:01:30	0.222000	0.000000	0.219000	NaN	78400.000000	26.500000	0.236000
2024-05-10 12:01:40	0.169000	0.000100	0.177000	1148732848.000000	56100.000000	19.000000	0.184000
2024-05-10 12:01:50	0.214000	0.001700	0.332000	NaN	58600.000000	19.900000	0.191000
2024-05-10 12:02:00	0.222000	0.000000	0.168000	1773548613.000000	64300.000000	22.800000	0.210000
2024-05-10 12:02:10	0.177000	0.000100	0.132000	NaN	54300.000000	18.300000	0.153000

Continued on next page

Time	OS - disk io time percent: OS worker 1	OS - disk io time percent: OS master 1	OS - disk io time percent: OS worker 2	OS - Index size	Vector - Input events per second	Vector - Geodata transform MB/s	Vector - Geodata transform utilization percent
2024-05-10 12:02:20	0.199000	0.001500	0.238000	2649988947.000000	44300.000000	15.100000	0.117000
2024-05-10 12:02:30	0.182000	0.000000	0.188000	NaN	42600.000000	14.400000	0.104000
2024-05-10 12:02:40	0.166000	0.000000	0.180000	3182579841.000000	58600.000000	19.800000	0.161000
2024-05-10 12:02:50	0.200000	0.001300	0.175000	NaN	44500.000000	15.000000	0.112000
2024-05-10 12:03:00	0.150000	0.000000	0.249000	4112531523.000000	55600.000000	18.900000	0.134000
2024-05-10 12:03:10	0.294000	0.000000	0.280000	NaN	58500.000000	19.800000	0.140000
2024-05-10 12:03:20	0.192000	0.000600	0.156000	4092683131.000000	71500.000000	24.200000	0.161000
2024-05-10 12:03:30	0.198000	0.000000	0.207000	NaN	52500.000000	17.900000	0.125000
2024-05-10 12:03:40	0.217000	0.000100	0.231000	4283226385.000000	31400.000000	10.500000	0.093300
2024-05-10 12:03:50	0.141000	0.001400	0.150000	NaN	71600.000000	24.300000	0.171000
2024-05-10 12:04:00	0.249000	0.000000	0.278000	4989713683.000000	55300.000000	18.700000	0.159000
2024-05-10 12:04:10	0.127000	0.000100	0.125000	NaN	41800.000000	14.200000	0.145000
2024-05-10 12:04:20	0.296000	0.000700	0.431000	5851799677.000000	50600.000000	17.100000	0.176000
2024-05-10 12:04:30	0.208000	0.000000	0.318000	NaN	65400.000000	22.200000	0.268000

Continued on next page

Time	OS - disk io time percent: OS worker 1	OS - disk io time percent: OS master 1	OS - disk io time percent: OS worker 2	OS - Index size	Vector - Input events per second	Vector - Geodata transform MB/s	Vector - Geodata transform utilization percent
2024-05-10 12:04:40	0.154000	0.000100	0.245000	4798923749.000000	61400.000000	20.800000	0.234000
2024-05-10 12:04:50	0.174000	0.001700	0.298000	NaN	58100.000000	19.600000	0.224000
2024-05-10 12:05:00	0.295000	0.000000	0.183000	5423664081.000000	71100.000000	24.200000	0.284000
2024-05-10 12:05:10	0.126000	0.000000	0.138000	NaN	107000.000000	36.200000	0.448000
2024-05-10 12:05:20	0.171000	0.000700	0.243000	5991881740.000000	114000.000000	38.800000	0.516000
2024-05-10 12:05:30	0.948000	0.000000	0.402000	NaN	49300.000000	16.600000	0.391000
2024-05-10 12:05:40	0.773000	0.000000	0.977000	6465162822.000000	10400.000000	3.540000	0.211000
2024-05-10 12:05:50	0.479000	0.001500	0.361000	NaN	12500.000000	5.570000	0.149000
2024-05-10 12:06:00	0.521000	0.000000	0.408000	6922533804.000000	54500.000000	20.700000	0.223000
2024-05-10 12:06:10	0.615000	0.000100	0.277000	NaN	34900.000000	11.700000	0.143000
2024-05-10 12:06:20	0.785000	0.000900	0.849000	6970992729.000000	44000.000000	15.000000	0.089400
2024-05-10 12:06:30	0.150000	0.000000	0.522000	NaN	59900.000000	20.200000	0.176000
2024-05-10 12:06:40	0.184000	0.000000	0.215000	6832975638.000000	75000.000000	25.400000	0.197000
2024-05-10 12:06:50	0.241000	0.001200	0.311000	NaN	99300.000000	33.800000	0.341000

Continued on next page

Time	OS - disk io time percent: OS worker 1	OS - disk io time percent: OS master 1	OS - disk io time percent: OS worker 2	OS - Index size	Vector - Input events per second	Vector - Geodata transform MB/s	Vector - Geodata transform utilization percent
2024-05-10 12:07:00	0.373000	0.000000	0.147000	7413786249.000000	119000.000000	40.200000	0.341000
2024-05-10 12:07:10	0.400000	0.000100	0.213000	NaN	128000.000000	43.300000	0.446000
2024-05-10 12:07:20	0.375000	0.000500	0.157000	7627923442.000000	89800.000000	30.500000	0.356000
2024-05-10 12:07:30	0.281000	0.000000	0.161000	NaN	48300.000000	19.600000	0.135000
2024-05-10 12:07:40	0.639000	0.000100	0.198000	8253154903.000000	83800.000000	29.200000	0.190000
2024-05-10 12:07:50	0.224000	0.001300	0.214000	NaN	51800.000000	17.700000	0.144000
2024-05-10 12:08:00	0.959000	0.000100	0.102000	8512823878.000000	51300.000000	17.300000	0.126000
2024-05-10 12:08:10	0.347000	0.000000	0.530000	NaN	47100.000000	15.900000	0.117000
2024-05-10 12:08:20	0.306000	0.000700	1.230000	8988471003.000000	34500.000000	11.700000	0.117000
2024-05-10 12:08:30	0.389000	0.000000	0.737000	NaN	17600.000000	5.860000	0.059300
2024-05-10 12:08:40	0.984000	0.000100	0.796000	9098940101.000000	23700.000000	8.050000	0.060400
2024-05-10 12:08:50	0.239000	0.001400	0.918000	NaN	36600.000000	12.400000	0.099000
2024-05-10 12:09:00	0.240000	0.000000	0.569000	9670858779.000000	54800.000000	18.600000	0.116000
2024-05-10 12:09:10	0.203000	0.000000	0.321000	NaN	82500.000000	28.100000	0.207000

Continued on next page

Time	OS - disk io time percent: OS worker 1	OS - disk io time percent: OS master 1	OS - disk io time percent: OS worker 2	OS - Index size	Vector - Input events per second	Vector - Geodata transform MB/s	Vector - Geodata transform utilization percent
2024-05-10 12:09:20	0.250000	0.000800	0.321000	9485984092.000000	114000.000000	38.700000	0.261000
2024-05-10 12:09:30	0.621000	0.000000	0.505000	NaN	92400.000000	31.200000	0.229000
2024-05-10 12:09:40	0.771000	0.000100	0.553000	9150629546.000000	112000.000000	38.300000	0.286000
2024-05-10 12:09:50	0.344000	0.001700	0.121000	NaN	142000.000000	48.100000	0.387000
2024-05-10 12:10:00	0.971000	0.000000	0.287000	9773216512.000000	119000.000000	40.500000	0.329000
2024-05-10 12:10:10	0.168000	0.000000	0.132000	NaN	133000.000000	45.200000	0.344000
2024-05-10 12:10:20	0.082300	0.000900	0.234000	10501662167.000000	134000.000000	45.200000	0.351000
2024-05-10 12:10:30	0.201000	0.000000	0.156000	NaN	103000.000000	35.100000	0.331000
2024-05-10 12:10:40	0.184000	0.000100	0.347000	10574149247.000000	46100.000000	21.600000	0.116000
2024-05-10 12:10:50	0.386000	0.001400	0.299000	NaN	71300.000000	33.300000	0.220000
2024-05-10 12:11:00	0.759000	0.000100	0.230000	11440971725.000000	119000.000000	40.500000	0.284000
2024-05-10 12:11:10	0.737000	0.000100	0.521000	NaN	86700.000000	29.600000	0.217000
2024-05-10 12:11:20	0.319000	0.000800	0.686000	11892489202.000000	136000.000000	46.200000	0.287000
2024-05-10 12:11:30	0.914000	0.000100	0.457000	NaN	150000.000000	51.000000	0.316000

Continued on next page

Time	OS - disk io time percent: OS worker 1	OS - disk io time percent: OS master 1	OS - disk io time percent: OS worker 2	OS - Index size	Vector - Input events per second	Vector - Geodata transform MB/s	Vector - Geodata transform utilization percent
2024-05-10 12:11:40	0.977000	0.000000	0.114000	11283806597.000000	135000.000000	45.800000	0.303000
2024-05-10 12:11:50	0.993000	0.001500	0.140000	NaN	99900.000000	33.900000	0.238000
2024-05-10 12:12:00	0.988000	0.000000	0.252000	11891516023.000000	82800.000000	27.900000	0.155000
2024-05-10 12:12:10	0.977000	0.000100	0.141000	NaN	41500.000000	14.000000	0.099200
2024-05-10 12:12:20	0.981000	0.000600	0.133000	11388369756.000000	25400.000000	8.680000	0.069200
2024-05-10 12:12:30	0.976000	0.000000	0.180000	NaN	43200.000000	14.600000	0.093100
2024-05-10 12:12:40	0.985000	0.000100	0.105000	11809859030.000000	31300.000000	10.600000	0.084600
2024-05-10 12:12:50	0.979000	0.001700	0.760000	NaN	33100.000000	11.300000	0.061700
2024-05-10 12:13:00	0.893000	0.000000	0.973000	12293041162.000000	49200.000000	16.700000	0.116000
2024-05-10 12:13:10	0.960000	0.000100	0.992000	NaN	17500.000000	6.050000	0.073500
2024-05-10 12:13:20	0.999000	0.000700	0.997000	12553321684.000000	634.000000	0.213000	0.060200
2024-05-10 12:13:30	0.996000	0.000100	0.992000	NaN	5350.000000	1.810000	0.038400
2024-05-10 12:13:40	0.995000	0.000100	0.989000	12351665413.000000	9400.000000	3.220000	0.037400
2024-05-10 12:13:50	0.992000	0.001500	0.991000	NaN	14300.000000	4.890000	0.031400

Continued on next page

Time	OS - disk io time percent: OS worker 1	OS - disk io time percent: OS master 1	OS - disk io time percent: OS worker 2	OS - Index size	Vector - Input events per second	Vector - Geodata transform MB/s	Vector - Geodata transform utilization percent
2024-05-10 12:14:00	0.996000	0.000000	0.995000	12480010130.000000	1780.000000	0.603000	0.029600
2024-05-10 12:14:10	0.989000	0.000100	0.985000	NaN	15800.000000	5.430000	0.035200
2024-05-10 12:14:20	0.981000	0.000600	0.961000	12457010988.000000	43800.000000	14.800000	0.072200
2024-05-10 12:14:30	0.997000	0.000000	0.992000	NaN	23400.000000	7.850000	0.057000
2024-05-10 12:14:40	0.989000	0.000100	0.991000	12846502111.000000	6530.000000	2.210000	0.035500
2024-05-10 12:14:50	0.996000	0.002000	0.989000	NaN	8510.000000	2.920000	0.031700
2024-05-10 12:15:00	0.995000	0.000200	0.997000	7057254550.000000	13700.000000	4.620000	0.042300
2024-05-10 12:15:10	0.996000	0.000200	0.990000	NaN	21800.000000	7.470000	0.047900
2024-05-10 12:15:20	0.998000	0.001000	0.993000	13183472133.000000	6440.000000	2.180000	0.035200
2024-05-10 12:15:30	0.987000	0.000200	0.988000	NaN	6350.000000	2.150000	0.035800
2024-05-10 12:15:40	0.988000	0.000000	0.995000	6728704177.000000	13100.000000	4.440000	0.028500
2024-05-10 12:15:50	1.000000	0.001400	0.999000	NaN	3200.000000	1.080000	0.031400
2024-05-10 12:16:00	0.990000	0.000200	0.994000	13327508250.000000	5410.000000	1.870000	0.034100
2024-05-10 12:16:10	0.999000	0.000200	0.997000	NaN	1350.000000	0.408000	0.033200

Continued on next page

Time	OS - disk io time percent: OS worker 1	OS - disk io time percent: OS master 1	OS - disk io time percent: OS worker 2	OS - Index size	Vector - Input events per second	Vector - Geodata transform MB/s	Vector - Geodata transform utilization percent
2024-05-10 12:16:20	0.995000	0.000900	0.997000	13546724005.000000	13100.000000	4.450000	0.030900
2024-05-10 12:16:30	0.996000	0.000100	0.995000	NaN	10900.000000	3.680000	0.038200
2024-05-10 12:16:40	0.999000	0.000000	0.989000	6805545273.000000	6330.000000	2.140000	0.036100
2024-05-10 12:16:50	0.997000	0.001800	0.983000	NaN	9050.000000	3.060000	0.033700
2024-05-10 12:17:00	0.996000	0.000000	0.992000	6817367109.000000	10000.000000	3.450000	0.036100
2024-05-10 12:17:10	0.997000	0.000100	0.998000	NaN	2670.000000	0.845000	0.036200
2024-05-10 12:17:20	0.993000	0.000800	0.993000	13715253355.000000	1010.000000	0.344000	0.030300
2024-05-10 12:17:30	0.967000	0.000000	0.995000	NaN	33900.000000	12.300000	0.070200
2024-05-10 12:17:40	0.990000	0.000100	0.989000	13874438350.000000	3630.000000	1.230000	0.039700
2024-05-10 12:17:50	0.996000	0.001400	0.993000	NaN	6280.000000	2.130000	0.023800
2024-05-10 12:18:00	0.993000	0.000100	0.992000	14044061909.000000	2780.000000	0.940000	0.026700
2024-05-10 12:18:10	0.986000	0.000100	0.996000	NaN	3380.000000	1.110000	0.021500
2024-05-10 12:18:20	0.843000	0.000700	0.997000	7233207464.000000	1270.000000	0.392000	0.020700
2024-05-10 12:18:30	0.997000	0.000000	0.990000	NaN	8630.000000	3.050000	0.018400

Continued on next page

Time	OS - disk io time percent: OS worker 1	OS - disk io time percent: OS master 1	OS - disk io time percent: OS worker 2	OS - Index size	Vector - Input events per second	Vector - Geodata transform MB/s	Vector - Geodata transform utilization percent
2024-05-10 12:18:40	0.998000	0.000100	0.996000	14352615997.000000	11400.000000	3.870000	0.033500
2024-05-10 12:18:50	0.989000	0.001400	0.995000	NaN	7820.000000	2.610000	0.019200
2024-05-10 12:19:00	0.997000	0.000000	0.997000	14399662927.000000	5030.000000	1.740000	0.018000
2024-05-10 12:19:10	0.900000	0.000100	0.995000	NaN	24100.000000	8.280000	0.027300
2024-05-10 12:19:20	0.792000	0.000700	0.901000	14078422956.000000	34300.000000	11.600000	0.053800
2024-05-10 12:19:30	1.150000	0.000000	0.995000	NaN	14300.000000	4.830000	0.036100
2024-05-10 12:19:40	0.827000	0.000100	0.993000	7979010021.000000	3970.000000	1.350000	0.019000
2024-05-10 12:19:50	0.996000	0.002200	1.000000	NaN	5370.000000	1.820000	0.017000
2024-05-10 12:20:00	0.997000	0.000000	0.996000	14869971136.000000	2840.000000	0.964000	0.016000
2024-05-10 12:20:10	0.996000	0.000000	0.988000	NaN	24100.000000	8.210000	0.038500
2024-05-10 12:20:20	0.999000	0.000600	0.986000	15061800403.000000	19100.000000	6.450000	0.046400
2024-05-10 12:20:30	0.998000	0.000000	0.976000	NaN	3540.000000	1.220000	0.021600
2024-05-10 12:20:40	0.999000	0.000200	0.996000	15179379443.000000	8100.000000	2.720000	0.024800
2024-05-10 12:20:50	0.995000	0.001600	0.991000	NaN	15800.000000	5.370000	0.028400

Continued on next page

Time	OS - disk io time percent: OS worker 1	OS - disk io time percent: OS master 1	OS - disk io time percent: OS worker 2	OS - Index size	Vector - Input events per second	Vector - Geodata transform MB/s	Vector - Geodata transform utilization percent
2024-05-10 12:21:00	0.994000	0.000000	0.995000	15177500830.000000	10100.000000	3.410000	0.027700
2024-05-10 12:21:10	0.995000	0.000100	0.995000	NaN	7560.000000	2.560000	0.024700
2024-05-10 12:21:20	0.984000	0.000700	0.991000	15551665976.000000	28500.000000	9.630000	0.049100
2024-05-10 12:21:30	0.974000	0.000000	0.993000	NaN	30400.000000	10.300000	0.057200
2024-05-10 12:21:40	0.983000	0.000100	0.993000	15292345356.000000	19100.000000	6.420000	0.075600
2024-05-10 12:21:50	0.992000	0.001700	0.997000	NaN	11200.000000	3.810000	0.027800
2024-05-10 12:22:00	0.997000	0.000100	0.970000	15652683829.000000	6480.000000	2.160000	0.032500
2024-05-10 12:22:10	0.980000	0.000100	0.882000	NaN	37500.000000	12.700000	0.087000
2024-05-10 12:22:20	0.990000	0.000900	0.304000	15723594108.000000	24600.000000	8.270000	0.071700
2024-05-10 12:22:30	0.999000	0.000000	0.144000	NaN	6440.000000	2.180000	0.041100
2024-05-10 12:22:40	0.998000	0.000000	0.950000	16177229267.000000	7050.000000	2.460000	0.042800
2024-05-10 12:22:50	0.997000	0.001700	0.994000	NaN	4230.000000	1.430000	0.035600
2024-05-10 12:23:00	0.994000	0.000000	0.993000	16439409851.000000	2550.000000	0.900000	0.031700
2024-05-10 12:23:10	0.990000	0.000100	0.998000	NaN	9010.000000	3.010000	0.036000

Continued on next page

Time	OS - disk io time percent: OS worker 1	OS - disk io time percent: OS master 1	OS - disk io time percent: OS worker 2	OS - Index size	Vector - Input events per second	Vector - Geodata transform MB/s	Vector - Geodata transform utilization percent
2024-05-10 12:23:20	0.998000	0.000700	0.995000	16588233046.000000	17100.000000	5.840000	0.054600
2024-05-10 12:23:30	0.990000	0.000000	0.990000	NaN	10300.000000	3.530000	0.047800
2024-05-10 12:23:40	0.995000	0.000100	0.988000	16696176068.000000	17000.000000	5.770000	0.062500
2024-05-10 12:23:50	0.997000	0.001400	0.999000	NaN	827.000000	0.280000	0.029800
2024-05-10 12:24:00	0.993000	0.000000	1.130000	16892630685.000000	820.000000	0.278000	0.031000
2024-05-10 12:24:10	0.988000	0.000100	0.844000	NaN	813.000000	0.276000	0.032600
2024-05-10 12:24:20	0.996000	0.000700	0.994000	16892630685.000000	807.000000	0.273000	0.033900
2024-05-10 12:24:30	0.989000	0.000000	0.990000	NaN	800.000000	0.271000	0.033600
2024-05-10 12:24:40	0.994000	0.000100	0.993000	16892630685.000000	797.000000	0.270000	0.033700
2024-05-10 12:24:50	0.988000	0.001900	0.993000	NaN	801.000000	0.271000	0.033800
2024-05-10 12:25:00	0.994000	0.000100	0.992000	16892630685.000000	801.000000	0.271000	0.033600

C.1.2 Jupyter notebook

In [10]:
```
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.ticker import MaxNLocator
from matplotlib.ticker import MultipleLocator
```

In [11]:
```
data = pd.read_csv('with_opensearch/merged-dataset-10-05-24.csv')

data
```

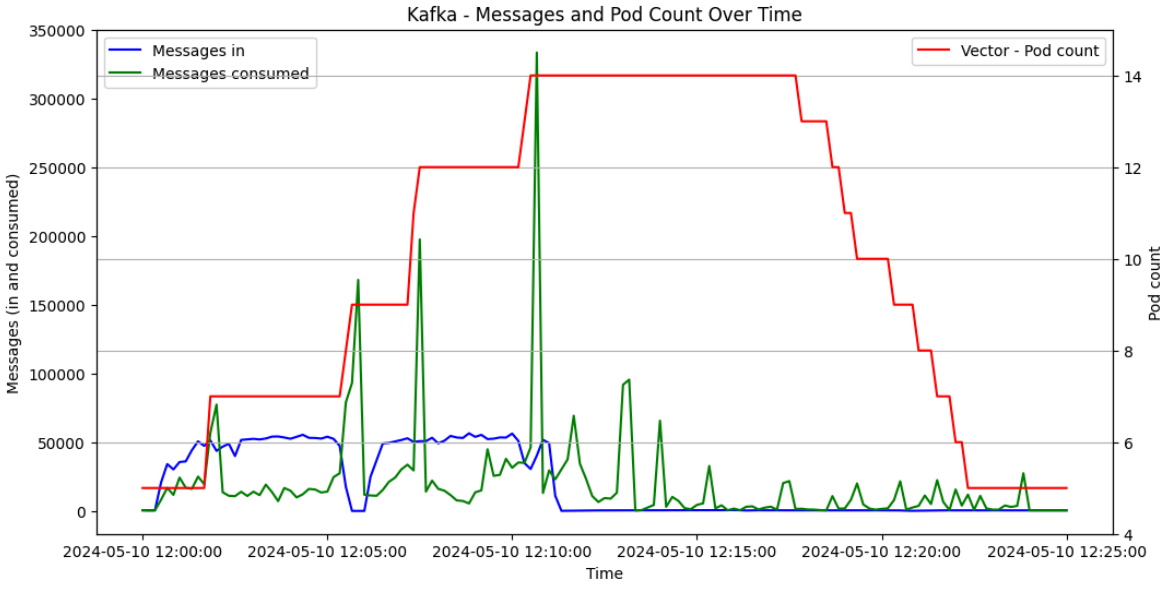
Out[11]:

	Time	Kafka - Messages behind	Kafka - Messages ingested	Kafka - Messages consumed	Vector - Pod count	OS - free memory percent: OS worker 1	OS - free memory percent: OS master 1	OS fr memo percer (work
0	2024-05-10 12:00:00	465.0	410.0	518	5	36.0	3	3
1	2024-05-10 12:00:10	1400.0	398.0	305	5	36.0	3	3
2	2024-05-10 12:00:20	1360.0	387.0	391	5	36.0	3	3
3	2024-05-10 12:00:30	127000.0	20807.0	8194	5	36.0	3	3
4	2024-05-10 12:00:40	301000.0	34103.0	16790	5	34.0	3	35
...	...	...	...	...	...	...	...	
146	2024-05-10 12:24:20	1420.0	404.0	404	5	3.0	3	2
147	2024-05-10 12:24:30	1400.0	400.0	402	5	9.0	3	2
148	2024-05-10 12:24:40	1410.0	399.0	398	5	9.0	3	2
149	2024-05-10 12:24:50	1410.0	400.0	399	5	8.0	3	2
150	2024-05-10 12:25:00	1420.0	401.0	401	5	13.0	3	2

151 rows × 15 columns

Kafka - Messages in, Messages Consumed and Vector pod count

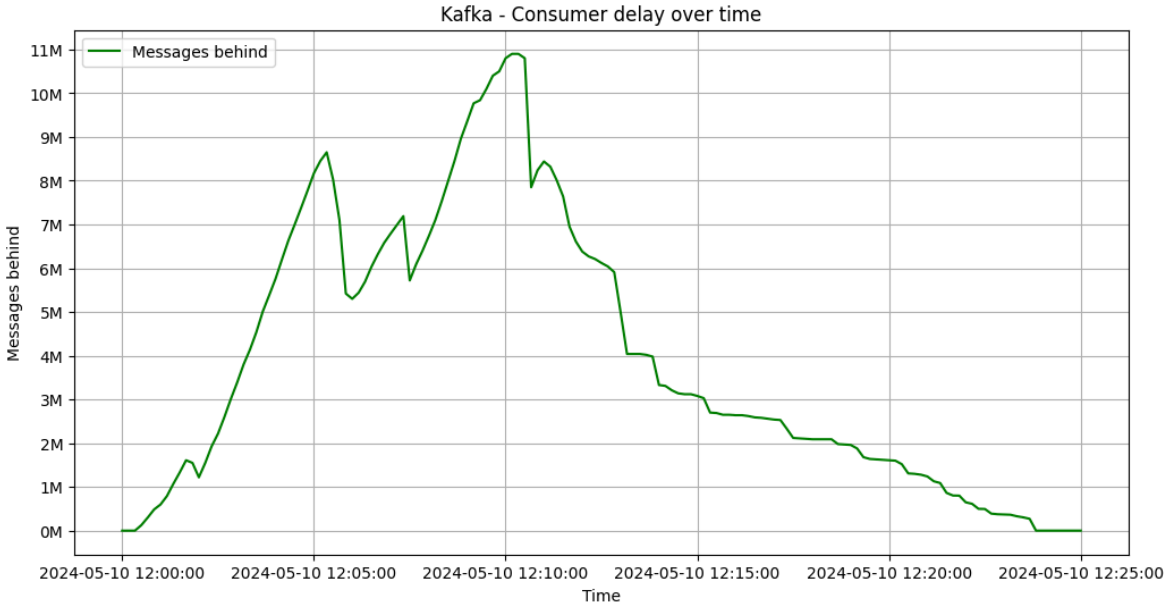
In [12]:
```
fig_kafka_in_out_pod_count, ax1 = plt.subplots(figsize=(12, 6))

ax1.plot(data['Time'], data['Kafka - Messages ingested'], label='Messages
ax1.plot(data['Time'], data['Kafka - Messages consumed'], label='Messages
ax1.set_xlabel('Time')
ax1.set_ylabel('Messages (in and consumed)')
ax1.tick_params(axis='y')
ax1.xaxis.set_major_locator(MaxNLocator(nbins=6))
ax1.legend(loc='upper left')

ax2 = ax1.twinx()
ax2.plot(data['Time'], data['Vector - Pod count'], label='Vector - Pod co
ax2.set_ylabel('Pod count')
ax2.set_ylim(4,15)
ax2.tick_params(axis='y')
ax2.legend(loc='upper right')

plt.title('Kafka - Messages and Pod Count Over Time')
plt.grid(True)

plt.savefig('./with_opensearch/kafka_in_consumed_podcount_with_opensearch
plt.show()
```



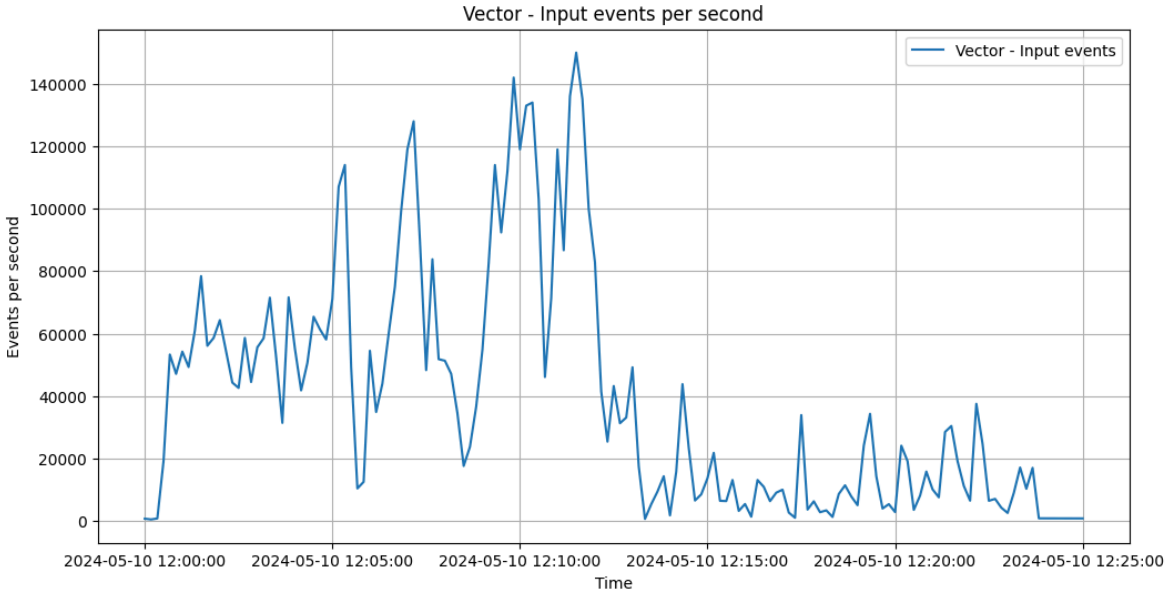
Kafka - Consumer lag

In [13]:
```
fig_kafka_consumer_delay, ax1 = plt.subplots(figsize=(12, 6))

# Plot Messages in and Messages consumed on the left y-axis
ax1.plot(data['Time'], data['Kafka - Messages behind'], label='Messages b
ax1.xaxis.set_major_locator(MaxNLocator(nbins=6))
ax1.yaxis.set_major_locator(MultipleLocator(1e6))
ax1.yaxis.set_major_formatter(plt.FuncFormatter(lambda x, pos: f'{int(x/1
ax1.set_xlabel('Time')
```

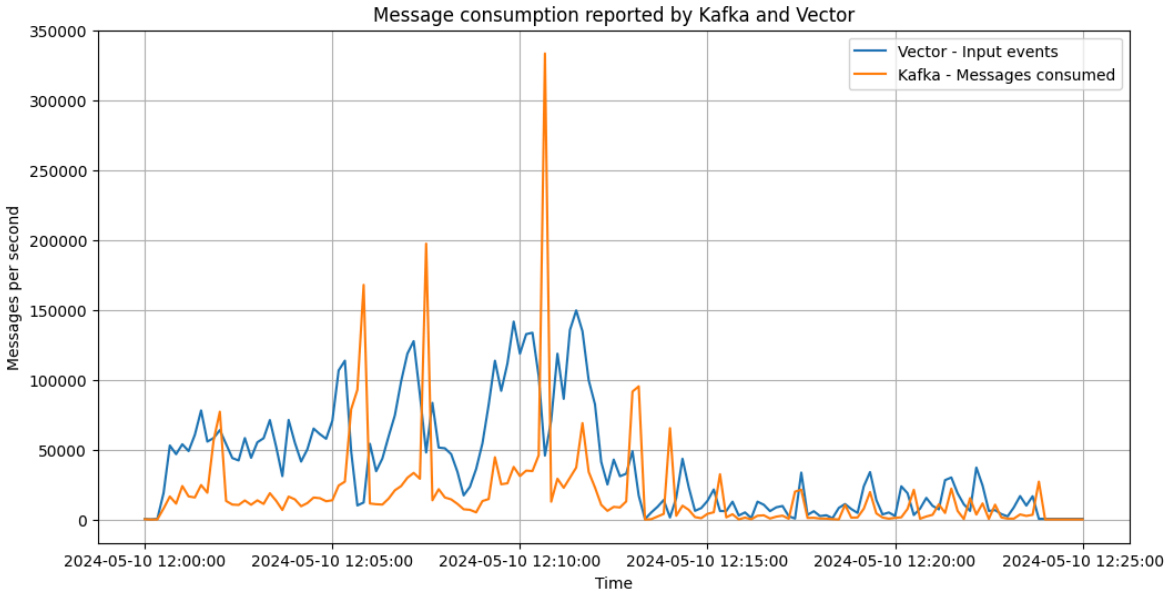
```
ax1.set_ylabel('Messages behind')
ax1.tick_params(axis='y')
ax1.legend(loc='upper left')

# Set title and grid
plt.title('Kafka - Consumer delay over time')
plt.grid(True)

plt.savefig('./with_opensearch/kafka_consumer_delay_with_opensearch.pdf',
plt.show()
```



In [14]:
```
fig_vector_events_per_second, ax1 = plt.subplots(figsize=(12,6))

# Plotting each column
ax1.plot(data['Time'], data['Vector - Input events per second'], label='V
ax1.xaxis.set_major_locator(MaxNLocator(nbins=6))

# Set titles, legend and grid
plt.title('Vector - Input events per second')
ax1.set_xlabel('Time')
ax1.set_ylabel('Events per second')
ax1.legend()
plt.grid(True)

plt.savefig('./with_opensearch/vector_input_events_with_opensearch.pdf',
plt.show()
```

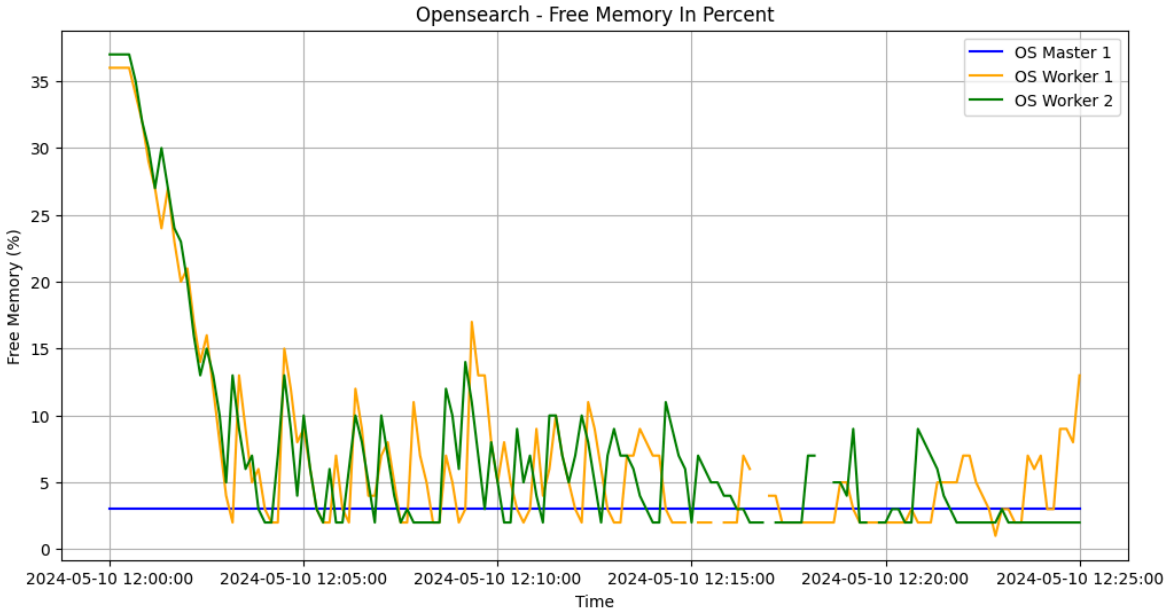


In [15]:
```
fig_kafka_vs_vector_events_per_second, ax1 = plt.subplots(figsize=(12,6))

# Plotting each column
ax1.plot(data['Time'], data['Vector - Input events per second'], label='V
ax1.plot(data['Time'], data['Kafka - Messages consumed'], label='Kafka -
ax1.xaxis.set_major_locator(MaxNLocator(nbins=6))

# Set titles, legend and grid
plt.title('Message consumption reported by Kafka and Vector')
ax1.set_xlabel('Time')
ax1.set_ylabel('Messages per second')
ax1.legend()
plt.grid(True)

plt.savefig('./with_opensearch/kafka_vector_message_consumption_compariso
plt.show()
```

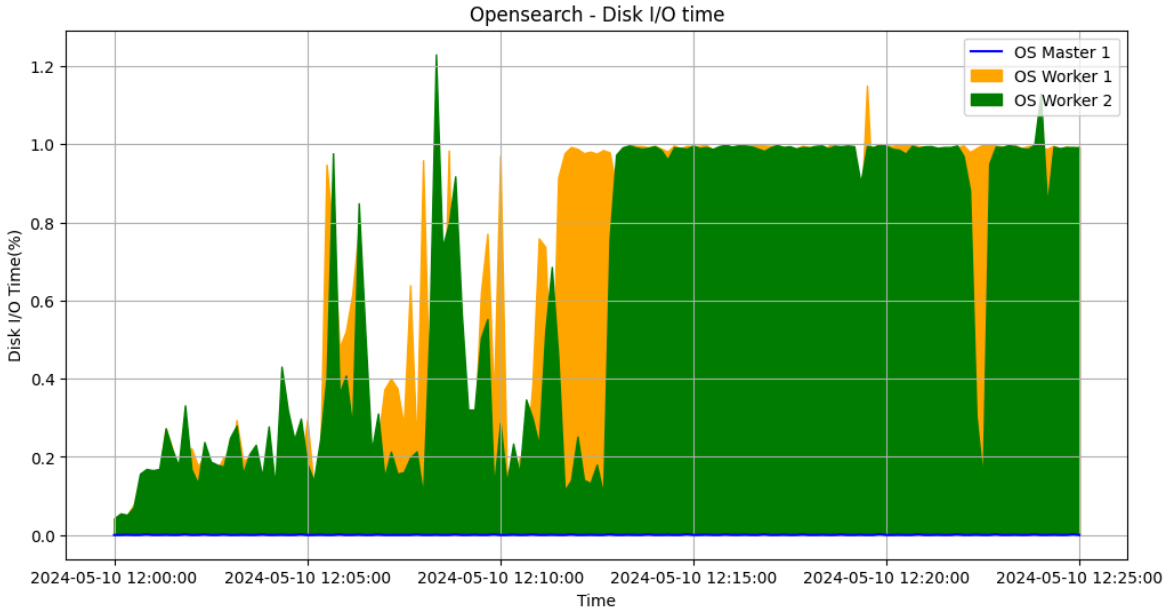


In [16]:
```
fig_opensearch_memory_free, ax1 = plt.subplots(figsize=(12,6))

# Plotting each column
ax1.plot(data['Time'], data['OS - free memory percent: OS master 1'], lab
ax1.plot(data['Time'], data['OS - free memory percent: OS worker 1'], lab
```

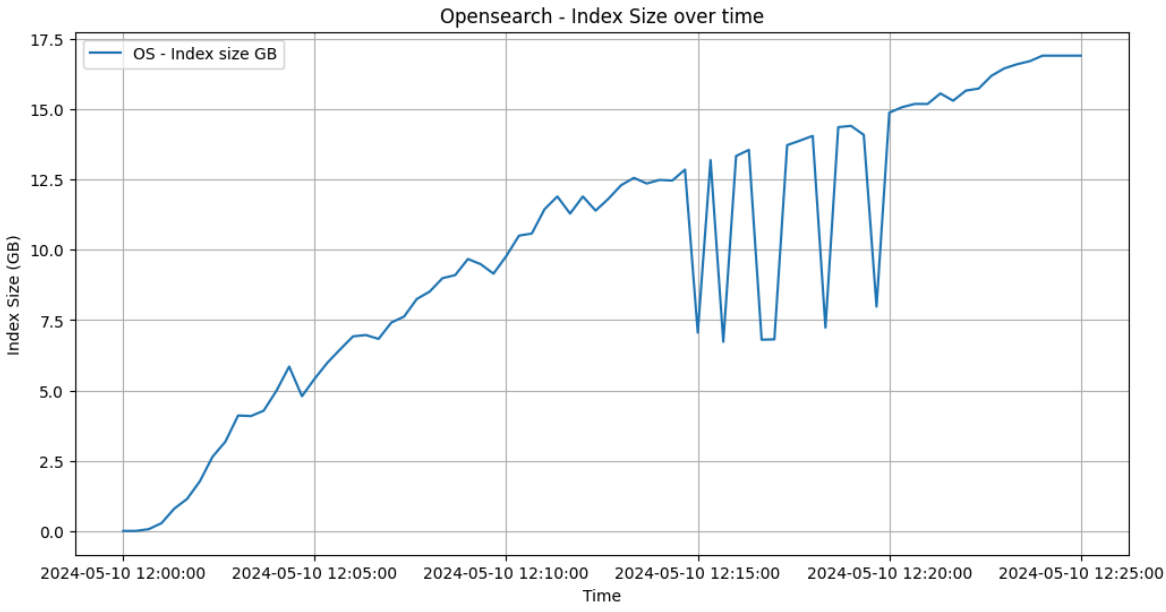
```
ax1.plot(data['Time'], data['OS - free memory percent: OS worker 2'], lab
ax1.xaxis.set_major_locator(MaxNLocator(nbins=6))

# Set titles, legend and grid
plt.title('Opensearch - Free Memory In Percent')
ax1.set_xlabel('Time')
ax1.set_ylabel('Free Memory (%)')
ax1.legend()
plt.grid(True)

plt.savefig('./with_opensearch/opensearch_free_memory.pdf', format='pdf')
plt.show()
```



In [17]:
```
fig_opensearch_disk_io, ax1 = plt.subplots(figsize=(12,6))

# Plotting each column
ax1.plot(data['Time'], data['OS - disk io time percent: OS master 1'], la
ax1.fill_between(data['Time'], data['OS - disk io time percent: OS worker
ax1.fill_between(data['Time'], data['OS - disk io time percent: OS worker
ax1.xaxis.set_major_locator(MaxNLocator(nbins=6))

# Set titles, legend and grid
plt.title('Opensearch - Disk I/O time')
ax1.set_xlabel('Time')
ax1.set_ylabel('Disk I/O Time(%)')
ax1.legend()
plt.grid(True)

plt.savefig('./with_opensearch/opensearch_disk_io_time.pdf', format='pdf'
plt.show()
```



In [18]:
```
# Convert to gigabytes and interpolate values (datapoints here are fetche
data['OS - Index size GB'] = data['OS - Index size'] / 1e9
data['OS - Index size GB'] = data['OS - Index size GB'].interpolate(metho

fig_opensearch_index_size, ax1 = plt.subplots(figsize=(12,6))


ax1.plot(data['Time'], data['OS - Index size GB'], label='OS - Index size
ax1.xaxis.set_major_locator(MaxNLocator(nbins=6))

# Set titles, legend and grid
plt.title('Opensearch - Index Size over time')
ax1.set_xlabel('Time')
ax1.set_ylabel('Index Size (GB)')
ax1.legend()
plt.grid(True)

plt.savefig('./with_opensearch/opensearch_index_size.pdf', format='pdf')
plt.show()
```



In [ ]:

C.2 Without OpenSearch

C.2.1 Dataset

Time	Kafka - Messages behind	Kafka - Messages ingested	Kafka - Messages consumed	Vector - Pod count	Vector - Input events per second	Vector - Geodata transform MB/s	Vector - Geodata transform utilization percent
2024-05-10 15:20:00	729.000000	291.000000	292.000000	5	582.000000	0.197000	0.008520
2024-05-10 15:20:10	698.000000	281.000000	284.000000	5	565.000000	0.191000	0.008640
2024-05-10 15:20:20	682.000000	273.000000	274.000000	5	547.000000	0.185000	0.007510
2024-05-10 15:20:30	16900.000000	2447.000000	825.000000	5	7580.000000	2.570000	0.019100
2024-05-10 15:20:40	89600.000000	32350.000000	25081.000000	5	54700.000000	18.500000	0.245000
2024-05-10 15:20:50	62400.000000	35170.000000	37893.000000	5	71500.000000	24.200000	0.311000
2024-05-10 15:21:00	77500.000000	46693.000000	45176.000000	5	90200.000000	30.500000	0.392000
2024-05-10 15:21:10	84500.000000	39862.000000	39162.000000	5	83200.000000	28.200000	0.377000
2024-05-10 15:21:20	55100.000000	38728.000000	41668.000000	7	72900.000000	25.100000	0.315000
2024-05-10 15:21:30	44800.000000	48973.000000	50001.000000	7	63000.000000	28.600000	0.351000
2024-05-10 15:21:40	102000.000000	47995.000000	42258.000000	7	62400.000000	29.600000	0.301000
2024-05-10 15:21:50	81700.000000	45469.000000	47517.000000	7	92100.000000	31.200000	0.257000
2024-05-10 15:22:00	79900.000000	36834.000000	37015.000000	8	74900.000000	25.400000	0.208000
2024-05-10 15:22:10	58700.000000	36877.000000	39005.000000	9	72500.000000	25.100000	0.191000

Continued on next page

Time	Kafka - Messages behind	Kafka - Messages ingested	Kafka - Messages consumed	Vector - Pod count	Vector - Input events per second	Vector - Geodata transform MB/s	Vector - Geodata transform utilization percent
2024-05-10 15:22:20	78700.000000	46904.000000	44896.000000	9	88800.000000	30.100000	0.217000
2024-05-10 15:22:30	76800.000000	50157.000000	50354.000000	9	95800.000000	32.500000	0.201000
2024-05-10 15:22:40	108000.000000	51712.000000	48549.000000	9	103000.000000	34.800000	0.196000
2024-05-10 15:22:50	81100.000000	52626.000000	55356.000000	9	105000.000000	35.600000	0.205000
2024-05-10 15:23:00	84800.000000	53528.000000	53151.000000	9	107000.000000	36.100000	0.208000
2024-05-10 15:23:10	84100.000000	53066.000000	53136.000000	9	107000.000000	36.100000	0.214000
2024-05-10 15:23:20	93300.000000	55383.000000	54465.000000	9	109000.000000	37.000000	0.215000
2024-05-10 15:23:30	90300.000000	55355.000000	55653.000000	9	112000.000000	37.800000	0.228000
2024-05-10 15:23:40	90100.000000	54759.000000	54779.000000	9	109000.000000	36.900000	0.216000
2024-05-10 15:23:50	84400.000000	53359.000000	53938.000000	9	107000.000000	36.400000	0.213000
2024-05-10 15:24:00	109000.000000	53180.000000	50676.000000	9	107000.000000	36.100000	0.216000
2024-05-10 15:24:10	82700.000000	53198.000000	55864.000000	9	107000.000000	36.100000	0.212000
2024-05-10 15:24:20	110000.000000	53897.000000	51220.000000	9	108000.000000	36.600000	0.208000
2024-05-10 15:24:30	81700.000000	49892.000000	52677.000000	9	102000.000000	34.500000	0.205000

Continued on next page

Time	Kafka - Messages behind	Kafka - Messages ingested	Kafka - Messages consumed	Vector - Pod count	Vector - Input events per second	Vector - Geodata transform MB/s	Vector - Geodata transform utilization percent
2024-05-10 15:24:40	84700.000000	42956.000000	42655.000000	9	85800.000000	29.100000	0.165000
2024-05-10 15:24:50	81500.000000	41753.000000	42073.000000	9	84100.000000	28.500000	0.169000
2024-05-10 15:25:00	84900.000000	42533.000000	42193.000000	9	84400.000000	28.600000	0.172000
2024-05-10 15:25:10	83800.000000	40991.000000	41100.000000	9	81900.000000	27.700000	0.161000
2024-05-10 15:25:20	69000.000000	38172.000000	39653.000000	9	80500.000000	27.300000	0.156000
2024-05-10 15:25:30	21100.000000	38396.000000	43185.000000	9	74000.000000	25.100000	0.183000
2024-05-10 15:25:40	0.000000	0.000000	2108.000000	9	6900.000000	2.340000	0.156000
2024-05-10 15:25:50	50100.000000	21248.000000	16239.000000	9	36700.000000	12.400000	0.091100
2024-05-10 15:26:00	57500.000000	35543.000000	34798.000000	9	69800.000000	23.600000	0.140000
2024-05-10 15:26:10	71400.000000	35585.000000	34199.000000	9	71100.000000	24.100000	0.143000
2024-05-10 15:26:20	68900.000000	34363.000000	34608.000000	9	69900.000000	23.700000	0.141000
2024-05-10 15:26:30	72100.000000	37815.000000	37501.000000	9	72500.000000	24.600000	0.143000
2024-05-10 15:26:40	76100.000000	36492.000000	36093.000000	9	74900.000000	25.400000	0.153000
2024-05-10 15:26:50	56600.000000	37769.000000	39713.000000	9	75500.000000	25.600000	0.150000

Continued on next page

Time	Kafka - Messages behind	Kafka - Messages ingested	Kafka - Messages consumed	Vector - Pod count	Vector - Input events per second	Vector - Geodata transform MB/s	Vector - Geodata transform utilization percent
2024-05-10 15:27:00	101000.000000	42699.000000	38220.000000	9	84000.000000	28.400000	0.167000
2024-05-10 15:27:10	85400.000000	53728.000000	55332.000000	9	105000.000000	35.700000	0.207000
2024-05-10 15:27:20	84500.000000	51354.000000	51442.000000	9	103000.000000	34.900000	0.209000
2024-05-10 15:27:30	108000.000000	51358.000000	49018.000000	9	103000.000000	35.000000	0.213000
2024-05-10 15:27:40	92800.000000	51246.000000	52754.000000	9	102000.000000	34.500000	0.204000
2024-05-10 15:27:50	79100.000000	39458.000000	40824.000000	9	81900.000000	27.700000	0.161000
2024-05-10 15:28:00	86600.000000	40710.000000	39966.000000	9	82800.000000	28.000000	0.162000
2024-05-10 15:28:10	84300.000000	52744.000000	52977.000000	9	101000.000000	34.200000	0.200000
2024-05-10 15:28:20	108000.000000	52115.000000	49770.000000	9	105000.000000	35.500000	0.211000
2024-05-10 15:28:30	104000.000000	51478.000000	51873.000000	9	104000.000000	35.200000	0.210000
2024-05-10 15:28:40	111000.000000	53966.000000	53252.000000	9	107000.000000	36.100000	0.210000
2024-05-10 15:28:50	82600.000000	53034.000000	55867.000000	9	106000.000000	36.000000	0.208000
2024-05-10 15:29:00	108000.000000	52930.000000	50402.000000	9	106000.000000	35.900000	0.203000
2024-05-10 15:29:10	112000.000000	55492.000000	55102.000000	9	110000.000000	37.300000	0.209000

Continued on next page

Time	Kafka - Messages behind	Kafka - Messages ingested	Kafka - Messages consumed	Vector - Pod count	Vector - Input events per second	Vector - Geodata transform MB/s	Vector - Geodata transform utilization percent
2024-05-10 15:29:20	102000.000000	51792.000000	52806.000000	9	104000.000000	35.200000	0.205000
2024-05-10 15:29:30	80700.000000	49185.000000	51279.000000	9	98700.000000	33.400000	0.196000
2024-05-10 15:29:40	80700.000000	54707.000000	54705.000000	9	109000.000000	37.100000	0.210000
2024-05-10 15:29:50	112000.000000	56371.000000	53249.000000	9	112000.000000	38.100000	0.212000
2024-05-10 15:30:00	86600.000000	57695.000000	60220.000000	9	115000.000000	38.800000	0.221000
2024-05-10 15:30:10	119000.000000	57296.000000	54058.000000	9	115000.000000	38.900000	0.221000
2024-05-10 15:30:20	115000.000000	57063.000000	57435.000000	9	115000.000000	38.800000	0.224000
2024-05-10 15:30:30	111000.000000	56066.000000	56508.000000	9	112000.000000	37.800000	0.219000
2024-05-10 15:30:40	80800.000000	46763.000000	49770.000000	9	98900.000000	33.500000	0.196000
2024-05-10 15:30:50	6.000000	15464.000000	23543.000000	9	36500.000000	12.400000	0.145000
2024-05-10 15:31:00	113.000000	35.600000	24.900000	9	61.800000	0.020900	0.014000
2024-05-10 15:31:10	223.000000	91.500000	80.500000	9	173.000000	0.058600	0.002150
2024-05-10 15:31:20	334.000000	148.000000	137.000000	9	285.000000	0.096500	0.003230
2024-05-10 15:31:30	447.000000	204.000000	193.000000	9	397.000000	0.135000	0.004510

Continued on next page

Time	Kafka - Messages behind	Kafka - Messages ingested	Kafka - Messages consumed	Vector - Pod count	Vector - Input events per second	Vector - Geodata transform MB/s	Vector - Geodata transform utilization percent
2024-05-10 15:31:40	554.000000	260.000000	249.000000	9	509.000000	0.172000	0.005380
2024-05-10 15:31:50	661.000000	316.000000	305.000000	9	618.000000	0.209000	0.006390
2024-05-10 15:32:00	648.000000	330.000000	332.000000	9	660.000000	0.224000	0.006450
2024-05-10 15:32:10	621.000000	317.000000	320.000000	9	635.000000	0.215000	0.006650
2024-05-10 15:32:20	603.000000	303.000000	304.000000	9	610.000000	0.206000	0.006690
2024-05-10 15:32:30	577.000000	290.000000	293.000000	9	581.000000	0.197000	0.006170
2024-05-10 15:32:40	545.000000	276.000000	279.000000	9	556.000000	0.188000	0.006030
2024-05-10 15:32:50	518.000000	263.000000	266.000000	9	529.000000	0.179000	0.005730
2024-05-10 15:33:00	534.000000	262.000000	260.000000	9	523.000000	0.177000	0.005710
2024-05-10 15:33:10	551.000000	269.000000	267.000000	9	535.000000	0.181000	0.005960
2024-05-10 15:33:20	559.000000	274.000000	274.000000	9	548.000000	0.186000	0.006030
2024-05-10 15:33:30	564.000000	281.000000	280.000000	9	561.000000	0.190000	0.006210
2024-05-10 15:33:40	577.000000	287.000000	286.000000	9	574.000000	0.194000	0.006370
2024-05-10 15:33:50	580.000000	293.000000	293.000000	9	587.000000	0.199000	0.006320

Continued on next page

Time	Kafka - Messages behind	Kafka - Messages ingested	Kafka - Messages consumed	Vector - Pod count	Vector - Input events per second	Vector - Geodata transform MB/s	Vector - Geodata transform utilization percent
2024-05-10 15:34:00	581.000000	297.000000	297.000000	9	594.000000	0.201000	0.006430
2024-05-10 15:34:10	589.000000	300.000000	299.000000	9	599.000000	0.203000	0.006490
2024-05-10 15:34:20	590.000000	302.000000	302.000000	9	604.000000	0.205000	0.006510
2024-05-10 15:34:30	601.000000	305.000000	304.000000	9	610.000000	0.207000	0.006610
2024-05-10 15:34:40	608.000000	308.000000	307.000000	9	615.000000	0.208000	0.006640
2024-05-10 15:34:50	609.000000	309.000000	309.000000	9	621.000000	0.210000	0.006540
2024-05-10 15:35:00	463.000000	313.000000	327.000000	9	621.000000	0.211000	0.006630
2024-05-10 15:35:10	607.000000	311.000000	296.000000	9	623.000000	0.211000	0.006570
2024-05-10 15:35:20	614.000000	312.000000	312.000000	9	623.000000	0.211000	0.006560
2024-05-10 15:35:30	620.000000	312.000000	311.000000	9	624.000000	0.211000	0.006810
2024-05-10 15:35:40	654.000000	318.000000	315.000000	9	635.000000	0.215000	0.006970
2024-05-10 15:35:50	487.000000	332.000000	349.000000	9	631.000000	0.214000	0.006250
2024-05-10 15:36:00	228.000000	97.600000	123.000000	9	270.000000	0.091300	0.003390
2024-05-10 15:36:10	328.000000	147.000000	137.000000	9	267.000000	0.090500	0.003450

Continued on next page

Time	Kafka - Messages behind	Kafka - Messages ingested	Kafka - Messages consumed	Vector - Pod count	Vector - Input events per second	Vector - Geodata transform MB/s	Vector - Geodata transform utilization percent
2024-05-10 15:36:20	280.000000	196.000000	201.000000	8	322.000000	0.109000	0.003940
2024-05-10 15:36:30	641.000000	245.000000	208.000000	7	454.000000	0.154000	0.004890
2024-05-10 15:36:40	377.000000	288.000000	315.000000	6	473.000000	0.160000	0.006200
2024-05-10 15:36:50	727.000000	293.000000	258.000000	6	572.000000	0.194000	0.006690
2024-05-10 15:37:00	718.000000	290.000000	291.000000	5	581.000000	0.197000	0.007040
2024-05-10 15:37:10	711.000000	287.000000	288.000000	5	574.000000	0.194000	0.007380
2024-05-10 15:37:20	703.000000	283.000000	284.000000	5	569.000000	0.193000	0.007430
2024-05-10 15:37:30	703.000000	281.000000	281.000000	5	561.000000	0.190000	0.007380
2024-05-10 15:37:40	703.000000	279.000000	279.000000	5	560.000000	0.190000	0.007350
2024-05-10 15:37:50	742.000000	292.000000	288.000000	5	580.000000	0.197000	0.007390
2024-05-10 15:38:00	779.000000	306.000000	303.000000	5	612.000000	0.207000	0.007570
2024-05-10 15:38:10	821.000000	323.000000	319.000000	5	642.000000	0.218000	0.007760
2024-05-10 15:38:20	861.000000	339.000000	335.000000	5	672.000000	0.228000	0.008130
2024-05-10 15:38:30	906.000000	354.000000	349.000000	5	704.000000	0.238000	0.007920

Continued on next page

Time	Kafka - Messages behind	Kafka - Messages ingested	Kafka - Messages consumed	Vector - Pod count	Vector - Input events per second	Vector - Geodata transform MB/s	Vector - Geodata transform utilization percent
2024-05-10 15:38:40	921.000000	367.000000	365.000000	5	732.000000	0.248000	0.008410
2024-05-10 15:38:50	916.000000	369.000000	369.000000	5	738.000000	0.250000	0.008220
2024-05-10 15:39:00	928.000000	368.000000	367.000000	5	734.000000	0.249000	0.007960
2024-05-10 15:39:10	909.000000	364.000000	366.000000	5	731.000000	0.247000	0.008490
2024-05-10 15:39:20	896.000000	363.000000	364.000000	5	726.000000	0.246000	0.007920
2024-05-10 15:39:30	912.000000	364.000000	362.000000	5	724.000000	0.245000	0.007950
2024-05-10 15:39:40	884.000000	357.000000	360.000000	5	717.000000	0.243000	0.008170
2024-05-10 15:39:50	847.000000	345.000000	349.000000	5	693.000000	0.235000	0.008090
2024-05-10 15:40:00	805.000000	329.000000	334.000000	5	663.000000	0.225000	0.007860
2024-05-10 15:40:10	770.000000	315.000000	318.000000	5	632.000000	0.214000	0.007580
2024-05-10 15:40:20	732.000000	299.000000	303.000000	5	602.000000	0.204000	0.006800
2024-05-10 15:40:30	696.000000	285.000000	288.000000	5	571.000000	0.194000	0.007070
2024-05-10 15:40:40	28.000000	155.000000	221.000000	5	361.000000	0.122000	0.006690
2024-05-10 15:40:50	0.000000	0.000000	2.800000	5	4.700000	0.001590	0.006690

Continued on next page

Time	Kafka - Messages behind	Kafka - Messages ingested	Kafka - Messages consumed	Vector - Pod count	Vector - Input events per second	Vector - Geodata transform MB/s	Vector - Geodata transform utilization percent
2024-05-10 15:41:00	17.000000	2.200000	0.500000	5	7.400000	0.002520	0.005420
2024-05-10 15:41:10	123.000000	54.500000	43.900000	5	94.800000	0.032100	0.003250
2024-05-10 15:41:20	371.000000	126.000000	101.000000	5	239.000000	0.080800	0.003980
2024-05-10 15:41:30	558.000000	199.000000	180.000000	5	383.000000	0.130000	0.005910
2024-05-10 15:41:40	744.000000	272.000000	254.000000	5	530.000000	0.179000	0.008140
2024-05-10 15:41:50	921.000000	343.000000	326.000000	5	673.000000	0.228000	0.009430
2024-05-10 15:42:00	1080.000000	415.000000	399.000000	5	810.000000	0.274000	0.010900
2024-05-10 15:42:10	1030.000000	420.000000	424.000000	5	841.000000	0.285000	0.012100
2024-05-10 15:42:20	978.000000	395.000000	401.000000	5	795.000000	0.269000	0.011200
2024-05-10 15:42:30	910.000000	371.000000	377.000000	5	746.000000	0.253000	0.010900
2024-05-10 15:42:40	850.000000	346.000000	352.000000	5	699.000000	0.237000	0.010300
2024-05-10 15:42:50	790.000000	323.000000	329.000000	5	650.000000	0.220000	0.009550
2024-05-10 15:43:00	732.000000	298.000000	304.000000	5	604.000000	0.205000	0.008490
2024-05-10 15:43:10	773.000000	303.000000	299.000000	5	603.000000	0.204000	0.008840

Continued on next page

Time	Kafka - Messages behind	Kafka - Messages ingested	Kafka - Messages consumed	Vector - Pod count	Vector - Input events per second	Vector - Geodata transform MB/s	Vector - Geodata transform utilization percent
2024-05-10 15:43:20	818.000000	321.000000	316.000000	5	637.000000	0.216000	0.008980
2024-05-10 15:43:30	850.000000	336.000000	333.000000	5	670.000000	0.227000	0.009710
2024-05-10 15:43:40	906.000000	355.000000	349.000000	5	704.000000	0.238000	0.010100
2024-05-10 15:43:50	950.000000	371.000000	366.000000	5	738.000000	0.250000	0.010300
2024-05-10 15:44:00	981.000000	386.000000	383.000000	5	771.000000	0.261000	0.010700
2024-05-10 15:44:10	991.000000	396.000000	395.000000	5	792.000000	0.268000	0.011300
2024-05-10 15:44:20	1010.000000	404.000000	402.000000	5	806.000000	0.273000	0.010500
2024-05-10 15:44:30	1030.000000	411.000000	409.000000	5	820.000000	0.278000	0.011400
2024-05-10 15:44:40	1050.000000	418.000000	416.000000	5	834.000000	0.282000	0.011500
2024-05-10 15:44:50	1070.000000	426.000000	424.000000	5	849.000000	0.288000	0.011700
2024-05-10 15:45:00	1090.000000	432.000000	430.000000	5	860.000000	0.292000	0.011600

C.2.2 Jupyter notebook

In [9]:
```
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.ticker import MaxNLocator, MultipleLocator
```

In [10]:
```
data = pd.read_csv('without_opensearch/merged-dataset-10-05-24.csv')
data_with_opensearch = pd.read_csv('with_opensearch/merged-dataset-10-05-

data
```

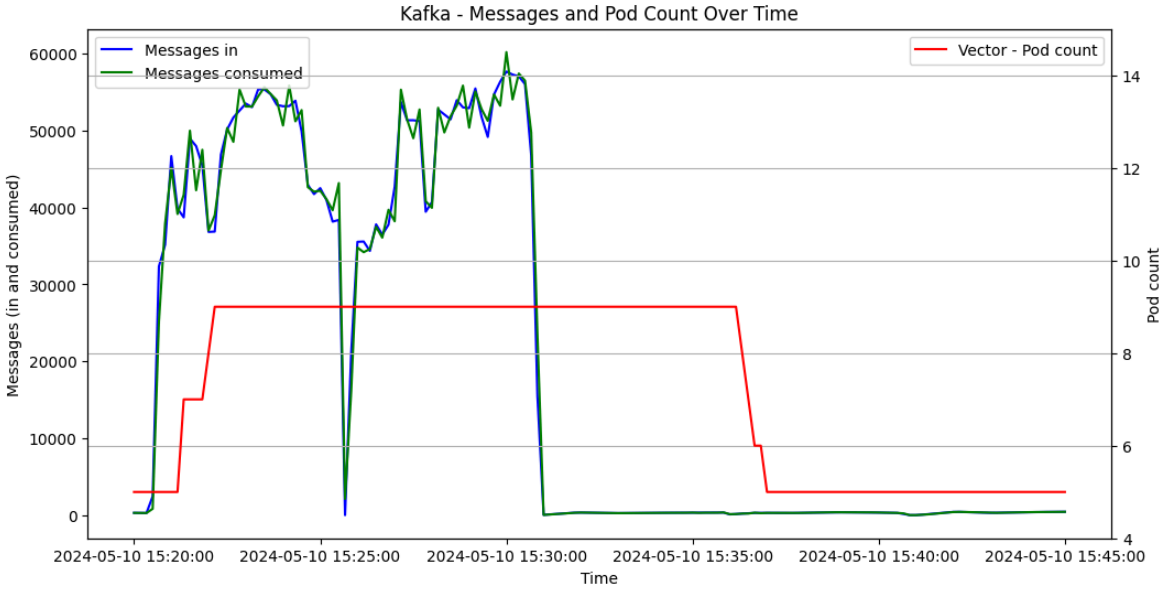
Out[10]:

	Time	Kafka - Messages behind	Kafka - Messages ingested	Kafka - Messages consumed	Vector - Pod count	Vector - Input events per second	Vector - Geodata transform MB/s	Vec Geo trans utiliz per
0	2024-05-10 15:20:00	729.0	291.0	292.0	5	582.0	0.197	0.0
1	2024-05-10 15:20:10	698.0	281.0	284.0	5	565.0	0.191	0.0
2	2024-05-10 15:20:20	682.0	273.0	274.0	5	547.0	0.185	0.0
3	2024-05-10 15:20:30	16900.0	2447.0	825.0	5	7580.0	2.570	0.0
4	2024-05-10 15:20:40	89600.0	32350.0	25081.0	5	54700.0	18.500	0.2
...	...	...	...	...	...	...	...	
146	2024-05-10 15:44:20	1010.0	404.0	402.0	5	806.0	0.273	0.0
147	2024-05-10 15:44:30	1030.0	411.0	409.0	5	820.0	0.278	0.0
148	2024-05-10 15:44:40	1050.0	418.0	416.0	5	834.0	0.282	0.0
149	2024-05-10 15:44:50	1070.0	426.0	424.0	5	849.0	0.288	0.0
150	2024-05-10 15:45:00	1090.0	432.0	430.0	5	860.0	0.292	0.0

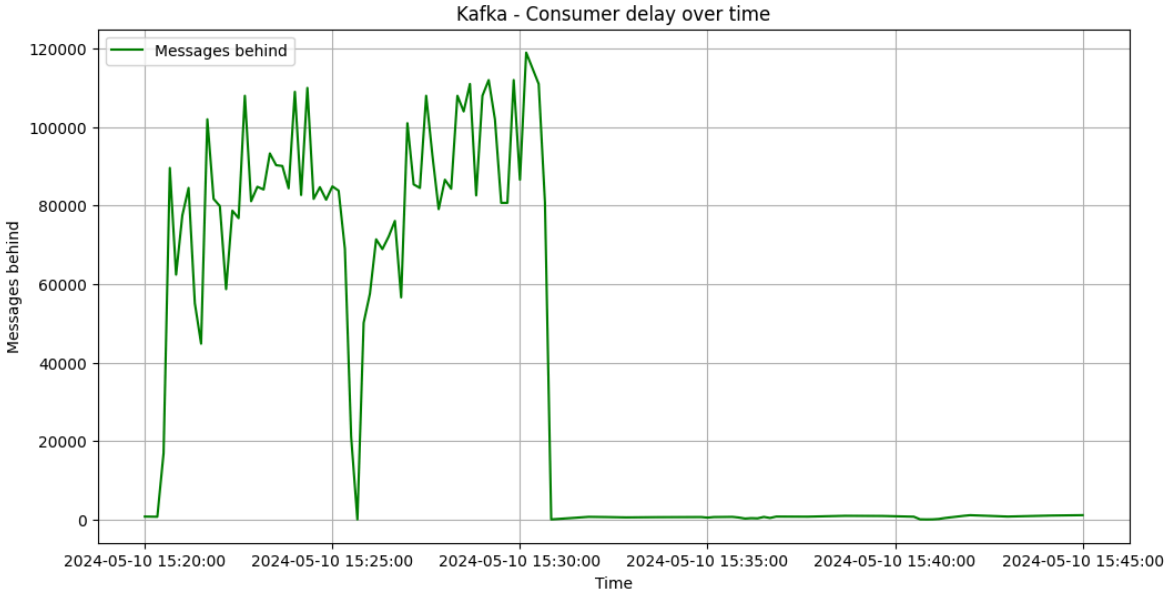
151 rows × 8 columns

In [11]:
```
fig_kafka_in_out_pod_count, ax1 = plt.subplots(figsize=(12, 6))
```

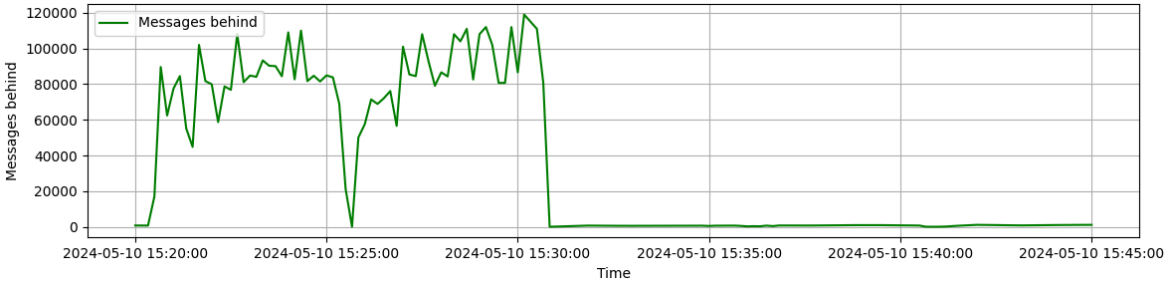
```
ax1.plot(data['Time'], data['Kafka - Messages ingested'], label='Messages
ax1.plot(data['Time'], data['Kafka - Messages consumed'], label='Messages
ax1.set_xlabel('Time')
ax1.set_ylabel('Messages (in and consumed)')
ax1.tick_params(axis='y')
ax1.xaxis.set_major_locator(MaxNLocator(nbins=6))
ax1.legend(loc='upper left')

ax2 = ax1.twinx()
ax2.plot(data['Time'], data['Vector - Pod count'], label='Vector - Pod co
ax2.set_ylabel('Pod count')
ax2.set_ylim(4,15)
ax2.tick_params(axis='y')
ax2.legend(loc='upper right')

plt.title('Kafka - Messages and Pod Count Over Time')
plt.grid(True)

plt.savefig('./without_opensearch/kafka_in_consumed_podcount_without_open
plt.show()
```

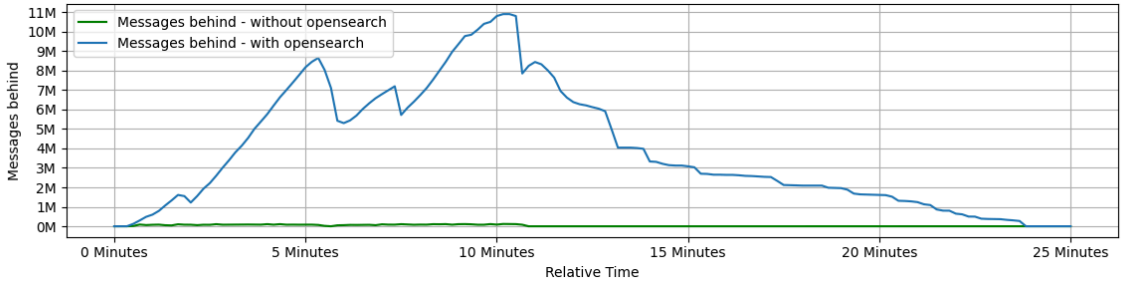


In [12]:
```
fig_kafka_consumer_delay, ax1 = plt.subplots(figsize=(12, 6))

# Plot Messages in and Messages consumed on the left y-axis
ax1.plot(data['Time'], data['Kafka - Messages behind'], label='Messages b
ax1.xaxis.set_major_locator(MaxNLocator(nbins=6))
ax1.set_xlabel('Time')
ax1.set_ylabel('Messages behind')
ax1.tick_params(axis='y')
ax1.legend(loc='upper left')

# Set title and grid
plt.title('Kafka - Consumer delay over time')
plt.grid(True)

plt.savefig('./without_opensearch/kafka_consumer_delay_without_opensearch
plt.show()
```

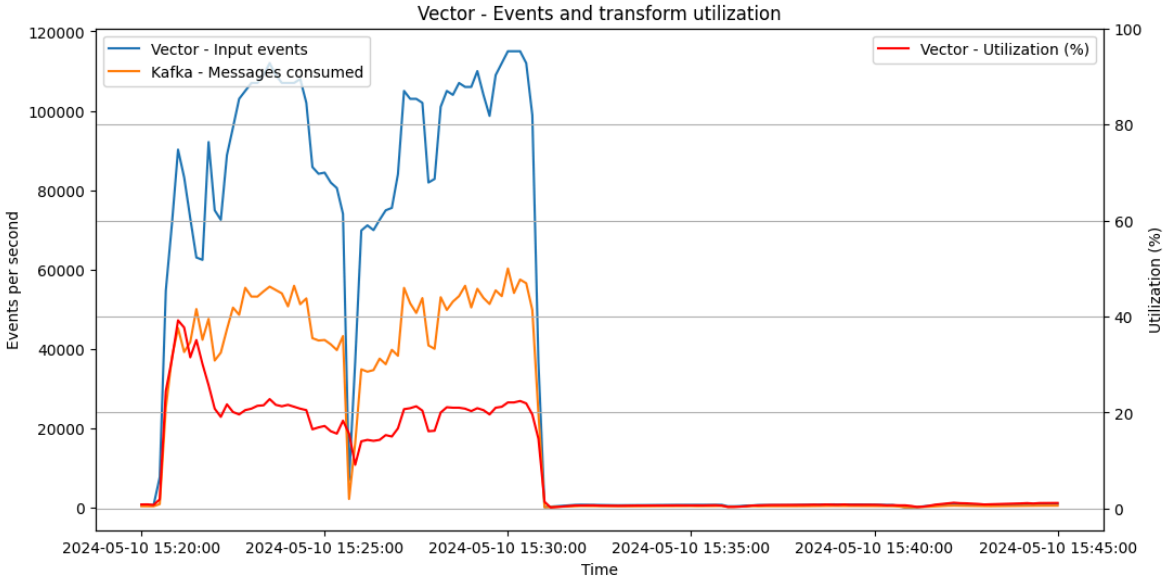


In [13]:
```
# Create temporary dataframes, and create relative timeframes
tmp_data = data.copy()
tmp_data_with_opensearch = data_with_opensearch.copy()
tmp_data_with_opensearch['Time'] = pd.to_datetime(tmp_data_with_opensearc
tmp_data['Time'] = pd.to_datetime(tmp_data['Time'])
tmp_data_with_opensearch.set_index('Time', inplace=True)
tmp_data.set_index('Time', inplace=True)
tmp_data_with_opensearch['Relative Time'] = (tmp_data_with_opensearch.ind
tmp_data['Relative Time'] = (tmp_data.index - tmp_data.index.min()).total

# Create figure
fig_kafka_consumer_delay_comparison, (ax1, ax2) = plt.subplots(nrows=2, n

# Plot standard vector consumer delay
ax1.plot(data['Time'], data['Kafka - Messages behind'], label='Messages b
ax1.xaxis.set_major_locator(MaxNLocator(nbins=6))
ax1.set_xlabel('Time')
ax1.set_ylabel('Messages behind')
ax1.tick_params(axis='y')
ax1.legend(loc='upper left')
ax1.grid(True)

# Plot relative time comparison graph
ax2.plot(tmp_data['Relative Time'], tmp_data['Kafka - Messages behind'],
ax2.plot(tmp_data_with_opensearch['Relative Time'], tmp_data_with_opensea

ax2.xaxis.set_major_locator(MaxNLocator(nbins=6))
ax2.xaxis.set_major_formatter(plt.FuncFormatter(lambda x, pos: f'{int(x/6
ax2.yaxis.set_major_locator(MultipleLocator(1e6))
ax2.yaxis.set_major_formatter(plt.FuncFormatter(lambda x, pos: f'{int(x/1
ax2.set_xlabel('Relative Time')
ax2.set_ylabel('Messages behind')
ax2.tick_params(axis='y')
ax2.legend(loc='upper left')
ax2.grid(True)

plt.tight_layout()
plt.savefig('./without_opensearch/kafka_consumer_delay_comparison.pdf', f
plt.show()
```

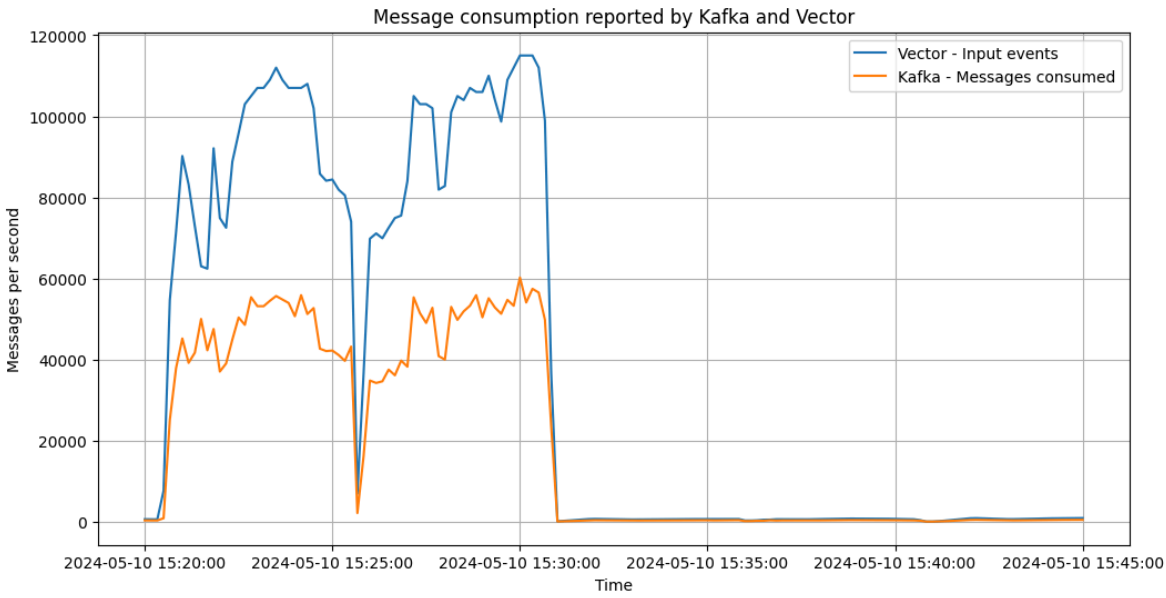




In [14]:
```
fig_vector_events_per_second, ax1 = plt.subplots(figsize=(12,6))

# Plotting each column
ax1.plot(data['Time'], data['Vector - Input events per second'], label='V
ax1.plot(data['Time'], data['Kafka - Messages consumed'], label='Kafka -
ax1.xaxis.set_major_locator(MaxNLocator(nbins=6))
ax1.set_ylabel('Events per second')
ax1.legend(loc='upper left')

ax2 = ax1.twinx()
ax2.plot(data['Time'], data['Vector - Geodata transform utilization perce
ax2.set_ylim(-4.5,100)
ax2.tick_params(axis='y')
ax2.legend(loc='upper right')

# Set titles, legend and grid
plt.title('Vector - Events and transform utilization')
ax1.set_xlabel('Time')
ax1.set_ylabel('Events per second')
ax2.set_ylabel('Utilization (%)')
plt.grid(True)

plt.savefig('./without_opensearch/vector_input_events_and_utilization_wit
plt.show()
```

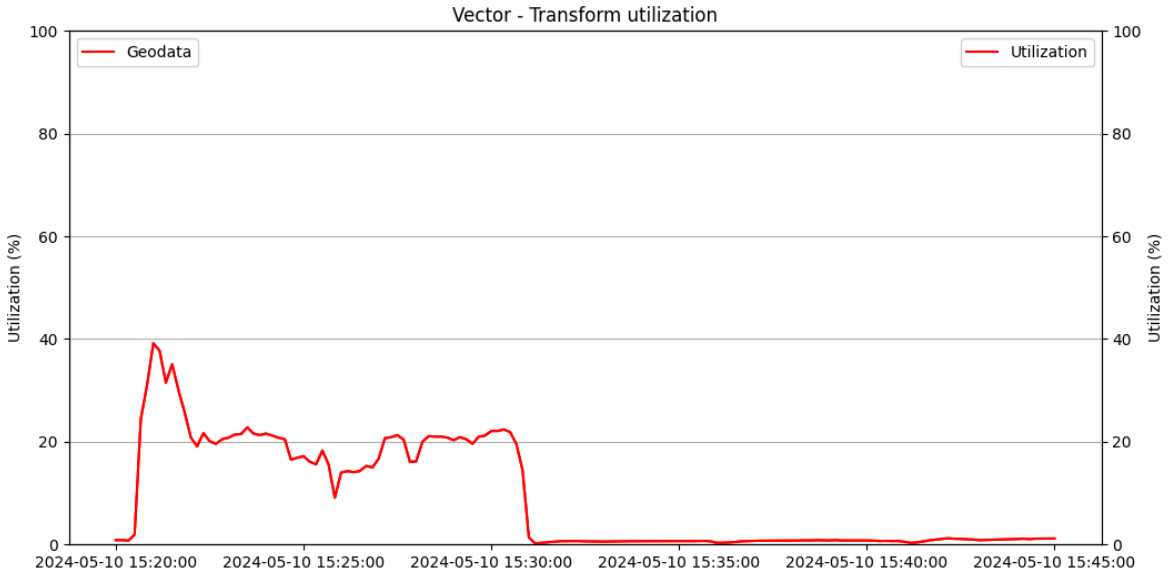


In [15]:
```
fig_kafka_vs_vector_events_per_second, ax1 = plt.subplots(figsize=(12,6))

# Plotting each column
ax1.plot(data['Time'], data['Vector - Input events per second'], label='V
ax1.plot(data['Time'], data['Kafka - Messages consumed'], label='Kafka -
ax1.xaxis.set_major_locator(MaxNLocator(nbins=6))

# Set titles, legend and grid
plt.title('Message consumption reported by Kafka and Vector')
ax1.set_xlabel('Time')
ax1.set_ylabel('Messages per second')
ax1.legend()
plt.grid(True)

plt.savefig('./without_opensearch/kafka_vector_message_consumption_compar
plt.show()
```



In [16]:
```
fig_vector_transform_utilization_and_rate, ax1 = plt.subplots(figsize=(12

ax1.plot(data['Time'], data['Vector - Geodata transform utilization perce
ax1.xaxis.set_major_locator(MaxNLocator(nbins=6))
ax1.set_ylabel('Utilization (%)')
ax1.set_ylim(0,100)
```

```
ax1.tick_params(axis='y')
ax1.legend(loc='upper left')

ax2 = ax1.twinx()
ax2.plot(data['Time'], data['Vector - Geodata transform utilization perce
ax2.xaxis.set_major_locator(MaxNLocator(nbins=6))
ax2.set_ylabel('Utilization (%)')
ax2.set_ylim(0,100)
ax2.tick_params(axis='y')
ax2.legend(loc='upper right')

plt.title('Vector - Transform utilization')
plt.grid(True)

plt.savefig('./without_opensearch/vector_geodata_transform.pdf', format='
plt.show()
```



In [16]:

Appendix D

Meeting Minutes

This appendix details the minutes of the project meetings with the academic supervisors, providing a chronological record of decisions, discussions, and progress made throughout the development of the log analytics pipeline.

Meeting Minutes

Møtereferat 15 Jan 2024

Oppmøte

Møtedato: [15. Jan 2024]

Møtetid: [11:00]

Møtested: [Teams]

Til stede

- Marcus Mathisen
- Fredrik Stenersen
- Karl-Henrik Horve
- Bjørn Kristian Strand

Fraværende

- Ingen

Agenda:

- Innledende møte for å få avklart grunnleggende punkter
- Planlegge forprosjektet

Saksliste:

Sak 1: Planlegging av forprosjekt

- Går gjennom eksempler av tidligere forprosjekt.

Background

- Marcus tar background

Project goals

- Performance Goals
- Result Goals
- Learning Goals

Framework

- Time frame 31th january for forprosjektet
- Frist for innlevering?

Scope

- Avventes

Roles

- Fredrik

Problem

- Skrive problemstillingen - denne kan gjøres av Marcus
- Problemavgrensing - avventes med intill vi er litt lengre inn i forprosjektet. Dette ble anbefalt av Christoffer Hallstensen

Risikoanalyse

- Karl-Henrik

Gannt chart

- Marcus

Sak 2: Delegerer ansvarsområder

- **Marcus Mathisen** som gruppeleder og kommunikasjonsansvarlig
- **Fredrik Stenersen** som sekretær
- **Bjørn Kristian Strand** som dokumentansvarlig
- **Karl-Henrik Horve** som QA - Kvalitetssikringansvarlig

Sak 3: Diskuterer prosjektstyringsapplikasjon

- Det foreslås
  - Jira
  - Microsoft Tasks
  - Trello
  - Gitlab Issues
- Konkluderer med gitlab issues for fin integrasjon med koden og for å unngå for mange applikasjoner

Sak 4: Arbeidsmetodikk

- Det foreslås litt forskjellig
- Marcus tar dette opp med Erik på møtet i morgen

Sak 5: Grupperegler

- Arbeidsmengde - min 30 timer i uken
- Timeføring - Skal føres ukentlig
- Ukentlig worksession - jobber fysisk sammen 1 gang i uka. dette må ha litt slingrinsmånn.
  - Fredrik legger inn jobb kalenderen sin.
- Statusmøter tar vi litt sporadisk, med litt lenger intervaller.
- Møte med veileder, Møte med Christopher. plikt å møte opp. 1 gang i uka
- Skrives referat for alle møter

Neste møte:

Agenda:

- Første møte med veileder Erik Hjelmås
- Skal få mer info om Jørn
- Generell orientering
- Arbeidsmetodikk

Oppmøte

- Dato: [16. Jan 2024]
- Tid: [12:00]
- Sted: [Teams og fysisk]

Referent: Fredrik Stenersen

Møtereferat 16 Jan 2024

Oppmøte

Møtedato: [16 Jan 2024]

Møtetid: [12:00]

Møtested: [Teams/fysisk]

Til stede

- Marcus Mathisen
- Fredrik Stenersen
- Karl-Henrik Horve
- Bjørn Kristian Strand

Veileder:

- Erik Hjelmås

Fraværende

- Ingen

Agenda:

- Første møte med veileder Erik Hjelmås
- Mer info om Jørn
- Innlede spørsmål til forprosjekt

Saksliste:

Sak 1: Orientering av Veileder

- Vi har startet
- Hatt møte med Christoffer
  - Fått en beskrivende problemstilling,
  - Hvilken resultater de forventer
  - Gjenstår å få dette ned på papir

Sak 2: Arbeidsmetodikk

- Kanban metoden
  - Fordi det er stort omfang
    - Vi har ingen klare avgrensninger
    - Scrum hadde virket ulogisk
- Har bedt om tilgang til roadmap i gitlab
- Får tilbakemelding fra Erik at dette høres fornuftig ut
- Gitlab eller github
  - Christoffer vil at vi skal bruke gitlab, det vi har begynt med

Sak 3: Informerer om møtestruktur

- Ingen faste møtetider
  - Vi praktiserer heller faste arbeidsmøter
  - Så å si ukentlige møter med Christoffer

Sak 4: Jørn

- Begynner i 20% stilling fra og med februar
- Jobber egentlig i orange som sysadmin i 15 år, veldig nyttig faglig person
- Kommer til Gjøvik en dag i uken
- Erfaring med å jobbe som sensor på dataingeniør studiet

Sak 5: Øvrige spørsmål til veileder

- Karl-Henrik tar opp spørsmål vedrørende muligens overflødig "fluff"

- Hvor er barriærene/hvor siterer vi? Hvorfor velger vi en tjeneste over en annen? Hvordan jobber vi med kilder og siterer her?
  - Fra Erik: "Det er en balanse, vi må bevise at arbeidet er fornuftig, det er en bacheloroppgave, ikke en filosofisk doktorgrad, det skal være vitenskap i arbeidet, der vi kommer med sterke påstander i teksten ber han oss underbygge det."
  - Ber oss gi utkast løpende; der vi hører om vi treffer riktig med sitering og kildehenvisning.
- Holder det at vi skriver om hvordan implementeringen har gått?
  - Vi behøver ikke skrive og sammenligne i alle ledd, store deler av oppgaven er at vi skal utrede og lage en testimplementasjon av opensearch, da er det jo hvilken.

> *"Pass på at vi ikke skriver det som en historiefortelling"*

> Det er ikke vitenskapelig riktig å skrive som en ungdomskolestil, mest mulig nåtid, styr unna personlige pronomen, se på tidligere rapporter

- Må vi scope ut alt i forprosjektet?
  - Sett opp en fremdriftsplan og en dimensjonering vi tror på. I slutten av oppgaven drøfter vi og reflekterer over hvorfor det ble avvik fra planen.

> Loggin er et tema det sannsynligvis finnes det mye vitenskapelige rapporter på. Dette er informasjon vi kan bruke.

Dos and donts fra Erik

- Begynn å skriv tidlig
- Bruk overleaf fra dag 1
- Risikovurdering av prosjektet og grupperegler
- "Best practice er stjel av andre"

Gode oppgaver: var en god oppgave

> "Securing the software development life cycle"
> "Tick-stack"
> "CI/CD pipeline"

Fremover

- Standardavtalen må fylles ut i løpet av **Januar**.
- Christoffer skal sjekke om vi trenger en utvidet konfidensialitetsavtale utover det som står i standardavtalen.
- Send gjerne utkast av rapporten tidlig i **april** men også fortløpende
  - Minst ett ferdig kapittel
  - Må ikke være ferdig
  - Sent utkast når vi ønsker det i mai, senest **to uker før levering, 5-10 mai området.**
- Erik informerer nærmere om hvilken dager Jørn er her

Referent: Fredrik Stenersen

Møtereferat 29 Jan 2024

Oppmøte

Møtedato: [29 Jan 2024]

Møtetid: [13:00]

Møtested: [Teams/fysisk NTNU SOC]

Til stede

- Marcus Mathisen
- Fredrik Stenersen
- Karl-Henrik Horve
- Bjørn Kristian Strand

Fraværende

- Ingen

Oppdragsgiver tilstedet:

- Christoffer
- Hans Åge

Agenda:

- Status-update for forprosjekt innlevering.
- Standardavtale

Saksliste:

Sak 1: Forprosjektets status

- Informerer Christoffer og Hans Åge om problemstillingen og rammene vi har satt
- Hovedpunktene i forprosjektet blir lagt frem
- Container based enrichment
- Kan få mye av konfigurasjonen for ansible fra oppdragsgiver
- Legger frem pipelinen Karl-Henrik har tegnet:
  - Diskuterer hvordan Middleware løses
  - Kafka inn i NiFi
  - Nifi kjøres best utenfor cluster, liker egne fysiske servere, bruker alt I/O som er tilgjengelig.

Sak 2: Generell diskutering av teknisk implementasjon

- Kafka, Nifi, inni et kubernetes-cluster
  - Endrer problemstillingen til at dette blir primærfokuset, tenk dataflyt
- Hvordan gjøre dataflyten dynamisk, visst de vil dele data med noen kan de legge inn en link i nifi, slik at dataflyten deles opp.
- Dataprepper istedet for logstash, man kan kanskje sette opp begge om vi vil.

Hva er treshholden deres?

- Gitlab runnere brukes en god del, de tenker å fortsette å bruke det, men vi har frihet til å velge det vi syns passer best.
- Ser på muligheten for å splitte opp tjenester i flere repo.

Statefile håndtering?

- Bør vi finne en annen måte å håndtere statefiles?
  - Nei, tror ikke det.

Skallerbart

- Hvordan skalerer vi noe når det er snakk om en logginginfrastruktur?
  - Legger på nye noder når det trengs mer kraft.
  - Opensearch og Elasticsearch liker x antall ytelse vs. lagring, de har et par flavours av noder som legges på.
- "Kaster penger på problemet" - Mer noder. mer kraft.

Dummy data

- Enda ikke bestemt om vi får tilgang til reell data
  - For nå blir det autogenerert data.

ISTIO

- Blir informert om at dette er noe vi kan ta ibruk

Sak 3: Øvrige spørsmål

Tilgang til cyber rangen

- Når det ikke er i bruk får vi tilgang til fasilitetene.

Hovedproblemet

> Dataflyt som skal kunne skaleres opp eller ned etter behov

Hvor fort bør vi ha en Minumum Viable Product?

- Til påske
- Tiden etter påske går ganske fort.

Referent: Fredrik Stenersen

Møtereferat 29 Jan 2024 Internt

Oppmøte

Møtedato: [29 Jan 2024]

Møtetid: [15:00]

Møtested: [Teams/fysisk]

Til stede

- Marcus Mathisen
- Fredrik Stenersen
- Karl-Henrik Horve
- Bjørn Kristian Strand

Fraværende

- Ingen

Agenda:

- Generelt møte om fremgangen og veien fremover mot innlevering av forprosjektet.

Saksliste:

Sak 1: Generelt - Forprosjekt

- Kalle omskriver problemstillingen

Sak 2: Splitting av repo

- Splitter de forskjellige delene opensearch, producers til kafka, fluentbit etc

Sak 3: Formelle mangler i forprosjektet

- Mangler risikomatrise
  - Konsensus om at dette er relativt likt på tvers av grupper
- Mangler gant bilde

Sak 4: Faste møter og arbeid

- **Sikter på onsdager som arbeidsdag fysisk**
- Sikter inn på å ha fast møte med veileder de dagene Jørn er på campus

---

Referent: Fredrik Stenersen

Møtereferat 31. Jan 2024

Oppmøte

Møtedato: [31. Jan 2024]

Møtetid: [12:00]

Møtested: [T431]

Til stede

- Marcus Mathisen
- Karl-Henrik Horve
- Bjørn Kristian Strand

Fraværende

- Fredrik Stenersen (Jobb)

Veileder:

- Erik Hjelmås
- Jørn Skjerven

Saksliste:

Sak 1: Prosjektplan - Problemstatement

- Omskriving av problem-statement
  - ingen kommentar fra veileder

Sak 2: SkyHigh

- SkyLow og SkyHigh Nede neste uke
- Lite å gjøre med det

Sak 3: Generelt

- Fokusere på teoretisk
- K8s as a service
- Se på å bruke openstack sin K8sAAS for å deployment av kubernetes-cluster
- Eventuellt-saker Møter
  - Mulig med møte med oppdragsgiver og veiledere hvis ønskelig / forskjellige oppfatninger o.l.
- 2 ting å notere
  - Hvem har gjort dette før? / mest lignende? Tegn figurer
  - Bør kunne tegnes for forståelse og presentasjon

Sak 4: Tilgjengelighet

- Bør repoet være closed source
- Skal bare være vi i gruppen som har tilgang på det
- Har uansett ingenting å si fra et plagiat standpunkt

Sak 5: Docker images

- Hvordan skal docker images lages, bygges og lagres
- Skal data inne i images være hemmelige eller kan det være offentlig tilgjengelig Er dette noe vi må ta for oss i prosjektet, eller er det en tjeneste de tar hånd om?

Sak 6: Kubernetes

- Hvordan rulle ut en applikasjon
- Helm vs kubernetes-native?

Sak 7: Goals

- Omskriv performance goals -> effect goals
- Hvilken impact vil prosjektet ha på NTNU SOC
- Result goals
- For generisk formulert
- Skreddesy det mer til prosjektet, selvom det kan virke repetitivt
- Numerer målene, ikke bruk punktliste

Referent: Marcus Mathisen

Møtereferat 7 Feb 2024

Oppmøte

Møtedato: [7 Feb 2024]

Møtetid: [13:50]

Møtested: [Teams]

Til stede

- Marcus Mathisen
- Fredrik Stenersen
- Karl-Henrik Horve
- Bjørn Kristian Strand

Fraværende

- Ingen

Agenda:

- Starte planleggingen av infrastrukturen:
- Utarbeide et kart over hvordan vi har tenkt å sette opp infrastrukturen, inkludert detaljnivået på parsing, antall klynger, namespaces i Kubernetes, og strukturering av Git-prosjektet for å oppnå det endelige målet.

Saksliste:

Sak 1: Infrastruktur

Notater fra diskusjonen om infrastruktur:

- Vi har besluttet å bruke GitLab og GitLab Runners for å hoste koden og for Continuous Integration (CI) delen.

Hva skal inkluderes innenfor og utenfor containere?

- Hovedsakelig ønsker vi å containerisere alt i clusteret.
  - Hvis vi møter utfordringer med å containerisere Nifi og Kafka, vil vi vurdere å kjøre dem utenfor containermiljøet.

Sak 2: Kort Roadmap

- Vi diskuterte veien videre.
  - Fokusområdet blir å få alle til å forstå Kafka først.
  - Deretter vil vi sette opp grunnleggende infrastruktur for å integrere Fluentbit og Kafka.

Sak 3: Hva har vi gjort?

- Vi har delt opp repoet i to separate deler:

Core Infra - Dette inneholder nødvendig infrastruktur for GitOps og CI/CD.
Kubernetes - Dette inneholder konfigurasjonsfiler og manifest for Kubernetes-deploys.

Sak 4: Vi må bestemme antall køer som er nødvendig

- Vi har besluttet å starte den tekniske implementasjonen med en kø for å få systemet opp og kjøre.

Neste møte:

Agenda:

- Hva har vi gjort til nå
- Hva skal vi gjøre videre?

Oppmøte

- Dato: [8. Feb 2024]
- Tid: [12:00]
- Sted: [Teams/T540]

Referent: Fredrik Stenersen

Møtereferat 8 Feb 2024

Oppmøte

Møtedato: [8 Feb 2024]

Møtetid: [12:00]

Møtested: [Teams/T540]

Til stede

- Marcus Mathisen
- Fredrik Stenersen
- Karl-Henrik Horve
- Bjørn Kristian Strand

Veileder:

- Erik Hjelmås
- Jørn Skjerven

Fraværende

- Ingen

Agenda:

- Hva har vi gjort til nå
- Hva skal vi gjøre videre?

Saksliste:

Sak 1: Informasjon til Veiledere om Utførte Oppgaver

Presentasjon av Git Oppsett:

- Implementering av GitLab Runners
- Bruk av Ansible
- Oppretting av Terraform Statefile via Ansible

Sak 2: Forprosjekt

- Visning av Gantt-diagrammet

Sak 3: Prosjektstyring

- På grunn av nedetid på GitLab kunne ikke fremgangsmåten for forprosjektet demonstreres
- Visning av video av Issue Boards

Sitat:
"En god bacheloroppgave inneholder mange egendefinerte tegninger og diagrammer."

Sak 5: Fast Møtetidspunkt

- Enighet om fast møtetidspunkt på torsdager kl. 12:00 passer for alle deltakere.

Neste møte:

Agenda:

- Statusmøte

Oppmøte

- Dato: [15 Februar 2024]
- Tid: [12:00]
- Sted: [Teams/T540]

Referent: Fredrik Stenersen

Møtereferat 22 Feb 2024

Oppmøte

Møtedato: [22 Feb 2024]

Møtetid: [12:00]

Møtested: [T540]

Til stede

- Marcus Mathisen
- Fredrik Stenersen
- Karl-Henrik Horve
- Bjørn Kristian Strand

Veileder:

- Erik Hjelmås
- Jørn Skjerven

Fraværende

- Ingen

Agenda:

- Fremvise skisse for tiltenkt infrastruktur
- Status den siste uken

Saksliste:

Sak 1: Fremviser skisse for infrastrukturen

- Vi holder på med kafka og nifi for øyeblikket
- Begynner sakte med fluentbit og opensearch
- **Tilbakemelding**:
  - Sier ikke så mye om selve dataflyten
  - Det kan være fint å ha et diagram som viser dataflyten mellom mikrotjenester med forklaring
    - Dette kan hjelpe i rapporten
  - Ønsker en dataskisse for hvordan dataflyten henger sammen
  -
- Vi har tenkt til å lage en skikkelig graf for dependencies

  > grafwiz for dependencies

Sak 2: Problematikk relatert til persistent storage

-

Sak 3: Hvordan ligger vi ann i forhold til planen?

- Vi ligger litt etter på selve rapportskrivingen
- Infrastruktur/implementasjonsmessig ligger vi greit ann
  - Forholder oss til gitlab issues når det kommer til prosjektstyring.

Sak 4: Referansearkitekturen

- Vi ser for oss å ha proof of concept ferdig til påske
  - NB: Påsken kommer tidlig i år!

Referent: Fredrik Stenersen

Møtereferat 29 Feb 2024

Oppmøte

Møtedato: [29 Feb 2024]

Møtetid: [12:00]

Møtested: [Teams/fysisk]

Til stede

- Marcus Mathisen
- Fredrik Stenersen
- Karl-Henrik Horve
- Bjørn Kristian Strand

Veileder:

- Erik Hjelmås
- Jørn Skjerven

Fraværende

- Ingen

Agenda:

- Status for teknisk implementasjon
- Tilbakemeldinger fra veileder så langt på rapporten
- Roadmap

Saksliste:

- Sak 1: Rapporten - Tilbakemeldinger
- Sak 2: Dypt tekniske problemstillinger? Med i rapporten?
- Sak 3: Roadmap

Sak 1: Rapporten - Tilbakemeldinger

- Bruk vitenskapelige kilder, gjerne bøker.
- Kubernetesteori - bruk rapporten som Erik viste fra google
  - Gjerne kubernetes.io
- WOKE ALARM!
  - Ikke kall ting for master/slave (det er ikke woke)
- Generell rapportkvalitet er viktig
- Tenk gjennom om all teorien vi skriver er relevant for VÅR oppgave.
- Når vi prater om noe teoretisk, gjerne beskriv hvordan vi bruker det i prosjektet.

- Om noe er kort nok til å beskrives i en setning kan det være i en glossary.
- Kapittel 2 (teorikapittelet) er gjerne et kapittel man går tilbake til å skriver om/legger til/reviderer.
- Gjerne spor opp den opprinnelige kilden til noe. Sensorene er dyktige og vil gjerne se/huske igjen en opprinnelig kilde.
- Det er lov å bruke fotnoter, men fem på en side er mye. Generelt bryter dette leseligheten

Sak 2: Dypt tekniske problemstillinger? Med i rapporten?

- Ikke kast dokumentasjonen
  - Tenk at noen skal ta over arbeidet vårt, ville de forstått nok?
  - Ta credit for arbeidet vårt.
- Dersom vi ikke oppnår ønsket resultat.
  - Det kan like godt være en A oppgave.

Sak 3: Roadmap

- Får tilbakemelding om at vi er i et godt spor - fortsett i det.

Referent: Fredrik Stenersen

Møtereferat 4 Mars 2024

Oppmøte

Møtedato: [4 Mars 2024]

Møtetid: [14:00]

Møtested: [Teams/A019 Møterom SOC]

Til stede

- Marcus Mathisen
- Fredrik Stenersen
- Karl-Henrik Horve
- Bjørn Kristian Strand

Oppdragsgiver:

- Christoffer Vargtass Hallstensen

Fraværende

- Ingen

Agenda:

- Presentere fremgangen
- Forklare valgene våre
- Hva vi tenker å gjøre videre.
- Innspill fra oppdragsgiver
- Spørsmål

Saksliste:

Sak 1: Presenterer fremgangen til nå og forskjellige problemstillinger

Gir en oppdatering til oppdragsgiver om fremgangen til nå.

Raft vs zookeper?

- Så lenge kafka funker er det egentlig det samme, funker det med zookeeper og det inngår i en POC er det greit for oppdragsgiver

JSON
JSON er å foretrekke det det passer best med de forskjellige tjenestene. Ex. opensearch lager key/value pairs basert på dette lett.

NiFi vs Vektor.

- NiFi er til å stole på.
- Trenger ikke være bleeding edge på alle punkter
- Det er viktigere at det funker enn at det er kult.

Problematikk med openstack vs andre skyløsninger.

- Vi så ikke for oss at git.gvk hadde de problemene som de hadde. Så for oss at opensearch var plug and play og at certs var noe vi kunne få utlevert enkelt. Men vi føler det er mye som går utenfor scope, som grunner i openstack som plattform.
  - Tenker det er viktig lærdom for SOCen sier Christoffer, vi bør dokumentere det i rapporten.

Sak 2: Ressurser i Openstack

Forespør mer ressurser på bakgrunn av at det begynner å bli lite igjen og autoscaling får vi ikke testet noe særlig da det ikke er for eksempel flere kjerner igjen.

Sak 3: Interne og eksterne sertifikater

- Kjører en intern CA for å unngå lekkasje ved bruka av tredjeparts CA.

Sak 4: UI interface?

- Med tanke på at det er en proof of concept trenger vi ikke fokusere på å konfigurere dette utover det som brukes i f.eks. NiFi by default.

Sak 5: Husk fokus på herding av kubernetes.

- Populært spørsmål fra en eventuel sensor er hvordan vi har tenkt på sikkerhet.
- **Ha et avsnitt i rapporten om herding av kubernetes.**

Referent: Fredrik Stenersen

Møtereferat 7 Mars 2024

Oppmøte

Møtedato: [7 Mars 2024]

Møtetid: [12:00]

Møtested: [Teams/T540]

Til stede

- Marcus Mathisen
- Fredrik Stenersen
- Karl-Henrik Horve
- Bjørn Kristian Strand

Veileder:

- Jørn Skjerven

Fraværende

- Ingen

Agenda:

- Gå gjennom nylig fremgang
- Diskutere problemstillinger vi har støtt på

Saksliste:

Sak 1: Problemer med nifikop operatoren og tillatelser

- Diskuterer måter å løse dette for å få funksjonalitet på plass.

Sak 2: Problematikk med opensearch

- Har fått inntrykket fra før av fra oppdragsgiver om at dette skulle være plug-and-play, det har derimot ikke vært tilfellet til nå.
- SOCen hadde vært borti noe av problematikken fra før av når det kommer til templates for openstack.

Sak 3: Igjen vurderer vi vector.dev i stedet for dataprepper

- Kommunikasjonsvikt fra forrige uke da vi hadde trodd det var snakk om vektor.com.
- Ser på muligheter for å gjøre datatransofmasjoner med vector.dev som et alternativ for dataprepper.

Sak 4: Cribl som et alternativ for nifi

- Diskuterer dette som en plan B.

Sak 5: Forespør tips om DNS og certifikatdistributør

- Ingen gode gratis alternativer
- Ser på muligheter om å forespør en offentlig IPv4 addresse fra NTNU på begrenset tid.

Referent: Fredrik Stenersen

Møtereferat 14 Mars 2024

Bør renskrives

Oppmøte

Møtedato: [14 Mars 2024]

Møtetid: [12:00]

Møtested: [Teams/T540]

Til stede

- Marcus Mathisen
- Fredrik Stenersen
- Karl-Henrik Horve

Veileder:

- Jørn Skjerven

Fraværende

- Bjørn Kristian Strand

Agenda:

- Presentere fremgangen og nylige problemstil
- Innspill fra oppdragsgiver
- Spørsmål

Saksliste:

Sak 1: Kubernetes plattform

- Diskuterer kubernetes på bare metal vs. openstack.
- Konkluderer med at ingen endring i planen er nødvendig.

Sak 2: Hvilke logger skal sendes?

- Syslog? Hva er hensiktsmessig?
- NTNU SOC har ingen spesifikke krav til prosjektets logger i seg selv, det er POC arkitekturen som er hovedleveransen.

Sak 3: Fremviser vellykket logg i Openstack

- Fremviser en enkel syslog som har gått gjennom pipelinen og inn i opensearch.
  - Fra kafka, gjennom Vector og inn til OpenSearch
- Forklarer hvordan arkitekturen funker til nå.

Sak 4: Stresstesting av robusthet

- Skalering og stresstesting, diskuterer måter å simulere ballast
- Filtrerings-funkjonalitet?
  - "Dette" mønsteret skal filtreres bort
  - Blackholing
- Skallering for å takle ballast
  - Horisontalt/vertikalt
  - Flere clustere? (Brainstorming fra Jørn)
    - Hva er av hensikt for socen? Er det grunn for å segmentere logger? Er det av interesse at noen systemer er veldig kritisk eller at andre KAN gå ned.
- Vurdere forskjellige problemstillinger å skrive i rapporten på dette punktet.

Sak 5: Tips til rapportskriving fra Jørn

- Ikke skriv som en dagbok
- Selv om vi potensielt ikke får til det vi ønsker å få til produktmessig er det ikke nødvendigvis stryk av den grunn
  - Det er prosessen som er viktig, ikke resultatet.
  - Forklar hva, hvorfor, hvordan vi gjør ting.
- Kildehenvisning

Referent: Fredrik Stenersen

Møtereferat 21 Mars 2024

Oppmøte

Møtedato: [21 Mars 2024]

Møtetid: [12:00]

Møtested: [Teams/T540]

Til stede

- Marcus Mathisen
- Fredrik Stenersen
- Karl-Henrik Horve
- Bjørn Kristian Strand
  Veileder:
- Erik Hjelmås
- Jørn Skjerven

Fraværende

- Ingen

Agenda:

- Gjennomgå rapporten
- Gå gjennom tilbakemeldingene fra Eirik
  - (Markeringer i rapporten vi har sendt til gjennomgang)
- Statusoppdatering

Saksliste:

- Sak 1: Tilbakemeldinger til rapporten

Sak 1: Tilbakemeldinger til rapporten

- Språk, ordlyd, noen korrigeringer
  - **LTeX** vscode extension blir foreslått fra Eirik.
- "Det skinner gjennom at vi skriver selv, at vi ordlegger oss med egne ord, dette er bra."
- Vi skal sitere etter IEE eller ACM citation style, pr. nå gjør vi ikke det.
- Dropp forkortelser, vi har ikke dårlig plass.
  - Ex. "k8s".
- Husk vektorgrafikk i figurer (svg fil)
- **Ikke** bruk master og worker/slave. Bruk main, control plane eller primary etc.
- Husk: Tilstreb å henvis til logoer og bilder som for eksempel er under creative commons lisenser.
- *(Dette er pirk, mange bsc. oppgaver synder på dette, men greit å gjøre det riktig)*
  - Ex. Openstack Cinder logo
- Når alt er helt ferdig, først DA får vi det til å se pent ut.
  - Gå gjennom punktene fra Erik ang. skrifttype f.eks.
- Dersom rekkefølge ikke er viktig - bulletpoints, om rekkefølgen er viktig, bruk nummerert liste
- Descriptionlist i latex kan være et alternativ til fet skrift i punktlister.
- Stor forbokstav etter punktum.

Referent: Fredrik Stenersen

Møtereferat 4 April 2024

Oppmøte

Møtedato: [4 April 2024]

Møtetid: [12:10]

Møtested: [Teams/T540]

Til stede

- Fredrik Stenersen
- Karl-Henrik Horve
- Bjørn Kristian Strand

Fraværende

- Marcus Mathisen (Gyldig, medisinsk)

Agenda:

- Statusoppdatering siden påske.

Saksliste:

- Sak 1: Statusoppdatering etter påske

Sak 1: Statusoppdatering etter påske

- **Grafana Oppdateringer:** Grafana har blitt oppdatert nylig, med fokus på designen av Grafana KCX for bedre visualisering av data.
- **Raftmode Kafka Overgang:** Overgangen til Raftmode Kafka har vært underveis, og det er behov for å se nærmere på hvordan dette samsvarer med den nye Grafana KCX.
  - Vi ser også på hvordan loggenerering oppskaleres, spesielt med tanke på å gjøre trafikken mer realistisk for Grafana KCX.
- **Skaleringsspørsmål:**
  - Hvordan skalerer vi metrikker?
  - Plan for autoskalering, som for øyeblikket er statisk.
  - Vektor skalerer basert på CPU-bruk.
  - Geo-database i en pod, skalerer den basert på CPU-bruk eller latency?
- **Status på Påskens Rapportskriving:**
  - Det har ikke blitt skrevet noe i påsken i rapporten. Vi trenger en oppdatering om fremdriftsplanen og statusen der. Det bør også nevnes at en person har vært ute, og dette må tas med i rapporten.

Referent: Fredrik Stenersen

Møtereferat 11 April 2024

Oppmøte

Møtedato: [11 April 2024]

Møtetid: [12:00]

Møtested: [Teams/T540]

Til stede

-
- Fredrik Stenersen
- Karl-Henrik Horve
- Bjørn Kristian Strand

Fraværende

- Marcus Mathisen (Gyldig Medisinsk)

Agenda:

- Oppdatering fra gruppemedlemmer
- Tilbakemelding på rapport

Saksliste:

Sak 1: Oppdatering fra gruppemedlemmer

- Deployment pipelines i demomiljø
- K6 log simulator
- Geodata enrichment
- Rapport oppdateringer

Sak 2: Tilbakemelding på rapport

- Gjerne bruk analogier
- Bibliografilisten er feil
  - Henviser du til bare en nettside kan den puttes i fotnote
    - Derimot spesifikt del på nettside kan du bruke bibliografien
  - Er det flere deler av siden som er brukt kan du introdusere avsnittet med det og legge til som fotnote.

- Ikke bruk et. al. inne i teksten. Det er ønskelig å ha alle navn i bibliografien.

Referent: Fredrik Stenersen

Møtereferat 24 April 2024

Oppmøte

Møtedato: [24 April 2024]

Møtetid: [12:00]

Møtested: [Teams/T540]

Til stede

- Marcus Mathisen
- Fredrik Stenersen
- Karl-Henrik Horve
- Bjørn Kristian Strand

Veileder:

- Erik Hjelmås
- Jørn Skjerven

Fraværende

- Ingen

Agenda:

- Gjennomgå spørsmål til rapporten
- Veien videre

Saksliste:

Sak 1: Veiledere har ikke fått tid til å lese gjennom rapporten

- Vi har blitt enige om å sende en endringslogg når vi skal ha tilbakemelding på rapporten
- Vi har blitt enige om mandag 6. Mai dato for gjennomgang av første utkast med veiledere
- Enighet om at vi tar peer review med en annen gruppe først
  - Dermed sende endret versjon til veiledere.
- Flytte møtet de to neste torsdagene, til ett møte 6. Mai

Sak 2: Hva gjør vi om vi ikke har tid til å fikse data?

- Vi må i all hovedsak bare skrive om det vi har gjort
- Husk at det er en muntlig fremføring der det er mulighet til å gjennomgå ting om det ikke skulle være mulig å få med i rapporten

Sak 3: Tenk kvalitet over kvalitet i skrivingen

Sak 4: Latex figurer på siden av tekst

- Blir det problematisk kan du ha dem på en egen side
  - Henvis dermed til figuren i teksten i stedet
  - Ikke noe vi ønsker å bruke tid på før rett før innlevering

Sak 5: Hvordan dokumentere valgene vi har tatt

- Husk å dokumenter hvorfor, hva funket, hvorfor ikke? Forklar kildene og hvordan vi brukte dem for å ta valgene vi gjør.
- Skriv hvordan vi systematisk løste problemene vi har møtt på
- Dokumenter at vi har gjort **noe**.
- Vi har jo brukt mye tid til å "gå forbi vegger" / feilsøking, kan godt nevne dette i rapporten, men resonner tilbake til dette, ikke trinn for trinn historiemessig.

Sak 6: Rapport: Husk å skrive om implementeringen

- Begynn å fokuser mer på hvordan vi har implementert det vi har gjort
- Metode, result, discussion

Referent: Fredrik Stenersen

Møtereferat 16. Mai 2024

Oppmøte

Møtedato: [16. Mai 2024]

Møtetid: [12:00]

Møtested: [Teams/fysisk]

Til stede

- Marcus Mathisen
- Fredrik Stenersen
- Karl-Henrik Horve

Veileder:

- Erik Hjelmås
- Jørn Skjerven

Fraværende

- Bjørn Kristian Strand

Agenda:

- Siste møte, gjennomgang av mangler og utbedringspotensiale

Saksliste:

Sak 1: Kapittel 4

- Mye tekst
- Bilde av arkitekturen hadde vært lurt (VIKTIG for forståelse)

Sak 2: Store grafer

4.20 er for stor, 4.19 er lesbar.

Sak 3: En del å jobbe med i kapittel 5 (sikkerhet)

- Hovedsakelig snakker om de tingene vi ville gjort annderledes og de viktigste delene generelt.
- Hva har vi gått på akkord med, hva er det som gjør at det ikke er produksjonsklart?

Sak 4: Grafer i kapittel 6.

- Det er veldig fine grafer, men forfatteren må beskrive og forklare grafene. Ikke bare skrive det samme du kan lese ut av grafen.
- Skiller gjerne A og B om du forankrer påstander i eksisterende litteratur.
- Ex. 6.10 Hadde vært kult å vise konkret hvor i pipelinen med en graf hvor i pipelinen flaskehalser oppstår.

Generelt:

- Unforseen setbacks 7.2.3. Hva gjorde vi for å komme oss forbi det.
- Kapittel 8. Her er det viktig at vi får med alle målene fra introduksjonen med tilhørende grad av hvordan vi har løst dem. (Ingen oppnår alle målene man setter). Drøft dette i kap 8.
- Gantt - en n to Ter
- Bibliografi. Sleng på en lærebok og en artikke. Ex kubernetes, da google borg - kubernetes.
- Tomme punkter - Bruk øks, eks. 7.7.6 har vi ikke pratet noe om det til nå, er det kanskje ikke så viktig.
- Åpenbare mangler: Få på plass helhet ovenfor det å putte inn noe ekstra punkter.
- Det skal tilstreves at det ikke ser ut som fire forskjellige har skrevet (fortellermessig). Veldig mye retroskpektivt språk, "was", "helped", "did".
- Få på plass future work.
- Bruk frode sin "rettemal" fra rapportskrivingskurset.

Referent: Fredrik Stenersen

Appendix E

Time sheet

Timeføring Karl-Henrik

Week	Day	Hours	Work	Week	Day	Hours	Work
	Monday				Monday	3	Meeting the client
Week 2	Tuesday			Week 5	Tuesday	5	Research
	Wednesday	2	Meeting with client		Wednesday	6	Methodology, meeting supervisors
	Thursday	2	Group Meeting		Thursday	5	Project plan
	Friday				Friday		
	Saturday				Saturday		
4	Sunday			19	Sunday		
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday				Monday	7	Research
Week 3	Tuesday	4	Meeting with supervisor, working on project-plan	Week 6	Tuesday	8	Research
	Wednesday	3	Research		Wednesday	8	Implementation FluxCD
	Thursday	5	Project plan		Thursday	8	Openstack, terraform
	Friday	5	Project plan		Friday	6	K8 cluster
	Saturday				Saturday		
17	Sunday			37	Sunday		
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	2	Meeting with supervisor, working on project-plan		Monday	8	Troubleshooting k8s
Week 4	Tuesday	7	Project plan	Week 7	Tuesday	6	Troubleshooting Openstack
	Wednesday	5	Project plan		Wednesday	5	Troubleshooting Openstack
	Thursday	6	Project plan		Thursday	7	Troubleshooting Openstack
	Friday	7	Project plan		Friday	8	Troubleshooting Openstack
	Saturday				Saturday		
27	Sunday			34	Sunday		
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	10	Implementation		Monday	6	Testing aggregation
Week 8	Tuesday	3	Dataprepper troubleshoot	Week 11	Tuesday	5	Troubleshooting k8s
	Wednesday	8	Troubleshooting Cinder		Wednesday	6	Troubleshooting k8s
	Thursday	7	Troubleshooting Cinder		Thursday	5	Troubleshooting k8s
	Friday	9	Certificates		Friday	4	Cleaning up flux config
	Saturday	10	Certificates		Saturday		
47	Sunday			26	Sunday		
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	2	Writing Thesis		Monday	7	TLS config
Week 9	Tuesday	9	Cleaning up repo's	Week 12	Tuesday	6	Logging to brokers
	Wednesday	8	Setting up Kafka		Wednesday	8	Implement Metric
	Thursday	9	Setting up Kafka		Thursday	4	Grafana config
	Friday	7	Setting up Kafka		Friday	4	Prometheus and Grafana config
	Saturday				Saturday		
35	Sunday			29	Sunday		
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	7	Openstack drivers		Monday	0	Easter
Week 10	Tuesday	8	Openstack drivers	Week 13	Tuesday	0	Easter
	Wednesday	9	Openstack drivers		Wednesday	0	Easter
	Thursday	7	Storage classes, Csi Cinder		Thursday	0	Easter
	Friday	6	Documentation		Friday	0	Easter
	Saturday				Saturday	0	Easter
37	Sunday			0	Sunday	0	Easter
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	7	Benchmarking pipeline		Monday	7	Writing Thesis
Week 14	Tuesday	6	Troubleshooting Autoscaling	Week 17	Tuesday	6	Writing Thesis
	Wednesday	7	Autoscaling Vector		Wednesday	7	Writing Thesis
	Thursday	8	Moving k6 to new k8 cluster		Thursday		Writing Thesis
	Friday	3	Testing the pipeline		Friday		Writing Thesis
	Saturday				Saturday	6	
31	Sunday			26	Sunday		
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	6	Change kafka replication		Monday	8	Writing Thesis
Week 15	Tuesday	7	Config vector/Geodata	Week 18	Tuesday	5	Writing Thesis
	Wednesday	5	Config vector/Geodata		Wednesday	6	Writing Thesis
	Thursday	7	Benchmarking pipeline Documentation		Thursday	1	Writing Thesis
	Friday	6			Friday		Writing Thesis
	Saturday				Saturday	5	
31	Sunday			30	Sunday	5	
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	10	Writing Thesis		Monday	6	Writing Thesis
Week 16	Tuesday	8	Writing Thesis	Week 19	Tuesday	7	Writing Thesis
	Wednesday	10	Writing Thesis		Wednesday	4	Writing Thesis
	Thursday	8	Writing Thesis		Thursday	7	Writing Thesis
	Friday	3	Writing Thesis		Friday		Writing Thesis
	Saturday				Saturday	9	Writing Thesis
39	Sunday			41	Sunday	8	Writing Thesis
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	6	Writing Thesis		Monday	6	Writing Thesis
Week 20	Tuesday	7	Writing Thesis	Week 21	Tuesday		
	Wednesday	6	Writing Thesis		Wednesday		
	Thursday	6	Writing Thesis		Thursday		
	Friday	5	Writing Thesis		Friday		
	Saturday	8	Writing Thesis		Saturday		
46	Sunday	8	Writing Thesis	6	Sunday		

Timeføring Marcus

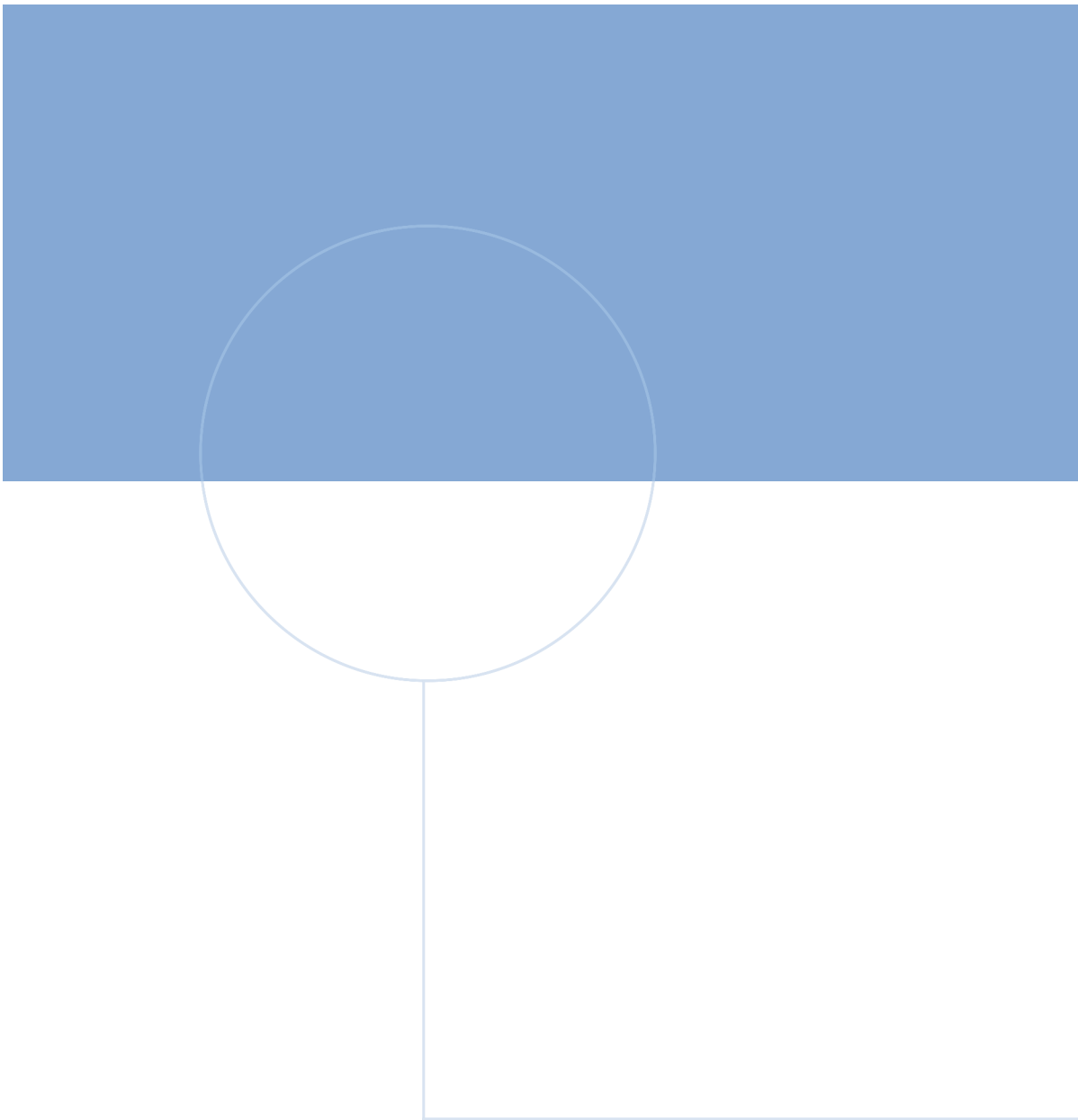
Week	Day	Hours	Work	Week	Day	Hours	Work
	Monday	0			Monday	4	Projectplan and meeting with client
Week 2	Tuesday	0		Week 5	Tuesday	7	Project plan Administrative tasks
	Wednesday	2	Meeting with client		Wednesda	4	Projectplan, meeting with supervisors
	Thursday	2	Group Meeting		Thursday	7	Project plan
	Friday	3	Researching thesis subject		Friday		
	Saturday	1	Researching project workflow		Saturday		
8	Sunday	0		22	Sunday		
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	3	Administrative tasks& Group meeting		Monday	5	Research and testing
Week 3	Tuesday	3	Research Meeting with supervisor	Week 6	Tuesday	7	Research and implementing
	Wednesday	4	Reserach		Wednesda	7	Reserch and seting up kafka locally
	Thursday	7	Writing in project plan		Thursday	4	Setting up kafka locally Meeting with supervisor
	Friday	5	writing in project plan		Friday	6	Research
	Saturday				Saturday		
22	Sunday			29	Sunday		
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	4	Writing in project plan		Monday	7	Implementing kafka
Week 4	Tuesday	3	Administrative tasks Meeting with supervisor	Week 7	Tuesday	7	Troubleshooting kafka in K8s
	Wednesday	0	Sick		Wednesda	7	Troubleshooting kafka in K8s
	Thursday	0	Sick		Thursday	7	Setting up GIT group and project management.
	Friday	0	Sick		Friday	3	Research
	Saturday	0	Sick		Saturday		
7	Sunday	0	Sick	31	Sunday		
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	7	Setting up kafka		Monday	6	Aggregation log
Week 8	Tuesday	7	Setting up Fluentbit	Week 11	Tuesday	5	Writing Thesis
	Wednesday	0			Wednesda	6	Writing Thesis
	Thursday	7	Connecting fluentbit to kafka brokers Meeting with supervisor		Thursday	5	Writing Thesis
	Friday	7	Writing in thesis		Friday	6	Writing Thesis
	Saturday	4	Writing in thesis		Saturday		
36	Sunday	4	Writing in thesis	28	Sunday		
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	7	Setting up kafka		Monday	7	Troubleshooting k8s
Week 9	Tuesday	8	Writing in thesis Administrating in GIT	Week 12	Tuesday	8	Troubleshooting k8s
	Wednesday	5	Writin and correcting in thesis		Wednesda	6	Troubleshooting k8s
	Thursday	6	Writing in thesis		Thursday	5	Fixing Kafka
	Friday	7	Writing in thesis		Friday	7	Fixing Kafka
	Saturday	0			Saturday		
33	Sunday	0		33	Sunday		
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	5	Setting up kafka		Monday	0	Easter
Week 10	Tuesday	6	Setting up kafka	Week 13	Tuesday	0	Easter
	Wednesday	4	Setting up kafka		Wednesda	0	Easter
	Thursday	6	Setting up kafka		Thursday	0	Easter
	Friday	6	Testing Kafka		Friday	0	Easter
	Saturday				Saturday	0	Easter
27	Sunday			0	Sunday	0	Easter
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	0	Injured/sick		Monday	4	Writing in thesis
Week 14	Tuesday	0	Injured/sick	Week 17	Tuesday	5	Writing in thesis
	Wednesday	0	Injured/sick		Wednesda	6	Writing in thesis
	Thursday	0	Injured/sick		Thursday	5	Writing in thesis
	Friday	0	Injured/sick		Friday	6	Writing in thesis
	Saturday				Saturday		
0	Sunday	0		26	Sunday		
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	0	Injured/sick		Monday	6	Writing in thesis
Week 15	Tuesday	0	Injured/sick	Week 18	Tuesday	6	Writing in thesis
	Wednesday	0	Injured/sick		Wednesda	8	Writing in thesis
	Thursday	0	Injured/sick		Thursday	7	Writing in thesis
	Friday	0	Injured/sick		Friday	8	Writing in thesis
	Saturday				Saturday	3	
0	Sunday			41	Sunday	3	
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	0	Injured/sick		Monday	6	Writing in thesis
Week 16	Tuesday	0	Injured/sick	Week 19	Tuesday	7	Writing in thesis
	Wednesday	0	Injured/sick		Wednesda	5	Writing in thesis
	Thursday	0	Injured/sick		Thursday	7	Writing in thesis
	Friday	7	Getting back		Friday	5	Writing in thesis
	Saturday				Saturday	7	
7	Sunday			44	Sunday	7	
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	5	Writing in thesis		Monday	12	Writing in thesis
Week 20	Tuesday	7	Writing in thesis	Week 21	Tuesday		
	Wednesday	8	Writing in thesis		Wednesday		
	Thursday	10	Writing in thesis		Thursday		
	Friday	0	Hospital		Friday		
	Saturday	0	Hospital		Saturday		
38	Sunday	8	Writing in thesis	12	Sunday		

Timeføring Fredrik

Week	Day	Hours	Work	Week	Day	Hours	Work
	Monday				Monday	6	Preparation for meeting with stakeholder, meeting with
Week 2	Tuesday			Week 5	Tuesday	0	(Work-part time)
	Wednesday	4	Meeting with client, researc		Wednesday	0	(Work-part time)
	Thursday	5	Group Meeting, planning		Thursday	0	(Work-part time)
	Friday	4	Research on opensearch		Friday	3	Researching nifi
	Saturday				Saturday		
13	Sunday			Total hours: 9	Sunday		
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	2	Group Meeting		Monday	5	Researching nifi
Week 3	Tuesday	1.5	Meeting with supervisor, working on issue board in gitlab	Week 6	Tuesday	7	Testing
	Wednesday	5	Working and planning project plan report		Wednesday	5	Coding
	Thursday				Thursday	7	Meeting with supervisors, research
	Friday	6	Research		Friday	3	Coding
	Saturday	3	Research		Saturday		
Total hours: 17.5	Sunday		*Sick*	27	Sunday		
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday		*Sick*		Monday	3	Researching gitlab runners
Week 4	Tuesday		Sick	Week 7	Tuesday	8	Setting up gitlab runners
	Wednesday		Sick		Wednesday	3	Setting up gitlab runners
Comment: Sick	Thursday		Sick		Thursday	5	Meeting with supervisors, troubleshooting
	Friday		Sick		Friday	7	Testing i Openstack
	Saturday		Sick		Saturday	3	Researching
Total hours: 2	Sunday	2	preparing for meeting on	29	Sunday		
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	6	Researching Nifi		Monday	5	Researching
Week 8	Tuesday	5	Testing Nifi helm charts	Week 11	Tuesday	6	Coding
	Wednesday	3	Troubleshooting		Wednesday	5	Setting up deployment pipelines
	Thursday	2	Deploying Nifi in k8s		Thursday	2	Meeting with supervisors
	Friday	5	Deploying Nifi in k8s		Friday	7	Data Prepper
	Saturday				Saturday		
21	Sunday			28	Sunday	3	Coding
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	5	Deploying Nifi in k8s		Monday	4	Writing in thesis
Week 9	Tuesday	6	Troubleshooting nifi	Week 12	Tuesday	7	Structuring git and overleaf, coding
	Wednesday	8	Troubleshooting nifi		Wednesday	7	Setting up deployment pipelines
	Thursday	7	Meeting with supervisor		Thursday	5	Meeting with supervisors
	Friday				Friday	4	writing in thesis
	Saturday				Saturday	3	Writing in thesis
29	Sunday	3	Writing in thesis	32	Sunday	2	Writing in thesis
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	6	Meeting with client		Monday	0	Easter
Week 10	Tuesday	7	Writing in thesis	Week 13	Tuesday	0	Easter
	Wednesday	4	Writing in thesis		Wednesday	0	Easter
	Thursday	2	Meeting with supervisors		Thursday	0	Easter
	Friday	3	Writing in thesis		Friday	0	Easter
	Saturday	10	Troubleshooting k8s		Saturday		
32	Sunday			2	Sunday	2	Research
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	6	Reviewing meeting minutes, research, report writing		Monday	5	Writing in thesis
Week 14	Tuesday	3	Illustrating	Week 18	Tuesday	6	Writing in thesis
	Wednesday	5			Wednesday	8	Writing in thesis
	Thursday	4	meeting with the team		Thursday	7	Writing in thesis
	Friday		Sick		Friday	7	Writing in thesis
	Saturday		Sick		Saturday		
18	Sunday		Sick	33	Sunday		
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	5	writing in thesis, research, coding		Monday	12	Writing in thesis
Week 15	Tuesday	6	Writing in thesis	Week 19	Tuesday	9	Writing in thesis
	Wednesday	4	Writing in thesis, coding		Wednesday	7	Writing in thesis
	Thursday	7	Writing in thesis, testing		Thursday	13	Writing in thesis
	Friday	7	Writing in thesis		Friday	7	Writing in thesis
	Saturday				Saturday		
29	Sunday			48	Sunday		
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	6	Writing in thesis		Monday	8	Writing in thesis
Week 16	Tuesday	8	Writing in thesis	Week 20	Tuesday	7	Writing in thesis
	Wednesday	5	Research		Wednesday	9	Writing in thesis
	Thursday	2	Research		Thursday	9	Finalizing thesis
	Friday	2	Writing in thesis, testing		Friday	8	Finalizing thesis
	Saturday	6	Writing in thesis		Saturday		
29	Sunday			46	Sunday	5	Finalizing thesis
Week	**Day**		**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	6	Git deployment pipeline, writing		Monday	12	Finalizing thesis
Week 17	Tuesday	9	Git deployment pipeline, writing	Week 21	Tuesday		
	Wednesday	10	Git deployment pipeline, writing		Wednesday		
	Thursday	7	Meeting with supervisors		Thursday		
	Friday				Friday		
	Saturday				Saturday		
26	Sunday			12	Sunday		

Timeføring Bjørn Kristian

Week	Day	Hours	Work	Week	Day	Hours	Work
	Monday	0	Work		Monday	4	Projectplan and meeting with client
Week 2	Tuesday	0	Work	Week 5	Tuesday	4	Projectplan & QA
	Wednesday	2	Meeting with client		Wednesday	4	Projectplan, research, meeting with supervisors
	Thursday	2	Group Meeting		Thursday	5	Project Plan , QA
	Friday	0			Friday	0	Work
	Saturday	4	Research		Saturday	3	Research
8	Sunday	0		24	Sunday	4	Research
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	2	Group Meeting		Monday	0	
Week 3	Tuesday	4	Meeting with supervisor, working on project-plan	Week 6	Tuesday	0	
	Wednesday	5	Project plan		Wednesday	4	report
	Thursday	0	Sick		Thursday	5	Meeting with supervisors, research
	Friday	0	Sick		Friday	8	Coding
	Saturday	0	Sick		Saturday	3	coding
11	Sunday	0	Sick	20	Sunday	0	
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	0			Monday	0	
Week 4	Tuesday	3	Project plan	Week 7	Tuesday	5	Report
	Wednesday	3	Project plan & Research		Wednesday	5	Coding, Research
	Thursday	6	Project plan & Research		Thursday	5	Meeting with supervisors, Coding
	Friday	3	Project plan		Friday	3	Report
	Saturday	0			Saturday	0	Work
18	Sunday	3	Project plan	18	Sunday	0	Work
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	4	Research		Monday	8	Coding, Implementation
Week 8	Tuesday	5	Coding	Week 11	Tuesday	7	Meeting with client, research
	Wednesday	9	Coding		Wednesday	6	Coding
	Thursday	10	Meeting with supervisors, research		Thursday	8	Research coding
	Friday	0	Work		Friday	10	Coding
	Saturday	0	Work		Saturday		
28	Sunday	0	Work	39	Sunday		
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	0			Monday	8	Implementation
Week 9	Tuesday	4	Research	Week 12	Tuesday	4	Connecting nifi to openstack
	Wednesday	5	Research		Wednesday	3	Creating sinks
	Thursday	6	Meeting with supervisors, report		Thursday	5	Benchmarking
	Friday	10	Coding		Friday	6	Research
	Saturday	5	Coding		Saturday		
30	Sunday			26	Sunday		
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	6	Meeting with client, coding		Monday	0	Easter
Week 10	Tuesday	0		Week 13	Tuesday	0	Easter
	Wednesday	5	Implementation		Wednesday	0	Easter
	Thursday	8	Meeting with supervisors, Coding		Thursday	0	Easter
	Friday	9	Implementation		Friday	0	Easter
	Saturday	3	Implementation		Saturday		
35	Sunday	4	Implementation	0	Sunday		
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	7	Writing in Thesis		Monday	4	Writing in Thesis
Week 14	Tuesday	5	Writing in Thesis	Week 17	Tuesday	5	Writing in Thesis
	Wednesday	4	Writing in Thesis		Wednesday	7	Writing in Thesis
	Thursday	5	Writing in Thesis		Thursday	9	Writing in Thesis
	Friday	3	Writing in Thesis		Friday	6	Writing in Thesis
	Saturday	0	Work		Saturday		
24	Sunday	0	Work	31	Sunday		
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	0	Work		Monday	5	Writing in Thesis
Week 15	Tuesday	7	Pipeline	Week 18	Tuesday	6	Writing in Thesis
	Wednesday	5	Troubleshooting		Wednesday	7	Writing in Thesis
	Thursday	10	Troubleshooting		Thursday	6	Writing in Thesis
	Friday	0	Work		Friday	5	Writing in Thesis
	Saturday	0	Work		Saturday		
22	Sunday	0	Work	29	Sunday		
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	4	Writing in Thesis		Monday	7	Writing in Thesis
Week 16	Tuesday	5	Writing in Thesis	Week 19	Tuesday	8	Writing in Thesis
	Wednesday	7	Writing in Thesis		Wednesday	5	Writing in Thesis
	Thursday	9	Writing in Thesis		Thursday	5	Writing in Thesis
	Friday	7	Writing in Thesis		Friday	7	Writing in Thesis
	Saturday	0			Saturday		
32	Sunday	0		32	Sunday		
Week	**Day**	**Hours**	**Work**	**Week**	**Day**	**Hours**	**Work**
	Monday	8	Writing in Thesis		Monday	15	Writing in Thesis
Week 20	Tuesday	9	Writing in Thesis	Week 21	Tuesday		
	Wednesday	4	Writing in Thesis (work)		Wednesday		
	Thursday	4	Writing in Thesis (work)		Thursday		
	Friday	10	Writing in Thesis		Friday		
	Saturday	9	Writing in Thesis		Saturday		
53	Sunday	9	Writing in Thesis	15	Sunday		



NTNU

Norwegian University of
Science and Technology