Tribhuvan University

Amrit Campus

An Internship Report

On

“ DevOps Engineer ”

At

Smart Ideas Pvt. Ltd (Hamro Patro)

Submitted to :

Department of Computer Science and Information Technology

In the partial fulfillment of the requirements for the Bachelor’s Degree in Information Technology awarded by IOST, Tribhuvan University

Submitted by

Saurav Karki (T.U. Exam Roll No. BIT 362/077)

Under the Supervision of

Janak Raj Joshi

MENTOR’S RECOMMENDATION

I hereby recommend that this internship report prepared by **Saurav Karki** (TU Roll NO: **BIT 362/077**) from **Amrit Campus, Thamel**, under my mentorship entitled “DevOps Engineer”, in partial fulfillment of the requirement for the degree of Bachelor in **Information Technology** of **Tribhuvan University,** be processed for evaluation.

…………………………

Mr. Shital Kumar Nyaupane

Mentor

Smart Ideas Pvt.Ltd (Hamro Patro)

Sifal, Kathmandu

SUPERVISOR’S RECOMMENDATION

I hereby recommend that the report entitled *“**An Internship Report on DevOps Engineer at Smart Ideas Pvt.Ltd**”*, prepared under my supervision by Saurav Karki in partial fulfillment of the requirements for the degree of Bachelor in Information Technology of Tribhuvan University, be processed for evaluation.

………………………

Janak Raj Joshi

Supervisor

Department of Computer Science and Information Technology

Amrit Campus

Lekhnath Marg, Thamel

LETTER OF APPROVAL

This is to certify that this internship report prepared by **Saurav Karki** entitled **“An Internship Report On DevOps Engineer at Smart Ideas Pvt.Ltd”** has been submitted to the Department of Information Technology for acceptance in partial fulfillment of the requirements for the degree of Bachelors in Information Technology. In our opinion, it is satisfactory in the scope and quality as an internship for the required degree.

……….………….

External Examiner

…..…..…………..

Asst. Prof. Dhirendra Kumar Yadav

Project Coordinator

Department of CSIT

ACKNOWLEDGEMENT

I would like to express my deepest appreciation to all those who provided me with the possibility to complete this internship report. Special gratitude to my supervisor **Janak Raj Joshi** for the complete support and guidance throughout the internship period.

Also, I would like to express my special gratitude to our Program Coordinator, **Akkal Bista** and administrative staff whose all time encouragement helped me coordinate the internship tasks systematically.

I would like to express my sincere thanks to mentors **Mr. Shital Kumar Nyaupane**, **Mr. Paritosh Bhattarai** and **Mr. Saurab Tharu** of Smart Ideas Pvt. Ltd for sharing their valuable knowledge and guiding me during the internship period, and making me learn new skills and abilities. I am also grateful to the entire staff of Smart Ideas Pvt. Ltd (Hamro Patro) for their constant support, guidance and opportunity.

With all due respect and gratitude, I would like to give a word of thanks to the members of the IT department of Amrit Campus, who encouraged me to perform work activities.

With Regards,

Saurav Karki (BIT 362/077)

ABSTRACT

This report summarizes the DevOps Engineer internship at Smart Ideas Pvt.Ltd (Hamro Patro), where the focus was on infrastructure setup, service deployment, and system orchestration. Key responsibilities included setting up CI/CD pipelines, Setting up and Deploying websites to kubernetes cluster, automating deployments, and setting up virtualization using Proxmox with bare-metal as well as cloud infrastructure. Practical experience was gained in configuring ScyllaDB and PostgreSQL clusters, creating reusable VM templates, and managing containerized workloads using Kubernetes (RKE2,K3s), Implementing proper devops practices to deploy 3-tier applications into EKS cluster.

Tools such as Helm (with custom CRDs), Flux CD, and MetalLB were used for declarative deployments and load balancing. Additional components like Envoy Gateway, NATS messaging, STUN/TURN servers, and Temporal service orchestration were also integrated to support distributed systems. The internship fostered a deeper understanding of the DevOps culture emphasizing automation, collaboration, and continuous improvement while bridging development and operations workflows.

Keywords: DevOps, CI/CD, Proxmox, Kubernetes, ScyllaDB, PostgreSQL, Helm, Flux CD, MetalLB, NATS, Temporal, containerization.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AWS: Amazon Web Services

CI/CD: Continuous Integration and Continuous Deployment

CRDs: Custom Resource Definitions

EC2: Elastic Compute Cloud

EKS: Elastic Kubernetes Service

ELK: ElasticSearch, Logstash, Kibana

JAR: Java Archiever

K8s: Kubernetes

LXC: Linux Containers

RBAC: Role Based Access Control

RKE2: Rancher Kubernetes Engine

S3: Simple Storage Service

SDK: Software Development Kit

SSL: Secure Socket Layer

UFW: Uncomplicated Firewall

VM: Virtual Machines

VPN: Virtual Private Networks

YAML: Yet Another Markup Language

Chapter 1: Introduction

1.1. Introduction

The internship experience at **Smart Ideas Pvt. Ltd. (Hamro Patro)** offered a valuable opportunity to gain practical exposure to DevOps methodologies in a real-world organizational setup. During the course of my DevOps internship, the chance was given to fully adopt the DevOps culture and observe how it has been used effectively to bridge the gap between software development and IT operations. DevOps defines the collection of practices that combines software development (Dev) team and IT operations (Ops) team to shrink the software development life cycle and provide high quality software permanently (Senapathi et al., 2019).

The main duty of a DevOps engineer includes understanding the implementation of DevOps practices to automate and simplify different parts of the software delivery process. This requires creation as well as optimization of Continuous Integration/Continuous Deployment (CI/CD) pipelines which automate integration of code changes frequently and reliably should be deployed; they are important in reducing timetomarket while increasing overall development team's productivity (KALEN WESSE, 2018). This internship served as a foundational experience to explore these DevOps practices in depth, contributing to a better understanding of how infrastructure automation, containerization, monitoring, and continuous delivery contribute to modern software engineering workflows.

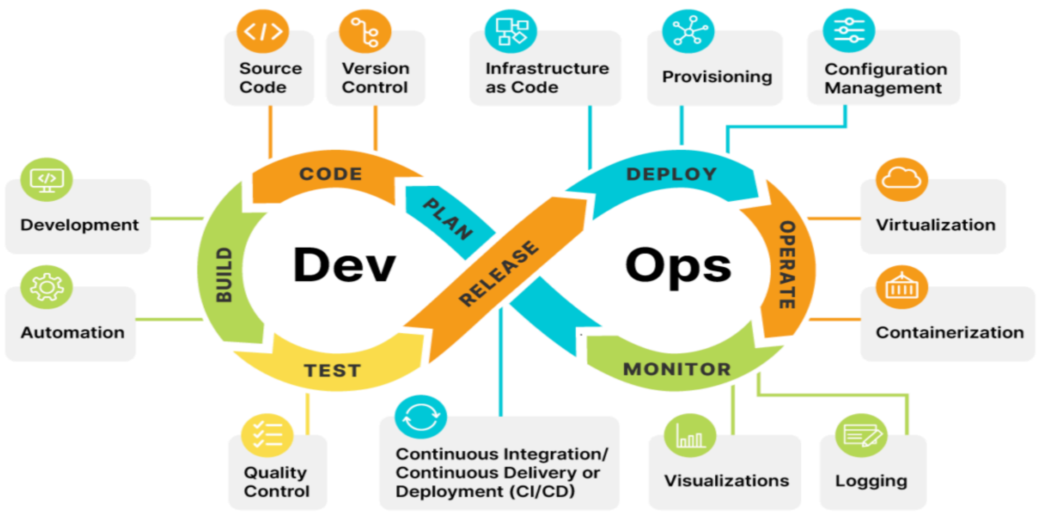


Figure 1.1 : DevOps Lifecycle

As a member of the DevOps team, my primary responsibility was to learn how to streamline and enhance both the development and operational workflows so that the underlying systems could be made robust, efficient, and scalable. I gained hands-on experience in managing CI/CD pipelines, automating deployment processes, and setting up virtualization using Proxmox with bare-metal servers.

There was also an opportunity to work with containerized environments using Kubernetes (RKE2) for orchestrating microservices and managing application deployments. Helm, along with custom CRDs, was used for resource provisioning, while Flux CD enabled GitOps-based continuous deployment. MetalLB was configured to provide load balancing for services within the Kubernetes cluster.

Further technical exposure included configuring ScyllaDB and PostgreSQL clusters, setting up infrastructure observation, Deploying web applications into cloud kubernetes clusters, creating reusable virtual machine templates, and integrating distributed system components such as Envoy Gateway, NATS messaging server, STUN and TURN servers, and the Temporal service orchestration platform.

1.2. Problem Statement

Smart Ideas, like many tech driven organizations, faced challenges related to the manual processes in software deployment, the scalability of their infrastructure, and the efficiency of their operational workflows. The primary issues included:

i) Manual Deployment Processes:

The existing deployment processes were largely manual, leading to inconsistencies, longer deployment times, and higher risk of errors.

ii) Limited Infrastructure Scalability

As the user base grew, the existing infrastructure struggled to scale efficiently. This impacted system performance, uptime, and overall user satisfaction.

iii) Lack of Automation and Monitoring

Key operational tasks such as incident response, resource provisioning, and performance monitoring lacked automation. This led to slower response times and reduced operational efficiency.

Addressing these problems was crucial for maintaining Hamropatro’s competitive edge, ensuring customer satisfaction, and supporting the company’s growth objectives.

1.3. Objectives

The key objectives included:

i) To develop a solid understanding of the DevOps lifecycle, emphasizing collaboration between development and operations teams, continuous integration, and continuous deployment (CI/CD).

ii) To gain hands-on experience in deploying and managing containerized applications using Kubernetes, along with resource definition through Helm charts and Custom Resource Definitions (CRDs).

iii) To design, configure, and manage CI/CD pipelines for automated testing, building, and deployment of applications, ensuring faster and more reliable software releases.

iv) To observe and gain real-world experience in production deployments, lifecycle of live services, gaining insights into challenges and best practices in operational environments.

1.4. Scope and Limitation

The scope of my DevOps internship at Smart Ideas Pvt.Ltd production practices in modern infrastructure management and software delivery. Key areas included:

1.4.1. Scope

i) Bare-Metal Server and Virtualization Management

Managing infrastructure on physical servers using Proxmox for virtualization.

ii) Container Orchestration and Resource Provisioning

Deploying and managing Kubernetes clusters (RKE2) to orchestrate microservices and containerized applications. Helm charts and Custom Resource Definitions (CRDs) were used to define and manage Kubernetes resources effectively.

iii) Infrastructure Monitoring and Alerting

Implementing observability tools such as Prometheus and Grafana to monitor system performance, visualize metrics.

iv) Networking and Load Balancing Configuration

Setting up MetalLB for load balancing within the Kubernetes cluster and integrating service mesh and proxy tools like Envoy Gateway for efficient traffic routing and security.

v) Distributed Systems Integration

Working with distributed components such as ScyllaDB, PostgreSQL clusters, the NATS messaging system, STUN/TURN servers, and the Temporal.

1.4.2. Limitations

Despite the comprehensive scope, there were some limitations during my internship:

Time Constraints:

The duration of the internship was limited, which restricted the depth of exploration and implementation of certain advanced DevOps practices and tools.

Resource Availability:

Access to certain hardware and software resources was limited, which occasionally hindered the implementation and testing of specific solutions on a larger scale.

Learning Curve:

The complexity of some tools and technologies, especially those I was unfamiliar with, required significant time to learn, reducing the time available for handson application.

Assigned Task Scope:

The tasks assigned were predetermined, leaving limited room to explore additional areas of personal or emerging interest within the DevOps field.

1.5. Report Organization

This report is structured into four main chapters, each detailing different aspects of my internship experience at Hamropatro. Here is a brief overview of each chapter:

Chapter 1: Introduction

This chapter introduces the work completed during my internship. It outlines the problem statement, the objectives of the internship, the scope and limitations of the project, and provides an overview of the report’s organization.

Chapter 2: Organization Details and Literature Review

In this chapter, a comprehensive introduction to Smart Ideas Pvt.Ltd has been provided. This includes an overview of the organization, its hierarchy, the various domains in which it operates, and a detailed description of the department where internship has been completed. Additionally, this chapter includes a literature review or related study, highlighting relevant theories and frameworks that underpin the works that have been performed during the internship.

Chapter 3: Internship Activities

This chapter delves into the specifics of my internship activities. It outlines my roles and responsibilities, provides a weekly log of the technical activities, describes the involved projects, and details the technical tasks and activities have been completed successfully. This section offers an indepth look at the handson experience obtained.

Chapter 4: Conclusion and Learning Outcomes

A brief overview of the experience gained during the internship is also stated in this last part, as well as the main conclusions. It mentions my skills and knowledge, challenges I faced and how I dealt with them. Additionally, the section talks about what the future holds in terms of career development after such an opportunity.

Chapter 2: Background Study and Literature Review

2.1. Introduction to Organization

Smart Ideas Pvt. Ltd., popularly known as **HamroPatro**, is a leading software company based in Kathmandu, Nepal. **Smart Ideas Pvt.Ltd** is a pioneering Nepali technology company recognized as the country's leading digital super-app. Established in 2010 by Shankar Uprety, Hamro Patro started as a personal initiative to digitize the traditional Nepali calendar for mobile platforms. Since then, the company has evolved into a comprehensive digital ecosystem, offering a diverse range of services aimed at enriching the lives of Nepalis both at home and abroad.

The platform's offerings include a Nepali calendar, news aggregation, forex and gold rates, horoscopes, Nepali FM radio, podcasts, health consultations, event ticketing, educational resources, and digital remittance services. In recent years, Hamro Patro has expanded into fintech by launching **Hamro Pay**, a digital wallet, further strengthening its role in digital inclusion and financial connectivity.

As of 2025, Hamro Patro has achieved over **10 million plus app downloads** and serves more than **15 million monthly active users**, making it the most widely used Nepali app globally. The app plays a significant role in helping the Nepali diaspora stay connected with their culture and community, while also serving as an indispensable digital utility for users within Nepal. Hamro Patro Remit has been especially impactful for international money transfers, primarily catering to Nepali migrant workers.

Hamro Patro continues to lead Nepal's digital transformation by providing trusted, locally-tailored services through a single unified platform. Its success is driven by a dedicated team of technologists, designers, and domain experts committed to building a secure, inclusive, and dynamic digital ecosystem.

Table 2.1 : Company Details

Year of establishment	2010
Key Service areas	Nepali calendar, News, Horoscope, Finance, Radio, Podcasts, E-learning, Remittance, Telehealth, Digital Wallet, Messaging.

Staff Size	60-120 employees
Location of clients	Sifal,Kathmandu
Expertise in	Multi service mobile platform, calendar, remit, health, fintech, content delivery
Noteworthy mentions	First nepali calendar app on ios, 10 million+ downloads

2.2. Organizational Hierarchy

Smart Ideas promotes innovation and agility through a lean and cross-functional organizational structure. The Board of Directors provides strategic direction, while the Executive Management team is responsible for translating this vision into actionable goals. The organization is structured into specialized departments, each contributing to the platform's growth and user experience.

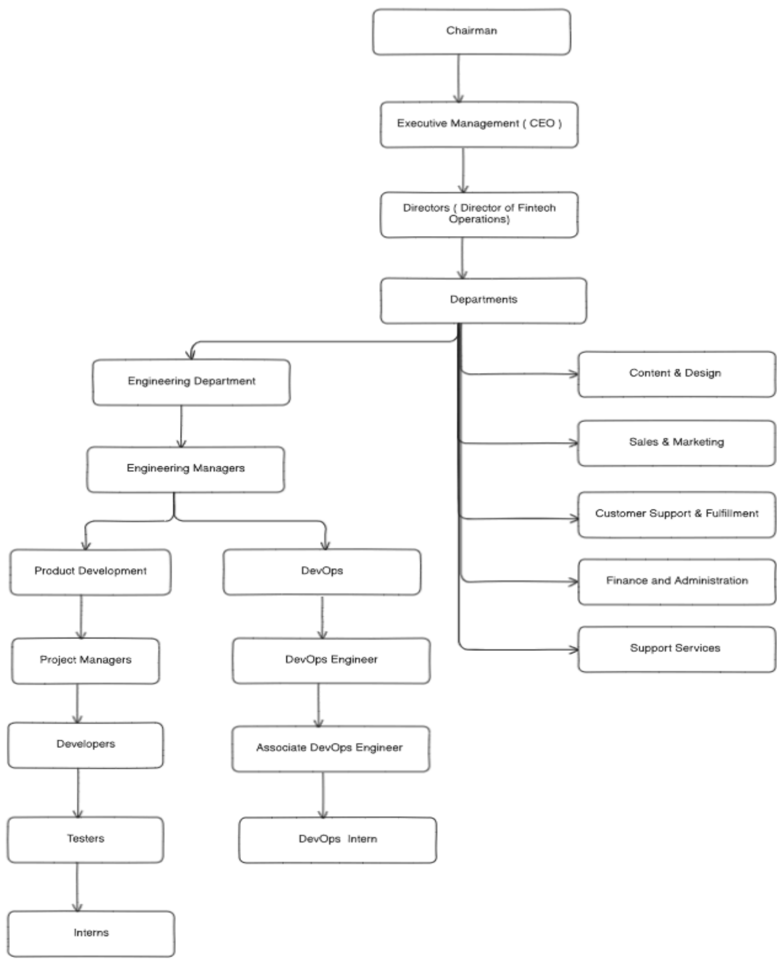


Figure 2.1 : Organizational Hierarchy

2.3. Working Domains of Organization

The company primarily operates in the following domains:

1. **Cultural and Calendar Services**:
   Smart Ideas is best known for its **Nepali calendar**, which remains the core feature of the platform. It provides festival updates, tithis, rashifal, Panchang details, and both Bikram Sambat (B.S.) and Gregorian calendar conversion.

2. **News and Information Aggregation**:
   The platform aggregates content from over 80 national and regional news portals, making it a go-to source for news consumption. Users can access real-time updates on national affairs, politics, finance, sports, and entertainment from a centralized interface.

3. **Media and Entertainment** :
   Smart Ideas features **Nepali FM radio stations**, allowing users to stream local radio from anywhere. It also hosts a wide collection of **podcasts**, along with features like ecards, music streaming, and video content.

4. **Fintech and Digital Payments** :
   Smart Ideas has introduced **Hamro Pay**, a digital wallet that supports mobile recharges, utility bill payments, merchant QR payments (NepalPay), and more.
   Additionally, **Hamro Patro Remit** enables fast and affordable **international money transfer** services, particularly aimed at the large Nepali diaspora working abroad.

5. **Health and Teleconsultation** :
   The app integrates **telehealth services**, allowing users to schedule virtual consultations with licensed doctors, access medical content, and receive wellness advice.

6. **Education and E-Learning** :
   Hamro Patro also offers **learning resources** such as academic materials, language tools, and exam preparation content. The app includes **dictionary tools**, Nepali typing keyboards, and multilingual support to enhance learning and communication.

7. **Astrology & Jyotish Consultations:**
   Hamro Jyotish connects users with certified Vedic astrologers (Jyotish) via live audio/video consultations and personalized remedies.

8. **Marketplace** :
   Through **Hamro Mart**, users can experience ecommerce services and buy different goods online.

2.4. Description of Intern Department

During my internship at **Smart Ideas Pvt.Ltd** (Hamro Patro)., I was placed in the Engineering Department, which plays a crucial role in the company's IT infrastructure and operations. The DevOps team is responsible for ensuring seamless integration and deployment processes, enabling continuous integration and continuous delivery (CI/CD) of applications. This involves managing infrastructure automation, monitoring system performance, and enhancing deployment efficiency through streamlined processes and tools. Each team within the department is led by a dedicated manager who oversees operations and delegates responsibilities to team members. Under the guidance of the Engineering Department, the department fosters a collaborative and energetic environment that enables its teams to deliver exceptional results.

As a DevOps intern, I had the opportunity to work under the guidance of my mentors, Shital Kumar Nyaupane and Saurab Tharu who provided invaluable assistance throughout my tenure. My responsibilities included assisting in the setup and maintenance of scylla, postgres database clusters,working with tools like Docker, Kubernetes, and Helms, Envoy , Prometheus , Grafana, ELK for infrastructure automation, and implementing monitoring tools to track system performance. Additionally, I wrote scripts to automate routine tasks, improving overall efficiency in deployment and maintenance processes. This handson experience in DevOps practices, coupled with the support and mentorship from my team, significantly enhanced my technical skills and prepared me for a future career in the DevOps field. The collaborative and energetic environment at HamroPatro allowed me to develop professionally and contribute effectively to the team's objectives.

2.5. Literature Review

The adoption of DevOps practices has significantly transformed the software development and IT operations landscape, promoting a culture of collaboration, continuous integration, and automation (Ebert et al., 2016). DevOps culture thrives on the breaking down of walls between development and operations teams thus enabling faster and more reliable software releases (Bass, 2015). This kind of transformation is supported by a shift towards this culture which is fostered by processes and tools of automation where quality can be delivered at speed without sacrificing stability of operations or efficiency in running such systems within an organization.

Continuous Integration/Continuous Deployment (CI/CD) is one such central pillar among other things that make up DevOps (Farley, 2015). CI/CD pipelines automate integration testing deployment, speeding up production cycles through reduction of manual labour errors and general slowness associated with them thus ultimately boosting overall productivity levels within development teams. Also, it sets a ground for receiving quick responses from clients during different stages (feedback loops) because developers can detect any problem at an early stage before proceeding further.

Additionally, if DevOps practices are adopted in organization then system monitoring and incident management become easier than ever before. There are continuous monitoring tools such as Prometheus, Grafana or ELK stack (Elasticsearch, Logstash, Kibana) among others which offer visibility into the performance and health status of a system real time (Ebert et al., 2016). Through them organizations can easily find anomalies proactively as well as respond quickly when incidents occur so as to improve reliability while reducing downtime for those depending on these systems most times in businesses world wide. More still, an effective monitoring combined with logging forms strong pillars towards achieving success through ensuring high availability levels & performances are maintained always within any given environment setting under consideration taking cognizance that downtime may translate into huge losses especially financially or even worse loss of lives due failure deliver mission critical services.

Chapter 3: Internship Activities

3.1. Roles and Responsibilities

While working as a DevOps Engineer intern for Smart Ideas , my main focus was on bringing together software development and IT operations. I had the following tasks:

i) CD Pipeline Implementation:
Automating software build, test, and deployment processes by setting up Flux CD pipelines.

ii) Infrastructure Management:
Designing infrastructure solutions that could be scaled using physical servers.

iii) Monitoring and Logging:
Ensuring system reliability and performance, setting up the monitoring tools such as Grafana, Prometheus, Uptime Kuma as well as setting up alerting systems.

iv) Containerization:
Setting up Kubernetes (RKE2) for managing Docker containers which were orchestrated using Rancher.

v) Documentation & Reporting:
Preparing documentation of every procedure undertaken along with their configurations before finally compiling performance reports at the end of each month.

vi) Collaboration:
Working hand in hand with developers and other team players so as to smoothen integration points between development and deployment workflows

vii) Continuous learning:
Staying updated with industry trends and applying new knowledge to improve existing systems.

viii) Deployment to k8s cluster:

Deploying three tier applications to kubernetes cluster following best practices.

3.2. Weekly log

The following table shows the weekly activities the intern performed throughout their internship period.

Table 3.1: Weekly log

Week	**Activities**
Week 1	● Onboarding session. ● Into to proxmox virtualization ● Getting used to and more familiar with linux commands. ● Understanding and learning of networking commands ● Learned about linux container and linux container daemons ● Containerization and deployment of simple java micronaut applications in a container runtime environment.
Week 2	● Proxmox setup in laptop ● Learned about server hardening best practices ● Learned about setting up a bastion host. ● Introduction to kafka and Distributed systems.
Week 3	● Explored wine to run windows application in linux VM. ● Explored and setup Envoy Gateway and Contour ingress controller to route traffic in kubernetes cluster ● Learned about load balancing and setup metallb in bare metal systems. ● Configured nginx hosted website with SSL certificate
Week 4	● Explored and used customization for configuration management in k8s cluster. ● Setup Flux CD for continuous delivery of applications into the kubernetes cluster. ● Explored all the controllers of Flux CD. ● Compared Flux CD and Argo CD for continuous deployment.
Week 5	● Learned about image automation controllers to automatically update images in deployment by fetching from container registry like docker hub. ● Learned about setting monitoring with Flux CD ● Learned and set up NATS in the kubernetes cluster with Helm Charts. ● Setuped RKE2 Kubernetes cluster ● Containerized and deployed java spring boot app to docker containers.

Week 6	● Introduction to scylladb and local setup. ● Setup scylladb cluster with 2 master and 2 workers nodes in vmware virtual machines. ● Introduction to Ipsec and setting up site to site vpn between data centers with libreswan. ● Learn and understand about ipsec.
Week 7	● Deployed sidecar container in RKE2 ● Explored Iptables and UFW firewall. ● Learned about K8s common Errors and troubleshooting process ● Learned about awk and sed command ● Created VM Template for future use.
Week 8	● Deployed pods in RKE2 cluster and checked where pod schedule to new node on failure or not ● Explored Hamro Patro Development environment repo structure flow. ● Learned about how Gateway VM is created. ● Explored influxdb for log collection. ● Added Metrix server in proxmox to collect old dallas data center logs into influxdb.
Week 9	● Setup Ipsec from old dallas data center to lax data center. ● Explored scylladb multi data center cluster. ● Explored the API Gateway in Kubernetes. ● Setup Envoy Gateway , Http Route to route the external traffic to k8s pods. ● Deployed Yelcamp a three tier app to EKS cluster.
Week 10	● Setup postgres database cluster in vm. ● Explored and Setup Reloader in kubernetes. ● Explored about kubens and kubectx tools for kubernetes. ● Explored monitoring of kubernetes cluster with octant, kubernetes dashboard, lens.
Week 11	● Implemented service level load balancing in envoy gateway. ● Explored envoy circuit breaker, client traffic policy in envoy gateway. ● Explored about proxmox vm backup in s3 bucket. ● Setup vm backup in digital ocean s3 bucket. ● Explored Hamro Patro Repo Structure.

Week 12	● Setup monitoring in RKE2 K8s cluster. ● Explored EBPF and its use cases. ● Setup ELK stack for monitoring log and visualizing them using Kibana in local system ● Devopsified and deployed java micronaut application to k3s kubernetes cluster.

3.3. Description of the Projects Involved During Internship

During my internship, I was involved in several tasks ranging from minor configurations to major deployments. Among these, three major projects stood out, both leveraging local infrastructure and DevOps practices. Here are the some of the minor to major learnings and tasks i have performed:

Project 1: Containerized and deployed java spring boot application into docker containers.

In this project I focused on containerizing a full-stack Java Spring Boot application and deploying it with Docker Compose in a cloud vm environment. It provided a hands-on understanding of containerization, environment management, and service orchestration all core concepts in modern DevOps practices.

i) Application Packaging with Multi-Stage Dockerfile

The application was built using a multi-stage Dockerfile to optimize image size and efficiency:

- In the first stage, a Maven image was used to compile and package the Spring Boot application (task-manager) into a runnable JAR file.
- The second stage used a lightweight OpenJDK runtime image (eclipse-temurin) to run the packaged application, reducing the final image size and improving startup speed.

Dockerfile:

```
# Stage 1: Build the application
FROM maven:3.9.6-eclipse-temurin-17-alpine AS build
WORKDIR /app
# Copy the Maven project files
```

```
COPY pom.xml .
COPY src ./src
# Package the application
RUN mvn clean package -DskipTests
# Stage 2: Run the application
FROM eclipse-temurin:17-jre-alpine
WORKDIR /app
# Copy the built jar from the previous stage
COPY --from=build /app/target/task-manager-0.0.1-SNAPSHOT.jar app.jar
EXPOSE 8080
```

ii) Service Composition with Docker Compose

The entire application stack was defined and orchestrated using docker-compose.yml, enabling seamless local deployment:

- A PostgreSQL 15 container served as the backend database, with persistent storage through Docker volumes.
- The Spring Boot container ran the main application, exposing it on port 8080, and was configured to connect to the PostgreSQL service using environment variables.

Docker Compose:

```
version: '3.8'
services:
  postgres:
    image: postgres:15
    container_name: pgdb
    restart: always
    environment:
      POSTGRES_DB: taskmanager
      POSTGRES_USER: taskapp
      POSTGRES_PASSWORD: securepassword
    ports:
      - "5433:5432"
    volumes:
      - pgdata:/var/lib/postgresql/data
  taskmanager:
    build: .
    ports:
      - "8080:8080"
    environment:
      SPRING_DATASOURCE_URL: jdbc:postgresql://postgres:5432/taskmanager
      SPRING_DATASOURCE_USERNAME: taskapp
      SPRING_DATASOURCE_PASSWORD: securepassword
    depends_on:
      - postgres
volumes:
  pgdata:
```

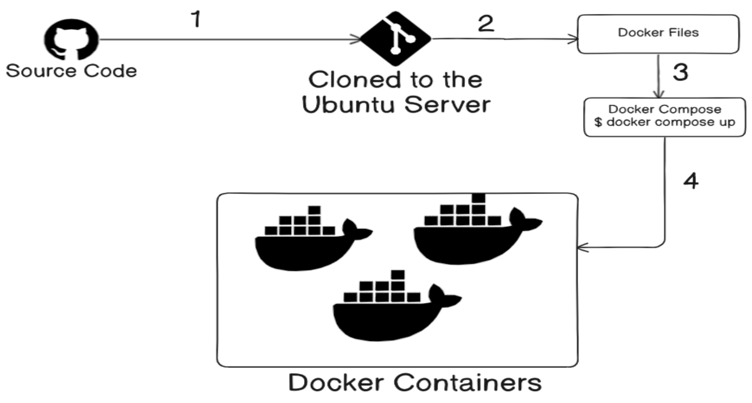
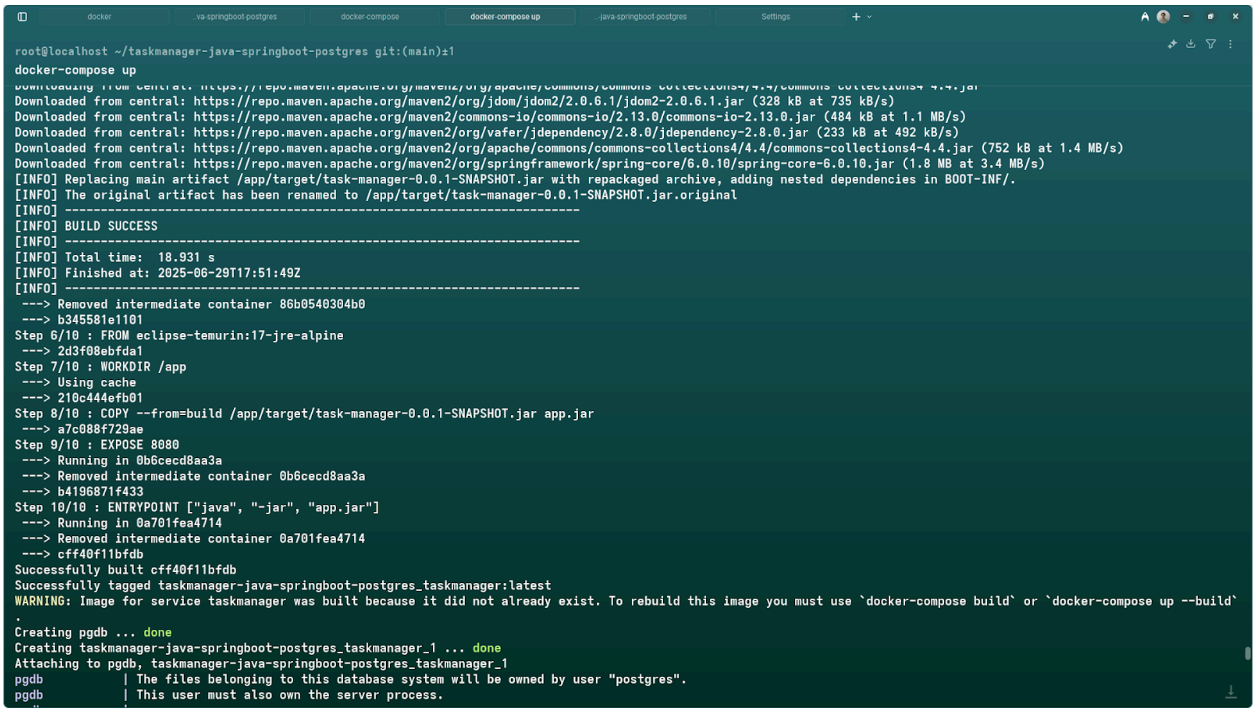
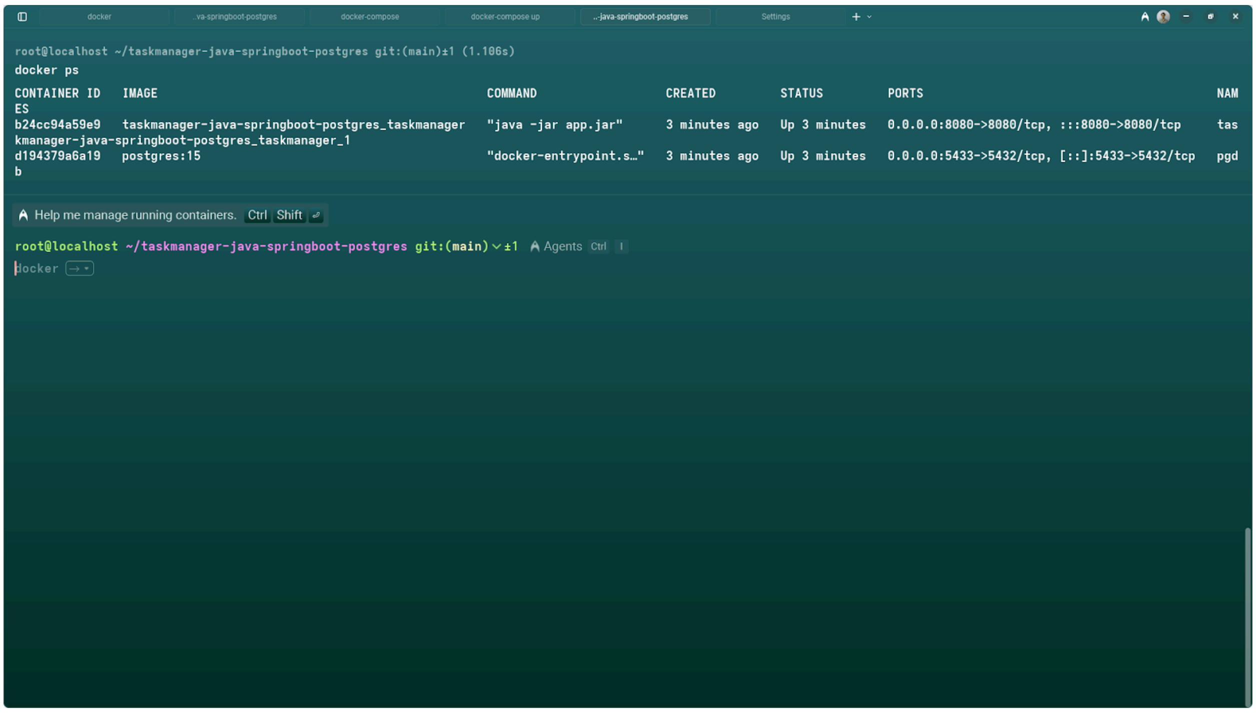


Figure 3.1 : Architecture Diagram of Project on Containerized and deployed java spring boot application into docker containers.



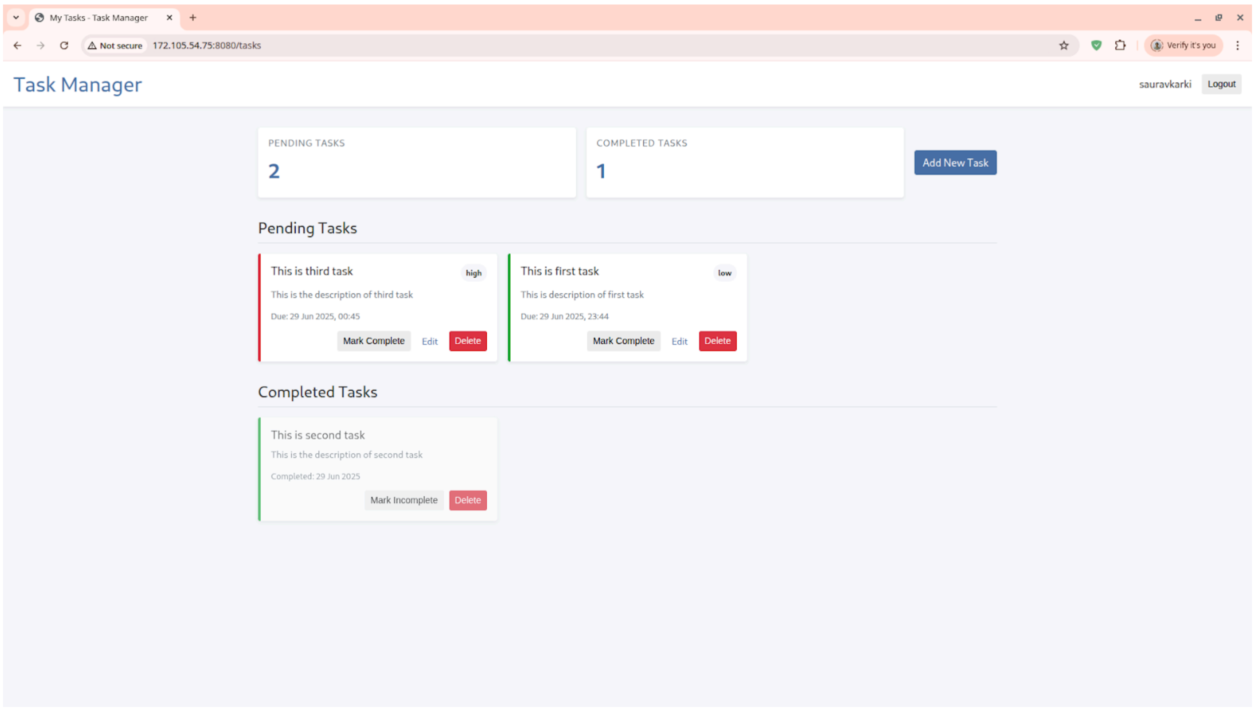
```
root@localhost ~/taskmanager-java-springboot-postgres git:(main)±1
docker-compose up
Downloaded from central: https://repo.maven.apache.org/maven2/org/jdom/jdom2/2.0.6.1/jdom2-2.0.6.1.jar (328 kB at 735 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/commons-io/commons-io/2.13.0/commons-io-2.13.0.jar (484 kB at 1.1 MB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/vafer/jdependency/2.8.0/jdependency-2.8.0.jar (233 kB at 492 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/commons/commons-collections4/4.4/commons-collections4-4.4.jar (752 kB at 1.4 MB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/springframework/spring-core/6.0.10/spring-core-6.0.10.jar (1.8 MB at 3.4 MB/s)
[INFO] Replacing main artifact /app/target/task-manager-0.0.1-SNAPSHOT.jar with repackaged archive, adding nested dependencies in BOOT-INF/.
[INFO] The original artifact has been renamed to /app/target/task-manager-0.0.1-SNAPSHOT.jar.original
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  18.931 s
[INFO] Finished at: 2025-06-29T17:51:49Z
[INFO] ------------------------------------------------------------------------
 ---> Removed intermediate container 86b0540304b0
 ---> b345581e1101
Step 6/10 : FROM eclipse-temurin:17-jre-alpine
 ---> 2d3f08ebfda1
Step 7/10 : WORKDIR /app
 ---> Using cache
 ---> 210c444efb01
Step 8/10 : COPY --from=build /app/target/task-manager-0.0.1-SNAPSHOT.jar app.jar
 ---> a7c088f729ae
Step 9/10 : EXPOSE 8080
 ---> Running in 0b6cecd8aa3a
 ---> Removed intermediate container 0b6cecd8aa3a
 ---> b4196871f433
Step 10/10 : ENTRYPOINT ["java", "-jar", "app.jar"]
 ---> Running in 0a701fea4714
 ---> Removed intermediate container 0a701fea4714
 ---> cff40f11bfdb
Successfully built cff40f11bfdb
Successfully tagged taskmanager-java-springboot-postgres_taskmanager:latest
WARNING: Image for service taskmanager was built because it did not already exist. To rebuild this image you must use `docker-compose build` or `docker-compose up --build`
.
Creating pgdb ... done
Creating taskmanager-java-springboot-postgres_taskmanager_1 ... done
Attaching to pgdb, taskmanager-java-springboot-postgres_taskmanager_1
pgdb            | The files belonging to this database system will be owned by user "postgres".
pgdb            | This user must also own the server process.
```

Figure 3.2: Docker compose up Steps



```
root@localhost ~/taskmanager-java-springboot-postgres git:(main)±1 (1.106s)
docker ps
CONTAINER ID   IMAGE                                              COMMAND                  CREATED         STATUS         PORTS                                       NAM
ES
b24cc94a59e9   taskmanager-java-springboot-postgres_taskmanager   "java -jar app.jar"      3 minutes ago   Up 3 minutes   0.0.0.0:8080->8080/tcp, :::8080->8080/tcp   tas
kmanager-java-springboot-postgres_taskmanager_1
d194379a6a19   postgres:15                                        "docker-entrypoint.s…"   3 minutes ago   Up 3 minutes   0.0.0.0:5433->5432/tcp, [::]:5433->5432/tcp   pgd
b

root@localhost ~/taskmanager-java-springboot-postgres git:(main)±1
docker
```

Figure 3.3: Verifying the Running Containers


My Tasks - Task Manager

Not secure 172.105.54.75:8080/tasks

Task Manager

sauravkarki Logout

PENDING TASKS
2

COMPLETED TASKS
1

Add New Task

Pending Tasks

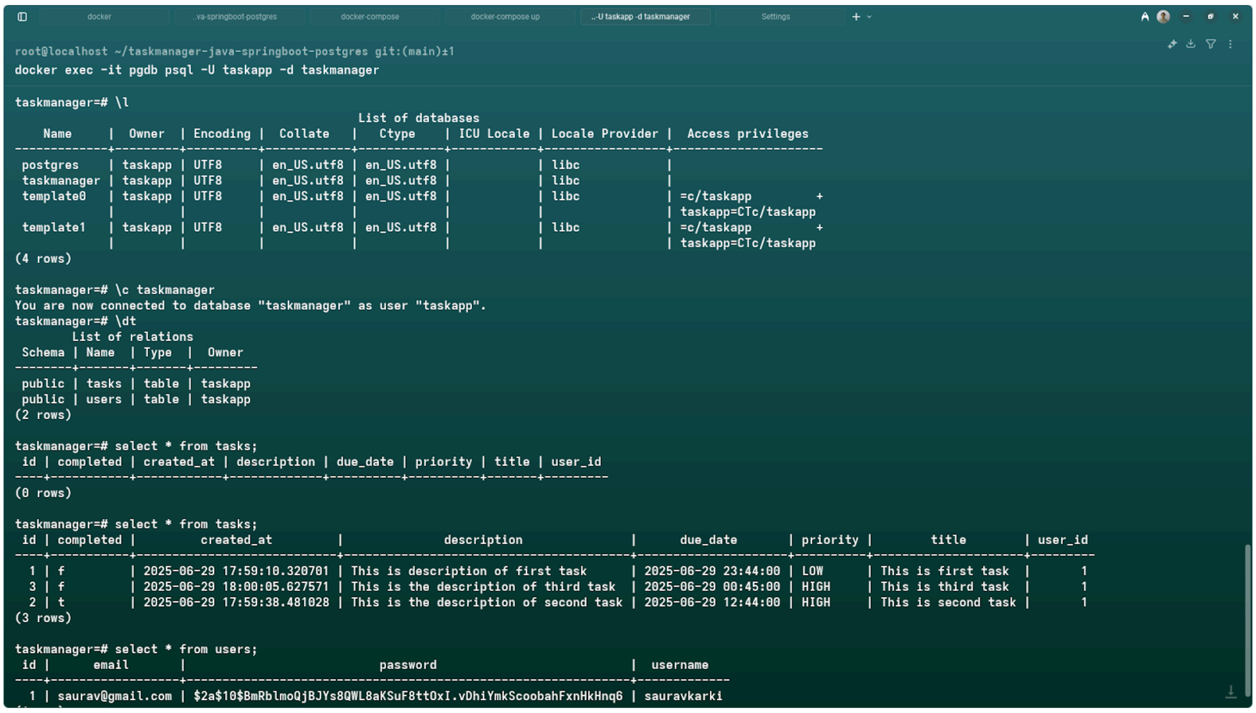
This is third task — high
This is the description of third task
Due: 29 Jun 2025, 00:45
Mark Complete Edit Delete

This is first task — low
This is description of first task
Due: 29 Jun 2025, 23:44
Mark Complete Edit Delete

Completed Tasks

This is second task
This is the description of second task
Completed: 29 Jun 2025
Mark Incomplete Delete

Figure 3.4: Accessing the deployed spring boot app from public ip.



```
root@localhost ~/taskmanager-java-springboot-postgres git:(main)±1
docker exec -it pgdb psql -U taskapp -d taskmanager

taskmanager=# \l
                                                     List of databases
    Name     |  Owner  | Encoding |  Collate   |   Ctype    | ICU Locale | Locale Provider |   Access privileges
-------------+---------+----------+------------+------------+------------+-----------------+-----------------------
 postgres    | taskapp | UTF8     | en_US.utf8 | en_US.utf8 |            | libc            |
 taskmanager | taskapp | UTF8     | en_US.utf8 | en_US.utf8 |            | libc            |
 template0   | taskapp | UTF8     | en_US.utf8 | en_US.utf8 |            | libc            | =c/taskapp          +
             |         |          |            |            |            |                 | taskapp=CTc/taskapp
 template1   | taskapp | UTF8     | en_US.utf8 | en_US.utf8 |            | libc            | =c/taskapp          +
             |         |          |            |            |            |                 | taskapp=CTc/taskapp
(4 rows)

taskmanager=# \c taskmanager
You are now connected to database "taskmanager" as user "taskapp".
taskmanager=# \dt
        List of relations
 Schema | Name  | Type  |  Owner
--------+-------+-------+---------
 public | tasks | table | taskapp
 public | users | table | taskapp
(2 rows)

taskmanager=# select * from tasks;
 id | completed | created_at | description | due_date | priority | title | user_id
----+-----------+------------+-------------+----------+----------+-------+---------
(0 rows)

taskmanager=# select * from tasks;
 id | completed |         created_at         |               description                |      due_date       | priority |        title        | user_id
----+-----------+----------------------------+------------------------------------------+---------------------+----------+---------------------+---------
  1 | f         | 2025-06-29 17:59:10.320701 | This is description of first task        | 2025-06-29 23:44:00 | LOW      | This is first task  |       1
  3 | f         | 2025-06-29 18:00:05.627571 | This is the description of third task    | 2025-06-29 00:45:00 | HIGH     | This is third task  |       1
  2 | t         | 2025-06-29 17:59:38.481028 | This is the description of second task   | 2025-06-29 12:44:00 | HIGH     | This is second task |       1
(3 rows)

taskmanager=# select * from users;
 id |      email       |                           password                           |  username
----+------------------+--------------------------------------------------------------+-------------
  1 | saurav@gmail.com | $2a$10$BmRblmoQjBJYs8QWL8aKSuF8ttOxI.vDhiYmkScoobahFxnHkHnq6 | sauravkarki
```

Figure 3.5: Verifying the data stored in postgresql db container

Project 2:End-to-End CI/CD Deployment of YelpCamp on AWS using Docker & Kubernetes

This project was about designing and implementing a complete DevOps workflow for a full-stack web application called YelpCamp. It involved transitioning from manual deployments to automated, containerized, and scalable deployment pipelines, leveraging modern DevOps tools and best practices.

i) Initial Manual Deployment

The project began with manual deployment on AWS EC2 instances, where:

- The Node.js backend was manually configured on a virtual machine.
- MongoDB Atlas was used as a cloud-hosted NoSQL database.
- The environment provided a baseline understanding of infrastructure provisioning and manual service setup.

ii) Containerized CI/CD Workflow

To streamline development and deployment, the application was containerized using Docker and integrated into a CI/CD pipeline using Jenkins:

- Jenkins Pipelines were configured to trigger on code changes, automating:
    - Unit Testing
    - Code Quality Checks using SonarQube
    - Security Scans using Trivy
- Docker images were built and pushed to DockerHub.
- A development environment on EC2 consumed these images for continuous integration and testing.

iii) Production Deployment on AWS EKS

The final stage of the project involved a fully automated production deployment on AWS EKS:

- The Dockerized application was orchestrated using Kubernetes, ensuring scalability, high availability, and fault tolerance.
- Kubernetes manifests define the desired state of application components.
- Security best practices were followed using RBAC, service accounts, and namespace isolation.
- External services such as Mapbox (for geolocation) and Cloudinary (for image hosting) were securely integrated.

Manifests:

```
apiVersion: v1
kind: Secret
metadata:
  name: yelp-camp-secrets
type: Opaque
data:
  CLOUDINARY_CLOUD_NAME: ZHgzdnlqaTRv
  CLOUDINARY_KEY: NTk1OTU4OTcxODM0NzU3
  CLOUDINARY_SECRET: N3ZSRnhfbF9YNllsc3pZMVFCaHNxTjJrQllB
MAPBOX_TOKEN:
cGsuZXlKMUlqb2ljMkYxY21GMmEyRnlhMmtpTENKaElqb2lZMjAzTmpGb01HTnJNR3RzZVR
KcGNYaHphakJ2WTJaNE15SjkuRjR3VzcxSFJfbGNfUDR4RFd0YnlEQQ==
DB_URL:
bW9uZ29kYitzcnY6Ly9zYXVyYXZrYXJraXRlY2g6MHRBQ0lTTDNReVpCZHZmUkBzYXVyY
XYtZGItdGhyZWV0aWVyLnRibmx0Lm1vbmdvZGIubmV0Lz9yZXRyeVdyaXRlcz10cnVlJnc9bW
Fqb3JpdHkmYXBwTmFtZT1zYXVyYXYtZGItdGhyZWV0aWVy
  SECRET: c2F1cmF2a2Fya2k=
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: yelp-camp-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: yelp-camp
  template:
    metadata:
      labels:
        app: yelp-camp
    spec:
```

```
containers:
      - name: yelp-camp-container
        image: sauravkarki/campapp:latest
        ports:
          - containerPort: 3000
  env:
          - name: CLOUDINARY_CLOUD_NAME
            valueFrom:
              secretKeyRef:
                name: yelp-camp-secrets
                key: CLOUDINARY_CLOUD_NAME
          - name: CLOUDINARY_KEY
            valueFrom:
              secretKeyRef:
                name: yelp-camp-secrets
                key: CLOUDINARY_KEY
          - name: CLOUDINARY_SECRET
            valueFrom:
              secretKeyRef:
                name: yelp-camp-secrets
                key: CLOUDINARY_SECRET
          - name: MAPBOX_TOKEN
            valueFrom:
              secretKeyRef:
                name: yelp-camp-secrets  key: MAPBOX_TOKEN
          - name: DB_URL
            valueFrom:
              secretKeyRef:
                name: yelp-camp-secrets
                key: DB_URL
```

```
    - name: SECRET
        valueFrom:
          secretKeyRef:
            name: yelp-camp-secrets
            key: SECRET
      livenessProbe:
        httpGet:
          path: /
          port: 3000
        initialDelaySeconds: 30   # Adjust the initial delay here
      readinessProbe:
         httpGet:
          path: /
          port: 3000
        initialDelaySeconds: 30   # Adjust the initial delay here
---
apiVersion: v1
kind: Service
metadata:
  name: yelp-camp-service
spec:
  selector:
    app: yelp-camp
  ports:
    - protocol: TCP
      port: 3000
      targetPort: 3000
  type: LoadBalancer
```

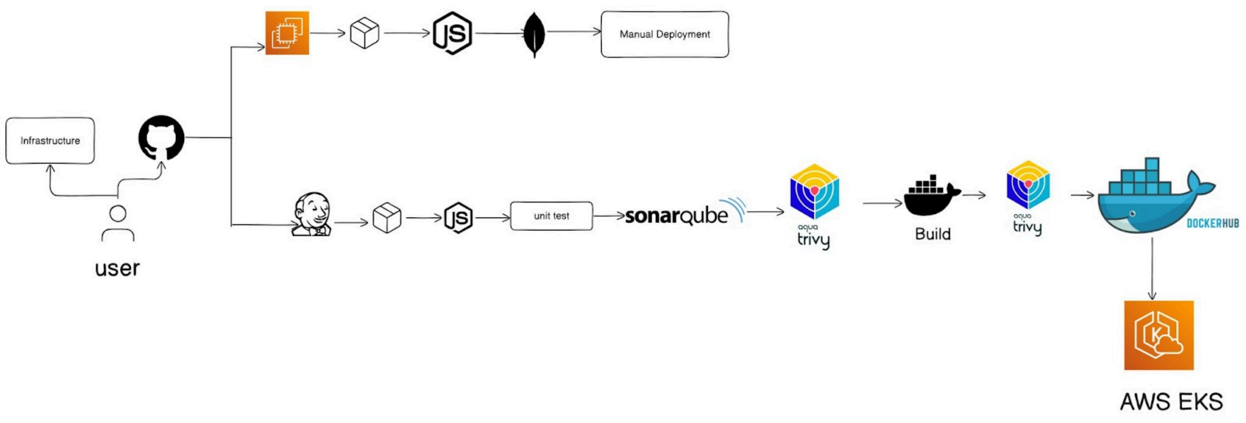
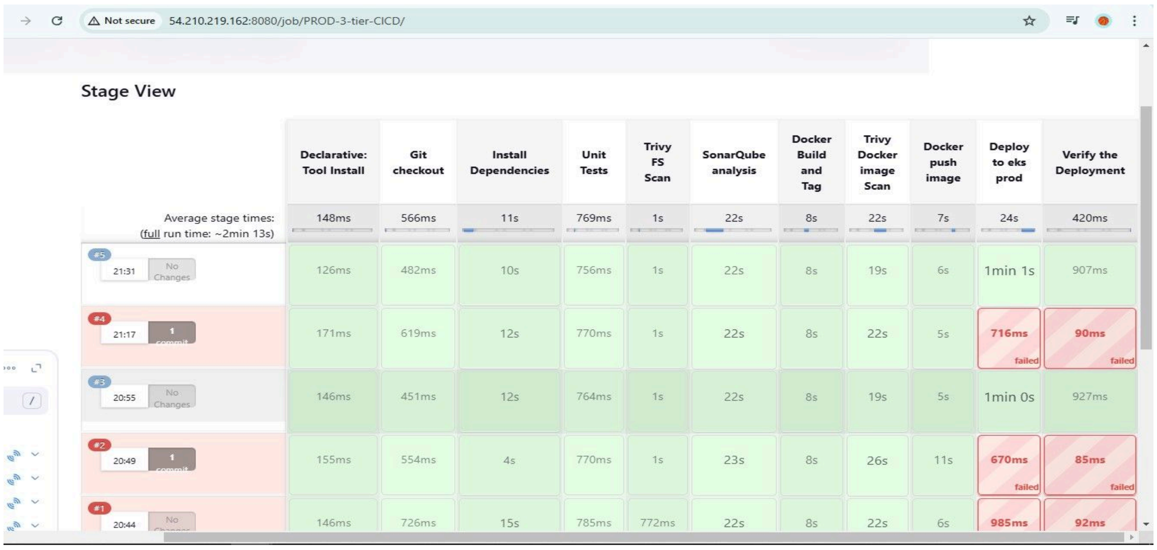


Figure 3.6 : Architecture Diagram of Project on End-to-End CI/CD Deployment of YelpCamp on AWS using Docker & Kubernetes



54.210.219.162:8080/job/PROD-3-tier-CICD/

Stage View

	Declarative: Tool Install	Git checkout	Install Dependencies	Unit Tests	Trivy FS Scan	SonarQube analysis	Docker Build and Tag	Trivy Docker image Scan	Docker push image	Deploy to eks prod	Verify the Deployment
Average stage times: (full run time: ~2min 13s)	148ms	566ms	11s	769ms	1s	22s	8s	22s	7s	24s	420ms
#5 21:31 No Changes	126ms	482ms	10s	756ms	1s	22s	8s	19s	6s	1min 1s	907ms
#4 21:17 1 commit	171ms	619ms	12s	770ms	1s	22s	8s	22s	5s	716ms failed	90ms failed
#3 20:55 No Changes	146ms	451ms	12s	764ms	1s	22s	8s	19s	5s	1min 0s	927ms
#2 20:49 1 commit	155ms	554ms	4s	770ms	1s	23s	8s	26s	11s	670ms failed	85ms failed
#1 20:44 No Changes	146ms	726ms	15s	785ms	772ms	22s	8s	22s	6s	985ms	92ms

Figure 3.7 : CICD Pipeline for the application in Jenkins

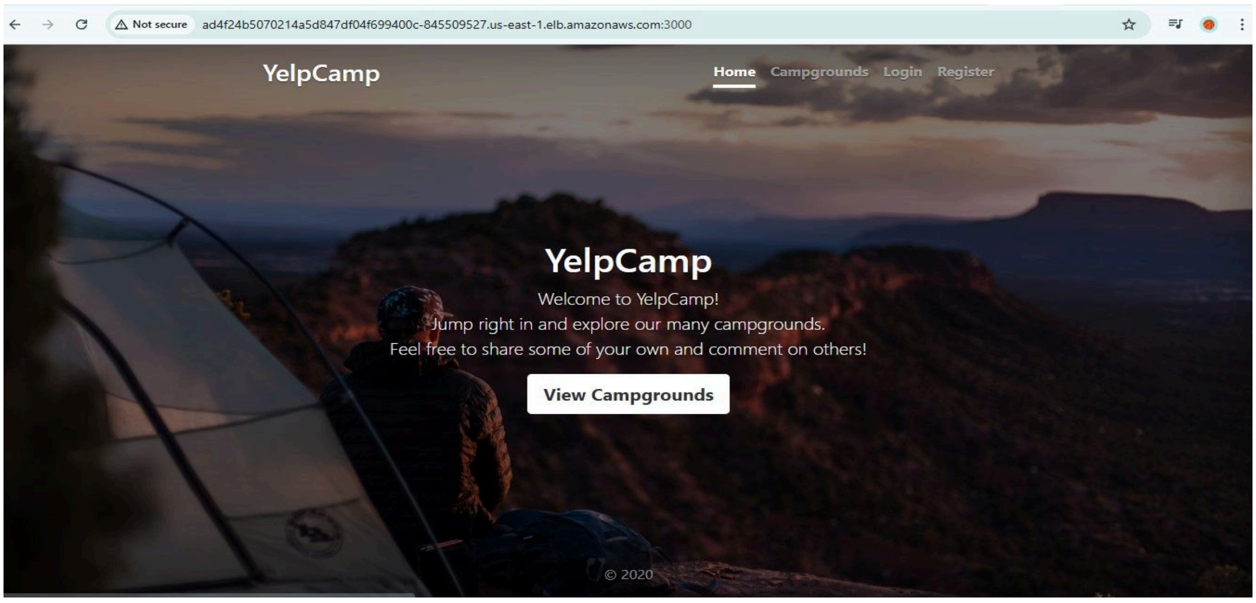

Not secure
ad4f24b5070214a5d847df04f699400c-845509527.us-east-1.elb.amazonaws.com:3000
YelpCamp
Home
Campgrounds
Login
Register
YelpCamp
Welcome to YelpCamp!
Jump right in and explore our many campgrounds.
Feel free to share some of your own and comment on others!
View Campgrounds
© 2020

Figure 3.8 : Final Deployed web application

Project 3: Devopsified and Deployment of Java Micronaut Web Application into k3s kubernetes cluster with custom domain , ssl certificates and monitoring with prometheus, grafana, newrelic.

In this project, I focused on complete DevOps implementation of a three-tier web application built using Micronaut, React, and ScyllaDB. The application was containerized, deployed in a self-hosted lightweight K3s Kubernetes cluster, and exposed securely over a custom domain with Let's Encrypt SSL certificates and Traefik Ingress.

i) Containerization and Image Management

The frontend (React) and backend (Micronaut) components were:

- **Dockerized** separately with production-ready Dockerfiles.
- **Pushed to Docker Hub** for use in Kubernetes deployment.
  This enabled version-controlled, reproducible deployments across environments.

ii) Kubernetes Deployment on K3s

A **K3s cluster** was manually provisioned and configured to run the application:

- Kubernetes **manifests** were written for deployments, services, and ingress rules.
- Each tier (frontend, backend, database) was independently deployed in its own pod.
- The backend services connected to a **ScyllaDB cluster** set up on two virtual machines.

iii) Ingress & Domain Setup with Traefik and SSL

Traefik was used as the Kubernetes ingress controller to handle traffic routing:

- **Ingress resources** were defined with custom routing rules for / and /api paths.
- The domain notes.ksaurav.com.np was mapped to the Traefik LoadBalancer External IP.
- Use Cloud Flare as a DNS Manager.
- **Let’s Encrypt certificates** were issued automatically using cert-manager and a configured ClusterIssuer.

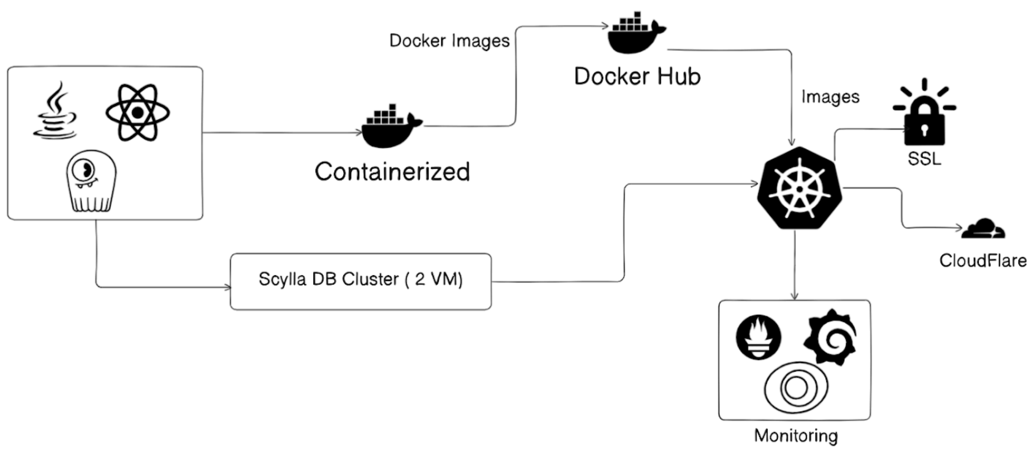
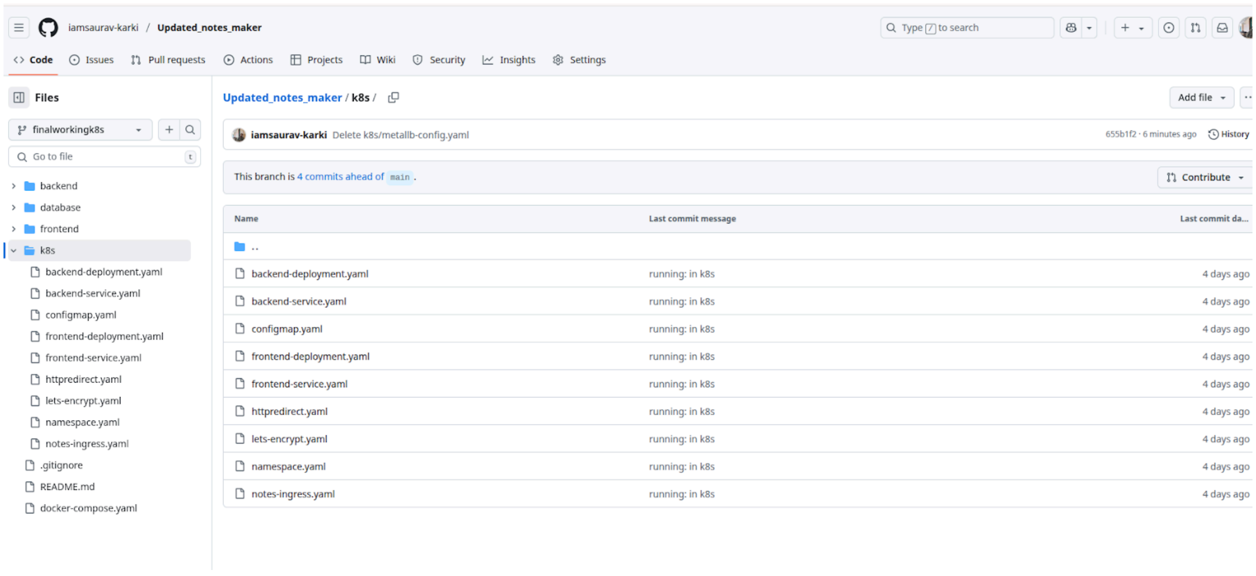


Figure 3.9 : Architecture Diagram of Deployment of Java Micronaut Web Application into k3s kubernetes cluster with custom domain , ssl certificates and monitoring with prometheus, grafana, newrelic.



Updated_notes_maker / k8s /

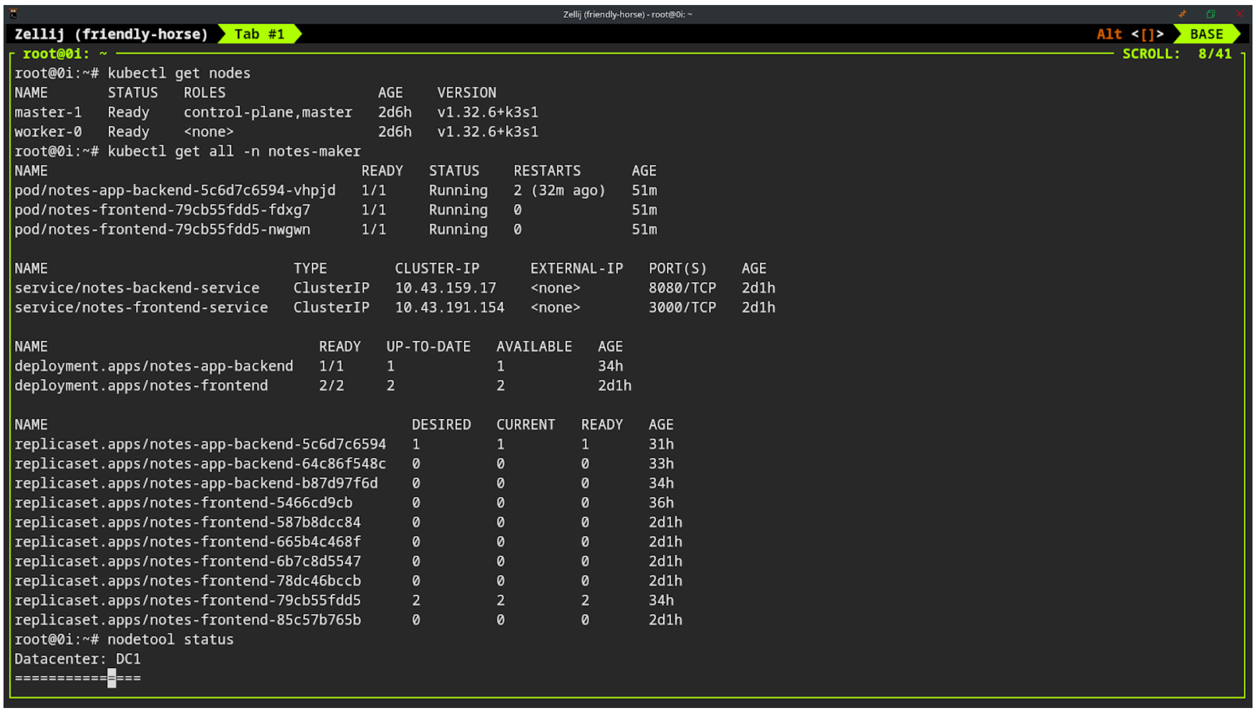
iamsaurav-karki Delete k8s/metallb-config.yaml 655b1f2 · 6 minutes ago

This branch is 4 commits ahead of `main`.

```
backend
database
frontend
k8s
    backend-deployment.yaml
    backend-service.yaml
    configmap.yaml
    frontend-deployment.yaml
    frontend-service.yaml
    httpredirect.yaml
    lets-encrypt.yaml
    namespace.yaml
    notes-ingress.yaml
.gitignore
README.md
docker-compose.yaml
```

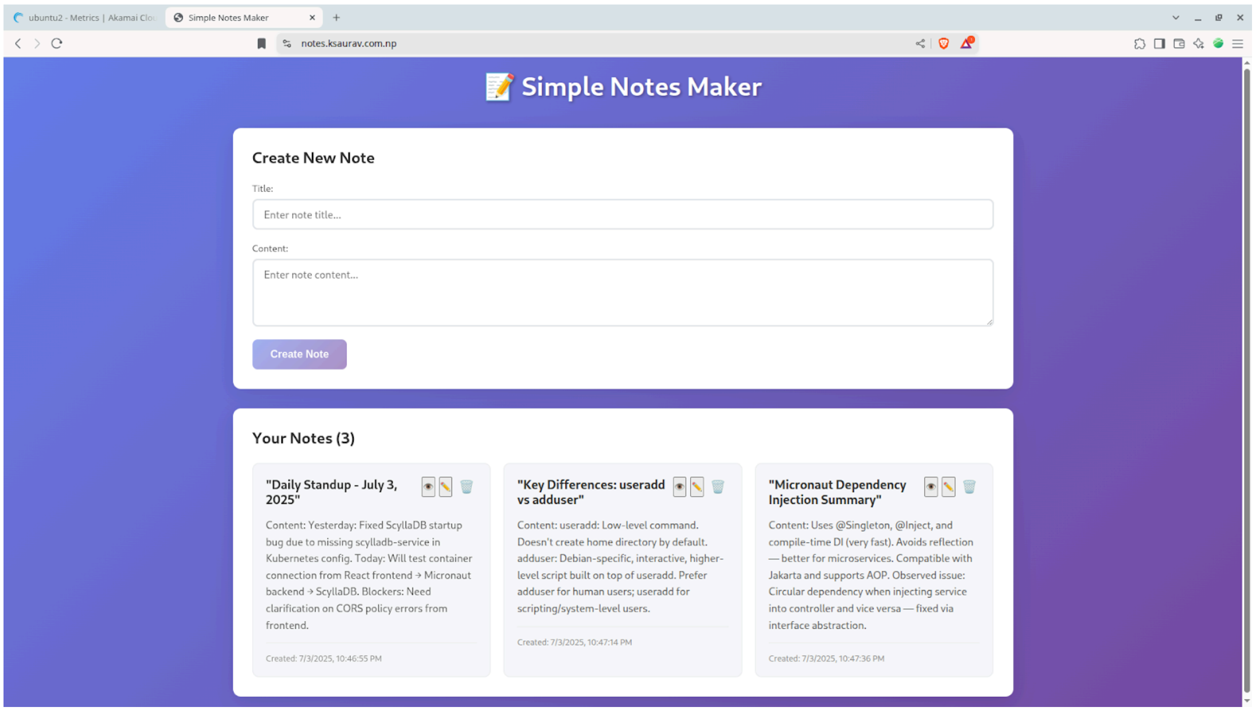
Name	Last commit message	Last commit da...
..		
backend-deployment.yaml	running: in k8s	4 days ago
backend-service.yaml	running: in k8s	4 days ago
configmap.yaml	running: in k8s	4 days ago
frontend-deployment.yaml	running: in k8s	4 days ago
frontend-service.yaml	running: in k8s	4 days ago
httpredirect.yaml	running: in k8s	4 days ago
lets-encrypt.yaml	running: in k8s	4 days ago
namespace.yaml	running: in k8s	4 days ago
notes-ingress.yaml	running: in k8s	4 days ago

Figure 3.10 : Kubernetes manifest structure of the project



```
root@0i:~# kubectl get nodes
NAME       STATUS   ROLES                  AGE    VERSION
master-1   Ready    control-plane,master   2d6h   v1.32.6+k3s1
worker-0   Ready    <none>                 2d6h   v1.32.6+k3s1
root@0i:~# kubectl get all -n notes-maker
NAME                                    READY   STATUS    RESTARTS      AGE
pod/notes-app-backend-5c6d7c6594-vhpjd  1/1     Running   2 (32m ago)   51m
pod/notes-frontend-79cb55fdd5-fdxg7     1/1     Running   0             51m
pod/notes-frontend-79cb55fdd5-nwgwn     1/1     Running   0             51m

NAME                             TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
service/notes-backend-service    ClusterIP   10.43.159.17    <none>        8080/TCP   2d1h
service/notes-frontend-service   ClusterIP   10.43.191.154   <none>        3000/TCP   2d1h

NAME                                 READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/notes-app-backend    1/1     1            1           34h
deployment.apps/notes-frontend       2/2     2            2           2d1h

NAME                                            DESIRED   CURRENT   READY   AGE
replicaset.apps/notes-app-backend-5c6d7c6594    1         1         1       31h
replicaset.apps/notes-app-backend-64c86f548c    0         0         0       33h
replicaset.apps/notes-app-backend-b87d97f6d     0         0         0       34h
replicaset.apps/notes-frontend-5466cd9cb        0         0         0       36h
replicaset.apps/notes-frontend-587b8dcc84       0         0         0       2d1h
replicaset.apps/notes-frontend-665b4c468f       0         0         0       2d1h
replicaset.apps/notes-frontend-6b7c8d5547       0         0         0       2d1h
replicaset.apps/notes-frontend-78dc46bccb       0         0         0       2d1h
replicaset.apps/notes-frontend-79cb55fdd5       2         2         2       34h
replicaset.apps/notes-frontend-85c57b765b       0         0         0       2d1h
root@0i:~# nodetool status
Datacenter: DC1
===============
```

Figure 3.11 : Details of the nodes and running resources in k8s cluster



ubuntu2 - Metrics | Akamai Clo | Simple Notes Maker

notes.ksaurav.com.np

📝 Simple Notes Maker

Create New Note

Title:

Enter note title...

Content:

Enter note content...

Create Note

Your Notes (3)

"Daily Standup - July 3, 2025"

Content: Yesterday: Fixed ScyllaDB startup bug due to missing scylladb-service in Kubernetes config. Today: Will test container connection from React frontend → Micronaut backend → ScyllaDB. Blockers: Need clarification on CORS policy errors from frontend.

Created: 7/3/2025, 10:46:55 PM

"Key Differences: useradd vs adduser"

Content: useradd: Low-level command. Doesn't create home directory by default. adduser: Debian-specific, interactive, higher-level script built on top of useradd. Prefer adduser for human users; useradd for scripting/system-level users.

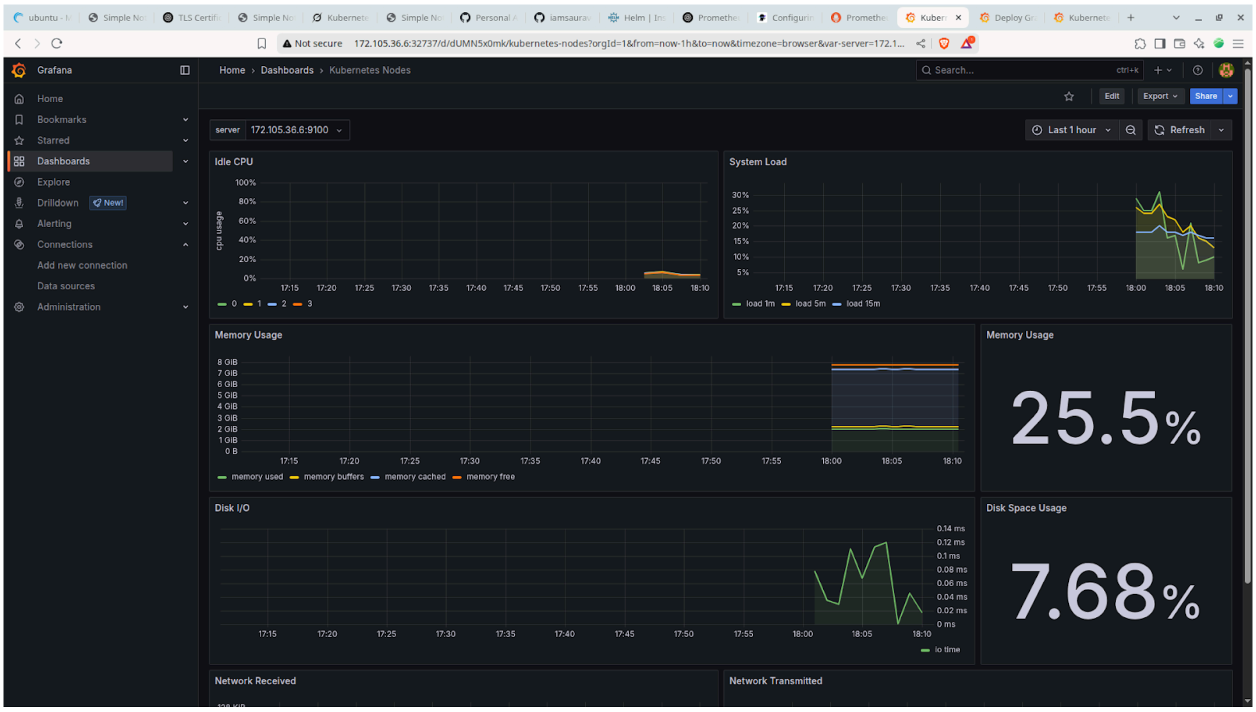
Created: 7/3/2025, 10:47:14 PM

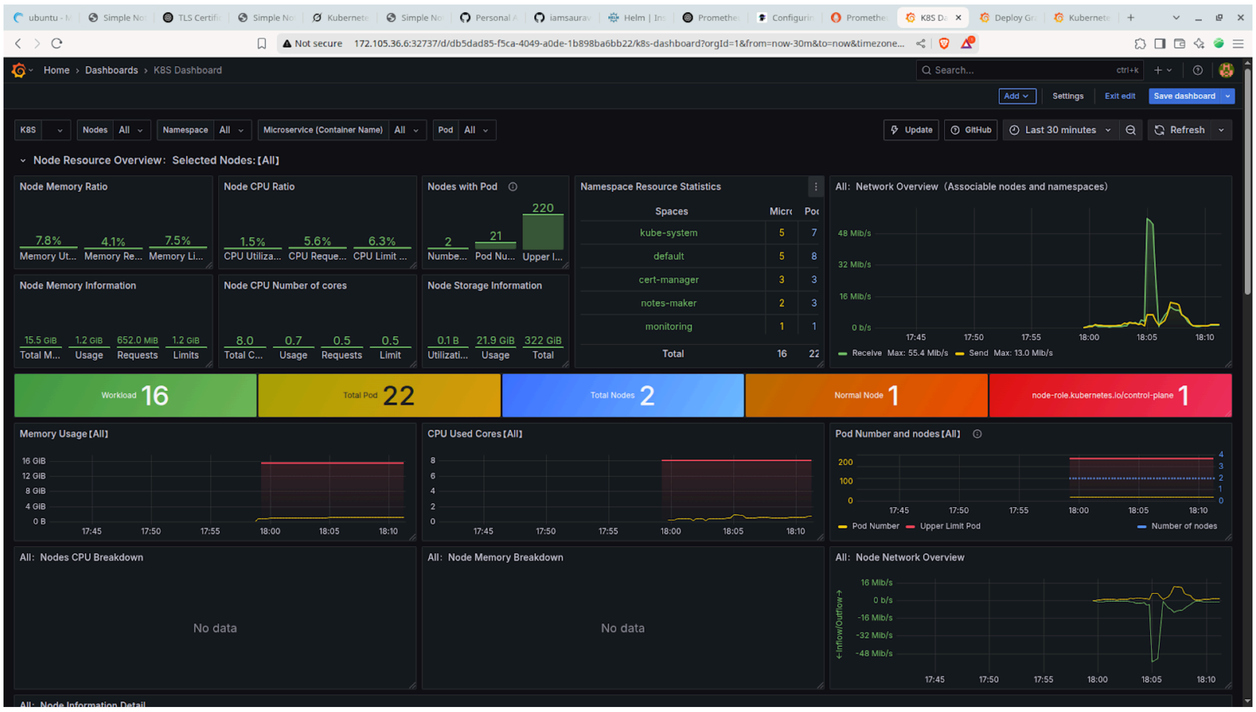
"Micronaut Dependency Injection Summary"

Content: Uses @Singleton, @Inject, and compile-time DI (very fast). Avoids reflection — better for microservices. Compatible with Jakarta and supports AOP. Observed issue: Circular dependency when injecting service into controller and vice versa — fixed via interface abstraction.

Created: 7/3/2025, 10:47:36 PM

Figure 3.12 : Final Deployed Three Tier notes maker app with SSL certificates and Custom Domain.

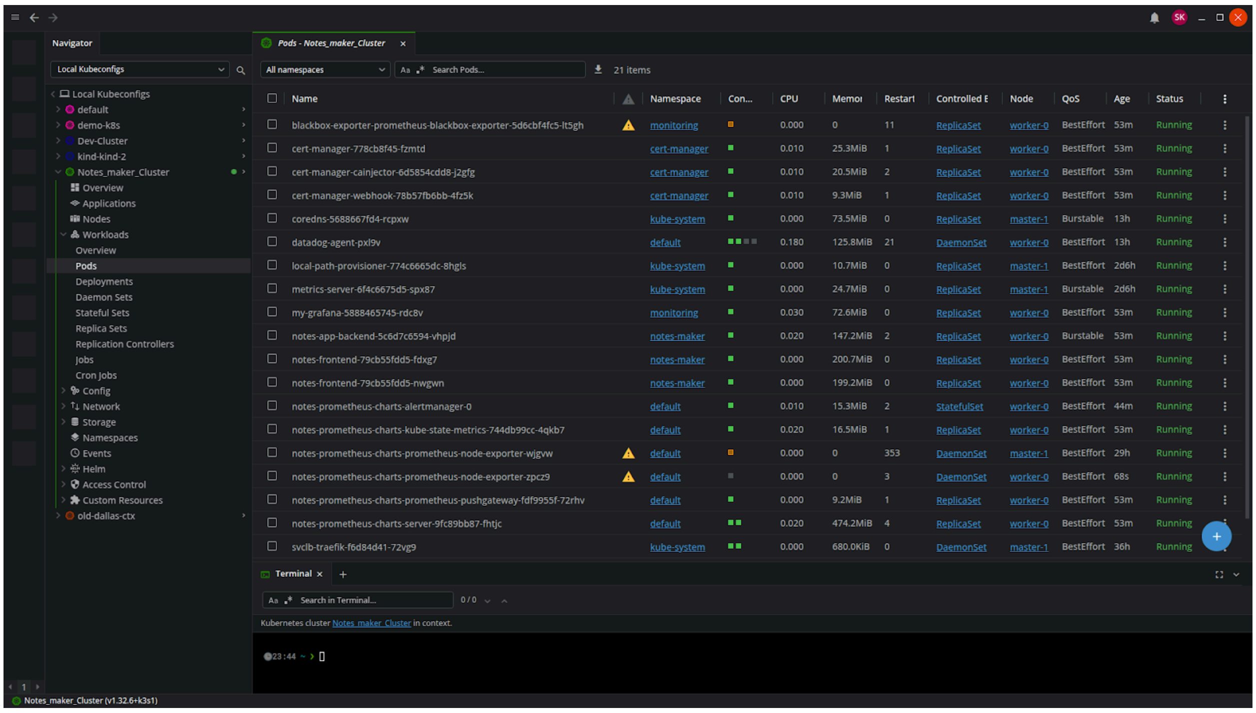




Home › Dashboards › K8S Dashboard
Node Resource Overview: Selected Nodes: [All]
Node Memory Ratio
7.8% 4.1% 7.5%
Memory Ut... Memory Re... Memory Li...
Node CPU Ratio
1.5% 5.6% 6.3%
CPU Utiliza... CPU Reque... CPU Limit ...
Nodes with Pod
2 21 220
Numbe... Pod Nu... Upper l...
Namespace Resource Statistics
Node Memory Information
15.5 GiB 1.2 GiB 652.0 MiB 1.2 GiB
Total M... Usage Requests Limits
Node CPU Number of cores
8.0 0.7 0.5 0.5
Total C... Usage Requests Limit
Node Storage Information
0.1 B 21.9 GiB 322 GiB
Utilizati... Usage Total
All: Network Overview (Associable nodes and namespaces)
Receive Max: 55.4 Mib/s
Send Max: 13.0 Mib/s
Workload 16
Total Pod 22
Total Nodes 2
Normal Node 1
node-role.kubernetes.io/control-plane 1
Memory Usage [All]
CPU Used Cores [All]
Pod Number and nodes [All]
Pod Number
Upper Limit Pod
Number of nodes
All: Nodes CPU Breakdown
No data
All: Node Memory Breakdown
No data
All: Node Network Overview
All: Node Information Detail

Spaces	Micr	Po
kube-system	5	7
default	5	8
cert-manager	3	3
notes-maker	2	3
monitoring	1	1
Total	16	22

Figure 3.13 : Grafana Dashboard for application metrics monitoring



Name		Namespace	Con...	CPU	Memor	Restart	Controlled E	Node	QoS	Age	Status
blackbox-exporter-prometheus-blackbox-exporter-5d6cbf4fc5-lt5gh	⚠	monitoring	🟧	0.000	0	11	ReplicaSet	worker-0	BestEffort	53m	Running
cert-manager-778cb8f45-fzmtd		cert-manager	🟩	0.010	25.3MiB	1	ReplicaSet	worker-0	BestEffort	53m	Running
cert-manager-cainjector-6d5854cdd8-j2gfg		cert-manager	🟩	0.010	20.5MiB	2	ReplicaSet	worker-0	BestEffort	53m	Running
cert-manager-webhook-78b57fb6bb-4fz5k		cert-manager	🟩	0.010	9.3MiB	1	ReplicaSet	worker-0	BestEffort	53m	Running
coredns-5688667fd4-rcpxw		kube-system	🟩	0.000	73.5MiB	0	ReplicaSet	master-1	Burstable	13h	Running
datadog-agent-pxl9v		default	🟩🟩⬛⬛	0.180	125.8MiB	21	DaemonSet	worker-0	BestEffort	13h	Running
local-path-provisioner-774c6665dc-8hgls		kube-system	🟩	0.000	10.7MiB	0	ReplicaSet	master-1	BestEffort	2d6h	Running
metrics-server-6f4c6675d5-spx87		kube-system	🟩	0.000	24.7MiB	0	ReplicaSet	master-1	Burstable	2d6h	Running
my-grafana-5888465745-rdc8v		monitoring	🟩	0.030	72.6MiB	0	ReplicaSet	worker-0	BestEffort	53m	Running
notes-app-backend-5c6d7c6594-vhpjd		notes-maker	🟩	0.020	147.2MiB	2	ReplicaSet	worker-0	Burstable	53m	Running
notes-frontend-79cb55fdd5-fdxg7		notes-maker	🟩	0.000	200.7MiB	0	ReplicaSet	worker-0	BestEffort	53m	Running
notes-frontend-79cb55fdd5-nwgwn		notes-maker	🟩	0.000	199.2MiB	0	ReplicaSet	worker-0	BestEffort	53m	Running
notes-prometheus-charts-alertmanager-0		default	🟩	0.010	15.3MiB	2	StatefulSet	worker-0	BestEffort	44m	Running
notes-prometheus-charts-kube-state-metrics-744db99cc-4qkb7		default	🟩	0.020	16.5MiB	1	ReplicaSet	worker-0	BestEffort	53m	Running
notes-prometheus-charts-prometheus-node-exporter-wjgvw	⚠	default	🟧	0.000	0	353	DaemonSet	master-1	BestEffort	29h	Running
notes-prometheus-charts-prometheus-node-exporter-zpcz9	⚠	default	⬛	0.000	0	3	DaemonSet	worker-0	BestEffort	68s	Running
notes-prometheus-charts-prometheus-pushgateway-fdf9955f-72rhv		default	🟩	0.000	9.2MiB	1	ReplicaSet	worker-0	BestEffort	53m	Running
notes-prometheus-charts-server-9fc89bb87-fhtjc		default	🟩🟩	0.020	474.2MiB	4	ReplicaSet	worker-0	BestEffort	53m	Running
svclb-traefik-f6d84d41-72vg9		kube-system	🟩🟩	0.000	680.0KiB	0	DaemonSet	master-1	BestEffort	36h	Running

```
Kubernetes cluster Notes_maker_Cluster in context.

23:44 ~ >
```

Figure 3.14 : Lens Dashboard for Cluster Management

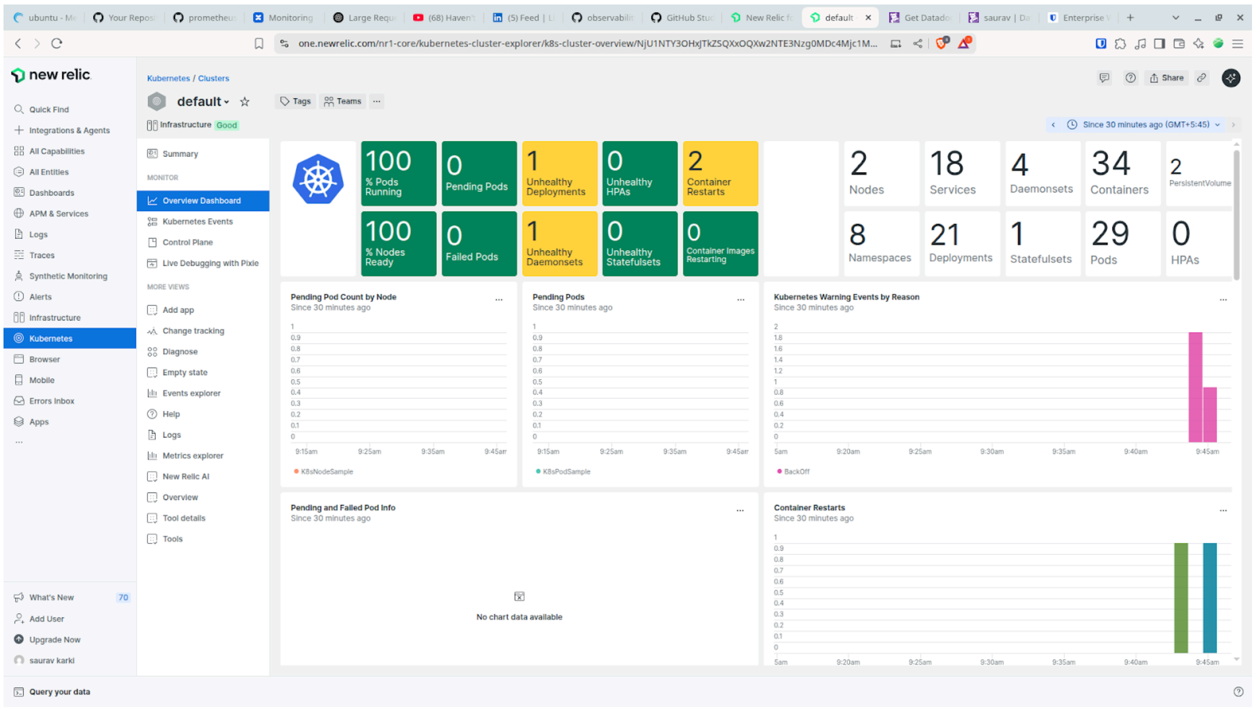


Figure 3.15 : New Relic Dashboard for K8s cluster and resource monitoring

3.4. Description of the Tools Used

During the internship, I worked extensively with a variety of DevOps, containerization, cloud-native, and infrastructure tools. These tools were instrumental in enabling automation, scalability, monitoring, secure deployments, and production-level reliability for the projects I was involved in.

1. Docker & Docker Compose

Docker was used extensively for containerizing applications, managing dependencies, and creating reproducible environments. Docker Compose simplified multi-container orchestration by defining services, volumes, and networking in a single YAML file.

Key Use Cases:

- Containerization of Java Spring Boot, Micronaut, and React applications.
- Orchestration of multi-tier applications with databases like PostgreSQL.
- Streamlining development-to-deployment lifecycle.

2. Kubernetes (RKE2, K3s, EKS)

Kubernetes served as the primary orchestration platform for managing containerized applications. Both lightweight (K3s) and production-grade (RKE2, EKS) clusters were used.

Key Use Cases:

- Deployment of applications like YelpCamp and Notes Maker.
- Setup of monitoring, ingress (Traefik, Envoy Gateway), and service routing.
- Handling scalability, fault tolerance, and zero-downtime deployments.

3. Jenkins

Jenkins was used to implement a CI/CD pipeline, automating build, test, scan, and deployment workflows for containerized applications.

Key Use Cases:

- Triggering CI/CD workflows on code changes.
- Integrating tools like SonarQube (code quality) and Trivy (security scanning).
- Deploying Docker images to environments via Kubernetes manifests.

4. HashiCorp Vault

Vault was deployed inside a Kubernetes cluster to securely manage and inject secrets into pods using Vault Agent Sidecar Injector and Kubernetes Auth method.

Key Use Cases:

- Dynamic secrets management for applications.
- Secure injection of credentials into pods via annotations.
- Integration with Kubernetes Service Accounts and Roles.

5. Helm

Helm acted as the package manager for Kubernetes, simplifying the deployment of complex applications like Vault, Envoy Gateway, NATS, and more using reusable charts.

Key Use Cases:

- Installing third-party services with custom values (e.g., vault-values.yaml).
- Managing version-controlled, declarative deployments.

6. Traefik Ingress Controller & Envoy Gateway

Both Traefik and Envoy Gateway were explored and deployed as ingress controllers to expose applications externally with custom domains and routing rules.

Key Use Cases:

- Custom domain setup with SSL using Let’s Encrypt via cert-manager.
- HTTP routing to internal services with annotations and HTTPRoute.
- Secure TLS termination and middleware configurations (e.g., HTTPS redirect).

7. ScyllaDB

ScyllaDB, a high-performance NoSQL database, was set up as a multi-node cluster for backend services needing high throughput and low latency.

Key Use Cases:

- Clustered database setup across VMs for high availability.
- Use of CQL for querying and testing replication.
- Backend storage for notes maker app.

8. MinIO

MinIO was deployed locally to simulate S3-compatible object storage with replication and policy-based access control.

Key Use Cases:

- Object storage for application backups, logs, and assets.
- Automated file upload and replication via Python SDK.
- Policy and bucket-level access configuration for secure storage.

9. Monitoring Tools (Prometheus, Grafana, New Relic, Octant, Lens)

A robust monitoring stack was set up to collect metrics, visualize system health, and debug deployments.

Key Use Cases:

- Prometheus for scraping application metrics
- Grafana and New Relic for visualizing cluster and app-level performance.
- Lens, Octant, and Kubernetes Dashboard for real-time cluster insights.

10. Flux CD

Flux CD was used for GitOps-based continuous delivery in the Kubernetes cluster, allowing automatic application updates from version-controlled Git repositories.

Key Use Cases:

- Declarative deployment management with Git as the source of truth.
- Image update automation using Image Automation Controllers.
- Policy-based delivery and rollback.

11. NATS & Kafka

Both NATS and Kafka were explored for real-time messaging and distributed communication.

- Deployment via Helm in Kubernetes.
- Introduction to publisher-subscriber architecture.
- Scalable messaging solutions for microservices.

12. Proxmox VE

Proxmox was used for virtualization and VM management, including creation of VM templates and running Linux containers (LXC).

Key Use Cases:

- Hosting Kubernetes clusters and database nodes in VMs.
- VM backup to S3 using DigitalOcean-compatible buckets.

3.5. Tasks / Activities Performed

3.5.1. Integrated HashiCorp Vault For secret management of Kubernetes Pods.

In this task, I have installed and set up a hashicorp vault in k8s cluster to manage kubernetes secrets for advanced security. Steps I followed:

Added helm repo for hashicorp and installed vault using helm releases:

- helm repo add hashicorp https://helm.releases.hashicorp.com
- helm repo update
- helm install vault hashicorp/vault -n vault -f vault-values.yaml

vault-values.yaml

```
server:
  ha:
    enabled: true
    replicas: 2
    raft:
      enabled: true
      config: |
        ui = true
        listener "tcp" {
          address = "0.0.0.0:8200"
          cluster_address = "0.0.0.0:8201"
          tls_disable = 1
        }
        storage "raft" {
          path = "/vault/data"
        }
        service_registration "kubernetes" {}
    dataStorage:
      enabled: true
      size: 10Gi
      storageClass: "local-path"
    extraEnvironmentVars:
      VAULT_LOG_LEVEL: "debug"
injector:
  enabled: true
ui:
  enabled: true
```

Create policy for access to secrets of hashicorp vault:

```
path “secret/data/mysql” { capabilities = [”create”, “update”, “read”, “delete”, “list” ]
}
path “secret/data/frontend” { capabilities = [”create”, “update”, “read”, “delete”, “list” ]
}
path “secret/metadata/mysql” { capabilities = [“list” ]
}
```

Created Role and policy is attached to the role:

```
kubectl exec -n vault -it vault-0 — vault write auth/kubernetes/role/vault-role \
bound_service_account_names=vault-auth \
bound_service_account_namespaces=”webapps” \
policies=myapp-policy \
ttl=24h
```

Now I have created the service account and attached the role to the service account so that the pod associated with the service account can get access to the secrets stored in the vault.

```
apiVersion: v1

kind: ServiceAccount

metadata: name: vault-auth

namespace: webapps
```

Now used the annotations to the deployment so that the pod can get and use the secrets:

```
annotations:
vault.hashicorp.com/agent-inject: “true”
vault.hashicorp.com/role: “vault-role”
vault.hashicorp.com/agent-inject-secret-MYSQL_ROOT_PASSWORD: “secret/mysql”
vault.hashicorp.com/agent-inject-template-MYSQL_ROOT_PASSWORD: |
  {{- with secret “secret/mysql” -}}
    export MYSQL_ROOT_PASSWORD=”{{ .Data.data.MYSQL_ROOT_PASSWORD }}”
  {{- end }}
```

3.5.2. Deployed Portainer for container management.

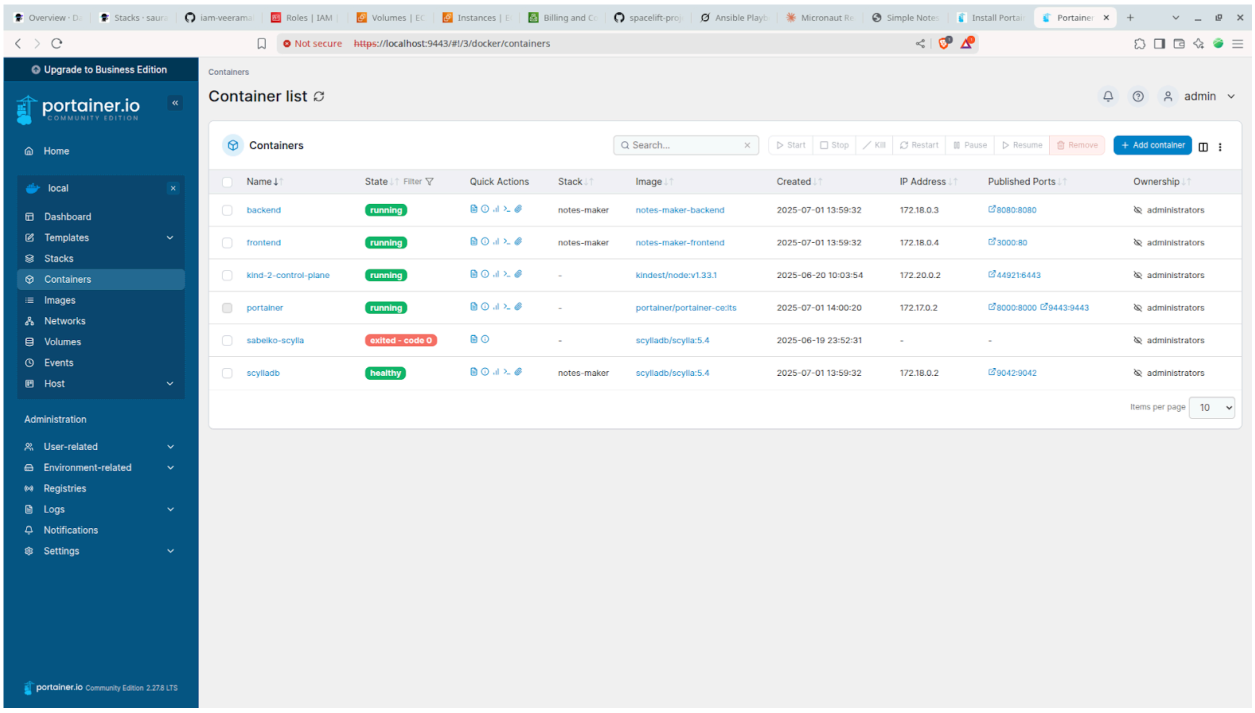
During this task, **Portainer** was deployed on local host machines to provide a graphical interface for managing Docker containers and resources running on the local system.

Portainer is a lightweight, open-source container management tool that simplifies container operations such as monitoring, creating, starting, stopping, and removing containers.

Key Steps and Features:

- Pulled and deployed the official portainer/portainer-ce image using Docker.
- Exposed Portainer for local UI access.
- Used Portainer to: Monitor running containers and their resource usage (CPU, memory, ports),Manage container lifecycle operations (start, stop, remove), view container logs, networks, and volumes.

- $ docker volume create portainer_data
- $ docker run -d -p 8000:8000 -p 9443:9443 --name portainer --restart=always -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-ce:lts
- $ docker ps


Container list

Name	State	Quick Actions	Stack	Image	Created	IP Address	Published Ports	Ownership
backend	running		notes-maker	notes-maker-backend	2025-07-01 13:59:32	172.18.0.3	8080:8080	administrators
frontend	running		notes-maker	notes-maker-frontend	2025-07-01 13:59:32	172.18.0.4	3000:80	administrators
kind-2-control-plane	running		-	kindest/node:v1.33.1	2025-06-20 10:03:54	172.20.0.2	44921:6443	administrators
portainer	running		-	portainer/portainer-ce:lts	2025-07-01 14:00:20	172.17.0.2	8000:8000 9443:9443	administrators
sabeiko-scylla	exited - code 0		-	scylladb/scylla:5.4	2025-06-19 23:52:31	-	-	administrators
scylladb	healthy		notes-maker	scylladb/scylla:5.4	2025-07-01 13:59:32	172.18.0.2	9042:9042	administrators

Figure 3.16: Portainer Dashboard for container management

3.5.3. Setup Scylla DB Cluster.

As part of this task, a **ScyllaDB NoSQL database cluster** was set up on multiple virtual machines to provide a highly available, horizontally scalable backend for applications requiring low-latency and high-throughput data access. The deployment focused on creating a basic multi-node cluster with replication and internal communication enabled between the nodes.

Key Steps and Configuration:

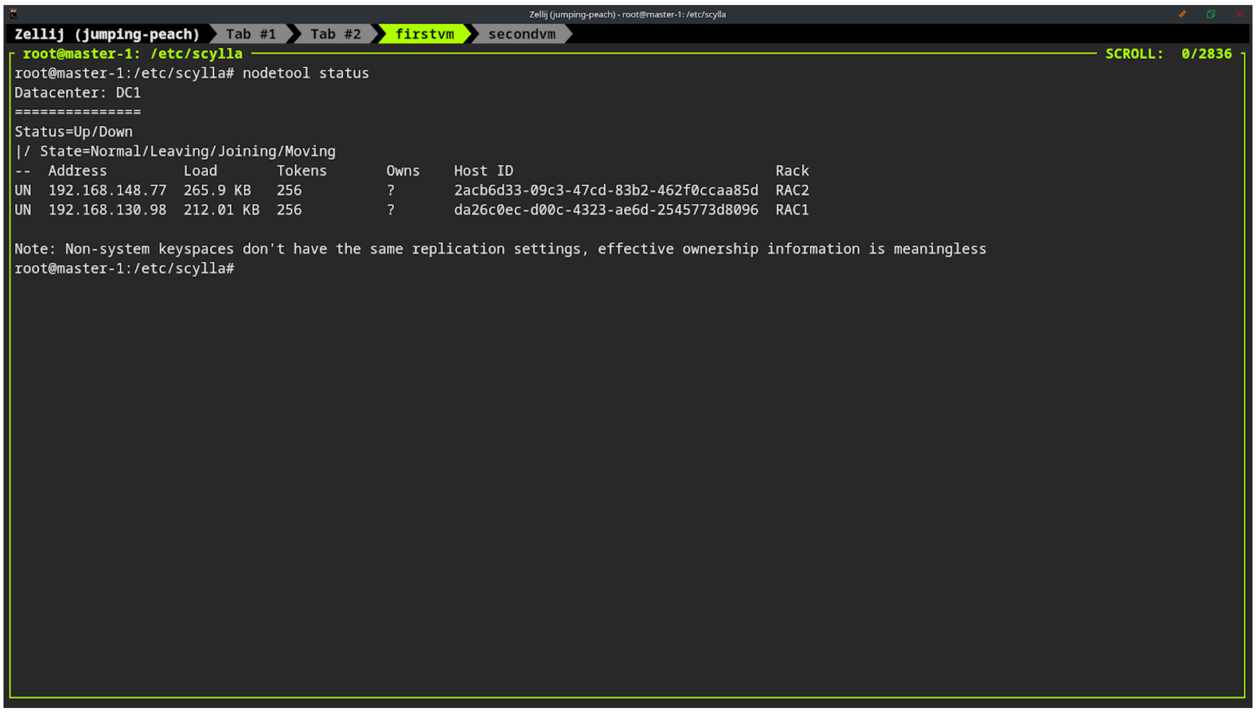
- Provisioned **two Linux-based virtual machines** with private and public IP addresses for the ScyllaDB nodes.
- Installed the latest ScyllaDB Enterprise/Open Source version on both VMs.
- Configured scylla.yaml on each node to: Assign cluster name and seed nodes, Bind to the correct IP address, Enable gossip protocol for node discovery
- Started Scylla services on both nodes and verified cluster formation via nodetool status.

Steps:-

- sudo mkdir -p /etc/apt/keyrings
- sudo gpg --homedir /tmp --no-default-keyring --keyring /etc/apt/keyrings/scylladb.gpg --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys a43e06657bac99e3
- sudo wget -O /etc/apt/sources.list.d/scylla.list
- http://downloads.scylladb.com/deb/debian/scylla-2025.1.list
- sudo apt-get update
- sudo apt-get install -y scylla
- sudo apt-get update
- sudo apt-get install -y openjdk-11-jre-headless
- sudo update-java-alternatives --jre-headless -s java-1.11.0-openjdk-amd64
- Configure and Run ScyllaDB

Configure the following parameters in the /etc/scylla/scylla.yaml configuration file.

- cluster_name - <Name>
- seeds - The IP address of the first node
- listen_address - The IP address that ScyllaDB uses to connect to other nodes in the cluster.
- rpc_address - The IP address of the interface for CQL client connections.
- sudo scylla_setup
- sudo systemctl start scylla-server
- nodetool status


Zellij (jumping-peach) - root@master-1: /etc/scylla
Zellij (jumping-peach)
Tab #1
Tab #2
firstvm
secondvm
root@master-1: /etc/scylla
SCROLL: 0/2836

```
root@master-1:/etc/scylla# nodetool status
Datacenter: DC1
===============
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address         Load       Tokens       Owns    Host ID                               Rack
UN  192.168.148.77  265.9 KB   256          ?       2acb6d33-09c3-47cd-83b2-462f0ccaa85d  RAC2
UN  192.168.130.98  212.01 KB  256          ?       da26c0ec-d00c-4323-ae6d-2545773d8096  RAC1

Note: Non-system keyspaces don't have the same replication settings, effective ownership information is meaningless
root@master-1:/etc/scylla#
```

Figure 3.17: Verifying the status of the scylla db cluster

3.5.4. Minio For Object Based Storage.

As part of this task, MinIO was deployed on a local Linux host to serve as a high-performance, S3-compatible object storage system, enabling structured and unstructured data storage, such as files, backups, and application assets. Additionally, bucket-level replication was configured so that objects uploaded to one bucket are automatically replicated to another, simulating a high-availability or backup-ready setup.

i) Installation and Setup

MinIO was installed in standalone mode using the official Linux binary:

- A dedicated directory (~/minio) was created for storage.

MinIO server was launched with:

```
minio server ~/minio --console-address :9001
```

- Accessed via:
  - API: http://127.0.0.1:9000
  - Console: http://127.0.0.1:9001
- Default credentials minioadmin:minioadmin were used during initial login and later secured.

ii) Bucket Creation and Policy Setup

- Created two buckets:
  - Source bucket: 00sauravnew
  - Target bucket: 00sauravnewmain
- Applied a custom replication policy (replication-policy.json) using MinIO's S3-compatible API.
- Granted necessary permissions in the policy for replication actions such as s3:ReplicateObject, s3:PutObject, and s3:DeleteObject.

iii) Replication Configuration

Set up server-side replication using MinIO's built-in functionality:

- Defined a JSON-based replication policy that:
  - Watches 00sauravnew (source)
  - Automatically replicates all changes (put, delete, tags) to 00sauravnewmain (target)
- Applied the replication configuration using the mc (MinIO Client) or SDK.

Example snippet of replication policy:

```
{
  "Effect": "Allow",
  "Action": ["s3:ReplicateObject", "s3:ReplicateDelete", "s3:ReplicateTags"],
  "Resource": ["arn:aws:s3:::00sauravnew/*"]
}
```

iv) Python SDK Integration and Automation

Developed a Python automation script (file_uploader.py) using MinIO Python SDK:

- Uploaded objects with custom metadata and tags.
- Enabled optional retention and legal hold policies.
- Automatically verified replication by listing objects in the target bucket after upload.
- Generated pre-signed URLs for controlled public access to objects.

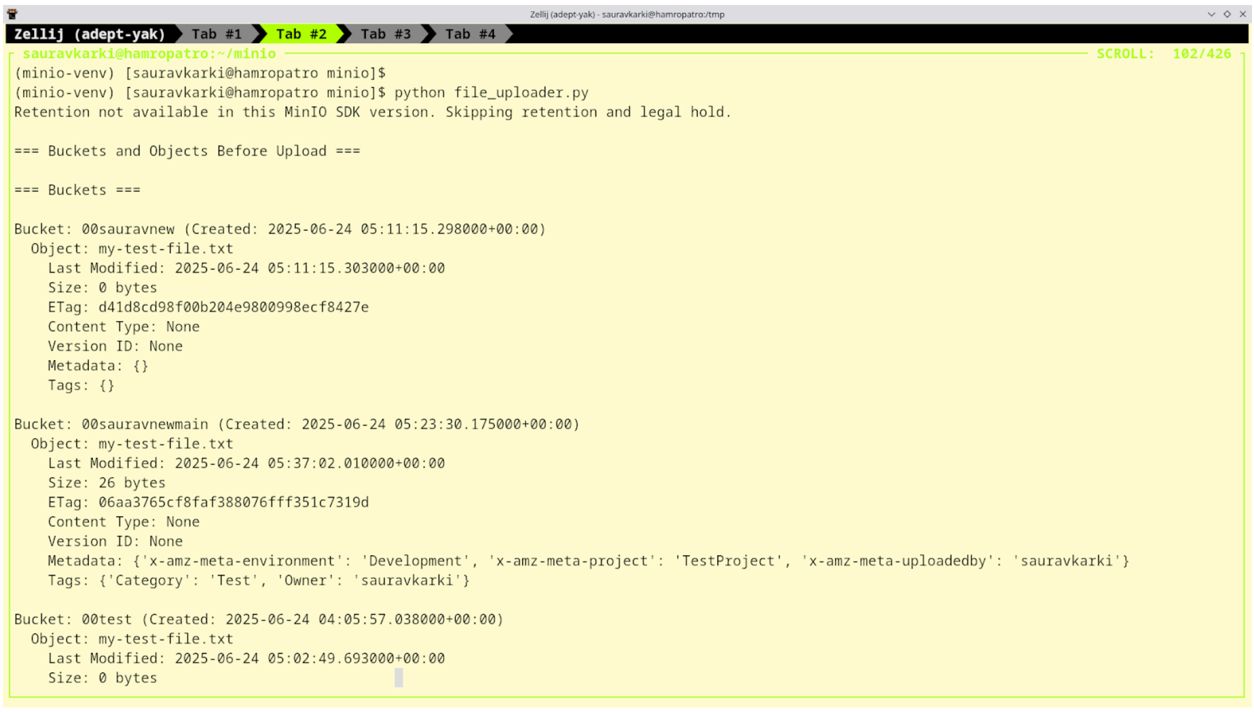
v) Monitoring and Administration via CLI

Installed the MinIO Client (mc):

- Used mc alias set to connect to the local MinIO instance.

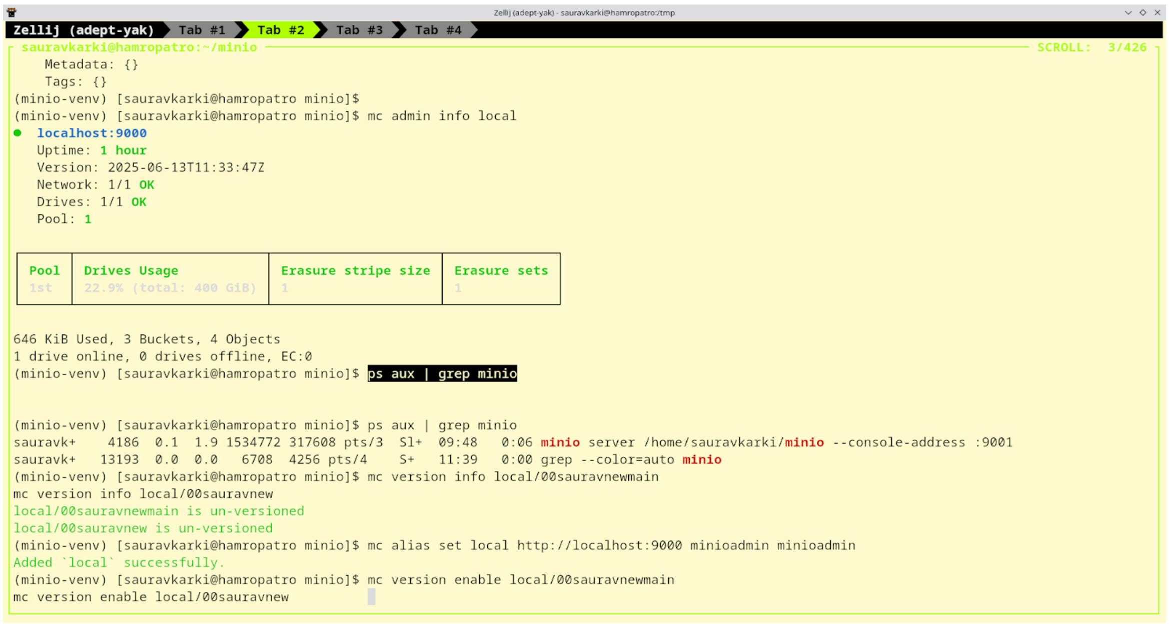
Verified bucket and object replication using:

mc admin info local && mc ls local/00sauravnewmain



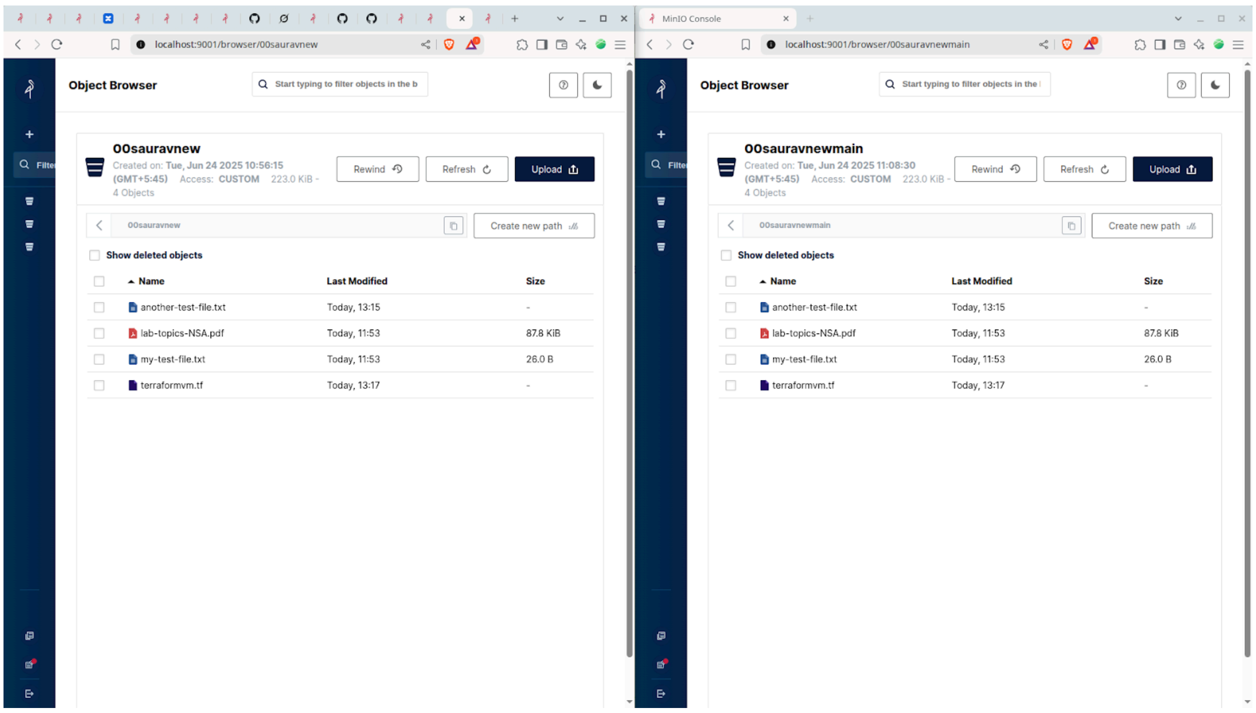
```
Zellij (adept-yak) - sauravkarki@hamropatro:/tmp
Zellij (adept-yak)  Tab #1  Tab #2  Tab #3  Tab #4
sauravkarki@hamropatro:~/minio                                              SCROLL:  102/426
(minio-venv) [sauravkarki@hamropatro minio]$
(minio-venv) [sauravkarki@hamropatro minio]$ python file_uploader.py
Retention not available in this MinIO SDK version. Skipping retention and legal hold.

=== Buckets and Objects Before Upload ===

=== Buckets ===

Bucket: 00sauravnew (Created: 2025-06-24 05:11:15.298000+00:00)
  Object: my-test-file.txt
    Last Modified: 2025-06-24 05:11:15.303000+00:00
    Size: 0 bytes
    ETag: d41d8cd98f00b204e9800998ecf8427e
    Content Type: None
    Version ID: None
    Metadata: {}
    Tags: {}

Bucket: 00sauravnewmain (Created: 2025-06-24 05:23:30.175000+00:00)
  Object: my-test-file.txt
    Last Modified: 2025-06-24 05:37:02.010000+00:00
    Size: 26 bytes
    ETag: 06aa3765cf8faf388076fff351c7319d
    Content Type: None
    Version ID: None
    Metadata: {'x-amz-meta-environment': 'Development', 'x-amz-meta-project': 'TestProject', 'x-amz-meta-uploadedby': 'sauravkarki'}
    Tags: {'Category': 'Test', 'Owner': 'sauravkarki'}

Bucket: 00test (Created: 2025-06-24 04:05:57.038000+00:00)
  Object: my-test-file.txt
    Last Modified: 2025-06-24 05:02:49.693000+00:00
    Size: 0 bytes
```

Figure 3.18: Verifying the files in the bucket



```
Zellij (adept-yak) - sauravkarki@hamropatro:/tmp
Zellij (adept-yak)  Tab #1  Tab #2  Tab #3  Tab #4
sauravkarki@hamropatro:~/minio                                              SCROLL:  3/426
    Metadata: {}
    Tags: {}
(minio-venv) [sauravkarki@hamropatro minio]$
(minio-venv) [sauravkarki@hamropatro minio]$ mc admin info local
●  localhost:9000
   Uptime: 1 hour
   Version: 2025-06-13T11:33:47Z
   Network: 1/1 OK
   Drives: 1/1 OK
   Pool: 1

┌──────┬────────────────────────┬─────────────────────┬───────────────┐
│ Pool │ Drives Usage           │ Erasure stripe size │ Erasure sets  │
│ 1st  │ 22.9% (total: 400 GiB) │ 1                   │ 1             │
└──────┴────────────────────────┴─────────────────────┴───────────────┘

646 KiB Used, 3 Buckets, 4 Objects
1 drive online, 0 drives offline, EC:0
(minio-venv) [sauravkarki@hamropatro minio]$ ps aux | grep minio


(minio-venv) [sauravkarki@hamropatro minio]$ ps aux | grep minio
sauravk+    4186  0.1  1.9 1534772 317608 pts/3  Sl+  09:48   0:06 minio server /home/sauravkarki/minio --console-address :9001
sauravk+   13193  0.0  0.0   6708  4256 pts/4    S+   11:39   0:00 grep --color=auto minio
(minio-venv) [sauravkarki@hamropatro minio]$ mc version info local/00sauravnewmain
mc version info local/00sauravnew
local/00sauravnewmain is un-versioned
local/00sauravnew is un-versioned
(minio-venv) [sauravkarki@hamropatro minio]$ mc alias set local http://localhost:9000 minioadmin minioadmin
Added `local` successfully.
(minio-venv) [sauravkarki@hamropatro minio]$ mc version enable local/00sauravnewmain
mc version enable local/00sauravnew
```

Figure 3.19: Verifying the bucket



00sauravnew
Created on: Tue, Jun 24 2025 10:56:15 (GMT+5:45) Access: CUSTOM 223.0 KiB - 4 Objects

Name	Last Modified	Size
another-test-file.txt	Today, 13:15	-
lab-topics-NSA.pdf	Today, 11:53	87.8 KiB
my-test-file.txt	Today, 11:53	26.0 B
terraformvm.tf	Today, 13:17	-

localhost:9001/browser/00sauravnewmain
Object Browser
Start typing to filter objects in the
Rewind
Refresh
Upload
Create new path
Show deleted objects

00sauravnewmain
Created on: Tue, Jun 24 2025 11:08:30 (GMT+5:45) Access: CUSTOM 223.0 KiB - 4 Objects

Name	Last Modified	Size
another-test-file.txt	Today, 13:15	-
lab-topics-NSA.pdf	Today, 11:53	87.8 KiB
my-test-file.txt	Today, 11:53	26.0 B
terraformvm.tf	Today, 13:17	-

Figure 3.20: Verifying the replication of files in a bucket

3.5.5. Deployed Envoy Gateway as Ingress Controller in RKE2 Kubernetes Cluster.

Envoy Gateway was deployed in an RKE2 cluster to serve as a modern, scalable ingress controller. Envoy Gateway provides dynamic traffic routing, security, observability, and extensibility, making it ideal for cloud-native microservices architecture.

i) Helm-Based Deployment of Envoy Gateway

Envoy Gateway was installed using the official Helm chart from DockerHub:

```
helm install eg oci://docker.io/envoyproxy/gateway-helm \
  --version v1.4.2 \
  -n envoy-gateway-system \
  --create-namespace
```

Key setup steps included:

- Created a dedicated namespace: envoy-gateway-system
- Installed the Gateway API CRDs (Custom Resource Definitions) for defining GatewayClass, Gateway, and HTTPRoute
- Waited for the envoy-gateway deployment to become available and healthy

ii) Gateway Configuration

Applied the official quickstart.yaml to:

- Register the GatewayClass that specifies Envoy as the controller
- Create a Gateway resource that listens on port 80
- Define HTTPRoute objects to route incoming traffic to backend services

iii) CRDs and Advanced Installation Options

- Gateway API CRDs and Envoy Gateway-specific CRDs were installed with fine-grained control using helm template and kubectl apply, to avoid Helm limitations with large CRDs.

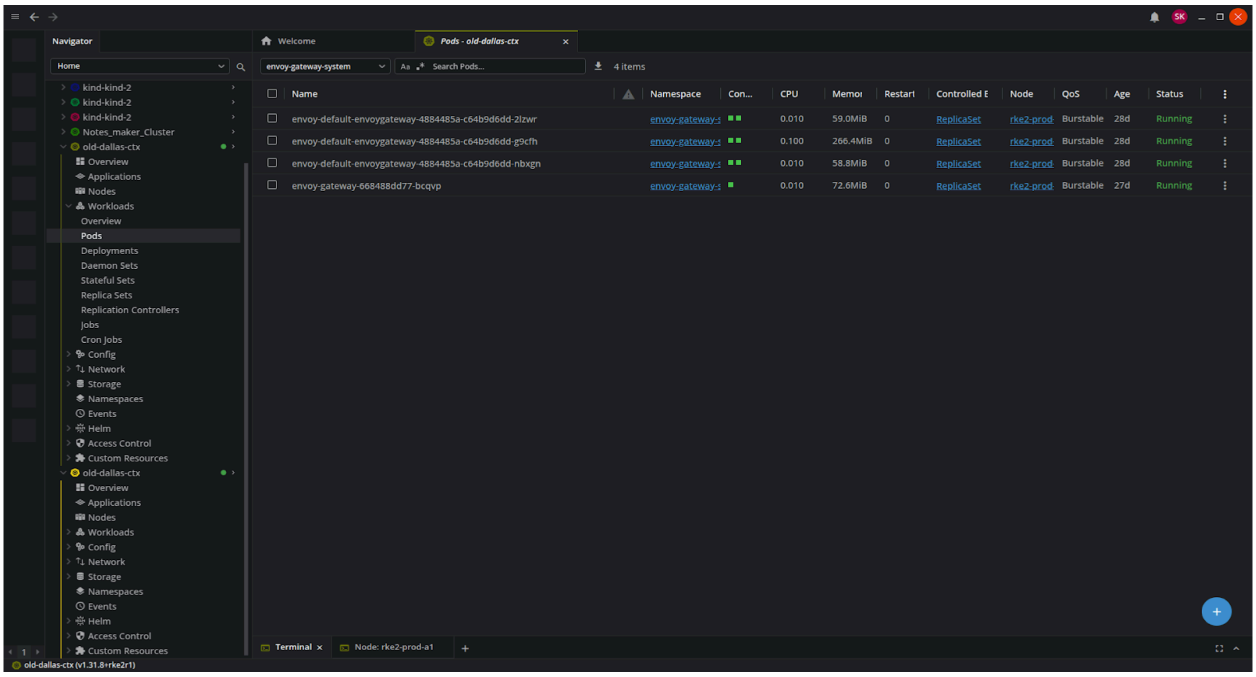
```
helm template eg oci://docker.io/envoyproxy/gateway-crds-helm \
  --version v1.4.2 \
  --set crds.gatewayAPI.enabled=true \
  --set crds.gatewayAPI.channel=standard \
  --set crds.envoyGateway.enabled=true | kubectl apply --server-side -f -
```

iv) Helm Customization and Resource Configuration

To optimize the deployment, a custom values.yaml file was used to:

- Increase CPU and memory limits for better performance

- Enable gRPC and rate-limiting ports for future extensibility
- Set logging level to debug for troubleshooting.



Name	Namespace	Con...	CPU	Memoı	Restart	Controlled E	Node	QoS	Age	Status
envoy-default-envoygateway-4884485a-c64b9d6dd-2lzwr	envoy-gateway-s	🟩🟩	0.010	59.0MiB	0	ReplicaSet	rke2-prod-	Burstable	28d	Running
envoy-default-envoygateway-4884485a-c64b9d6dd-g9cfh	envoy-gateway-s	🟩🟩	0.100	266.4MiB	0	ReplicaSet	rke2-prod-	Burstable	28d	Running
envoy-default-envoygateway-4884485a-c64b9d6dd-nbxgn	envoy-gateway-s	🟩🟩	0.010	58.8MiB	0	ReplicaSet	rke2-prod-	Burstable	28d	Running
envoy-gateway-668488dd77-bcqvp	envoy-gateway-s	🟩	0.010	72.6MiB	0	ReplicaSet	rke2-prod-	Burstable	27d	Running

Figure 3.21: Lens Dashboard showing envoy gateway deployment

Chapter 4: Conclusion and Learning Outcomes

4.1. Conclusion

In conclusion, the internship at **Smart Ideas Pvt.Ltd (Hamro Patro)** as a DevOps Intern exposed me to various aspects of DevOps. The internship has been a valuable and enriching experience, allowing me to apply the theoretical knowledge gained throughout my academic journey into a real-world software development environment. This included knowledge gained in diverse DevOps areas like deployment of websites with SSL/TLS certificate, Virtual Machine networking, containerization and utilizing Kubernetes (K3s, RKE2) for container orchestration, implementation of monitoring tools such as Prometheus and Grafana.

Practical tasks were done using theoretical knowledge acquired from the program. At the end of it all, adopting different DevOps methodologies that were learnt brought about better system performance and reliability. With the help of mentors and other colleagues, my interpersonal skills were also sharpened during this internship period henceforth having a huge positive influence on me both professionally and personally.

4.2. Learning Outcome

Here are the key areas where I gained substantial knowledge and practical experience:

1. **Technical Skills**

   Developed my technical skills like operating system troubleshooting, managing bare metal infrastructure, automating repeated tasks with bash script. Got the opportunity to dig deep into using linux command line as well as monitoring tools like monit, prometheus, and grafana. Learned about monitoring various processes of linux and sending email alerts if needed. Along with this various other tools like ansible, used for DevOps were learned.

2. **Professional Development**

   Enhanced the ability to work effectively in a team oriented environment. Problem solving skills were improved by addressing realworld technical challenges and implementing practical solutions. Professional growth was further shaped by the mentorship and guidance received from experienced professionals.

3. **Time Management**

   Effective time management was crucial during the internship, as multiple tasks were assigned simultaneously. Work was prioritized, achievable goals were set, and deadlines were consistently met.

4. **Documentation**

   The importance of thorough documentation was emphasized throughout the internship. Detailed records of configurations, processes, and performance metrics were maintained, ensuring transparency and reproducibility. Comprehensive reports and documentation were compiled to communicate project progress, improving technical writing and communication skills.

5. **Continuous Learning**

   A habit of continuous learning was cultivated during the internship, encouraging the intern to stay updated with the latest industry trends and advancements. Self Directed learning was regularly engaged in, exploring new tools and technologies to enhance existing systems and processes

References

Ebert, Christof, Gallardo, Gorka, Hernantes, et al. (2016). DevOps. *IEEE Software*,33(3), 94–100. https://doi.org/10.1109/MS.2016.68

Hamro Patro. (2010). *About Hamro Patro.* https://www.hamropatro.com/about

Farley, J., David;Humble. (2015). *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation) (Tenth printing, Vol. 0).* AddisonWesley Professional.

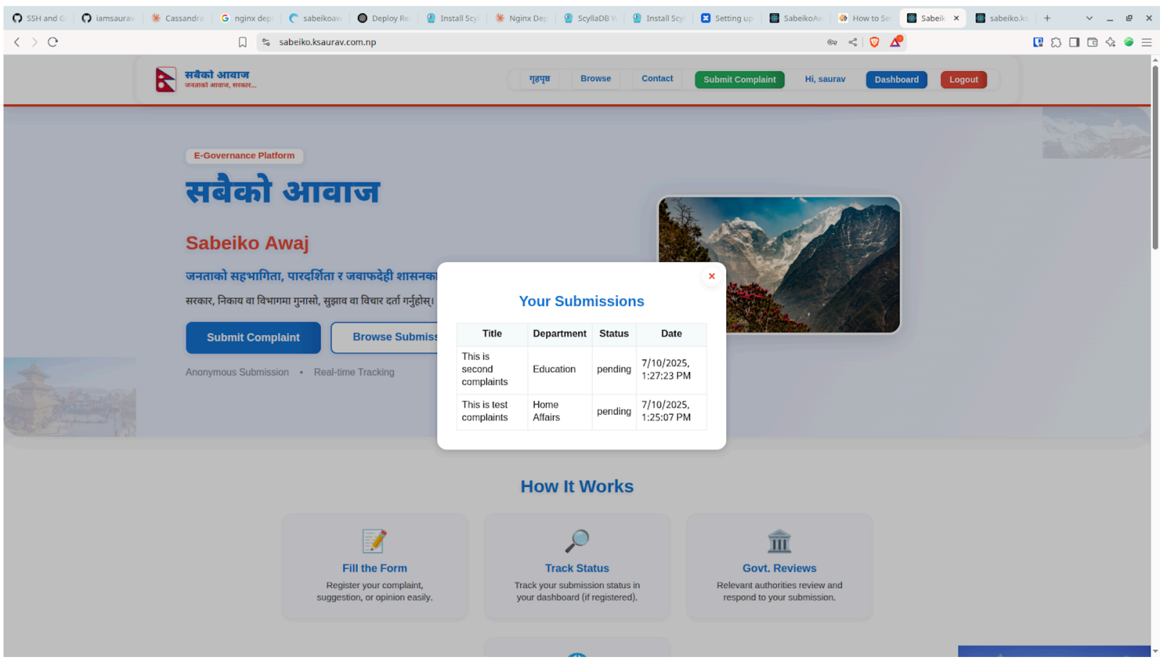
KALEN WESSE, D. &. S. (2018). *A Day in the Life of a DevOps Engineer.*

Len Bass,L. Z.,Ingo Weber. (2015). DevOps: *A Software Architect's Perspective* (1st ed., Vol. 0). AddisonWesley Professional.

N, S. (2020). Automation of Software Development using DevOps and its Benefits. *International Journal of Engineering Research and,* . https://doi.org/10.17577/IJERTV9IS060369

Senapathi, M., Buchan, J., & Osman, H. (2019). DevOps Capabilities, Practices, and Challenges: Insights from a Case Study. *Corr.* http://arxiv.org/abs/1907.10201

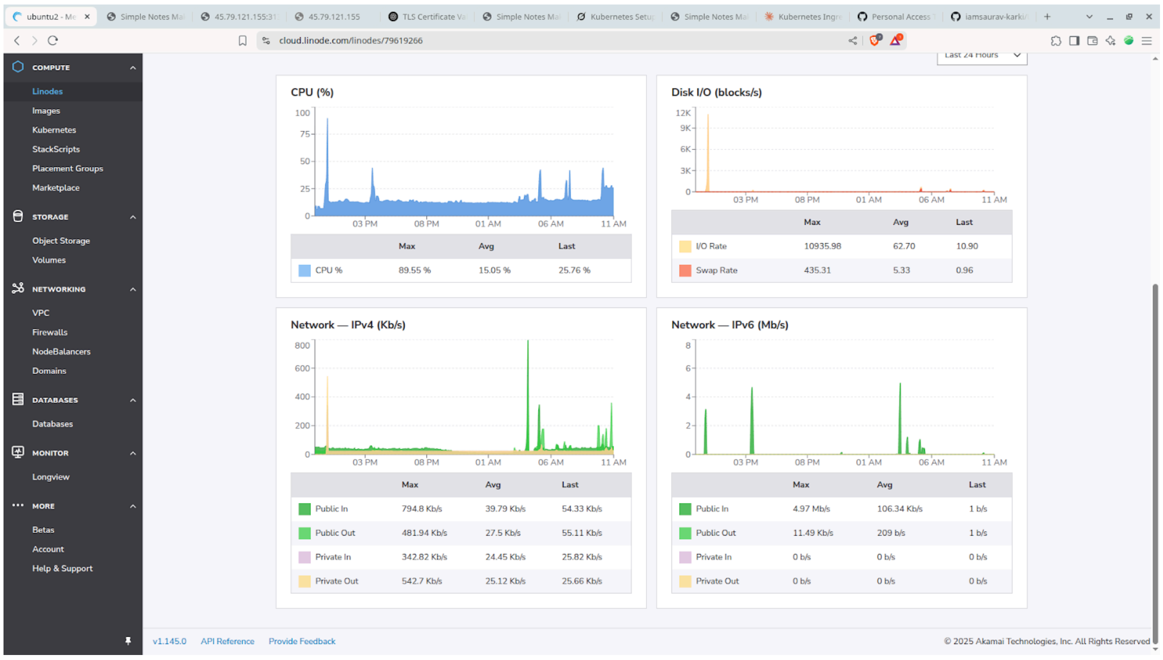
Appendices


sabeiko.ksaurav.com.np
सबैको आवाज
गृहपृ
Browse
Contact
Submit Complaint
Hi, saurav
Dashboard
Logout
E-Governance Platform
सबैको आवाज
Sabeiko Awaj
जनताको सहभागिता, पारदर्शिता र जवाफदेही शासनका
सरकार, निकाय वा विभागमा गुनासो, सुझाव वा विचार दर्ता गर्नुहोस्।
Submit Complaint
Browse Submiss
Anonymous Submission • Real-time Tracking
How It Works
Fill the Form
Register your complaint, suggestion, or opinion easily.
Track Status
Track your submission status in your dashboard (if registered).
Govt. Reviews
Relevant authorities review and respond to your submission.

Your Submissions

Title	Department	Status	Date
This is second complaints	Education	pending	7/10/2025, 1:27:23 PM
This is test complaints	Home Affairs	pending	7/10/2025, 1:25:07 PM

Deployed website with nginx and custom domain setup



CPU (%)

	Max	Avg	Last
CPU %	89.55 %	15.05 %	25.76 %

Disk I/O (blocks/s)

	Max	Avg	Last
I/O Rate	10935.98	62.70	10.90
Swap Rate	435.31	5.33	0.96

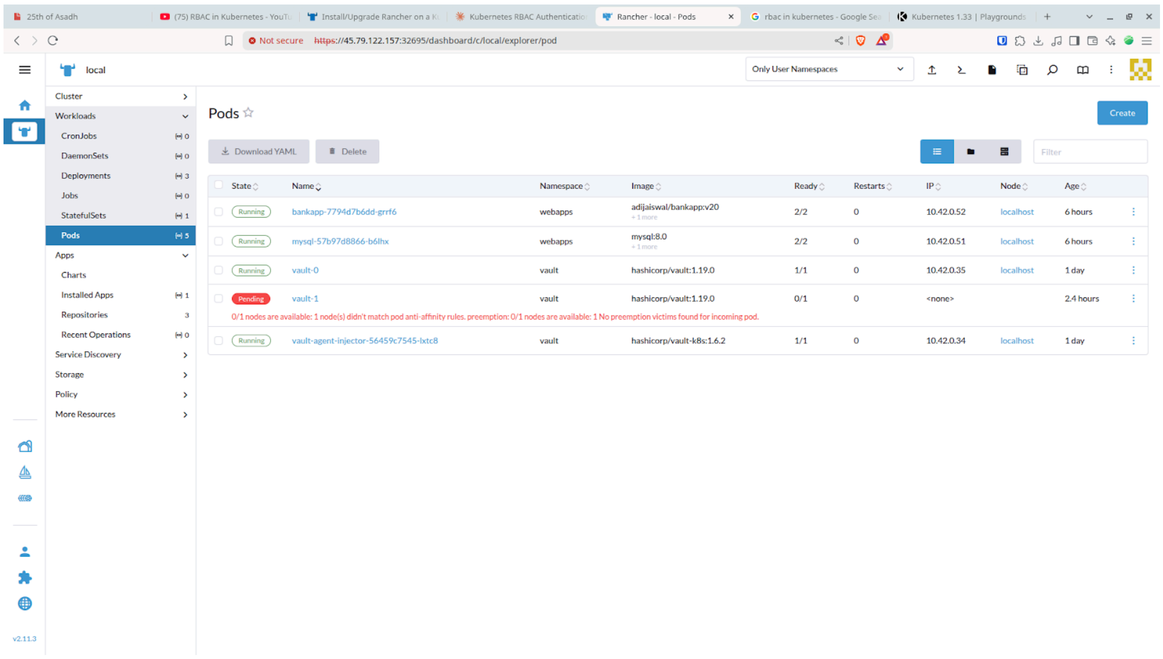
Network — IPv4 (Kb/s)

	Max	Avg	Last
Public In	794.8 Kb/s	39.79 Kb/s	54.33 Kb/s
Public Out	481.94 Kb/s	27.5 Kb/s	55.11 Kb/s
Private In	342.82 Kb/s	24.45 Kb/s	25.82 Kb/s
Private Out	542.7 Kb/s	25.12 Kb/s	25.66 Kb/s

Network — IPv6 (Mb/s)

	Max	Avg	Last
Public In	4.97 Mb/s	106.34 Kb/s	1 b/s
Public Out	11.49 Kb/s	209 b/s	1 b/s
Private In	0 b/s	0 b/s	0 b/s
Private Out	0 b/s	0 b/s	0 b/s

Linode Cloud VM Resources Usage

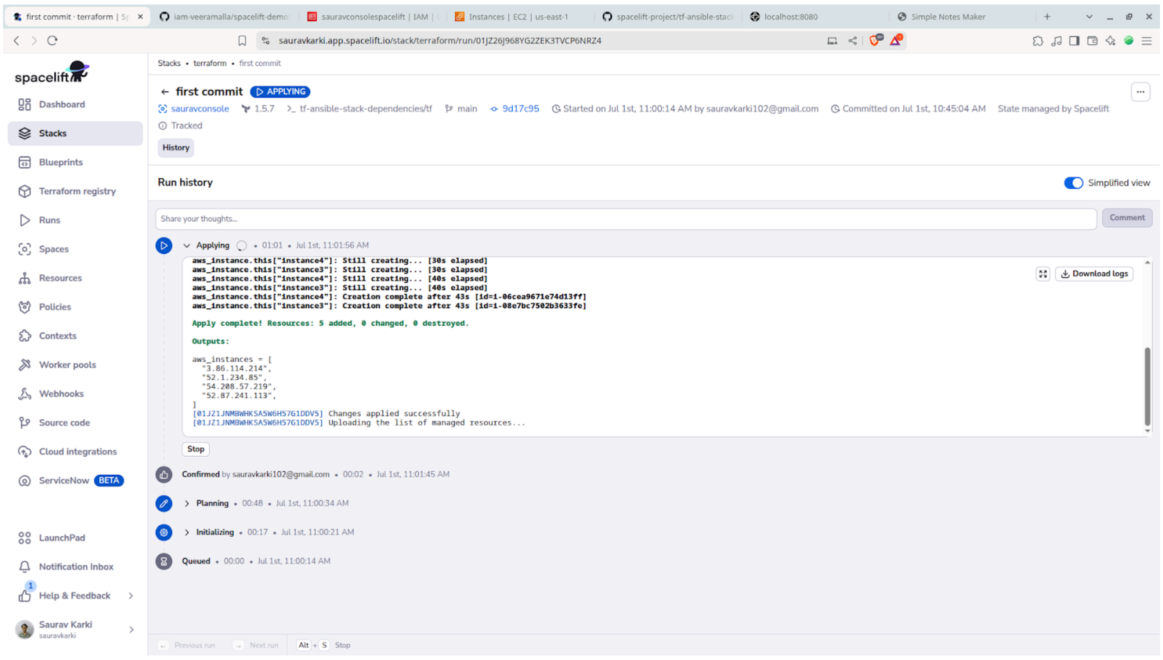


Pods

State	Name	Namespace	Image	Ready	Restarts	IP	Node	Age
Running	bankapp-7794d7b6dd-grrf6	webapps	adijaiswal/bankapp:v20 + 1 more	2/2	0	10.42.0.52	localhost	6 hours
Running	mysql-57b97d8866-b6lhx	webapps	mysql:8.0 + 1 more	2/2	0	10.42.0.51	localhost	6 hours
Running	vault-0	vault	hashicorp/vault:1.19.0	1/1	0	10.42.0.35	localhost	1 day
Pending	vault-1	vault	hashicorp/vault:1.19.0	0/1	0	<none>		2.4 hours
Running	vault-agent-injector-56459c7545-lxtc8	vault	hashicorp/vault-k8s:1.6.2	1/1	0	10.42.0.34	localhost	1 day

0/1 nodes are available: 1 node(s) didn't match pod anti-affinity rules. preemption: 0/1 nodes are available: 1 No preemption victims found for incoming pod.

Rancher UI for a Cluster Dashboard



first commit APPLYING

Run history

Applying • 01:01 • Jul 1st, 11:01:56 AM

```
aws_instance.this["instance4"]: Still creating... [30s elapsed]
aws_instance.this["instance3"]: Still creating... [30s elapsed]
aws_instance.this["instance4"]: Still creating... [40s elapsed]
aws_instance.this["instance3"]: Still creating... [40s elapsed]
aws_instance.this["instance4"]: Creation complete after 43s [id=i-06cea9671e74d13ff]
aws_instance.this["instance3"]: Creation complete after 43s [id=i-08e7bc7502b3633fe]

Apply complete! Resources: 5 added, 0 changed, 0 destroyed.

Outputs:

aws_instances = [
  "3.86.114.214",
  "52.1.234.85",
  "54.208.57.219",
  "52.87.241.113",
]
[01JZ1JNMBWHKSA5W6H57G1DDV5] Changes applied successfully
[01JZ1JNMBWHKSA5W6H57G1DDV5] Uploading the list of managed resources...
```

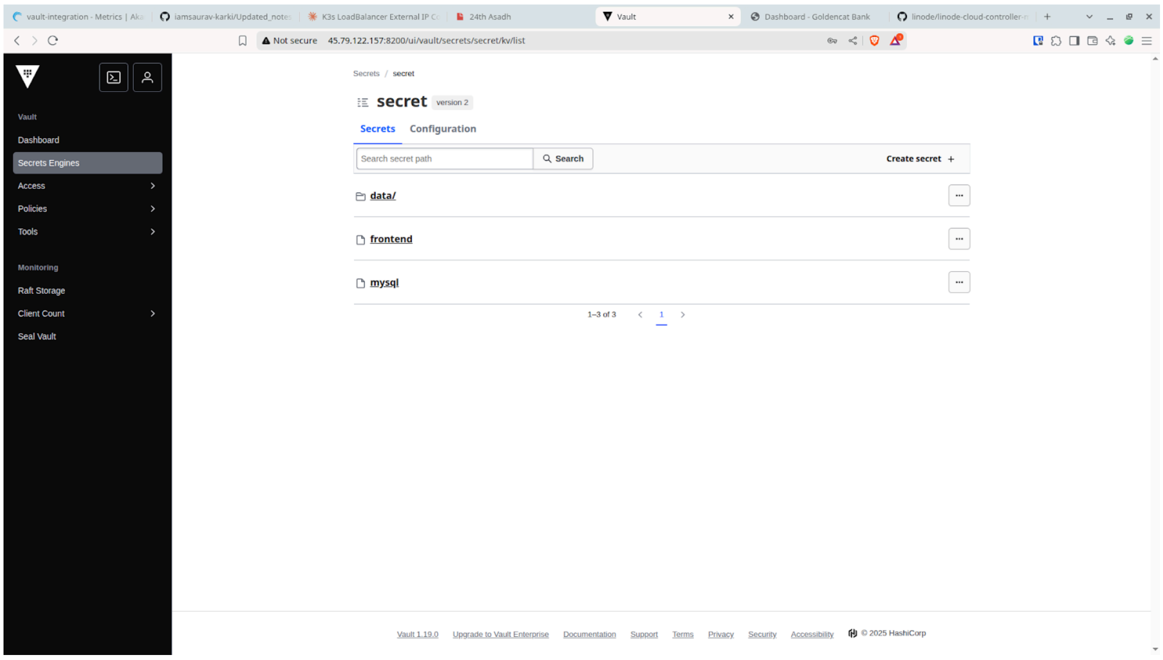
Confirmed by sauravkarki102@gmail.com • 00:02 • Jul 1st, 11:01:45 AM

Planning • 00:48 • Jul 1st, 11:00:34 AM

Initializing • 00:17 • Jul 1st, 11:00:21 AM

Queued • 00:00 • Jul 1st, 11:00:14 AM

Spacelift Dashboard For Infrastructure Creation and Configuration in AWS


Secrets / secret

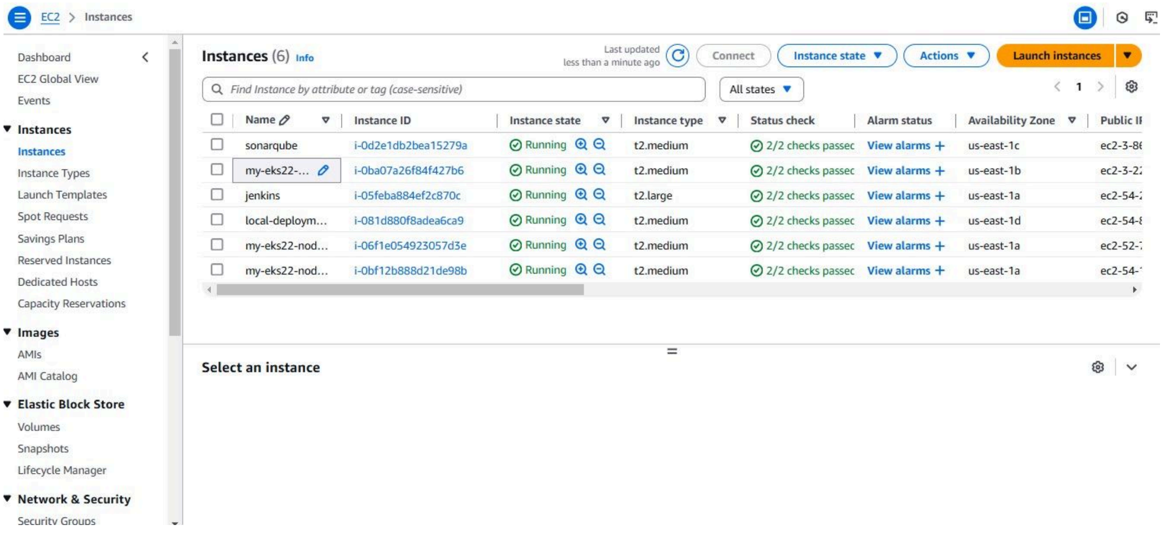
secret version 2

Secrets Configuration

data/
frontend
mysql

1–3 of 3

Hashicorp Vault UI showing stored secrets


EC2 > Instances

Instances (6) Info

Last updated less than a minute ago

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public I
sonarqube	i-0d2e1db2bea15279a	Running	t2.medium	2/2 checks passec	View alarms +	us-east-1c	ec2-3-8
my-eks22-…	i-0ba07a26f84f427b6	Running	t2.medium	2/2 checks passec	View alarms +	us-east-1b	ec2-3-2
jenkins	i-05feba884ef2c870c	Running	t2.large	2/2 checks passec	View alarms +	us-east-1a	ec2-54-
local-deploym…	i-081d880f8adea6ca9	Running	t2.medium	2/2 checks passec	View alarms +	us-east-1d	ec2-54-
my-eks22-nod…	i-06f1e054923057d3e	Running	t2.medium	2/2 checks passec	View alarms +	us-east-1a	ec2-52-
my-eks22-nod…	i-0bf12b888d21de98b	Running	t2.medium	2/2 checks passec	View alarms +	us-east-1a	ec2-54-

Select an instance

AWS Console showing different EC2 server for different services