Republic of Tunisia
Ministry of Higher Education
and Scientific Research

University of Sfax

FACULTY OF SCIENCES OF SFAX

Engineer in :
Computer Science

Option :
Data Engineering

End of The Year Project

Presented at

FACULTY OF SCIENCES OF SFAX
Department of Computer Science and Communicaations

in order to obtain the

National Engineering Diploma in :
Computer Science

Option:
Data Engineering

by

Ibrahim GHALI, Hamza ZARAI

Deployment of JupyterHub on Kubernetes with task management via a queue and resource optimization (CPU/GPU)

Presented on: 30/05/2025, before the jury of commission:

Mrs. Amina AMARA	**President**
Mrs. Imen KETATA	**Supervisor**

TABLE OF CONTENTS

LIST OF FIGURES

Acknowledgments

We would like to express our sincere gratitude to **Mrs. Imen KETATA** for her invaluable support, guidance, and encouragement throughout the course of this project.

Her expertise, availability, and insightful advice were instrumental in helping us overcome challenges and achieve our objectives.

Her dedication and commitment greatly contributed to the successful completion of this work, and we are truly thankful for her continuous accompaniment and assistance at every stage of the project.

Chapter 1

General Introduction

1.1 Context

The growing adoption of data science and machine learning at institutions like the **Data Engineering and Semantics Research Unit at Faculty of Sciences of Sfax (FSS)** necessitates scalable and flexible computing environments. Students and researchers in data engineering often require powerful resources (CPUs, GPUs) and isolated environments to ensure reproducibility and efficient execution of workflows. Traditional infrastructures often lead to underutilization and bottlenecks. Open-source technologies such as **Kubernetes** offer dynamic and resilient platforms. Integrated with **JupyterHub**, they enable on-demand containerized environments-though efficient resource and task management remains a challenge.

1.2 Problem Statement

Managing Jupyter-based interactive environments at scale poses several challenges:

- **Resource Allocation:** Fair distribution of CPU, memory, and GPU among users is complex and can lead to performance degradation if unmanaged.
- **Heavy Workloads:** Intensive notebook tasks can monopolize resources and render the environment unresponsive.

  A solution is needed that combines user-friendly access with robust backend management and scalability.

1.3 Objective

This project aims to implement **JupyterHub on a high-availability Kubernetes cluster** to provide a **scalable, secure, and resource-optimized environment** for data science and engineering tasks. The platform integrates a **task queuing system** and persistent storage to support multi-user workflows reliably. The key objectives are:

1. Set up a highly available Kubernetes cluster to host containerized Jupyter environments.
2. Configure **HAProxy** for load balancing JupyterHub services.
3. Deploy JupyterHub with support for multiple user sessions and secure authentication.
4. Ensure persistence of user data (e.g., notebooks ...) using **NFS-based shared storage**.
5. Integrate **Apache YuniKorn** for efficient scheduling and resource management.

Chapter 2

System Architecture

This chapter details the overall architecture of the implemented system, showcasing how JupyterHub interacts with Kubernetes and the task queuing mechanism for data engineering tasks.

2.1System Architecture

The deployed architecture follows a **highly available Kubernetes cluster** design with the following components:

2.1.1 Load Balancing Layer

- **HAProxy** serves as the entry point, distributing traffic across Kubernetes master nodes
- Provides high availability for API server access
- Eliminates single point of failure for control plane access

2.1.2 Kubernetes Cluster

2.1.2.1 Control Plane

- **3 Master Nodes** in HA configuration

2.1.2.2 Worker Nodes

- **2 Worker Nodes** running user workloads
- Hosts multiple **JupyterLab** instances (deployed as pods)

2.1.3 Storage Layer

- **NFS Server:** Provides centralized storage.
- **Used for:**
  * Persistent volumes (e.g., Jupyter notebook storage)

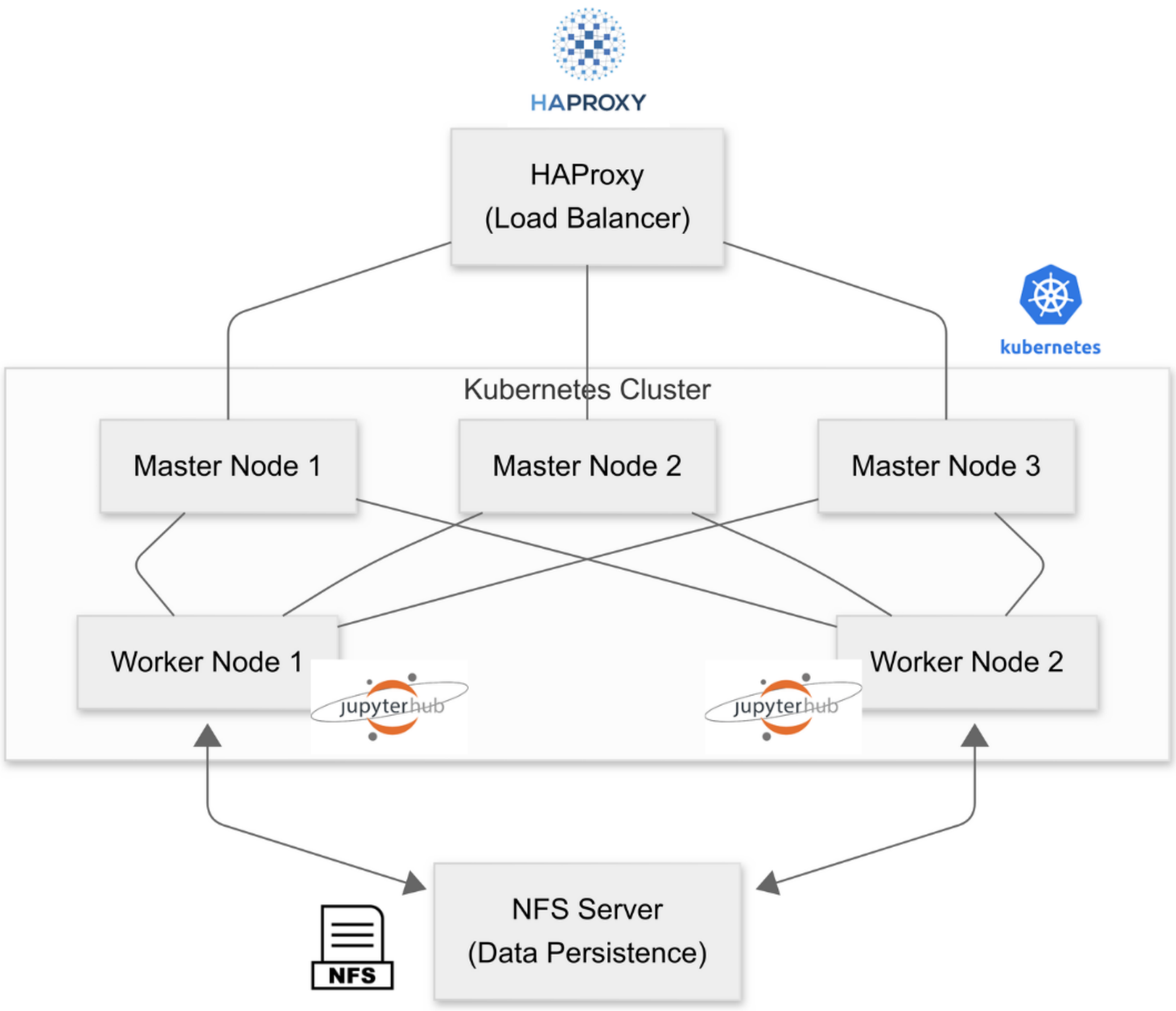


Figure 2.1: Deployment Architecture with HAProxy, Kubernetes, and NFS Data Persistence

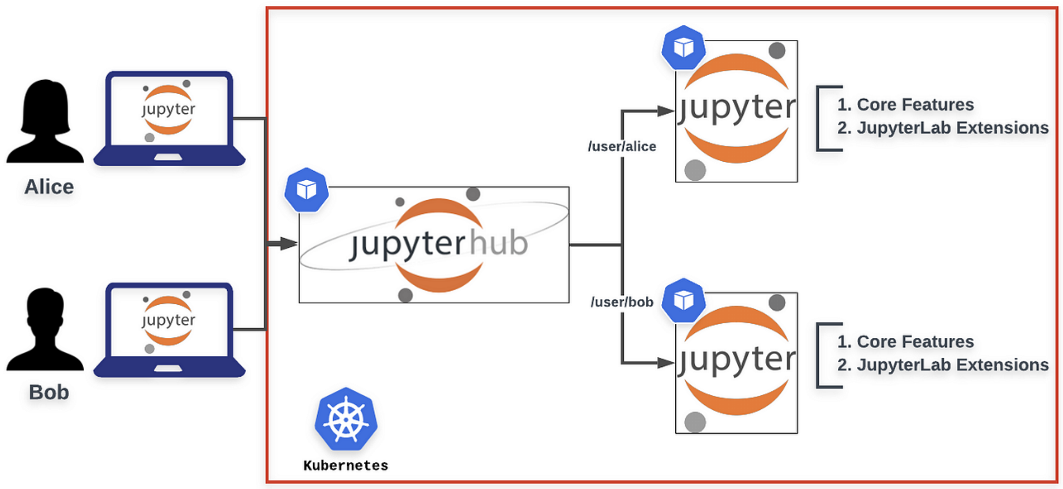


Figure 2.2: JupyterHub on Kubernetes

Chapter
3
Technologies Used

We employed modern cloud-native technologies in an on-premise environment. This chapter summarizes each key technology and its benefits.

Core Technologies

Table 3.1: Technology Overview

Technology	Purpose
Kubernetes	Container orchestration backbone
JupyterHub	Multi-user notebook environment
Helm	Kubernetes package management
Apache YuniKorn	Advanced resource scheduling
Network File System (NFS)	Persistent shared storage

3.1 Kubernetes

Definition: Kubernetes is an open-source container orchestration platform designed to automate the deployment, scaling, and management of containerized applications.

Role: Kubernetes serves as the infrastructure backbone by efficiently managing container lifecycle, resource allocation, service discovery, and fault tolerance.

3.2 JupyterHub

Definition: JupyterHub is a multi-user server for Jupyter notebooks, allowing users to access individual Jupyter environments on shared infrastructure.

Role: JupyterHub enables interactive, reproducible computing environments for research, education, and data science.

3.3 Helm

Definition: Helm is a package manager for Kubernetes that uses charts to define, install, and upgrade complex applications on clusters.

Role: Helm simplify the deployment and lifecycle management of Kubernetes applications by providing reusable and customizable configuration templates.

3.4 Apache YuniKorn

Definition: Apache YuniKorn is resource scheduler for Kubernetes to manage resources in large-scale, multi-tenant clusters.

Role: Apache YuniKorn supports hierarchical queues and fairness across tenants.
This makes it ideal for AI/ML workloads where managing compute quotas and job priorities is critical for performance .

3.5 Network File System (NFS)

Definition: Network File System (NFS) allows a system to share directories and files with others over a network.
By using NFS, users and programs can access files on remote systems.

Role: In Kubernetes, NFS provides persistent storage for user data.
It's commonly used with JupyterHub to ensure user notebooks and datasets accessibility across sessions and pods.

3.6 Screenshots

This section will contain relevant screenshots illustrating the key aspects of the system in operation, such as the JupyterHub interface, Kubernetes resource views, and task queue monitoring.

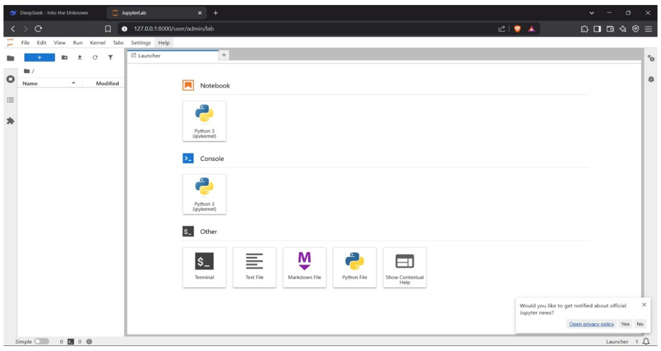
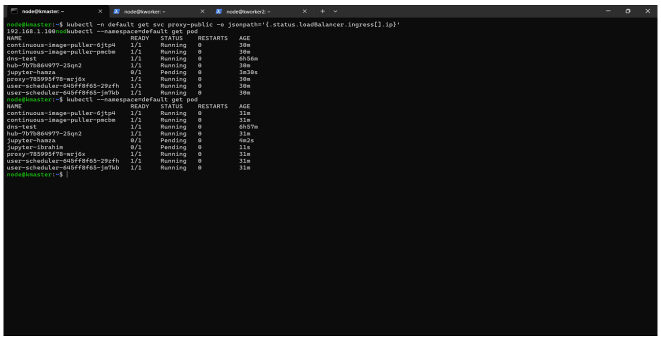
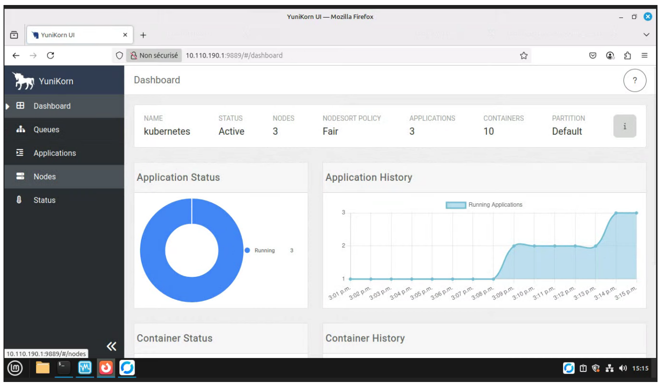


Figure 3.1: JupyterHub Login Interface



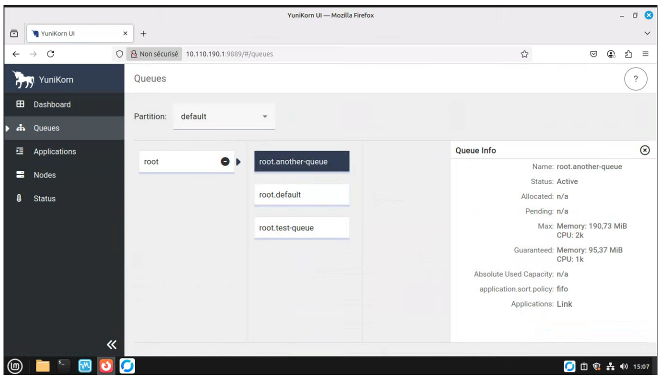
```
node@kmaster:~$ kubectl -n default get svc proxy-public -o jsonpath='{.status.loadBalancer.ingress[].ip}'
192.168.1.100nodkubectl --namespace=default get pod
NAME                                READY   STATUS    RESTARTS   AGE
continuous-image-puller-6jtp4       1/1     Running   0          30m
continuous-image-puller-pmcbm       1/1     Running   0          30m
dns-test                            1/1     Running   0          6h56m
hub-7b7b864977-25qn2                1/1     Running   0          30m
jupyter-hamza                       0/1     Pending   0          3m30s
proxy-785995f78-wrj6x               1/1     Running   0          30m
user-scheduler-645ff8f65-29zfh      1/1     Running   0          30m
user-scheduler-645ff8f65-jm7kb      1/1     Running   0          30m
node@kmaster:~$ kubectl --namespace=default get pod
NAME                                READY   STATUS    RESTARTS   AGE
continuous-image-puller-6jtp4       1/1     Running   0          31m
continuous-image-puller-pmcbm       1/1     Running   0          31m
dns-test                            1/1     Running   0          6h57m
hub-7b7b864977-25qn2                1/1     Running   0          31m
jupyter-hamza                       0/1     Pending   0          4m2s
jupyter-ibrahim                     0/1     Pending   0          11s
proxy-785995f78-wrj6x               1/1     Running   0          31m
user-scheduler-645ff8f65-29zfh      1/1     Running   0          31m
user-scheduler-645ff8f65-jm7kb      1/1     Running   0          31m
node@kmaster:~$ |
```

Figure 3.2: JupyterHub Pods



NAME	STATUS	NODES	NODESORT POLICY	APPLICATIONS	CONTAINERS	PARTITION
kubernetes	Active	3	Fair	3	10	Default

Figure 3.3: Yunikorn Dashboard



Queue Info
Name: root.another-queue
Status: Active
Allocated: n/a
Pending: n/a
Max: Memory: 190,73 MiB CPU: 2k
Guaranteed: Memory: 95,37 MiB CPU: 1k
Absolute Used Capacity: n/a
application.sort.policy: fifo
Applications: Link

Figure 3.4: Yunikorn Queues

Chapter 4

General Conclusion

This project delivered a platform to support data science activities at the Faculty of Sciences of Sfax. By implementing JupyterHub on a Kubernetes cluster with task queuing and resource management via Apache Yunikorn, we established an environment that leverages resources and ensures data persistence via NFS storage.
The solution supports multiple users. Its key value lies in maintaining system responsiveness under intensive workloads

Chapter 5

Future Work

Future enhancements could incorporate advanced monitoring tools like Prometheus and Grafana to further optimize performance and observability.

5.1 Monitoring with Prometheus & Grafana

Integrating **Prometheus** for metrics collection (cluster resources, JupyterHub performance, and task queue stats) alongside **Grafana** for visualization would provide real-time and historical insights. Custom dashboards help administrators to track cluster health and resource usage.

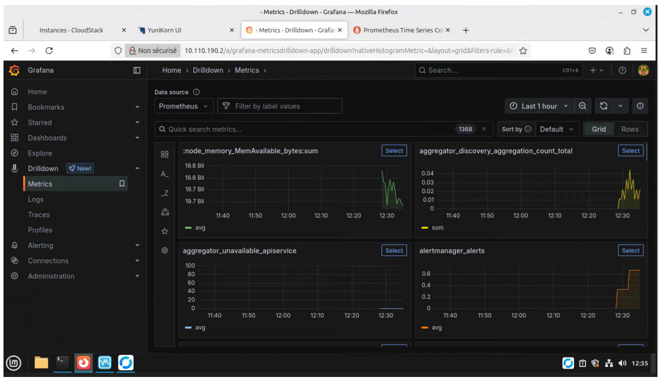

Grafana
Home › Drilldown › Metrics
Data source
Prometheus
Last 1 hour
:node_memory_MemAvailable_bytes:sum
aggregator_discovery_aggregation_count_total
aggregator_unavailable_apiservice
alertmanager_alerts

Figure 5.1: Grafana Interface

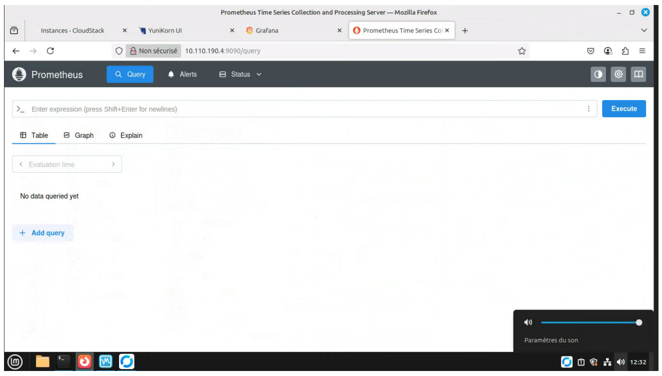


Figure 5.2: Prometheus Interface

Chapter 6

Bibliography

- Kubernetes Authors. Kubernetes Documentation. Available: https://kubernetes.io/docs/
- Project Jupyter. JupyterHub Documentation. Available: https://jupyterhub.readthedocs.io/
- Project Jupyter. Zero to JupyterHub with Kubernetes. Available: https://zero-to-jupyterhub.readthedocs.io/
- Apache YuniKorn. YuniKorn Documentation.
  Available: https://yunikorn.apache.org/docs/
- Prometheus Authors. Prometheus Documentation.
  Available: https://prometheus.io/docs/introduction/overview/
- Grafana Labs. Grafana Documentation.
  Available: https://grafana.com/docs/
- Prometheus Authors. Alertmanager Documentation.
  Available: https://prometheus.io/docs/alerting/latest/alertmanager/
- R. Sandberg, D. Goldberg, D. Walsh, B. Lyon. NFS: Network File System Protocol Specification. RFC 1094. March 1989. Available: https://datatracker.ietf.org/doc/html/rfc1094
- Kubernetes Authors. Persistent Volumes - NFS. Kubernetes Documentation. Available: https://kubernetes.io/docs/concepts/storage/volumes/#nfs