# **Project Name:** Cloud Odyssey
# **Requirements and Specification Document**
# **Date:** 28/03/2025
# **Version:** 1.0

---


| | |
|---|---|
| 21116034 | Vaishnavi Virat Dave |
| 21116048 | Kaashvi Jain |
| 23114043 | Kajal |
| 23114052 | Kumud |
| 21323003 | Abhijna Raghavendra |
| 23114008 | Anushka Jangid |


---

## 1. Project Abstract


Cloud Odyssey is a sophisticated centralized Beowulf cluster designed to harness idle computational resources, transforming them into a distributed cloud computing system. This infrastructure enables users to securely authenticate via SSH into a Master Node, execute computationally intensive parallel workloads utilizing OpenMPI and SLURM, and dynamically orchestrate worker nodes for optimal resource allocation. By facilitating seamless collaboration among disparate computing entities across networks, Cloud Odyssey advances the paradigm of high-performance computing (HPC) and distributed systems.


## 2. Document Revision History

- **Rev. 1.0** 28/03/2025 - Initial version

## 3. Customer

The prospective beneficiaries of Cloud Odyssey encompass researchers, computational scientists, academic institutions, and enterprises that necessitate extensive parallel computing capabilities but are constrained by the prohibitive costs associated with proprietary cloud infrastructure. This can further be extended to larger cloud applications like V2V and V2X communication for autonomous cars.

## 4. Competitive Analysis

Cloud Odyssey distinguishes itself from established commercial cloud computing solutions such as AWS EC2, Google Cloud Compute Engine, and Microsoft Azure Virtual Machines. Unlike these proprietary platforms, Cloud Odyssey provides the following advantages:

- **Cost-Efficiency:** Leverages idle computational resources, mitigating capital expenditures associated with dedicated cloud services.

- **Decentralization:** Operates independently of centralized cloud providers, fostering resilience and reducing vendor lock-in.

- **Extensibility:** Implements an open-source framework, allowing for modular enhancements and domain-specific customization.

Cloud Odyssey eliminates dependency on centralized data centers, reducing susceptibility to large-scale outages. The system is designed to dynamically scale worker nodes, ensuring high availability.

**Relevant Patent Domains:**

1. **Cloud Resource Management and Scheduling** – Patents related to dynamic resource allocation, workload scheduling, and parallel computing optimization.

2. **Decentralized Cloud Computing –** Existing patents cover federated cloud models, but Cloud Odyssey's decentralized architecture based on idle compute resources provides a unique framework.

3. **Remote SSH and Secure Network Access –** Technologies related to secure remote access, NAT traversal, and encryption protocols.

## 5. System Requirements

**Functional Requirements:**

1. Users must be able to establish secure SSH connections to the Master Node.

2. The Master Node must allocate and distribute computation workloads efficiently via OpenMPI and SLURM.

3. Worker nodes should autonomously integrate and withdraw from the cluster based on real-time resource availability.

4. Computational tasks should be executed in a parallelized fashion to maximize throughput and efficiency.

5. Secure remote SSH access should be facilitated behind NAT/firewalls through LocalTunnel/ngrok.

6. The system should incorporate real-time monitoring and performance analytics via Prometheus and Grafana.

**Non-Functional Requirements:**

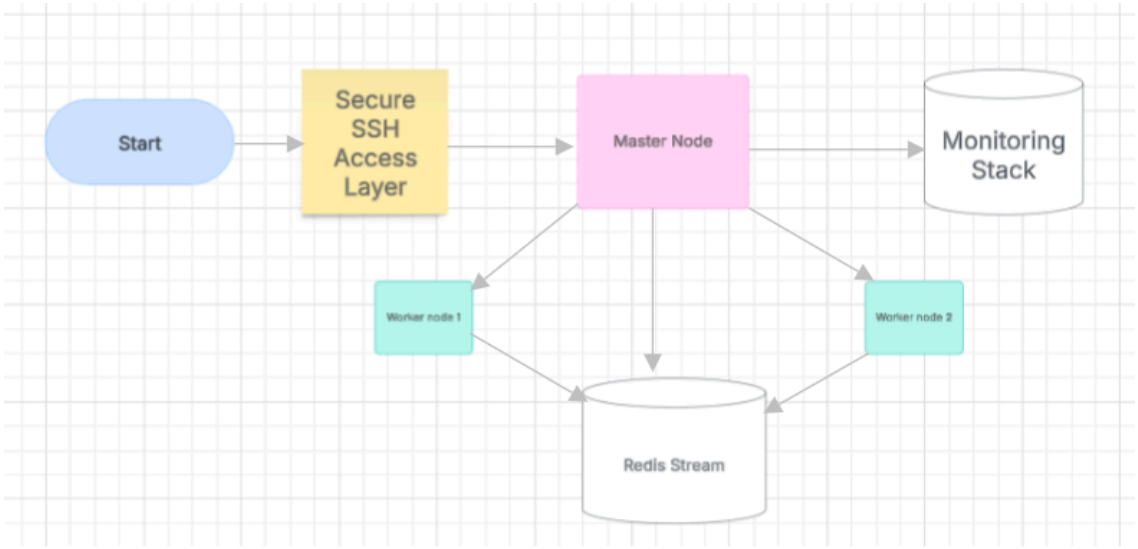
1. The system must implement robust encryption mechanisms to ensure secure communication via SSH.

2. The architecture must exhibit horizontal scalability, allowing seamless expansion of worker nodes.

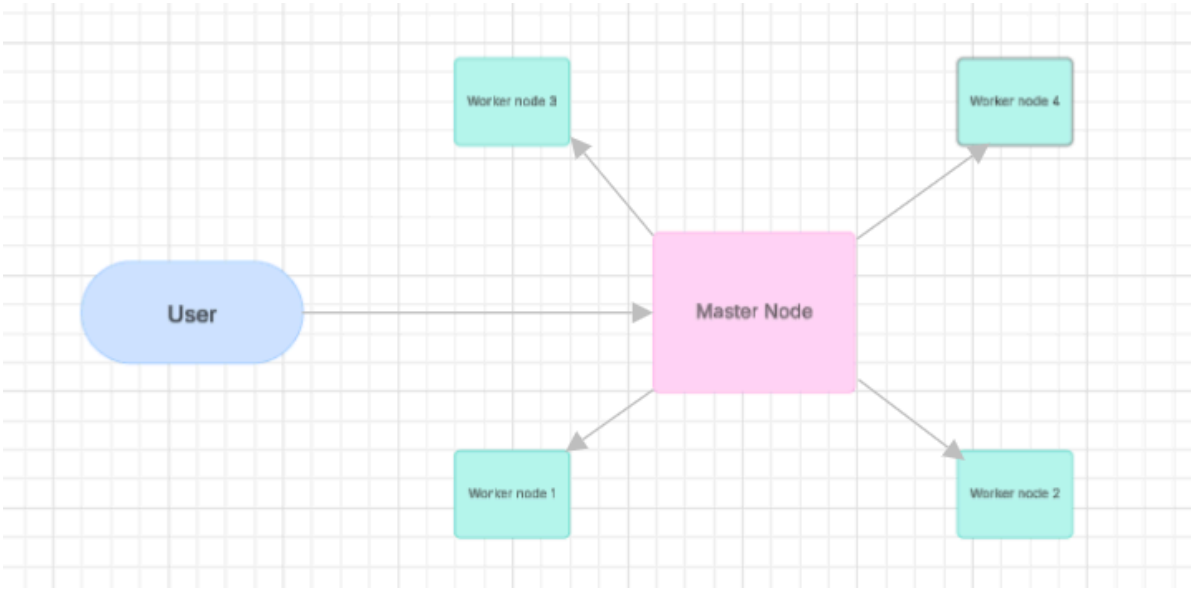
3. The system's operational footprint should be minimal to prevent excessive resource consumption on worker nodes.

4. Efficient logging and telemetry mechanisms should be incorporated to facilitate performance diagnostics and anomaly detection.

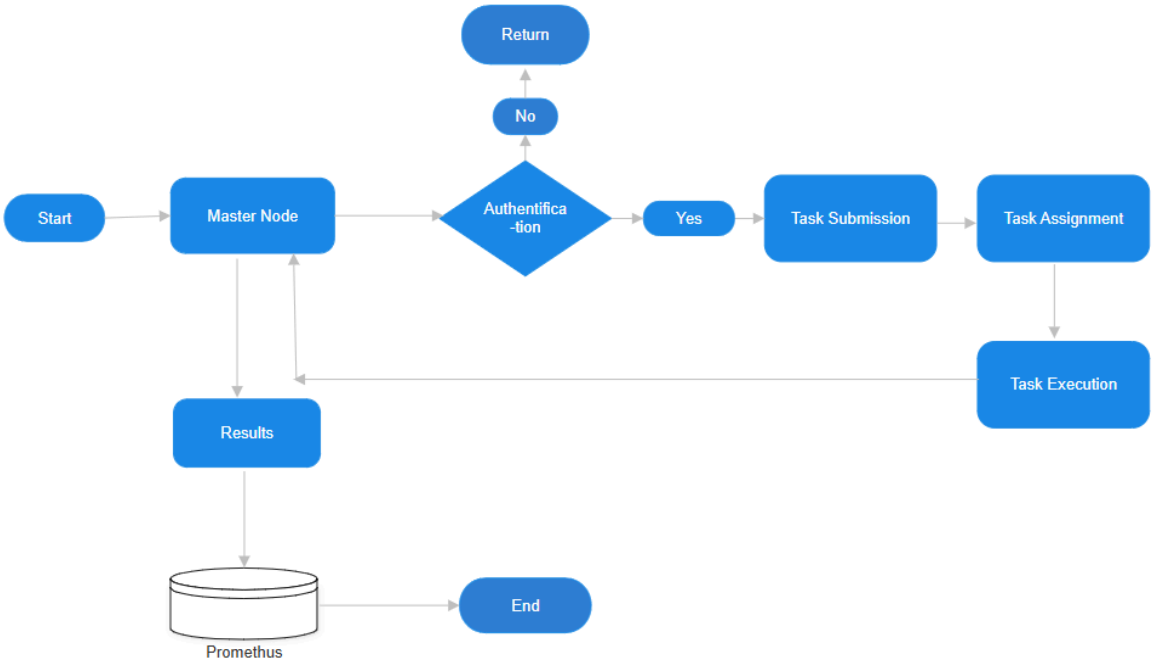
System Architecture



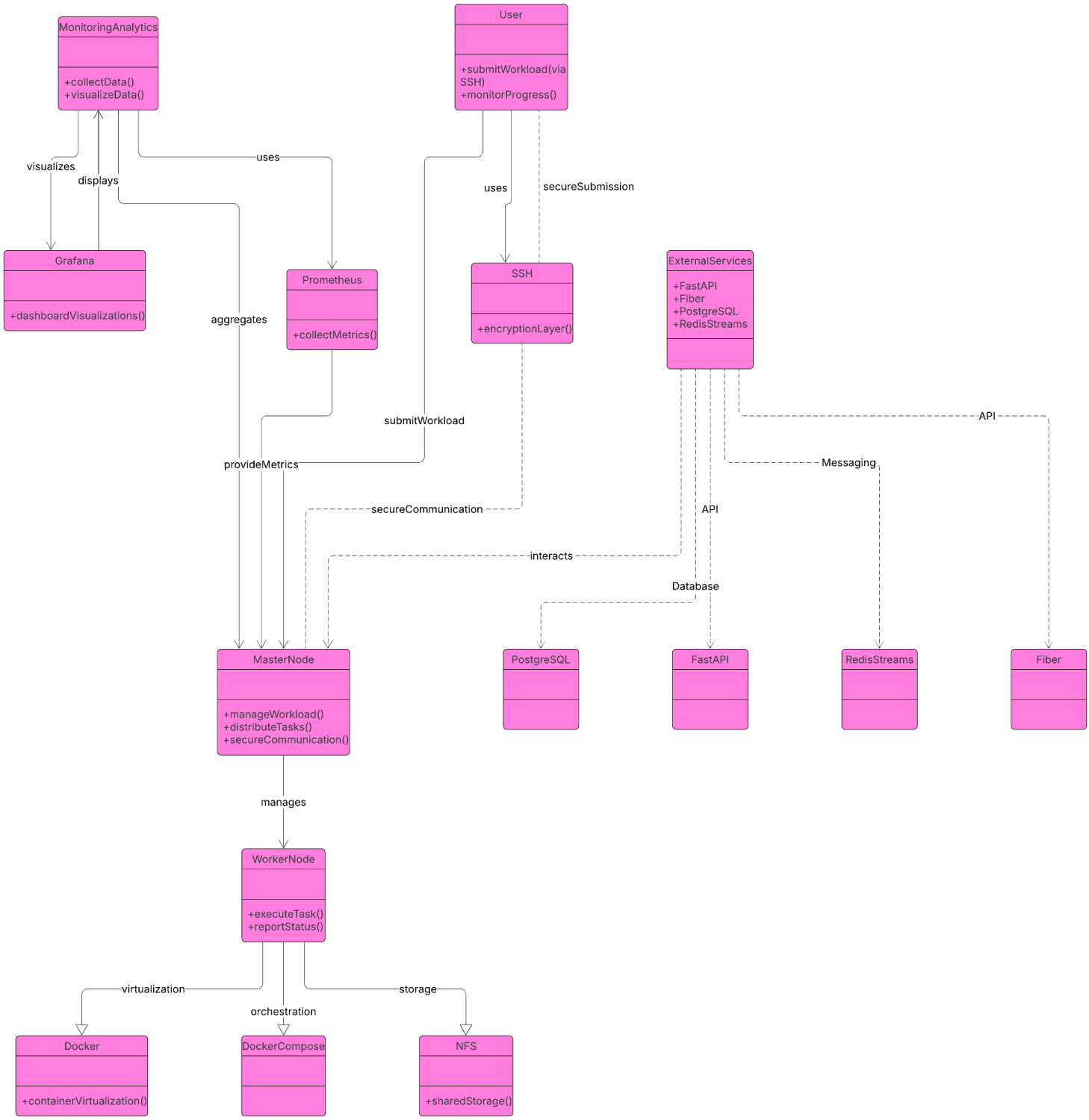
## Use Case Diagram




## Sequence diagram (Task execution flow)



# UML diagram


MonitoringAnalytics
+collectData()
+visualizeData()
User
+submitWorkload(via SSH)
+monitorProgress()
visualizes
displays
uses
uses
secureSubmission
Grafana
+dashboardVisualizations()
aggregates
Prometheus
+collectMetrics()
SSH
+encryptionLayer()
ExternalServices
+FastAPI
+Fiber
+PostgreSQL
+RedisStreams
submitWorkload
provideMetrics
API
Messaging
secureCommunication
API
interacts
Database
MasterNode
+manageWorkload()
+distributeTasks()
+secureCommunication()
PostgreSQL
FastAPI
RedisStreams
Fiber
manages
WorkerNode
+executeTask()
+reportStatus()
virtualization
orchestration
storage
Docker
+containerVirtualization()
DockerCompose
NFS
+sharedStorage()

**External Dependencies:**

- **Software:** OpenMPI, SLURM, OpenSSH, NFS (Network File System), FastAPI (for high performance HTTP based service APIs), Fiber (Golang) for low memory footprint, PostgreSQL as it supports both relational and non-relational queries, Redis Streams for dynamic data analysis, Prometheus for event monitoring, Grafana for data visualization, Docker for OS-level virtualization and Docker Compose for multi-container applications.

- **Hardware:** Multi-core processors, a minimum of 8GB RAM per node, and high-speed network connectivity.

## 6. Verification Checklist

✔ Have the requirements of all stakeholders been comprehensively analyzed? **Yes.**
✔ Are both functional and non-functional specifications rigorously articulated? **Yes.**
✔ Have security, scalability, and performance metrics been explicitly defined? **Yes.**
✔ Are all external dependencies and technological prerequisites properly documented? **Yes.**
✔ Does the system provide an advantage over conventional cloud computing solutions? **Yes.**