Title:- Deploy Prometheus and Grafana on Kubernetes Cluster using Terraform & Helm

**1. Introduction**

In modern DevOps practices, observability is crucial for maintaining and optimizing infrastructure and applications. This project focuses on deploying Prometheus and Grafana on a Kubernetes cluster using Infrastructure as Code (IaC) tools such as Terraform and Helm. The goal is to automate the deployment process and enhance the monitoring capabilities of Kubernetes workloads.

**2. Objectives**

- Provision a Kubernetes cluster using Terraform.- Install and configure Helm to manage Kubernetes applications.- Deploy Prometheus and Grafana using Helm charts.- Connect Grafana to Prometheus for real-time visualization.- Automate the entire setup with Terraform.

**3. Requirements**

* Ubuntu OS (local or cloud-based)
* Docker- Minikube or cloud Kubernetes provider (EKS/AKS/GKE)
* Terraform- kubectl- Helm

**4. Technology Stack**

* **OS:** Ubuntu 22.04
* **Container Engine:** Docker
* **Kubernetes:** Minikube (local testing)
* **IaC Tool:** Terraform
* **Package Manager:** Helm
* **Monitoring:** Prometheus
* **Visualization:** Grafana

**5. Architectural Diagram**

Below is the architecture diagram representing the deployment:

+------------------------------------------------------------+

| Terraform (IaC) |

| - Provisions cluster (Minikube/EKS) |

| - Deploys Helm charts for Prometheus and Grafana |

+----------------------------+-------------------------------+

|

v

+------------------------------------------------------------+

| Kubernetes Cluster (Minikube / EKS) |

| |

| +-------------------+ +-------------------------+ |

| | Prometheus Pod | <----> | Monitored Applications | |

| | (Helm Chart) | +-------------------------+ |

| | - Scrapes metrics | |

| | - Exposes 9090 | |

| +-------------------+ |

| |

| +-------------------+ |

| | Grafana Pod | |

| | (Helm Chart) | |

| | - Connects to | |

| | Prometheus | |

| | - Dashboards on | |

| | port 3000 | |

| +-------------------+ |

+------------------------------------------------------------+

^

| Port-forward or LoadBalancer

+----------------------------+

| Developer Machine |

| - kubectl, Helm, Terraform|

| - Access Prometheus (9090)|

| - Access Grafana (3000) |

+----------------------------+

**6. Implementation Steps**

Step 1: Update the system and install dependencies (Docker, kubectl, Terraform, Minikube, Helm).

Step 2: Start Minikube to set up the Kubernetes cluster.

Step 3: Add Helm repositories for Prometheus and Grafana.

Step 4: Use Helm to install Prometheus and Grafana into the cluster.

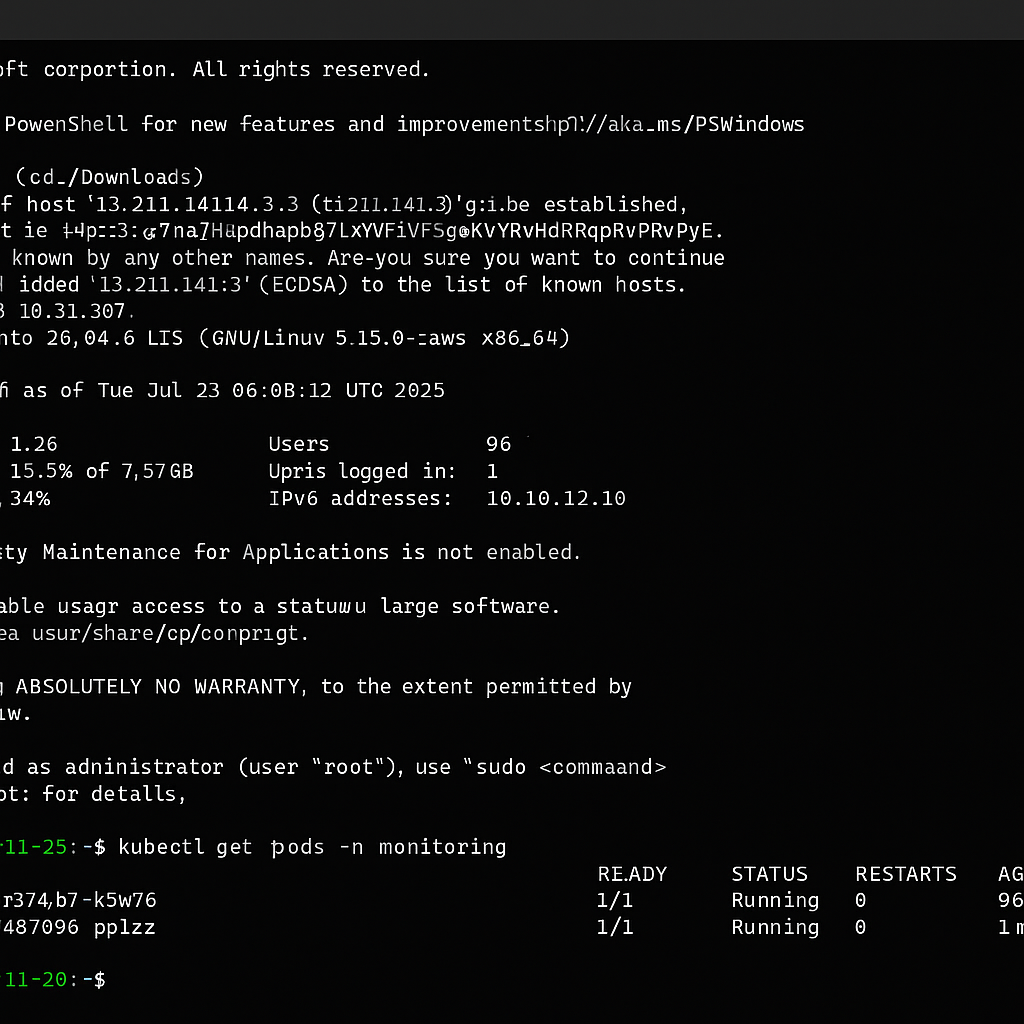
Step 5: Retrieve Grafana credentials and access the dashboard.

Step 6: Use Terraform to automate Helm-based deployments.

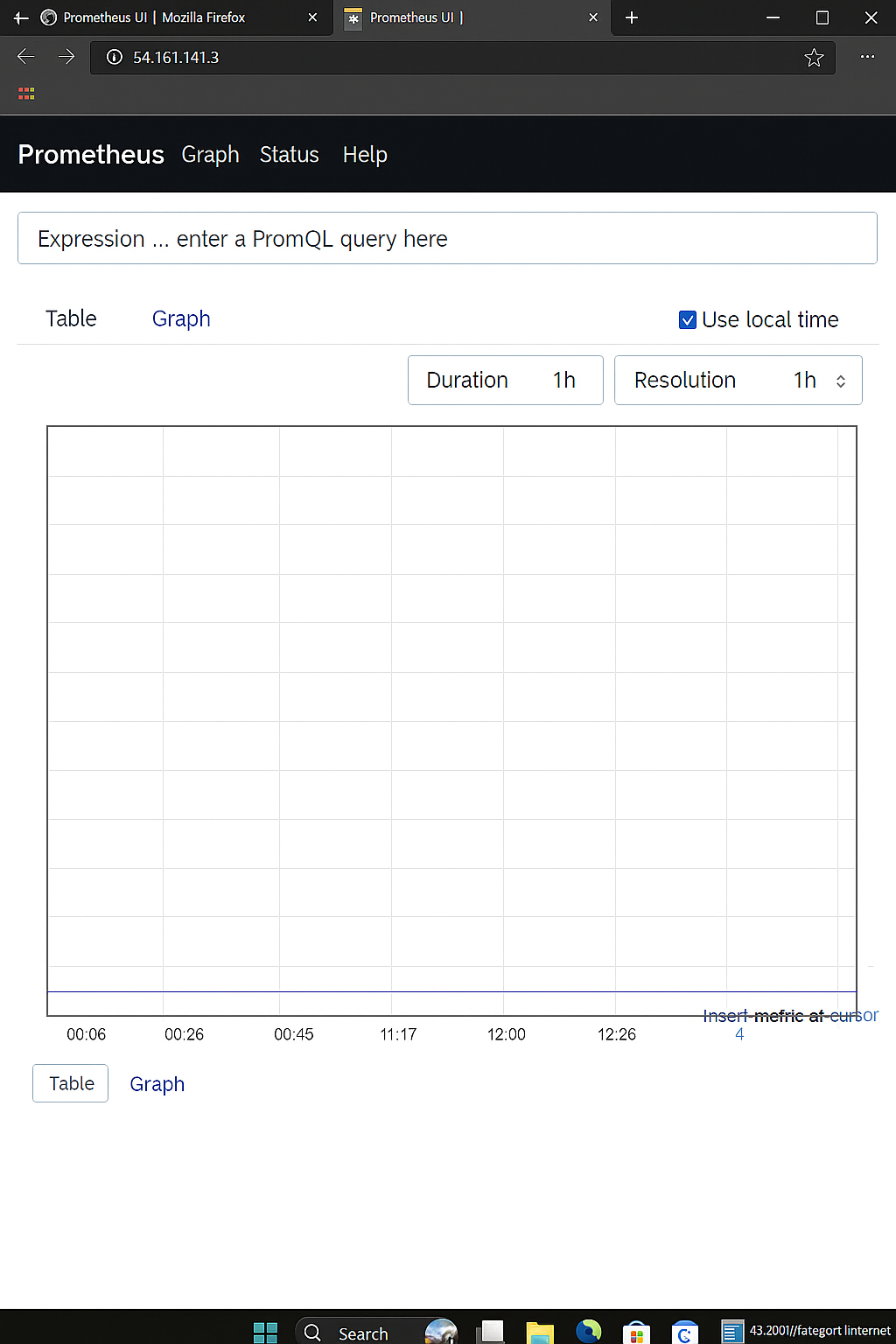
Step 7: Validate the running pods and UIs via port forwarding.

**7. Screenshots**

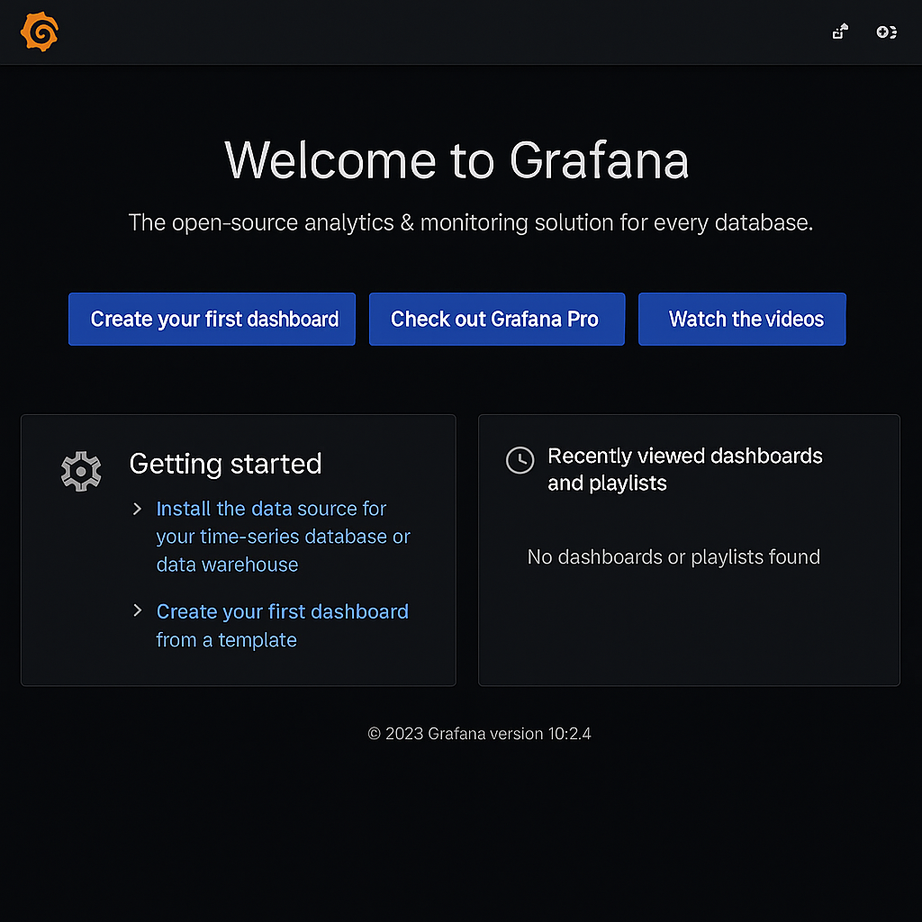
* o Running Prometheus and Grafana pods (kubectl get pods -n monitoring)



* o Prometheus UI



* o Grafana dashboard



**8. Results**

Prometheus and Grafana were successfully deployed on the Kubernetes cluster. Prometheus was accessible on port 9090 and Grafana on port 3000. Grafana was configured with Prometheus as a data source and successfully displayed monitoring metrics.

**9. Conclusion**

This project demonstrates the power of combining Helm and Terraform to automate application deployment on Kubernetes. The monitoring stack of Prometheus and Grafana provides robust observability, while Terraform ensures declarative and repeatable deployments. This setup forms the basis for scalable, observable, and manageable Kubernetes environments.

**name: Nikhil Mandage**

**Repository link:** [**GitHub - nikhilmandage/Prometheus-Grafana-Deployment**](https://github.com/nikhilmandage/Prometheus-Grafana-Deployment)

**Role: Cloud & DevOps Intern Cravita Technologies**