

**SYMBIOSIS INTERNATIONAL (DEEMED UNIVERSITY)**

(Established under section 3 of the UGC Act 1956) **Re-accredited by NAAC with ‘A++’ Grade**

**Founder: Prof. Dr. S. B. Mujumdar, M.Sc.,Ph.D. (Awarded Padma Bhushan and Padma Shri by the President of India)**

**DevOps Project Report**

**ON**

**“Automating Deployment and Monitoring of Instagram Reach Prediction using DevOps”**

## Submitted By:

|  |  |
| --- | --- |
| Ansh Sehgal (th1) | 22070122021 |
| Aryan Bhoi (th1) | 22070122033 |
| Shubham Upadhyay (th2) | 22070122209 |
| Shrishti Bajpai (th2) | 22070122205 |

**1. Introduction**

In the modern software industry, DevOps has become an essential practice to bridge the gap between development and operations. It ensures faster delivery, scalability, automation, and reliability of applications. This report documents the implementation of a complete DevOps pipeline for a machine learning-based project: *Instagram Reach Prediction*.

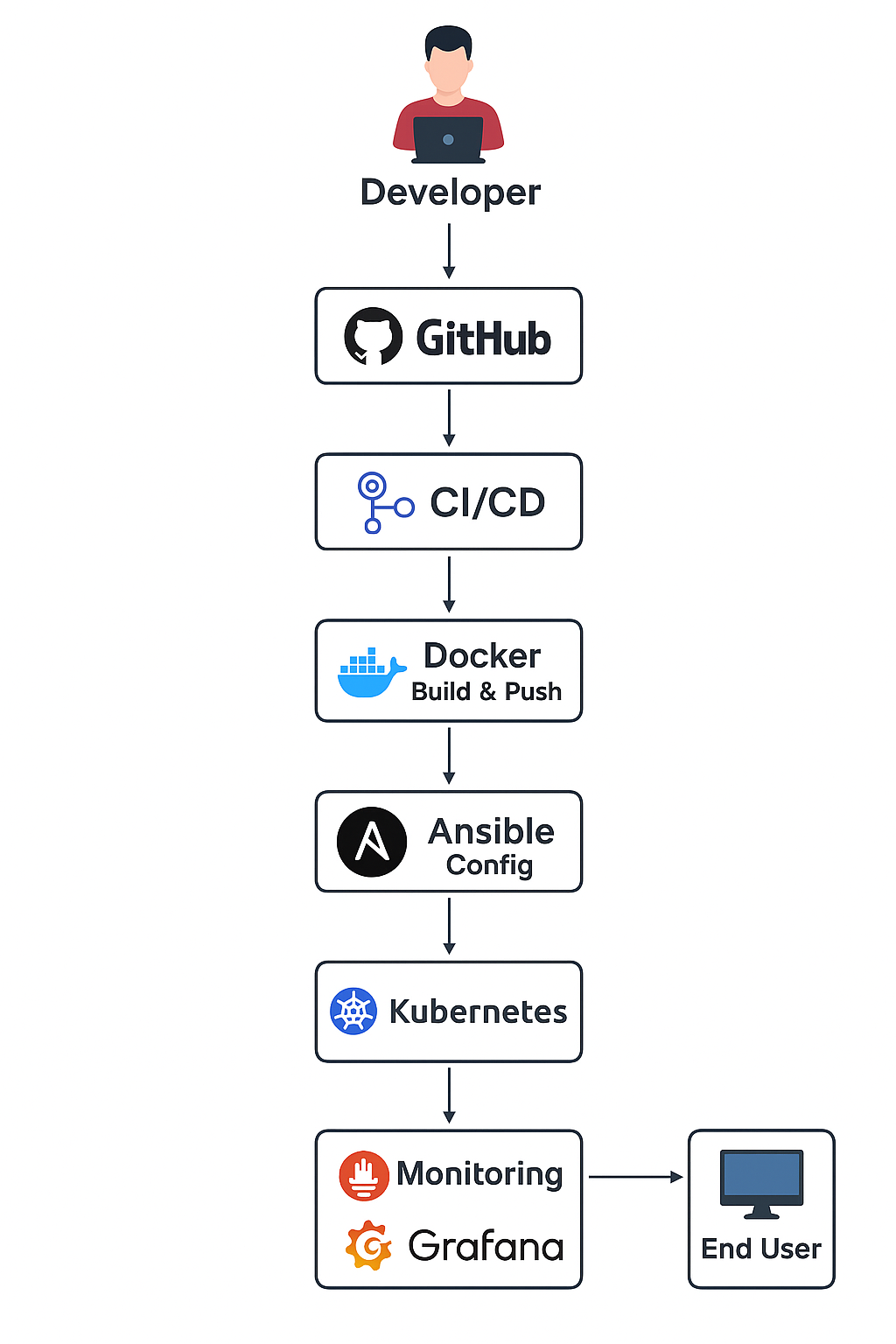
The project’s objective is to predict the reach of an Instagram post based on engagement metrics such as likes, comments, and shares. While the model itself was built earlier using machine learning techniques, the focus of this work is on automating its deployment, monitoring, and scalability using DevOps principles.

The report highlights the steps taken to build an end-to-end DevOps pipeline, including Continuous Integration/Continuous Deployment (CI/CD), Infrastructure as Code (IaC), Containerization, Orchestration, and Monitoring.

**2. Project Architecture**

The architecture designed for this project integrates multiple DevOps tools in a seamless flow.

* **Source Control:** GitHub hosts the project code.
* **CI/CD Tool:** GitHub Actions automates build, test, and deployment.
* **Configuration Management:** Ansible is used to provision runtime dependencies and manage infrastructure.
* **Containerization:** The Flask service is packaged into Docker images for consistency.
* **Orchestration:** Kubernetes (K8s) handles deployment, scaling, rolling updates, and rollbacks.
* **Monitoring:** Prometheus collects metrics from the Flask app, and Grafana visualizes them via custom dashboards.



The architecture ensures that every change in the source code triggers automated pipelines, resulting in a Dockerized service deployed to Kubernetes, with monitoring and logging enabled.

**3. Deployment Strategy (Step 1)**

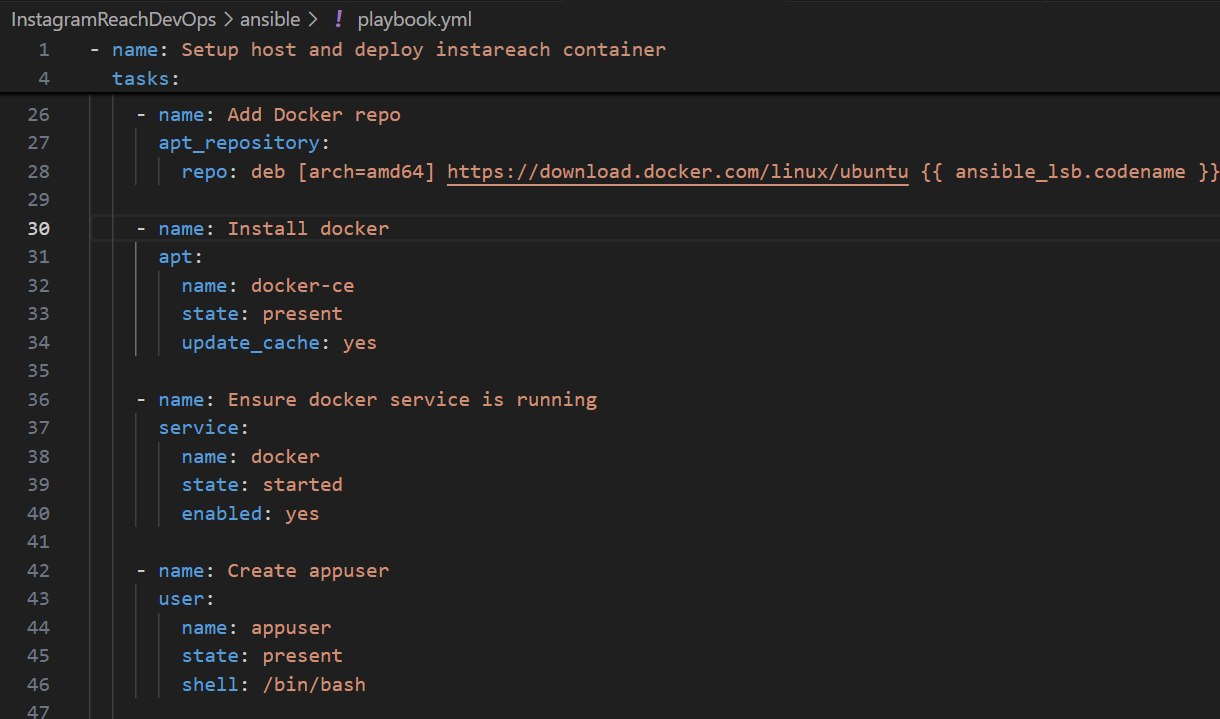
The deployment pipeline was implemented using **GitHub Actions**.

* A workflow file was created to automate the process.
* On every push to the repository, the pipeline executes the following steps:
  1. Checkout code from GitHub.
  2. Build the Docker image of the Flask app.
  3. Push the image to Docker Hub.
  4. Deploy to Kubernetes using manifests.

**Outcome:** Automated build and deployment ensured reliability and faster iteration of updates. The pipeline diagram clearly depicts this flow.

**4. Configuration Management & Infrastructure as Code (Step 2)**

To ensure consistency across environments, **Ansible** was used for provisioning.

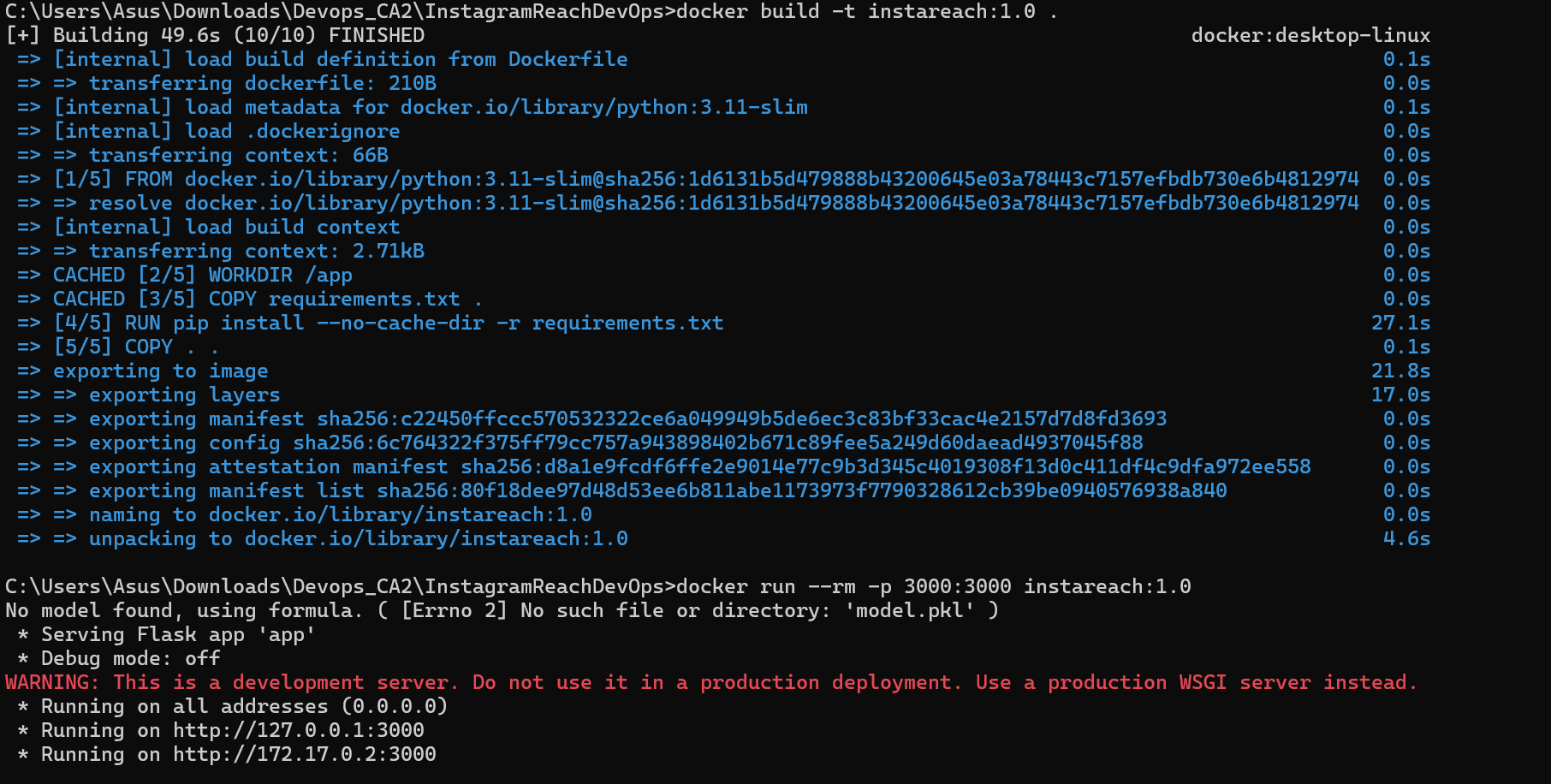
* An Ansible **Playbook** was written to install Python, Flask dependencies, and Docker.
* Inventory files defined target nodes and their configurations.
* This automated the runtime setup, eliminating manual configuration errors.
* 

**Outcome:** With IaC, the runtime environment can be replicated easily on any machine or server.

**5. Containerization & Orchestration (Step 3)**

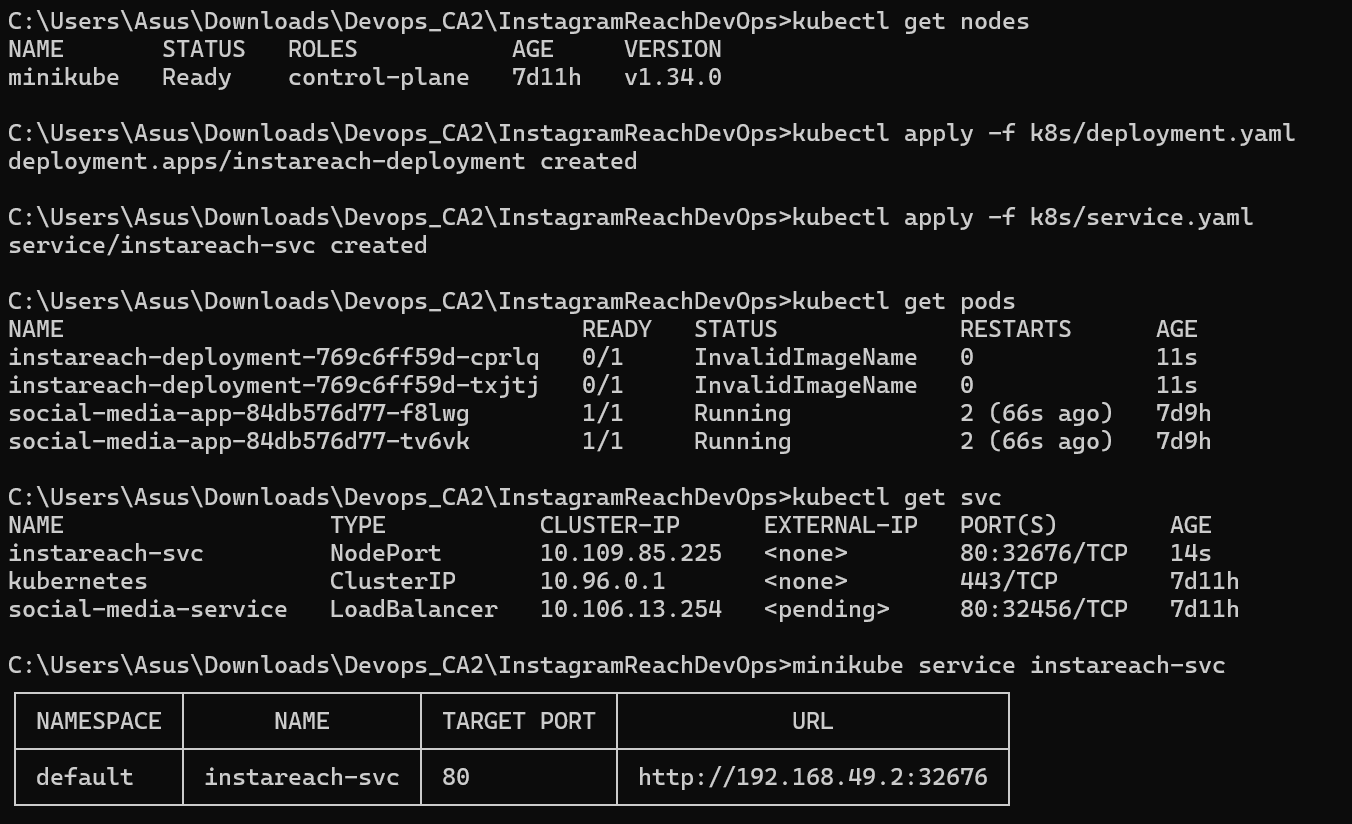
**Containerization**

* The application was Dockerized using a Dockerfile.
* This guaranteed that the app runs in a consistent environment, regardless of the host machine.



**Orchestration**

* Kubernetes manifests were created for:
  + **Deployment** (to define pods and replicas).
  + **Service** (to expose the app internally/externally).
* Rolling update and rollback features were tested to show reliability during updates.

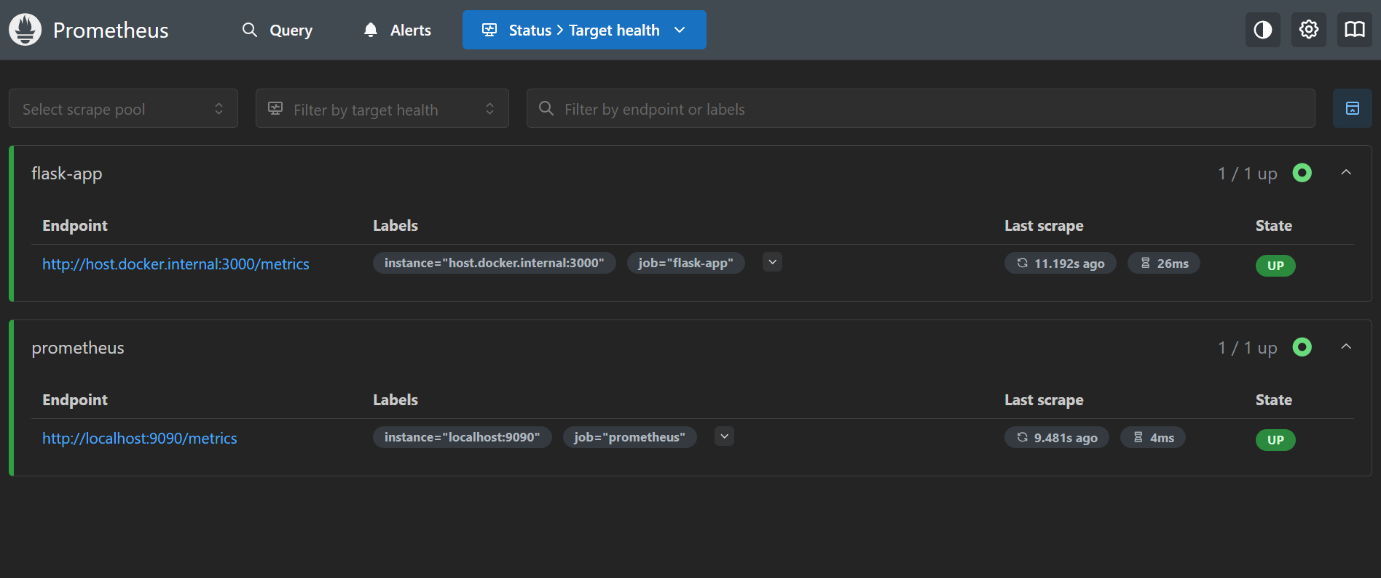


**Outcome:** Kubernetes enabled automatic scaling, high availability, and zero-downtime deployments.

**6. Monitoring & Logging (Step 4)**

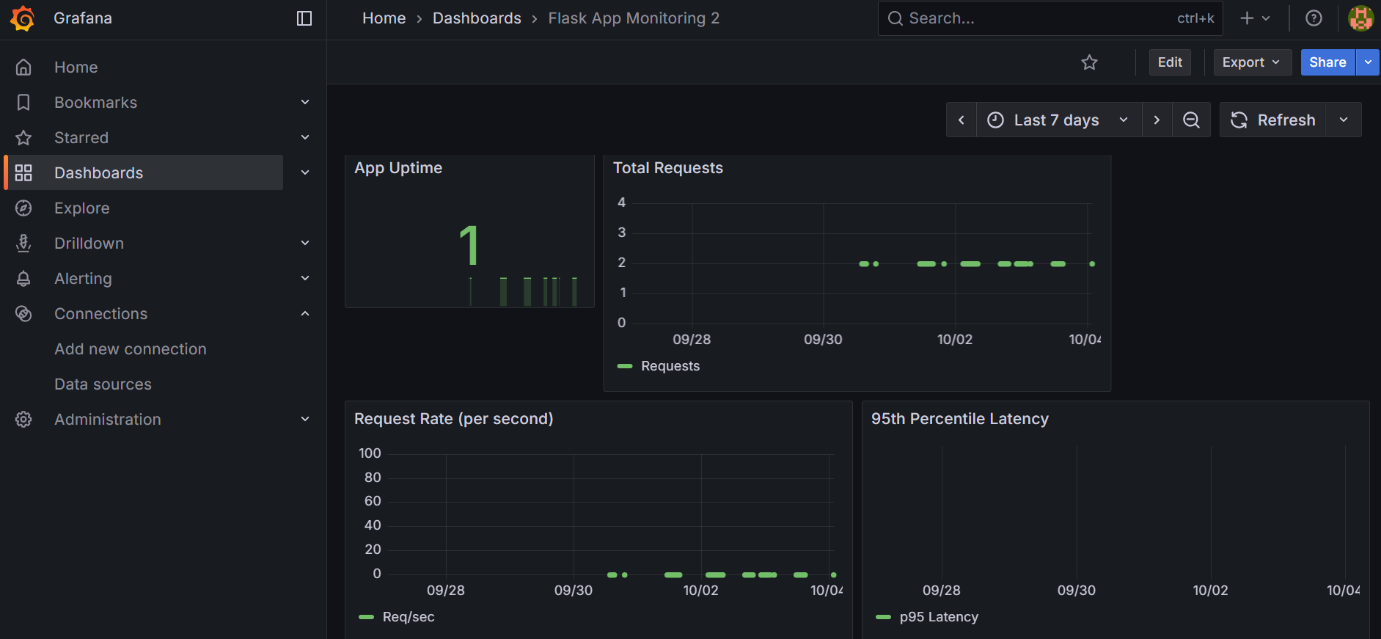
For application monitoring, **Prometheus and Grafana** were integrated.

* The Flask app was instrumented with the **Prometheus client library**, exposing metrics at /metrics.
* Prometheus scraped these metrics, including request counts, latency, memory usage, and uptime.
* Grafana dashboards were created to visualize these metrics, showing application health in real time.



**Sample Metrics Exposed:**

* Total requests served.
* Request latency distribution.
* Application uptime.
* CPU and memory usage.



**Outcome:** Monitoring allowed proactive detection of issues and performance bottlenecks.

**7. Reflection**

**Challenges Faced:**

1. Debugging GitHub Actions workflow YAMLs.
2. Docker networking configuration for Prometheus scraping.
3. Grafana dashboard setup – initial queries did not display correctly.
4. Managing Kubernetes manifest syntax and rollbacks.

**Lessons Learned:**

1. Hands-on understanding of CI/CD pipelines and GitHub Actions.
2. Practical experience with Ansible for configuration management.
3. Deployed and managed containers in Kubernetes.
4. Integrated monitoring and visualization tools for observability.

**8. Conclusion**

This project successfully demonstrated the implementation of a complete DevOps lifecycle for a machine learning-based Flask application. Starting from code management to automated deployment, scaling, and monitoring, every stage of the pipeline was automated and tested.

The integration of GitHub Actions, Ansible, Docker, Kubernetes, Prometheus, and Grafana showcased the potential of DevOps practices in ensuring scalability, reliability, and faster software delivery.

**Future Scope:**

* Extend logging with ELK (Elasticsearch, Logstash, Kibana) stack.
* Implement alerting mechanisms in Grafana for real-time notifications.
* Deploy to a cloud platform like AWS or GCP for real-world scalability.