Experiment No 09

Aim:

To implement and automate server monitoring using Prometheus.

Theory:

Prometheus is an **open-source monitoring and alerting toolkit** designed for reliability and scalability. It collects metrics from servers and applications, stores them in a time-series database, and allows queries using its powerful query language **PromQL**. It is often paired with **Grafana** for visualization.

Key features:

- Pull-based data collection via HTTP endpoints.
- Multi-dimensional data model (metrics with labels).
- Built-in alert manager.
- Easy integration with containerized and cloud-native environments.

---

2. Why Use Prometheus for Server Monitoring?

- Continuous monitoring of CPU, memory, disk, and network usage.
- Detect failures, performance bottlenecks, or downtime.
- Automation of alerts (email, Slack, PagerDuty, etc.).
- Scalability for microservices and containerized infrastructure.

---

3. Architecture for Server Monitoring

The monitoring system typically includes:

1. **Prometheus Server** – pulls metrics from exporters and stores them.
2. **Node Exporter** – installed on each server to expose system-level metrics (CPU, RAM, Disk, I/O, Network).
3. **Alertmanager** – sends alerts based on defined rules.
4. **Grafana (Optional)** – for real-time dashboards and visualization.

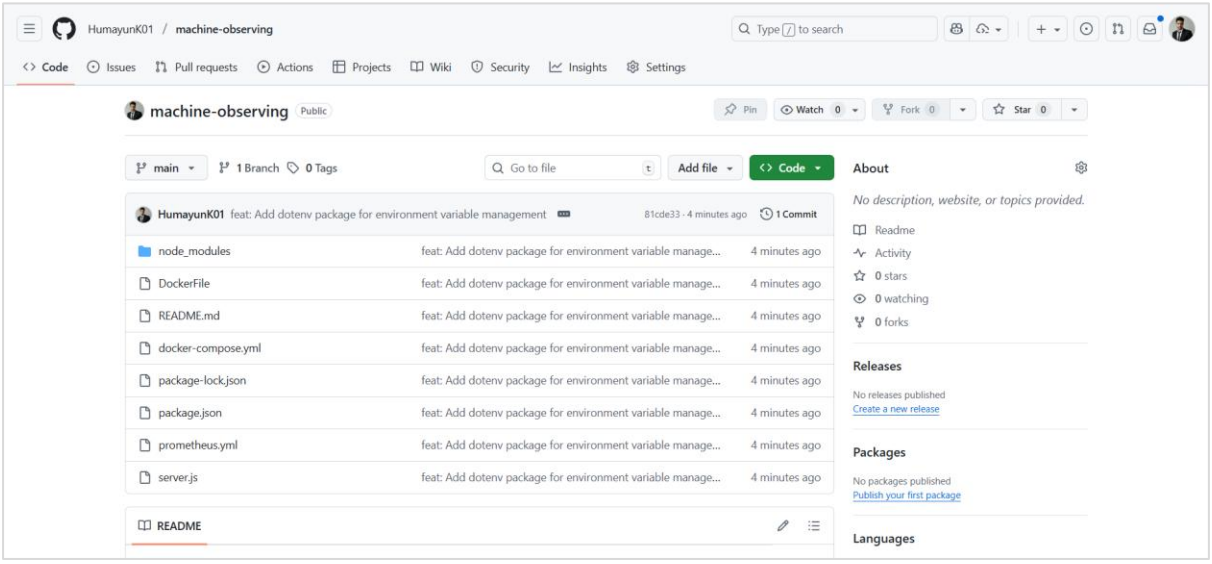
---

4. Steps to Implement

a) Get the Code

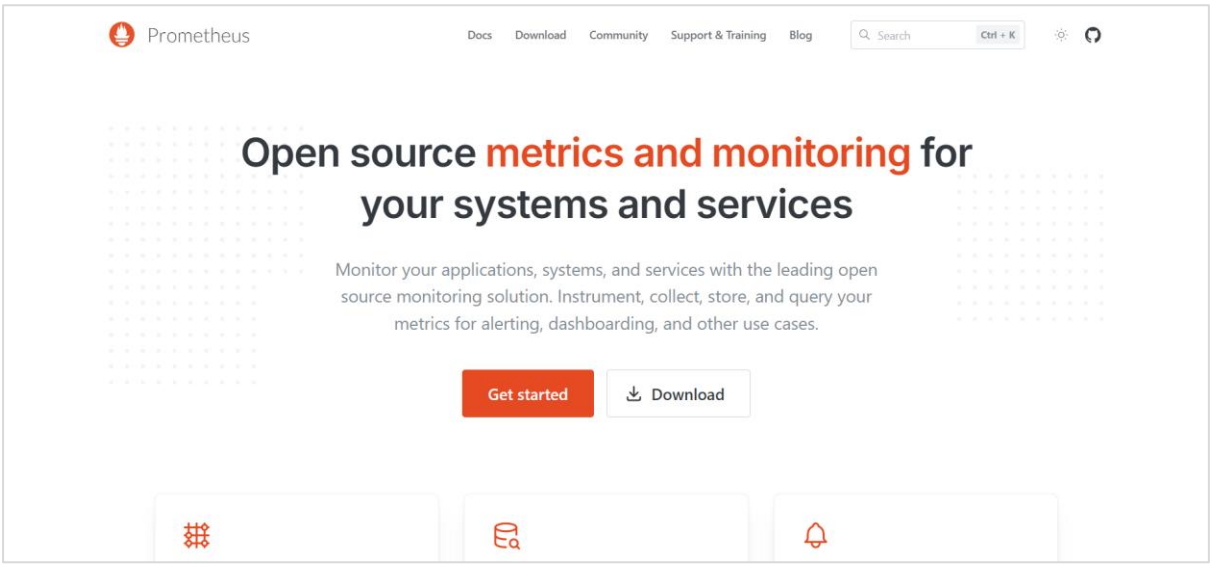
- Clone the repository for the full setup:

```
>> git clone https://github.com/HumayunK01/machine-observing
>> cd machine-observing
```



b) Install Prometheus

- Deploy Prometheus on your monitoring server.
- Edit prometheus.yml in the repo to configure the targets you want Prometheus to monitor.



c) Install Node Exporter on Servers

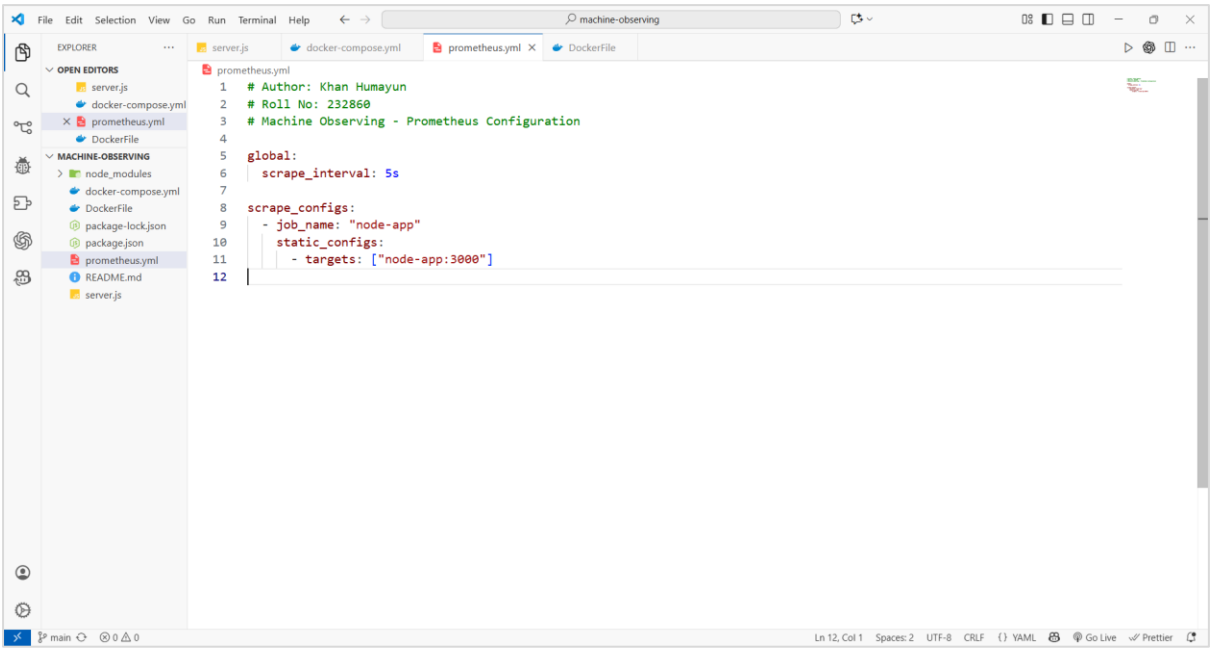
- **On each server to be monitored, install Node Exporter**:

By default, metrics are available at http://<server_ip>:9100/metrics.

Add each server as a scrape target in prometheus.yml

d) Configure Alerting

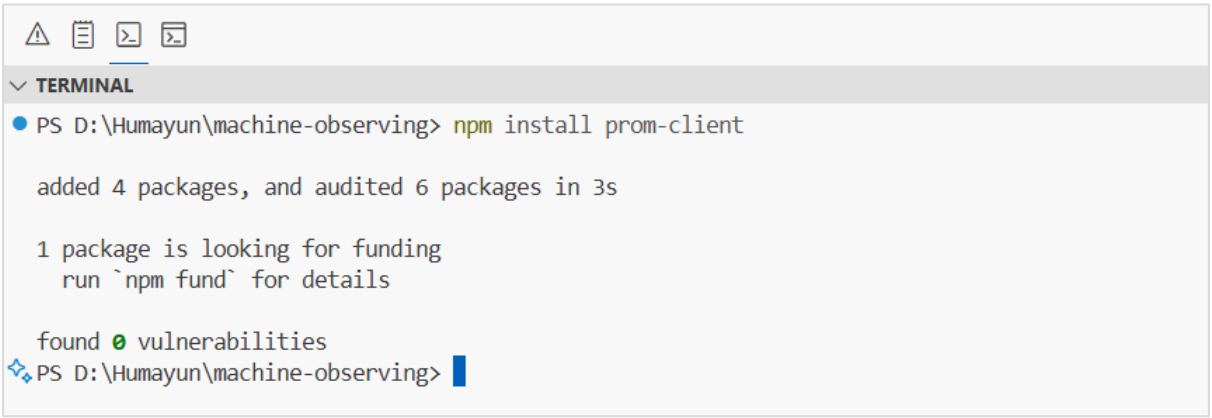
- Create alert rules in Prometheus (for example, CPU > 80%).
- Example rule in prometheus.yml or a separate file:

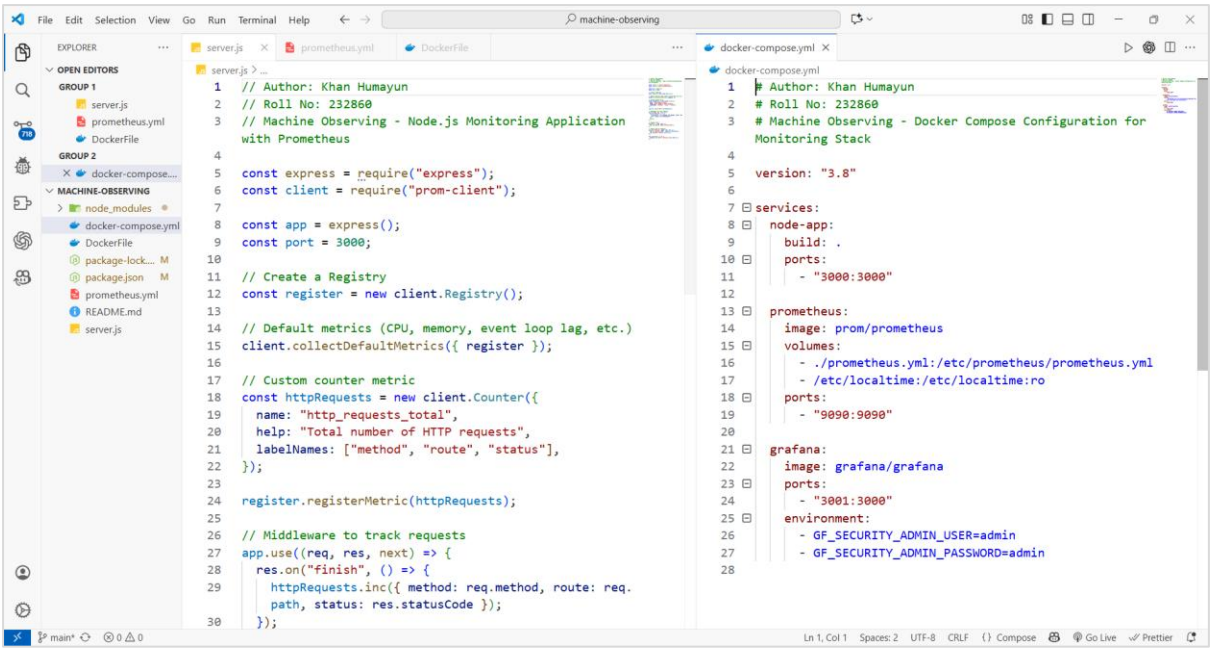


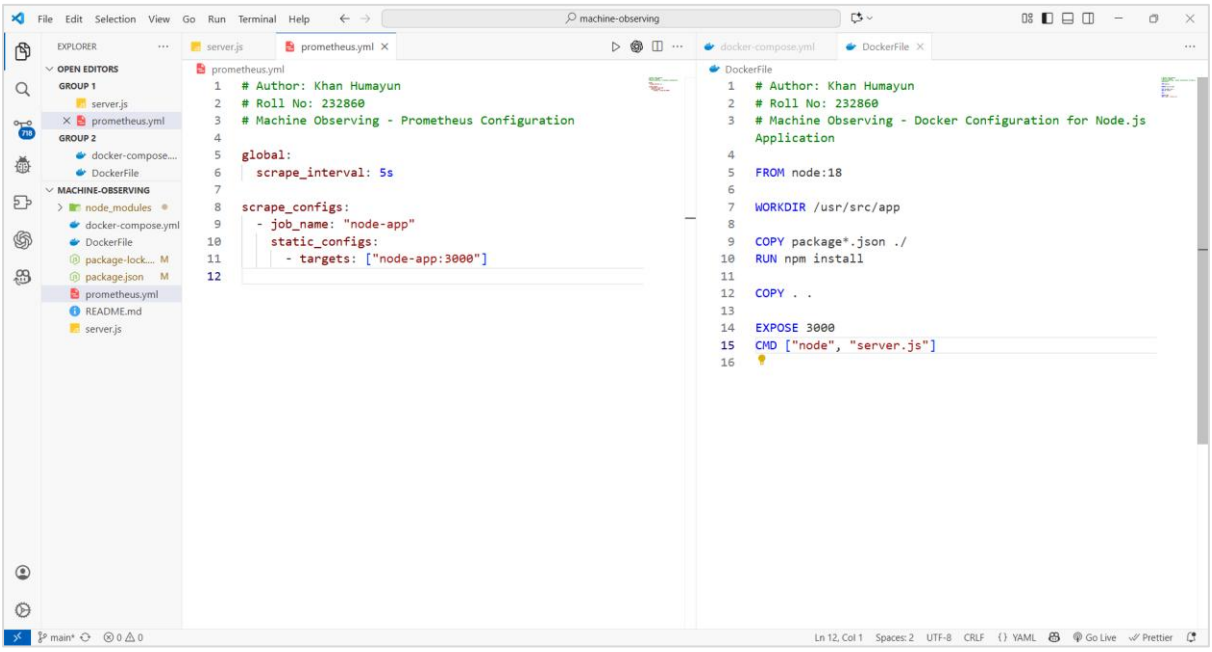
e) Automate Setup

- Add Prometheus Metrics to Node.js
- We'll use the prom-client library to expose metrics.

```
>> npm install prom-client
```







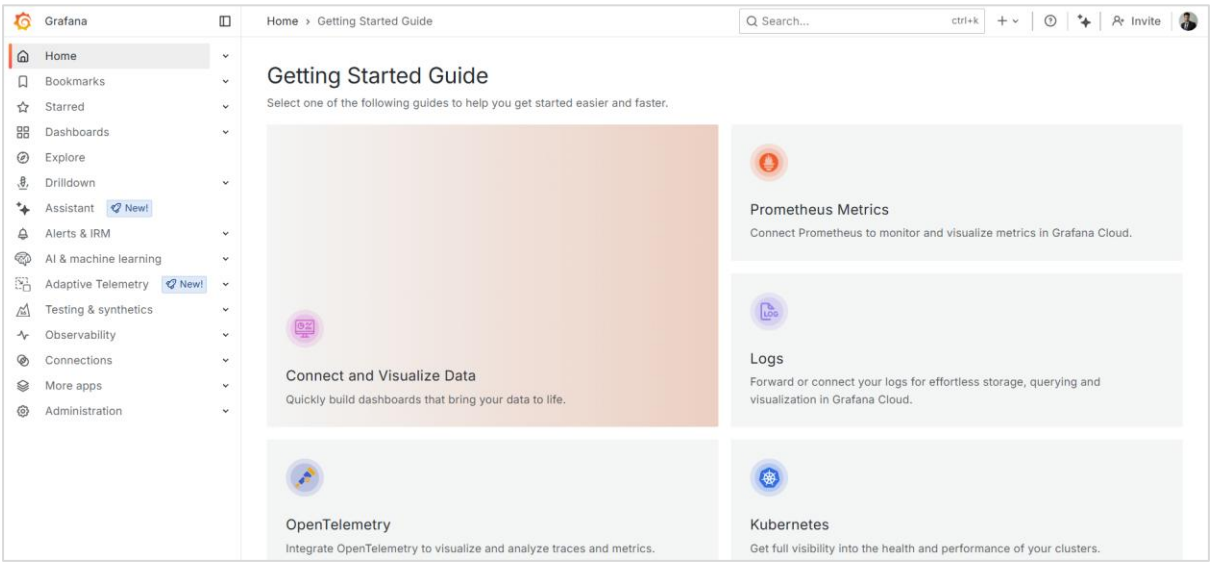
f) Docker Compose

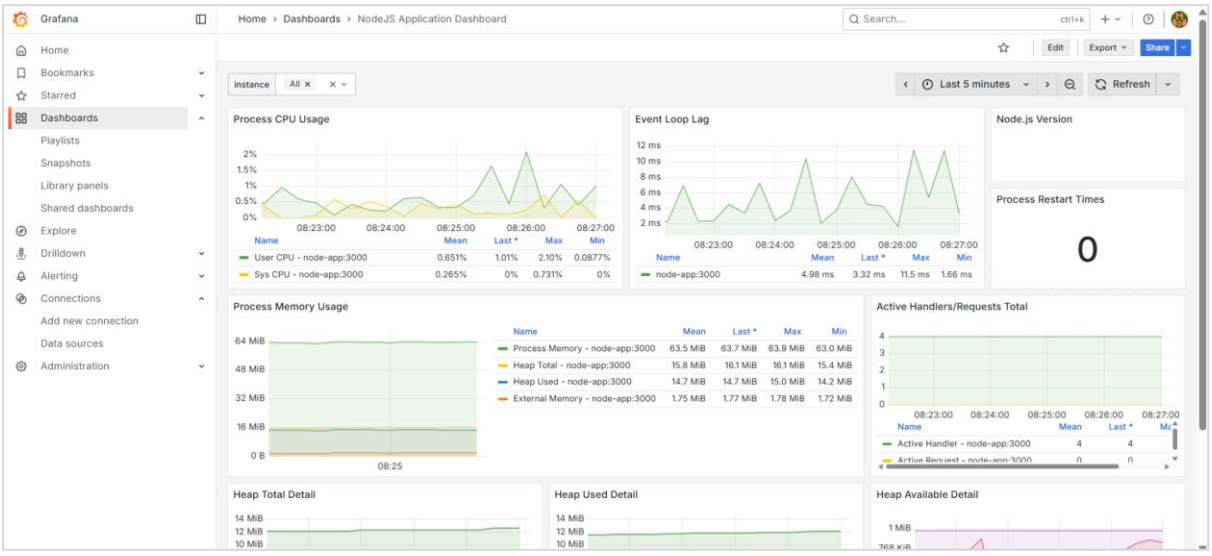
In last you should get:

```
 [+] Running 5/5
  ✓ node-app                                Built
  ✓ Network machine-observing_default       Created
  ✓ Container machine-observing-node-app-1  Started
  ✓ Container machine-observing-prometheus-1  Started
  ✓ Container machine-observing-grafana-1   Started
PS D:\Humayun\machine-observing>
```

g) Visualization

- Start Grafana (included in the repo's Docker Compose file).
- Log into Grafana at http://localhost:3001 (username/password default: admin/admin).
- Connect Grafana to Prometheus (URL: http://prometheus:9090).
- Import official Node Exporter dashboards for server metrics.





Conclusion:

Implementing and automating server monitoring using Prometheus ensures that system performance, availability, and resource utilization are continuously tracked in real time. With its powerful data collection, time-series storage, and alerting capabilities, Prometheus helps teams quickly detect anomalies, resolve issues before they escalate, and maintain overall infrastructure health. By integrating automation, organizations reduce manual overhead, gain consistent observability across servers and services, and improve reliability and efficiency in operations. This makes Prometheus a robust and scalable solution for proactive server monitoring in modern IT environments.