

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.
-

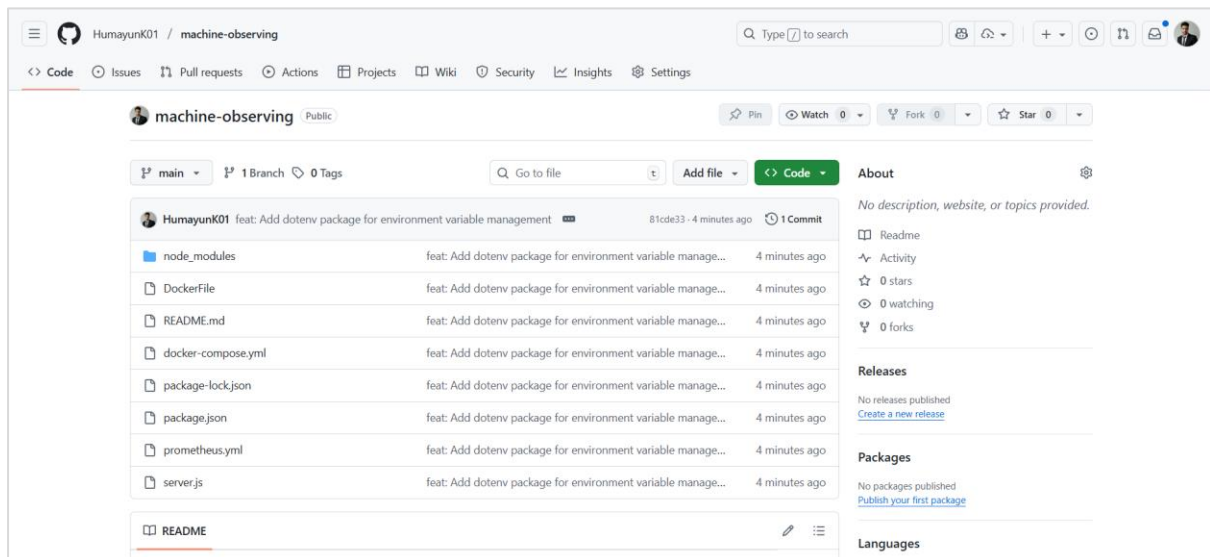
4. Steps to Implement

a) Get the Code

- Clone the repository for the full setup:

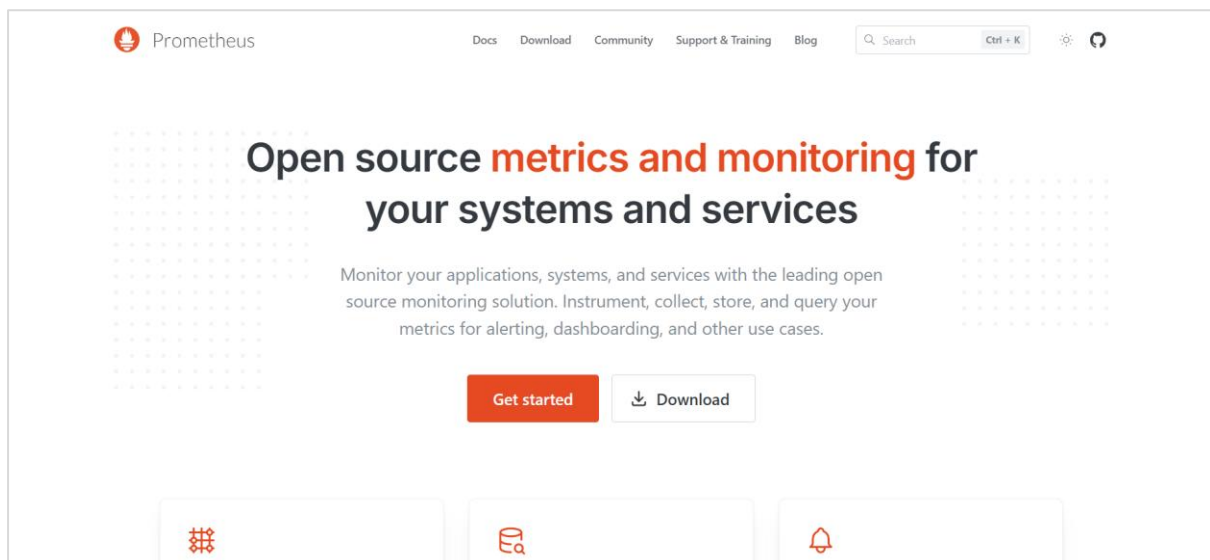
```
>> git clone https://github.com/HumayunK01/machine-observing
```

```
>> 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

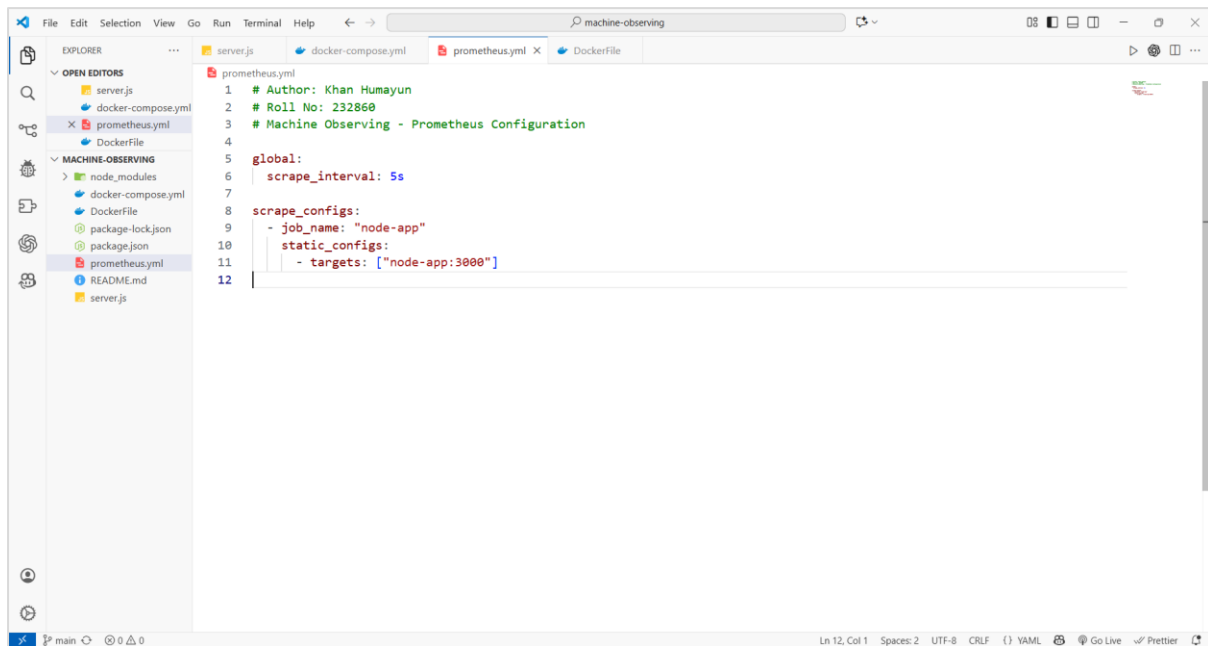
- On each server to be monitored, install Node Exporter:

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

Add each server as a scrape target in `prometheus.yml`

d) Configure Alerting

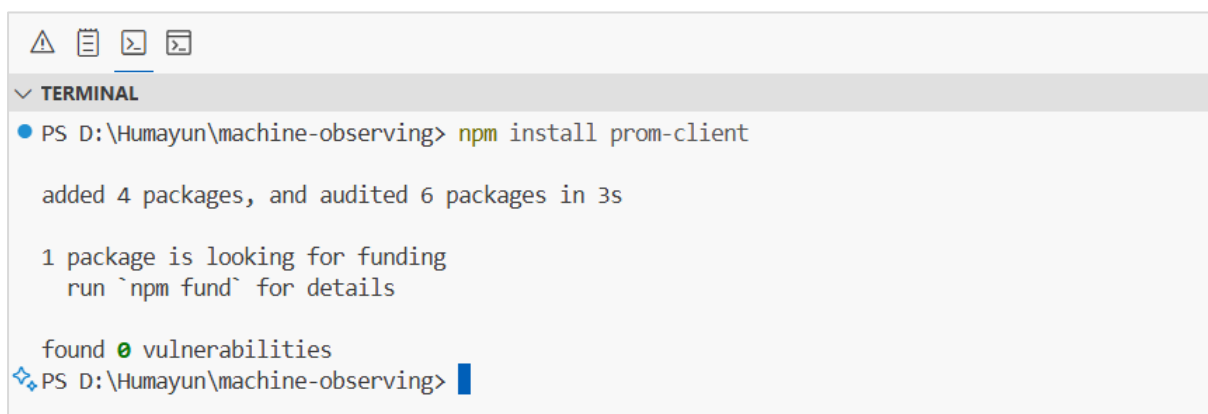
- Create alert rules in Prometheus (for example, CPU > 80%).
- Example rule in `prometheus.yml` or a separate file:



e) Automate Setup

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

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



```

1 // Author: Khan Humayun
2 // Roll No: 232860
3 // Machine Observing - Node.js Monitoring Application
  with Prometheus
4
5 const express = require("express");
6 const client = require("prom-client");
7
8 const app = express();
9 const port = 3000;
10
11 // Create a Registry
12 const register = new client.Registry();
13
14 // Default metrics (CPU, memory, event loop lag, etc.)
15 client.collectDefaultMetrics({ register });
16
17 // Custom counter metric
18 const httpRequests = new client.Counter({
19   name: "http_requests_total",
20   help: "Total number of HTTP requests",
21   labelNames: ["method", "route", "status"],
22 });
23
24 register.registerMetric(httpRequests);
25
26 // Middleware to track requests
27 app.use((req, res, next) => {
28   res.on("finish", () => {
29     httpRequests.inc({ method: req.method, route: req.
30       path, status: res.statusCode });
31   });
32 });
  
```

```

1 # Author: Khan Humayun
2 # Roll No: 232860
3 # Machine Observing - Docker Compose Configuration for
  Monitoring Stack
4
5 version: "3.8"
6
7 services:
8   node-app:
9     build: .
10    ports:
11      - "3000:3000"
12
13   prometheus:
14     image: prom/prometheus
15     volumes:
16       - ./prometheus.yml:/etc/prometheus/prometheus.yml
17       - /etc/localtime:/etc/localtime:ro
18     ports:
19       - "9090:9090"
20
21   grafana:
22     image: grafana/grafana
23     ports:
24       - "3001:3000"
25     environment:
26       - GF_SECURITY_ADMIN_USER=admin
27       - GF_SECURITY_ADMIN_PASSWORD=admin
  
```

```

1 # Author: Khan Humayun
2 # Roll No: 232860
3 # Machine Observing - Prometheus Configuration
4
5 global:
6   scrape_interval: 5s
7
8 scrape_configs:
9   - job_name: "node-app"
10     static_configs:
11       - targets: ["node-app:3000"]
  
```

```

1 # Author: Khan Humayun
2 # Roll No: 232860
3 # Machine Observing - Docker Configuration for Node.js
  Application
4
5 FROM node:18
6
7 WORKDIR /usr/src/app
8
9 COPY package*.json ./
10 RUN npm install
11
12 COPY . .
13
14 EXPOSE 3000
15 CMD ["node", "server.js"]
  
```

f) Docker Compose

```

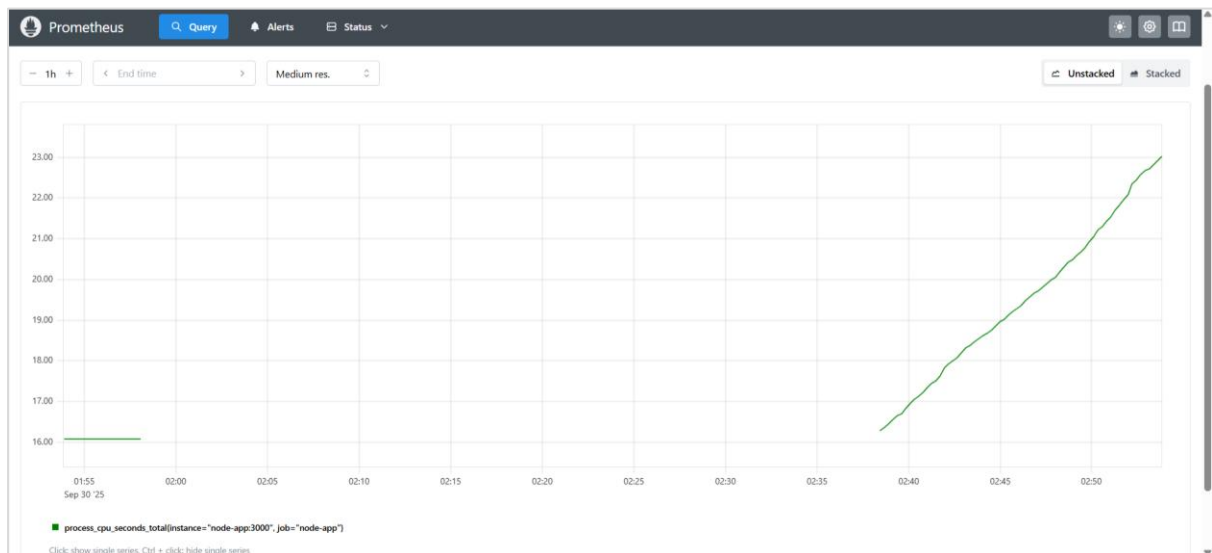
PS D:\Humayun\machine-observing>
PS D:\Humayun\machine-observing>
PS D:\Humayun\machine-observing> docker-compose up -d
time="2025-09-29T16:31:05+05:30" level=warning msg="D:\Humayun\machine-observing\docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
unable to get image 'grafana/grafana': error during connect: Get "http://%2F%2F%2FdockerDesktopLinuxEngine/v1.49/images/grafana/grafana/json": open //./pipe/dockerDesktopLinuxEngine: The system cannot find the file specified.
PS D:\Humayun\machine-observing> docker-compose up -d
time="2025-09-29T16:34:29+05:30" level=warning msg="D:\Humayun\machine-observing\docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 15/21
- grafana [██████████] 89.13MB / 200.5MB Pulling 78.0s
- prometheus [██████████] 80.8MB / 122.6MB Pulling 78.0s
  
```

In last you should get:

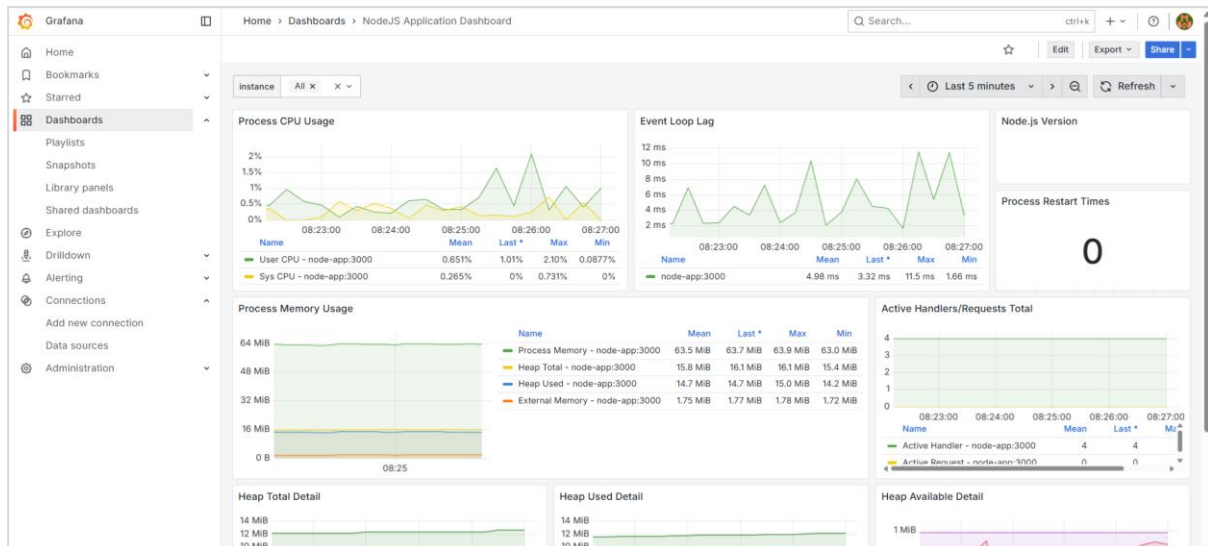
```
[+] Running 5/5
✓ node-app                               Built
✓ Network machine-observing_default      Created
✓ Container machine-observing-node-app-1 Started
✓ Container machine-observing-prometheus-1 Started
✓ Container machine-observing-grafana-1  Started
○ PS D:\Humayun\machine-observing>
```

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.



The image shows the Grafana 'Getting Started Guide' dashboard. The left sidebar contains a navigation menu with options like Home, Bookmarks, Starred, Dashboards, Explore, Drilldown, Assistant, Alerts & IIR, AI & machine learning, Adaptive Telemetry, Testing & synthetics, Observability, Connections, More apps, and Administration. The main content area is titled 'Getting Started Guide' and includes a search bar. Below the title, there's a section 'Select one of the following guides to help you get started easier and faster.' with four cards: 'Prometheus Metrics', 'Logs', 'OpenTelemetry', and 'Kubernetes'. Each card provides a brief description of the guide.



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.