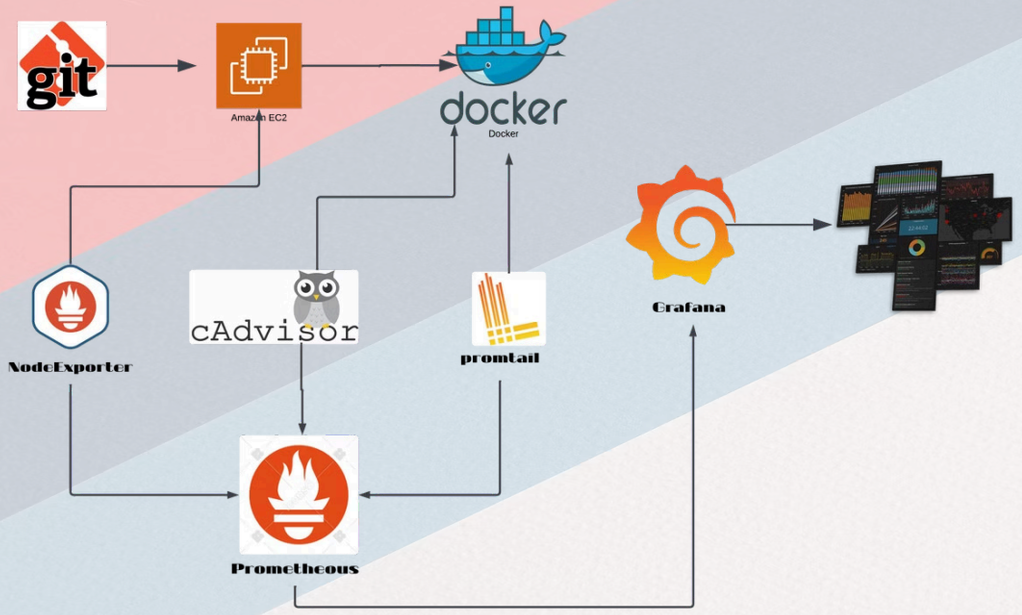
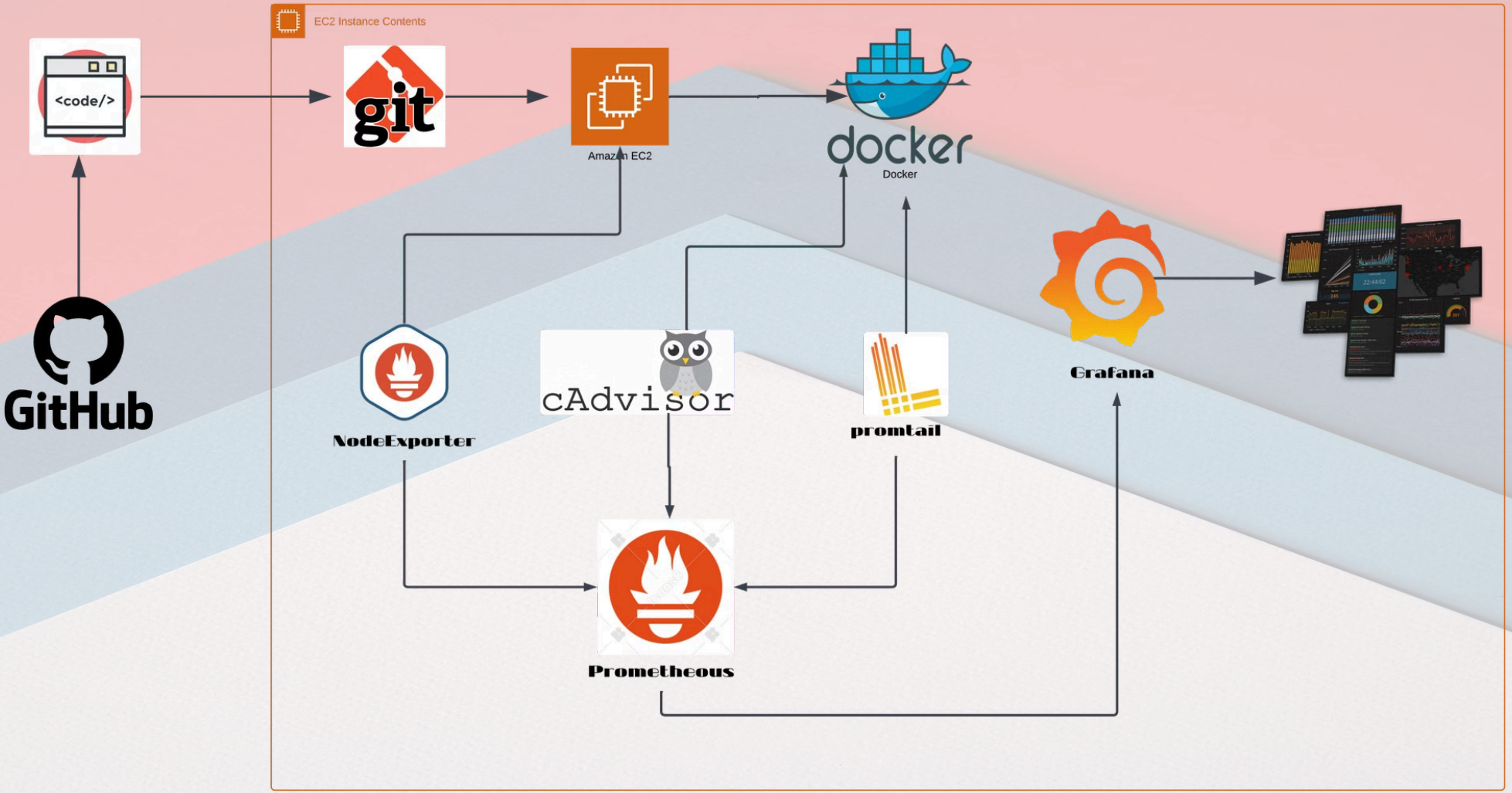


Building a Powerful Observability Stack for Dockerized Applications: A Complete Guide

- **Docker & Docker Compose**: Containerization and orchestration.
- **Prometheus**: Metrics collection and monitoring.
- **Grafana**: Data visualization and dashboard creation.
- **cAdvisor**: Container resource monitoring.
- **Node Exporter**: Hardware and OS metrics exporter.
- **Notes App**: A custom service to demonstrate monitoring.

Author : Abhisheke Kumar

Mentor : Shubham Londhe



Project Objective:

The goal is to set up a complete **Observability Stack** for Dockerized applications using the following tools:

- **Grafana** for data visualization.
- **Prometheus** as a time-series database and metric collector.
- **Node Exporter** for system-level metrics collection.
- **cAdvisor** for container resource usage monitoring.
- **Loki** for log aggregation.
- **Promtail** for log shipping.

This integrated solution will provide comprehensive monitoring of system and container metrics, application logs, and data visualization in Grafana.

Step-by-Step Implementation:

Step 1: Create a Virtual Server

- Provision a virtual server with:
  - **4GB RAM**
  - **2 vCPUs**

Step 2: Install Docker and Docker Compose

1. Update and upgrade system packages: `apt-get update apt-get upgrade`
2. Install Docker: `apt-get install docker.io`
3. Add your user to the Docker group: `usermod -aG docker $USER && newgrp docker`
4. Install Docker Compose: `apt-get install docker-compose`
5. Clone the repository: `git clone https://github.com/LondheShubham153/aws-node-http-api-project`

Step 3: Build the Application Docker Image

- Create a Dockerfile to build the image for your application:

```
# Build Stage
FROM python:3.9 AS builder
WORKDIR /app
COPY requirements.txt .
RUN apt-get update && apt-get install -y gcc libffi-dev libssl-dev python3-dev \
    && python -m venv /venv \
    && . /venv/bin/activate \
    && pip install --no-cache-dir -r requirements.txt \
    && apt-get remove --purge -y gcc libffi-dev libssl-dev python3-dev \
    && apt-get autoremove -y \
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/*

# Final Stage
FROM python:3.9-alpine
WORKDIR /app
COPY --from=builder /venv /venv
COPY . .
ENV PATH="/venv/bin:$PATH"
EXPOSE 8000
CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]
```

--

Step 4: Create a Docker Compose Configuration

- Create a `docker-compose.yml` file to orchestrate the application and observability tools:

```
version: '3.8'
volumes:
  notesapp: {}
  prometheus-vol: {}
  grafana-vol: {}

networks:
  web-network:
    driver: bridge

services:
  notes-web:
    build:
      context: ./notes-app
    container_name: "notes-app"
    ports:
      - "8000:8000"
    volumes:
      - "notesapp"
    networks:
      - web-network
```

Step 5: Set Up Prometheus and Grafana

1. **Download the Prometheus configuration file:** wget https://raw.githubusercontent.com/prometheus/prometheus/main/documentation/examples/prometheus.yml
2. **Add Prometheus to Docker Compose:**

```
prometheus:
  image: prom/prometheus:latest
  container_name: prometheus
  ports:
    - "9090:9090"
  volumes:
    - prometheus-vol:/prometheus
    - ./prometheus.yml:/etc/prometheus/prometheus.yml:ro
  networks:
    - web-network
  restart: unless-stopped
  depends_on:
    - node-exporter
    - cadvisor
```

3. Add Grafana to Docker Compose:

```
grafana:
  image: grafana/grafana-enterprise
  container_name: grafana
  ports:
    - "3000:3000"
  volumes:
    - grafana-vol:/var/lib/grafana
    - ./data/grafana/provisioning/datasources:/etc/grafana/provisioning/datasources
    - ./data/grafana/provisioning/dashboards:/etc/grafana/provisioning/dashboards
  networks:
    - web-network
  restart: unless-stopped
```

Step 6: Set Up Node Exporter and cAdvisor

- **Node Exporter** configuration:

```
node-exporter:
  image: prom/node-exporter:latest
  container_name: node-exporter
  ports:
    - "9100:9100"
  networks:
    - web-network
  restart: unless-stopped
```

- **cAdvisor** configuration:

```
cadvisor:
  image: gcr.io/cadvisor/cadvisor:latest
  container_name: cadvisor
  ports:
    - "8080:8080"
  volumes:
    - /:/rootfs:ro
    - /var/run:/var/run:rw
    - /sys:/sys:ro
    - /var/lib/docker/:/var/lib/docker:ro
  networks:
    - web-network
  restart: unless-stopped
```

- Add Node Exporter and cAdvisor targets to prometheus.yml:

```
scrape_configs:
  - job_name: "Docker-CaAdvisor"
    static_configs:
      - targets: ["cadvisor:8080"]
  - job_name: "Node-exporter"
    static_configs:
      - targets: ["node-exporter:9100"]
```

Step 7: Set Up Loki and Promtail

- **Loki** configuration in Docker Compose:

```
loki:
  image: grafana/loki:latest
  container_name: loki
  ports:
    - "3100:3100"
  volumes:
    - ./loki-config.yml:/etc/loki/local-config.yml
  networks:
    - web-network
  restart: unless-stopped
```

- **Promtail** configuration in Docker Compose:

```
promtail:
  image: grafana/promtail:latest
  container_name: promtail
  volumes:
    - /var/log:/var/log
    - ./promtail-config.yaml:/etc/promtail/config.yml
  networks:
    - web-network
  restart: unless-stopped
```

- Add Loki target to `prometheus.yml`:

```
scrape_configs:
  - job_name: "Loki"
    static_configs:
      - targets: ["loki:3100"]
```

Step 8: Start the Observability Stack

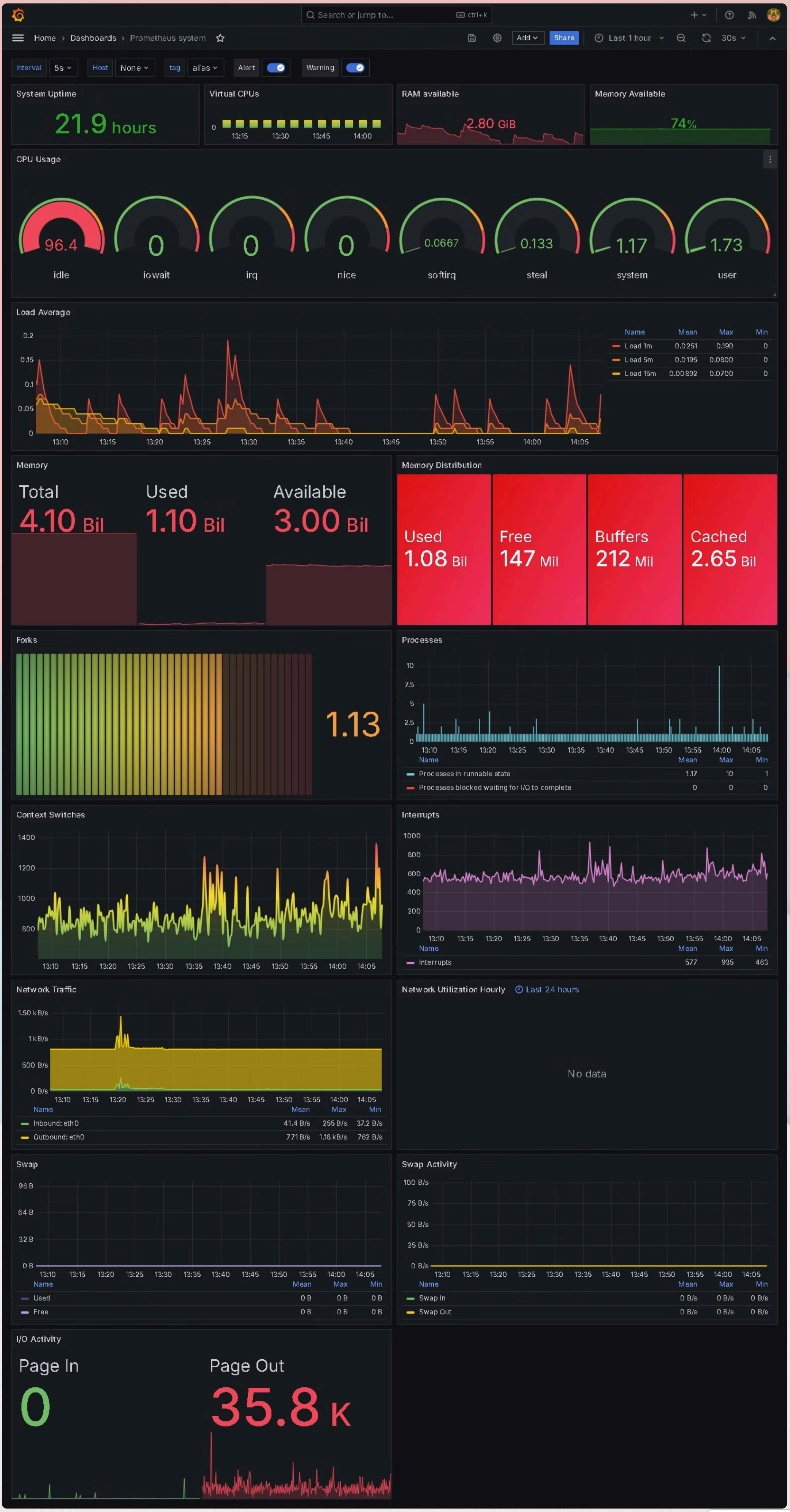
- Run Docker Compose:`docker-compose up -d`

Step 9: Configure Grafana

Access Grafana at `http://<your-server-ip>:3000` and set up the following:

1. **Add Datasources** for Prometheus and Loki.
2. **Create Dashboards** for visualization.
    - You can either:
        - Manually create dashboards.
        - Import pre-configured dashboard templates.

Result :



Thank you ! ❤️