

# Parthiv Patel

## 202412069

### IT644 WSSOA - LAB Assignment 11

## CI/CD Pipeline

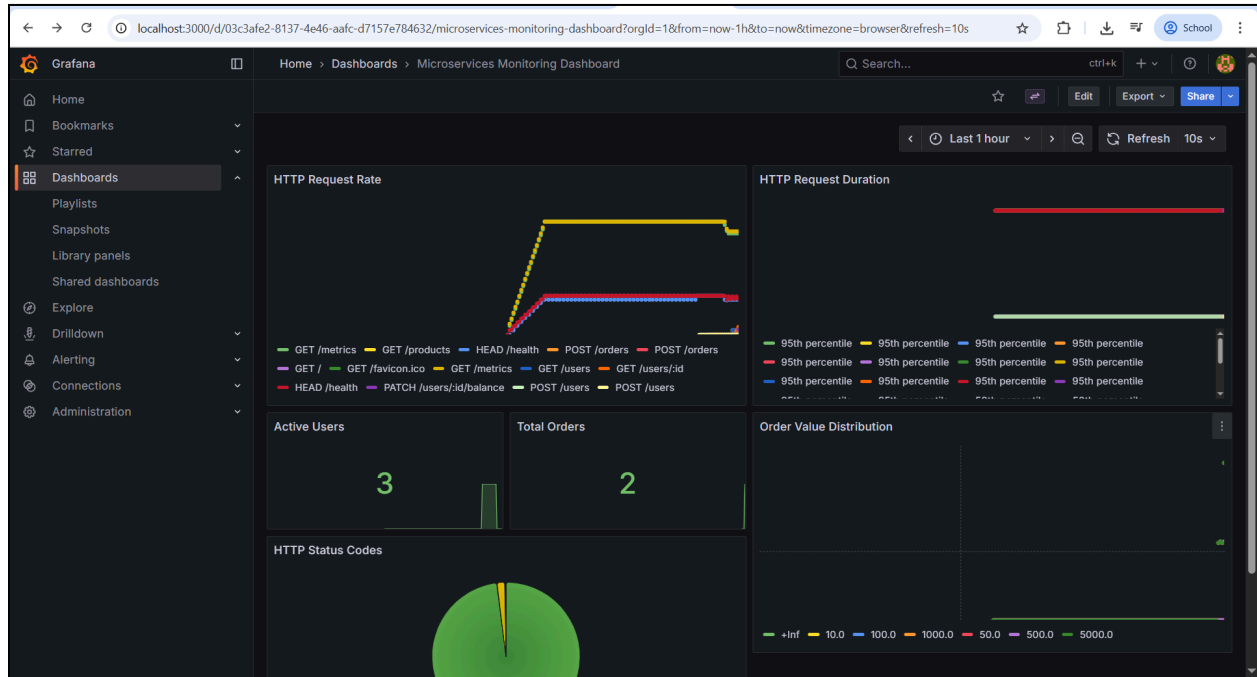
The screenshot displays a GitHub Actions workflow run for a repository named 'parthivppatel / CI-CD'. The workflow is titled 'CI/CD Pipeline' and has a status of 'Success'. It was triggered by a push to the 'master' branch by user 'parthivppatel' at 1 minute ago. The total duration of the run is 1m 44s. The workflow file is named 'ci-cd.yml' and is triggered on a push event.

The workflow consists of three main jobs:

- Matrix Test Services**: 2 jobs completed.
- Matrix Build Docker Images**: 2 jobs completed.
- Deploy Services**: 4s duration.

The left sidebar shows a summary of the run, including a list of jobs and their statuses. The right sidebar shows the workflow file 'ci-cd.yml' and its trigger event 'push'.

# Grafna Dashboard



## WorkFlow

### Pipeline Stages

#### 1. Test Stage

- Runs Jest test suites for both services on every push/PR
- Validates code quality before building

#### 2. Build Stage

- Builds Docker images for both services
- Pushes images to GitHub Container Registry (GHCR)
- Only runs on push events after tests pass

#### 3. Deploy Stage

- Placeholder for production deployment
- Currently outputs deployment information
- Runs only on main/master branch

## Architecture

- User Service (Port 8003): Manages user accounts and balances
- Order Service (Port 8002): Handles product catalog and order processing
- Prometheus: Metrics collection and storage
- Grafana: Metrics visualization dashboard
- Frontend: Web interface (Port 8080)

## Setup

### Prerequisites

- Node.js 18+, Docker Desktop, Git

### Local Setup

1. Install dependencies: ``npm install`` in each service directory
2. Run tests: ``npm test``
3. Start services: ``docker-compose up -d``
4. Access:
  - Frontend: `http://localhost:8080`
  - Grafana: `http://localhost:3000` (admin/admin)
  - Prometheus: `http://localhost:9090`

### GitHub Actions Setup

1. Push code to GitHub repository
2. Workflow automatically runs on push/PR
3. Images published to GHCR: ``ghcr.io/<username>/<service-name>``

## **Key Observations**

### **Strengths**

- All changes tested before deployment
- Immediate validation of code quality
- Docker ensures environment consistency

### **Challenges**

1. Jest configuration required special handling for ES modules
2. Required proper network configuration for service communication
3. Needed explicit `packages: write` permission in workflow
4. curl commands needed PowerShell-specific syntax

### **Solutions**

- Used `@jest/globals` for ES module test compatibility
- Created Docker network for service discovery
- Added workflow permissions for GHCR access
- Provided PowerShell-compatible commands in documentation