CI/CD Pipeline and Monitoring Setup for Web Application

Name: Alkaios Stelios

Overview

This document explains the CI/CD pipeline setup for deploying a web application, including the technologies, tools, and steps taken to build, test, and deploy the application. It also details the monitoring solution used to ensure application health and performance. The pipeline uses GitHub Actions and incorporates a manual approval step before deployment to production.

Technologies and Tools Used

Pipeline

- GitHub Actions: Automates the CI/CD workflow.

- Docker and Docker Compose: Builds and deploys the application.

- Linode Cloud Server: Hosts the application.

Monitoring

- Prometheus: Collects metrics from the infrastructure and application.

- Grafana: Visualizes metrics and provides dashboards.

- Node Exporter: Captures server resource metrics.

- Blackbox Exporter: Probes service endpoints for uptime and response time.

CI/CD Pipeline

Pipeline Workflow

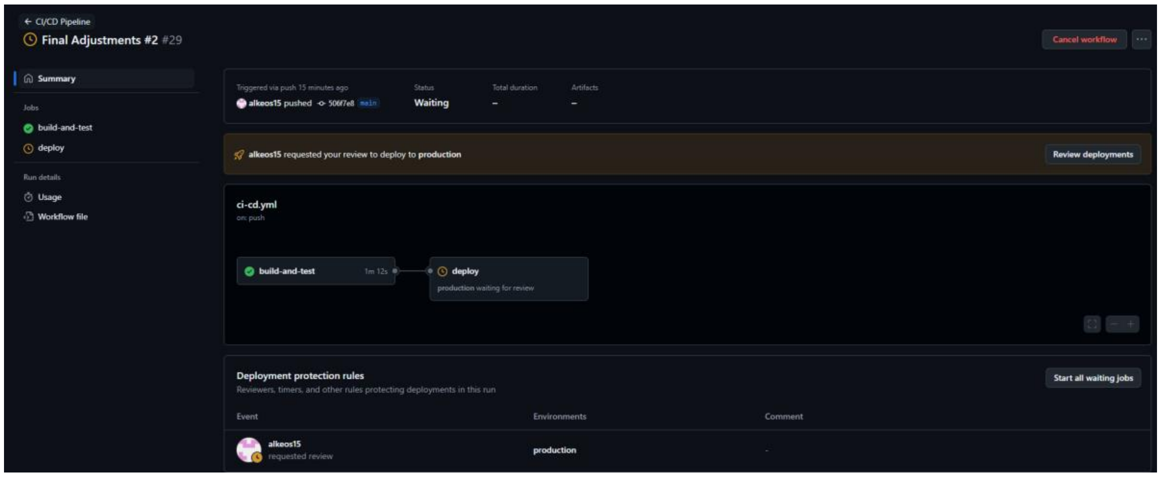
Trigger

The pipeline is triggered automatically on:
- Push events to the main branch.
- Pull request creation targeting the main branch.

Workflow Stages

- Build and Test: Builds Docker images for the frontend and backend.

- Runs automated tests with memory limits to simulate production constraints.


← CI/CD Pipeline
Final Adjustments #2 #29

Cancel workflow

Summary

Jobs
- build-and-test
- deploy

Run details
- Usage
- Workflow file

Triggered via push 15 minutes ago	Status	Total duration	Artifacts
alkeos15 pushed -o- 506f7e8 main	Waiting	–	–

alkeos15 requested your review to deploy to **production** — Review deployments

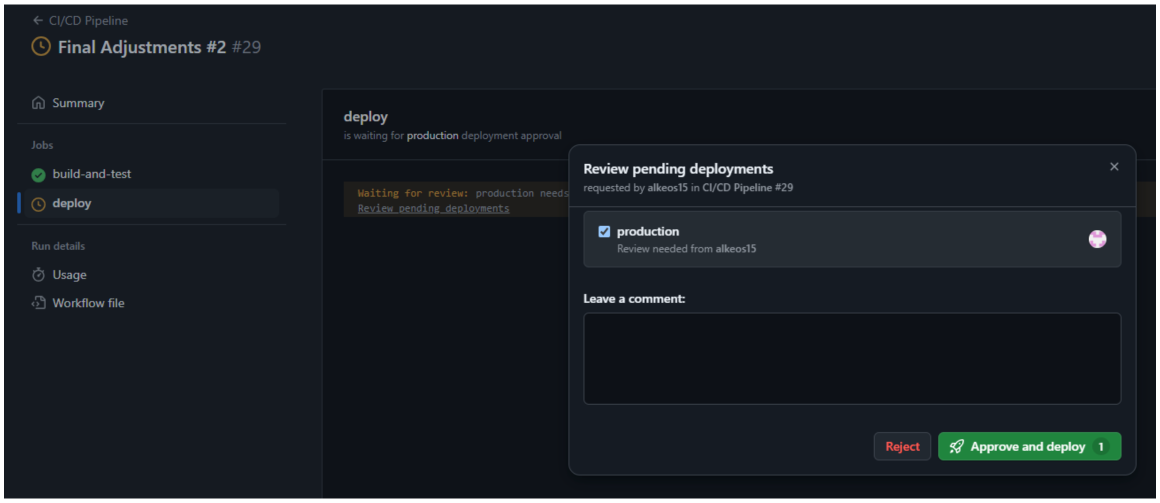
ci-cd.yml
on: push

build-and-test 1m 12s → deploy
production waiting for review

Deployment protection rules
Reviewers, timers, and other rules protecting deployments in this run — Start all waiting jobs

Event	Environments	Comment
alkeos15 requested review	production	-

- Manual Approval: Before deployment, the pipeline pauses and requires manual approval to proceed.


← CI/CD Pipeline
Final Adjustments #2 #29

Summary

Jobs
- build-and-test
- deploy

Run details
- Usage
- Workflow file

deploy
is waiting for **production** deployment approval

```
Waiting for review: production needs
Review pending deployments
```

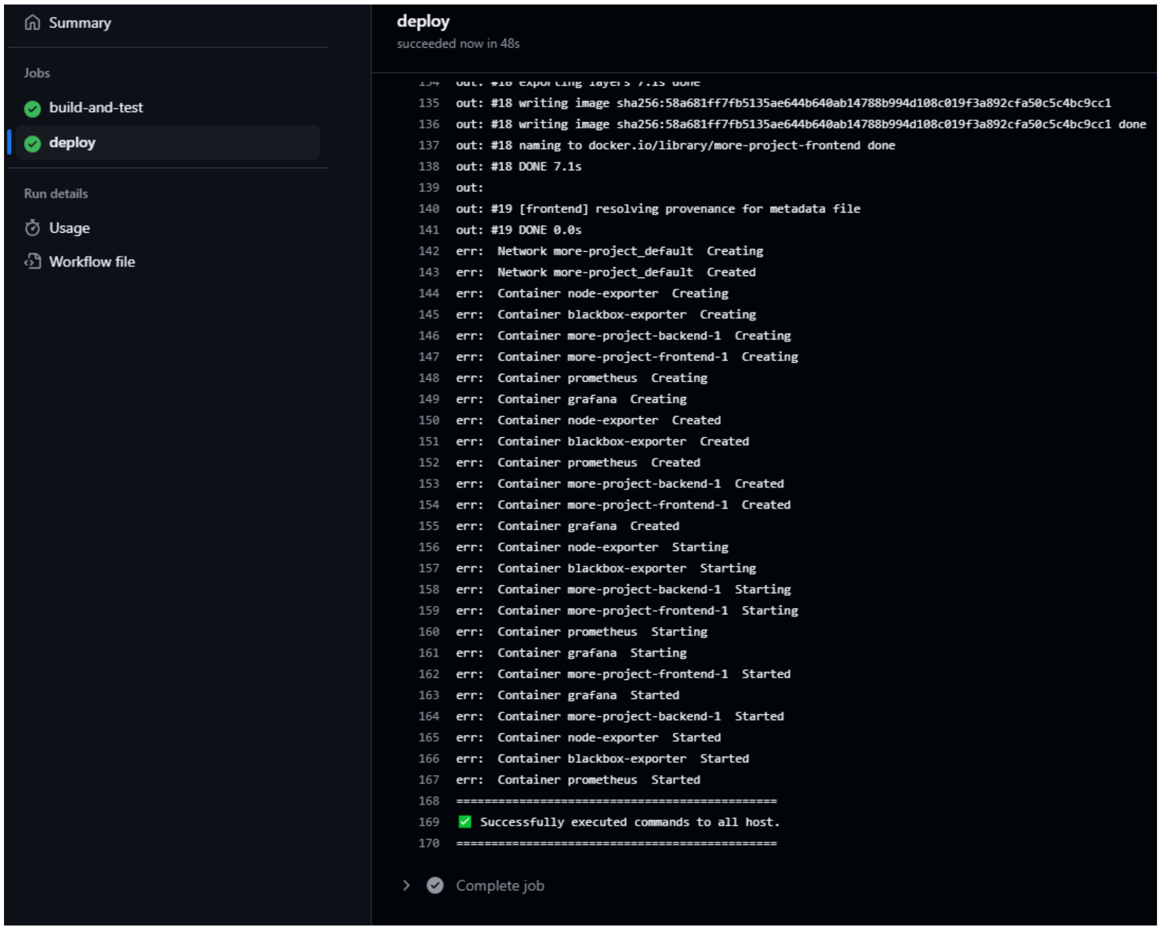
Review pending deployments
requested by alkeos15 in CI/CD Pipeline #29

[✓] production
Review needed from alkeos15

Leave a comment:

Reject | Approve and deploy 1

- Deployment: Deploys the application to the production server using Docker Compose.



```
out: #18 exporting layers 7.1s done
out: #18 writing image sha256:58a681ff7fb5135ae644b640ab14788b994d108c019f3a892cfa50c5c4bc9cc1
out: #18 writing image sha256:58a681ff7fb5135ae644b640ab14788b994d108c019f3a892cfa50c5c4bc9cc1 done
out: #18 naming to docker.io/library/more-project-frontend done
out: #18 DONE 7.1s
out:
out: #19 [frontend] resolving provenance for metadata file
out: #19 DONE 0.0s
err:  Network more-project_default  Creating
err:  Network more-project_default  Created
err:  Container node-exporter  Creating
err:  Container blackbox-exporter  Creating
err:  Container more-project-backend-1  Creating
err:  Container more-project-frontend-1  Creating
err:  Container prometheus  Creating
err:  Container grafana  Creating
err:  Container node-exporter  Created
err:  Container blackbox-exporter  Created
err:  Container prometheus  Created
err:  Container more-project-backend-1  Created
err:  Container more-project-frontend-1  Created
err:  Container grafana  Created
err:  Container node-exporter  Starting
err:  Container blackbox-exporter  Starting
err:  Container more-project-backend-1  Starting
err:  Container more-project-frontend-1  Starting
err:  Container prometheus  Starting
err:  Container grafana  Starting
err:  Container more-project-frontend-1  Started
err:  Container grafana  Started
err:  Container more-project-backend-1  Started
err:  Container node-exporter  Started
err:  Container blackbox-exporter  Started
err:  Container prometheus  Started
==============================================
✅ Successfully executed commands to all host.
==============================================
```

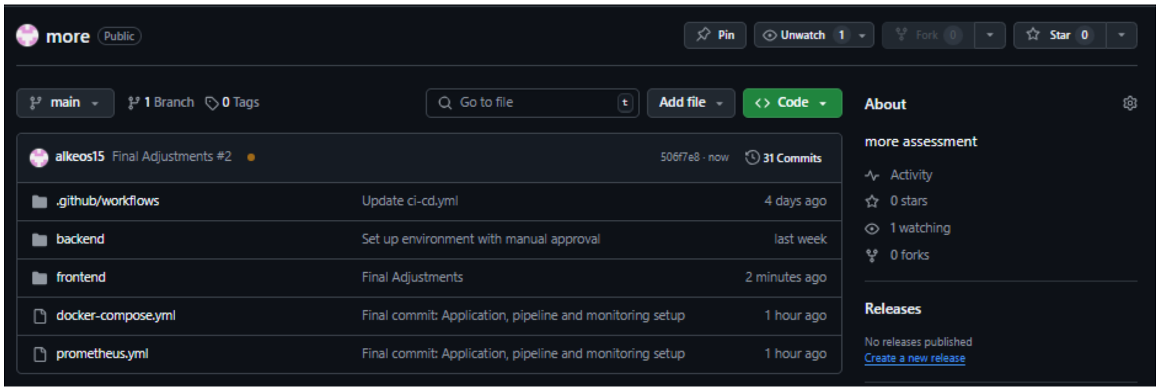
Pipeline Configuration

The CI/CD pipeline is defined in `.github/workflows/ci-cd.yml`:

Steps Taken to Set Up the Pipeline

1. Create the GitHub Repository

Created a new repository: https://github.com/alkeos15/more.git.
Added the application code (backend, frontend) and configuration files:
- docker-compose.yml
- prometheus.yml
- .github/workflows/ci-cd.yml


more Public

Pin | Unwatch 1 | Fork 0 | Star 0

main | 1 Branch | 0 Tags | Go to file | Add file | Code

alkeos15 Final Adjustments #2 · 506f7e8 · now · 31 Commits

.github/workflows	Update ci-cd.yml	4 days ago
backend	Set up environment with manual approval	last week
frontend	Final Adjustments	2 minutes ago
docker-compose.yml	Final commit: Application, pipeline and monitoring setup	1 hour ago
prometheus.yml	Final commit: Application, pipeline and monitoring setup	1 hour ago

About

more assessment

Activity
0 stars
1 watching
0 forks

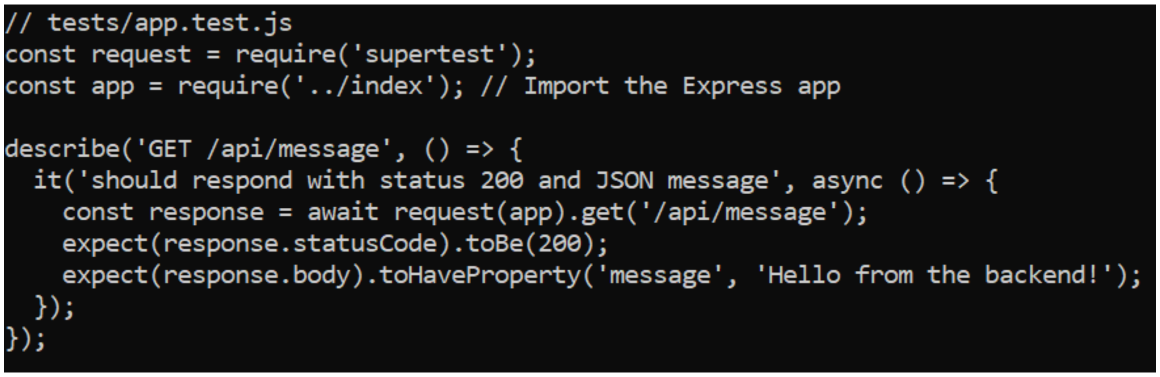
Releases

No releases published
Create a new release

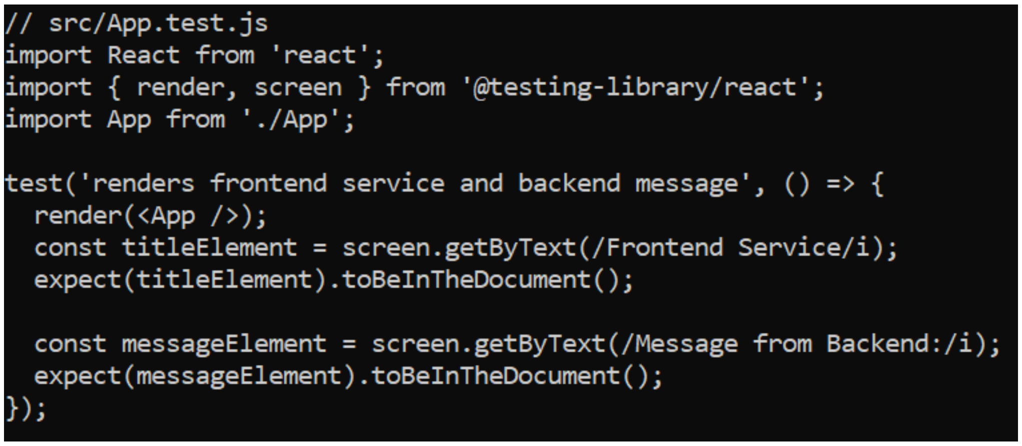
2. Define the CI/CD Pipeline

Created `.github/workflows/ci-cd.yml` to define the pipeline.
Configured the following stages:
- Build and test steps to ensure code correctness.

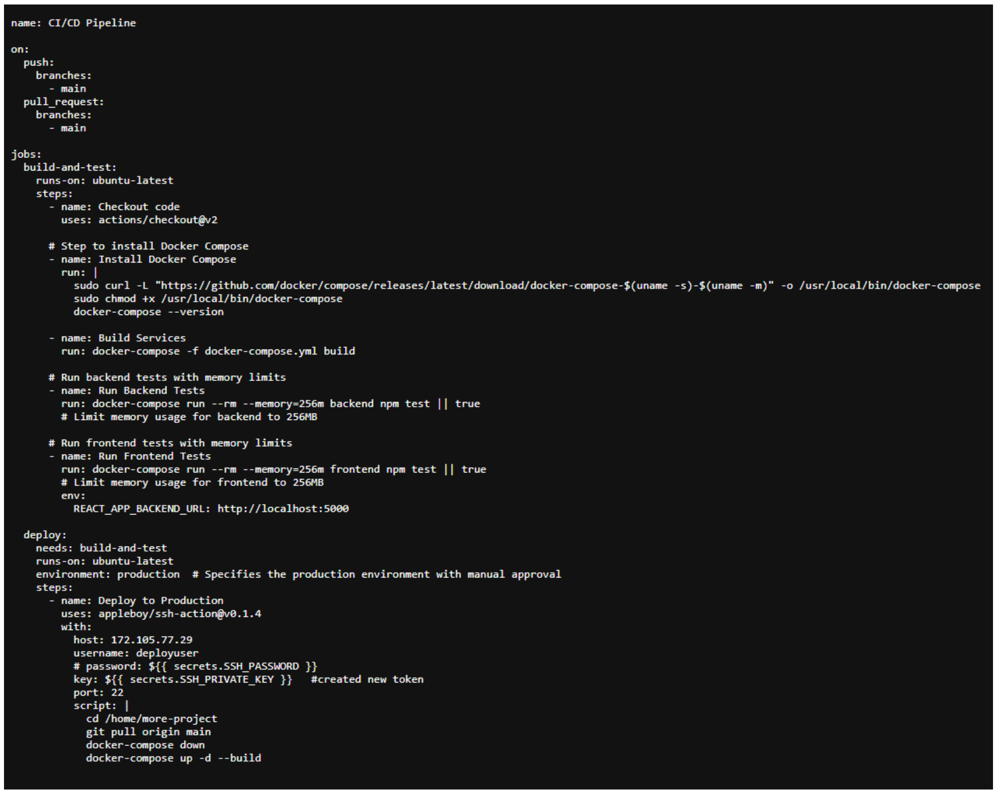
i. Backend test is in /home/more-project/backend/tests/app.test.js and ensures that the /api/message responds as expected.


```
// tests/app.test.js
const request = require('supertest');
const app = require('../index'); // Import the Express app

describe('GET /api/message', () => {
  it('should respond with status 200 and JSON message', async () => {
    const response = await request(app).get('/api/message');
    expect(response.statusCode).toBe(200);
    expect(response.body).toHaveProperty('message', 'Hello from the backend!');
  });
});
```

ii. Frontend test is in /home/more-project/frontend/src/App.test.js and ensures that the UI renders correctly and communicates with the backend.


```
// src/App.test.js
import React from 'react';
import { render, screen } from '@testing-library/react';
import App from './App';

test('renders frontend service and backend message', () => {
  render(<App />);
  const titleElement = screen.getByText(/Frontend Service/i);
  expect(titleElement).toBeInTheDocument();

  const messageElement = screen.getByText(/Message from Backend:/i);
  expect(messageElement).toBeInTheDocument();
});
```

- Manual approval before deployment to production.

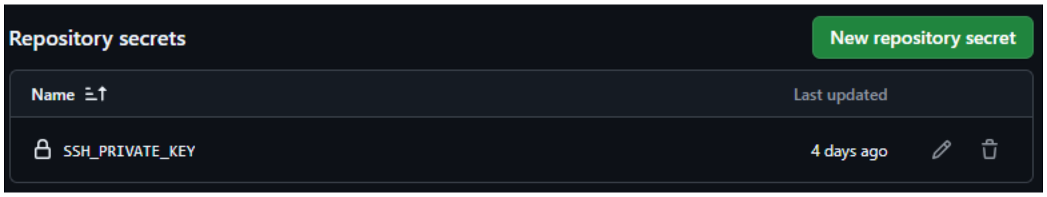


```yaml
name: CI/CD Pipeline

on:
  push:
    branches:
      - main
  pull_request:
    branches:
      - main

jobs:
  build-and-test:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v2

      # Step to install Docker Compose
      - name: Install Docker Compose
        run: |
          sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
          sudo chmod +x /usr/local/bin/docker-compose
          docker-compose --version

      - name: Build Services
        run: docker-compose -f docker-compose.yml build

      # Run backend tests with memory limits
      - name: Run Backend Tests
        run: docker-compose run --rm --memory=256m backend npm test || true
        # Limit memory usage for backend to 256MB

      # Run frontend tests with memory limits
      - name: Run Frontend Tests
        run: docker-compose run --rm --memory=256m frontend npm test || true
        # Limit memory usage for frontend to 256MB
        env:
          REACT_APP_BACKEND_URL: http://localhost:5000

  deploy:
    needs: build-and-test
    runs-on: ubuntu-latest
    environment: production  # Specifies the production environment with manual approval
    steps:
      - name: Deploy to Production
        uses: appleboy/ssh-action@v0.1.4
        with:
          host: 172.105.77.29
          username: deployuser
          # password: ${{ secrets.SSH_PASSWORD }}
          key: ${{ secrets.SSH_PRIVATE_KEY }}   #created new token
          port: 22
          script: |
            cd /home/more-project
            git pull origin main
            docker-compose down
            docker-compose up -d --build
```

- Deployment using appleboy/ssh-action.

3. Configure Secrets

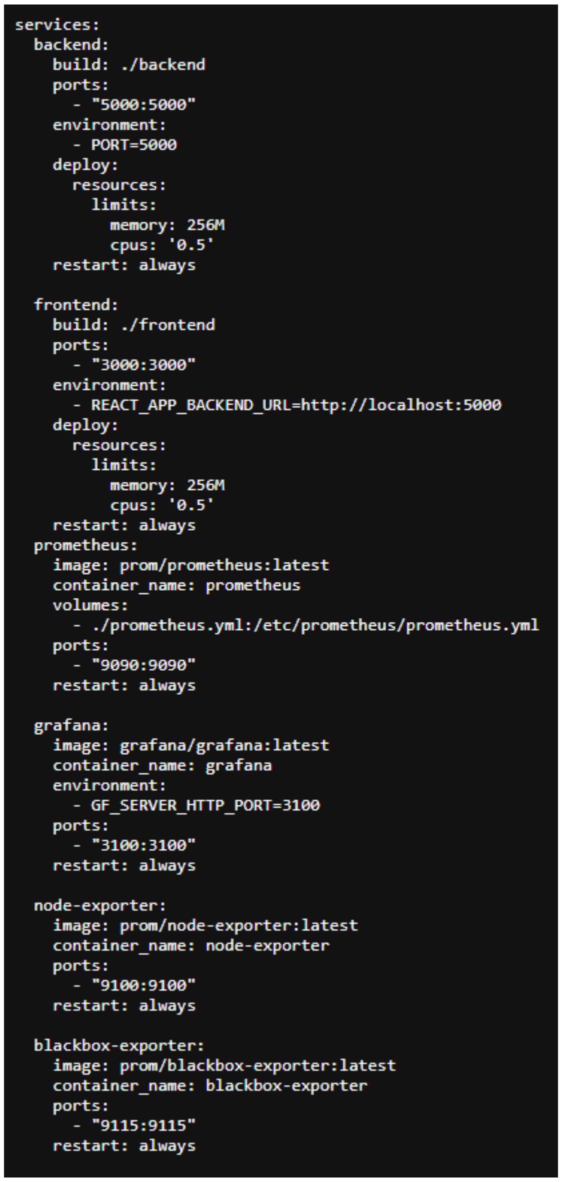
Added the SSH_PRIVATE_KEY as a secret in GitHub to authenticate with the Linode server.



Name	Last updated
SSH_PRIVATE_KEY	4 days ago

4. Implement the Application Deployment

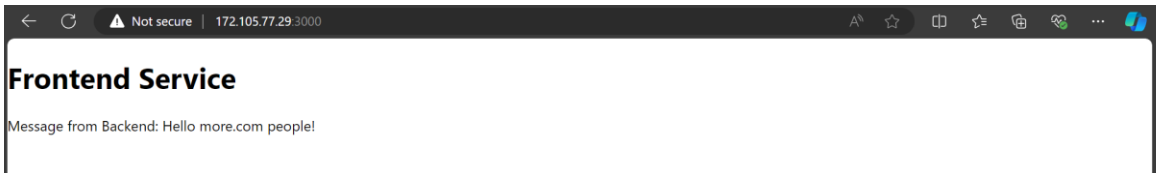
Docker Compose is used to manage the deployment:
- Stops running containers with `docker-compose down`.
- Rebuilds and starts containers with `docker-compose up -d --build`.



```yaml
services:
  backend:
    build: ./backend
    ports:
      - "5000:5000"
    environment:
      - PORT=5000
    deploy:
      resources:
        limits:
          memory: 256M
          cpus: '0.5'
    restart: always

  frontend:
    build: ./frontend
    ports:
      - "3000:3000"
    environment:
      - REACT_APP_BACKEND_URL=http://localhost:5000
    deploy:
      resources:
        limits:
          memory: 256M
          cpus: '0.5'
    restart: always
  prometheus:
    image: prom/prometheus:latest
    container_name: prometheus
    volumes:
      - ./prometheus.yml:/etc/prometheus/prometheus.yml
    ports:
      - "9090:9090"
    restart: always

  grafana:
    image: grafana/grafana:latest
    container_name: grafana
    environment:
      - GF_SERVER_HTTP_PORT=3100
    ports:
      - "3100:3100"
    restart: always

  node-exporter:
    image: prom/node-exporter:latest
    container_name: node-exporter
    ports:
      - "9100:9100"
    restart: always

  blackbox-exporter:
    image: prom/blackbox-exporter:latest
    container_name: blackbox-exporter
    ports:
      - "9115:9115"
    restart: always
```

5. Validation

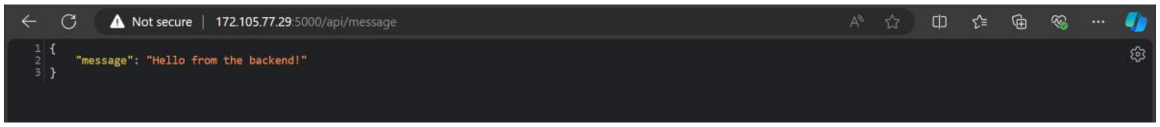
Accessed the deployed frontend at http://172.105.77.29:3000.



Frontend Service

Message from Backend: Hello more.com people!

Tested the backend API at http://172.105.77.29:5000/api/message.


Not secure | 172.105.77.29:5000/api/message

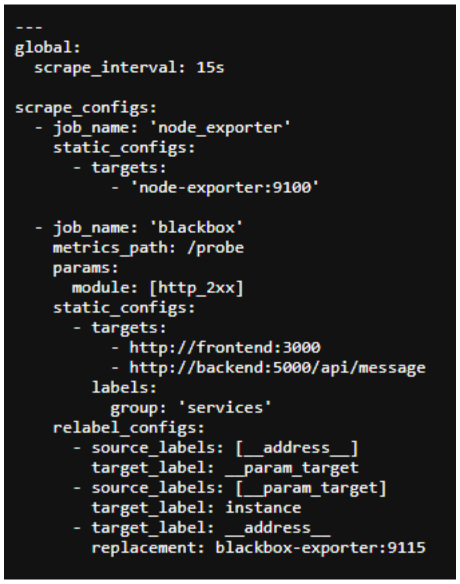
```
{
  "message": "Hello from the backend!"
}
```

Monitoring Setup

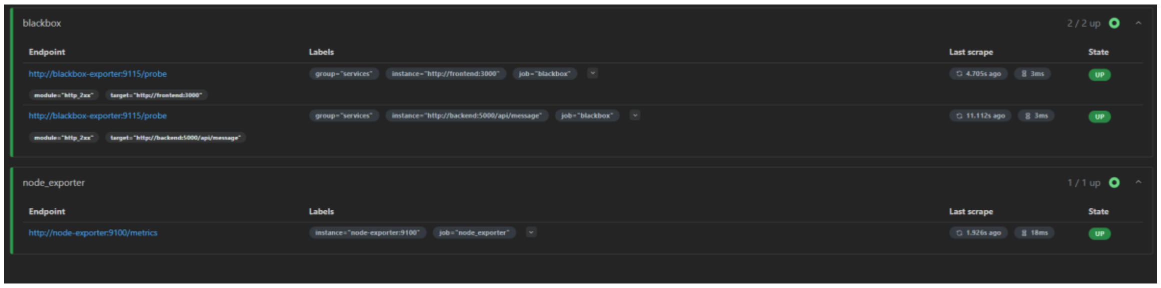
To ensure continuous application monitoring, Prometheus and Grafana were set up as part of the stack.

Prometheus Configuration

Prometheus collects metrics from Node Exporter and Blackbox Exporter. The configuration file `prometheus.yml` defines the targets to monitor.



```
---
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: 'node_exporter'
    static_configs:
      - targets:
          - 'node-exporter:9100'

  - job_name: 'blackbox'
    metrics_path: /probe
    params:
      module: [http_2xx]
    static_configs:
      - targets:
          - http://frontend:3000
          - http://backend:5000/api/message
        labels:
          group: 'services'
    relabel_configs:
      - source_labels: [__address__]
        target_label: __param_target
      - source_labels: [__param_target]
        target_label: instance
      - target_label: __address__
        replacement: blackbox-exporter:9115
```


blackbox 2 / 2 up

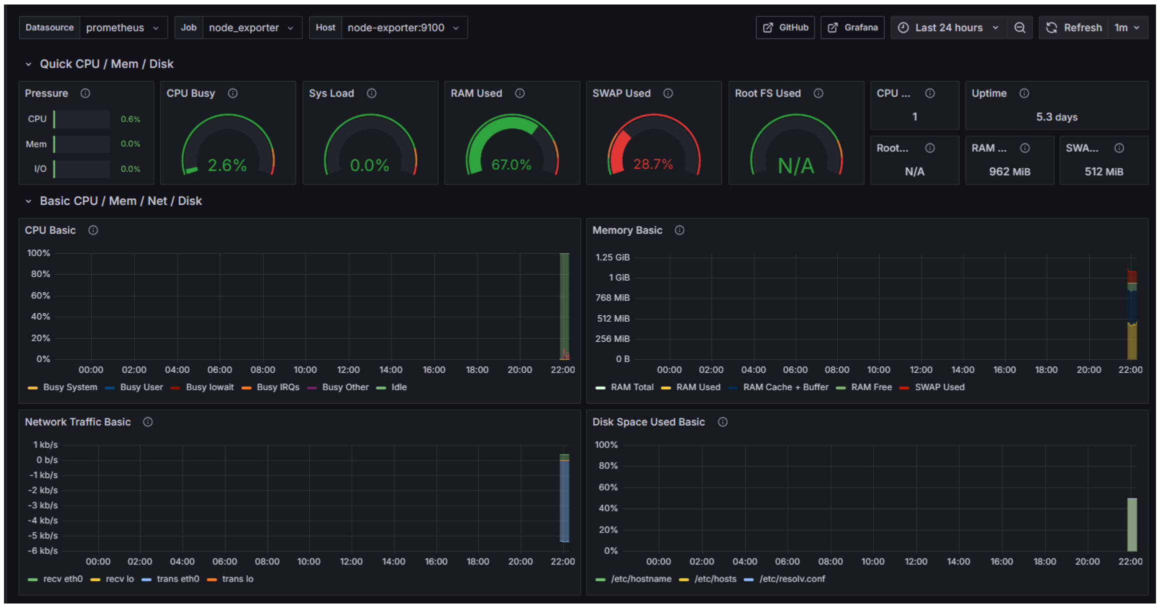
Endpoint	Labels	Last scrape	State
http://blackbox-exporter:9115/probe module="http_2xx" target="http://frontend:3000"	group="services" instance="http://frontend:3000" job="blackbox"	4.705s ago 3ms	UP
http://blackbox-exporter:9115/probe module="http_2xx" target="http://backend:5000/api/message"	group="services" instance="http://backend:5000/api/message" job="blackbox"	11.112s ago 3ms	UP

node_exporter 1 / 1 up

Endpoint	Labels	Last scrape	State
http://node-exporter:9100/metrics	instance="node-exporter:9100" job="node_exporter"	1.926s ago 18ms	UP

Grafana Configuration

Grafana visualizes the metrics collected by Prometheus. Two dashboards were set up:
- Node Exporter Dashboard (Dashboard ID: 1860): Displays server resource metrics.



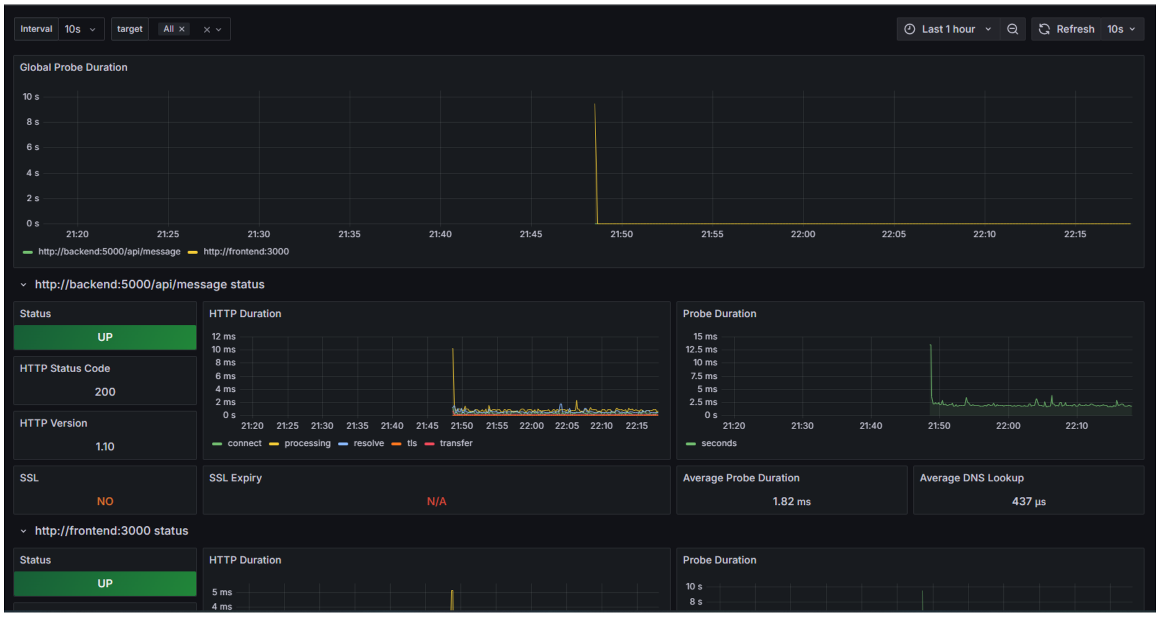
Datasource prometheus | Job node_exporter | Host node-exporter:9100 | Last 24 hours | Refresh 1m

Quick CPU / Mem / Disk

Panel	Value
Pressure CPU	0.6%
Pressure Mem	0.0%
Pressure I/O	0.0%
CPU Busy	2.6%
Sys Load	0.0%
RAM Used	67.0%
SWAP Used	28.7%
Root FS Used	N/A
CPU ...	1
Uptime	5.3 days
Root...	N/A
RAM ...	962 MiB
SWA...	512 MiB

Basic CPU / Mem / Net / Disk

- Blackbox Exporter Dashboard (Dashboard ID: 7587): Tracks uptime and response time for service endpoints.



Interval 10s | target All | Last 1 hour | Refresh 10s

Global Probe Duration

http://backend:5000/api/message status

Panel	Value
Status	UP
HTTP Status Code	200
HTTP Version	1.10
SSL	NO
SSL Expiry	N/A
Average Probe Duration	1.82 ms
Average DNS Lookup	437 µs

http://frontend:3000 status

Panel	Value
Status	UP