observability-demo

A minimal observability demo stack for a Go microservice. This repository boots a small mock Go service that emits structured logs, exposes Prometheus metrics, and publishes messages to RabbitMQ. The stack includes Prometheus, Loki + Promtail, Grafana, and RabbitMQ, orchestrated with Docker Compose.

> This README assumes you've cloned the repo and are running **Docker Desktop** on Windows (or Docker CE on Linux). All commands use the modern `docker compose` (space) CLI.

Overview

The `mock-service`: - serves HTTP endpoints (`/`, `/health`, `/metrics`) - logs structured JSON to stdout - exposes Prometheus metrics (request count + latency histogram) - publishes messages to RabbitMQ for each request

The stack includes: - **Prometheus** → collects metrics - **Loki + Promtail** → collects logs - **Grafana** → dashboards - **RabbitMQ** → message broker - **mock-service** → demo Go service

---

Prerequisites

- Docker Desktop (Windows) or Docker CE (Linux)
- Docker Compose v2 (bundled with Docker Desktop)
- Git installed

Verify Docker and Compose work:

```
docker --version
docker compose version
```

How to run

Clone the repo:

```
git clone https://github.com/you/observability-demo.git
cd observability-demo
```

Start the stack:

```
docker compose up -d --build
```

Check running containers:

```
docker compose ps
```

View logs of a container:

```
docker compose logs -f mock-service
```

Stop everything:

```
docker compose down
```

Stop and remove volumes (clean state):

```
docker compose down -v
```

Quick tests

- Test service response:

  ```
  curl http://localhost:8080/
  ```

- Health endpoint:

  ```
  curl http://localhost:8080/health
  ```

- Metrics endpoint:

  ```
  curl http://localhost:8080/metrics
  ```

- Generate load:

```
for ($i=0; $i -lt 20; $i++) { curl http://localhost:8080/ > $null; Start-Sleep -Milliseconds 200 }
```

Web UIs

- Grafana → http://localhost:3000 (user: `admin`, pass: `admin`)
- Prometheus → http://localhost:9090
- RabbitMQ management → http://localhost:15672 (guest/guest)
- Loki API → http://localhost:3100

Validation checklist

Follow these steps to confirm everything works end-to-end:

1. **Check service is reachable**
   Run:

   ```
   curl http://localhost:8080/
   ```

   You should see output like:

   ```
   hello from mock-service at 2025-08-31T10:15:30Z
   ```

2. **Check health endpoint**

   ```
   curl http://localhost:8080/health
   ```

   Should return `ok`.

3. **Check Prometheus metrics**

   ```
   curl http://localhost:8080/metrics | findstr http_requests_total
   ```

   After some requests, you should see metrics like:

   ```
   http_requests_total{method="GET",path="/",status="200"} 5
   ```

4. **Confirm Prometheus is scraping**
   Go to http://localhost:9090/targets. The `mock-service` target should show `UP`.

5. **Check logs in Grafana (via Loki)**

6. Open Grafana at http://localhost:3000
7. Login with `admin/admin`
8. Add a Loki datasource with URL `http://loki:3100`
9. Go to *Explore* → run query:

```
{service="mock-service"}
```

10. You should see structured logs like:

```
level=info msg="handled request" path=/ method=GET status=200
```

11. **Confirm RabbitMQ messages**

12. Go to http://localhost:15672 (guest/guest)
13. Check the `tasks` exchange → you should see messages published whenever `/` is requested.

Next improvements

- Add Grafana dashboards for request rates & latency
- Add Prometheus alerting rules
- Integrate distributed tracing

This README is meant as a quick start and validation guide. For deeper setup details, check the configuration files in the repo.