### ****Roteiro de Testes - Monitoramento e Observabilidade****

🔍 **Objetivo:** Validar se as ferramentas de monitoramento e observabilidade (Actuator, Prometheus, Grafana, Datadog) estão funcionando corretamente e exibindo os dados esperados.

## ****1️⃣ Verificar Status dos Containers****

📌 **Objetivo:** Certificar-se de que todos os serviços necessários estão em execução.

docker ps

✅ **Serviços esperados em execução:**  
✔ order-api (porta 8080)  
✔ rabbitmq (porta 5672)  
✔ mongodb (porta 27017)  
✔ redis (porta 6379)  
✔ prometheus (porta 9090)  
✔ grafana (porta 3000)  
✔ datadog-agent (se aplicável)

❌ **Se algum serviço não estiver rodando, reinicie todos os containers:**

docker-compose down && docker-compose up -d --build

## ****2️⃣ Acessar a API de Monitoramento via Actuator****

📌 **Objetivo:** Validar a exposição dos endpoints do Actuator.

### ****Verificar status geral da aplicação****

curl http://localhost:8080/actuator/health

✅ **Saída esperada:**

{"status":"UP"}

### ****Listar todas as métricas disponíveis****

curl http://localhost:8080/actuator/metrics

### ****Obter detalhes de uma métrica específica (exemplo: tempo de resposta)****

curl http://localhost:8080/actuator/metrics/http.server.requests

## ****3️⃣ Validar Exposição de Métricas no Prometheus****

📌 **Objetivo:** Garantir que as métricas estão sendo coletadas corretamente.

### ****Acessar o endpoint de métricas do Prometheus****

curl http://localhost:8080/actuator/prometheus

✅ **Se o retorno for grande, significa que os dados estão sendo expostos corretamente.**

📌 **Acesse o Prometheus via navegador:**  
🔗 **http://localhost:9090**

### ****Buscar algumas métricas importantes dentro do Prometheus****

1. **Contagem de requisições HTTP**

http\_server\_requests\_seconds\_count

1. **Tempo médio de resposta da API**

http\_server\_requests\_seconds\_sum / http\_server\_requests\_seconds\_count

1. **Uso de memória da aplicação**

nginx

CopiarEditar

jvm\_memory\_used\_bytes

✅ **Se as métricas estiverem aparecendo, significa que o Prometheus está coletando corretamente!**

## ****4️⃣ Visualizar os Dashboards no Grafana****

📌 **Objetivo:** Certificar-se de que as métricas do Prometheus estão sendo exibidas corretamente no Grafana.

📌 **Acesse o Grafana via navegador:**  
🔗 **http://localhost:3000**

### ****Passos para ver os dashboards no Grafana:****

1. **Fazer login** (usuário padrão: admin, senha: admin).
2. Ir em **"Configuration" > "Data Sources"**.
3. Verificar se a fonte de dados do Prometheus está adicionada (http://prometheus:9090).
4. Acessar a aba **"Dashboards"** e verificar gráficos como:
   * Requisições HTTP por segundo.
   * Tempo de resposta médio da API.
   * Uso de CPU e memória da aplicação.

✅ **Se os dados estiverem aparecendo corretamente, o Grafana está configurado!**

## ****5️⃣ Validar Logs Estruturados no Datadog (se configurado)****

📌 **Objetivo:** Certificar-se de que os logs estão sendo enviados corretamente ao Datadog.

### ****Verificar logs no Datadog diretamente pela API****

curl -X GET "https://api.datadoghq.com/api/v2/logs/events" \

-H "DD-API-KEY: <SUA\_CHAVE\_API>" \

-H "DD-APPLICATION-KEY: <SUA\_CHAVE\_APP>"

📌 **Acesse o Datadog via navegador:**  
🔗 **https://app.datadoghq.com/logs**

✅ **Se os logs da aplicação estiverem aparecendo na interface do Datadog, a configuração está correta!**

## ****6️⃣ Simular Alto Volume de Requisições para Testar Escalabilidade****

📌 **Objetivo:** Simular carga para analisar o comportamento da API e validar os gráficos de monitoramento.

### ****Executar um teste de carga usando Apache Benchmark****

ab -n 1000 -c 50 -T application/json -p request.json http://localhost:8080/orders

📌 request.json contém um exemplo de pedido JSON para simulação.

✅ **Após o teste, verificar novamente os gráficos no Grafana e as métricas no Prometheus!**

Esse roteiro permite validar o funcionamento do monitoramento da aplicação, garantindo que Actuator, Prometheus, Grafana e Datadog estão capturando corretamente as informações de métricas e logs.