DESAFIO TECNICO

DevOps Pleno

Observabilidade e Integracao Jimi IoT Gateway

Foco em Alta Disponibilidade e Monitoramento

Data: Fevereiro de 2025
Nivel: Pleno

Desafio tecnico para implementacao de gateway de recepcao de dados IoT com infraestrutura como codigo, redes seguras e telemetria completa.

Conteúdo

1 Visao Geral e Contexto **2**
1.1 Introducao . . . 2
1.2 Arquitetura de Alto Nivel . . . 2

2 Requisitos Tecnicos Detalhados **3**
2.1 1. Orquestracao e Redes (Docker Compose) . . . 3
2.1.1 1.1 Estrutura do docker-compose.yml . . . 3
2.1.2 1.2 Proxy Reverso: Nginx ou Traefik . . . 3
2.1.3 1.3 Seguranca . . . 3
2.2 2. Implementacao de Webhook e Logica . . . 4
2.2.1 2.1 Endpoints Obrigatorios . . . 4
2.2.2 2.2 Validacao de Payload . . . 4
2.3 3. Stack de Observabilidade . . . 5
2.3.1 3.1 Pilar 1: Metricas (Prometheus) . . . 5
2.3.2 3.2 Pilar 2: Logs (Loki) . . . 5

3 Tarefas do Candidato **6**
3.1 Tarefa 1: Configuracao de Infraestrutura . . . 6
3.2 Tarefa 2: Instrumentacao e Metricas . . . 6
3.3 Tarefa 3: Dashboard Grafana . . . 7
3.4 Tarefa 4: Documentacao de Troubleshooting . . . 7

4 Criterios de Avaliacao **9**
4.1 Rubrica Detalhada . . . 9
4.1.1 Infraestrutura (25 porcento) . . . 9
4.1.2 Seguranca (20 porcento) . . . 9
4.1.3 Observabilidade (30 porcento) . . . 9
4.1.4 Codigo (15 porcento) . . . 10
4.1.5 Documentacao (10 porcento) . . . 10

5 Perguntas Esperadas em Entrevista **11**
5.1 Tecnicas . . . 11
5.2 Comportamentais . . . 11

6 Recursos e Referencias **12**
6.1 Documentacao Oficial . . . 12
6.2 Ferramentas Recomendadas . . . 12

1. Visao Geral e Contexto

Introducao

Este desafio tecnico avalia a capacidade de um profissional DevOps Pleno em implementar uma solucao de integracao robusta para a plataforma Jimi Cloud, com foco especial em:

1. **Infraestrutura como Codigo (IaC)**: Provisionar e orquestrar conteineres
2. **Seguranca de Redes**: Implementar padroes de proxy reverso e isolamento
3. **Observabilidade**: Consolidar metricas, logs e healthchecks
4. **Confiabilidade**: Garantir alta disponibilidade e recuperacao

Arquitetura de Alto Nivel

Objetivo Principal: Provisionar um ambiente conteinerizado que receba webhooks da plataforma Jimi Cloud, utilize um Proxy Reverso para terminacao de trafego e implemente uma stack completa de Observabilidade para monitorar o estado da aplicacao e da infraestrutura.

2. Requisitos Tecnicos Detalhados

1. Orquestracao e Redes (Docker Compose)

2.1.1 1.1 Estrutura do docker-compose.yml

O arquivo `docker-compose.yml` deve cumprir os seguintes requisitos:

R1.1 Subir toda a stack de aplicacao e monitoramento com um unico comando

R1.2 Definir uma rede customizada em modo `bridge` para isolamento

R1.3 Nao expor a aplicacao backend diretamente na porta do host

R1.4 Configurar volumes persistentes para Prometheus, Grafana e Loki

R1.5 Incluir variaveis de ambiente no arquivo `.env`

2.1.2 1.2 Proxy Reverso: Nginx ou Traefik

Opcao A: Nginx

- Configurar escuta em portas 80 (HTTP) e 443 (HTTPS)
- Resolver dominio `api.jimi.local`
- Encaminhar trafego para o backend via nome do servico
- Implementar health checks antes de rotear requisicoes
- Configurar rate limiting para prevenir abuso

Opcao B: Traefik

- Definir rota com `Host(àpi.jimi.local)`
- Utilizar middleware de health check
- Configurar certificado TLS autoassinado
- Adicionar labels customizados para descoberta automatica

2.1.3 1.3 Seguranca

Requisitos de Seguranca Criticos:

- O servico de webhook **NAO** deve ser exposto diretamente
- Toda requisicao deve passar pelo proxy reverso
- Implementar validacao de origem das requisicoes
- Usar redes segregadas para aplicacao e monitoramento

2. Implementacao de Webhook e Logica

2.2.1 2.1 Endpoints Obrigatorios

A aplicacao backend deve implementar os seguintes endpoints:

Endpoint	Metodo	Descricao
`/v1/telemetry`	POST	Recebe dados de telemetria
`/v1/alarms`	POST	Recebe alertas e alarmes
`/v1/heartbeat`	POST	Recebe sinais de vida

Tabela 1: Endpoints Obrigatorios

2.2.2 2.2 Validacao de Payload

Cada endpoint deve:

1. Validar o payload JSON contra um schema predefinido
2. Retornar HTTP 400 se o payload for invalido
3. Retornar HTTP 200 com JSON {`"status": "ok"`} se bem-sucedido
4. Registrar (log) cada requisicao com timestamp e status

3. Stack de Observabilidade

Este e o diferencial de um profissional DevOps Pleno. Tres pilares obrigatorios:

2.3.1 3.1 Pilar 1: Metricas (Prometheus)

Requisitos:

M1 A aplicacao backend deve expor endpoint `/metrics` no formato Prometheus

M2 Coletar as seguintes metricas:

M3 Prometheus scrape a cada 15 segundos

M4 Retencao minima de 7 dias

M5 Alertas basicos (ex: taxa de erro ¿ 5 porcento)

2.3.2 3.2 Pilar 2: Logs (Loki)

Requisitos:

L1 Centralizar logs de:

- Nginx (access.log e error.log)
- Backend (stdout/stderr)

L2 Formato estruturado (JSON Lines)

L3 Cada log com: timestamp, nivel, mensagem, contexto

L4 Retencao minima de 7 dias

3. Tarefas do Candidato

Tarefa 1: Configuracao de Infraestrutura

Objetivo: Provisionar o ambiente completo.

Entregaveis:

1. `docker-compose.yml` funcional
2. `.env` com variaveis de configuracao
3. `README.md` com instrucoes de inicializacao
4. Diretorio `nginx/conf` com configuracao
5. Diretorio `certs/` com certificado TLS

Criterios de Aceitacao:

- Comando `docker-compose up -d` executa sem erros
- Todos os servicos alcancam status `healthy`
- Requisicao para `https://api.jimi.local/health` retorna 200 OK
- Nenhum servico exposto diretamente no host

```
# Verificar status dos conteineres
docker-compose ps

# Verificar logs
docker-compose logs -f backend

# Testar endpoint
curl -k https://api.jimi.local/health
```

Listing 1: Validar Inicializacao

Tarefa 2: Instrumentacao e Metricas

Objetivo: Implementar coleta de metricas.

Entregaveis:

1. Codigo da aplicacao com endpoint `/metrics`
2. Middleware para capturar latencia e taxa de erro
3. Arquivo `prometheus.yml` com configuracao
4. Alertas Prometheus basicos (`alerts.yml`)

Metricas Obrigatorias:

```
jimi_webhooks_received_total{endpoint="telemetry"} 1234
jimi_webhooks_received_total{endpoint="alarms"} 456
jimi_request_latency_ms_bucket{endpoint="telemetry"} 1200
jimi_http_errors_total{endpoint="telemetry",status="500"} 2
```

Listing 2: Exemplo de Metricas Esperadas

Tarefa 3: Dashboard Grafana

Objetivo: Criar dashboard visual.
Entregaveis:

1. Arquivo JSON do dashboard Grafana
2. Nome: `grafana-dashboard-jimi-iot.json`
3. Conteudo minimo:
   - Taxa de webhooks por tipo (grafico temporal)
   - Taxa de erro HTTP (gauge)
   - Latencia P95 (grafico de linhas)
   - Saude dos conteineres (status indicator)
   - Ultimos webhooks (tabela)

Instrucoes para Exportar:

1. Acessar `http://localhost:3000` (Grafana)
2. Login: `admin` / `admin`
3. Clique em Dashboard Settings
4. Export → Export for Sharing
5. Salvar JSON

Tarefa 4: Documentacao de Troubleshooting

Objetivo: Guia para diagnosticar gargalos.
Cenario: *Os dados da Jimi Cloud comecam a atrasar. Como identificar e resolver?*
Estrutura Esperada:

1. **Checklist de Diagnostico**
   - Verificar saude dos conteineres
   - Monitorar CPU e memoria
   - Analisar taxa de erro HTTP
   - Verificar latencia de rede

2. **Metricas-Chave**
    - Requisicoes por segundo (RPS)
    - Latencia P95/P99
    - Taxa de erro HTTP 5xx
    - Tamanho da fila de processamento
3. **Estrategias de Resolucao**
    - Aumentar recursos (CPU/RAM)
    - Implementar circuit breaker
    - Cache no Nginx
    - Rate limiting
4. **Comandos de Troubleshooting**
    - Consultar logs
    - Executar queries Prometheus
    - Analisar healthchecks

4. Criterios de Avaliacao

Criterio	Peso	Descricao
Infraestrutura	25%	Docker Compose, Networks, Volumes
Seguranca	20%	Isolamento, TLS, Validacao
Observabilidade	30%	Metricas, Logs, Healthchecks
Codigo	15%	Qualidade, Tratamento de Erros
Documentacao	10%	README, Troubleshooting

Tabela 2: Matriz de Avaliacao

Rubrica Detalhada

4.1.1 Infraestrutura (25 porcento)

1. **100 porcento**
   - Stack completo sobe com um comando
   - Redes segregadas corretamente
   - Volumes persistentes para dados
   - Variaveis de ambiente em .env
2. **75 porcento**
   - Stack sobe mas com warnings
   - Redes nao estao bem segregadas
3. **50 porcento**
   - Stack sobe com erros menores
   - Requer ajustes manuais

4.1.2 Seguranca (20 porcento)

1. **100 porcento**: Backend nao exposto, TLS ok, validacao rigorosa
2. **75 porcento**: Backend nao exposto, TLS ok, validacao basica
3. **50 porcento**: Algumas exposicoes, TLS ausente
4. **25 porcento**: Multiplas vulnerabilidades

4.1.3 Observabilidade (30 porcento)

1. **100 porcento**: 3 pilares completos
2. **75 porcento**: 2 pilares bem implementados
3. **50 porcento**: Implementacao parcial
4. **25 porcento**: Minimo ou incompleto

4.1.4 Codigo (15 porcento)

1. **100 porcento**: Clean Code, tratamento de excecoes
2. **75 porcento**: Codigo funcional com melhorias
3. **50 porcento**: Funcional mas desorganizado
4. **25 porcento**: Com bugs ou problemas

4.1.5 Documentacao (10 porcento)

1. **100 porcento**: README completo, troubleshooting detalhado
2. **75 porcento**: README ok, troubleshooting basico
3. **50 porcento**: Documentacao minima
4. **25 porcento**: Pouca documentacao

5. Perguntas Esperadas em Entrevista

Tecnicas

1. *Por que usar um proxy reverso em vez de expor a aplicacao diretamente?*
   - Isolamento de seguranca
   - Terminacao de TLS centralizada
   - Load balancing
   - Rate limiting
   - Logging de trafego
2. *Como voce diferencia entre metricas, logs e traces?*
   - Metricas: valores agregados (RPS, latencia)
   - Logs: eventos pontuais com contexto
   - Traces: fluxo atraves do sistema
3. *O que fazer se o backend ficar saturado?*
   - Aumentar recursos (vertical scaling)
   - Adicionar replicas (horizontal scaling)
   - Implementar circuit breaker
   - Cache no proxy

Comportamentais

1. *Descreva um incidente de producao que voce resolveu*
2. *Como voce documenta infraestrutura complexa?*
   - Diagramas de arquitetura
   - Runbooks de operacao
   - Comentarios no codigo
   - Versionamento de configuracoes

6. Recursos e Referencias

Documentacao Oficial

1. **Docker Compose:** https://docs.docker.com/compose/
2. **Nginx:** https://nginx.org/en/docs/
3. **Prometheus:** https://prometheus.io/docs/
4. **Grafana:** https://grafana.com/docs/
5. **Loki:** https://grafana.com/docs/loki/latest/

Ferramentas Recomendadas

Ferramenta	Proposito	Alternativa
Docker Compose	Orquestracao	Kubernetes
Nginx	Proxy Reverso	Traefik, HAProxy
Prometheus	Metricas	InfluxDB
Grafana	Visualizacao	Kibana
Loki	Logs	ELK Stack
FastAPI	Backend	Flask, Django

Tabela 3: Ferramentas e Alternativas

Conclusao

Este desafio tecnico e uma excelente oportunidade para demonstrar dominio em:

- **Infraestrutura como Codigo**: Provisionar ambientes reproduziveis
- **Seguranca em Redes**: Implementar isolamento e criptografia
- **Observabilidade Avancada**: 3 pilares (metricas, logs, traces)
- **Operacao de Sistemas**: Monitoramento e troubleshooting
- **Comunicacao Tecnica**: Documentacao e runbooks

Um candidato que completa este desafio com excelencia demonstra estar pronto para:

- Liderar projetos de infraestrutura
- Mentorizar junior engineers
- Resolver incidentes complexos
- Implementar padroes de alta disponibilidade

Boa sorte! >>>