MONITORAMENTO E OBSERVABILIDADE EM CLUSTER KUBERNETES

Trabalho final - PSPD 2025.2

GRUPO

Guilherme Westphall
211061805

Lucas Martins
221022088

REVISÃO GRPC

Serviços

Aplicação bancária

P

- Stub
- Camada de tradução: JSON → Protobuf
- Recebe as requisições e redireciona para A e B
- Python

A

- Serviço do cliente
- CRUD de clientes
- Comunicação com o Postgres via Prisma (ORM)
- Node.js

B

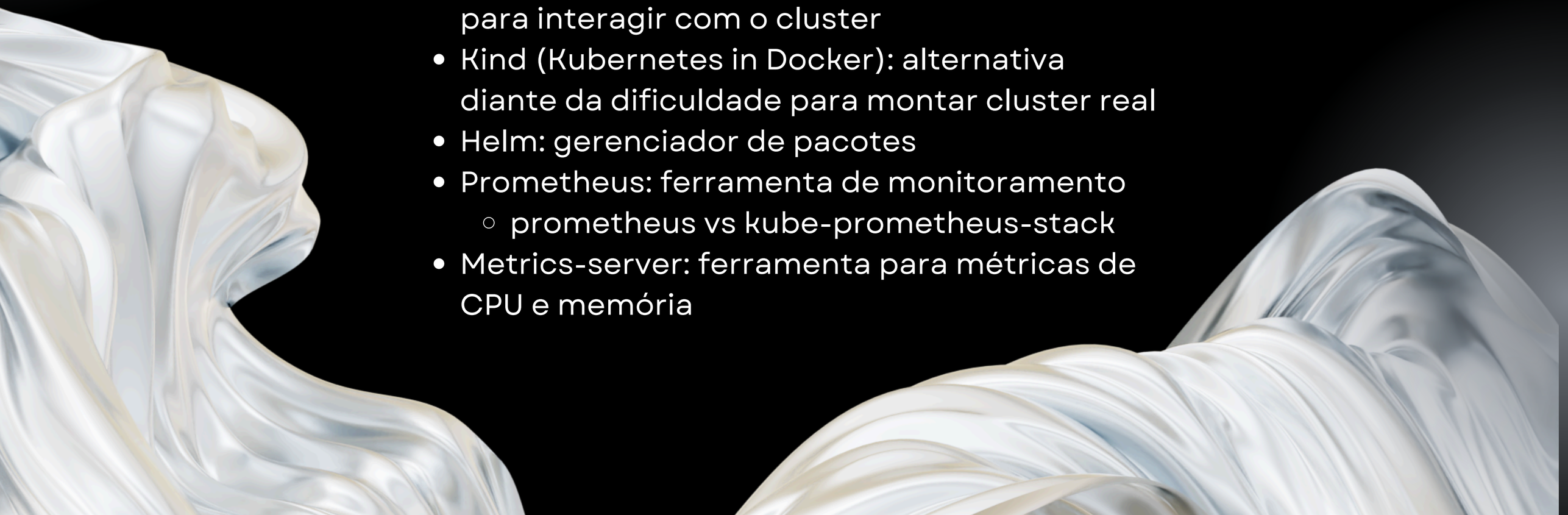
- Serviço de contas
- CRUD de contas e transações
- Comunicação com o Postgres via Prisma (ORM)
- Node.js

OBJETIVOS

Montagem do cluster multinode

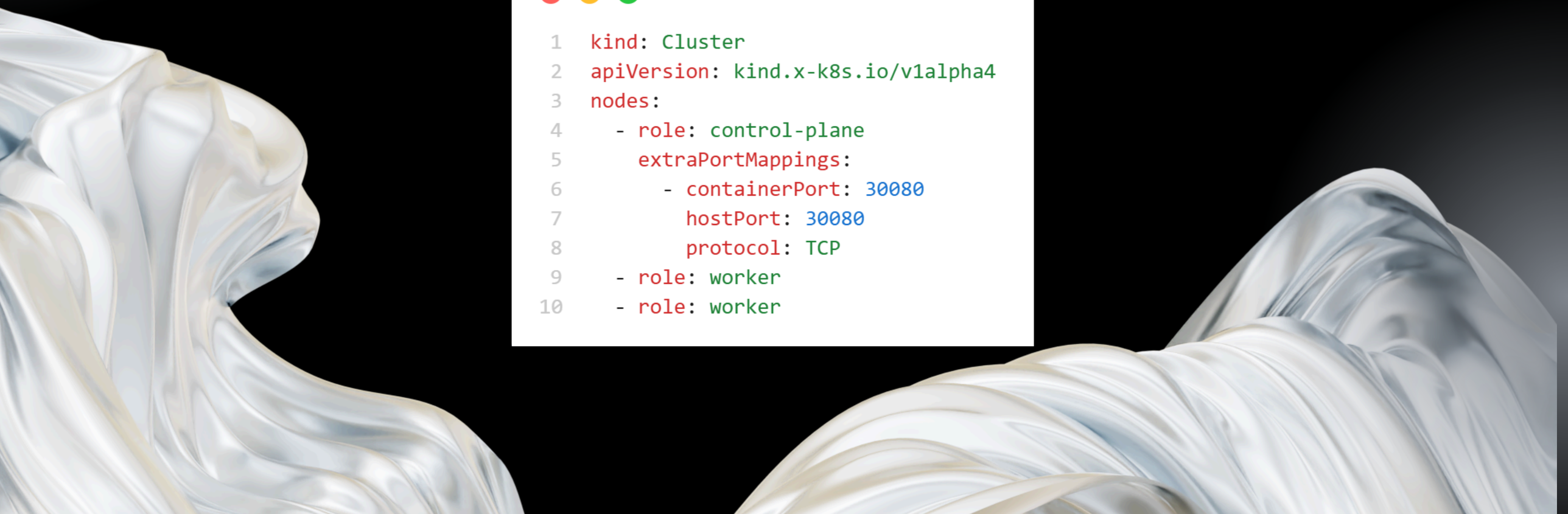
- 1 Master + 2 Workers
- 4 Serviços (Stub, Account, Client, Postgres)
- Avaliar o impacto de réplicas e workers
  - Throughput médio
  - Latência (média e P95)
  - Taxa de erros

AMBIENTE E DEPLOY



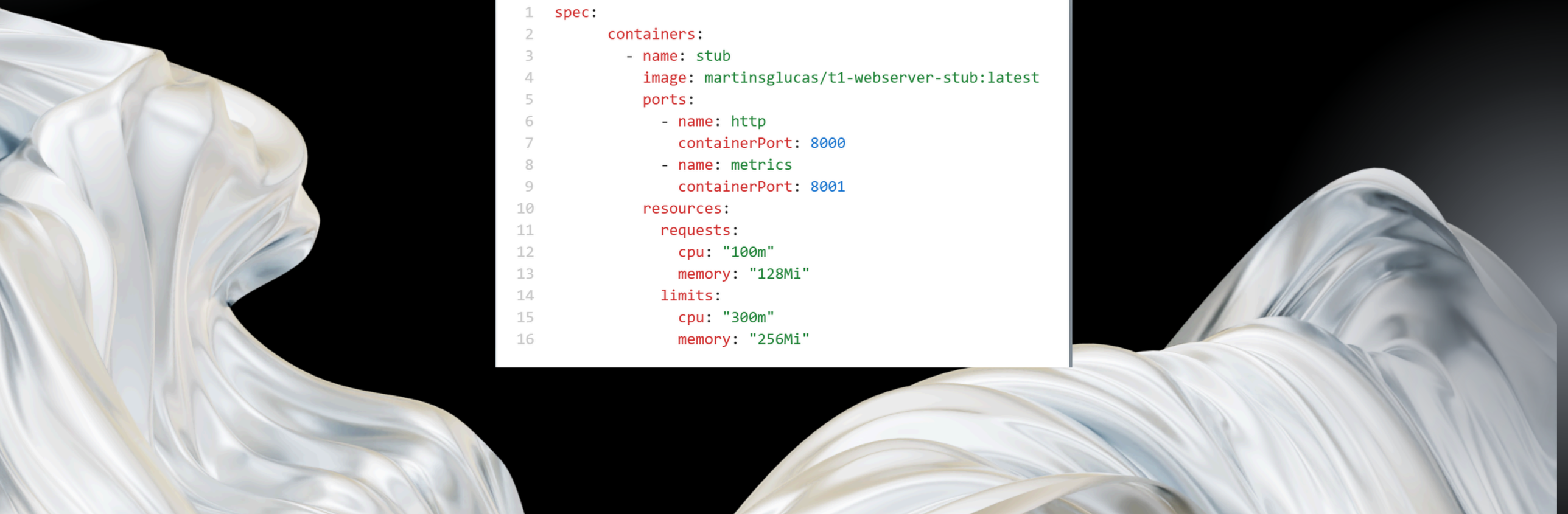
- Kubectl: ferramenta de linha de comando para interagir com o cluster
- Kind (Kubernetes in Docker): alternativa diante da dificuldade para montar cluster real
- Helm: gerenciador de pacotes
- Prometheus: ferramenta de monitoramento
  - prometheus vs kube-prometheus-stack
- Metrics-server: ferramenta para métricas de CPU e memória

AMBIENTE E DEPLOY



```yaml
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
nodes:
  - role: control-plane
    extraPortMappings:
      - containerPort: 30080
        hostPort: 30080
        protocol: TCP
  - role: worker
  - role: worker
```

MODIFICAÇÃO DOS MANIFESTS



```yaml
spec:
      containers:
        - name: stub
          image: martinsglucas/t1-webserver-stub:latest
          ports:
            - name: http
              containerPort: 8000
            - name: metrics
              containerPort: 8001
          resources:
            requests:
              cpu: "100m"
              memory: "128Mi"
            limits:
              cpu: "300m"
              memory: "256Mi"
```

CRIAÇÃO YAMLS HPA

```yaml
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: stub-hpa-cpu
  namespace: t3
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: stub
  minReplicas: 1
  maxReplicas: 3
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50

```

TESTES

Cenários 3/6

Baseline

- Configuração: 1 master, 2 workers; stub, client, account e postgres com 1 réplica cada.
- Objetivo: estabelecer um cenário de referência funcional, com carga estável e sem otimizações de escalonamento.

Réplica Stub

- Configuração: 1 master, 2 workers; **3 réplicas do stub**, demais serviços com 1 réplica.
- Objetivo: verificar o impacto de escalar apenas a borda HTTP (API Gateway) na capacidade de atendimento

Réplicas Serviços

- Configuração: 1 master, 2 workers; **3 réplicas para stub, client e account, postgres com 1 réplica**.
- Objetivo: verificar o impacto de escalar apenas a borda HTTP (API Gateway) na capacidade de atendimento

TESTES

Cenários 6/6

4 Workers

- Configuração: 1 master, 4 workers; stub, client, account e postgres com 1 réplica cada.
- Objetivo: avaliar se apenas aumentar o número de nós do cluster, sem mudar réplicas, traz benefício de desempenho.

4 Workers + 3rep

- Configuração: 1 master, 4 workers; 3 réplicas para stub, client e account, postgres com 1 réplica.
- Objetivo: combinar mais nós e mais réplicas de aplicação para buscar um cenário de maior throughput mantendo estabilidade.

Autoscaling (HPA)

- Configuração: 1 master, 2 workers; HPA configurado para stub, client e account, permitindo variar automaticamente o número de réplicas (1–N) conforme uso de CPU.
- Objetivo: avaliar se o escalonamento automático do Kubernetes consegue reagir à carga de forma mais eficiente que as configurações manuais de réplicas.

TESTES

RESULTADOS

Cenário	Workers / Réplicas	Throughput (req/s)	Latência	p95	http_req_failed	checks_failed
Baseline	2w - 1x stub, 1x account, 1x client	301	231 ms	711 ms	0%	0%
Replica Stub	2w - 3x stub, 1x account, 1x client	670	48 ms	133 ms	10,14%	12,62%
Replica 3xServiços	2w - 3x stub, 3x account, 3x client	670	48 ms	142 ms	10,14%	≈12%
4 Workers	4w - 1x stub, 1x account, 1x client	199	396 ms	1,37 s	0%	0%
4 Workers + 3xServiços	4w - 3x stub, 3x account, 3x client	333,77	195,92 ms	611,58 ms	0%	0%
Autoscaling (HPA Stub)	2w - HPA stub (1–N), 1x account, 1x client	125,31	770 ms	2,92 s	1,47%	1,52%

PROMETHEUS

- O stub-service expõe as métricas através de um endpoint `/metrics`

QUERIES

THROUGHPUT (req/s)

```
sum(
  rate(http_requests_total{
    namespace="t3",
    service="stub-service",
  }[15m])
)
```

THROUGHPUT p/ endpoint (req/s)

```
sum by (endpoint)(
  rate(http_requests_total{
    namespace="t3",
    service="stub-service"
    }[15m])
)
```

Latência (ms)

```
sum(
  rate(http_request_duration_seconds_sum{
    namespace="t3",
    service="stub-service"
  }[15m])
)
/
sum(
  rate(http_request_duration_seconds_count{
    namespace="t3",
    service="stub-service"
  }[15m])
)
```

PROMETHEUS

QUERIES

P95 (ms)

```
histogram_quantile(
  0.95,
  sum by (le)(
    rate(http_request_duration_seconds_bucket{
      namespace="t3",
      service="stub-service"
    }[15m])
  )
)
```

CPU (cores de CPU)

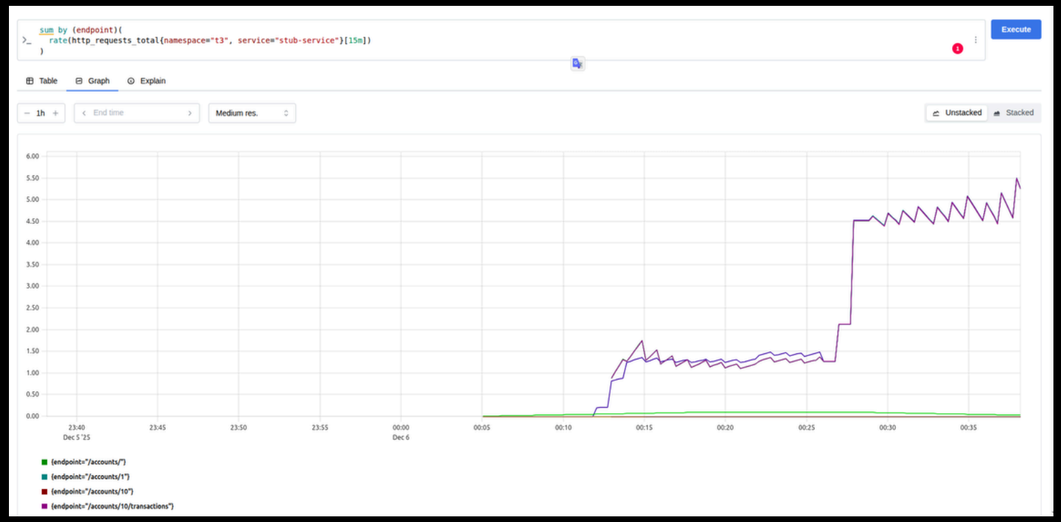
```
sum by (pod)(
  rate(container_cpu_usage_seconds_total{
    namespace="t3",
    pod=~"stub-.*"
  }[5m])
)
```

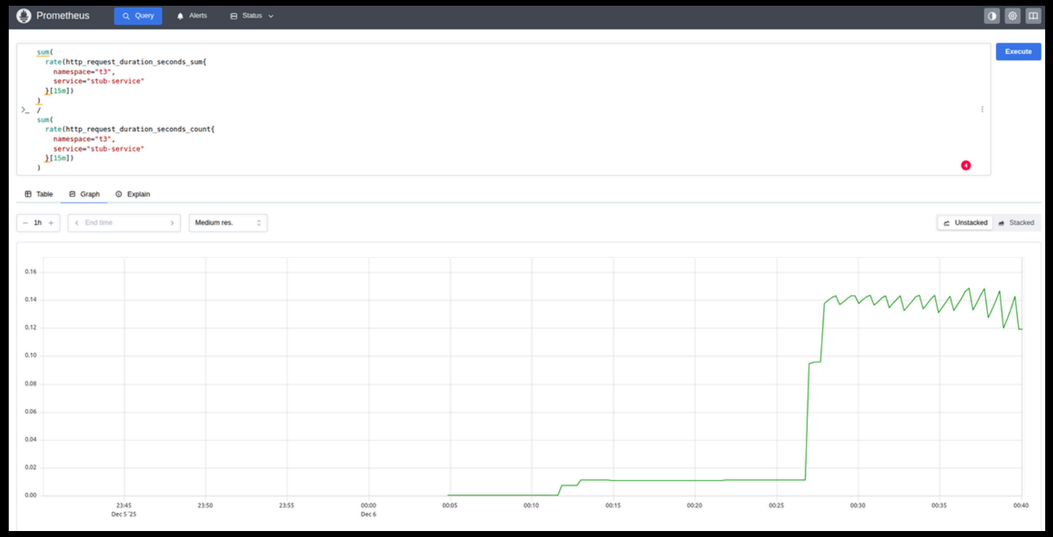
Memória (bytes)

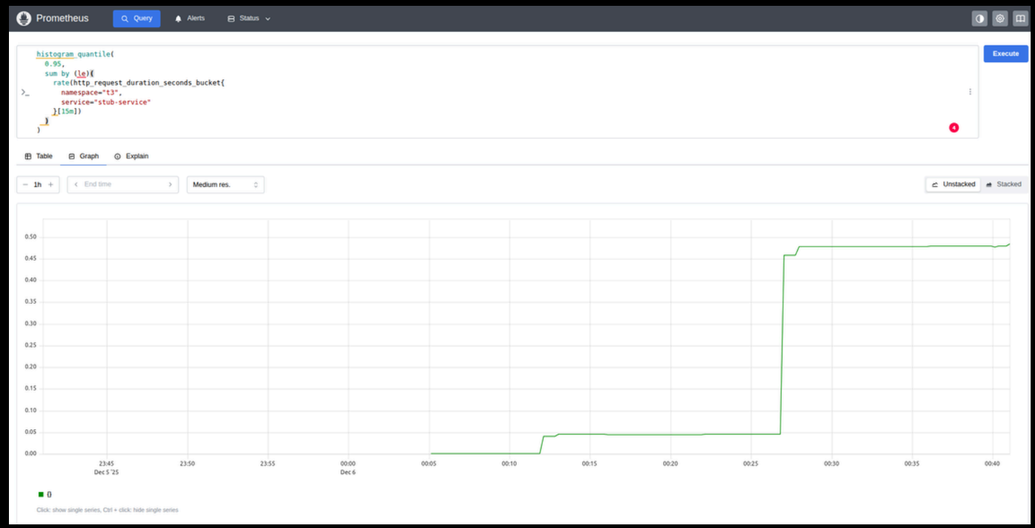
```
sum by (pod)(
  container_memory_usage_bytes{
    namespace="t3",
    pod=~"stub-.*"
  }
)
```

PROMETHEUS

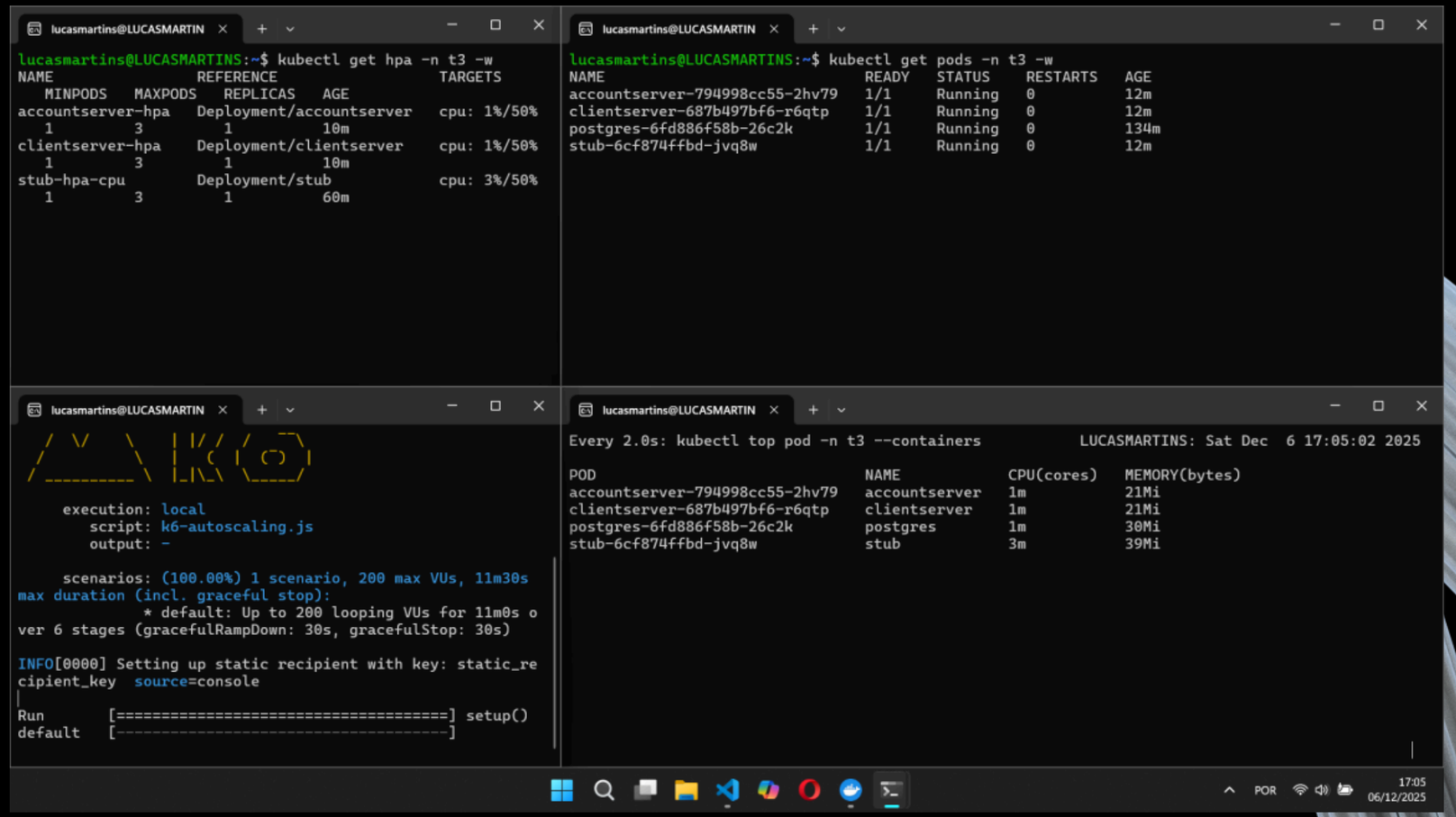

Prometheus
Query
Alerts
Status
sum(
rate(http_requests_total{namespace="t3", service="stub-service"}[10m])
)
Execute
Table
Graph
Explain
1h
End time
Medium res.
Unstacked
Stacked
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.00
0.00
23:40
Dec 5 '25
23:45
23:50
23:55
00:00
Dec 6
00:05
00:10
00:15
00:20
00:25
00:30
00:35
{}
Click: show single series, Ctrl + click: hide single series


sum by (endpoint)(
rate(http_requests_total{namespace="t3", service="stub-service"}[15m])
)
Execute
Table
Graph
Explain
1h
End time
Medium res.
Unstacked
Stacked
6.00
5.50
5.00
4.50
4.00
3.50
3.00
2.50
2.00
1.50
1.00
0.50
0.00
23:40
Dec 5 '25
23:45
23:50
23:55
00:00
Dec 6
00:05
00:10
00:15
00:20
00:25
00:30
00:35
{endpoint="/accounts/"}
{endpoint="/accounts/1"}
{endpoint="/accounts/10"}
{endpoint="/accounts/10/transactions"}

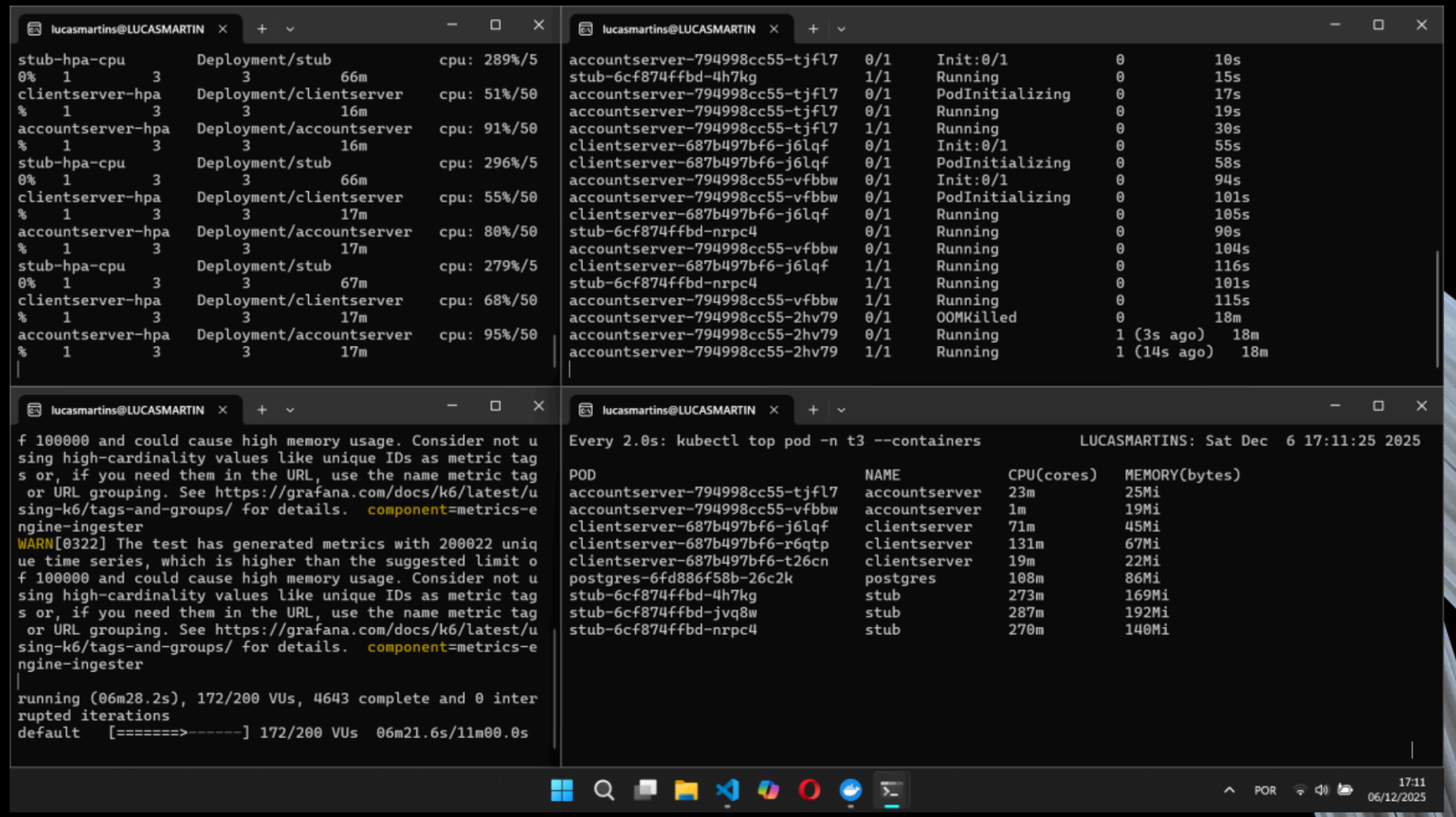

Prometheus
Query
Alerts
Status
sum(
rate(http_request_duration_seconds_sum{
namespace="t3",
service="stub-service"
}[15m])
)
/
sum(
rate(http_request_duration_seconds_count{
namespace="t3",
service="stub-service"
}[15m])
)
Execute
Table
Graph
Explain
1h
End time
Medium res.
Unstacked
Stacked
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
23:45
Dec 5 '25
23:50
23:55
00:00
Dec 6
00:05
00:10
00:15
00:20
00:25
00:30
00:35
00:40


Prometheus
Query
Alerts
Status
histogram_quantile(
0.95,
sum by (le)(
rate(http_request_duration_seconds_bucket{
namespace="t3",
service="stub-service"
}[15m])
)
)
Execute
Table
Graph
Explain
1h
End time
Medium res.
Unstacked
Stacked
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
23:45
Dec 5 '25
23:50
23:55
00:00
Dec 6
00:05
00:10
00:15
00:20
00:25
00:30
00:35
00:40
0
Click: show single series, Ctrl + click: hide single series

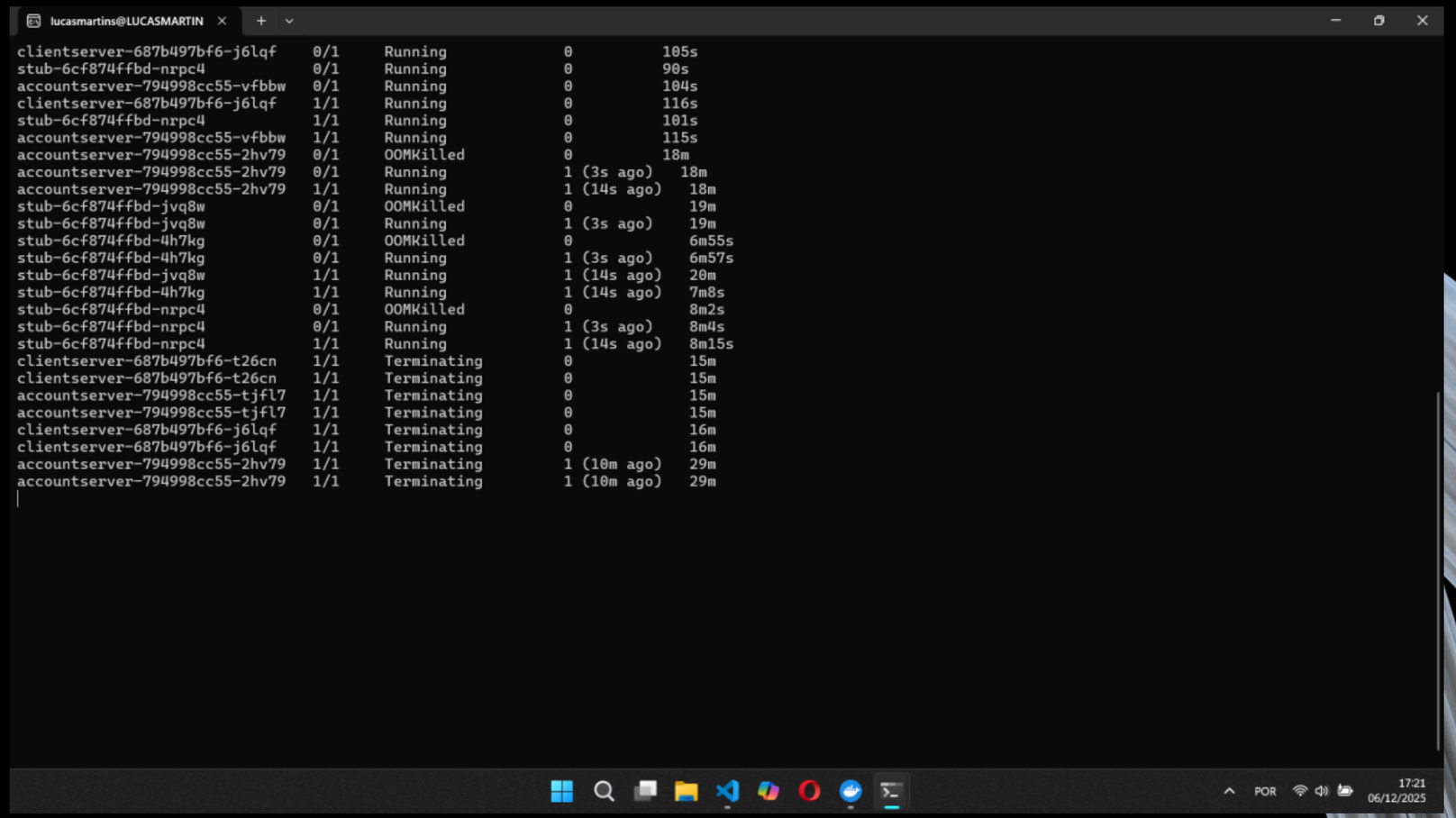
MONITORAMENTO HPA



MONITORAMENTO HPA



MONITORAMENTO HPA



DIFICULDADES

LIMITAÇÃO DO AMBIENTE
(1 HOST)

TEMPO LONGO DE BUILD

EXECUÇÃO PESADA
(CPU/RAM)

TRAVAMENTO E
REINICIALIZAÇÃO

OBRIGADO