WEBAPP COM DOCKER SWARM

MEGSI - G3TP2

2025/2026

AGENDA

Introdução e Objetivos

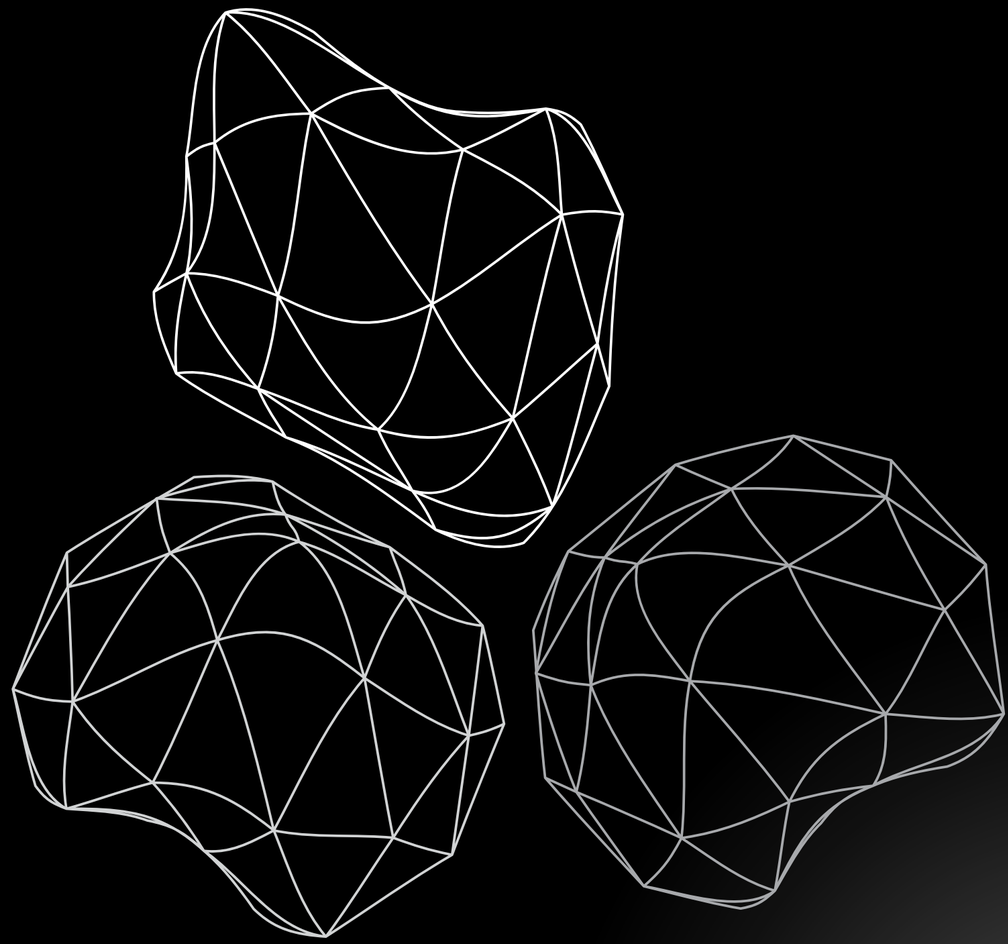
Evolução do Projeto (4 fases)

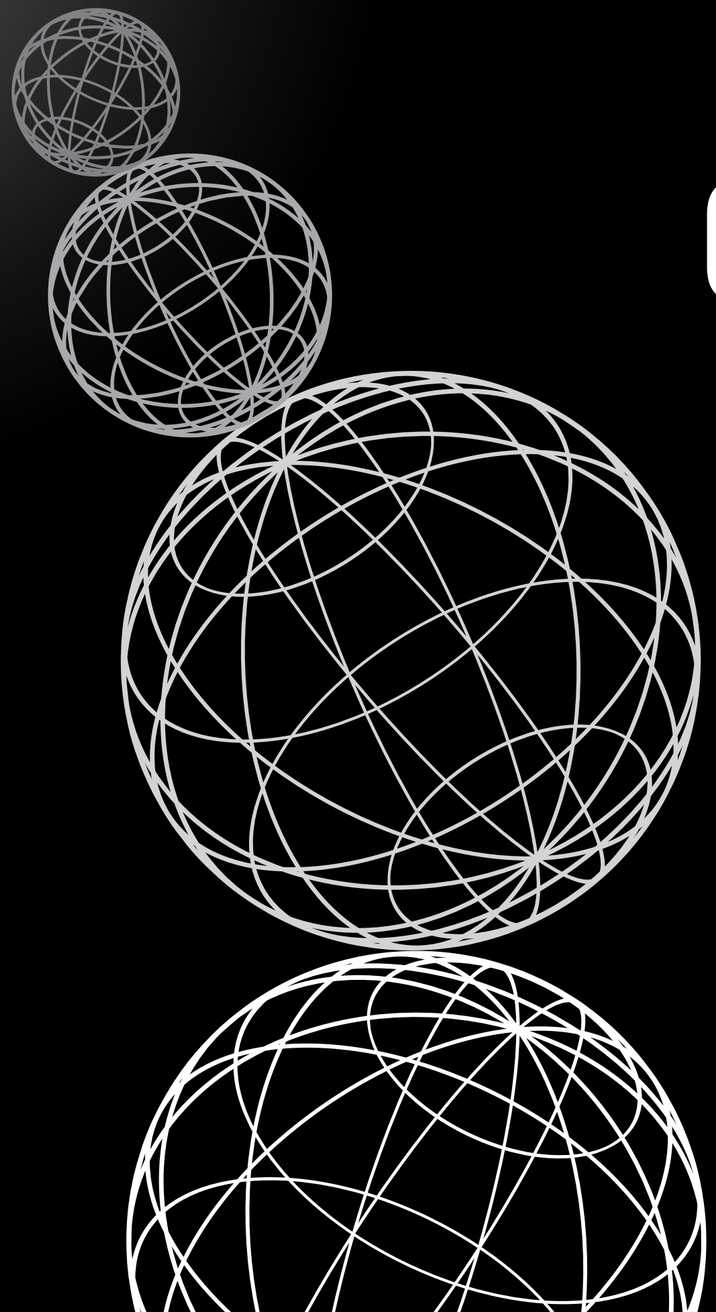
Arquitetura Final

Tecnologias e Implementação

Demonstração de Resultados

Conclusões





OBJETIVOS DO PROJETO

Desenvolver, gradualmente, um sistema web escalável e resiliente.

Alta Disponibilidade - Zero downtime

Load Balancing - Distribuição automática de carga

Container Orchestration - Gestão automatizada

Monitoring Completo - Visibilidade total do sistema

Auto-Recovery - Recuperação automática de falhas

EVOLUÇÃO DO PROJETO

Fase 1: Sistema Monolítico

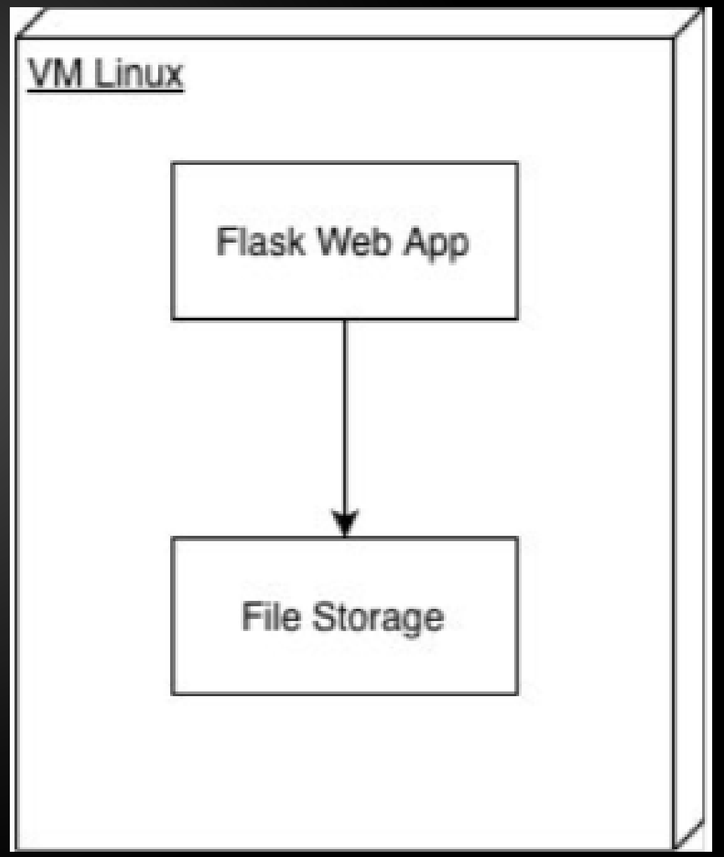
Fase 2: Separação de Serviços

Fase 3: Containerização

Fase 4: Orquestração e Monitorização

FASE 1: SISTEMA MONOLÍTICO

Single Server – Tudo numa única VM

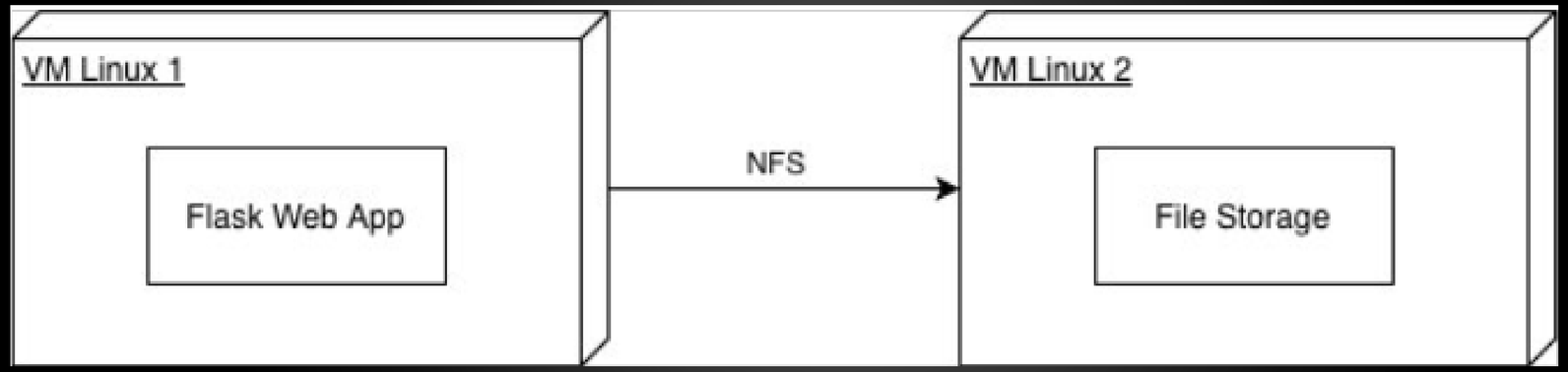


Limitações:

- Single point of failure
- Recursos partilhados (CPU, RAM)
- Escalabilidade limitada
- Sem separação de responsabilidades

FASE 2: SEPARAÇÃO DE SERVIÇOS

Dois Servidores em VMs Linux



Melhorias:

Separação de responsabilidades

Melhor isolamento

Storage dedicado

Ainda com limitações:

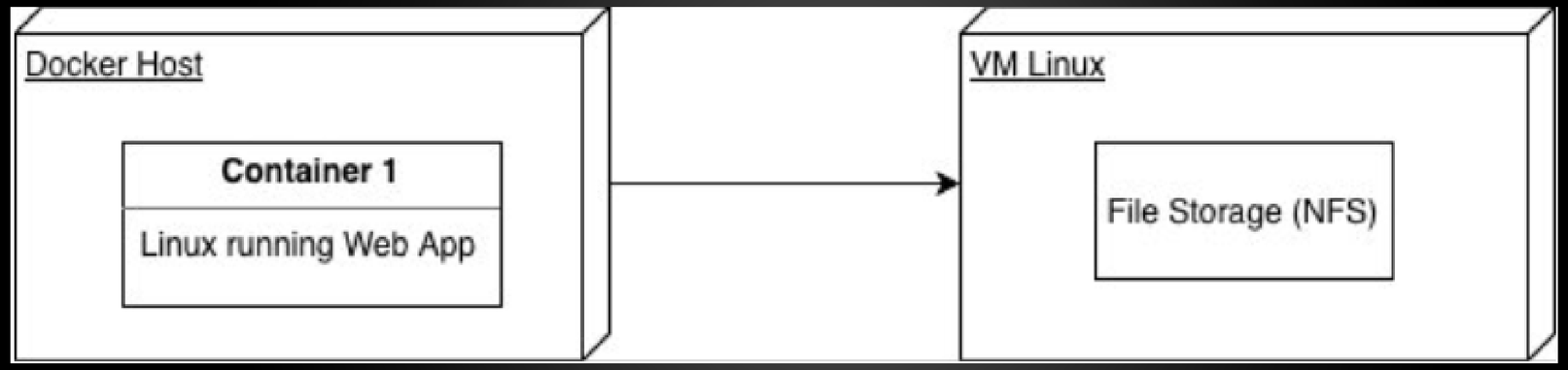
Sem redundância

Downtime em updates

Sem load balancing

FASE 3: CONTAINERIZAÇÃO

WebApp em Container + VM Storage



Melhorias:

Portabilidade

Isolamento melhorado

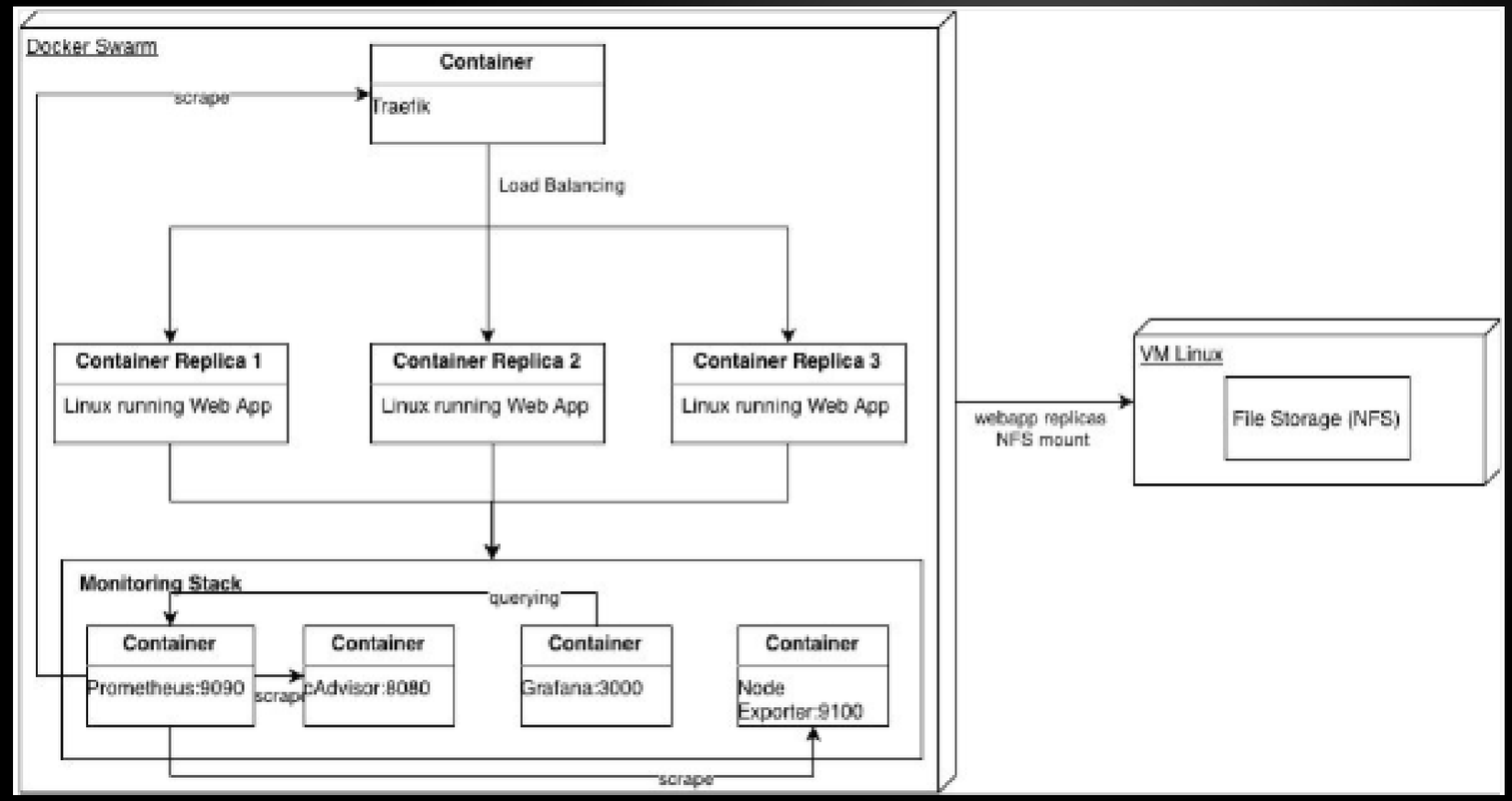
Deploy mais rápido com Dockerfilee Docker Compose

Próximo desafio:

Alta disponibilidade e escalabilidade

FASE 4: ORQUESTRAÇÃO E MONITORING

Docker Swarm + Load Balancing + Monitoring



Implementado:

Docker Swarm

Traefik Load Balancer

Stack de Monitoring completa

RESUMO DA EVOLUÇÃO

Fase	Descrição	Beneficio Chave
1	Monolítico (1 VM)	Simplicidade
2	Separação (2 VMs)	Isolamento
3	Containerização	Portabilidade
4	Orquestração	Alta Disponibilidade

STACK DE SERVIÇOS

Componente	Função	Réplicas
Traefik	Load Balancer & Reverse Proxy	1
WebApp	Aplicação Flask	3
Prometheus	Coleta de métricas	1
Grafana	Dashboards e visualização	1
cAdvisor	Métricas de containers	1
Node Exporter	Métricas do sistema	1

Total:
8 containers orquestrados

TECNOLOGIAS E IMPLEMENTAÇÃO

Stack Tecnológica

Aplicação

Python 3 + Flask + Gunicorn

Bcrypt (autenticação)

Swagger/Flasgger (API docs)

Infraestrutura:

Docker + Docker Swarm

Traefik v2.10 (load balancing)

NFS (storage)

Monitoring:

Prometheus (time-series DB)

Grafana (visualização)

cAdvisor + Node Exporter

FUNCIONALIDADES CHAVE

1. Load Balancing

```yaml
webapp:
  deploy:
    replicas: 3  # 3 instâncias
    labels:
      - "traefik.http.services.webapp.loadbalancer.sticky.cookie=true"
      - "traefik.http.services.webapp.loadbalancer.healthcheck.path=/health"
```

FUNCIONALIDADES CHAVE

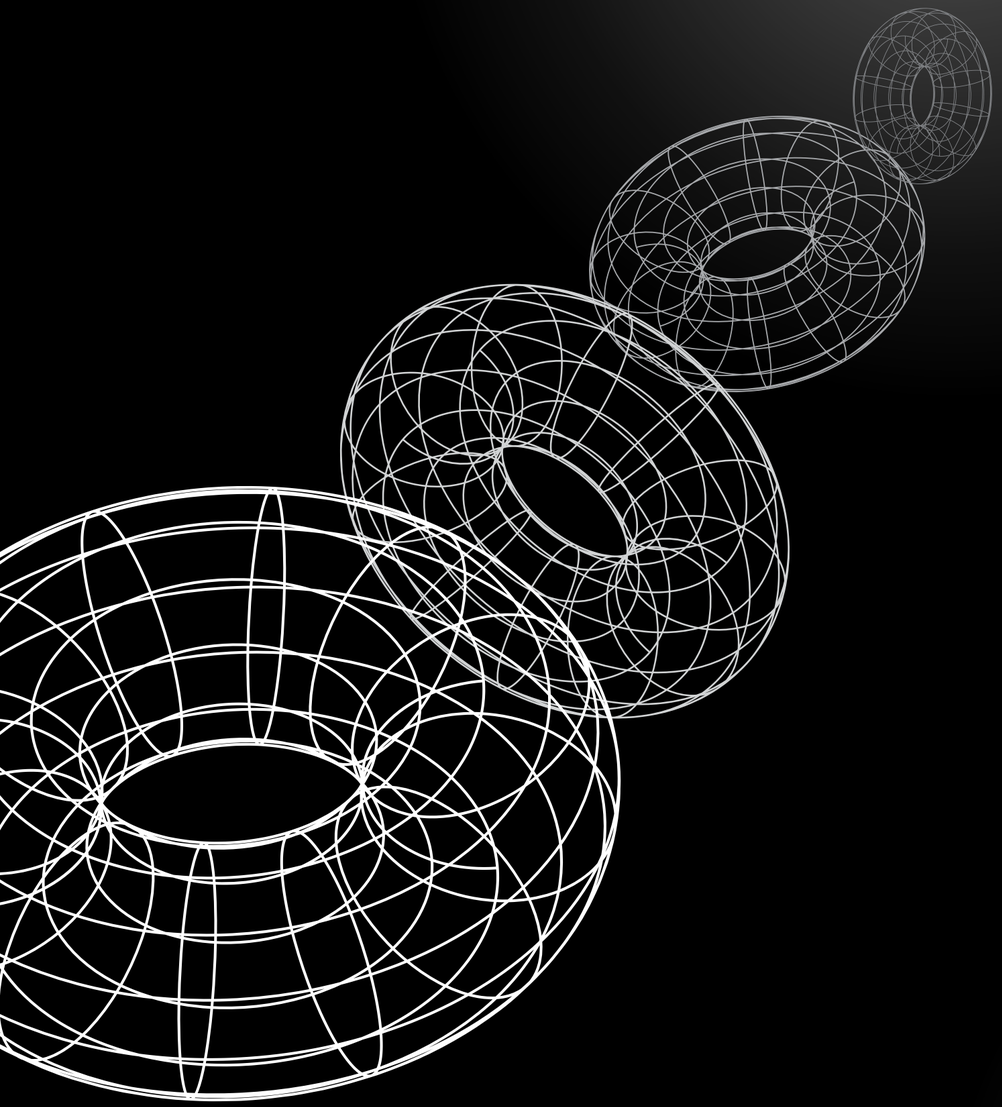
2. Zero-Downtime Updates

```
update_config:
  parallelism: 1
  order: start-first   # Nova réplica antes de matar antiga
```

FUNCIONALIDADES CHAVE

3. Auto-Recovery

```
restart_policy:
  condition: on-failure
  max_attempts: 3
```



DEMONSTRAÇÃO DE RESULTADOS

OBRIGADA