Filosofia de Operação

Introdução

O cluster não é apenas um conjunto de servidores executando containers, mas uma plataforma operacional construída sobre alguns princípios fundamentais.

Esses princípios orientam a forma como aplicações são desenvolvidas, empacotadas, implantadas e operadas dentro do ambiente.

O objetivo dessa filosofia é manter o ambiente previsível, auditável, escalável e operável por qualquer pessoa da equipe com conhecimento da plataforma.

Infraestutura como Plataforma

O cluster deve ser entendido como uma **plataforma interna de execução de software**, e não apenas como um conjunto de máquinas onde aplicações são instaladas manualmente.

Aplicações não devem depender de configurações manuais em servidores específicos.
Toda aplicação deve ser executada através de containers e publicada via stack.

Isso garante:

- Reprodutibilidade
- Portabilidade
- Versionamento de deploy
- Facilidade de rollback
- Independência de servidor específico
- Escalabilidade controlada

O servidor deixa de ser importante; o que importa é o serviço.

Containers são Efêmeros

Containers devem ser tratados como **descartáveis**.

Nenhuma informação importante deve existir apenas dentro de um container.
Tudo que precisa persistir deve estar em:

- Bancos de dados
- Redis
- Volumes NFS
- Storage externo
- Serviços de logs
- Sistemas de métricas

Se um container puder ser destruído e recriado sem perda de dados, então o sistema está corretamente arquitetado.

Deploy é Versionado

Toda aplicação publicada no cluster deve possuir:

- Repositório Git
- Versionamento semântico
- Imagem Docker versionada
- Deploy via stack
- Histórico de versões
- Possibilidade de rollback

O cluster não deve rodar código sem versão identificável.

Isso permite:

- Rastreabilidade
- Auditoria
- Rollback rápido
- Reprodutibilidade de ambiente
- Comparação entre versões

Um Processo por Container

O cluster adota o princípio de:

> Um container deve executar apenas uma responsabilidade.

Isso melhora:

- Isolamento de falhas
- Escalabilidade independente
- Observabilidade
- Logging
- Atualizações
- Diagnóstico de problemas
- Consumo de recursos previsível

Exemplo de separação comum:

Responsabilidade	Serviço
Web	nginx
Aplicação	php-fpm
Scheduler	schedule
Worker	queue
Migration	migrate
Seeder	seeder

Essa separação é parte fundamental da arquitetura do cluster.

Logs e Métricas são Obrigatórios

Todo serviço deve gerar logs e métricas.

Se um serviço não gera logs, ele não pode ser operado corretamente.
Se não existem métricas, não existe monitoramento real.

A observabilidade não é opcional, ela faz parte da plataforma.

Automação antes de operação manual

Sempre que possível, tarefas devem ser automatizadas.

Evitar:

- Alterações manuais em containers
- Alterações manuais em servidores
- Configurações feitas apenas via interface gráfica
- Deploy manual sem versionamento
- Execução manual de procedimentos repetitivos

Preferir:

- Scripts
- CI/CD
- Stack versionada
- Dockerfile
- Infraestrutura como código
- Secrets do Swarm
- Configs do Swarm

O objetivo é que qualquer ambiente possa ser recriado do zero apenas com código e configurações versionadas.

Simplicidade Operacional

Uma das decisões arquiteturais do cluster foi priorizar **simplicidade operacional**.

Isso significa:

- Menos ferramentas quando possível
- Menos camadas de abstração
- Sistemas que a equipe entende completamente
- Troubleshooting simples
- Menor overhead operacional
- Menor curva de aprendizado
- Menor dependência de especialistas

A complexidade é considerada um custo operacional.

Padronização

Aplicações publicadas no cluster devem seguir padrões definidos para:

- Estrutura de containers
- Deploy
- Versionamento
- Uso de secrets
- Uso de Redis
- Uso de volumes
- Logs
- Redes
- Exposição via Traefik
- WAF
- Observabilidade

Padronização reduz erros, facilita troubleshooting e permite que qualquer pessoa da equipe opere qualquer serviço.

Responsabilidade do Cluster vs Responsabilidade da Aplicação

Divisão clara de responsabilidades:

Cluster	Aplicação
Rede	Regra de negócio
Proxy	Código
TLS	Queries
Logs	Validação
Métricas	Processamento
Storage	Modelos
Autenticação	Interface
Observabilidade	APIs
Deploy	Features

O cluster fornece a **infraestrutura e a plataforma**, a aplicação fornece a **lógica de negócio**.

Resumo da Filosofia

O cluster foi projetado para ser previsível, reproduzível, observável e operável.

Aplicações devem ser empacotadas como containers, implantadas de forma versionada, gerar logs, expor métricas e não depender de servidores específicos.

A plataforma deve abstrair a infraestrutura para que as equipes possam focar no desenvolvimento de software, enquanto o cluster fornece rede, segurança, storage, observabilidade e mecanismos de deploy.

A simplicidade operacional, a padronização e a automação são princípios fundamentais da operação do ambiente.