KUBERNETES: VISÃO GERAL E FERRAMENTAS BÁSICAS

Responsável: João Pedro Toledo Gonçalves.

Data: 26/01/2026

MANUAL TÉCNICO - KUBERNETES: VISÃO GERAL E FERRAMENTAS BÁSICAS

Código: ITGENG 0015/26 | **Classificação:** RESTRITO
Responsável: João Pedro Toledo Gonçalves | **Data:** 26/01/2026

1. HISTÓRICO DE REVISÃO

NOTA

REGRA DE OURO:
1. **Autor:** João Pedro Toledo Gonçalves.
2. **Descrição:** Criação do documento.

Data	Versão	Descrição	Autor
26/01/2026	1.0	Criação Inicial	João Pedro Toledo Gonçalves

2. OBJETIVO

Introduzir os conceitos fundamentais do Kubernetes (K8s) e o uso da ferramenta de linha de comando `kubectl` para administração de clusters.

3. PRÉ-REQUISITOS

[ ] Um cluster Kubernetes ativo (Pode ser Minikube, K3s, EKS, AKS).
[ ] Ferramenta `kubectl` instalada localmente.
[ ] Arquivo de configuração `~/.kube/config` (kubeconfig).

4. CONCEITOS CHAVE

NOTA

NOTA: O K8s é declarativo. Você define o estado desejado (YAML) e ele trabalha para manter esse estado.

Componente	Função	Equivalente Docker
Pod	Menor unidade deployável. Pode ter 1+ containers.	Container "Grupo"
Deployment	Gerencia réplicas e updates de Pods.	Service (Swarm)

Componente	Função	Equivalente Docker
Service	Expõe o Deployment na rede (ClusterIP, NodePort).	Port Mapping
Namespace	Segregação virtual de recursos.	~Stack Name

5. PASSO A PASSO (EXECUÇÃO)

Etapa 1: Verificação de Conexão

1. [x] Verifique se o `kubectl` consegue falar com o cluster.

```
kubectl cluster-info
kubectl get nodes
```

Etapa 2: Deploy Imperativo (Rápido)

1. [x] Crie um deployment simples do Nginx.

```
kubectl create deployment demo-nginx --image=nginx
```

Etapa 3: Deploy Declarativo (YAML - Recomendado)

1. [x] Crie um arquivo `app.yaml`:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: meu-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: web
  template:
    metadata:
      labels:
        app: web
    spec:
      containers:
      - name: nginx
        image: nginx:latest
```

2. [x] Aplique o arquivo:

```
kubectl apply -f app.yaml
```

Etapa 4: Debug e Logs

1. [x] Liste os pods para pegar o nome.

```
kubectl get pods
```

2. [x] Veja os logs de um pod específico.

```
kubectl logs -f meu-app-xxxxx-xxxxx
```

3. [x] Acesse o shell interativo.

```
kubectl exec -it meu-app-xxxxx-xxxxx -- /bin/bash
```

6. SOLUÇÃO DE PROBLEMAS (TROUBLESHOOTING)

Problema 1: `ErrImagePull` ou `ImagePullBackOff`

- **Causa:** Nome da imagem errado, tag inexistente ou falta de credenciais (Private Registry).
- **Solução:**

1. [x] `kubectl describe pod [NOME_POD]` -> Veja a seção "Events".

Problema 2: `CrashLoopBackOff`

- **Causa:** O container iniciou e "morreu" imediatamente (Erro na aplicação).
- **Solução:**

1. [x] `kubectl logs [NOME_POD] previous` para ver o log antes do crash.

7. VALIDAÇÃO FINAL

[ ] `kubectl get nodes` retorna status **Ready**?
[ ] `kubectl get pods` mostra todos os pods como **Running**?
[ ] É possível aplicar mudanças editando o YAML e rodando `apply` novamente?