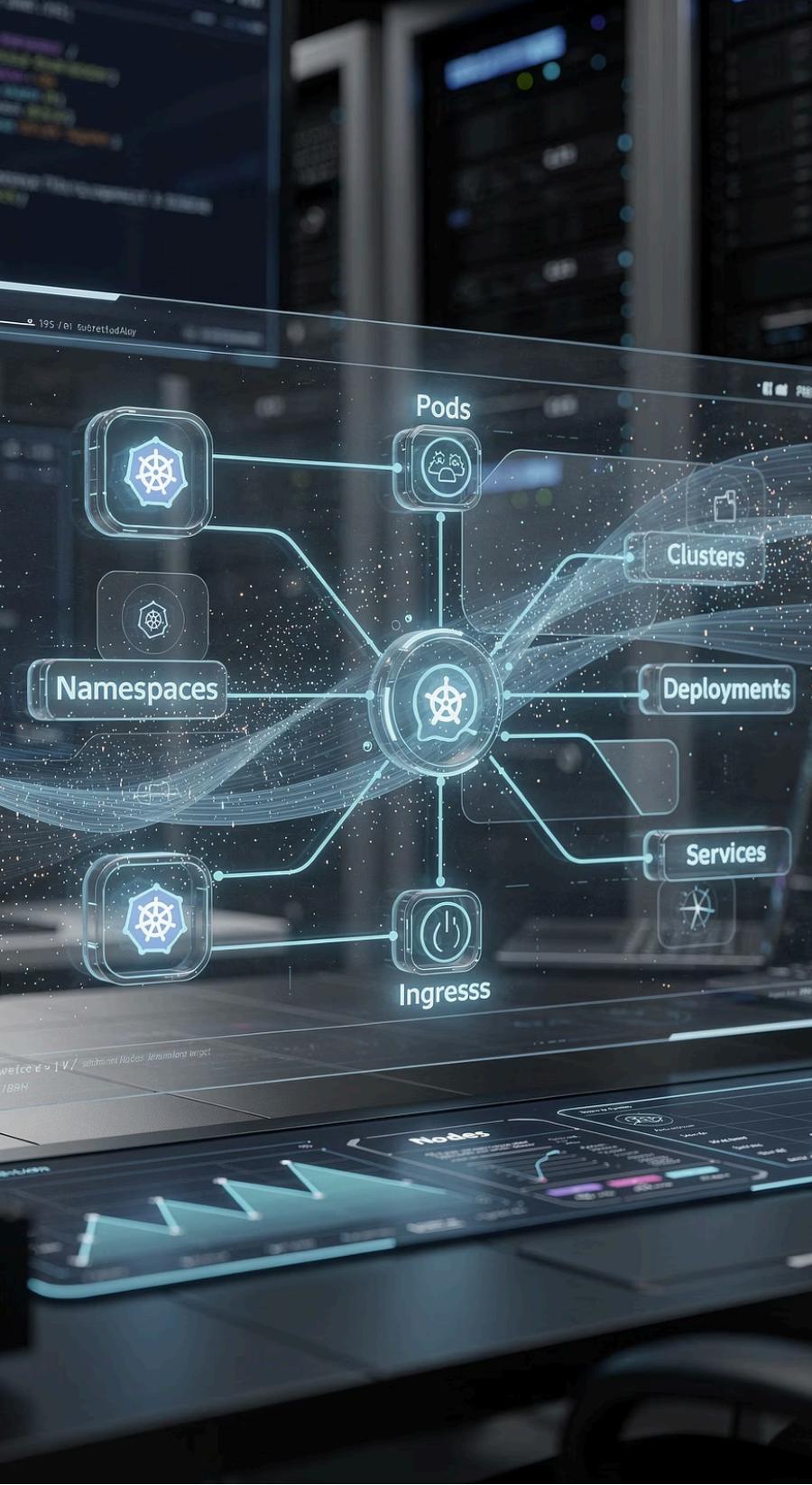


# Kubernetes Pod Priority, PriorityClass e Preemption na Prática

Entendendo PriorityClass e Preemption no Scheduler do Kubernetes

Márcio "Zamp" Zampiron



# Objetivos da Aula



## Pod Priority e PriorityClass

Compreender o que são e como funcionam esses objetos no Kubernetes



## Mecanismo de Preemption

Entender quando e como o scheduler desaloja pods de menor prioridade



## PriorityClass vs QoS

Diferenciar dois conceitos complementares mas independentes



## Laboratório Prático

Executar manifests YAML e observar o comportamento em um cluster real



# Pré-requisitos

## Conhecimentos Necessários

- Conceitos básicos de Pods e Deployments
- Uso de Namespaces e Services
- Comandos essenciais do kubectl
- Leitura e interpretação de manifests YAML

## Ambiente de Lab

- Cluster Kubernetes local (kind, minikube ou k3d)
- kubectl instalado e configurado
- Acesso ao repositório Git com os manifests
- Mínimo 4GB RAM disponível para o cluster

# Pod Priority e PriorityClass

01

**PriorityClass** é um objeto não-namespaced

Define classes de prioridade globais no cluster, mapeando um nome para um valor inteiro

02

Campo **priorityClassName** em **Pods/Deployments**

Referencia a PriorityClass desejada no spec do workload

03

**Regra fundamental: maior valor = maior prioridade**

O scheduler usa esse valor para decidir ordem de agendamento e elegibilidade para preempção

- Importante:** PriorityClass com `globalDefault: true` será aplicada automaticamente a pods que não especificarem uma classe



# Preemption (Preempção)

## O que é Preempção?

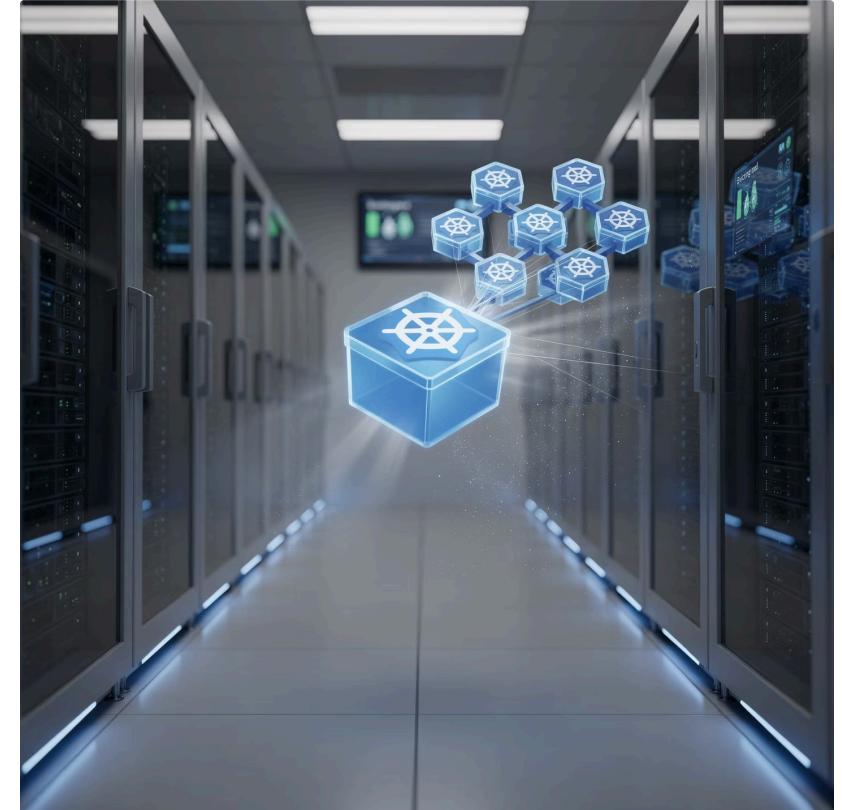
Preempção é o processo pelo qual o scheduler do Kubernetes **desaloja pods de menor prioridade** para liberar recursos e agendar pods de maior prioridade quando não há capacidade disponível no cluster.

## Quando é acionada?

- Pod de alta prioridade está *Pending*
- Não há recursos suficientes em nenhum node
- Scheduler identifica pods de baixa prioridade que podem ser removidos
- Preempção só ocorre se liberar recursos suficientes

## PreemptionPolicy

- PreemptLowerPriority: permite desalojar pods (padrão)
- Never: nunca desaloja outros pods, apenas aguarda



# PriorityClass x QoS

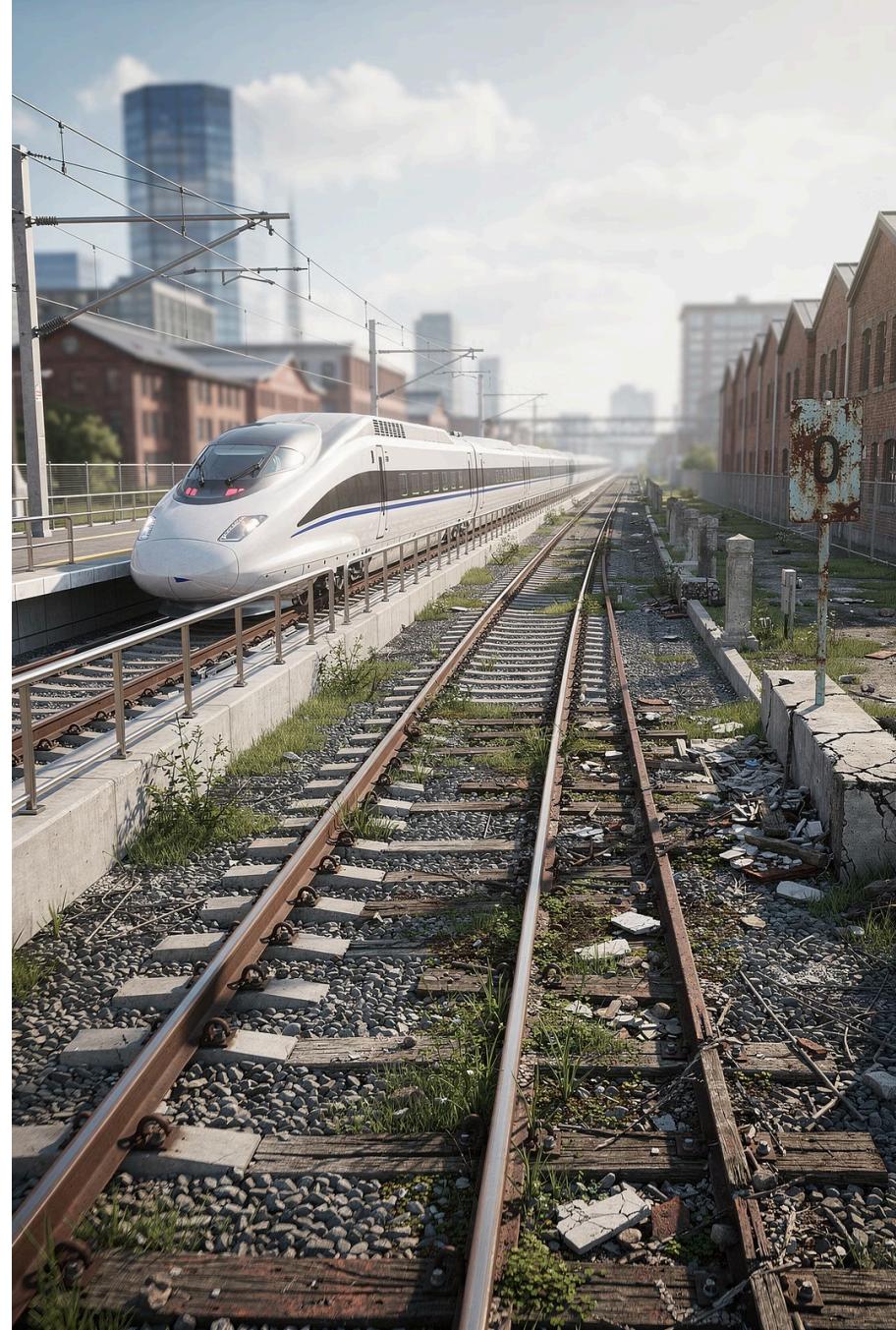
## PriorityClass

- Usado pelo **scheduler** para decidir ordem de agendamento
- Define quais pods podem ser **preemptados**
- Configurado explicitamente via priorityClassName
- Valores numéricos comparáveis (0 a 1.000.000.000+)

## QoS (Quality of Service)

- Usado pelo **kubelet** em situações de pressão de recursos no node
- Classes: Guaranteed, Burstable, BestEffort
- Calculado automaticamente com base em requests/limits
- Define ordem de eviction quando o node está sob pressão

PriorityClass e QoS são **eixos diferentes**: um influencia o scheduler, o outro influencia o kubelet. Ambos podem coexistir no mesmo pod.



# Fila de Pods por Prioridade

## 1 Pods Baixa Prioridade

Valor: 100

Jobs batch, cargas não críticas

Elegíveis para preempção

## 2 Pods Média Prioridade

Valor: 1000

Serviços padrão, backends

Agendamento normal

## 3 Pods Alta Prioridade

Valor: 10000

Apps críticos, infraestrutura

Podem causar preempção

## Cenário de Preempção

Quando um pod de **alta prioridade** entra na fila e não encontra recursos disponíveis, o scheduler identifica pods de baixa prioridade em nodes que, se removidos, permitiriam o agendamento do pod prioritário. Os pods selecionados recebem um **graceful termination** e são desalojados.



# Arquitetura do Laboratório

Nosso laboratório está organizado em três manifests principais que demonstram progressivamente os conceitos de prioridade e preempção:



## 01-priorityclasses.yaml

Define as classes de prioridade que serão usadas: high-priority, medium-priority e low-priority



## 02-pods-priorities.yaml

Cria workloads (Deployments) com diferentes prioridades para observar comportamento de agendamento

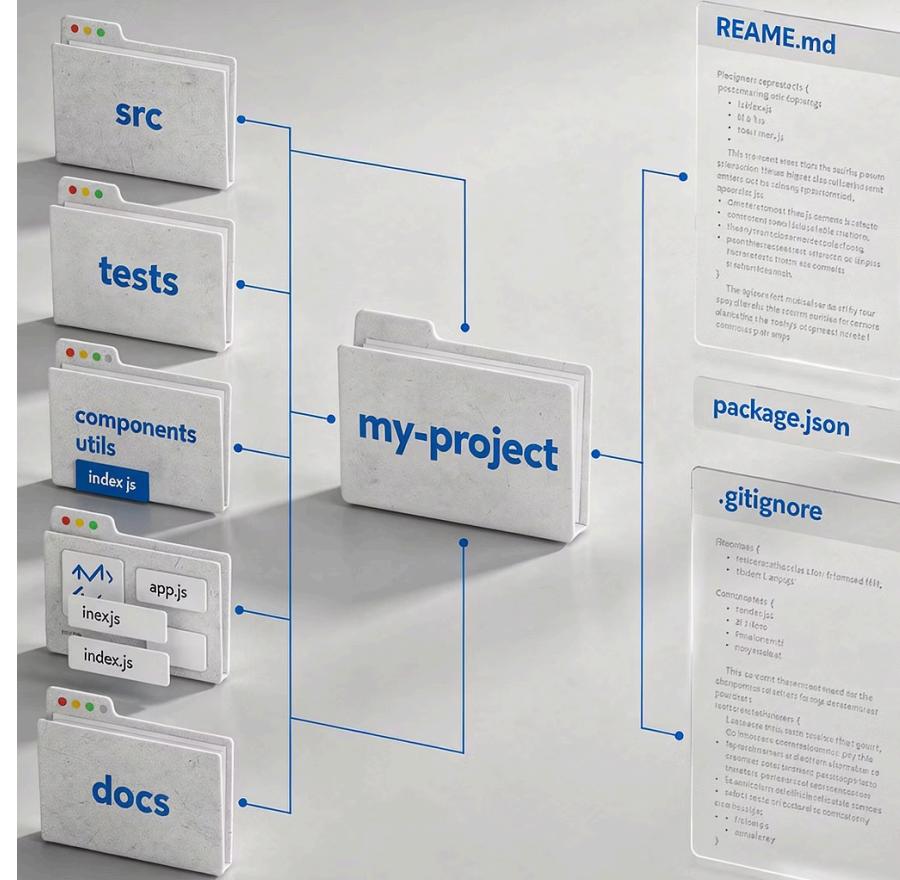


## 03-preemption-demo.yaml

Simula cenário de pressão de recursos e força preempção de pods de baixa prioridade

- Todos os arquivos completos, incluindo README com instruções detalhadas, estão disponíveis no repositório Git público da aula

## Git Repository Structure



# Lab 1: Criando PriorityClasses

## Passos Principais

### 1 Aplicar o manifest

```
kubectl apply -f \  
manifests/01-  
priorityclasses.yaml
```

### 2 Listar classes criadas

```
kubectl get priorityclass
```

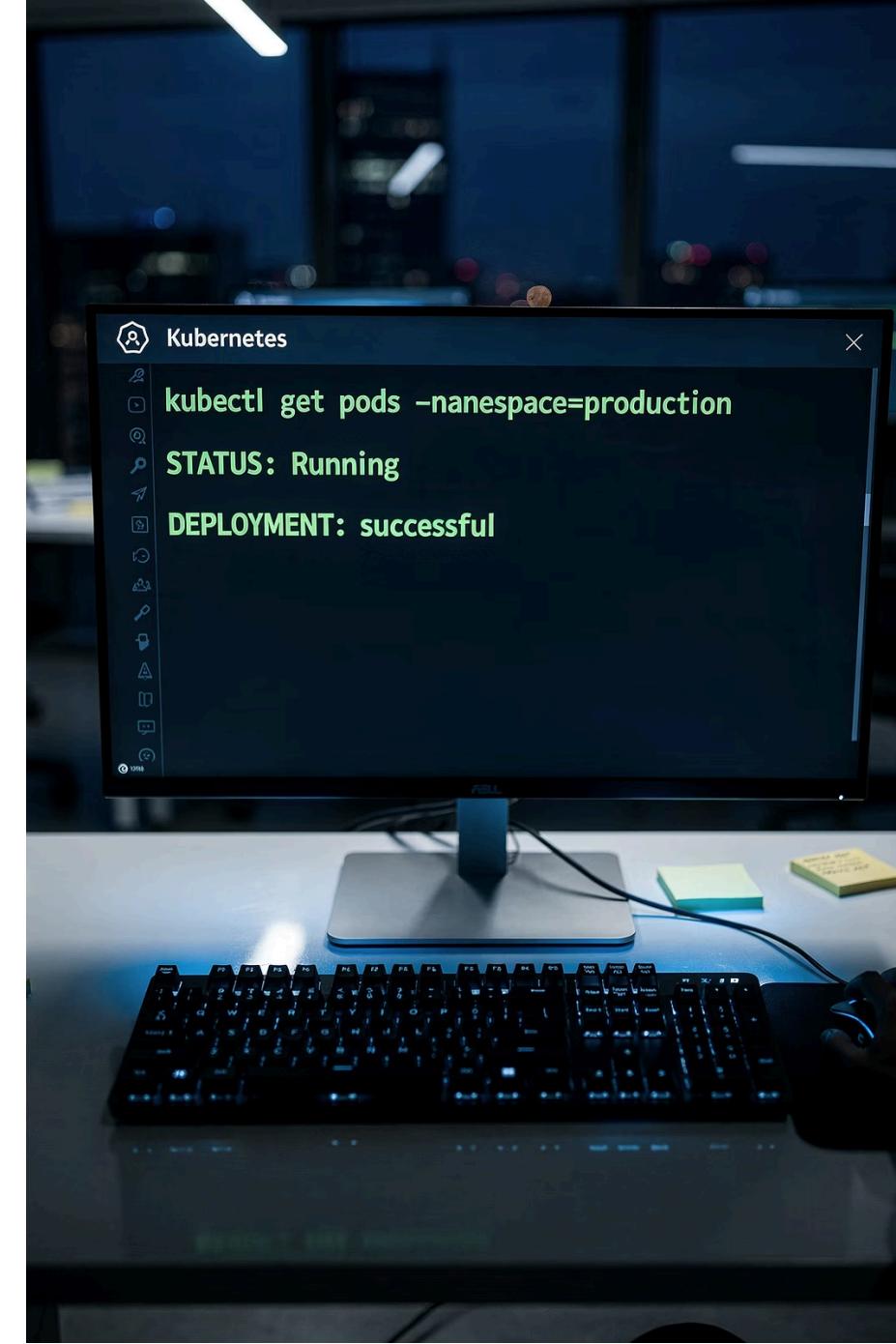
### 3 Inspecionar detalhes

```
kubectl describe  
priorityclass \  
high-priority
```

## Exemplo de PriorityClass

```
apiVersion: scheduling.k8s.io/v1  
kind: PriorityClass  
metadata:  
  name: high-priority  
  value: 10000  
  globalDefault: false  
  description: "Alta prioridade para apps  
  críticos"
```

Observe os campos principais: `value` define a prioridade numérica, e `globalDefault` indica se será usada por padrão.





# Lab 2: Workloads com Prioridades

**frontend-high**

**PriorityClass:** `high-priority`

**Valor:** 10000

**Tipo:** Aplicação crítica de frontend

**backend-medium**

**PriorityClass:** `medium-priority`

**Valor:** 1000

**Tipo:** Serviços backend padrão

**batch-low**

**PriorityClass:** `low-priority`

**Valor:** 100

**Tipo:** Jobs batch e processamento não crítico

## Verificando Prioridades

```
kubectl apply -f \
manifests/02-pods-priorities.yaml
```

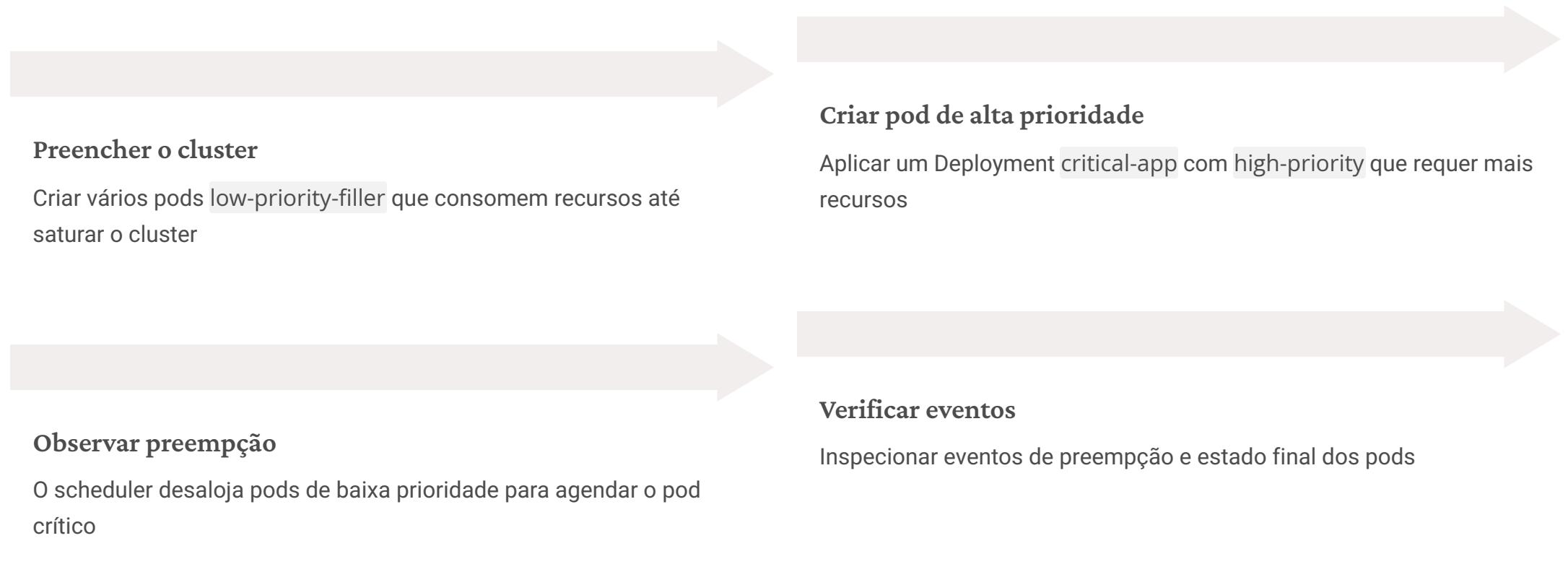
```
kubectl get pods -o wide
```

```
kubectl describe pod frontend-
high-xxxx | \
grep -i priority
```

```
# Saída esperada:
# Priority Class Name: high-priority
# Priority: 10000
```

# Lab 3: Demo de Preemption

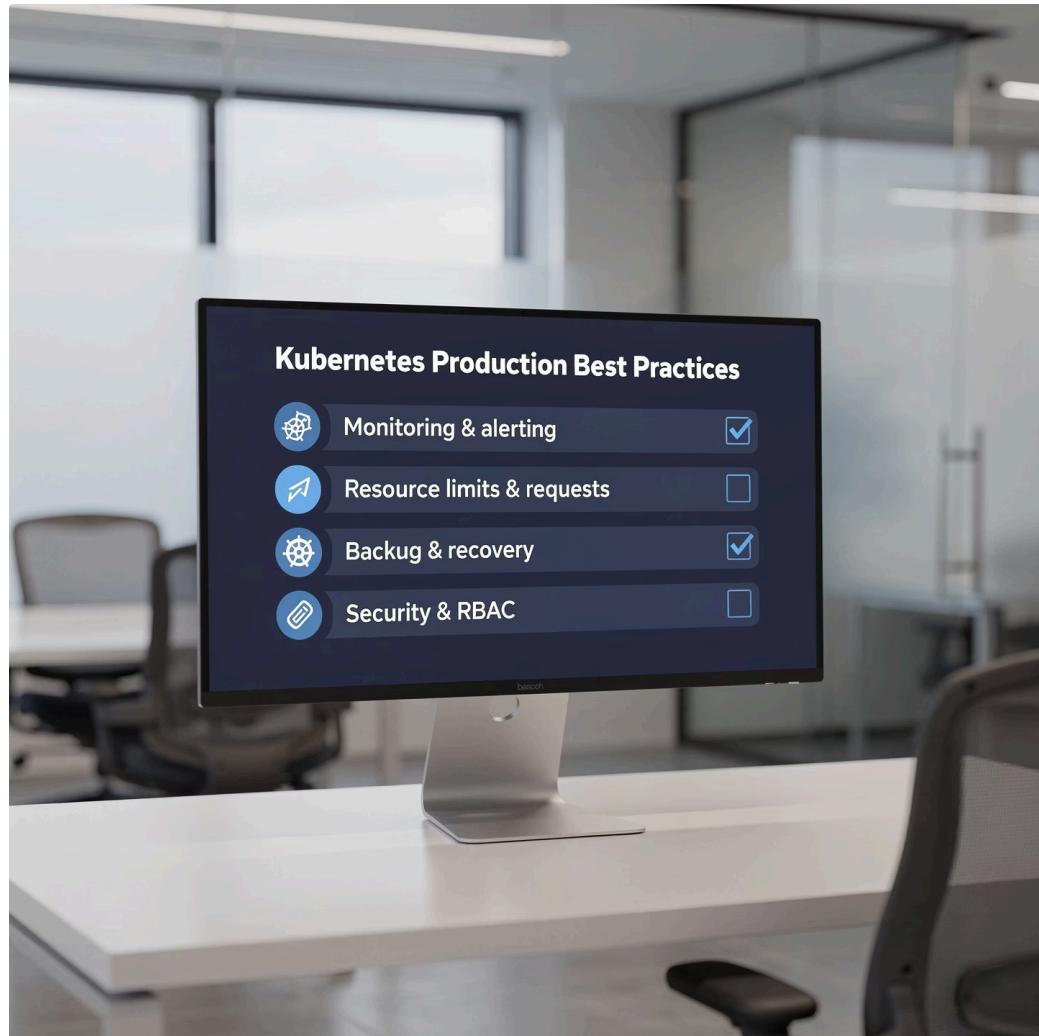
## Fluxo do Experimento



## Comandos de Observação

```
kubectl get pods -o wide --watch  
kubectl describe pod critical-app-xxx | grep -A 10 Events  
kubectl get events --sort-by='.lastTimestamp' | grep -i preempt
```

# Boas Práticas em Produção



1

## Evite globalDefault sem estratégia

Não defina globalDefault: true sem planejamento. Pode causar comportamento inesperado em workloads existentes

2

## Defina poucas faixas claras

Recomendação: 3-4 níveis (crítico, alto, normal, batch) são suficientes para a maioria dos casos

3

## Não distribua alta prioridade indiscriminadamente

Reserve prioridades altas apenas para workloads verdadeiramente críticos: sistema, observabilidade, rede

4

## Use para infraestrutura essencial

Aplique PriorityClass em componentes como: CoreDNS, kube-proxy, CNI, monitoring, ingress controllers

"Se tudo é prioridade alta, nada é prioridade alta"

# Quiz de Revisão

- PriorityClass é um recurso namespaced ou cluster-scoped?

Resposta: Cluster-scoped (não-namespaced)

- O que acontece quando um pod de alta prioridade não encontra recursos disponíveis?

Resposta: O scheduler pode desalojar (preemptar) pods de menor prioridade

- Qual a diferença entre PriorityClass e QoS?

Resposta: PriorityClass influencia o scheduler (agendamento/preempção), QoS influencia o kubelet (eviction sob pressão)

- O que significa **preemptionPolicy: Never**?

Resposta: O pod não pode desalojar outros pods, apenas aguarda recursos ficarem disponíveis

- Um pod sem priorityClassName especificado recebe qual prioridade?

Resposta: 0 (zero), ou a PriorityClass marcada com globalDefault: true, se existir



# Obrigado!

## Recapitulando

- PriorityClass define prioridades numéricas globais
- Scheduler usa prioridades para ordenar agendamento
- Preemption desaloja pods de baixa prioridade quando necessário
- PriorityClass e QoS são conceitos complementares e independentes

[Acessar Repositório Git](#)

[Acessar linkedin](#)

## Referências

- [Pod Priority and Preemption - Kubernetes Docs](#)
- [Pod Scheduling - Official Guide](#)

## Repositório do Lab

Acesse todos os manifests YAML e instruções completas no repositório Git público da aula