Colaboradores: **HECTOR JOSE RODRIGUES SALGUEIROS, WILLIANS SILVA SANTOS e UESLEI FERREIRA DOS REIS RIBEIRO**

Guia para baixar e implantar uma imagem NGINX no Kubernetes usando o Minikube

**Usando Windows:**

---

## Passo 1: Configurar o Cluster Kubernetes

Se você já configurou o Minikube no Notebook A, pode pular esta etapa. Caso contrário:

### 1.1. Instalar Pré-requisitos

- Docker Desktop : Instale o Docker Desktop para Windows (https://www.docker.com/products/docker-desktop/ ).
- kubectl : Instale o cliente Kubernetes via Chocolatey:
- powershell
- `choco install kubernetes-cli`
- Minikube : Baixe e instale o Minikube para Windows (https://minikube.sigs.k8s.io/docs/start/ ).

### 1.2. Iniciar o Cluster Minikube

Abra o PowerShell como administrador e execute:

powershell

```
# Iniciar o cluster com recursos adequados (ajuste conforme sua máquina)
minikube start --driver=hyperv --cpus=2 --memory=2200 --disk-size=20g
```

### 1.3. Verificar o Cluster

powershell

```
    kubectl get nodes
# Saída esperada: STATUS Ready
```

---

## Passo 2: Implantar o NGINX

Vamos criar um deployment usando a imagem oficial do NGINX (`nginx:latest`).

### 2.1. Criar o Deployment

Crie um arquivo `nginx-deployment.yaml`:

yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx-deployment
spec:
 replicas: 2
 selector:
   matchLabels:
     app: nginx
 template:
   metadata:
     labels:
       app: nginx
   spec:
     containers:
     - name: nginx
       image: nginx:latest
       ports:
       - containerPort: 80
       resources:
         requests:
           cpu: 100m
           memory: 128Mi
         limits:
           cpu: 500m
           memory: 256Mi
```

Aplicar o deployment :

powershell

```
kubectl apply -f nginx-deployment.yaml
```

### 2.2. Expor o Serviço

Crie um serviço para acessar o NGINX:

powershell

```
kubectl expose deployment nginx-deployment
--type=NodePort --port=80
```

### 2.3. Obter a URL do Serviço

Execute o seguinte comando para obter a URL do serviço:

powershell

```
minikube service nginx-deployment --url
```

- Copie a URL gerada e abra no navegador. Você verá a página padrão do NGINX.

---

## Passo 3: Habilitar Auto-Healing e HPA

### 3.1. Auto-Healing

O Kubernetes já possui mecanismos nativos de auto-healing. Para testar:

powershell

```
kubectl delete pod <nome-do-pod>  # Substitua pelo nome do pod real
kubectl get pods  # O pod será recriado automaticamente
```

### 3.2. Configurar Horizontal Pod Autoscaler (HPA)

Primeiro, instale o metrics-server :

powershell

```
minikube addons enable metrics-server
```

Crie um arquivo `hpa.yaml`:

yaml

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
 name: nginx-hpa
```

```
spec:
 scaleTargetRef:
   apiVersion: apps/v1
   kind: Deployment
   name: nginx-deployment
 minReplicas: 2
 maxReplicas: 5
 metrics:
 - type: Resource
   resource:
     name: cpu
     target:
       type: Utilization
       averageUtilization: 10
```

Aplicar o HPA :

powershell
```
kubectl apply -f hpa.yaml
```

---

Para configurar o Prometheus e monitorar o cluster Kubernetes vamos seguir os passos abaixo. Iremos usar o Prometheus para coletar métricas e exibir informações em tempo real sobre o cluster.

## Passo 1: Pré-requisitos

Certifique-se de que:

1. O Notebook A está rodando o cluster Kubernetes com o Minikube.
2. O Notebook B tem acesso ao cluster Kubernetes (configuração do `kubeconfig` já foi feita anteriormente).
3. Instale o kubectl , Helm e Docker Desktop no Notebook B.

**Instalar Helm:** O Helm é um gerenciador de pacotes para Kubernetes. Instale-o no Notebook B:

powershell
```
choco install kubernetes-helm
```

---

## Passo 2: Configurar o Prometheus no Notebook B

Vamos implantar o Prometheus no cluster Kubernetes usando o Helm.

### 2.1. Adicionar o Repositório do Prometheus

Execute o seguinte comando no Notebook B:

powershell
```
helm repo add prometheus-community
https://prometheus-community.github.io/helm-charts
helm repo update
```

### 2.2. Criar um Namespace para o Prometheus

Crie um namespace separado para o Prometheus:

powershell
```
kubectl create namespace monitoring
```

### 2.3. Instalar o Prometheus

Instale o Prometheus no namespace `monitoring`:

powershell
```
helm install prometheus prometheus-community/prometheus
--namespace monitoring
```

- Isso irá implantar o Prometheus e outros componentes necessários (como o servidor de alertas).

### 2.4. Verificar os Pods do Prometheus

Verifique se os pods do Prometheus estão rodando:

powershell
```
kubectl get pods -n monitoring
```

Você deve ver algo como:

```
NAME                                                READY
STATUS    RESTARTS   AGE
prometheus-server-<hash>                            2/2
Running   0          1m
```

```
prometheus-kube-state-metrics-<hash>              1/1
Running   0          1m
prometheus-alertmanager-<hash>                    2/2
Running   0          1m
```

---

## Passo 3: Configurar o Acesso ao Prometheus

### 3.1. Obter a URL do Prometheus

Use o seguinte comando para obter a URL do Prometheus:

powershell
```
kubectl port-forward svc/prometheus-server -n monitoring
9090:80
```

- Isso expõe o Prometheus na porta `9090` do Notebook B.
- Abra o navegador no Notebook B e acesse: `http://localhost:9090`.

### 3.2. Explorar Métricas no Prometheus

No painel do Prometheus (`http://localhost:9090`), você pode consultar métricas diretamente. Algumas consultas úteis:

1. Número de Pods Ativos :
2. promql
3. `sum(kube_pod_info) by (namespace)`
4. Uso de CPU por Pod :
5. promql
6. `sum(rate(container_cpu_usage_seconds_total{container!="POD", container!="", pod=~".+"}[5m])) by (pod)`
7. Estado dos Pods (Running, Failed, Pending) :
8. promql
9. `kube_pod_status_phase{phase="Running"}`
10. `kube_pod_status_phase{phase="Failed"}`
11. `kube_pod_status_phase{phase="Pending"}`
12. Ações Disparadas pelo HPA :
13. promql
14. `kube_horizontalpodautoscaler_status_current_replicas`
15. `kube_horizontalpodautoscaler_status_desired_replicas`

## Para exibir as métricas em tempo real :

*Note que pode ser testado localmente com os mesmos passos

### Passo 1: Acessar o Prometheus

Certifique-se de que o Prometheus está rodando no Notebook B e acessível via `kubectl port-forward`:

powershell

```
kubectl port-forward svc/prometheus-server -n monitoring 9090:80
```

Abra o navegador no Notebook B e acesse:
http://localhost:9090

### Passo 2: Consultas no Prometheus

#### 2.1. Número de Pods Ativos

sum(kube_pod_info) by (namespace)

#### 2.2 Uso de CPU por Pod:

promql

sum(rate(container_cpu_usage_seconds_total{container!="POD"}[5m])) by (pod)

#### 2.3. Estado dos Pods (Running, Failed, Pending)

Você pode usar as métricas `kube_pod_status_phase` para verificar o estado dos pods. Aqui estão as consultas para cada estado:

- Pods Running :

promql

```
sum(kube_pod_status_phase{phase="Running"}) by (namespace)
```

- Pods Failed :

promql

```
sum(kube_pod_status_phase{phase="Failed"}) by (namespace)
```

- Pods Pending :

promql

```
sum(kube_pod_status_phase{phase="Pending"}) by (namespace)
```

Essas consultas mostram o número de pods em cada estado, agrupados por namespace.

## Demonstração de Tolerância a Falhas

Mostrar Pods rodando:
kubectl get pods

Deletar um pod:

kubectl delete pod </>

Sobrecarga de CPU (Escalonamento Horizontal):

## Verificar o HPA Configurado

Primeiro, confirme que o HPA está configurado para escalar com base no uso de CPU. Execute:

powershell

```
kubectl get hpa
```

## Gerar Carga de CPU Artificial

kubectl exec -it nginx-deployment-97666ffd5-spnj2 -- /bin/sh -c "while true; do :; done"

## Monitorar o HPA

kubectl get hpa -w

## Passo 3: Parar a Carga de CPU

Pressione `Ctrl+C` no terminal onde o loop está rodando para interromper a carga.

## Passo 4: Verificar o Escalonamento

kubectl get pods