Gerenciamento de Nodes no Kubernetes

1. Gerenciamento de Serviços e ReplicaSets

Serviços e ReplicaSets são componentes essenciais no Kubernetes para garantir que suas aplicações estejam sempre disponíveis e funcionem conforme esperado.

• **Serviços (Services):**

Um Serviço em Kubernetes é uma abstração que define um conjunto lógico de Pods e uma política para acessá-los. Ele permite que diferentes partes de uma aplicação se comuniquem entre si, independente da localização dos Pods.

• **Tipos de Serviços:**

ClusterIP: Exponha o Serviço internamente dentro do cluster.

NodePort: Exponha o Serviço em uma porta específica de cada nó no cluster.

LoadBalancer: Exponha o Serviço externamente usando um balanceador de carga.

• **ReplicaSets**:

- Um ReplicaSet assegura que um número especificado de réplicas de um Pod esteja rodando a qualquer momento. Ele substitui Pods que falham ou ficam indisponíveis, garantindo a alta disponibilidade.

- Configuração Básica:

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: my-replicaset
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: my-container
        image: my-image:latest
        ports:
        - containerPort: 8080
```

Em um sistema de gerenciamento de pedidos, um ReplicaSet pode garantir que sempre haja três instâncias do serviço backend em execução, enquanto o Serviço Kubernetes permite que o frontend se comunique com essas instâncias de forma transparente.

2. Hello Node Kubernetes

O “Hello Node” é uma aplicação básica de exemplo em Kubernetes, que pode ser usada para entender os conceitos de implantação e gerenciamento de nós no cluster.

Implantação do Hello Node:

1. Escreva o Manifesto do Pod:

- Crie um arquivo YAML para o Pod do “Hello Node”:

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: hello-node
spec:
  containers:
  - name: hello-node
    image: gcr.io/hello-minikube-zero-install/hello-node
    ports:
    - containerPort: 8080
```

1. Implante o Pod:

Use kubectl apply para implantar o Pod no cluster: kubectl apply -f hello-node.yaml

2. Verifique o Status:

Verifique se o Pod está rodando corretamente: kubectl get pods

O “Hello Node” é um bom ponto de partida para quem está começando a trabalhar com Kubernetes, permitindo que você veja como o Kubernetes gerencia os contêineres em execução em seus nós.

3. Configuração de um Cluster via CLI

Configurar um cluster Kubernetes via linha de comando (CLI) no Google Cloud é um processo direto, mas poderoso, permitindo automação e controle granular.

Passos para Configurar o Cluster via CLI:

1. Inicie o SDK do Google Cloud:

- Certifique-se de que o Google Cloud SDK está instalado e autenticado.

2. Crie o Cluster:

Use o comando gcloud para criar um novo cluster: gcloud container clusters create my-cluster --num-nodes=3 --zone=us-central1-a

3. Configuração do Kubernetes:

Configure o kubectl para usar o novo cluster: gcloud container clusters get-credentials my-cluster --zone us-central1-a

Ao configurar um cluster via CLI, você pode automatizar a criação de clusters em diferentes ambientes, facilitando a implantação contínua e o gerenciamento de infraestruturas complexas.

4. Interpretação e Criação de Pods

Pods são a menor unidade de execução em Kubernetes, e entender como interpretá-los e criá-los é fundamental para o gerenciamento eficaz de aplicações.

Criação e Interpretação de Pod

1. Especificação do Pod:

Um Pod é especificado em YAML e inclui informações sobre os contêineres, volumes, rótulos, e outros aspectos. Exemplo básico:

1. Interpretação:

- Verifique a definição do Pod usando kubectl describe pod my-pod. Isso fornece detalhes sobre o status do contêiner, volumes, eventos recentes, etc.

Quando você precisa depurar uma aplicação que não está funcionando corretamente, interpretar a definição do Pod e os eventos associados pode ajudar a identificar problemas, como falhas de imagem ou problemas de conectividade.

5. Permissão de Tráfego Externo

Expor serviços Kubernetes ao tráfego externo é essencial para aplicações que precisam ser acessíveis publicamente.

Permissão de Tráfego Externo:

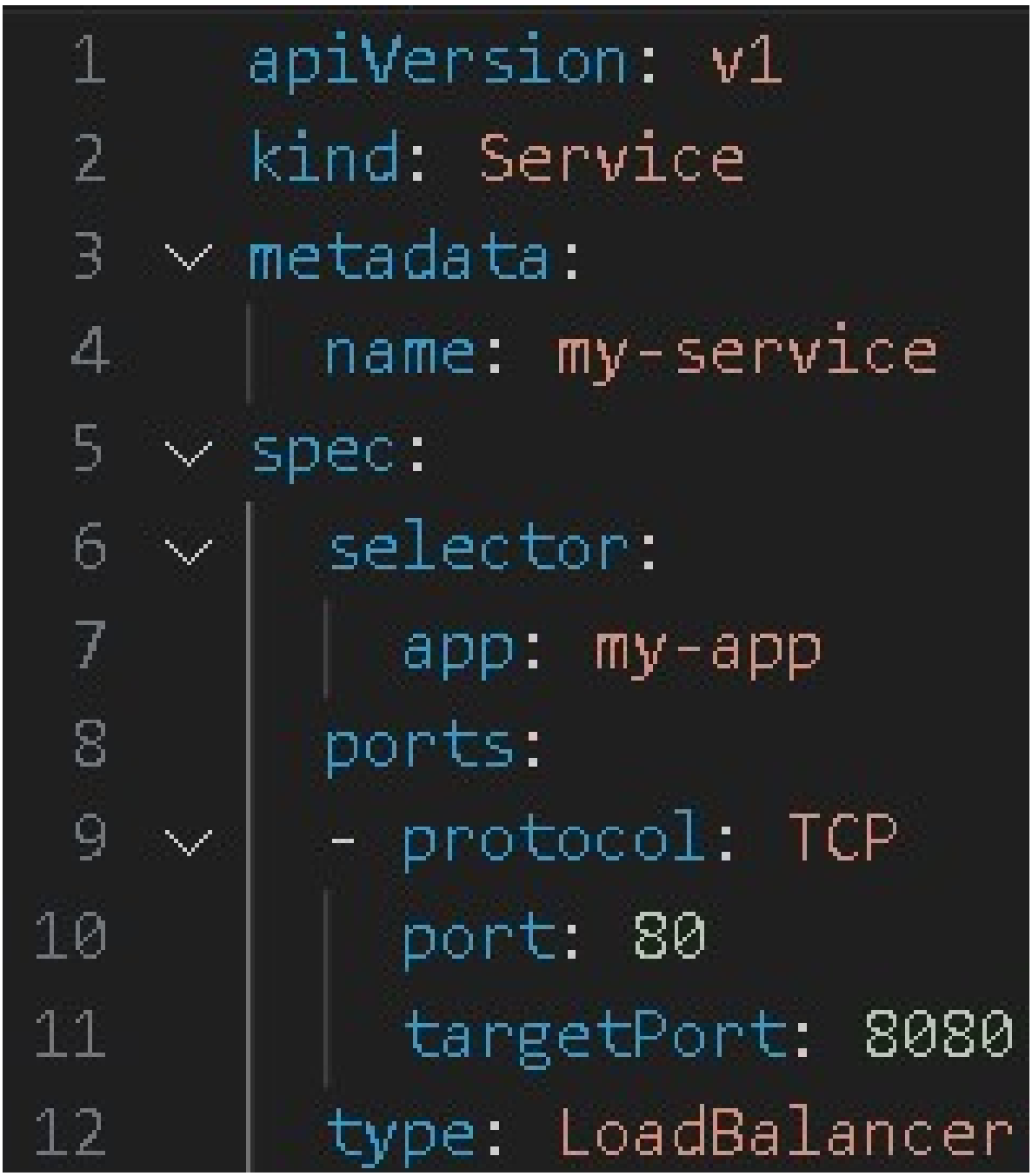
Serviço NodePort:

Um serviço do tipo NodePort permite que um serviço seja acessado através de uma porta específica em cada nó no cluster:

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: my-app
  ports:
  - protocol: TCP
    port: 80
    targetPort: 8080
    nodePort: 30007
  type: NodePort
```

Serviço LoadBalancer:

• Um serviço do tipo LoadBalancer cria um IP externo e balanceia o tráfego entre os Pods associados:



Para uma aplicação web que precisa ser acessível na internet, como um site de e-commerce, configurar um LoadBalancer no GKE permite distribuir o tráfego de entrada entre várias réplicas do serviço, garantindo alta disponibilidade e resiliência.

6. Escalonamento de Serviços

O escalonamento é crucial para manter a performance das aplicações sob diferentes cargas de trabalho.

Escalonamento de Serviços:

1. Escalonamento Manual:

Aumente ou diminua o número de réplicas manualmente: kubectl scale deployment my-deployment --replicas=5

2. Escalonamento Horizontal Automático (HPA):

Configure o Horizontal Pod Autoscaler para ajustar automaticamente o número de réplicas com base na utilização de recursos: kubectl autoscale deployment my-deployment --cpu-percent=80 --min=2 --max=10

Em uma aplicação de streaming de vídeo, onde o número de usuários pode variar drasticamente, o escalonamento automático garante que novos pods sejam adicionados para lidar com picos de tráfego, mantendo a experiência do usuário consistente.

7. Implementação de Atualizações em Serviços

Kubernetes suporta a atualização contínua de serviços com mínimo ou nenhum tempo de inatividade.

Implementação de Atualizações:

1. Rolling Updates:

Kubernetes substitui pods antigos por novos de forma gradual: kubectl set image deployment/my-deployment my-container=my-image:v2

2. Monitoramento de Atualizações:

Verifique o progresso da atualização: kubectl rollout status deployment/my-deployment

3. Rollback em Caso de Problemas:

Se houver problemas na atualização, você pode reverter para a versão anterior: kubectl rollout undo deployment/my-deployment

Em uma aplicação crítica, como um sistema bancário, onde a disponibilidade é essencial, usar rolling updates permite implantar novas versões do serviço sem interromper os usuários, e o rollback rápido ajuda a mitigar quaisquer problemas que possam surgir.

Extra: Pods, Services e Nodes no Kubernetes

1. O Que São Pods no Kubernetes?

Pods são a menor e mais básica unidade de execução em Kubernetes. Eles representam uma camada de abstração em torno de contêineres, agrupando um ou mais contêineres que compartilham a mesma rede, armazenamento e ciclo de vida. Dentro de um Pod, os contêineres podem se comunicar entre si facilmente, usando a rede local do Pod.

• **Estrutura de um Pod:**

- Contêineres: O componente central de um Pod. Pod pode conter um único contêiner (o mais comum) ou vários contêineres que trabalham juntos.

- Volumes: Os Pods podem ter volumes anexados que são usados para persistir dados compartilhados entre contêineres no Pod.

- Rede: Cada Pod tem seu próprio endereço IP. Todos os contêineres dentro do Pod compartilham o mesmo namespace de rede, podendo se comunicar

através de localhost.

• **Ciclo de Vida dos Pods:**

- Diferente de contêineres individuais, Pods são criados e destruídos de acordo com a necessidade do cluster. Quando um Pod é destruído (seja por falha ou por atualizações), ele não é reiniciado diretamente — em vez disso, um novo Pod é criado.

Em uma aplicação web simples, um Pod pode ser usado para rodar um contêiner de servidor web como Nginx, que responde a requisições HTTP. Se você precisar adicionar um cache local, outro contêiner de cache poderia ser adicionado ao mesmo Pod.

2. O Que São Services no Kubernetes?

Services são abstrações que definem uma maneira de expor os Pods como um conjunto lógico e estável, facilitando a comunicação entre eles. Como os Pods podem ser efêmeros e seus IPs mudam frequentemente, os Services oferecem uma interface consistente para acessar os Pods.

• **Tipos de Services:**

- ClusterIP: O tipo de serviço padrão. Exponha o serviço dentro do cluster, permitindo que outros Pods se comuniquem com ele.

- NodePort: Exponha o serviço em uma porta específica em cada nó, permitindo o acesso de fora do cluster.

- LoadBalancer: Cria um balanceador de carga externo que distribui o tráfego entre os Pods.

- ExternalName: Mapeia o serviço para um nome DNS externo.

• **Como Funciona:**

O Kubernetes usa um mecanismo de seleção de rótulos (labels) para associar os Pods ao Serviço. Um Serviço acompanha as mudanças nos Pods e garante que qualquer Pod que corresponda ao rótulo associado seja automaticamente incluído no serviço.

Se você tem uma aplicação de backend rodando em vários Pods, um serviço de tipo ClusterIP pode ser usado para garantir que o frontend possa se comunicar com o backend, independentemente de qual Pod esteja em execução naquele momento.

3. O Que São Nodes no Kubernetes?

Nodes (Nós) são as máquinas (físicas ou virtuais) onde os Pods são executados. Um cluster Kubernetes é composto por um ou mais nodes, e cada node contém os serviços necessários para gerenciar a execução dos Pods.

• **Componentes de um Node:**

- Kubelet: O agente que roda em cada node, responsável por garantir que os contêineres estejam rodando conforme as especificações dos manifestos de Pods.

- Kube-proxy: Gerencia as regras de rede e o roteamento de tráfego para garantir que os serviços Kubernetes funcionem corretamente.

- Container Runtime: O software que realmente executa os contêineres, como Docker ou containerd.

• **Master Node vs. Worker Node:**

- Master Node: É o nó que controla o cluster, gerenciando as operações do Kubernetes, como o agendamento de Pods e a manutenção do estado do cluster.

- Worker Nodes: São os nós que executam os Pods e suas aplicações. O master node distribui as tarefas entre os worker nodes.

Um cluster Kubernetes com três nodes pode ter Pods distribuídos uniformemente entre esses nodes, com o Kubernetes garantindo que, se um node falhar, os Pods sejam redistribuídos para os nodes restantes.

Como Pods, Services e Nodes Trabalham Juntos

• **Distribuição de Pods nos Nodes:** Quando um Pod é criado, o Kubernetes decide em qual node ele deve ser executado, levando em consideração fatores como recursos disponíveis e balanceamento de carga.

• **Acesso aos Pods via Services**: Para facilitar o acesso aos Pods, um Serviço é criado. Esse serviço distribui o tráfego entre os Pods, independentemente de onde eles estejam no cluster. Isso significa que o cliente (seja um frontend ou outro serviço) não precisa saber em qual node o Pod está sendo executado.

• **Roteamento de Tráfego**: Quando o tráfego externo chega a um Serviço do tipo LoadBalancer, ele é encaminhado automaticamente para um Pod disponível. Se houver múltiplos Pods para balancear a carga, o Serviço divide o tráfego de maneira eficiente.

Uma aplicação web pode estar distribuída em um cluster Kubernetes com três nodes. Cada node executa múltiplos Pods da aplicação backend. Um Serviço do tipo LoadBalancer é configurado para garantir que o tráfego de usuários seja distribuído igualmente entre os Pods em todos os nodes, garantindo disponibilidade e alta performance, mesmo que um node falhe.

Conteúdo Bônus

Tutorial: “Nodes”

Plataforma: Kubernetes.io

Este tutorial ensina como adicionar, remover e gerenciar nodes dentro de um cluster Kubernetes.

Referências Bibliográficas

BASSO, D. E. **Administração de Redes de Computadores**. Contentus, 2020.

KUROSE, J. F.; ROSS, K. W. **Redes de Computadores e a Internet: Uma Abordagem Top-Down**. 8. ed. Pearson, 2021.

MARINHO, A. L.; CRUZ, J. L. da. (Orgs.). **Desenvolvimento de Aplicações para Internet**. 2. ed. Pearson, 2020.

PUGA, S.; RISSETTI, G. **Lógica de Programação e Estruturas de Dados, com Aplicações em Java**. 3. ed. Pearson, 2016.

ROHLING, L. J. **Segurança de Redes de Computadores**. Contentus, 2020.

SILVA, C. F. da. **Projeto Estruturado e Gerência de Redes**. Contentus, 2020.

TANENBAUM, A. S.; FEAMSTER, N.; WETHERALL, D. J. **Redes de Computadores**. 6. ed. Pearson, 2021.

TOCCI, R. J.; WIDMER, N. S.; MOSS, G. L. **Sistemas Digitais: Princípios e Aplicações**. 12. ed. Pearson, 2018.

Ir para exercício