Orquestração na nuvem com o Kubernetes

1. Cluster Kubernetes e Contêiner Nginx

O Kubernetes permite a orquestração de contêineres em um ambiente de nuvem, onde diferentes serviços e aplicações podem ser gerenciados de forma eficiente. Um exemplo clássico é a implantação de um servidor Nginx dentro de um cluster Kubernetes.

Configuração do Cluster e Contêiner Nginx:

1. Criar um Cluster Kubernetes:

- Usando o Google Kubernetes Engine (GKE), crie um cluster conforme descrito nos tópicos anteriores.

2. Criação do Pod com Nginx:

- Escreva um manifesto YAML para criar um Pod que execute o Nginx:

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
spec:
  containers:
  - name: nginx
    image: nginx:latest
    ports:
    - containerPort: 80
```

3. Implantação do Pod:

Use kubectl apply para implantar o Pod: kubectl apply -f nginx-pod.yaml Implantar um contêiner Nginx em um cluster Kubernetes é uma maneira prática de configurar

rapidamente um servidor web básico para servir conteúdo estático ou atuar como um proxy

reverso.

4. Configuração de Pods e Serviços para Monolith

Muitas aplicações começam como monolitos, onde todos os componentes estão integrados em um único código base. Kubernetes facilita a execução e gerenciamento dessas aplicações monolíticas.

Configuração de Pods e Serviços:

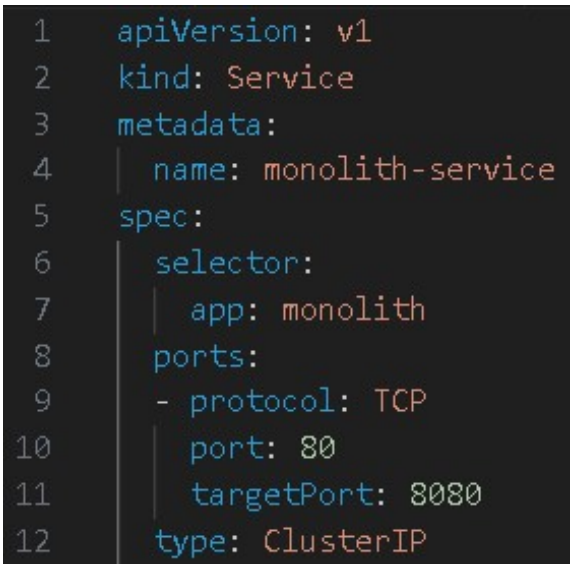
5. Criação do Pod Monolith:

- Crie um manifesto YAML para o Pod que representa o monolito:

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: monolith-pod
spec:
  containers:
  - name: monolith
    image: my-monolith-image:latest
    ports:
    - containerPort: 8080
```

Criação do Serviço:

- Um serviço Kubernetes pode ser criado para expor o Pod internamente no cluster:



```yaml
apiVersion: v1
kind: Service
metadata:
  name: monolith-service
spec:
  selector:
    app: monolith
  ports:
  - protocol: TCP
    port: 80
    targetPort: 8080
  type: ClusterIP
```

Uma aplicação monolítica que serve tanto o frontend quanto o backend pode ser facilmente implantada em um Pod no Kubernetes, com um serviço que a expõe dentro do cluster para outras partes da aplicação.

6. Tráfego para Monolith no NodePort

Para expor um serviço para fora do cluster, Kubernetes oferece a opção de usar o tipo de serviço NodePort.

Configuração do NodePort:

7. Modificar o Serviço Existente:

- Atualize o serviço existente para expor o monolito externamente usando NodePort:

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

8. Acessando o Serviço:

Agora, o serviço pode ser acessado através do IP de qualquer nó no cluster, usando a porta especificada:
http://<NodeIP>:30007

Ao precisar expor uma aplicação monolítica para acesso externo, como um sistema legado que precisa ser integrado a novos serviços, o NodePort fornece um método simples e direto para disponibilizar o serviço.

9. Adição de Rótulos aos Pods

Rótulos (labels) são pares chave-valor que podem ser adicionados a objetos Kubernetes como Pods. Eles são usados para selecionar e agrupar conjuntos de objetos e facilitam a organização e gerenciamento.

Adicionando Rótulos:

10. Adicionar Rótulos no Manifesto do Pod:

- No manifesto YAML, adicione rótulos para o Pod:

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: monolith-pod
  labels:
    app: monolith
    tier: backend
spec:
  containers:
  - name: monolith
    image: my-monolith-image:latest
    ports:
    - containerPort: 8080
```

11. Filtragem de Pods por Rótulos:

Use kubectl para filtrar Pods com base em rótulos:kubectl get pods -l app=monolith

Quando várias versões de uma aplicação estão em execução em diferentes Pods, o uso de rótulos permite identificar rapidamente qual versão ou componente está sendo executado em cada Pod.

12. Criação de Deployments (Autenticação, Hello e Frontend)

Deployments no Kubernetes gerenciam o ciclo de vida de Pods, permitindo escalonamento, atualizações e rollbacks automáticos.

Criação de Deployments:

13. Deployment para Serviço de Autenticação:

- Crie um manifesto YAML para o deployment do serviço de autenticação:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: auth-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: auth
  template:
    metadata:
      labels:
        app: auth
    spec:
      containers:
      - name: auth
        image: my-auth-image:latest
        ports:
        - containerPort: 8080
```

Deployment para Serviço Hello:

- Crie um deployment para o serviço “Hello”:

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

Deployment para o Frontend:

- Crie um deployment para o frontend:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: frontend
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
      - name: frontend
        image: my-frontend-image:latest
        ports:
        - containerPort: 80
```

14. Implantação e Gerenciamento:

- Implante os deployments usando kubectl apply.
- Use kubectl get deployments para verificar o status dos deployments.

Em uma arquitetura de microserviços, onde você tem serviços como autenticação, um serviço “Hello” para saudações, e um frontend, a criação de deployments permite gerenciar cada serviço independentemente, com escalabilidade e resiliência automáticas.

Extra: Monolito e Nginx no Contexto de Kubernetes

Monolito: O Que É e Quando Usá-lo

Monolito é um estilo de arquitetura de software em que todos os componentes de uma aplicação estão integrados e executados como uma única unidade. Em uma aplicação monolítica, o frontend, o backend, a lógica de negócios e o acesso a dados estão todos unidos em um único código base e processo de execução.

- **Características de um Monolito:**

- Código Centralizado: Toda a aplicação reside em um único código base.

- Desenvolvimento e Implantação Unificados: Todas as partes da aplicação são desenvolvidas, testadas e implantadas juntas.

- Facilidade de Desenvolvimento Inicial: Para pequenas equipes e projetos simples, um monolito pode ser mais fácil de desenvolver, testar e implantar.

- Complexidade em Escalonamento: Escalonar uma parte específica da aplicação monolítica pode ser difícil, pois a aplicação inteira precisa ser escalada, mesmo que apenas uma parte esteja sob carga pesada.

- **Quando Usar um Monolito:**
  - Projetos Simples ou Pequenos: Aplicações com uma base de código pequena e uma equipe de desenvolvimento pequena podem se beneficiar da simplicidade de um monolito.

  - Prova de Conceito (PoC): Durante a fase inicial de um projeto, quando a arquitetura ainda não está completamente definida.

  - Ambientes com Requisitos de Baixa Escalabilidade: Se não há necessidade de escalabilidade massiva ou componentes independentes, um monolito pode ser suficiente.

Imagine uma aplicação de gerenciamento de conteúdo para um pequeno blog. Todo o sistema, incluindo a interface do usuário, a lógica de negócios e o acesso ao banco de dados, pode estar em um único código base e ser facilmente gerenciado como um monolito.

Nginx: O Papel do Servidor Web e Proxy Reverso

Nginx é um servidor web de código aberto que também pode ser usado como proxy reverso, balanceador de carga, e cache HTTP. Ele é amplamente utilizado por sua alta performance, escalabilidade e capacidade de lidar com grandes volumes de tráfego.

- **Funções do Nginx:**
  - Servidor Web: Nginx pode servir conteúdo estático, como HTML, CSS, JavaScript e imagens diretamente aos clientes.
  - Proxy Reverso: Redireciona o tráfego de clientes para diferentes servidores ou serviços, equilibrando a carga e melhorando a escalabilidade.
  - Balanceamento de Carga: Distribui o tráfego de entrada entre várias instâncias de uma aplicação, melhorando a disponibilidade e a resiliência.
  - Terminação SSL: Nginx pode gerenciar a terminação SSL, descarregando o trabalho de criptografia/descriptografia dos servidores de aplicação.
- **Nginx e Monolitos em Kubernetes:**
  - Em um cenário monolítico, o Nginx pode ser usado como um proxy reverso para rotear o tráfego de entrada para o único serviço backend da aplicação monolítica.
  - O Nginx também pode atuar como um balanceador de carga dentro do cluster Kubernetes, distribuindo as solicitações entre as réplicas de um Pod monolítico, melhorando a performance e a resiliência.

Suponha que você tenha uma aplicação monolítica que serve tanto a interface do usuário quanto as APIs de backend. O Nginx pode ser

configurado como proxy reverso para redirecionar as solicitações HTTP para as diferentes partes da aplicação, garantindo que o tráfego seja roteado corretamente e que a carga seja balanceada entre várias instâncias.

Integração de Monolito com Nginx em Kubernetes

Ao implantar uma aplicação monolítica em um cluster Kubernetes, você pode usar o Nginx para gerenciar o tráfego e melhorar a escalabilidade e a disponibilidade.

Passos de Integração:

1. Configuração do Pod Monolítico:

- O Pod contém a aplicação monolítica que processa todas as funcionalidades da aplicação.

2. Implantação do Nginx como Proxy Reverso:

- Implante um Pod separado executando o Nginx, configurado para rotear o tráfego para o Pod monolítico.

Exemplo de configuração básica de Nginx:

nginx

```
server {
    listen 80;
    location / {
        proxy_pass http://monolith-service:8080;
        proxy_set_header Host $host;
```

```
        proxy_set_header X-Real-IP $remote_addr;

        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

        proxy_set_header X-Forwarded-Proto $scheme;

    }

}
```

3. Configuração do Serviço:

- Crie um serviço Kubernetes para expor o Nginx, que então rotea as solicitações para o Pod monolítico.

Benefícios:

- Melhor Gerenciamento de Tráfego: O Nginx pode otimizar o fluxo de tráfego, reduzir a latência e melhorar a experiência do usuário.

- Escalabilidade: Usando o Nginx em conjunto com Kubernetes, você pode escalar as instâncias do Pod monolítico para lidar com aumentos de tráfego, enquanto o Nginx gerencia a distribuição das solicitações.

Em um sistema de comércio eletrônico que ainda opera como um monolito, o Nginx pode ser configurado para gerenciar e distribuir o tráfego de clientes entre várias réplicas da aplicação monolítica, garantindo que a aplicação continue a funcionar eficientemente durante picos de tráfego, como em vendas promocionais.

Conteúdo Bônus

Artigo: "Introducing containers"

Plataforma: Cloud.Google

Aborda como o Kubernetes facilita a orquestração de contêineres em ambientes de nuvem, destacando benefícios e desafios.

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