Kubernetes

Para Quem Não Entende Nada

Aula 01

Comandos, Pods,
Deployments e Services

kubectl

O kubectl é o comando padrão de interação com a API Rest do Kubernetes.

```
kubectl --help
```

kubectl

Todos os comandos do kubectl seguem a sintaxe:

kubectl <verbo> <tipo> [nome]

kubectl

Como exemplo, `kubectl get nodes`:

```
NAME       STATUS   ROLES           AGE    VERSION
minikube   Ready    control-plane   2m16s  v1.28.3
```

Este é o nó do minikube, temos apenas uma máquina.

kubectl

Outro exemplo, `kubectl get service`:

```
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   8m44s
```

Endereço interno da API que os componentes utilizam para se comunicar com o kubernetes.

Pod

O Pod é a menor unidade de aplicação do Kubernetes.
A tradução de pod é um conjunto de baleias ou cápsula/casulo.

Pod

```
apiVersion: v1
kind: Pod
metadata:
  name: apache
spec:
  containers:
  - image: docker.io/httpd:alpine
    name: httpd
```

YAML

O YAML é uma linguagem legível de serialização de dados muito usada na escrita de arquivos de configuração.
É fácil de ler e de entender.

Fonte: Red Hat

YAML - Tipos

Dicionário

```yaml
---
curso: kubernetes
alunos: 10
ativo: true
```

Lista

```yaml
---
aulas:
- Introdução
- Pods
- Deployments
```

Combinação

```yaml
---
curso: kubernetes
alunos:
- nome: Hector
  Idade: 33
aulas:
- Introdução
- Pods
```

kubectl - interagindo

Podemos obter muitas informações a respeito do pod apenas com o kubectl:

```
kubectl create -f pod.yml
kubectl get pods
kubectl get pods -o wide
kubectl describe pod apache
kubectl logs apache
kubectl exec -ti apache sh
```

Pod - MySQL

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mysql
spec:
  containers:
  - image: docker.io/mysql:8.3
    name: mysql
```

kubectl - investigando

Às vezes, como qualquer plano na vida, um pode pod dar errado. Nestes casos utilizamos describe e logs para analisá-lo:

```
kubectl describe pod mysql
kubectl logs mysql
Kubectl edit pod mysql
```

Pod - Alterações

Não é possível alterar a maioria das propriedades dos Pods.
Para isso precisaremos de objetos de mais alto nível, como o Deployment ou o Replicaset.

Deployment

O Deployment é uma forma de prover atualizações declarativas para Pods e ReplicaSets.
Um ReplicaSet mantém um número estável de réplicas de um Pod.

Deployment

Deployment -> ReplicaSets -> Pods.
O Deployment atualiza o ReplicaSet que por sua vez atualiza os Pods.
Não é recomendado utilizar ReplicaSets sozinhos.

garbage collector

Deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
spec:
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
      - name: mysql
        image: docker.io/mysql:8.3
```

ReplicaSet

Pod

Labels

Os labels são marcações arbitrárias que colocamos em qualquer objeto do Kubernetes para facilitar sua identificação.

Deployment

Remova o Pod e crie o Deployment:

```
kubectl delete pod mysql
kubectl create deploy mysql --image docker.io/mysql:8.3
```

Deployment

É possível alterar praticamente qualquer campo de um Deployment. No MySQL podemos adicionar as variáveis necessárias para a inicialização.

Recuperação Automática

Self-healing é uma capacidade do Kubernetes de manter uma aplicação funcionando. Remova as variáveis do MySQL e a versão anterior continuará “no ar” enquanto a nova continuará tentando subir.

Balanceamento de Carga

Load balancing é a capacidade do Kubernetes de balancear as requisições entre diferentes pods, evitando sobrecarga da aplicação.

Réplicas

Réplicas são cópias de um mesmo Pod, normalmente em máquinas diferentes. Um Pod com 3 réplicas indica que existem 3 pods iguais ao analisado espalhado pelo cluster.

Réplicas

Crie um novo Deployment com a imagem docker.io/hectorvido/sh-cgi:

```
kubectl create deploy cgi --image docker.io/hectorvido/sh-cgi
```

E adicione mais réplicas:

```
kubectl scale deploy cgi --replicas=3
```

Serviços

Service é uma forma de expor aplicações na rede que estão rodando em um ou mais Pods dentro do cluster.

Service - cgi

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: cgi
  name: cgi
spec:
  ports:
  - port: 8080
    protocol: TCP
    targetPort: 8080
  selector:
    app: cgi
```

Service

Assim como acontece com o Deployment, podemos criar um Service pela cli:

```
kubectl expose deploy/cgi
```

```
kubectl get svc
Kubectl describe svc cgi
```

Service

Entre no minikube com “minikube ssh” e execute o seguinte comando para testar o balanceador (CTRL + C cancela):

```
while true; do curl <ip_service>:8080; sleep 1; done
```

Pod: + de 1 Contêiner

Os Pods podem ter 1 ou mais contêineres definidos dentro de si.
Os contêineres compartilham rede e volumes.
Dentro do pod, comunicam através de localhost.

Pod: + de 1 Contêiner