garbage collector

Kubernetes

Para Quem Não Entende Nada

Aula 03

StorageClass, ConfigMaps,
Secrets e Métricas

StorageClass

Um StorageClass é um objeto que permite a classificação das formas de storage que os administradores oferecem.

ConfigMaps

Um ConfigMap é um objeto utilizado para guardar dados não confidenciais em formato “chave:valor”.
Pods consomem ConfigMaps como variáveis de ambiente ou arquivos em forma de volume.

Secrets

Um Secret é um objeto utilizado para guardar dados sensíveis em formato “chave:valor”.
Pods consomem Secrets como variáveis de ambiente ou arquivos em forma de volume.

CM e Secret - Utilidade

Ambos são similares e permitem que determinadas configurações fiquem guardadas no ambiente ao invés de estarem no código da aplicação.

CM e Secret - Utilidade

Diferenças nas configurações de Produção e Homologação.

Dados sensíveis fora do repositório de códigos.

nginx.conf

A aplicação em lua precisa utilizar o DNS do Kubernetes e não o seu próprio, para isso precisamos modificar o arquivo:

/usr/local/openresty/nginx/conf/nginx.conf

Porém este arquivo é somente leitura no contêiner.

ConfigMap — nginx-conf

https://github.com/hector-vido/lua-app

container → nginx-resolver-local.conf

Habilitar o lua_code_cache
Renomeá-lo como nginx.conf

```
$ kubectl create cm nginx-conf --from-file nginx.conf
```

ConfigMap — nginx-conf

```
containers:
...
  ports:
  - containerPort: 8080
    protocol: TCP
  volumeMounts:
  - name: nginx-conf
    mountPath: /usr/local/openresty/nginx/conf/nginx.conf
    subPath: nginx.conf
volumes:
- name: nginx-conf
  configMap:
    name: nginx-conf
dnsPolicy: ClusterFirst
```

Secret — mysql-env

```
MYSQL_ROOT_PASSWORD=kub3rn3t3s
MYSQL_USER=lua
MYSQL_PASSWORD=lua
MYSQL_DATABASE=lua
MYSQL_HOST=mysql.default.svc.cluster.local
MYSQL_PORT=3306
```

```
$ kubectl create secret generic mysql-env --from-env-file mysql.env
```

Secret – mysql-env

```
containers:
- name: mysql
  image: docker.io/mysql:8.3
- envFrom:
  - secretRef:
      name: mysql-env
```

MySQL

Lua

```
containers:
- name: lua
  image: docker.io/hectorvido/lua-app
- envFrom:
  - secretRef:
      name: mysql-env
```

Métricas

Podemos consultar as métricas dos nós e dos pods do Kubernetes, mas para isso precisamos instalar o metrics-server:

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

metrics-server

O metrics-server é um pod comum, e neste caso podemos visualizá-lo e editá-lo em kube-system:

```
$ kubectl get pods -n kube-system
NAME                               READY   STATUS    RESTARTS      AGE
coredns-5dd5756b68-cq2f6           1/1     Running   2 (12m ago)   37h
etcd-minikube                      1/1     Running   2 (12m ago)   37h
kube-apiserver-minikube            1/1     Running   2 (12m ago)   37h
kube-controller-manager-minikube   1/1     Running   2 (12m ago)   37h
kube-proxy-rdvk6                   1/1     Running   2 (12m ago)   37h
kube-scheduler-minikube            1/1     Running   2 (12m ago)   37h
metrics-server-7c66d45ddc-gnkv4    1/1     Running   0             9m3s
storage-provisioner                1/1     Running   4 (11m ago)   37h
```

Consular métricas

Para consultar as métricas utilizamos:

```
$ kubectl top nodes
```

Mostra o consumo de memória e CPU dos nós.

```
$ kubectl top pods
```

Mostra o consumo de memória e CPU dos pods.

Requests and Limits

É uma boa prática – e obrigatório para AutoScaling – configurarmos requests e limits para as nossas aplicações. Aplicamos estas configurações para CPU e memória.

Requests and Limits

Requests é o mínimo de recursos que a aplicação precisa para funcionar em um nó.
Limits é o limite de recursos que ela utilizará.

Requests and Limits

```
containers:
- name: mysql-env
  image: docker.io/hectorvido/lua-app
  name: lua-app
  resources:
    limits:
      cpu: 500m
      memory: 16Mi
    requests:
      cpu: 100m
      memory: 8Mi
```

HorizontalPodAutoscaler

O HPA é um objeto que atualiza automaticamente um recurso com o intuito de dimensioná-lo com base na demanda.

garbage collector

HorizontalPodAutoscaler

O cálculo para dimensar os pods acontece em cima da propriedade `resources.requests` do pod.

```
T1$ kubectl autoscale deploy/lua --min=1 --max=3 --cpu-percent=70

T1$ while true; do curl lua.192-168-39-103.nip.io; sleep 0.1; done

T2$ kubectl get hpa

T3$ kubectl top pods
```