Kubernetes Operator 开发与部署

Operator介绍

[Kubernetes Operator](https://www.redhat.com/zh/resources/oreilly-kubernetes-operators-automation-ebook?intcmp=701f2000001OMH6AAO) 是一种封装、部署和管理 Kubernetes应用的方法。我们使用

Kubernetes API（应用编程接口）和 kubectl 工具在 [Kubernetes](https://www.redhat.com/zh/topics/containers/what-is-kubernetes) 上部署并管理 Kubernetes 应用。Kubernetes Operator 是一种特定于应用的控制器，可扩展 Kubernetes API 的功能，来代表 Kubernetes 用户创建、配置和管理复杂应用的实例。它基于基本 Kubernetes 资源和控制器概念构建，但又涵盖了特定于域或应用的知识，用于实现其所管理软件的整个生命周期的自动化。(Redhat)

Kubernetes CRD

Custom resources：是K8S API的扩展,属于用户对Kubernetes资源(Kind)的定制化部署。自定义资源可以动态注册到Kubernetes；用户则通过kubelet客户端创建和访问这个自定义的对象。

Custom controllers：自定义控制器根据用户的需求控制与管理自定义资源，并且可以维护和更新自定义资源状态。

[Operator Framework](https://github.com/operator-framework/getting-started)是属于CRD开发的一种工具,提供以下工作流开发Operator：

1. 使用 SDK 创建一个新的 Operator 项目
2. 通过添加自定义资源（CRD）定义新的资源 API
3. 指定使用 SDK API 来 watch 的资源
4. 定义 Operator 的协调（reconcile）逻辑
5. 使用 Operator SDK 构建并生成 Operator 部署清单文件

进入证书目录(Master-1)

[root@master-1 ~]# cd /root/kubernetes

#配置请求文件

[root@master-1 kubernetes]# cat opsuser-csr.json

{

"CN": "opsuser",

"hosts": [],

"key": {

"algo": "rsa",

"size": 2048

},

"names": [

{

"C": "CN",

"ST": "BeiJing",

"L": "BeiJing",

"O": "k8s",

"OU": "System"

}

]

}

#生成客户端证书

[root@master-1 kubernetes]# cfssl gencert -ca=ca.pem -ca-key=ca-key.pem \

-config=ca-config.json \

-profile=kubernetes opsuser-csr.json | cfssljson -bare opsuser

#复制证书

[root@master-1 kubernetes]# cp /root/kubernetes/opsuser.pem /etc/kubernetes/ssl/

[root@master-1 kubernetes]# cp /root/kubernetes/opsuser-key.pem /etc/kubernetes/ssl/

#配置授权

[root@master-1 kubernetes]# cd /root/config

# 设置集群参数

[root@master-1 kubernetes]# export KUBE\_APISERVER="https://192.168.91.254:6443"

[root@master-1 kubernetes]# kubectl config set-cluster kubernetes \

--certificate-authority=/etc/kubernetes/ssl/ca.pem \

--embed-certs=true \

--server=${KUBE\_APISERVER} \

--kubeconfig=opsuser.kubeconfig

# 设置客户端认证参数

[root@master-1 kubernetes]# kubectl config set-credentials opsuser \

--client-certificate=/etc/kubernetes/ssl/opsuser.pem \

--client-key=/etc/kubernetes/ssl/opsuser-key.pem \

--embed-certs=true \

--kubeconfig=opsuser.kubeconfig

# 设置上下文参数

[root@master-1 kubernetes]# kubectl config set-context kubernetes \

--cluster=kubernetes \

--user=opsuser \

--kubeconfig=opsuser.kubeconfig

# 设置默认上下文

[root@master-1 kubernetes]# kubectl config use-context \

kubernetes --kubeconfig=opsuser.kubeconfig

#配置权限

[root@master-1 kubernetes]# kubectl create clusterrolebinding kubernetes \

--clusterrole=cluster-admin --user=opsuser

#测试指定配置文件连接

[root@master-1 config]# kubectl get pod --kubeconfig=/root/config/opsuser.kubeconfig

#scp认证文件到master-2

[root@master-1 config]# scp /root/config/opsuser.kubeconfig \

master-2:/root/opsuser.kubeconfig

#安装operator(master-2)

[root@master-2 ~]# yum install yum-plugin-fastestmirror -y

[root@master-2 ~]# wget -O /etc/yum.repos.d/epel-7.repo \

http://mirrors.aliyun.com/repo/epel-7.repo

[root@master-2 ~]# yum install golang-1.15.2-1.el7.x86\_64 -y

[root@master-2 ~]# yum localinstall go-dep-0.5.4-1.el7.harbottle.x86\_64.rpm

[root@master-2 ~]# echo "export GOROOT=/usr/lib/golang/" >> /etc/profile

[root@master-2 ~]# source /etc/profile

[root@master-2 ~]# mkdir -p /opt/operator/src

[root@master-2 ~]# export GOPATH=/opt/operator

#下载依赖

[root@master-2 ~]# export GO111MODULE=on

[root@master-2 ~]# go env -w GO111MODULE=on

#配置proxy

[root@master-2 ~]# go env -w GOPROXY=https://goproxy.cn,direct

#编译安装(master-2)

[root@master-2 ~]# cd /soft/operator-sdk-0.12.x

[root@master-2 ~]# make install

[root@master-2 ~]# ln -s /opt/operator/bin/operator-sdk /usr/bin/

#创建passport项目

[root@master-2 ~]# cd /opt/operator/src

[root@master-2 src]# operator-sdk new passport

[root@master-2 src]# cd passport

#passport 生成的项目清单文件

Gopkg.toml Gopkg.lock — Go Dep 清单，用来描述当前 Operator 的依赖包。

cmd - 包含 main.go 文件，使用 operator-sdk API 初始化和启动当前 Operator 的入口。

deploy - 包含一组用于在 Kubernetes 集群上进行部署的通用的 Kubernetes 资源清单文件。

pkg/apis - 包含定义的 API 和自定义资源（CRD）的目录树，这些文件允许 sdk 为 CRD 生成代码并注册对应的类型，以便正确解码自定义资源对象。

pkg/controller - 用于编写所有的操作业务逻辑的地方

vendor - golang vendor 文件夹，其中包含满足当前项目的所有外部依赖包，通过 go dep 管理该目录。

build 为项目构建目录

#添加API

[root@master-2 passport]# operator-sdk add api \

--api-version=app.tony.com/v1 --kind=SpringCloud

#添加控制器

[root@master-2 passport]# operator-sdk add controller \

--api-version=app.tony.com/v1 --kind=SpringCloud

#复制认证文件到master-2

[root@master-2 passport]#scp \

192.168.91.18:/root/config/opsuser.kubeconfig /opt/operator/src/passport

#创建CRD

[root@master-2 ~]# cd /opt/operator/src/passport

[root@master-2 passport]# kubectl apply \

-f deploy/crds/app.tony.com\_springclouds\_crd.yaml

#启动项目

[root@master-2 passport]#cd /opt/operator/src/passport

[root@master-2 passport]#operator-sdk up \

local --kubeconfig=/opt/operator/src/passport/opsuser.kubeconfig

#安装samba

[root@master-2 ~]#yum install -y samba

#添加用户

[root@master-2 ~]#adduser public

[root@master-2 ~]#smbpasswd -a public #设置密码

#修改配置文件

[root@master-2 ~]#vi /etc/samba/smb.conf

[share]

comment = my share

path = /opt/operator

valid user = public

writable = yes

browseable = yes

create mode = 0777

directory mode = 0777

#启动smb

#启动

[root@master-2 ~]#chkconfig smb on

[root@master-2 ~]#service smb restart

[root@master-2 ~]#chown public.public /opt/operator/ -R

#Windows客户端访问

\\192.168.91.19\share

#编写Operator

#添加接口

#生成代码

[root@master-2 passport]#operator-sdk generate k8s

##创建服务

[root@master-19 deploy]# cat crds/app.tony.com\_v1\_springcloud\_passport.yaml

apiVersion: app.tony.com/v1

kind: SpringCloud

metadata:

name: app-passport

spec:

replicas: 1

image: repo.hostscc.com/springcloud/passport:v1.0

clusterip: 10.0.0.101

imagePullSecrets: "image-se"

initContainersImage: skywalking-agent-sidecar:6.1.0

ports:

- port: 8086

targetPort: 8086

#Node导入镜像

#导入镜像(所有Node节点)

[root@node-1 ~]# docker load -i provider-passport.v1.0.tar.gz

[root@node-1 ~]# docker load -i openjdk8.tar.gz

[root@node-1 ~]# docker load -i skywalking-agent-sidecar.6.1.0.tar.gz

#测试启动(master-2)

[root@master-19 passport]# kubectl apply \

-f crds/app.tony.com\_v1\_springcloud\_passport.yaml

[root@master-19 passport]# operator-sdk up \

local --kubeconfig=/opt/operator/src/passport/opsuser.kubeconfig

#部署项目

#构建镜像

[root@master-19 passport]# cd /opt/operator/src/passport

[root@master-19 passport]# operator-sdk build daemon-process

#导出镜像 daemon-process

#传到node节点

[root@master-19 passport]# cd /opt/operator/src/passport

[root@master-19 passport]# docker save -o daemon-process.tar.gz daemon-process:latest

[root@master-19 passport]# scp daemon-process.tar.gz 192.168.91.21:/soft/

[root@master-19 passport]# scp daemon-process.tar.gz 192.168.91.22:/soft/

#修改镜像名称

[root@master-19 passport]# sed \

-i 's|REPLACE\_IMAGE|daemon-process|g' deploy/operator.yaml

#启动之前修改镜像策略imagePullPolicy: IfNotPresent

[root@master-19 passport]# sed \

-i 's|imagePullPolicy: Always|imagePullPolicy: IfNotPresent|g' deploy/operator.yaml

#创建crd与权限

[root@master-19 passport]# kubectl apply \

-f deploy/crds/app.tony.com\_springclouds\_crd.yaml

[root@master-19 passport]# kubectl create -f deploy/service\_account.yaml

[root@master-19 passport]# kubectl create -f deploy/role.yaml

[root@master-19 passport]# kubectl create -f deploy/role\_binding.yaml

#导入镜像到node 节点

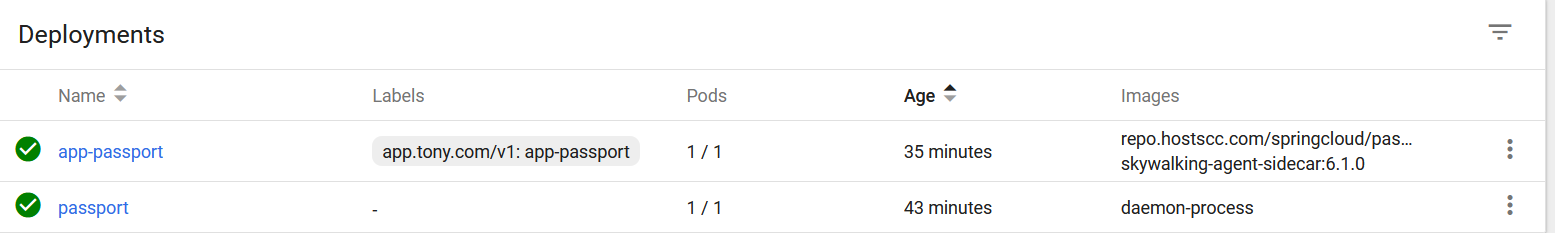
[root@node-1 ~]# docker load -i daemon-process.tar.gz

#创建服务

[root@master-19 passport]# kubectl apply \

-f deploy/crds/app.tony.com\_v1\_springcloud\_passport.yaml

#服务查看



#查看注册中心

