# 云计算自动化运维开发

# **容器云运维开发**

## **基于Kubernetes Restful API实现Deployment创建**

在提供的OpenStack私有云平台上，使用k8s-python-dev镜像创建1台云主机，云主机类型使用4vCPU/12G内存/100G硬盘。该主机中已经默认安装了所需的开发环境，登录默认账号密码为“root/1DaoYun@2022”。

使用Kubernetes Restful API库，在/root目录下，创建api\_deployment\_manager.py文件，要求编写python代码，代码实现以下任务：

（1）编写Python程序实现Deployment资源的创建。Deployment配置信息如下。如果同名Deployment存在，先删除再创建。

（2）创建完成后，查询该Deployment的详细信息，执行结果控制台输出，以yaml格式展示。

创建Deployment 的yaml的配置如下：

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

labels:

app: nginx

spec:

replicas: 3

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:1.15.4

ports:

- containerPort: 80

# ===========================================

# Copyright Jiangsu One Cloud Technology Development Co. LTD. All Rights Reserved.

# 版权： 江苏一道云科技发展有限公司，版权所有！

# ===========================================

import requests,json,yaml,time

import urllib3

urllib3.disable\_warnings(urllib3.exceptions.InsecureRequestWarning)

def get\_token(api\_token,):

    bearer\_token = "bearer " + api\_token

    return bearer\_token

class dep\_manager:

    def \_\_init\_\_(self,node\_url,bearer\_token):

        self.node\_url = node\_url

        self.bearer\_token = bearer\_token

    def create\_dep(self,yamlFile,namespace):

        headers = {

            "Content-Type": "application/json",

            "Authorization": self.bearer\_token

        }

        with open(yamlFile, encoding='utf-8') as f:

            body = json.dumps(yaml.safe\_load(f))

        url = self.node\_url + "/apis/apps/v1/namespaces/" + namespace + "/deployments"

        req = json.loads(requests.post(url, headers=headers, data=body, verify=False).text)

        return req

    def get\_dep(self,dep\_name,namespace):

        headers = {

            "Authorization": self.bearer\_token

        }

        url = self.node\_url + "/apis/apps/v1/namespaces/" + namespace + "/deployments/" + dep\_name

        req = json.loads(requests.get(url, headers=headers, verify=False).text)

        return req

if \_\_name\_\_ == "\_\_main\_\_":

    #token换成本机的token，创建token方法如下

    #kubectl create serviceaccount admin1 -n kube-system

    #kubectl create clusterrolebinding admin1 --clusterrole=cluster-admin --serviceaccount=kube-system:admin1

    api\_token = "换成创建好的token"

    node\_url = "https://{换成本机的IP}:6443"

    bearer\_token = get\_token(api\_token, )

    dep\_m = dep\_manager(node\_url,bearer\_token)

    #create

dep\_create = dep\_m.create\_dep("nginx-deployment.yaml","default")

time.sleep(10)

    print(f"创建deployment信息为:{dep\_create}")

    #get

    get\_dep = dep\_m.get\_dep("nginx-dep","default")

print(f"获取到Deployment的信息为{get\_dep}")

------------------------------------------------执行结果----------------------------------------------------------

创建deployment信息为:{'kind': 'Deployment', 'apiVersion': 'apps/v1', 'metadata': {'name': 'nginx-deployment', 'namespace': 'default', 'uid': '3b526406-895e-4c05-86a8-636c4e857f70', 'resourceVersion': '65978', 'generation': 1, 'creationTimestamp': '2022-10-11T02:17:46Z', 'labels': {'app': 'nginx'}, 'managedFields': [{'manager': 'python-requests', 'operation': 'Update', 'apiVersion': 'apps/v1', 'time': '2022-10-11T02:17:46Z', 'fieldsType': 'FieldsV1', 'fieldsV1': {'f:metadata': {'f:labels': {'.': {}, 'f:app': {}}}, 'f:spec': {'f:progressDeadlineSeconds': {}, 'f:replicas': {}, 'f:revisionHistoryLimit': {}, 'f:selector': {}, 'f:strategy': {'f:rollingUpdate': {'.': {}, 'f:maxSurge': {}, 'f:maxUnavailable': {}}, 'f:type': {}}, 'f:template': {'f:metadata': {'f:labels': {'.': {}, 'f:app': {}}}, 'f:spec': {'f:containers': {'k:{"name":"nginx"}': {'.': {}, 'f:image': {}, 'f:imagePullPolicy': {}, 'f:name': {}, 'f:ports': {'.': {}, 'k:{"containerPort":80,"protocol":"TCP"}': {'.': {}, 'f:containerPort': {}, 'f:protocol': {}}}, 'f:resources': {}, 'f:terminationMessagePath': {}, 'f:terminationMessagePolicy': {}}}, 'f:dnsPolicy': {}, 'f:restartPolicy': {}, 'f:schedulerName': {}, 'f:securityContext': {}, 'f:terminationGracePeriodSeconds': {}}}}}}]}, 'spec': {'replicas': 3, 'selector': {'matchLabels': {'app': 'nginx'}}, 'template': {'metadata': {'creationTimestamp': None, 'labels': {'app': 'nginx'}}, 'spec': {'containers': [{'name': 'nginx', 'image': 'nginx:1.15.4', 'ports': [{'containerPort': 80, 'protocol': 'TCP'}], 'resources': {}, 'terminationMessagePath': '/dev/termination-log', 'terminationMessagePolicy': 'File', 'imagePullPolicy': 'IfNotPresent'}], 'restartPolicy': 'Always', 'terminationGracePeriodSeconds': 30, 'dnsPolicy': 'ClusterFirst', 'securityContext': {}, 'schedulerName': 'default-scheduler'}}, 'strategy': {'type': 'RollingUpdate', 'rollingUpdate': {'maxUnavailable': '25%', 'maxSurge': '25%'}}, 'revisionHistoryLimit': 10, 'progressDeadlineSeconds': 600}, 'status': {}}

获取到Deployment的信息为{'kind': 'Deployment', 'apiVersion': 'apps/v1', 'metadata': {'name': 'nginx-deployment', 'namespace': 'default', 'uid': '3b526406-895e-4c05-86a8-636c4e857f70', 'resourceVersion': '66033', 'generation': 1, 'creationTimestamp': '2022-10-11T02:17:46Z', 'labels': {'app': 'nginx'}, 'annotations': {'deployment.kubernetes.io/revision': '1'}, 'managedFields': [{'manager': 'python-requests', 'operation': 'Update', 'apiVersion': 'apps/v1', 'time': '2022-10-11T02:17:46Z', 'fieldsType': 'FieldsV1', 'fieldsV1': {'f:metadata': {'f:labels': {'.': {}, 'f:app': {}}}, 'f:spec': {'f:progressDeadlineSeconds': {}, 'f:replicas': {}, 'f:revisionHistoryLimit': {}, 'f:selector': {}, 'f:strategy': {'f:rollingUpdate': {'.': {}, 'f:maxSurge': {}, 'f:maxUnavailable': {}}, 'f:type': {}}, 'f:template': {'f:metadata': {'f:labels': {'.': {}, 'f:app': {}}}, 'f:spec': {'f:containers': {'k:{"name":"nginx"}': {'.': {}, 'f:image': {}, 'f:imagePullPolicy': {}, 'f:name': {}, 'f:ports': {'.': {}, 'k:{"containerPort":80,"protocol":"TCP"}': {'.': {}, 'f:containerPort': {}, 'f:protocol': {}}}, 'f:resources': {}, 'f:terminationMessagePath': {}, 'f:terminationMessagePolicy': {}}}, 'f:dnsPolicy': {}, 'f:restartPolicy': {}, 'f:schedulerName': {}, 'f:securityContext': {}, 'f:terminationGracePeriodSeconds': {}}}}}}, {'manager': 'kube-controller-manager', 'operation': 'Update', 'apiVersion': 'apps/v1', 'time': '2022-10-11T02:17:48Z', 'fieldsType': 'FieldsV1', 'fieldsV1': {'f:metadata': {'f:annotations': {'.': {}, 'f:deployment.kubernetes.io/revision': {}}}, 'f:status': {'f:availableReplicas': {}, 'f:conditions': {'.': {}, 'k:{"type":"Available"}': {'.': {}, 'f:lastTransitionTime': {}, 'f:lastUpdateTime': {}, 'f:message': {}, 'f:reason': {}, 'f:status': {}, 'f:type': {}}, 'k:{"type":"Progressing"}': {'.': {}, 'f:lastTransitionTime': {}, 'f:lastUpdateTime': {}, 'f:message': {}, 'f:reason': {}, 'f:status': {}, 'f:type': {}}}, 'f:observedGeneration': {}, 'f:readyReplicas': {}, 'f:replicas': {}, 'f:updatedReplicas': {}}}, 'subresource': 'status'}]}, 'spec': {'replicas': 3, 'selector': {'matchLabels': {'app': 'nginx'}}, 'template': {'metadata': {'creationTimestamp': None, 'labels': {'app': 'nginx'}}, 'spec': {'containers': [{'name': 'nginx', 'image': 'nginx:1.15.4', 'ports': [{'containerPort': 80, 'protocol': 'TCP'}], 'resources': {}, 'terminationMessagePath': '/dev/termination-log', 'terminationMessagePolicy': 'File', 'imagePullPolicy': 'IfNotPresent'}], 'restartPolicy': 'Always', 'terminationGracePeriodSeconds': 30, 'dnsPolicy': 'ClusterFirst', 'securityContext': {}, 'schedulerName': 'default-scheduler'}}, 'strategy': {'type': 'RollingUpdate', 'rollingUpdate': {'maxUnavailable': '25%', 'maxSurge': '25%'}}, 'revisionHistoryLimit': 10, 'progressDeadlineSeconds': 600}, 'status': {'observedGeneration': 1, 'replicas': 3, 'updatedReplicas': 3, 'readyReplicas': 3, 'availableReplicas': 3, 'conditions': [{'type': 'Available', 'status': 'True', 'lastUpdateTime': '2022-10-11T02:17:48Z', 'lastTransitionTime': '2022-10-11T02:17:48Z', 'reason': 'MinimumReplicasAvailable', 'message': 'Deployment has minimum availability.'}, {'type': 'Progressing', 'status': 'True', 'lastUpdateTime': '2022-10-11T02:17:48Z', 'lastTransitionTime': '2022-10-11T02:17:46Z', 'reason': 'NewReplicaSetAvailable', 'message': 'ReplicaSet "nginx-deployment-746ccc65d8" has successfully progressed.'}]}}

## **基于Kubernetes Python SDK实现Job创建**

在前面已建好的Kubernetes开发环境云平台上。使用Kubernetes python SDK的“kubernetes”Python库，在/root目录下，创建sdk\_job\_manager.py文件，要求编写python代码，代码实现以下任务：

（1）编写Python程序实现Job资源的创建。Job配置信息如下。如果同名Job存在，先删除再创建。

（2）创建完成后，查询该Job的详细信息，执行结果控制台输出，以json格式展示。

Job创建yaml的信息如下：

apiVersion: batch/v1

kind: Job

metadata:

name: pi

spec:

template:

spec:

containers:

- name: pi

image: perl

command: ["perl", "-Mbignum=bpi", "-wle", "print bpi(2000)"]

restartPolicy: Never

backoffLimit: 4

# ===========================================

# Copyright Jiangsu One Cloud Technology Development Co. LTD. All Rights Reserved.

# 版权： 江苏一道云科技发展有限公司，版权所有！

# ===========================================

import json

import os

import yaml

from kubernetes import client, config

from os import path

class job\_manager():

    def \_\_init\_\_(self, config\_file):

        # 传入配置文件

        config.load\_kube\_config(config\_file)

        # 获取API,管理Job

        self.api = client.BatchV1Api()

    #创建job

    def create\_job(self, yamlFile):

        v1 = self.api

        # 获取当前文件的绝对路径

        fileNamePath = os.path.split(os.path.realpath(\_\_file\_\_))[0]

        # 获取yaml配置文件的路径

        yamlPath = os.path.join(fileNamePath, yamlFile)

        # 读取yaml文件，并转化为JSON数据

        print("-------create job-------------")

        with open(yamlPath, encoding="utf8") as f:

            result = yaml.safe\_load(f)  # 转化成JSON格式

            resp = v1.create\_namespaced\_job(namespace="default",body=result)

            print(resp)

    #查看Job

    def get\_job(self):

        v1 = self.api

        print("-------read job-------------")

        resp = v1.read\_namespaced\_job(name="pi", namespace="default")

        print(resp)

    #删除Job

    def delete\_job(self):

        v1 = self.api

        resp = v1.delete\_namespaced\_job(name="pi", namespace="default", propagation\_policy='Background',)

        print(resp)

if \_\_name\_\_ == '\_\_main\_\_':

    job\_manager(config\_file="config").create\_job(yamlFile="spec-pi-job.yaml")

job\_manager(config\_file="config").get\_job()

------------------------------------------------执行结果----------------------------------------------------------

-------create job-------------

{'api\_version': 'batch/v1',

'kind': 'Job',

'metadata': {'annotations': None,

'cluster\_name': None,

'creation\_timestamp': datetime.datetime(2022, 10, 11, 2, 35, 46, tzinfo=tzutc()),

.......

'name': 'pi',

'namespace': 'default',

'owner\_references': None,

'resource\_version': '67328',

'self\_link': None,

'uid': 'd68f1112-a523-4846-836f-c69e3d96b159'},

.......