3.1.1 OpenShift API 的调用

在开发的过程中，有时候我们需要调用 OpenShift 的 Master API 来完成一些任务。OpenShift 包括 Kubernetes v1 REST API 和 OpenShift v1 REST API 两类 API，主要是为了保证与 Kubernetes 的兼容性，也就是说大部分的 Kubernetes API 是可以直接在 OpenShift 上调用的。

1. OpenShift API 的认证

访问 OpenShift API 需要提供认证，支持以下两种方式：

- OAuth Access Tokens：使用 OpenShift 内的 OAuth server 颁发 Access Token 认证，可以通过用户登陆或者 API 获取。
- X.509 Client Certificates：可以通过证书认证，证书大多数用于集群组件向 API Server 认证。

任何具有无效 Token 或无效证书的请求都将被身份验证层拒绝，并返回出现 401 错误。

（1） 证书认证

默认在 OpenShift 安装过程中，已经生成了 cluster-admin 权限的证书，保存在 /etc/origin/master 目录下，调用 API 的示例如下：

curl -v --cacert /etc/origin/master/ca.crt --cert /etc/origin/master/admin.crt --key /etc/origin/master/admin.key https://master.example.com:8443/api/v1/namespaces

（2） Token 认证

OpenShift 同时也提供了使用 Token 认证访问 API 的方式，需要通过 Authorization: Bearer 的 header 传入 Token，提供了两种类型的 Token：Session Token 、Service Account Token。

1）获取 Session Token

Session Token 是临时有效的，默认有效期为 24 小时，Session Token 代表一个用户，可以通过命令行用户登陆或者 API 调用 Oauth Server API 获取 Session Token。

● 命令行登陆获取 Token

获取 Session Token 的方法如下：

首先使用 oc 客户端执行用户登陆

oc login

Authentication required for https://master.example.com:8443 (openshift)

Username: admin

Password:

Login successful.

使用命令行获取 Token：

oc whoami -t

ydqLcGjJsdpyO79bJxRo_D2qT9jobsNdYqu4mV5iUv0

登陆的用户具有什么权限，这个 Token 就有什么权限。

● 调用 Oauth Server API 获取

通过调用 Oauth Server API 获取 Token，同样代表一个用户，主要用于在程序中模拟用户登陆获取 Token。通过这种方法获取 Token 的 URL 写法有很多，我们仅列出一种：

curl -k -L -u admin:admin

'https://master.example.com:8443/oauth/authorize?response_type=code&client_id=openshift-browser-client'

……

    <h2>Your API token is</h2>

    <code>oGlenZSmvJOgjvfgkWnlt9tZ_9carJ_55u9rCeMbBI0</code>

……

2）获取 Service Account Token

Service Account Token 是长期有效的，代表着一个 Service Account，Service Account 具备什么权限，Token 就具备什么权限。由于 Service Account 默认是属于某个 Namespace 的，可执行的操作在 Namespace 中，除非为 Service Account 赋予集群级别的权限，如

cluster-admin、cluster-viewer 等。获取 SA token 的方法如下：

首先进入一个项目中

oc project myproject

创建一个 Service Account：

oc create serviceaccount davidtest

serviceaccount/davidtest created

为 Service Account 赋予 cluster-admin 权限

oc adm policy add-cluster-role-to-user cluster-admin -z davidtest

role "cluster-admin" added: "davidtest"

获取 sa token：

oc serviceaccounts get-token davidtest

eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm………

2. OpenShift API 调用演示

获取到 Token 之后，就可以使用 HTTP 请求操作 OpenShift API 了。

为了方便，将 Token 和 Master URL 设置为环境变量：

TOKEN=ydqLcGjJsdpyO79bJxRo_D2qT9jobsNdYqu4mV5iUv0

ENDPOINT=master.example.com:8443

首先，我们通过 API 的方式，创建一个名为 redhat 的项目

curl -k -v -XPOST -H "Authorization: Bearer $TOKEN" -H "Accept: application/json"

-H "Content-Type: application/json"

https://$ENDPOINT/apis/project.openshift.io/v1/projectrequests -d

"{\"kind\":\"ProjectRequest\",\"apiVersion\":\"project.openshift.io/v1\",\"metadata\":{\"name\":\"r

edhat\",\"creationTimestamp\":null}}"

从返回结果看，项目创建成功，如下图 3-27 所示：

图 3-27 创建项目成功

利用 Token 查看刚刚创建的 redhat 项目：

curl -k -H "Authorization: Bearer $TOKEN" -H 'Accept: application/json'
https://$ENDPOINT/api/v1/watch/namespaces/redhat

{"type":"ADDED","object":{"kind":"Namespace","apiVersion":"v1","metadata":{"name":"redhat","selfLink":"/api/v1/namespaces/redhat","uid":"ccf6148c-8a09-11e9-8082-000c2981d8ae","resourceVersion":"1111281","creationTimestamp":"2019-06-08T16:23:50Z","annotations":{"alm-manager":"operator-lifecycle-manager.olm-operator","openshift.io/description":"","openshift.io/display-name":"","openshift.io/requester":"admin","openshift.io/sa.scc.mcs":"s0:c19,c9","openshift.io/sa.scc.supplemental-groups":"1000360000/10000","openshift.io/sa.scc.uid-range":"1000360000/10000"}},"spec":{"finalizers":["kubernetes"]},"status":{"phase":"Active"}}
}

查看 default 项目中的 DeploymentConfig：

curl -k -v -XGET -H "Authorization: Bearer $TOKEN" -H "Accept: application/json"
https://master.example.com:8443/apis/apps.openshift.io/v1/namespaces/default/deploymentconfigs |grep -i namespaces

< HTTP/1.1 200 OK

< Cache-Control: no-store

< Content-Type: application/json

< Date: Sun, 09 Jun 2019 05:42:13 GMT

< Transfer-Encoding: chunked

<

{ [data not shown]

100 20996   0 20996   0   0   117k   0 --:--:-- --:--:-- --:--:--   117k

* Connection #0 to host master.example.com left intact

"selfLink": "/apis/apps.openshift.io/v1/namespaces/default/deploymentconfigs",

"selfLink":

"/apis/apps.openshift.io/v1/namespaces/default/deploymentconfigs/docker-registry",

"selfLink":

"/apis/apps.openshift.io/v1/namespaces/default/deploymentconfigs/registry-console",

"selfLink":

"/apis/apps.openshift.io/v1/namespaces/default/deploymentconfigs/router",

从返回结果可以看到 default 项目中有 3 个 DeploymenConfig：docker-registry、registry-console、router，上面结果得到的信息，和我们直接调用 oc 命令查看的内容是一样的。

我们查看 default 项目中的所有 Pod，由于信息较多，我们只展示 Router Pod 的内容：

curl -k \

    -H "Authorization: Bearer $TOKEN" \

    -H 'Accept: application/json' \

https://$ENDPOINT/api/v1/namespaces/default/pods

从返回结果，可以看到 Router 的 Pod 名称为：router-3-v2xpt

```
{
  "metadata": {
    "name": "router-3-v2xpt",
    "generateName": "router-3-",
    "namespace": "default",
    "selfLink": "/api/v1/namespaces/default/pods/router-3-tnm7f",
```

```
      "uid": "1ed9648e-d442-11e8-8fe0-000c2981d8ae",

      "resourceVersion": "782098",

      "creationTimestamp": "2018-10-20T08:28:28Z",

      "labels": {

        "deployment": "router-3",

        "deploymentconfig": "router",

        "router": "router"

      },

      "annotations": {

        "openshift.io/deployment-config.latest-version": "3",

        "openshift.io/deployment-config.name": "router",

        "openshift.io/deployment.name": "router-3",

        "openshift.io/scc": "hostnetwork"

      },

      "ownerReferences": [

        {

          "apiVersion": "v1",

          "kind": "ReplicationController",

          "name": "router-3",

          "uid": "1b93ad47-d442-11e8-8fe0-000c2981d8ae",

          "controller": true,

          "blockOwnerDeletion": true

        }

      ]

    },
```

通过 API 删除 Router Pod

```
# curl -k \
    -X DELETE \
    -d @- \
    -H "Authorization: Bearer $TOKEN" \
```

-H 'Accept: application/json' \

-H 'Content-Type: application/json' \

https://$ENDPOINT/api/v1/namespaces/default/pods/router-3-v2xpt <<'EOF'

{

"body":"v1.DeleteOptions"

}

执行命令以后，我们用 oc 命令进行验证。发现旧的 Router Pod 正在终止、新的 Pod 正在创建，如下图 3-28 所示：


```
}[root@node ~]#oc get pods
NAME                                          READY   STATUS            RESTARTS   AGE
couchbase-operator-85c97bb74c-hljlf           0/1     Running           9          34d
docker-registry-1-2nd2n                       1/1     Running           11         48d
etcd-operator-7b49974f5b-p5c6c                3/3     Running           39         34d
mongodb-enterprise-operator-7b7b8b9889-nmbfn  0/1     ImagePullBackOff  0          34d
registry-console-1-td87b                      1/1     Running           12         48d
router-3-97r76                                0/1     Pending           0          5s
router-3-v2xpt                                0/1     Terminating       0          51s
[root@node ~]#
```

图 3-28 Router Pod 状态

通过 API 查看新创建 Pod 的日志：

curl -k   -H "Authorization: Bearer $TOKEN"   -H 'Accept: application/json'

https://$ENDPOINT/api/v1/namespaces/default/pods/router-3-97r76/log

执行结果如下图所示 3-29：

图 3-29 Pod 日志

在本小节中，我们介绍了如何调用 OpenShift API，并演示了一些对 OpenShift 的操作。OpenShift 提供了丰富的 API，感兴趣的读者可以参照本小节的介绍，对照红帽官网（https://docs.openshift.com/container-platform/3.11/rest_api/）的 API 描述进行操作。