天翼云 容器实例

API文档说明

中国电信股份有限公司云计算分公司

目 录

目录

1 概述

文档提供了容器实例（Ctyun Container Instance，简称CCI）API的功能描述、语法、参数定义说明及示例等内容。

2 接口说明

2.1. 接口概述

包括了两类接口：

- 容器实例 CCI 接口，包括：镜像仓库、实例开通接口。
- Kubernetes 原生接口封装，是基于 Kubernetes 原生接口技术规范之上，对用户认证等逻辑调用进行了封装。当前是基于 kubernetes v1.18.0 版本。Kubernetes API 官方文档：https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.18/ 定义了 Kubernetes 原生接口技术规范，包括：Path 参数，Query 参数，请求消息体及返回值等。

表 接口表：

类型	子类型	说明
容器实例CCI接口	镜像仓库	镜像仓库接口，包括查看用户镜像仓库及用户镜像。
	应用监控	无状态应用监控接口及容器组或容器监控接口，包括：查看应用或容器CPU、内存及网络IO近1小时，近3小时，近12小时使用情况。
Kubernetes原生接口封装	Deployment	Deployment对象管理接口，包括Deployment对象的创建，查询，修改，删除等接口。
	Replicaset	Replicaset对象的查询。
	Pod	Pod对象管理接口，包括Pod对象的创建，查询，修改，删除等接口。
	Service	Service对象管理接口，包括Service对象的创建，查询，修改，删除等接口。
	Ingress	Ingress对象管理接口，包括Ingress对象的创建，更新，删除等接口。

	ConfigMap	ConfigMap对象管理接口，包括ConfigMap对象的创建，查询，修改，删除等接口。
	Secret	Secret对象管理接口，包括Sercret对象的创建，查询，修改，删除等接口。
	Event	Event对象管理的查询。

2.2. 认证请求

对于容器实例CCI接口和Kubernetes原生接口封装调用前都需要完成认证请求。

认证请求接口：

GET /cas

示例如下：

```
curl https://${PONTUS}/cas -kv

返回示例：

HTTP/1.1 302 Found
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Content-Type, Hawkular-Tenant, Content-Length, Accept-Encoding, X-CSRF-Token, Authorization, X-Requested-With, If-Modified-Since, X-HTTP-Method-Override
Access-Control-Allow-Methods: GET, POST, OPTIONS, DELETE, PUT, PATCH
Access-Control-Allow-Origin:
Content-Type: text/html; charset=utf-8
Location: /
Set-Cookie: csi=ON6Yib7qNGlgmnaLuLuwKg==; Expires=Wed, 10 Jul 2019 01:44:05 GMT; HttpOnly
Set-Cookie: dcn=ZWNsb3VkY2Fzcy1kZXY=; Expires=Wed, 10 Jul 2019 01:44:05 GMT
Set-Cookie: ns=default; Expires=Wed, 10 Jul 2019 01:44:05 GMT
Set-Cookie: account_id=bac_80605bfc9c534dd3aa377f40d0b93ea2; Expires=Wed, 10 Jul 2019 01:44:05 GMT
Set-Cookie: account_name=caasuser; Expires=Wed, 10 Jul 2019 01:44:05 GMT
Date: Tue, 09 Jul 2019 02:44:05 GMT
Content-Length: 24
```

```
Set-Cookie: e29199d7a7e8c1854c67df374f94b588=4ff78b21c59bfa68b524b35c4f6a9b11; path=/;
HttpOnly; Secure
```

登录认证流程如下：

1、用户访问应用，应用判断该用户是否拥有本地session（根据cookie或者其他状态判定依据）,如果有则返回用户要访问的资源，如果没有则重定向到认证中心走登录流程；

2、认证中心判断是否有全局session存在，如果有则返回应用ticket,如果没有则跳转到统一登录页面进行登录，登录成功后创建全局session，返回应用ticket;

3、应用获取到ticket之后，走认证中心校验ticket接口，校验通过返回应用登录用户用户名，应用创建本地session，失败则返回鉴权失败页面。

2.3.1 Open API认证

2.3.1 概述

OpenApi采用ApiKey认证方式，通过使用Access Key Id / Secret Access Key加密的方法来验证某个请求的发送者身份。Access Key Id（AK）用于标示用户，Secret Access Key（SK）是用户用于加密认证字符串和CCE用来验证认证字符串的密钥，其中SK必须保密，只有用户和CCI知道。

CCI通过认证算法对HTTP请求的指定内容进行计算并输出认证字符串用于认证。开发者需要首先将HTTP请求的指定内容连接成字符串，结合CCI分配的SK，通过HMAC算法计算密文摘要，这个过程也就是对HTTP请求进行签名过程。CCI API使用基于认证字符串的HTTP请求签名机制来验证用户身份。对于每个HTTP请求，都需要携带一个认证字符串然后通过以下方式将这个认证字符串包含在请求中。

在HTTP请求的Authorization头域中包含认证字符串，认证字符串由五个元素组成，分别是版本号，Access Key Id ，时间戳，签名头，签名字符串。格式为如下所示：

Authorization: auth-v1/secretId/timestamp/signedHeaders/signature

1. 版本号固定填写 auth-v1。

2. secretId，由 CCE 提供，用来验证发送请求者的身份。

3. timestamp，发送请求时的时间戳，精确到秒。

4. signedHeaders，用来指定生成签名使用的 http header，如果填写为空字符串，默认使用 Host header。可以指定多个 http header，使用;分隔。

signature是将HTTP请求的指定内容连接成字符串，结合CCE分配的SK，通过HMAC算法计算出来的密文摘要。

2.3.2 生成签名字符串

签名的计算公式为signature = HMAC-SHA256-HEX(Secret Access Key，CanonicalRequest)，从公式可以看出，想要获得签名需要得到CanonicalRequest这个参数，首先介绍如何获取这个参数。

生成CanonicalRequest

CanonicalRequest的计算公式为： CanonicalRequest = HTTP Method + "\n" + CanonicalURI + "\n" + CanonicalQueryString + "\n" + CanonicalHeaders。

从公式可以看出CanonicalRequest由HTTP Method、CanonicalURI、CanonicalQueryString和CanonicalHeaders四部分组成，它们的具体含义及获取方式如下。

1. HTTP Method

指HTTP协议中定义的GET、PUT、POST等请求，必须使用全大写的形式。

2. CanonicalURI

CanonicalURI是对URL中的绝对路径进行编码后的结果，即CanonicalURI = UriEncodeExceptSlash(Path)。要求绝对路径Path必须以“/”开头，不以“/”开头的需要补充上，空路径为“/”。

如果URL为https://cce.container.ctyun.cn/example/测试，则其URL Path为/example/测试，将之

规范化得到CanonicalURI = UriEncodeExceptSlash(/example/测试)=

/example/%E6%B5%8B%E8%AF%95。

3. CanonicalQueryString

CanonicalQueryString是对于URL中的Query String（Query String即URL中“?”后面的“key1 = valve1 & key2 = valve2 ” 字符串）进行编码后的结果。

编码步骤如下：

1. 提取 URL 中的 Query String 项，即 URL 中 “?” 后面的 “key1 = valve1 & key2 = valve2 ” 字符串。

2. 将 Query String 按 Key 进行字典升序排列，value 保持原有顺序不变。

3. 当该项只有 key 时，转换公式为 UriEncode(key) + "="的形式。

4. 当该项是 key=value 的形式时，转换公式为 UriEncode(key) + "=" + UriEncode(value) 的形式。这里 value 可以是空字符串。

5. 将每一项转换后的字符串按照使用& 符号连接起来，生成相应的 CanonicalQueryString。

4. CanonicalHeaders

CanonicalHeaders是对HTTP请求中的Header部分进行选择性编码的结果。您可以自行决定哪些Header需要编码。默认使用Host。

1. 如果指定的签名 Header 没有全部出现在您的 HTTP 请求里面，那么没有出现的部分无需进行编码。如果发送的请求里包含以上 header，出现的 header 必须签名。

2. 如果您使用 Host Header 进行编码，那么认证字符串中的 {signedHeaders} 可以直接留空，无需填写。如果您传入了 signedHeaders，此时会根据 signedHeaders 内容进行签名。

3. 您也可以自己选择想要编码的 Header。如果您选择 Host 之外的 header 进行编码，那么您必须在认证字符串中填写 {signedHeaders} 。填写方法为，把所有在这一阶段进行了编码的 Header 使用分号（;）连接。

4. 选择哪些 Header 进行编码不会影响 API 的功能，但是如果选择太少则可能遭到中间人攻击。

对Header进行编码获取CanonicalHeaders，编码步骤如下。

1. Header 的名字变成全小写，注意仅改名字。

2. 将 Header 的值去掉开头和结尾的空白字符。

3. 经过上一步之后值为空字符串的 Header 忽略，其余的转换为 UriEncode(name) + ":" + UriEncode(value) 的形式。

4. 把上面转换后的所有字符串按照字典序进行排序。

将排序后的字符串按顺序用\n符号连接起来得到最终的CanonicalHeaders。

2.4. 接口状态码

容器引擎CCI接口和Kubernetes原生接口封装接口状态码参见附录状态码表。

3 API指南

3.1 无状态应用

3.1.1 添加无状态应用

功能介绍：

该API用于在指定Namespace下创建新的无状态应用

URI：

POST /namespaces/{namespace}/deployments/{name}

请求示例：

```
POST /namespaces/{namespace}/deployments/{name}
Content-Type: application/json
{
    "kind":"Deployment",
    "apiVersion":"apps/v1",
    "metadata":{
        "name":"test01",
        "namespace":"default",
        "labels":{
            "app":"test01"
        }
    },
    "spec":{
        "replicas":4,
        "selector":{
            "matchLabels":{
               "app":"test01"
            }
        },
        "template":{
            "metadata":{
            "labels":{
                "app":"test01"
            }
            },
```

```
                "spec":{
                   "containers":[
                         {
                              "name":"container-0",
                              "image":"172.18.141.128/library/redis:1.0",
                              "imagePullPolicy":"IfNotPresent"
                         }
                ]
                }
          }
     }
}
```

3.1.2 删除无状态应用

功能介绍：

该API用于删除指定Namespace下的指定名称无状态应用

URI：

DELETE /namespaces/{namespace}/deployments/{name}

请求示例：

```
DELETE /namespaces/{namespace}/deployments/{name}
```

3.1.3 修改无状态应用

功能介绍：

该API用于修改指定Namespace下的指定名称无状态应用，如果成功，返回修改后的无状态应用信息。

URI：

PUT /namespaces/{namespace}/deployments/{name}

请求示例：

```
PUT /namespaces/{namespace}/deployments/{name}
Content-Type: application/json
{
      "kind":"Deployment",
      "apiVersion":"apps/v1",
      "metadata":{
            "name":"test01",
            "namespace":"default",
            "labels":{
                  "app":"test01"
            }
      },
      "spec":{
            "replicas":5,
            "selector":{
                  "matchLabels":{
                     "app":"test01"
                  }
            },
            "template":{
                  "metadata":{
                  "labels":{
                        "app":"test01"
                  }
                  },
                  "spec":{
                  "containers":[
                        {
                            "name":"container-0",
                          "image":"172.18.141.128/library/redis:1.0",
                            "imagePullPolicy":"IfNotPresent"
                        }
                  ]
                  }
            }
       }
}
```

3.1.4 查看无状态应用

功能介绍：

该API用于查询在指定Namespace下指定名称的无状态应用

URI：

GET /namespaces/{namespace}/deployments/{name}

请求示例：

```
GET /namespaces/{namespace}/deployments/{name}
返回body格式：
{
    "kind":"Deployment",
    "apiVersion":"apps/v1",
    "metadata":{
        "name":"test01",
        "namespace":"default",
        "labels":{
            "app":"test01"
        }
    },
    "spec":{
        "replicas":5,
        "selector":{
            "matchLabels":{
               "app":"test01"
            }
        },
        "template":{
            "metadata":{
            "labels":{
                "app":"test01"
            }
            },
            "spec":{
            "containers":[
                {
                    "name":"container-0",
                  "image":"172.18.141.128/library/redis:1.0",
                    "imagePullPolicy":"IfNotPresent"
                }
            ]
            }
        }
    }
}
```

3.1.5 查看无状态应用列表

功能介绍：

该API用于查询在指定Namespace下所有的无状态应用

URI：

GET /namespaces/{namespace}/deployments

请求示例：

```
GET /namespaces/{namespace}/deployments
```

3.1.6 监听单个无状态应用变化

功能介绍：

该API用于监听在指定Namespace下指定无状态应用的变化

URI：

GET /watch/namespaces/{namespace}/deployments/{name}?resourceVersion={version}

Connection: Upgrade

Upgrade: websocket

请求示例：

```
GET /namespaces/{namespace}/deployments/{name}?resourceVersion=1231231
```

3.1.7 监听命名空间下无状态应用变化

功能介绍：

该API用于监听在指定Namespace下所有无状态应用的变化

URI：

GET /watch/namespaces/{namespace}/deployments?resourceVersion={version}

Connection: Upgrade

Upgrade: websocket

请求示例：

```
GET /namespaces/{namespace}/deployments?resourceVersion=1231231
```

3.1.8 监听所有无状态应用变化

功能介绍：

该API用于监听集群所有无状态应用的变化

URI：

GET /watch/deployments?resourceVersion={version}

Connection: Upgrade

Upgrade: websocket

请求示例：

```
GET /watch/deployments?resourceVersion=1231231
```

3.2 应用监控

3.2.1 应用CPU资源利用率

功能介绍：

该API用于查询应用的CPU资源利用率

URI：

GET /prometheus/api/v1/query_range

请求示例：

```
GET
/prometheus/api/v1/query_range?query=sum(rate(container_cpu_usage_seconds_total{namespace=%22
```

```
default%22,container_name!=%22POD%22,container_name!=%22%22,pod_name=%22test01-54d4574bc-ks5cg%22}[2m]))&start=1562566356.551&end=1562569956.551&step=60
```

3.2.2 应用内存资源利用率

功能介绍：

该API用于查询应用的内存资源利用率

URI：

GET /prometheus/api/v1/query_range

请求示例：

```
GET /prometheus/api/v1/query_range?query=sum(container_memory_usage_bytes{namespace=%22default%22,container_name!=%22POD%22,container_name!=%22%22,pod_name=%22test01-54d4574bc-ks5cg%22})&start=1562566356.555&end=1562569956.555&step=60
```

3.2.3 应用上行网络速率

功能介绍：

该API用于查询应用的上行网络速率

URI：

GET /prometheus/api/v1/query_range

请求示例：

```
GET /prometheus/api/v1/query_range?query=sum(rate(container_network_transmit_bytes_total{namespace=%22default%22,pod_name=%22test01-54d4574bc-ks5cg%22}[2m]))&start=1562566356.555&end=1562569956.555&step=60
```

3.2.4 应用下行网络速率

功能介绍：

该API用于查询应用的下行网络速率

URI：

GET /prometheus/api/v1/query_range

请求示例：

```
GET
/prometheus/api/v1/query_range?query=sum(rate(container_network_receive_bytes_total{namespace=
%22default%22,pod_name=%22test01-54d4574bc-
ks5cg%22}[2m]))&start=1562566356.555&end=1562569956.555&step=60
```

3.3 容器组

3.3.1 添加容器组

功能介绍：

该API用于在指定Namespace下创建新的容器组

URI：

POST /namespaces/{namespace}/pods/{name}

请求示例：

```
POST /namespaces/{namespace}/pods/{name}
Content-Type: application/json
{
    "kind":"Pod",
    "apiVersion":"core/v1",
    "metadata":{
        "name":"test",
        "namespace":"default"
    },
    "spec":{
        "containers":[
            {
                "name":"container-0",
```

```
                    "image":"172.18.141.128:5000/library/nginx:latest",
                    "imagePullPolicy":"Always"
                }
            ]
        }
}
```

3.3.2 删除容器组

功能介绍：

该API用于删除指定Namespace下的指定名称容器组

URI：

DELETE /namespaces/{namespace}/pods/{name}

请求示例：

```
DELETE /namespaces/{namespace}/pods/{name}
```

3.3.3 修改容器组

功能介绍：

该API用于修改指定Namespace下的指定名称容器组，如果成功，返回修改后的容器组信息

URI：

PUT /namespaces/{namespace}/pods/{name}

请求示例：

```
PUT /namespaces/{namespace}/pods/{name}
Content-Type: application/json
{
    "kind":"Pod",
    "apiVersion":"core/v1",
    "metadata":{
```

```
        "name":"test",
        "namespace":"default"
    },
    "spec":{
        "containers":[
            {
                "name":"container-0",
                "image":"172.18.141.128:5000/library/nginx:1",
                "imagePullPolicy":"Always"
            }
        ]
    }
}
```

3.3.4 查询容器组

功能介绍：

该API用于查询在指定Namespace下指定名称的容器组

URI：

GET /namespaces/{namespace}/pods/{name}

请求示例：

```
GET /namespaces/{namespace}/pods/{name}
返回示例：
{
    "kind":"Pod",
    "apiVersion":"core/v1",
    "metadata":{
        "name":"test",
        "namespace":"default"
    },
    "spec":{
        "containers":[
            {
                "name":"container-0",
                "image":"172.18.141.128:5000/library/nginx:latest",
                "imagePullPolicy":"Always"
            }
        ]
```

```
    }
}
```

3.3.5 查询容器组列表

功能介绍：

该API用于查询在指定Namespace下所有的容器组

URI：

GET /namespaces/{namespace}/pods

请求示例：

```
GET /namespaces/{namespace}/pods
```

3.3.6 监听单个容器组变化

功能介绍：

该API用于监听在指定Namespace下指定容器组的变化

URI：

GET /watch/namespaces/{namespace}/pods/{name}?resourceVersion={version}

Connection: Upgrade

Upgrade: websocket

请求示例：

```
GET /namespaces/{namespace}/pods/{name}?resourceVersion=1231231
```

3.3.7 监听同命名空间下容器组变化

功能介绍：

该API用于监听在指定Namespace下所有容器组的变化

URI：

GET /watch/namespaces/{namespace}/pods?resourceVersion={version}
Connection: Upgrade
Upgrade: websocket

请求示例：

```
GET /namespaces/{namespace}/pods?resourceVersion=1231231
```

3.3.8 监听所有容器组变化

功能介绍：

该API用于监听集群所有容器组的变化

URI：

GET /watch/pods?resourceVersion={version}
Connection: Upgrade
Upgrade: websocket

请求示例：

```
GET /watch/pods?resourceVersion=1231231
```

3.4 事件

3.4.1 获取指定命名空间下所有的事件

功能介绍：

该API用于获取指定命名空间下所有的事件。

URI：

GET /api/v1/namespaces/{namespace}/events

请求示例：

```
GET /api/v1/namespaces/default/events HTTP/1.1
```

返回示例：


```
{
        "kind": "EventList",
        "apiVersion": "v1",
        "metadata": {
        "selfLink": "/api/v1/namespaces/default/events",
        "resourceVersion": "6496348"
        },
        "items": [
        {
                "metadata": {
                "name": "fanbin-9fd4c8495-2m2cf.15ae63f8e3e0ed42",
                "namespace": "default",
                "selfLink": "/api/v1/namespaces/default/events/fanbin-9fd4c8495-
2m2cf.15ae63f8e3e0ed42",
                "uid": "01ae31ca-9ed2-11e9-b4da-0050568ccbe7",
                "resourceVersion": "6495675",
                "creationTimestamp": "2019-07-05T03:07:22Z"
                },
                "involvedObject": {
                "kind": "Pod",
                "namespace": "default",
                "name": "fanbin-9fd4c8495-2m2cf",
                "uid": "fd7f64fd-9ed1-11e9-b4da-0050568ccbe7",
                "apiVersion": "v1",
                "resourceVersion": "6100341",
                "fieldPath": "spec.containers{container-0}"
              },
                "reason": "Failed",
                "message": "Error: ImagePullBackOff",
                "source": {
                "component": "kubelet",
                "host": "node1"
                },
                "firstTimestamp": "2019-07-05T03:07:21Z",
                "lastTimestamp": "2019-07-07T02:22:19Z",
                "count": 11589,
                "type": "Warning",
                "eventTime": null,
                "reportingComponent": "",
                "reportingInstance": ""
        }
        ]
}
```


3.4.2 获取集群下所有的事件

功能介绍：

该API用于获取集群下所有的事件。

URI：

GET /api/v1/events

请求示例：

```
GET /api/v1/events HTTP/1.1
```

返回示例：


```
{
        "kind": "EventList",
        "apiVersion": "v1",
        "metadata": {
"selfLink": "/api/v1/events",
"resourceVersion": "6582324"
        },
        "items": [
        {
                "metadata": {
                "name": "fanbin-9fd4c8495-2m2cf.15ae63f8e3e0ed42",
                "namespace": "default",
                "selfLink": "/api/v1/namespaces/default/events/fanbin-9fd4c8495-
2m2cf.15ae63f8e3e0ed42",
                "uid": "01ae31ca-9ed2-11e9-b4da-0050568ccbe7",
                "resourceVersion": "6495675",
                "creationTimestamp": "2019-07-05T03:07:22Z"
                },
                "involvedObject": {
                "kind": "Pod",
                "namespace": "default",
                "name": "fanbin-9fd4c8495-2m2cf",
                "uid": "fd7f64fd-9ed1-11e9-b4da-0050568ccbe7",
                "apiVersion": "v1",
                "resourceVersion": "6100341",
                "fieldPath": "spec.containers{container-0}"
                },
                "reason": "Failed",
                "message": "Error: ImagePullBackOff",
                "source": {
                "component": "kubelet",
                "host": "node1"
                },
                "firstTimestamp": "2019-07-05T03:07:21Z",
                "lastTimestamp": "2019-07-07T02:22:19Z",
                "count": 11589,
                "type": "Warning",
                "eventTime": null,
```

```
                "reportingComponent": "",
                "reportingInstance": ""
        }
        ]
}
```

3.5 弹性伸缩

3.5.1 添加弹性伸缩

功能介绍：

该API用于在指定Namespace下创建新的弹性伸缩

URI：

POST /namespaces/{namespace}/horizontalpodautoscalers/{name}

请求示例：

```
POST /namespaces/{namespace}/horizontalpodautoscalers/{name}
Content-Type: application/json
{
    "apiVersion":"autoscaling/v2beta2",
    "kind":"HorizontalPodAutoscaler",
    "metadata":{
        "name":"test111",
        "namespace":"default"
    },
    "spec":{
        "scaleTargetRef":{
            "apiVersion":"apps/v1",
          "kind":"Deployment",
            "name":"fanbin"
        },
        "minReplicas":1,
        "maxReplicas":10,
        "metrics":[
            {
              "type":"Resource",
            "resource":{
                "name":"cpu",
                 "target":{
                    "type":"Utilization",
                    "averageUtilization":80
```

```
                        }
                }
                }
        ]
    }
}
```

3.5.2 删除弹性伸缩

功能介绍：

该API用于删除指定Namespace下的指定名称弹性伸缩

URI：

DELETE /namespaces/{namespace}/horizontalpodautoscalers/{name}

请求示例：

```
DELETE /namespaces/{namespace}/horizontalpodautoscalers/{name}
```

3.5.3 修改弹性伸缩

功能介绍：

该API用于修改指定Namespace下的指定名称弹性伸缩，如果成功，返回修改后的弹性伸缩信息

URI：

PUT /namespaces/{namespace}/horizontalpodautoscalers/{name}

请求示例：

```
PUT /namespaces/{namespace}/horizontalpodautoscalers/{name}
Content-Type: application/json
{
    "apiVersion":"autoscaling/v2beta2",
    "kind":"HorizontalPodAutoscaler",
```

```
    "metadata":{
        "name":"test111",
        "namespace":"default"
    },
    "spec":{
        "scaleTargetRef":{
            "apiVersion":"apps/v1",
            "kind":"Deployment",
              "name":"fanbin"
        },
        "minReplicas":5,
        "maxReplicas":10,
        "metrics":[
                {
                   "type":"Resource",
                "resource":{
                        "name":"cpu",
                         "target":{
                             "type":"Utilization",
                             "averageUtilization":80
                         }
                }
                }
        ]
    }
}
```

3.5.4 查询弹性伸缩

功能介绍：

该API用于查询在指定Namespace下指定名称的弹性伸缩

URI：

GET /namespaces/{namespace}/horizontalpodautoscalers/{name}

请求示例：

```
GET /namespaces/{namespace}/horizontalpodautoscalers/{name}
返回示例：
{
    "apiVersion":"autoscaling/v2beta2",
    "kind":"HorizontalPodAutoscaler",
```

```
    "metadata":{
          "name":"test111",
          "namespace":"default"
    },
        "spec":{
          "scaleTargetRef":{
                "apiVersion":"apps/v1",
                "kind":"Deployment",
                  "name":"fanbin"
          },
          "minReplicas":1,
          "maxReplicas":10,
          "metrics":[
                  {
                     "type":"Resource",
                  "resource":{
                            "name":"cpu",
                             "target":{
                                  "type":"Utilization",
                                  "averageUtilization":80
                             }
                          }
                  }
          ]
    }
}
```

3.5.5 查询弹性伸缩列表

功能介绍：

该API用于查询在指定Namespace下所有的弹性伸缩

URI：

GET /namespaces/{namespace}/horizontalpodautoscalers

请求示例：

```
GET /namespaces/{namespace}/horizontalpodautoscalers
```

3.6 服务

服务（Service）是一种抽象，它定义了一组容器组（Pods）的逻辑集合和一个用于访问它们的策略 - 有的时候被称之为微服务。一个Service的目标Pod集合通常是由Label Selector 来决定的，举个例子，想象一个处理图片的后端运行了三个副本。这些副本都是可以替代的 - 前端不关心它们使用的是哪一个后端。尽管实际组成后端集合的Pod可能会变化，前端的客户端却不需要知道这个变化，也不需要自己有一个列表来记录这些后端服务。Service抽象能让你达到这种解耦。

3.6.1 创建服务

功能介绍：

该API用于创建指定Namespace下的服务资源类型

URI：

POST /namespaces/{namespace}/services

请求示例：

```
POST /namespaces/default/services
{
      "apiVersion":"v1",
      "kind":"Service",
      "metadata":{
            "name":"example",
            "labels":{
                  "app":"test01"
          }
          },
      "spec":{
            "selector":{
                  "app":"test01"
          },
            "ports":[
                  {
                        "targetPort":6666,
                        "port":6666,
                        "protocol":"TCP",
                        "name":"port-0"
                  }
          ],
            "type":"ClusterIP"
          }
}
```

3.6.2 更新服务

功能介绍：

该API用于更新指定Namespace下的服务资源类型

URI：

PUT /namespaces/{namespace}/services/{name}

请求示例：

```
PUT /namespaces/default/services/example
{
      "kind":"Service",
      "apiVersion":"v1",
      "metadata":{
            "name":"example",
            "namespace":"default",
            "selfLink":"/api/v1/namespaces/default/services/example",
            "uid":"21222523-9e29-11e9-b4da-0050568ccbe7",
            "resourceVersion":"5917936",
            "creationTimestamp":"2019-07-04T06:58:29Z",
            "labels":{
                  "app":"test01"
          }
          },
      "spec":{
            "ports":[
                  {
                        "targetPort":7777,
                        "port":7777,
                        "protocol":"TCP",
                        "name":"port-0"
                    }
          ],
            "selector":{
                  "app":"test01"
          },
            "clusterIP":"10.98.41.232",
            "type":"ClusterIP",
            "sessionAffinity":"None"
          },
      "status":{
            "loadBalancer":{
          }
          }
}
```

3.6.3 部分更新服务

功能介绍：

该API用于部分更新指定Namespace下的服务资源类型

URI：

PATCH /namespaces/{namespace}/services/{name}

请求示例：

```
PATCH /namespaces/default/services/example
Content-Type: application/strategic-merge-patch+json
{
      "spec":{
            "sessionAffinity":"ClientIP"
          }
}
```

3.6.4 删除服务

功能介绍：

该API用于删除指定Namespace下的服务资源类型

URI：

DELETE /namespaces/{namespace}/services/{name}

请求示例：

```
DELETE /namespaces/default/services/example
```

3.6.5 查询服务

功能介绍：

该API用于查询指定Namespace下的服务资源类型

URI：

GET /namespaces/{namespace}/services/{name}

请求示例：

```
GET/namespaces/default/services/example
```

3.6.6 查询指定Namespace下的所有服务

功能介绍：

该API用于查询指定Namespace下的所有服务资源类型

URI：

GET /namespaces/{namespace}/services

请求示例：

```
GET /namespaces/default/services
```

3.6.7 查询集群下的所有服务

功能介绍：

该API用于查询集群下的所有服务资源类型

URI：

GET /services

请求示例：

```
GET /services
```

3.6.8 监听指定的服务

功能介绍：

该API用于Watch指定Namespace下的服务资源类型

URI：

GET /watch/namespaces/{namespace}/services/{name}

请求示例：

```
GET /namespaces/default/services/example
```

3.6.9 监听指定Namespace下的所有服务

功能介绍：

该API用于Watch指定Namespace下的所有服务资源类型

URI：

GET /watch/namespaces/{namespace}/services

请求示例：

```
GET /namespaces/default/services
```

3.6.10 监听集群下的所有服务

功能介绍：

该API用于Watch集群下的所有服务资源类型

URI：

GET /watch/services

请求示例：

```
GET /services
```

3.7 路由

3.7.1 创建路由

功能介绍：

该API用于创建指定Namespace下的路由资源类型

URI：

POST /namespaces/{namespace}/ingresses

请求示例：

```
POST /namespaces/default/ingresses
{
      "apiVersion":"extensions/v1beta1",
      "kind":"Ingress",
      "metadata":{
            "name":"example",
            "namespace":"default",
            "annotations":{
                  "ingress.kubernetes.io/ssl-redirect":"false"
          }
          },
      "spec":{
            "rules":[
                  {
                        "host":"example.ctyun.com",
                        "http":{
                              "paths":[
                              {
                                          "backend":{
                                                "serviceName":"httpserver",
                                                "servicePort":8080
                                          },
                                          "path":"/"
                              }
                              ]
                        }
                }
          ]
          }
}
```

3.7.2 更新路由

功能介绍：

该API用于更新指定Namespace下的路由资源类型

其中以下字段支持更新：

metadata.name

metadata.namespace

metadata.selfLink

metadata.resourceVersion

metadata.generation

metadata.uid

metadata.creationTimestamp

metadata.deletionTimestamp

metadata.labels

metadata.annotations

spec.rules

spec.loadBalancerIP

URI：

PATCH /namespaces/{namespace}/ingresses/{name}

请求示例：

```
PATCH /namespaces/default/ingresses/example
Content-Type:application/strategic-merge-patch+json
{
      "spec":{
            "rules":[
                  {
                        "host":"example.ctyun.com",
                        "http":{
                              "paths":[
                              {
                                    "backend":{
                                          "serviceName":"httpserver",
                                     "servicePort":8080
                                    },
                                     "path":"/abc"
```

```
                        }
                        ]
                    }
                }
            ]
            }
}
```

3.7.3 替换指定的路由

功能介绍：

该API用于替换指定Namespace下的路由资源类型

URI：

PUT /namespaces/{namespace}/ingresses/{name}

请求示例：

```
PUT /namespaces/default/ingresses/example
{
    "apiVersion":"extensions/v1beta1",
    "kind":"Ingress",
    "metadata":{
        "name":"example",
        "namespace":"default",
        "annotations":{
            "ingress.kubernetes.io/ssl-redirect":"false"
        }
        },
    "spec":{
        "rules":[
            {
                "host":"example.ctyun.com",
                "http":{
                    "paths":[
                    {
                        "backend":{
                            "serviceName":"httpserver",
                            "servicePort":8080
                        },
                        "path":"/aaa"
                    }
                    ]
                }
            }
```

```
        ]
        }
}
```

3.7.4 删除路由

功能介绍：

该API用于删除指定Namespace下的路由资源类型

URI：

DELETE /namespaces/{namespace}/ingresses/{name}

请求示例：

```
DELETE /namespaces/default/ingresses/example
```

3.7.5 删除指定Namespace下所有的路由

功能介绍：

该API用于删除指定Namespace下所有的路由资源类型

URI：

DELETE /namespaces/{namespace}/ingresses

请求示例：

```
DELETE /namespaces/default/ingresses
```

3.7.6 删除集群中所有的路由

功能介绍：

该API用于删除集群中所有的路由资源类型

URI：

DELETE /ingresses

请求示例：

```
DELETE /ingresses
```

3.7.7 获取指定的路由

功能介绍：

该API用于获取指定的路由资源类型

URI：

GET /namespaces/{namespaces}/ingresses/{name}

请求示例：

```
GET /namespaces/default/ingresses/example
```

3.7.8 获取指定Namespac下所有的路由

功能介绍：

该API用于获取指定Namespace下所有的路由资源类型

URI：

GET /namespaces/{namespaces}/ingresses/{name}

请求示例：

```
GET /namespaces/default/ingresses
```

3.7.9 获取集群中所有的路由

功能介绍：

该API用于获取集群中所有的路由资源类型

URI：

GET /ingresses

请求示例：

```
GET /ingresses
```

3.8 配置项

3.8.1 创建配置项

功能介绍：

该API用于创建配置项

URI：

POST /api/v1/namespaces/{namespace}/configmaps

请求示例：

```
POST /api/v1/namespaces/namespace0/configmaps HTTP/1.1
Accept: application/json
Content-Type: application/json
{
        "apiVersion": "v1",
        "kind": "ConfigMap",
        "data": {
        "config.json": "{\"key\": \"value\"}"
        },
        "metadata": {
        "name": "configmap1"
        }
}
```

返回示例：

```
{
        "kind": "ConfigMap",
        "apiVersion": "v1",
        "metadata": {
        "name": "configmap1",
        "namespace": "namespace0",
        "selfLink": "/api/v1/namespaces/namespace0/configmaps/configmap1",
        "uid": "96b69913-9efd-11e9-b4da-0050568ccbe7",
        "resourceVersion": "6145658",
        "creationTimestamp": "2019-07-05T08:19:20Z"
        },
        "data": {
        "config.json": "{\"key\": \"value\"}"
        }
}
```

3.8.2 部份更新配置项

功能介绍：

该API用于部分更新配置项

URI：

PATCH /api/v1/namespaces/{namespace}/configmaps/{name}

请求示例：

```
PATCH /api/v1/namespaces/namespace0/configmaps/configmap1 HTTP/1.1
Accept: application/json
Content-Type: application/json
{
        "data": {
        "config.json": "{\"key1\": \"value1\"}"
        }
}
```

返回示例：

```
{
    "kind": "ConfigMap",
    "apiVersion": "v1",
    "metadata": {
        "name": "configmap1",
        "namespace": "namespace0",
        "selfLink": "/api/v1/namespaces/namespace0/configmaps/configmap1",
        "uid": "96b69913-9efd-11e9-b4da-0050568ccbe7",
        "resourceVersion": "6146786",
        "creationTimestamp": "2019-07-05T08:19:20Z"
```

```
        },
    "data": {
          "config.json": "{\"key1\": \"value1\"}"
        }
}
```

3.8.3 替换配置项

功能介绍：

该API用于替换配置项

URI：

PUT /api/v1/namespaces/{namespace}/configmaps/{name}

请求示例：

```
PUT /api/v1/namespaces/namespace0/configmaps/configmap1 HTTP/1.1
Accept: application/json
Content-Type: application/json
{
    "apiVersion": "v1",
    "kind": "ConfigMap",
    "data": {
          "config.json": "{\"key\": \"value\"}"
        },
    "metadata": {
          "name": "configmap1"
        }
}
```

返回示例：

```
{
    "kind": "ConfigMap",
    "apiVersion": "v1",
    "metadata": {
          "name": "configmap1",
          "namespace": "namespace0",
          "selfLink": "/api/v1/namespaces/namespace0/configmaps/configmap1",
          "uid": "96b69913-9efd-11e9-b4da-0050568ccbe7",
          "resourceVersion": "6147508",
          "creationTimestamp": "2019-07-05T08:19:20Z"
        },
    "data": {
          "config.json": "{\"key\": \"value\"}"
        }
}
```

3.8.4 删除配置项

功能介绍：

该API用于删除配置项

URI：

DELETE /api/v1/namespaces/{namespace}/configmaps/{name}

请求示例：

```
DELETE /api/v1/namespaces/namespace0/configmaps/configmap1 HTTP/1.1
返回示例：
{
      "kind": "Status",
      "apiVersion": "v1",
      "metadata": {},
      "status": "Success",
      "details": {
            "name": "configmap1",
            "kind": "configmaps",
            "uid": "96b69913-9efd-11e9-b4da-0050568ccbe7"
          }
}
```

3.8.5 删除指定命名空间下所有的配置项

功能介绍：

该API用于删除指定命名空间下所有的配置项

URI：

DELETE /api/v1/namespaces/{namespace}/configmaps

请求示例：

```
DELETE /api/v1/namespaces/namespace0/configmaps HTTP/1.1
返回示例：
```

```
{
    "kind": "ConfigMapList",
    "apiVersion": "v1",
    "metadata": {
        "selfLink": "/api/v1/namespaces/namespace0/configmaps",
        "resourceVersion": "6148830"
      },
    "items": [
      {
          "metadata": {
              "name": "configmap1",
              "namespace": "namespace0",
              "selfLink": "/api/v1/namespaces/namespace0/configmaps/configmap1",
              "uid": "b4457a3f-9f00-11e9-b4da-0050568ccbe7",
              "resourceVersion": "6148767",
              "creationTimestamp": "2019-07-05T08:41:38Z"
            },
          "data": {
              "config.json": "{\"key\": \"value\"}"
          }
      }
      ]
}
```


3.8.6 获取指定的配置项信息

功能介绍：

该API用于获取指定的配置项信息

URI：

GET /api/v1/namespaces/{namespace}/configmaps/{name}

请求示例：


```
GET /api/v1/namespaces/namespace0/configmaps/namespace0 HTTP/1.1
返回示例：
{
    "kind": "ConfigMap",
    "apiVersion": "v1",
    "metadata": {
        "name": "configmap1",
        "namespace": "namespace0",
        "selfLink": "/api/v1/namespaces/namespace0/configmaps/configmap1",
        "uid": "4dec18c5-9f01-11e9-b4da-0050568ccbe7",
```

```
            "resourceVersion": "6149332",
            "creationTimestamp": "2019-07-05T08:45:56Z"
        },
    "data": {
            "config.json": "{\"key\": \"value\"}"
        }
}
```

3.8.7 获取指定命名空间下所有的配置项信息

功能介绍：

该API用于获取指定命名空间下所有的配置项信息。

URI：

GET /api/v1/namespaces/{namespace}/configmaps

请求示例：

```
GET /api/v1/namespaces/namespace0/configmaps HTTP/1.1
返回示例：
{
        "kind": "ConfigMapList",
        "apiVersion": "v1",
        "metadata": {
        "selfLink": "/api/v1/namespaces/namespace0/configmaps",
          "resourceVersion": "6149715"
        },
        "items": [
        {
                "metadata": {
                "name": "configmap1",
                    "namespace": "namespace0",
                    "selfLink": "/api/v1/namespaces/namespace0/configmaps/configmap1",
                "uid": "4dec18c5-9f01-11e9-b4da-0050568ccbe7",
                    "resourceVersion": "6149332",
                    "creationTimestamp": "2019-07-05T08:45:56Z"
                  },
                "data": {
                    "config.json": "{\"key\": \"value\"}"
                }
        }
        ]
}
```

3.8.8 获取集群下所有的配置项信息

功能介绍：

该API用于获取集群下所有的配置项信息。

URI：

GET /api/v1/configmaps

请求示例：

```
GET /api/v1/configmaps HTTP/1.1
```

返回示例：


```
{
    "kind": "ConfigMapList",
    "apiVersion": "v1",
    "metadata": {
        "selfLink": "/api/v1/configmaps",
        "resourceVersion": "6150511"
      },
    "items": [
      {
          "metadata": {
              "name": "configmap1",
              "namespace": "namespace0",
              "selfLink": "/api/v1/namespaces/namespace0/configmaps/configmap1",
              "uid": "4dec18c5-9f01-11e9-b4da-0050568ccbe7",
              "resourceVersion": "6149332",
              "creationTimestamp": "2019-07-05T08:45:56Z"
          },
          "data": {
              "config.json": "{\"key\": \"value\"}"
          }
      }
      ]
}
```


3.8.9 监听指定的配置项

功能介绍：

该API用于监控指定的配置项

URI：

GET /api/v1/watch/namespaces/{namespace}/configmaps/{name}

请求示例：

```
GET   /api/v1/watch/namespaces/namespace0/configmaps/configmap1 HTTP/1.1
返回示例：
{
   "type": "DELETED",
   "object": {
                    "kind": "ConfigMap",
                    "apiVersion": "v1",
                    "metadata": {
                       "name": "configmap1",
                       "namespace": "namespace0",
                       "selfLink": "/api/v1/namespaces/namespace0/configmaps/configmap1",
                       "uid": "4dec18c5-9f01-11e9-b4da-0050568ccbe7",
                       "resourceVersion": "6151981",
                       "creationTimestamp": "2019-07-05T08:45:56Z"
                    },
                    "data": {
                       "config.json": "{\"key\": \"value\"}"
                    }
   }
}
```

3.8.10 监听指定命名空间下所有的配置项

功能介绍：

该API用于监控指定命名空间下的所有配置项

URI：

GET /api/v1/watch/namespaces/{namespace}/configmaps

请求示例：

```
GET   /api/v1/watch/namespaces/namespace0/configmaps HTTP/1.1
返回示例：
{
   "type": "DELETED",
   "object": {
                    "kind": "ConfigMap",
                    "apiVersion": "v1",
                    "metadata": {
```

```
                    "name": "configmap1",
                    "namespace": "namespace0",
                    "selfLink": "/api/v1/namespaces/namespace0/configmaps/configmap1",
                    "uid": "c4801b13-9f05-11e9-b4da-0050568ccbe7",
                    "resourceVersion": "6153842",
                    "creationTimestamp": "2019-07-05T09:17:53Z"
                },
                "data": {
                    "config.json": "{\"key\": \"value\"}"
                }
    }
}
```

3.8.11 监听集群下所有的配置项

功能介绍：

该API用于监控集群下所有的配置项。

URI：

GET /api/v1/watch/configmaps

请求示例：

```
GET /api/v1/watch/configmaps HTTP/1.1
```

返回示例：

```
{
  "type": "DELETED",
  "object": {
                "kind": "ConfigMap",
                "apiVersion": "v1",
                "metadata": {
                    "name": "configmap1",
                    "namespace": "namespace0",
                    "selfLink": "/api/v1/namespaces/namespace0/configmaps/configmap1",
                    "uid": "b43be7a4-9f06-11e9-b4da-0050568ccbe7",
                    "resourceVersion": "6154739",
                    "creationTimestamp": "2019-07-05T09:24:35Z"
                },
                "data": {
                    "config.json": "{\"key\": \"value\"}"
                }
  }
}
```

3.9 私密凭据

3.9.1 创建私密凭据

功能介绍：

该API用于创建私密凭据。

Secret主要有五种类型：

- kubernetes.io/service-account-token
- Opaque
- kubernetes.io/dockerconfigjson
- kubernetes.io/dockercfg
- kubernetes/tls

URI：

POST /api/v1/namespaces/{namespace}/secrets

请求示例：

```
POST /api/v1/namespaces/namespace0/secrets HTTP/1.1
Accept: application/json
Content-Type: application/json
{
   "apiVersion": "v1",
   "kind": "Secret",
   "metadata": {
        "name": "secret0"
   },
   "type": "Opaque",
   "data": {
         "username": "YWRtaW4=",
             "password": "MWYyZDFlMmU2N2Rm"
   }
}
```

返回示例：

```
{
"kind": "Secret",
```

```
"apiVersion": "v1",
"metadata": {
"name": "secret0",
"namespace": "namespace0",
"selfLink": "/api/v1/namespaces/namespace0/secrets/secret0",
"uid": "8f3d7049-a06d-11e9-b4da-0050568ccbe7",
"resourceVersion": "6511097",
"creationTimestamp": "2019-07-07T04:13:22Z"
},
"data": {
"password": "MWYyZDFlMmU2N2Rm",
"username": "YWRtaW4="
},
"type": "Opaque"
}
```


3.9.2 部分更新私密凭据

功能介绍：

该API用于部分更新私密凭据。

URI：

PATCH /api/v1/namespaces/{namespace}/secrets/{name}

请求示例：

```
PATCH /api/v1/namespaces/namespace0/secrets/secret0 HTTP/1.1
Accept: application/json
Content-Type: application/json
{
   "data": {
          "username": "dXNlcjA=",
          "password": "MTIzNDU2Nzg="
    }
}
```

返回示例：


```
 {
"kind": "Secret",
"apiVersion": "v1",
"metadata": {
"name": "secret0",
"namespace": "namespace0",
"selfLink": "/api/v1/namespaces/namespace0/secrets/secret0",
"uid": "68276161-a06a-11e9-b4da-0050568ccbe7",
"resourceVersion": "6508262",
"creationTimestamp": "2019-07-07T03:50:48Z"
```

```
},
"data": {
"password": "MTIzNDU2Nzg=",
"username": "dXNlcjA="
},
"type": "Opaque"
}
```


3.9.3 替换私密凭据

功能介绍：

该API用于替换私密凭据。

URI：

PUT /api/v1/namespaces/{namespace}/secrets/{name}

请求示例：

```
PUT /api/v1/namespaces/namespace0/secrets/secret0 HTTP/1.1
Accept: application/json
Content-Type: application/json
{
   "apiVersion": "v1",
   "kind": "Secret",
   "metadata": {
         "name": "secret0"
   },
   "type": "Opaque",
   "data": {
              "username": "YWRtaW4=",
              "password": "MWYyZDFlMmU2N2Rm"
   }
}
```

返回示例：


```
{
"kind": "Secret",
"apiVersion": "v1",
"metadata": {
"name": "secret0",
"namespace": "namespace0",
"selfLink": "/api/v1/namespaces/namespace0/secrets/secret0",
"uid": "8f3d7049-a06d-11e9-b4da-0050568ccbe7",
"resourceVersion": "6513309",
"creationTimestamp": "2019-07-07T04:13:22Z"
},
"data": {
```

```
"password": "MWYyZDFlMmU2N2Rm",
"username": "YWRtaW4="
},
"type": "Opaque"
}
```

3.9.4 删除私密凭据

功能介绍：

该API用于删除私密凭据。

URI：

DELETE /api/v1/namespaces/{namespace}/secrets/{name}

请求示例：

```
DELETE /api/v1/namespaces/namespace0/secrets/secret0 HTTP/1.1
Accept: application/json
```

返回示例：

```
{
"kind": "Status",
"apiVersion": "v1",
"metadata": {},
"status": "Success",
"details": {
"name": "secret0",
"kind": "secrets",
"uid": "8f3d7049-a06d-11e9-b4da-0050568ccbe7"
}
}
```

3.9.5 删除指定Namespace下的所有私密凭据

功能介绍：

该API用于删除指定Namespace下所有的私密凭据。

URI：

DELETE /api/v1/namespaces/{namespace}/secrets

请求示例：

```
DELETE /api/v1/namespaces/namespace0/secrets HTTP/1.1
Accept: application/json
```

返回示例：


```
{
        "kind": "SecretList",
        "apiVersion": "v1",
        "metadata": {
        "selfLink": "/api/v1/namespaces/namespace0/secrets",
        "resourceVersion": "6514879"
        },
        "items": [
        {
                "metadata": {
                "name": "secret0",
                "namespace": "namespace0",
                "selfLink": "/api/v1/namespaces/namespace0/secrets/secret0",
                "uid": "491bf535-a071-11e9-b4da-0050568ccbe7",
                "resourceVersion": "6514818",
                "creationTimestamp": "2019-07-07T04:40:02Z"
           },
                "data": {
                "password": "MWYyZDFlMmU2N2Rm",
                "username": "YWRtaW4="
                },
                "type": "Opaque"
        }
        ]
}
```


3.9.6 获取指定的私密凭据信息

功能介绍：

该API用于获取指定的私密凭据信息 。

URI：

GET /api/v1/namespaces/{namespace}/secrets/{name}

请求示例：

```
GET /api/v1/namespaces/namespace0/secrets/secret0 HTTP/1.1
Accept: application/json
```

返回示例：


```
{
```

```
"kind": "Secret",
"apiVersion": "v1",
"metadata": {
"name": "secret0",
"namespace": "namespace0",
"selfLink": "/api/v1/namespaces/namespace0/secrets/secret0",
"uid": "66343023-a072-11e9-b4da-0050568ccbe7",
"resourceVersion": "6515914",
"creationTimestamp": "2019-07-07T04:48:01Z"
},
"data": {
"password": "MWYyZDFlMmU2N2Rm",
"username": "YWRtaW4="
},
"type": "Opaque"
}
```


3.9.7 获取指定Namespace下的所有私密凭据

功能介绍：

该API用于获取指定Namespace下所有的私密凭据信息。

URI：

GET /api/v1/namespaces/{namespace}/secrets

请求示例：

```
GET /api/v1/namespaces/namespace0/secrets HTTP/1.1
Accept: application/json
```

返回示例：


```
{
        "kind": "SecretList",
        "apiVersion": "v1",
        "metadata": {
        "selfLink": "/api/v1/namespaces/namespace0/secrets",
        "resourceVersion": "6516376"
        },
        "items": [
        {
                "metadata": {
                "name": "secret0",
                "namespace": "namespace0",
                "selfLink": "/api/v1/namespaces/namespace0/secrets/secret0",
                "uid": "66343023-a072-11e9-b4da-0050568ccbe7",
                "resourceVersion": "6515914",
```

```
                "creationTimestamp": "2019-07-07T04:48:01Z"
                },
                "data": {
                "password": "MWYyZDFlMmU2N2Rm",
                "username": "YWRtaW4="
                },
                "type": "Opaque"
        }
        ]
}
```

3.9.8 获取集群下的所有私密凭据信息

功能介绍：

该API用于获取集群下所有的私密凭据信息。

URI：

GET /api/v1/secrets

请求示例：

```
GET /api/v1/secrets HTTP/1.1
Accept: application/json
```

返回示例：

```
{
        "kind": "SecretList",
        "apiVersion": "v1",
        "metadata": {
        "selfLink": "/api/v1/secrets",
        "resourceVersion": "6516618"
        },
        "items": [
        {
                "metadata": {
                "name": "ctyun-registry-auth",
                "namespace": "default",
                "selfLink": "/api/v1/namespaces/default/secrets/ctyun-registry-auth",
                "uid": "9c83f21d-869c-11e9-8b35-0050568ccbe7",
                "resourceVersion": "1074733",
                "creationTimestamp": "2019-06-04T07:44:41Z"
                },
                "data": {
                ".dockerconfigjson":
"eyJhdXRocyI6eyJodHRwczovLzIwMi44MC4xOTIuMzk6ODQ0MyI6eyJ1c2VybmFtZSI6ImNhYXN1c2VyIiwicGFzc3dvcmQiOiIzN2ZhYjNkOTgyYTdhNmYyMDVjOWFlODlhODQ4OWYyNiIsImVtYWlsIjoiY2Fhc3VzZXJAY3R5d
```

```
W4uY24iLCJhdXRoIjoiWTJGaGMzVnpaWEk2TXpkbVlXSXpaRGs0TW1FM1lUWm1NakExWXpsaFpUZzVZVGcwT0RsbU1qWT0ifX19"
                    },
                    "type": "kubernetes.io/dockerconfigjson"
          }
          ]
}
```

3.9.9 监听指定的私密凭据

功能介绍：

该API用于监控指定的私密凭据。

URI：

GET /api/v1/watch/namespaces/{namespace}/secrets/{name}

请求实例：

```
GET /api/v1/watch/namespaces/{namespace}/secrets/{name} HTTP/1.1
Accept: application/json
```

返回示例：

```
{
   "type": "ADDED",
   "object": {
                    "kind": "Secret",
                    "apiVersion": "v1",
                    "metadata": {
                       "name": "secret0",
                       "namespace": "namespace0",
                       "selfLink": "/api/v1/namespaces/namespace0/secrets/secret0",
                       "uid": "66343023-a072-11e9-b4da-0050568ccbe7",
                       "resourceVersion": "6515914",
                       "creationTimestamp": "2019-07-07T04:48:01Z"
                    },
                    "data": {
                       "password": "MWYyZDFlMmU2N2Rm",
                        "username": "YWRtaW4="
                    },
                    "type": "Opaque"
   }
}
```

3.9.10 监听指定Namespace下所有的私密凭据

功能介绍：

该API用于监控指定Namespace下所有的私密凭据。

URI：

GET /api/v1/watch/namespaces/{namespace}/secrets

请求实例：

```
GET /api/v1/watch/namespaces/{namespace}/secrets HTTP/1.1
Authorization: Bearer $TOKEN
Accept: application/json
```

返回示例：

```
{
   "type": "ADDED",
   "object": {
                    "kind": "Secret",
                    "apiVersion": "v1",
                    "metadata": {
                       "name": "secret0",
                       "namespace": "namespace0",
                       "selfLink": "/api/v1/namespaces/namespace0/secrets/secret0",
                       "uid": "66343023-a072-11e9-b4da-0050568ccbe7",
                       "resourceVersion": "6515914",
                       "creationTimestamp": "2019-07-07T04:48:01Z"
                    },
                    "data": {
                       "password": "MWYyZDFlMmU2N2Rm",
                        "username": "YWRtaW4="
                    },
                    "type": "Opaque"
   }
}
```

3.9.11 监听集群下所有的私密凭据

功能介绍：

该API用于监控集群下所有的私密凭据。

URI：

GET /api/v1/watch/secrets

请求实例：

```
GET /api/v1/watch/secrets HTTP/1.1
Accept: application/json
```

返回示例：

```
{
   "type": "ADDED",
   "object": {
                    "kind": "Secret",
                    "apiVersion": "v1",
                    "metadata": {
                       "name": "secret0",
                       "namespace": "namespace0",
                       "selfLink": "/api/v1/namespaces/namespace0/secrets/secret0",
                       "uid": "66343023-a072-11e9-b4da-0050568ccbe7",
                       "resourceVersion": "6515914",
                       "creationTimestamp": "2019-07-07T04:48:01Z"
                    },
                    "data": {
                       "password": "MWYyZDFlMmU2N2Rm",
                        "username": "YWRtaW4="
                    },
                    "type": "Opaque"
   }
}
```

3.10 镜像仓库

3.10.1 创建镜像仓库

功能介绍：

创建用户公有或私有仓库

URI：

POST /registry/api/projects

请求示例：

```
POST /registry/api/projects
{
                "project_name":"test",
                 "metadata":{
                                    "public":"true"
```

```
        }
}
```

注意public字段对应的值为字符串，不是布尔型。

3.10.2 查询镜像仓库

功能说明：

查询仓库列表，包括自己所有的仓库和其他人的公开仓库

URI：

GET /registry/api/projects?page=1&page_size=15

参数说明：

参数名	选项	参数类型	说明
page_size	*必填	Int	分页参数，表示每一页包含的仓库数目
page	*必填	Int	页号

请求示例：

```
GET /registry/api/projects?page=1&page_size=15
返回示例：
[
        {
                "project_id": 16,
                "owner_id": 1,
                "name": "ctyun", //仓库名
                "creation_time": "2018-11-28T05:15:12Z",
                "creation_time_str": "",
                "deleted": 0,
                "owner_name": "",
                "public": 1,
                "Togglable": false,
                "update_time": "2018-11-28T05:15:12Z",
                "current_user_role_id": 3,
                "repo_count": 1
        },
        ...
]
```

3.10.3 查询镜像仓库的镜像列表

功能说明：

查询指定id的仓库里的镜像列表

URI：

GET /registry/api/repositories?project_id={id}&page=1&page_size=15

参数说明：

参数名	选项	参数类型	说明
project_id	*必填	Int	仓库的id
page_size	*必填	Int	分页参数，表示每一页包含的仓库数目
page	*必填	Int	页号

请求示例：

```
GET /registry/api/repositories?project_id={id}&page=1&page_size=15
返回示例：
[
        {
                "id": "1",
                "name": "openshift3/openvswitch", //格式为 仓库名/镜像名
                "owner_id": 1,
                "project_id": 3,
                "description": "",
                "pull_count": 0,
                "star_count": 0,
                "tags_count": 3,
                "creation_time": "2017-12-12T02:54:24Z",
                "update_time": "0001-01-01T00:00:00Z"
        },
      (omitted)
]
```

3.10.4 获取镜像版本

功能说明：

获取指定镜像的所有tag信息

URI：

GET /registry/api/repositories/{仓库名}/{镜像名}/tags?detail=1

参数说明：

参数名	选项	参数类型	说明
detail	选填	Int	设置为1为返回更为详细的tag信息，一般不需要

请求示例：

```
GET /registry/api/repositories/library/mysql/tags?detail=1
```

返回示例：


```
[
   {
          "digest": "sha256:98e9c25bb312a85d6660df5bb164b8c8e5486eaea611583e99e5070afc6ed9b2",
          "name": "latest",
          "size": 129401576,
          "architecture": "amd64",
          "os": "linux",
          "os.version": "",
          "docker_version": "18.06.1-ce",
          "author": "",
          "created": "2019-06-10T23:45:17.187524046Z",
          "config": {
          "labels": null
          },
          "signature": null,
          "labels": []
   }
]
```


3.10.5 获取镜像仓库认证信息

功能说明：

获取拉取镜像所需要的所有信息，包含仓库地址、用户名、密码以及用于拉取镜像的secret

URI：

GET /namespaces/{namespace}/images/auth

请求示例：

```
GET /namespaces/{namespace}/images/auth
返回示例：
{
        "UserName":"caasuser",
        "Password":"MzdmYWIzZDk4MmE3YTZmMjA1YzlhZTg5YTg0ODlmMjY=",
        "SecretName":"ctyun-registry-auth",
        "RegistryUrl":"https://10.150.1.20:443"
}
```

3.10.6 修改镜像仓库属性

功能说明：

修改指定仓库的属性为公有或者私有

URI：

PUT /registry/api/projects/{project_id}

参数说明：

参数名	选项	参数类型	说明
project_id	*必填	Int	仓库的id

请求示例：

```
PUT /registry/api/projects/{project_id}
{
        "metadata": {
                "public": "false"
        }
}
```

3.10.7 删除镜像仓库

功能说明：

删除指定仓库

URI：

DELETE /registry/api/projects/{project_id}

参数说明：

参数名	选项	参数类型	说明
project_id	*必填	Int	仓库的id

请求示例：

```
DELETE /registry/api/projects/{project_id}
```

3.10.8 删除镜像

功能说明：

删除指定镜像

URI：

DELETE /registry//api/repositories/{project_id}/{image_name}

参数说明：

参数名	选项	参数类型	说明
project_id	*必填	Int	仓库的id
image_name	*必填	String	镜像名

请求示例：

```
DELETE /registry/api/projects/{project_id}/{image_name}
```

4 附录

状态码：

状态码如表所示

表 状态码

状态码	编码	状态说明
100	Continue	继续请求。 这个临时响应用来通知客户端，它的部分请求已经被服务器接收，且仍未被拒绝。
101	Switching Protocols	切换协议。只能切换到更高级的协议。 例如，切换到HTTP的新版本协议。
201	Created	创建类的请求完全成功。
202	Accepted	已经接受请求，但未处理完成。
203	Non-Authoritative Information	非授权信息，请求成功。
204	NoContent	请求完全成功，同时HTTP响应不包含响应体。 在响应OPTIONS方法的HTTP请求时返回此状态码。
205	Reset Content	重置内容，服务器处理成功。
206	Partial Content	服务器成功处理了部分GET请求。
300	Multiple Choices	多种选择。请求的资源可包括多个位置，相应可返回一个资源特征与地址的列表用于用户终端（例如：浏览器）选择。
301	Moved Permanently	永久移动，请求的资源已被永久的移动到新的URI，返回信息会包括新的URI。
302	Found	资源被临时移动。
303	See Other	查看其它地址。 使用GET和POST请求查看。
304	Not Modified	所请求的资源未修改，服务器返回此状态码时，不会返回任何资源。
305	Use Proxy	所请求的资源必须通过代理访问。
306	Unused	已经被废弃的HTTP状态码。

400	BadRequest	非法请求。 建议直接修改该请求，不要重试该请求。
401	Unauthorized	在客户端提供认证信息后，返回该状态码，表明服务端指出客户端所提供的认证信息不正确或非法。
402	Payment Required	保留请求。
403	Forbidden	请求被拒绝访问。 返回该状态码，表明请求能够到达服务端，且服务端能够理解用户请求，但是拒绝做更多的事情，因为该请求被设置为拒绝访问，建议直接修改该请求，不要重试该请求。
404	NotFound	所请求的资源不存在。 建议直接修改该请求，不要重试该请求。
405	MethodNotAllowed	请求中带有该资源不支持的方法。 建议直接修改该请求，不要重试该请求。
406	Not Acceptable	服务器无法根据客户端请求的内容特性完成请求。
407	Proxy Authentication Required	请求要求代理的身份认证，与401类似，但请求者应当使用代理进行授权。
408	Request Time-out	服务器等候请求时发生超时。 客户端可以随时再次提交该请求而无需进行任何更改。
409	Conflict	服务器在完成请求时发生冲突。 返回该状态码，表明客户端尝试创建的资源已经存在，或者由于冲突请求的更新操作不能被完成。
410	Gone	客户端请求的资源已经不存在。 返回该状态码，表明请求的资源已被永久删除。
411	Length Required	服务器无法处理客户端发送的不带Content-Length的请求信息。
412	Precondition Failed	未满足前提条件，服务器未满足请求者在请求中设置的其中一个前提条件。
413	Request Entity Too Large	由于请求的实体过大，服务器无法处理，因此拒绝请求。为防止客户端的连续请求，服务器可能会关闭连接。如果只是服务器暂时无法处理，则会包含一个Retry-After的响应信息。
414	Request-URI Too Large	请求的URI过长（URI通常为网址），服务器无法处理。
415	Unsupported Media Type	服务器无法处理请求附带的媒体格式。
416	Requested range not satisfiable	客户端请求的范围无效。
417	Expectation Failed	服务器无法满足Expect的请求头信息。

422	UnprocessableEntity	请求格式正确，但是由于含有语义错误，无法响应。
429	TooManyRequests	表明请求超出了客户端访问频率的限制或者服务端接收到多于它能处理的请求。建议客户端读取相应的Retry-After首部，然后等待该首部指出的时间后再重试。
500	InternalServerError	表明服务端能被请求访问到，但是不能理解用户的请求。
501	Not Implemented	服务器不支持请求的功能，无法完成请求。
502	Bad Gateway	充当网关或代理的服务器，从远端服务器接收到了一个无效的请求。
503	ServiceUnavailable	被请求的服务无效。 建议直接修改该请求，不要重试该请求。
504	ServerTimeout	请求在给定的时间内无法完成。客户端仅在为请求指定超时（Timeout）参数时会得到该响应。
505	HTTP Version not supported	服务器不支持请求的HTTP协议的版本，无法完成处理。