容器实例服务

操作指南

产品文档

腾讯云

版权所有：腾讯云计算（北京）有限责任公司

【版权声明】

©2013-2019 腾讯云版权所有

本文档著作权归腾讯云单独所有，未经腾讯云事先书面许可，任何主体不得以任何形式复制、修改、抄袭、传播全部或部分本文档内容。

【商标声明】

腾讯云

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。

【服务声明】

本文档意在向客户介绍腾讯云全部或部分产品、服务的当时的整体概况，部分产品、服务的内容可能有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或模式的承诺或保证。

文档目录

版权所有：腾讯云计算（北京）有限责任公司

操作指南
创建实例

最近更新时间：2019-09-11 12:08:49

操作步骤

1. 登录 腾讯云 CIS 控制台 。

2. 单击容器实例列表页的【新建】。

3. 设置实例的基本信息。

   实例名称：要创建的服务的名称，不超过 63 个字符。服务名称由小写字母、数字和 - 组成，且由小写字母开头，小写字母或数字结尾。

   所在地域：选择您部署容器的所在地域。

   实例网络：选择容器所属的私有网络和网段。

4. 设置实例容器。

   名称：要创建容器的名称，不超过 63 个字符。

镜像：单击【选择镜像】。可选择在我的镜像、我的收藏、TencentHub 镜像、DockerHub 镜像内的镜像。

版本：默认选择最新版本。如果您需要使用镜像的其它版本，单击版本显示框选择。

CPU：选择运行容器所需使用的 CPU 量。该值是容器可用 CPU 的上限值，请谨慎选择。

内存：选择运行容器所需使用的内存量。该值是容器可用内存的上限值，请谨慎选择。

5. 高级设置支持为容器设置环境变量、启动命令和参数，本步骤可选。

6. 单击【添加容器】可以在一个实例内添加多个相关容器。

7. 重启策略。

Always：不管容器实例退出状态码是什么始终重启。该值是默认值。

OnFailure：当容器实例以非 0 状态码退出时重启。

Never：容器实例退出时不会自动重启。

8. 单击【下一步】完成容器实例的创建，在单击【完成】前，请重新浏览并确认实例的各项设置。

查看实例

最近更新时间：2019-09-11 12:07:54

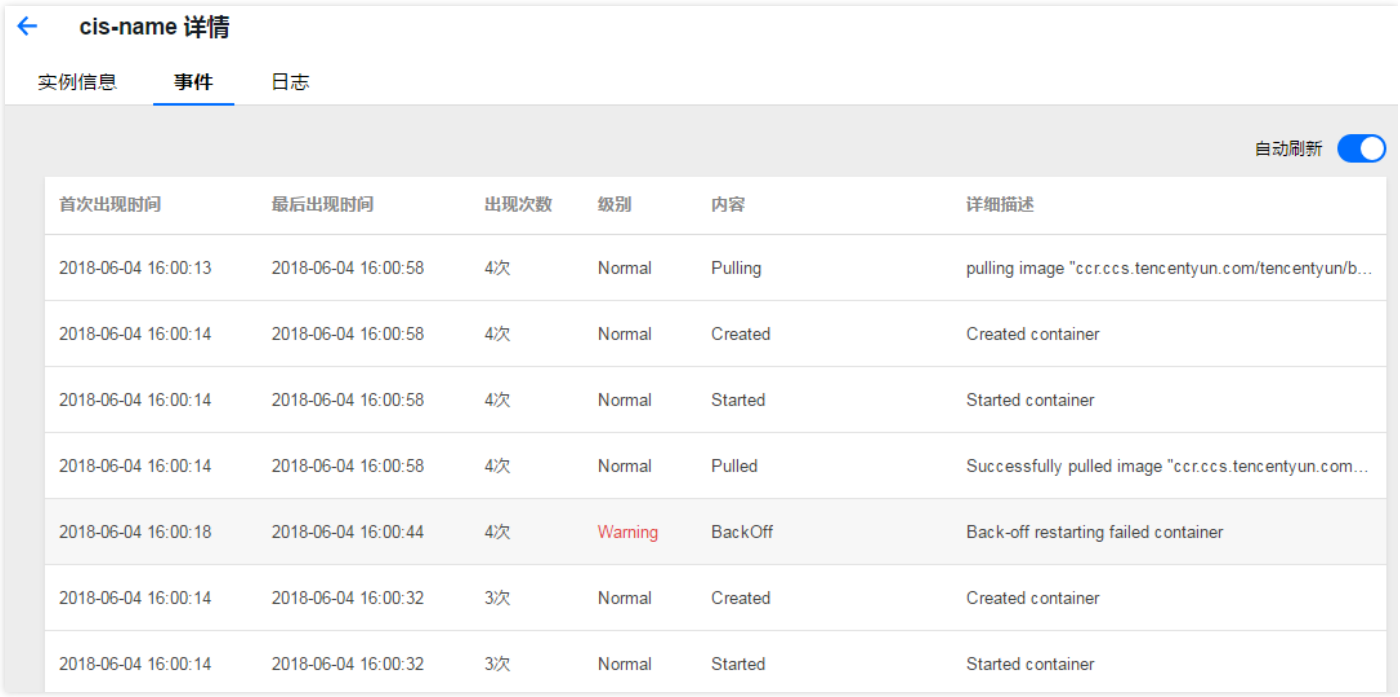
操作步骤

1. 登录 腾讯云 CIS 控制台 。

2. 容器实例页面会展示用户在所选地域的所有容器实例，包括创建中、运行中和已结束状态的实例。用户可以单击右侧【删除】操作删除任何状态的实例，**删除后该实例无法恢复。**

3. 单击实例名，则可查看实例详细信息，包括实例的 VPC 属性、网络地址、所含的容器列表以及每个容器的详细信息。

4. 单击【事件】和【日志】可以查看实例的 Kubernetes event 和实例内容器输出的日志。



配置 NAT 网关访问公网

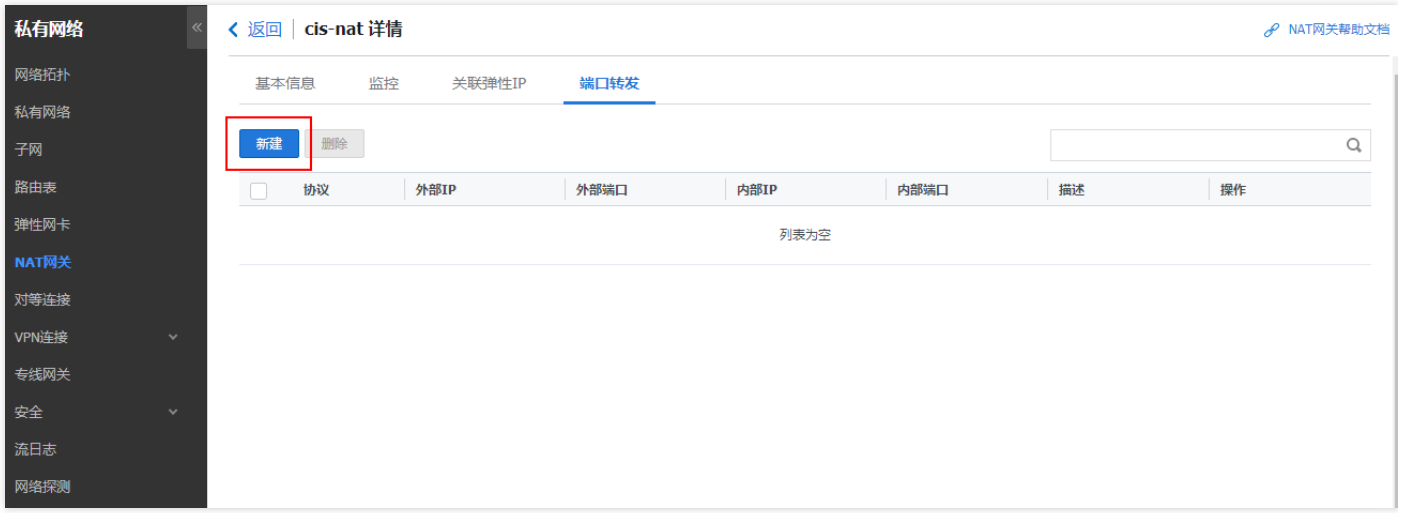
最近更新时间：2019-09-11 12:05:36

概述

CIS 实例支持配置 NAT 网关 和 路由表 ，实现与 Internet 上的资源互访。

配置 Internet 访问 CIS 实例

1. 登录 私有网络控制台，进入左侧导航 **NAT网关** 控制台，单击【新建】，创建与 CIS 实例同地域、同 VPC 的 NAT 网关。



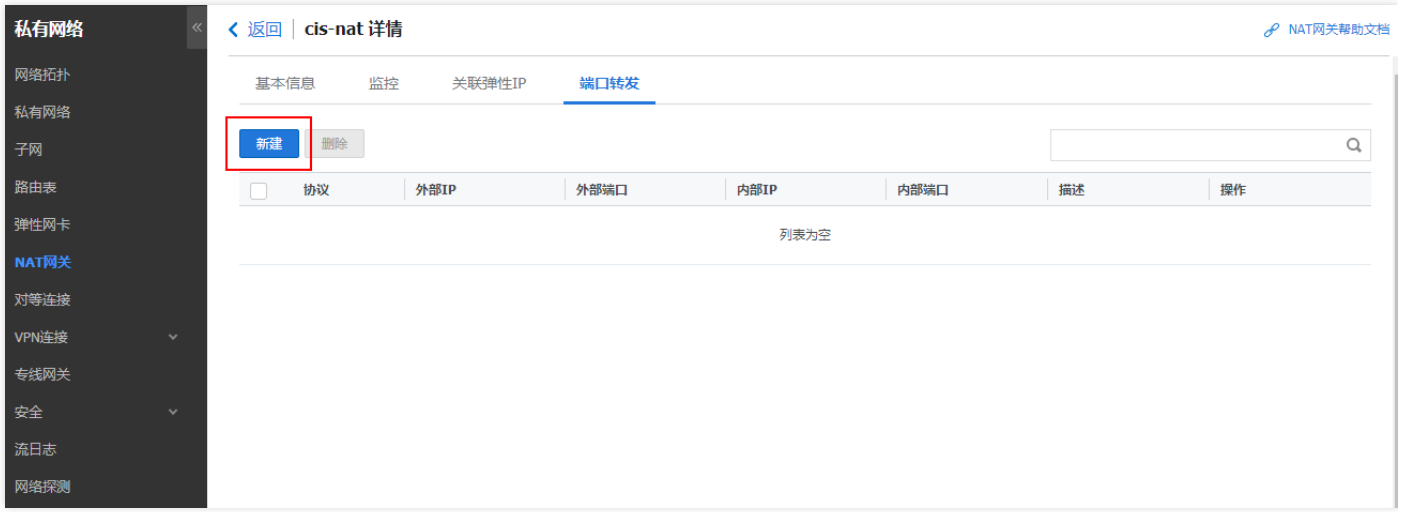
2. NAT网关创建完成后，单击新建 NAT 网关的 **ID/名称**，进入详情页。单击【端口转发】>【新建】开始创建端口转发规则。
   配置规则：
   - 外部 IP 端口：选择 NAT 网关的外网 IP 和期望访问的端口；

- 内部 IP 端口：选择期望访问的 CIS 实例的内网 IP 和占用端口。

3. 完成配置后，即可通过 NAT 网关的外网 IP 和端口访问对应 CIS 实例提供的服务。

配置 CIS 实例访问 Internet

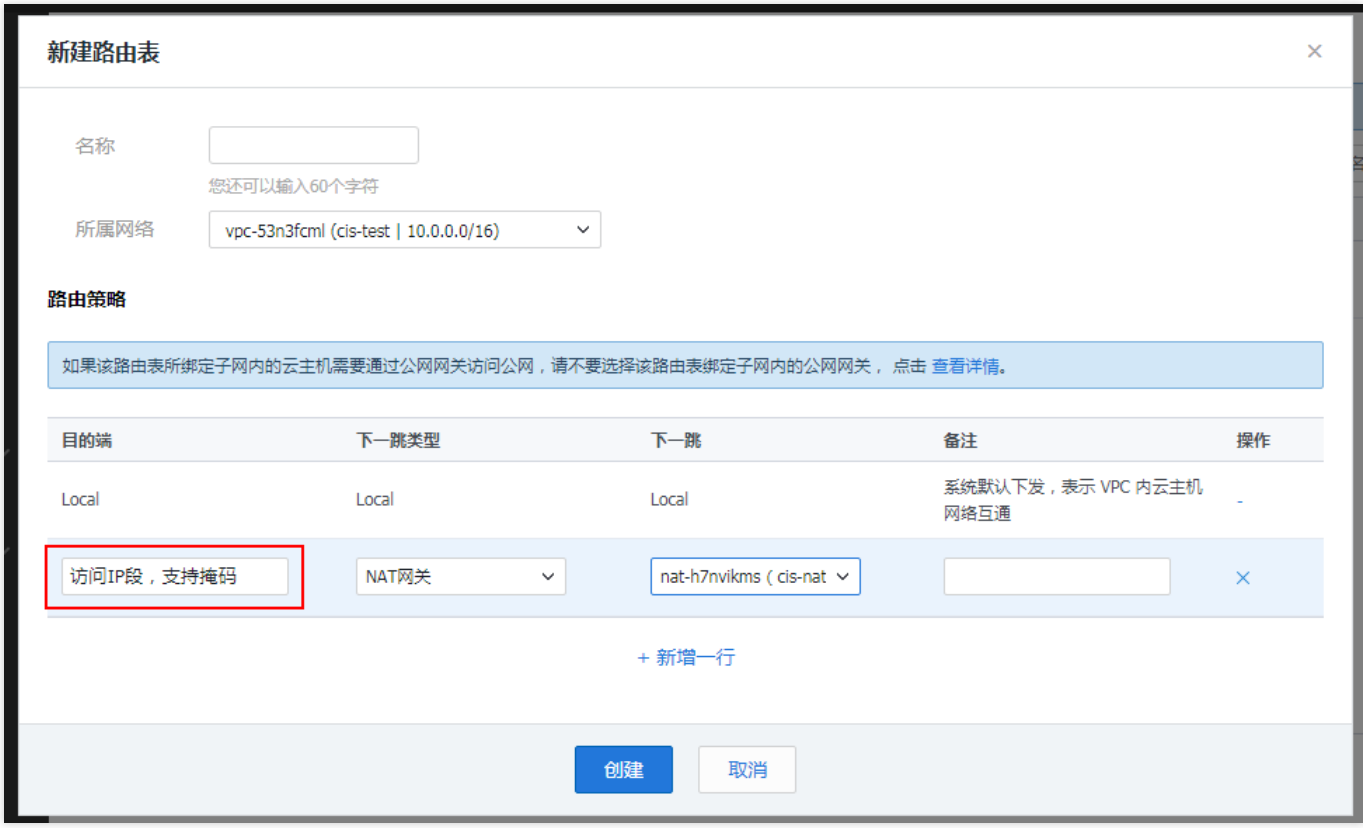
1. 登录 私有网络控制台，进入左侧导航 **NAT网关** 控制台，单击【新建】，创建与 CIS 实例同地域、同 VPC 的 NAT 网关。



2. 进入左侧导航 **路由表** 控制条，单击【新建】，弹出新建路由表窗口，开始创建与 CIS 实例同地域、同 VPC 的路由表。

3. 填写路由表名称、选择所属网络，配置路由策略。
   路由策略配置规则：
   - 目的端：选择要访问的外网 IP 地址，支持配置 CIDR，例如填写 0.0.0.0/0 会转发所有流量到 NAT 网关。
   - 下一跳类型：选择 【 NAT 网关】类型。

- 下一跳：选择 第1步 创建的 NAT 网关。



4. 完成配置路由后，同 VPC 的 CIS 实例即可通过 NAT 网关的外网 IP 访问 Internet。

使用 Kubernetes API 调度

最近更新时间：2019-09-24 11:14:44

概述

在 Kubernetes 集群中部署 Virtual Kubelet 后，即可通过该集群的 apiserver 组件调度管理 CIS 实例。

操作步骤

以在腾讯云容器服务的 Kubernetes 集群中使用 CIS 调度部署 Deployment 为例。

1. 登录任一和腾讯云网络互通的 Kubernetes 节点，例如 TKE 集群、您使用 CVM 自建的 Kubernetes 集群、您使用专线和腾讯云互通的 IDC 自建的 Kubernetes 集群等。
2. 使用 Kubectl 或调用 Kubernetes API 在集群的节点上部署 Virtual Kubelet，详情请参考 virtual-kubelet 部署指南。
3. 完成 Virtual Kubelet 部署后，查看节点和 Pod。

   ```
   kubectl get nodes -o wide
   ```

   ```
   kubectl get pods -o wide
   ```

   会发现集群中会新增一个 Pod 和虚拟节点，名称相同：virtual-kubelet。

   ```
   [root@VM_168_12_centos vk-for-node]# kubectl get nodes -o wide
   NAME              STATUS    ROLES     AGE       VERSION          EXTERNAL-IP       OS-IMAGE                KERNEL-VERSION                CONTAINER-RUNTIME
   10.0.168.12       Ready     <none>    10h       qcloud/v1.8.13   193.112.144.154   CentOS Linux 7 (Core)   3.10.0-514.26.2.el7.x86_64    docker://17.12.1-ce
   virtual-kubelet   Ready     agent     9h        v1.8.3           <none>            <unknown>               <unknown>                     <unknown>
   [root@VM_168_12_centos vk-for-node]# kubectl get pods -o wide
   NAME              READY     STATUS    RESTARTS   AGE       IP           NODE
   virtual-kubelet   1/1       Running   0          9h        172.16.0.7   10.0.168.12
   ```

4. 您可根据实际需求创建特定配置文件，并执行以下命令部署 Deployment ，并指定 nodeName 为 virtual-kubelet，把该 Deployment 的 Pod 调度到虚拟节点 virtual-kubelet 上。

   ```
   kubectl create -f xxxxxx.yaml
   ```

   输出结果如下，则成功部署。

   ```
   [root@VM_168_12_centos vk-for-node]# kubectl create -f service-nginx.yaml
   deployment "vk-nginx0" created
   ```

查看 Pod 状态，示例如下：

```
[root@VM_168_12_centos vk-for-node]# kubectl get pods -o wide
NAME                          READY     STATUS    RESTARTS   AGE       IP            NODE
virtual-kubelet               1/1       Running   0          9h        172.16.0.7    10.0.168.12
vk-nginx0-695469d876-79pn7    1/1       Running   0          52s       10.0.168.5    virtual-kubelet
```

5. 完成部署后，在 TKE 控制台 的【服务】中可以看到刚刚创建的 Deployment 。

但 Deployment 并没有使用 TKE 集群节点的资源，而是把 Pod 创建到了 CIS 中，所以其 Pod 可以在 CIS控制台 的【容器实例】中看到。

virtual-kubelet 部署指南

最近更新时间：2020-01-08 20:39:54

使用准备

virtual-kubelet 依赖文件：

- qcloud-vkubelet.yaml
- virtual-kubelet.yaml
- config/config.toml
- config/server.crt
- config/server.key

```
[root@VM_168_12_centos virtual-kubelet]# ll
total 12
drwxr-xr-x 2 root root 4096 Jun  4 15:29 config
-rwxr-xr-x 1 root root  743 Jun  4 15:29 qcloud-vkubelet.yaml
-rwxr-xr-x 1 root root 1086 Jun  4 15:29 virtual-kubelet.yaml
[root@VM_168_12_centos virtual-kubelet]# ll config/
total 12
-rw-r--r-- 1 root root  209 Jun  4 15:29 config.toml
-rw-r--r-- 1 root root 1285 Jun  4 15:29 server.crt
-rw-r--r-- 1 root root 1675 Jun  4 15:29 server.key
```

详情请参考 virtual kubelet 部署模版 ，该模版支持上传 TKE 节点，解压缩，修改特定参数后直接使用。

1. virtual-kubelet 启动配置文件 config.toml

```
config.toml:
Region = "ap-guangzhou" #创建的 CIS 所在地域
Zone = "ap-guangzhou-4" #创建的 CIS 所在可用区
Version = "2018-04-08"
SecretId = "" #CIS 用户的 SecretId
SecretKey = ""#CIS 用户的 SecretKey
CPU = "100"
Memory = "100Gi"
Pods = "50"
```

其中，Region 和 Zone 的格式分别为 `Region = "地域"` 、 `Zone = "可用区"` ，上述代码以广州可用区4为例。更多可用区信息，参考 地域和可用区。

版权所有：腾讯云计算（北京）有限责任公司

2. virtual-kubelet 10250 端口认证 certfile 及 keyfile：server.crt 和 server.key。
   该 10250 端口主要用于 kubectl logs 功能，当我们使用 kubectl logs 获取 Pod 容器日志时，kube-apiserver 会访问节点的10250端口，获取日志的相关信息。
   在腾讯云 TKE 服务中，我们没有设置 10250 的端口认证，但是 kube-apiserver 需要以 HTTPS 方式访问节点的 10250 端口，否则 kube-apiserver 端将报错。因此，这里需要设置假的 server.key 和 server.crt 用于实现 kubectl logs 功能。

3. virtual-kubelet 的部署文件：
   qcloud-vkubelet.yaml 创建 virtual-kubelet 对应的 serviceaccount，可以操作 Pod 等资源权限：

```
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
name: vkubelet
subjects:
- kind: ServiceAccount
name: vkubelet
namespace: default
roleRef:
kind: ClusterRole
name: vkubelet
apiGroup: rbac.authorization.k8s.io
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRole
metadata:
name: vkubelet
labels:
k8s-app: vkubelet
rules:
- apiGroups: [""] # "" indicates the core API group
resources:
- namespaces
- pods
- pods/status
- nodes
- nodes/status
- secrets
- configmaps
verbs:
- create
- update
```

```
- get
- watch
- list
- delete
---
apiVersion: v1
kind: ServiceAccount
metadata:
name: vkubelet
namespace: default
labels:
k8s-app: vkubelet
---
```

virtual-kubelet.yaml 创建 Pod 运行 virtual-kubelet 程序：

```
apiVersion: v1
kind: Pod
metadata:
name: virtual-kubelet
labels:
k8s-app: vkubelet
spec:
serviceAccountName: vkubelet
restartPolicy: "Never"
imagePullSecrets:
- name: qcloudregistrykey
containers:
- name: virtual-kubelet
image: ccr.ccs.tencentyun.com/tencentyun/virtual-kubelet:dev
imagePullPolicy: Always
env:
- name: KUBELET_PORT
value: "10250"
- name: APISERVER_CERT_LOCATION
value: /etc/virtual-kubelet/server.crt
- name: APISERVER_KEY_LOCATION
value: /etc/virtual-kubelet/server.key
- name: VKUBELET_POD_IP
valueFrom:
fieldRef:
fieldPath: status.podIP
volumeMounts:
- name: credentials
```

```
mountPath: "/etc/virtual-kubelet"
command: ["/usr/bin/virtual-kubelet"]
args: ["--provider", "qcloud", "--namespace", "default", "--provider-config", "/etc/virtual-kubelet/ config.toml"]
volumes:
- name: credentials
hostPath:
path: /home/ubuntu/for-show/config
```

> 说明：
> 请自行修改为部署模版解压后的 config 文件夹路径，内包含 config.toml、server.crt 和 server.key。

使用步骤（ubuntu系统）

1. 登录安装了 kubectl 并已完成了初始化的 Kubernetes 节点服务器。
   kubectl 安装和初始化可参考 使用 kubectl 操作集群。

2. 执行以下命令：

```
kubectl create -f qcloud-vkubelet.yaml
```

执行结果示例：

```
ubuntu@VM-66-110-ubuntu:~/for-show$ kubectl create -f qcloud-vkubelet.yaml
clusterrolebinding "vkubelet" created
clusterrole "vkubelet" created
serviceaccount "vkubelet" created
```

3. 请修改virtual-kubelet.yaml最后一行hostPath内path参数为部署模版解压后的config文件夹路径，并执行以下命令：

```
kubectl create -f virtual-kubelet.yaml
```

执行结果示例：

```
ubuntu@VM-66-110-ubuntu:~/for-show$ kubectl create -f virtual-kubelet.yaml
pod "virtual-kubelet" created
ubuntu@VM-66-110-ubuntu:~/for-show$ kubectl get po
NAME READY STATUSRESTARTS AGE
virtual-kubelet 1/1 Running 0 3s
ubuntu@VM-66-110-ubuntu:~/for-show$ kubectl get no
NAME STATUS ROLES AGE VERSION
192.168.66.110 Ready none 3d v1.8.7-qcloud
192.168.66.16 Ready none 9d v1.8.7-qcloud
virtual-kubelet Ready agent 5s v1.8.3
```

示例及相关说明

示例

```
ubuntu@VM-66-110-ubuntu:~/for-show$ cat busybox-pod-pass.yaml
apiVersion: v1
kind: Pod
metadata:
name: busybox
annotations:
kubernetes.io/cis.vpcId: vpc-lpaa5xe3
kubernetes.io/cis.subnetId: subnet-7z46i306
labels:
qcloud-app: busybox
spec:
containers:
- image: busybox
imagePullPolicy: Always
name: busybox
resources:
requests:
memory: 1Gi
cpu: "1"
limits:
memory: 1Gi
cpu: "1"
command: ["/bin/sh"]
args: ["-c", "while true; do echo hello world; sleep 2; done"]
dnsPolicy: ClusterFirst
nodeName: virtual-kubelet
```

版权所有：腾讯云计算（北京）有限责任公司

说明

1. 指定 CIS 运行所在的 vpcId 和 subnetId。

```
kubernetes.io/cis.vpcId: vpc-lpaa5xe3
kubernetes.io/cis.subnetId: subnet-7z46i306
```

2. 指定 CIS 运行的规格，注意 request 和 limit 保持一致。

```
resources:
requests:
memory: 1Gi
cpu: "1"
limits:
memory: 1Gi
cpu: "1"
```

3. 指定 CIS 运行的节点是 virtual-kubelet。

```
nodeName: virtual-kubelet
```

版权所有：腾讯云计算（北京）有限责任公司