目录

金山云KCE集群对接KCI

如果您已经通过金山云容器服务（Kingsoft Container Engine，简称KCE）创建Kubernetes集群，可以通过集群中部署虚拟节点的方式来接入KCI，使用KCI来承载弹性业务。下面将介绍如何在KCE集群中创建虚拟节点。

前提

- 您已经开通容器实例服务。
- 您已经在金山云容器服务提前创建好一个Kubernetes集群，操作步骤详见创建集群 。

> 注意：当前版本KCI中虚拟节点仅支持适配kubenetes v1.23以下的版本，请确保您创建的容器服务集群对应的kubenetes版本小于v1.23

步骤1：配置Kubeconfig

登录容器服务控制台，获取集群kubeconfig文件，导入集群node节点。以导入路径为“/root/.kube/config”为例，创建secret保存到集群里，供后续virtual-kubelet挂载使用。

```
kubectl create secret generic --from-file=config=/root/.kube/config rbkci-kubeconfig-secret -n kube-system   # /root/.kube/config 可替换为实际保存路径
```

步骤2：创建KCI相关的RBAC资源

登录KCE中的目标集群，验证集群中有无system:kubelet-api-admin这个clusterrole，若没有需通过下述配置创建，若已有则跳过此步：

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: system:kubelet-api-admin
rules:
- apiGroups:
  - ""
  resources:
  - nodes
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - ""
  resources:
  - nodes
  verbs:
  - proxy
- apiGroups:
  - ""
  resources:
  - nodes/log
  - nodes/metrics
  - nodes/proxy
  - nodes/spec
  - nodes/stats
  verbs:
  - '*'
```

创建KCI相关的RBAC资源：

```
# kubectl apply -f rbkci-rbac.yaml
serviceaccount/kcilet-client created
clusterrolebinding.rbac.authorization.k8s.io/kcilet-rb created
serviceaccount/rbkci-virtual-kubelet-sa created
clusterrolebinding.rbac.authorization.k8s.io/rbkci-virtual-kubelet-rb created
```

rbkci-rbac.yaml详情如下：

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: kcilet-client
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: kcilet-rb
```

```
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: system:kubelet-api-admin
subjects:
  - apiGroup: ""
    kind: ServiceAccount
    name: kcilet-client
    namespace: kube-system
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: virtual-kubelet-sa
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: virtual-kubelet-rb
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
  - apiGroup: ""
    kind: ServiceAccount
    name: virtual-kubelet-sa
    namespace: kube-system
```

步骤3：基于virtual-kubelet组件创建虚拟节点

方式一：通过启动参数与环境变量方式创建VK

您需要通过部署virtual-kubelet组件来创建虚拟节点，部署前需准备环境变量与启动参数，详见VK配置参数说明。

按照如下示例部署virtual-kubelet组件，替换示例中的对应参数值。rbkci-virtual-kubelet.yaml示例如下：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: rbkci-virtual-kubelet
  namespace: kube-system
  labels:
    k8s-app: rbkci-virtual-kubelet
spec:
  replicas: 1
  selector:
    matchLabels:
      k8s-app: rbkci-virtual-kubelet
  template:
    metadata:
      name: rbkci-virtual-kubelet
      labels:
        k8s-app: rbkci-virtual-kubelet
    spec:
      serviceAccountName: virtual-kubelet-sa
      containers:
        - name: virtual-kubelet
          image: hub.kce.ksyun.com/ksyun/rbkci-virtual-kubelet:v1.6.0
          args:
            # 自定义虚拟节点名称，默认值为rbkci-virtual-kubelet
            - --nodename=rbkci-virtual-kubelet
            # 指定虚拟节点的DNS配置，KCE集群为集群内coredns服务的IP地址
            - --cluster-dns=10.254.0.10
            # 指定集群域名，KCE集群默认为‘cluster.local’
            - --cluster-domain=cluster.local
            # 指定集群kubeconfig文件路径，值为kubeconfig volume对应的mountPath
            - --kcilet-kubeconfig-path=/root/.kube/config
            # kubernetes版本大于等于1.13建议启用lease资源用于节点心跳
            - --enable-node-lease
            # 虚拟节点管理的所有实例使能kube-proxy
            - --kube-proxy-enable
            # 节点管理的所有实例基础系统盘大小，单位GB，范围20-500GB，Local_SSD最大100GB，其他类型最大500GB。该值为空时CPU机型默认为
20GB，GPU机型默认为50GB
            - --base-system-disk-size=50
          imagePullPolicy: Always
          env:
            - name: VKUBELET_POD_IP
              valueFrom:
                fieldRef:
```

```
                fieldPath: status.podIP
          - name: TEMP_AKSK_CM
            value: user-temp-aksk
          - name: KCI_CLUSTER_ID
            value: ${cluster_id}
          - name: KCI_SUBNET_ID
            value: ${subnet_id}
          - name: KCI_SECURITY_GROUP_IDS
            value: ${security_group_ids}
        volumeMounts:
          - mountPath: /root/.kube
            name: kubeconfig
          - mountPath: /var/log/kci-virtual-kubelet
            name: kci-provider-log
    volumes:
      - name: kubeconfig
        secret:
          secretName: rbkci-kubeconfig-secret
      - name: kci-provider-log
        hostPath:
          path: /var/log/kci-virtual-kubelet
```

方式二：通过ConfigMap配置文件方式创建VK

通过ConfigMap配置文件的方式部署virtual-kubelet组件来创建虚拟节点，virtual-kubelet版本需要**大于等于v1.6.0**，部署前需准备ConfigMap配置参数，详见VK配置参数说明。 按照如下示例部署virtual-kubelet组件，替换示例中的对应参数值。

1.创建virtual-kubelet-config配置文件：

```
apiVersion: v1
data:
  config.yaml: |
      # 用于连接Kubernetes API server (默认: "~/.kube/config")
      kubeconfig: "/root/.kube/config"
      # （必填）虚拟节点名称，集群内需唯一性校验
      nodename: rbkci-virtual-kubelet
      # kubernetes版本大于等于1.13建议启用lease资源用于节点心跳
      enableNodeLease: true
      openapi:
        # KCE集群中临时AK/SK configmap，若未提供用户AccessKey和用户SecretKey则必填
        akskConfigMap:
          name: "user-temp-aksk"
          namespace: "kube-system"
        # 地区名称
        region: "cn-beijing-6"
        # 用户AccessKey,若已配置akskConfigMap此处不用填写
        accessKey: ""
        # 用户SecretKey,若已配置akskConfigMap此处不用填写
        secretKey: ""
       # （必填）集群 ID
      clusterId: "606e4666-9af7-41e5-abee-9411d848f79e"
      # 用于实例内部组件连接到Kubernetes APIServer的kubecconfig的路径
      kciletKubeconfigPath: "/root/.kube/config"
      # （必填）kubernetes集群DNS，若为KCE集群则为集群内coreDNS服务的IP地址
      clusterDNS:
      - 10.96.0.10
      #（必填）kubernetes cluster-domain
      clusterDomain: cluster.local
      # vk上配置默认值，实例可通过pod annotation特例声明
      instanceSettings:
        # （必填）容器实例所属的安全组，支持1-3个
        k8s.ksyun.com/kci-security-group-id: "68e423f2-dc4c-424d-a490-b8ac08250486"
        # （必填）容器实例部署的子网id
        k8s.ksyun.com/kci-subnet-id: "b1086c4b-3e81-4ef5-8de6-4c4bdc93d4ba"
kind: ConfigMap
metadata:
  name: virtual-kubelet-config
  namespace: kube-system
```

2.部署rbkci-virtual-kubelet.yaml：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: rbkci-virtual-kubelet
  namespace: kube-system
  labels:
    k8s-app: rbkci-virtual-kubelet
spec:
  replicas: 1
```

```
  selector:
    matchLabels:
      k8s-app: rbkci-virtual-kubelet
  template:
    metadata:
      name: rbkci-virtual-kubelet
      labels:
        k8s-app: rbkci-virtual-kubelet
    spec:
      serviceAccountName: virtual-kubelet-sa
      containers:
        - name: virtual-kubelet
          image: hub.kce.ksyun.com/ksyun/rbkci-virtual-kubelet:v1.6.0
          args:
            - --kciVirtualKubeletConfigFile=/root/virtual-kubelet/config.yaml
          imagePullPolicy: Always
          env:
            - name: VKUBELET_POD_IP
              valueFrom:
                fieldRef:
                  fieldPath: status.podIP
          volumeMounts:
            - mountPath: /root/.kube
              name: kubeconfig
            - mountPath: /var/log/kci-virtual-kubelet
              name: kci-provider-log
            - mountPath: /root/virtual-kubelet
              name: virtual-kubelet-config
      volumes:
        - name: kubeconfig
          secret:
            secretName: rbkci-kubeconfig-secret
        - name: kci-provider-log
          hostPath:
            path: /var/log/kci-virtual-kubelet
        - name: virtual-kubelet-config
          configMap:
            name: virtual-kubelet-config
```

提示：推荐使用ConfigMap配置文件的方式创建VK来部署虚拟节点，但是如果rbkci-virtual-kubelet.yaml中同时使用了ConfigMap配置文件、环境变量、启动参数，那么他们之间的优先级关系为：
环境变量、启动参数 > 配置文件

部署并验证：

```
# kubectl apply -f rbkci-virtual-kubelet.yaml

# kubectl get deployment -n kube-system | grep rbkci
rbkci-virtual-kubelet                1/1     1            1           35s

# kubectl get node
NAME                     STATUS   ROLES    AGE    VERSION
192.168.1.106            Ready    master   127d   v1.21.3
192.168.1.141            Ready    master   127d   v1.21.3
192.168.1.149            Ready    master   127d   v1.21.3
192.168.3.212            Ready    node     51d    v1.21.3
192.168.3.73             Ready    node     51d    v1.21.3
rbkci-virtual-kubelet    Ready    agent    40s    v1.19.3-vk-v1.5.0
```

步骤4：调度KCI容器实例到虚拟节点

通过YAML创建

Kubernetes集群通过虚拟节点创建Pod到KCI时，可以通过在 yaml 中定义 annotation 的方式，实现为 Pod 绑定安全组、分配资源等能力。 KCI目前支持的Annotation列表如下：

Annotation Key	Annotation Value示例	是否必填	描述
k8s.ksyun.com/kci-instance-cpu	4	否	指定容器实例CPU核数，单位：核
k8s.ksyun.com/kci-instance-memory	8	否	指定容器实例内存，单位：GiB

k8s.ksyun.com/kci-instance-type	如'S6.4B'	否	指定云服务器资源套餐类型，仅支持单个套餐类型
k8s.ksyun.com/kci-security-group-id	xxxxxxxx,xxxxxxxxx	否	支持填写多个，virtual-kubelet启动时，通过配置参数设置默认安全组，所有创建在虚拟节点上的Pod默认使用virtual-kubelet配置的安全组创建KCI实例，如果用户希望使用同VPC下其他安全组创建KCI实例，需要通过注解方式显示指定安全组
k8s.ksyun.com/kci-subnet-id	xxxxxxx	否	virtual-kubelet启动时，通过配置参数设置默认的子网，所有创建在虚拟节点上的Pod默认在virtual-kubelet配置的子网下创建KCI实例，如果用户希望在同VPC下其他子网创建KCI实例，需要通过注解方式显示指定子网
k8s.ksyun.com/kci-kube-proxy-enabled	'true'/ 'false'	否	默认值：'false'。当为true时，为该pod开启kube-proxy，使该pod具备访问集群内ClusterIP类型服务的能力；否则不开启。
k8s.ksyun.com/kci-dns-config	'{"nameservers":["1.1.1.1"],"options":[{"name":"ndots","value":"2"},{"name":"timeout","value":"3"}],"searches":["test1.com"]}'	否	使用自建镜像仓库时，若未在vk维度配置，可在实例维度配置dnsconfig，生效于实例底层系统内，用于解析镜像仓库地址，必须为json字符串格式（参考Pod's DNS Config）
k8s.ksyun.com/kci-host-aliases	'[{"ip":"1.2.3.4","hostnames":["www.privaterepo1.com"]},{"ip":"2.3.4.5","hostnames":["www.privaterepo2.com"]}]'	否	使用自建镜像仓库时，若未在vk维度配置，可在实例维度配置hostAliases，生效于实例底层系统内，用于解析镜像仓库地址，必须为json字符串格式（参考Adding additional entries with hostAliases）
k8s.ksyun.com/kci-base-system-disk-size	"50"	否	设置实例基础系统盘大小，单位GB，默认20GB，范围20-500GB，Local_SSD最 100GB，其他类型最 500GB
k8s.ksyun.com/kci-base-system-disk-type	Local_SSD/SSD3.0/EHDD	否	设置实例基础系统盘类型， 持三种类型：Local_SSD/SSD3.0/EHDD，该值为空时系统会 动适配
k8s.ksyun.com/kci-base-image	xxxxxxx	否	指定容器实例的基础镜像ID
k8s.ksyun.com/kci-charge-type	HourlyInstantSettlement/Spot	否	容器实例的计费方式，支持按小时实时结算和竞价型实例。不指定计费方式时，默认为按小时实时结算
k8s.ksyun.com/kci-spot-strategy	SpotAsPriceGo	否，若k8s.ksyun.com/kci-charge-type设置为Spot则必填	竞价实例的抢占策略。 SpotAsPriceGo：系统自动出价，出价为固定折扣乘以列表价
k8s.ksyun.com/kci-retain-ip	'true'/ 'false'	否	默认值：'false'。当为true时，为该Pod开启固定IP功能。开启此功能的Pod，当Pod被销毁后默认会保留这个Pod的IP 24小时。24小时内重建同名的Pod，还能使用该IP。24小时以后，该IP有可能被其他Pod抢占。仅对statefulset、rawpod生效
k8s.ksyun.com/kci-retain-ip-hours	"48"	否	修改Pod固定IP的保留时长，单位是小时。如Pod 销毁之后超过这个时长没有创建回来，IP将被释放。默认是24小时，最大可支持保留一年。仅对statefulset、rawpod生效
k8s.ksyun.com/kci-eip-allocation-id	xxxxxxx	否	指定容器实例绑定的EIP实例ID
k8s.ksyun.com/kci-core-pattern	"/tmp/cores/core.%h.%e.%p.%t"	否	设置容器实例的Core dump文件保存目录
k8s.ksyun.com/kci-http-registry	192.168.xx.xx:5000	否	使用HTTP协议的自建镜像仓库中的镜像创建容器实例时，需配置该Annotation使得容器实例使用HTTP协议拉取镜像，避免因协议不同而导致镜像拉取失败。支持配置多个仓库域名或IP，以逗号隔开

k8s.ksyun.com/kci-insecure-registry	harbor.example.com	否	使用自签发证书的自建镜像仓库中的镜像创建容器实例时，需配置该Annotation来跳过证书认证，避免因证书认证失败而导致镜像拉取失败。支持配置多个仓库域名或IP，以逗号隔开
k8s.ksyun.com/kci-project-id	“1700”	否	指定容器实例的项目ID
k8s.ksyun.com/kci-image-cache-id	xxxxxxx	否	指定镜像缓存ID。 如果指定了镜像缓存id同时开启了自动匹配镜像缓存，则仍以指定的镜像缓存id为准，自动匹配镜像缓存的功能不会生效
k8s.ksyun.com/kci-auto-image-cache	’true’/ ’false’	否	默认值：’false’，当为true时，为该Pod开启自动匹配镜像缓存的功能

注：

1. 若指定KCI Pod的规格，需要同时设置k8s.ksyun.com/kci-instance-cpu和k8s.ksyun.com/kci-instance-memory参数。
2. 除了通过指定CPU和内存的方式创建KCI实例，在对实例规格有特殊需求的场景（如：网络吞吐量、网卡队列数等），您也可以指定KCI实例底层所使用的云服务器套餐规格来创建实例，套餐规格可参考支持的云服务器类型。
3. 在不指定KCI规格或套餐时，金山云容器服务会自动规整KCI Pod的规格，详细计算方法请参考指定KCI Pod规格。
4. 虚拟节点创建时默认存在污点rbkci-virtual-kubelet.io/provider:kingsoftcloud，若要将容器实例调度到虚拟节点，可通过为pod配置容忍度或指定节点调度实现。
5. 如您从私有镜像仓库拉取镜像，需配置镜像访问凭证，详细配置方法请参考此文档的镜像访问凭证章节。

您可以通过节点亲和性配合节点容忍调度pod到虚拟节点上运行，yaml示例如下：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-rbkci
  namespace: default
  labels:
    app: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
        annotations:
            k8s.ksyun.com/kci-instance-type: N3.2B  #指定N3机型，2核4G的云服务器为容器实例底层资源
        labels:
            app: nginx
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: type
                operator: In
                values:
                - virtual-kubelet
      containers:
      - name: nginx
        image: nginx:latest
      tolerations:
      - key: rbkci-virtual-kubelet.io/provider
        value: kingsoftcloud
        effect: NoSchedule
```

验证

登录容器服务控制台，点击**工作负载**>**Deployment**>**Pod列表**，查询资源的创建状态。

登录容器实例控制台，查看KCI的创建情况。

后续操作

升级Virtual Kubelet

随容器实例功能迭代，Virtual Kubelet版本会相应进行更新，目前您可通过手动修改VK镜像版本的方式进行升级。具体操作步骤如下：

步骤1：将对应虚拟节点置为不可调度

```
# kubectl cordon rbkci-virtual-kubelet
node/rbkci-virtual-kubelet cordoned
```

步骤2：更新虚拟节点对应的Deployment镜像版本

```
# kubectl edit deployments.apps -n kube-system rbkci-virtual-kubelet
```

示例如下：

```
...
    spec:
      containers:
      - args:
        - --nodename=rbkci-virtual-kubelet
        - --cluster-dns=10.254.0.10
        - --cluster-domain=cluster.local
        - --kcilet-kubeconfig-path=/root/.kube/config
        env:
        - name: VKUBELET_POD_IP
          valueFrom:
            fieldRef:
              apiVersion: v1
              fieldPath: status.podIP
        - name: TEMP_AKSK_CM
          value: user-temp-aksk
        - name: KCI_CLUSTER_ID
          value: b4327dda-bfeb-43b4-b424-d6dbcb1a388f
        - name: KCI_SUBNET_ID
          value: 588e25a2-e052-4620-913a-38519f59563c
        - name: KCI_SECURITY_GROUP_IDS
          value: 379103c8-2439-44ac-95e7-82e77e6fdf75
        image: hub.kce.ksyun.com/ksyun/rbkci-virtual-kubelet:v1.5.0    #将镜像tag改为最新版本
        imagePullPolicy: Always
        name: virtual-kubelet
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
        volumeMounts:
        - mountPath: /root/.kube/config
          name: kubeconfig
        - mountPath: /var/log/kci-virtual-kubelet
          name: kci-provider-log
 ...
```

步骤3：将对应节点恢复为可调度状态

在rbkci-virtual-kubelet Deployment滚动更新完成后，将对应节点恢复为可调度状态。

```
# kubectl uncordon rbkci-virtual-kubelet
node/rbkci-virtual-kubelet uncordoned
```

自建Kubernetes集群使用KCI

虚拟节点支持在用户自建的Kubernetes集群中接入，下面将介绍如何在自建Kuberentes集群中部署虚拟节点，以及如何通过虚拟节点创建容器实例。

前提

- 您已经部署好Kubernetes集群，Kubernetes版本为v1.15及以上，且v1.23以下；
- 您的Kubernetes集群已经通过专线或VPN和金山云的VPC网络打通。

步骤1：配置Kubeconfig

将集群Kubeconfig文件导入node节点，创建为secret保存到集群里（以路径为“/root/.kube/config”为例），供后续

virtual-kubelet挂载使用。

```
kubectl create secret generic --from-file=config=/root/.kube/config rbkci-kubeconfig-secret -n kube-system   # /root/.kube/config 可替换为实际保存路径
```

步骤2：创建KCI相关的RBAC资源

登录自建集群，验证集群中有无system:kubelet-api-admin这个clusterrole，若没有需通过下述配置创建，若已有则跳过此步：

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: system:kubelet-api-admin
rules:
- apiGroups:
  - ""
  resources:
  - nodes
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - ""
  resources:
  - nodes
  verbs:
  - proxy
- apiGroups:
  - ""
  resources:
  - nodes/log
  - nodes/metrics
  - nodes/proxy
  - nodes/spec
  - nodes/stats
  verbs:
  - '*'
```

创建KCI相关的RBAC资源：

```
# kubectl apply -f rbkci-rbac.yaml
serviceaccount/kcilet-client created
clusterrolebinding.rbac.authorization.k8s.io/kcilet-rb created
serviceaccount/rbkci-virtual-kubelet-sa created
clusterrolebinding.rbac.authorization.k8s.io/rbkci-virtual-kubelet-rb created
```

rbkci-rbac.yaml详情如下：

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: kcilet-client
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: kcilet-rb
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: system:kubelet-api-admin
subjects:
  - apiGroup: ""
    kind: ServiceAccount
    name: kcilet-client
    namespace: kube-system
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: virtual-kubelet-sa
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: virtual-kubelet-rb
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
```

```
  name: cluster-admin
subjects:
  - apiGroup: ""
    kind: ServiceAccount
    name: virtual-kubelet-sa
    namespace: kube-system
```

步骤3：基于virtual-kubelet组件创建虚拟节点

方式一：通过启动参数与环境变量方式创建VK

您需要通过部署virtual-kubelet组件来创建虚拟节点，部署前需准备环境变量与启动参数，详见VK配置参数说明。

按照如下示例部署virtual-kubelet组件，替换示例中的对应参数值。rbkci-virtual-kubelet.yaml示例如下：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: rbkci-virtual-kubelet
  namespace: kube-system
  labels:
    k8s-app: rbkci-virtual-kubelet
spec:
  replicas: 1
  selector:
    matchLabels:
      k8s-app: rbkci-virtual-kubelet
  template:
    metadata:
      name: rbkci-virtual-kubelet
      labels:
        k8s-app: rbkci-virtual-kubelet
    spec:
      serviceAccountName: virtual-kubelet-sa
      containers:
        - name: virtual-kubelet
          image: hub.kce.ksyun.com/ksyun/rbkci-virtual-kubelet:v1.6.0
          args:
            # 自定义虚拟节点名称，默认值为rbkci-virtual-kubelet
            - --nodename=rbkci-virtual-kubelet
            # 指定虚拟节点的DNS配置，KCE集群为集群内coredns服务的IP地址
            - --cluster-dns=10.254.0.10
            # 指定集群域名，KCE集群默认为‘cluster.local’
            - --cluster-domain=cluster.local
            # 指定集群kubeconfig文件路径，值为kubeconfig volume对应的mountPath
            - --kcilet-kubeconfig-path=/root/.kube/config
            # kubernetes版本大于等于1.13建议启用lease资源用于节点心跳
            - --enable-node-lease
            # 虚拟节点管理的所有实例使能kube-proxy
            - --kube-proxy-enable
            # 节点管理的所有实例基础系统盘大小，单位GB，范围20-500GB，Local_SSD最大100GB，其他类型最大500GB。该值为空时CPU机型默认为20GB，GPU机型默认为50GB
            - --base-system-disk-size=50
          imagePullPolicy: Always
          env:
            - name: VKUBELET_POD_IP
              valueFrom:
                fieldRef:
                  fieldPath: status.podIP
            - name: TEMP_AKSK_CM
              value: user-temp-aksk
            - name: KCI_CLUSTER_ID
              value: ${cluster_id}
            - name: KCI_SUBNET_ID
              value: ${subnet_id}
            - name: KCI_SECURITY_GROUP_IDS
              value: ${security_group_ids}
          volumeMounts:
            - mountPath: /root/.kube
              name: kubeconfig
            - mountPath: /var/log/kci-virtual-kubelet
              name: kci-provider-log
      volumes:
        - name: kubeconfig
          secret:
            secretName: rbkci-kubeconfig-secret
        - name: kci-provider-log
          hostPath:
            path: /var/log/kci-virtual-kubelet
```

方式二：通过ConfigMap配置文件方式创建VK

通过ConfigMap配置文件的方式部署virtual-kubelet组件来创建虚拟节点，virtual-kubelet版本需要**大于等于v1.6.0**，部署前需准备ConfigMap配置参数，详见VK配置参数说明。 按照如下示例部署virtual-kubelet组件，替换示例中的对应参数值。

1.创建virtual-kubelet-config配置文件：

```
apiVersion: v1
data:
  config.yaml: |
      # 用于连接Kubernetes API server (默认: "~/.kube/config")
      kubeconfig: "/root/.kube/config"
      # （必填）虚拟节点名称，集群内需唯一性校验
      nodename: rbkci-virtual-kubelet
      # kubernetes版本大于等于1.13建议启用lease资源用于节点心跳
      enableNodeLease: true
      openapi:
        # KCE集群中临时AK/SK configmap，若未提供用户AccessKey和用户SecretKey则必填
        akskConfigMap:
          name: "user-temp-aksk"
          namespace: "kube-system"
        # 地区名称
        region: "cn-beijing-6"
        # 用户AccessKey,若已配置akskConfigMap此处不用填写
        accessKey: ""
        # 用户SecretKey,若已配置akskConfigMap此处不用填写
        secretKey: ""
       # （必填）集群 ID
      clusterId: "606e4666-9af7-41e5-abee-9411d848f79e"
      # 用于实例内部组件连接到Kubernetes APIServer的kubecconfig的路径
      kciletKubeconfigPath: "/root/.kube/config"
      # （必填）kubernetes集群DNS，若为KCE集群则为集群内coreDNS服务的IP地址
      clusterDNS:
      - 10.96.0.10
      #（必填）kubernetes cluster-domain
      clusterDomain: cluster.local
      # vk上配置默认值，实例可通过pod annotation特例声明
      instanceSettings:
        # （必填）容器实例所属的安全组，支持1-3个
        k8s.ksyun.com/kci-security-group-id: "68e423f2-dc4c-424d-a490-b8ac08250486"
        # （必填）容器实例部署的子网id
        k8s.ksyun.com/kci-subnet-id: "b1086c4b-3e81-4ef5-8de6-4c4bdc93d4ba"
kind: ConfigMap
metadata:
  name: virtual-kubelet-config
  namespace: kube-system
```

2.部署rbkci-virtual-kubelet.yaml：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: rbkci-virtual-kubelet
  namespace: kube-system
  labels:
    k8s-app: rbkci-virtual-kubelet
spec:
  replicas: 1
  selector:
    matchLabels:
      k8s-app: rbkci-virtual-kubelet
  template:
    metadata:
      name: rbkci-virtual-kubelet
      labels:
        k8s-app: rbkci-virtual-kubelet
    spec:
      serviceAccountName: virtual-kubelet-sa
      containers:
        - name: virtual-kubelet
          image: hub.kce.ksyun.com/ksyun/rbkci-virtual-kubelet:v1.6.0
          args:
            - --kciVirtualKubeletConfigFile=/root/virtual-kubelet/config.yaml
          imagePullPolicy: Always
          env:
            - name: VKUBELET_POD_IP
              valueFrom:
                fieldRef:
                  fieldPath: status.podIP
          volumeMounts:
            - mountPath: /root/.kube
              name: kubeconfig
            - mountPath: /var/log/kci-virtual-kubelet
```

```
            name: kci-provider-log
          - mountPath: /root/virtual-kubelet
            name: virtual-kubelet-config
      volumes:
        - name: kubeconfig
          secret:
            secretName: rbkci-kubeconfig-secret
        - name: kci-provider-log
          hostPath:
            path: /var/log/kci-virtual-kubelet
        - name: virtual-kubelet-config
          configMap:
            name: virtual-kubelet-config
```

提示：推荐使用ConfigMap配置文件的方式创建VK来部署虚拟节点，但是如果rbkci-virtual-kubelet.yaml中同时使用了ConfigMap配置文件、环境变量、启动参数，那么他们之间的优先级关系为：
环境变量、启动参数 > 配置文件

部署并验证：

```
# kubectl apply -f rbkci-virtual-kubelet.yaml

# kubectl get deployment -n kube-system | grep rbkci
rbkci-virtual-kubelet                1/1     1            1           35s

# kubectl get node
NAME                     STATUS   ROLES    AGE    VERSION
192.168.1.106            Ready    master   127d   v1.21.3
192.168.1.141            Ready    master   127d   v1.21.3
192.168.1.149            Ready    master   127d   v1.21.3
192.168.3.212            Ready    node     51d    v1.21.3
192.168.3.73             Ready    node     51d    v1.21.3
rbkci-virtual-kubelet    Ready    agent    40s    v1.19.3-vk-v1.5.0
```

步骤4：调度KCI容器实例到虚拟节点

通过YAML创建

Kubernetes集群通过虚拟节点创建Pod到KCI时，可以通过在 yaml 中定义 annotation 的方式，实现为 Pod 绑定安全组、分配资源等能力。 KCI目前支持的Annotation列表如下：

Annotation Key	Annotation Value示例	是否必填	描述
k8s.ksyun.com/kci-instance-cpu	4	否	指定容器实例CPU核数，单位：核
k8s.ksyun.com/kci-instance-memory	8	否	指定容器实例内存，单位：GiB
k8s.ksyun.com/kci-instance-type	如 'S6.4B'	否	指定云服务器资源套餐类型，仅支持单个套餐类型
k8s.ksyun.com/kci-security-group-id	xxxxxxxx,xxxxxxxxx	否	支持填写多个，virtual-kubelet启动时，通过配置参数设置默认安全组，所有创建在虚拟节点上的Pod默认使用virtual-kubelet配置的安全组创建KCI实例，如果用户希望使用同VPC下其他安全组创建KCI实例，需要通过注解方式显示指定安全组
k8s.ksyun.com/kci-subnet-id	xxxxxxx	否	virtual-kubelet启动时，通过配置参数设置默认的子网，所有创建在虚拟节点上的Pod默认在virtual-kubelet配置的子网下创建KCI实例，如果用户希望在同VPC下其他子网创建KCI实例，需要通过注解方式显示指定子网
k8s.ksyun.com/kci-kube-proxy-enabled	true' / 'false'	否	默认值：'false'。当为true时，为该pod开启kube-proxy，使该pod具备访问集群内ClusterIP类型服务的能力；否则不开启。
k8s.ksyun.com/kci-dns-config	'{"nameservers":["1.1.1.1"],"options":[{"name":"ndots","value":"2"},{"name":"timeout","value":"3"}],"searches":["test1.com"]}'	否	使用自建镜像仓库时，若未在vk维度配置，可在实例维度配置dnsconfig，生效于实例底层系统内，用于解析镜像仓库地址，必须为json字符串格式（参考Pod's DNS Config）

k8s.ksyun.com/kci-host-aliases	'[{"ip":"1.2.3.4","hostnames":["www.privaterepo1.com"]},{"ip":"2.3.4.5","hostnames":["www.privaterepo2.com"]}]'	否	使用自建镜像仓库时，若未在vk维度配置，可在实例维度配置hostAliases，生效于实例底层系统内，用于解析镜像仓库地址，必须为json字符串格式（参考Adding additional entries with hostAliases）
k8s.ksyun.com/kci-base-system-disk-size	"50"	否	设置实例基础系统盘大小，单位GB，默认20GB，范围20-500GB，Local_SSD最 100GB，其他类型最 500GB
k8s.ksyun.com/kci-base-system-disk-type	Local_SSD/SSD3.0/EHDD	否	设置实例基础系统盘类型， 持三种类型：Local_SSD/SSD3.0/EHDD，该值为空时系统会 动适配
k8s.ksyun.com/kci-base-image	xxxxxxx	否	指定容器实例的基础镜像ID
k8s.ksyun.com/kci-charge-type	HourlyInstantSettlement/Spot	否	容器实例的计费方式，支持按小时实时结算和竞价型实例。不指定计费方式时，默认为按小时实时结算
k8s.ksyun.com/kci-spot-strategy	SpotAsPriceGo	否，若k8s.ksyun.com/kci-charge-type设置为Spot则必填	竞价实例的抢占策略。 SpotAsPriceGo：系统自动出价，出价为固定折扣乘以列表价
k8s.ksyun.com/kci-retain-ip	'true'/ 'false'	否	默认值：'false'。当为true时，为该Pod开启固定IP功能。开启此功能的Pod，当Pod被销毁后默认会保留这个Pod的IP 24小时。24小时内重建同名的Pod，还能使用该IP。24小时以后，该IP有可能被其他Pod抢占。仅对statefulset、rawpod生效
k8s.ksyun.com/kci-retain-ip-hours	"48"	否	修改Pod固定IP的保留时长，单位是小时。如Pod 销毁之后超过这个时长没有创建回来，IP将被释放。默认是24小时，最大可支持保留一年。仅对statefulset、rawpod生效
k8s.ksyun.com/kci-eip-allocation-id	xxxxxxx	否	指定容器实例绑定的EIP实例ID
k8s.ksyun.com/kci-core-pattern	"/tmp/cores/core.%h.%e.%p.%t"	否	设置容器实例的Core dump文件保存目录
k8s.ksyun.com/kci-http-registry	192.168.xx.xx:5000	否	使用HTTP协议的自建镜像仓库中的镜像创建容器实例时，需配置该Annotation使得容器实例使用HTTP协议拉取镜像，避免因协议不同而导致镜像拉取失败。支持配置多个仓库域名或IP，以逗号隔开
k8s.ksyun.com/kci-insecure-registry	harbor.example.com	否	使用自签发证书的自建镜像仓库中的镜像创建容器实例时，需配置该Annotation来跳过证书认证，避免因证书认证失败而导致镜像拉取失败。支持配置多个仓库域名或IP，以逗号隔开
k8s.ksyun.com/kci-project-id	“1700”	否	指定容器实例的项目ID
k8s.ksyun.com/kci-image-cache-id	xxxxxxx	否	指定镜像缓存ID。 如果指定了镜像缓存id同时开启了自动匹配镜像缓存，则仍以指定的镜像缓存id为准，自动匹配镜像缓存的功能不会生效
k8s.ksyun.com/kci-auto-image-cache	'true'/ 'false'	否	默认值：'false'，当为true时，为该Pod开启自动匹配镜像缓存的功能

注：

1. 若指定KCI Pod的规格，需要同时设置k8s.ksyun.com/kci-instance-cpu和k8s.ksyun.com/kci-instance-memory参数。
2. 除了通过指定CPU和内存的方式创建KCI实例，在对实例规格有特殊需求的场景（如：网络吞吐量、网卡队列数等），您也可以指定KCI实例底层所使用的云服务器套餐规格来创建实例，套餐规格可参考支持的云服务器类型。
3. 在不指定KCI规格或套餐时，金山云容器服务会自动规整KCI Pod的规格，详细计算方法请参考指定KCI Pod

规格。

4. 虚拟节点创建时默认存在污点rbkci-virtual-kubelet.io/provider:kingsoftcloud，若要将容器实例调度到虚拟节点，可通过为pod配置容忍度或指定节点调度实现。
5. 如您从私有镜像仓库拉取镜像，需配置镜像访问凭证，详细配置方法请参考此文档的镜像访问凭证章节。

您可以通过节点亲和性配合节点容忍调度pod到虚拟节点上运行，yaml示例如下：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-rbkci
  namespace: default
  labels:
    app: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
        annotations:
            k8s.ksyun.com/kci-instance-type: N3.2B  #指定N3机型，2核4G的云服务器为容器实例底层资源
        labels:
            app: nginx
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: type
                operator: In
                values:
                - virtual-kubelet
      containers:
      - name: nginx
        image: nginx:latest
      tolerations:
      - key: rbkci-virtual-kubelet.io/provider
        value: kingsoftcloud
        effect: NoSchedule
```

验证

登录容器实例控制台，查看KCI的创建情况。

后续操作

升级Virtual Kubelet

随容器实例功能迭代，Virtual Kubelet版本会相应进行更新，目前您可通过手动修改VK镜像版本的方式进行升级。具体操作步骤如下：

步骤1：将对应虚拟节点置为不可调度

```
# kubectl cordon rbkci-virtual-kubelet
node/rbkci-virtual-kubelet cordoned
```

步骤2：更新虚拟节点对应的Deployment镜像版本

```
# kubectl edit deployments.apps -n kube-system rbkci-virtual-kubelet
```

示例如下：

```
...
    spec:
      containers:
      - args:
        - --nodename=rbkci-virtual-kubelet
        - --cluster-dns=10.254.0.10
        - --cluster-domain=cluster.local
        - --kcilet-kubeconfig-path=/root/.kube/config
        env:
        - name: VKUBELET_POD_IP
```

```
          valueFrom:
            fieldRef:
              apiVersion: v1
              fieldPath: status.podIP
        - name: TEMP_AKSK_CM
          value: user-temp-aksk
        - name: KCI_CLUSTER_ID
          value: b4327dda-bfeb-43b4-b424-d6dbcb1a388f
        - name: KCI_SUBNET_ID
          value: 588e25a2-e052-4620-913a-38519f59563c
        - name: KCI_SECURITY_GROUP_IDS
          value: 379103c8-2439-44ac-95e7-82e77e6fdf75
        image: hub.kce.ksyun.com/ksyun/rbkci-virtual-kubelet:v1.5.0     #将镜像tag改为最新版本
        imagePullPolicy: Always
        name: virtual-kubelet
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
        volumeMounts:
        - mountPath: /root/.kube/config
          name: kubeconfig
        - mountPath: /var/log/kci-virtual-kubelet
          name: kci-provider-log
...
```

步骤3：将对应节点恢复为可调度状态

在rbkci-virtual-kubelet Deployment滚动更新完成后，将对应节点恢复为可调度状态。

```
# kubectl uncordon rbkci-virtual-kubelet
node/rbkci-virtual-kubelet uncordoned
```

VK配置参数说明

通过部署virtual-kubelet创建虚拟节点的方式有两种：

1. 通过启动参数与环境变量方式创建VK；
2. 通过ConfigMap配置文件方式创建VK。

具体环境变量、启动参数，以及ConfigMap配置文件参数详情，请参考以下内容：

环境变量

环境变量	含义	是否必填
KCI_ACCESS_KEY	用户AccessKey，如何获取AccessKey，请参考为IAM子用户创建访问密钥	否，若未提供TEMP_AKSK_CM则必填
KCI_SECRET_KEY	用户SecretKey，如何获取SecretKey，请参考为IAM子用户创建访问密钥	否，若未提供TEMP_AKSK_CM则必填
TEMP_AKSK_CM	KCE集群中临时AK/SK configmap name，见kube-system命名空间下，原始名称为user-temp-aksk	否，若未提供KCI_ACCESS_KEY和KCI_SECRET_KEY则必填
KCI_CLUSTER_ID	集群ID,若该集群是金山云容器服务的集群，则为ClusterId； 若集群是自建集群，则用户自定义一个唯一标识作为集群ID，建议使用uuid格式	是
KCI_REGION	地域名称，查询容器实例支持的地域，请参考支持地域	否，使用TEMP_AKSK_CM时可不填，使用KCI_ACCESS_KEY和KCI_SECRET_KEY时必填
KCI_SUBNET_ID	容器实例部署的子网	是
KCI_SECURITY_GROUP_IDS	容器实例所属的安全组，支持设置多个，请以","分割，最多允许设置3个	是
KCI_HOST_ALIASES	若需使用自建镜像仓库，在vk级别配置实例hostAliases，生效于该vk所管理的实例底层系统内，用于解析镜像仓库地址，必须为json字符串格式（参考Adding additional entries with hostAliases）	否
KCI_DNS_CONFIG	若需使用自建镜像仓库，在vk级别配置实例dnsconfig，生效于该vk所管理的实例底层系统内，用于解析镜像仓库地址，必须为json字符串格式（参考Pod's DNS Config）	否
VKUBELET_POD_IP	virtual-kubelet的pod的InternalIP，固定值，引入virtual-kubelet的pod.status.podIP	是

KCI_BASE_IMAGE	vk全局指定所管理的所有实例使用的基础镜像（对配置该值后创建的实例生效）	否
KCI_INSTANCE_FAMILY	vk全局指定自动匹配机型，如用户创建Pod时未通过Annotation指定机型套餐，则会根据Pod的规格从自动匹配机型中按顺序依次自动匹配套餐	否
KCI_PROJECT_ID	vk全局指定所管理的所有实例使用的项目ID（对配置该值后创建的实例生效）	否

启动参数

启动参数	含义	是否必填
nodename	虚拟节点名称	是
cluster-dns	kubernetes集群DNS	是
cluster-domain	集群域名，KCE集群默认为‘cluster.local’	是
kcilet-kubeconfig-path	指定集群kubeconfig文件路径，值为kubeconfig volume对应的mountPath	是
enable-node-lease	kubernetes版本大于等于1.13建议启用lease资源用于节点心跳	否
kci-provider-log-level	设置日志级别，例如：debug、info、warn、error（默认： info）	否
heartbeat-check-period	心跳检查kcilet的周期（默认：1m）	否
heartbeat-check-failure-threshold	最小连续失败阈值，超过此值则被认定为失败（默认0，表示无限制）	否
kube-proxy-enable	是否开启kube-proxy	否
anonymous-auth	是否启用对Kubelet服务器的匿名请求（默认：true）	否
base-system-disk-size	实例系统盘大小，单位GB，范围20-500GB。该值为空时CPU机型默认为20GB，GPU机型默认为50GB	否
podDeletionCost	容器实例pod声明的删除成本，在replicaset缩容时判断删除优先级，数值范围[-2147483648, 2147483647]，数值越小表示越优先删除	否

ConfigMap配置文件参数参考

```
# 用于连接Kubernetes API server（默认："~/.kube/config"）
kubeconfig: "/root/.kube/config"
# （必填）虚拟节点名称，集群内唯一性校验
nodename:
# 监控pod与其他资源的命名空间（默认：""，表示全部命名空间）
namespace: ""
# 是否启用对Kubelet服务器的匿名请求（默认：true）
anonymousAuth: true
# kubernetes版本大于等于1.13建议启用lease资源用于节点心跳
enableNodeLease: true
server:
  listenPort: 10250
  serverCertFile: ""
  serverKeyFile: ""
openapi:
  # KCE集群中临时AK/SK configmap，若未提供用户AccessKey和用户SecretKey则必填
  akskConfigMap:
    name: "user-temp-aksk"
    namespace: "kube-system"
  # 地区名称
  region: "cn-beijing-6"
  # 用户AccessKey,若已配置akskConfigMap此处不用填写
  accessKey: ""
  # 用户SecretKey,若已配置akskConfigMap此处不用填写
  secretKey: ""
metricsServer:
  # 监听 metrics/stats 的请求地址(默认 ：":10255")
  address: ":10255"
  # 从所有实例同步指标的时间间隔(默认：30s)
  syncInterval: 30s
# 设置日志级别，例如：debug、info、warn、error（默认： info）
logLevel: info
# 启动virtual-kubelet的等待时间（默认：0s，表示无限制）
startupTimeout: 1m
# 为虚拟节点自定义标签，示例如下
# customLabels:
  # label1:value1
# 禁用虚拟节点污点（false表示开启污点）
disableTaint: false
# 为虚拟节点添加额外污点，示例如下
# taints:
```

```
# - key: testkey
  # value: testvalue
  # effect可选值：NoSchedule、NoExecute、PreferNoSchedule
  # effect: NoSchedule
# 虚拟节点容量
capacity:
  cpu: "2000"
  memory: "1Ti"
  pods: "2000"
  nvidiaGPU: "2000"

# 管理实例的全局参数
# 是否允许创建特权容器
allowPrivileged: ""
# （必填）集群 ID
clusterId: "xxxxxxxxx"
kciletHeartbeat:
  # 最小连续失败阈值，超过此值则被认定为失败（默认0，表示无限制）
  failuretThreshold: 0
  # 心跳检查kcilet的周期（默认：1m）
  period: 1m
# 用于实例内部组件连接到Kubernetes APIServer的kubecconfig的路径
kciletKubeconfigPath: "/root/.kube/config"
# 容器实例pod声明的删除成本，在replicaset缩容时判断删除优先级，数值范围[-2147483648, 2147483647]，数值越小表示越优先删除
kciPodDeletionCost:
# （必填）kubernetes集群DNS，若为KCE集群则为集群内coreDNS服务的IP地址
clusterDNS:
- 10.96.0.10
# （必填）集群域名，KCE集群默认为‘cluster.local’
clusterDomain: cluster.local
# 容器实例默认值设置（以下各值可通过在pod annotation中覆盖重写，若不重写则以以下设置值为默认值）
instanceSettings:
  # （必填）容器实例所属的安全组，支持1-3个（多个以英文","间隔）
  k8s.ksyun.com/kci-security-group-id: " "
  # （必填）容器实例部署的子网id
  k8s.ksyun.com/kci-subnet-id: " "
  # 容器实例所属的项目id
  k8s.ksyun.com/kci-project-id: ""
  # 在vk级别配置实例主机别名，可用于自建镜像仓库或其他情况
  k8s.ksyun.com/kci-host-aliases: ""
  # 在vk级别配置实例DNS配置，可用于自建镜像仓库或其他情况
  k8s.ksyun.com/kci-dns-config: ""
  # 指定虚拟节点所管理的所有实例使用的基础镜像id（默认为最新镜像）
  k8s.ksyun.com/kci-base-image: ""
  # 匹配机型，可输入多个机型，多个机型以英文逗号间隔，如“S6,N3”，若创建Pod时未通过Annotation指定机型套餐，则会从指定的机型中按顺序依次自动匹配机型套餐
  k8s.ksyun.com/kci-instance-family: ""
  # 系统盘大小（默认为0，表示不设置此值）
  k8s.ksyun.com/kci-base-system-disk-size: "0"
  # 系统盘类型(SSD3.0/EHDD)
  k8s.ksyun.com/kci-base-system-disk-type: "SSD3.0"
  # 计费类型（默认：HourlyInstantSettlement）
  k8s.ksyun.com/kci-charge-type: HourlyInstantSettlement
  # 实例是否启用kubeproxy(默认：false)
  k8s.ksyun.com/kci-kube-proxy-enabled: "false"
  # 是否启用内置filebeat sidecar采集日志并输出到金山云Klog服务
  k8s.ksyun.com/kci-klog-enabled: "false"
```

镜像缓存

镜像缓存概述

使用镜像缓存可以在创建容器实例时加速拉取镜像，减少实例的启动耗时。本文主要介绍镜像缓存的工作原理、计费说明、创建和使用方式等。

工作原理

镜像缓存加速启动实例时会事先启动一个容器实例进行镜像拉取，该镜像存储在一个可自定义大小的数据盘中，镜像拉取完成后使用该数据盘创建快照，将镜像数据保存在快照中。在创建容器实例时通过指定镜像缓存ID或自动匹配功能使用镜像缓存，使用镜像缓存时会基于快照创建云硬盘（数据盘），并直接挂载到容器实例上，避免镜像层下载，从而提升容器实例的创建速度。

使用前提

- virtual-kubelet使用的镜像版本需≥1.5.0，升级virtual-kubelet镜像方式参考金山云KCE集群对接KCI
- 已授权容器实例访问其他云产品的资源权限，具体操作步骤详见如何授权容器实例访问其他云产品资源
- 已开通快照服务，具体开通步骤详见开通快照

计费说明

在创建镜像缓存时，会涉及到以下资源，相应的计费情况如下：

计费项	计费说明	计费文档
容器实例	创建镜像缓存时，需要运行一个2核4GB容器实例以拉取镜像。在镜像缓存创建完成后，该容器实例会自动释放并停止计费。	容器实例计费（计费方式为按vCPU和内存）
云硬盘	创建镜像缓存时，需要绑定一个SSD3.0云硬盘存储镜像，该云硬盘的大小等于镜像缓存大小，默认为20G。在镜像缓存创建完成后，该云硬盘会自动释放并停止计费。	云硬盘计费
快照	基于上述数据盘会创建一个快照，该快照的生命周期与镜像缓存的生命周期一致。快照按照保留时长及容量收费。	快照计费

操作步骤

步骤1：通过控制台创建镜像缓存

> **提示：** 创建镜像缓存前，需要保证已授权容器实例访问其他云产品的资源权限，开通步骤详见如何授权容器实例访问其他云产品资源。

1.登录容器实例控制台。

2.在左侧导航栏中，选择**镜像缓存**，在镜像缓存页面单击**新建镜像缓存**。

3.在**新建镜像缓存**页面配置相关参数，如下图所示：

- 数据中心、可用区：按需选择。
- 网络类型：选择创建镜像缓存时的临时容器实例所在的VPC网络和子网。
- 安全组：选择临时容器实例所属的安全组。
- 镜像缓存名称：用户自定义镜像缓存的名称，1-63个字符，只能包含小写字母、数字、和分隔符（“-”，“.”），不能以分隔符开头或结尾。
- 镜像缓存大小：该大小与创建临时容器实例时绑定的数据盘及之后创建快照的大小相同，可选范围20-500G。
- 镜像缓存保留时间：选择镜像缓存的保留时间，不勾选则为无限期保留。
- 镜像选择：按需选择需要进行缓存的镜像及其版本，一个镜像缓存最多可以添加20个镜像。
- 镜像仓库登录凭证：若您的镜像是私有镜像，则需在此填写镜像仓库的地址、用户名及密码。

4.点击**确认配置**，确认镜像缓存各项配置是否正确。

5.点击**立即购买**，执行镜像缓存创建的操作，在镜像缓存列表页可查看创建的镜像缓存。当镜像缓存的状态变为**创建完成**时，表示镜像缓存创建成功。

6.点击镜像缓存名称，可以打开详情页面，查看镜像缓存基本信息。

步骤2：使用已创建的镜像缓存

创建容器实例时，支持在Pod metadata中添加Annotation来使用已创建的镜像缓存。

相关Annotation如下：

Annotation key	Annotation value示例	是否必填	描述
k8s.ksyun.com/kci-image-cache-id	xxxxxxx	否	指定镜像缓存ID。 如果指定了镜像缓存id同时开启了自动匹配镜像缓存，则仍以指定的镜像缓存id为准，自动匹配镜像缓存的功能不会生效
k8s.ksyun.com/kci-auto-image-cache	'true'/'false'	否	默认值：'false'，当为true时，为该Pod开启自动匹配镜像缓存的功能

指定镜像缓存ID

指定镜像缓存ID后，会直接按照该镜像缓存快照创建数据盘并绑定到容器实例上，请注意：

- 请确保指定的镜像缓存为创建完成（Ready）状态，其它状态的镜像缓存会导致Pod创建失败。
- 若数据盘中并没有创建容器实例时填写的镜像（即手动指定了错误的镜像缓存），会重新在新创建的数据盘中拉取镜像。
- 容器镜像image需要保证在镜像缓存ID k8s.ksyun.com/kci-image-cache-id的镜像列表中存在，不然镜像缓存功能将不

生效，依旧需要重新拉取镜像。

配置示例如下：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        k8s.ksyun.com/kci-image-cache-id: e53993d9-7b24-43aa-b844-22c447******  # 指定镜像缓存ID
    spec:
      containers:
      - name: nginx
        image: nginx:latest  # 此镜像需要在缓存ID对应的镜像列表中存在，否则镜像缓存将不生效
        imagePullPolicy: IfNotPresent
      nodeName: rbkci-virtual-kubelet  # 指定nodeName将pod调度到虚拟节点上
```

自动匹配镜像缓存

根据匹配策略选择最匹配的镜像缓存，匹配策略的优先级从高到低依次为：镜像匹配度、镜像缓存大小、创建时间。

- 镜像匹配度：镜像仓库名称及版本完全相同，则可匹配。当创建的容器实例有多个镜像时，容器实例和镜像缓存两者匹配的镜像数量越多，则镜像匹配度越高。

  注：在使用自动匹配镜像缓存时，最多只取Pod中前5个镜像与镜像缓存进行自动匹配，具体规则如下：

  - 取pod.spec.containers中前5个镜像，如不同容器的镜像相同，则只算一个镜像。
  - 当pod.spec.containers中镜像不足5个时，从initcontainers中继续获取镜像，但镜像总和仍最多为5个。

- 镜像缓存大小：小容量优先匹配。
- 创建时间：创建时间晚的优先匹配。

只会匹配到为创建完成（Ready）状态的镜像缓存，若没有匹配到对应的镜像缓存，则会自动正常拉取镜像。

如果匹配到的镜像缓存对应的快照已被删除，则自动将该镜像缓存状态置为失败，并按照匹配策略优先级自动匹配到下一个最匹配的镜像缓存。

配置示例如下：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        k8s.ksyun.com/kci-auto-image-cache: 'true'  # 开启自动匹配镜像缓存
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        imagePullPolicy: IfNotPresent
      nodeName: rbkci-virtual-kubelet  # 指定nodeName将pod调度到虚拟节点上
```

使用自建仓库

当您使用容器实例从自建仓库中拉取镜像时，可能会遇到如下问题：

- 不能解析自建仓库的域名
- 因协议不同或者证书认证失败而导致镜像拉取失败

本文介绍如何配置容器实例解析自建仓库的域名，以及在自建镜像仓库采用HTTP协议和使用自签发证书的情况下，如何拉取自建镜像仓库中的镜像来创建容器实例。

场景1：容器实例解析自建仓库域名

您可以通过vk维度/pod维度两种配置方式，来更新容器实例底层的DNS相关配置，实现对自建镜像仓库的域名解析。

vk维度配置通过环境变量指定，对vk所管理的所有容器实例生效：

环境变量	含义	是否必填
KCI_HOST_ALIASES	若需使用自建镜像仓库，在vk级别配置实例hostAliases，生效于该vk所管理的实例底层系统内，用于解析镜像仓库地址，必须为json字符串格式（参考Adding additional entries with hostAliases）	否
KCI_DNS_CONFIG	若需使用自建镜像仓库，在vk级别配置实例dnsconfig，生效于该vk所管理的实例底层系统内，用于解析镜像仓库地址，必须为json字符串格式（参考Pod's DNS Config）	否

Yaml示例如下：

```
...
          env:
            - name: VKUBELET_POD_IP
              valueFrom:
                fieldRef:
                  fieldPath: status.podIP
            - name: TEMP_AKSK_CM
              value: user-temp-aksk
            - name: KCI_CLUSTER_ID
              value: ${cluster_id}
            - name: KCI_SUBNET_ID
              value: ${subnet_id}
            - name: KCI_SECURITY_GROUP_IDS
              value: ${security_group_ids}
            # 使用自建镜像仓库时，指定该虚拟节点所管理的实例底层系统的hostAliases配置
            - name: KCI_HOST_ALIASES
              value: '[{"ip":"1.2.3.4","hostnames":["www.privaterepo1.com"]},{"ip":"2.3.4.5","hostnames":["www.privaterepo2.com"]}]'
            # 使用自建镜像仓库时，指定该虚拟节点所管理的实例底层系统的dnsconfig配置
            - name: KCI_DNS_CONFIG
              value: '{"nameservers":["1.1.1.1"],"options":[{"name":"ndots","value":"2"},{"name":"timeout","value":"3"}],"searches":["test1.com"]}'
...
```

Pod维度通过 annotation 配置：

Annotation Key	Annotation Value示例	是否必填	描述
k8s.ksyun.com/kci-dns-config	'{"nameservers":["1.1.1.1"],"options":[{"name":"ndots","value":"2"},{"name":"timeout","value":"3"}],"searches":["test1.com"]}'	否	使用自建镜像仓库时，若未在vk维度配置，可在实例维度配置dnsconfig，生效于实例底层系统内，用于解析镜像仓库地址，必须为json字符串格式（参考Pod's DNS Config）
k8s.ksyun.com/kci-host-aliases	'[{"ip":"1.2.3.4","hostnames":["www.privaterepo1.com"]},{"ip":"2.3.4.5","hostnames":["www.privaterepo2.com"]}]'	否	使用自建镜像仓库时，若未在vk维度配置，可在实例维度配置hostAliases，生效于实例底层系统内，用于解析镜像仓库地址，必须为json字符串格式（参考Adding additional entries with hostAliases）

Yaml示例如下：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-rbkci
  namespace: default
  labels:
    app: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
        annotations:
```

```
            k8s.ksyun.com/kci-dns-config: '{"nameservers":["1.1.1.1"],"options":[{"name":"ndots","value":"2"},{"name":"timeout","value":"3"}],"searches":["test1.com"]}'    #实例维度配置dnsconfig，生效于实例底层系统内，用于解析镜像仓库地址
            k8s.ksyun.com/kci-host-aliases: '[{"ip":"1.2.3.4","hostnames":["www.privaterepo1.com"]},{"ip":"2.3.4.5","hostnames":["www.privaterepo2.com"]}]'    #实例维度配置hostAliases，生效于实例底层系统内，用于解析镜像仓库地址
        labels:
            app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
      nodeName: rbkci-virtual-kubelet    #指定nodeName将pod调度到虚拟节点上
```

场景2：容器实例支持采用HTTP协议和使用自签发证书的镜像仓库

功能说明

拉取自建镜像仓库中的镜像时，可能会出现镜像拉取失败的问题。在保证容器实例与镜像仓库网络连通性的前提下，问题原因和解决方法如下：

场景	原因	解决方法
自建镜像仓库采用HTTP协议	容器实例默认使用HTTPS协议拉取镜像，协议不同导致镜像拉取失败	配置容器实例使用HTTP协议与镜像仓库进行交互
自建镜像仓库采用HTTPS协议，但使用的证书是自签发证书	使用自签发证书的情况下，拉取镜像时无法通过证书认证，导致镜像拉取失败	配置跳过证书认证

配置说明

拉取自建镜像仓库中的镜像时，如果镜像仓库采用HTTP协议，或者使用自签发证书，需配置Annotation来避免镜像拉取失败。相关Annotation说明如下：

Annotation Key	Annotation Value示例	描述
k8s.ksyun.com/kci-http-registry	192.168.xx.xx:5000	使用HTTP协议的自建镜像仓库中的镜像创建容器实例时，需配置该Annotation使得容器实例使用HTTP协议拉取镜像，避免因协议不同而导致镜像拉取失败。支持配置多个仓库域名或IP，以逗号隔开。
k8s.ksyun.com/kci-insecure-registry	harbor.example.com	使用自签发证书的自建镜像仓库中的镜像创建容器实例时，需配置该Annotation来跳过证书认证，避免因证书认证失败而导致镜像拉取失败。支持配置多个仓库域名或IP，以逗号隔开。

说明：

- 如果有多个容器的镜像需要从不同的镜像仓库中拉取，支持填写多个镜像仓库地址，各个地址之间采用半角逗号隔开，例如harbor.example.com,192.168.XX.XX。
- 如果镜像仓库地址有端口号，则需要带上端口号，例如：镜像地址为192.168.XX.XX:5000/nginx:latest，则Annotation的值需设置为192.168.XX.XX:5000。

配置示例

配置时，Annotation请添加在Pod的metadata下，例如配置Deployment时，Annotation需添加在spec>template>metadata下。

自建镜像仓库采用HTTP协议

Yaml示例如下：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-rbkci
  namespace: default
  labels:
    app: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      annotations:
```

```
        k8s.ksyun.com/kci-http-registry: 192.168.xx.xx:5000  # 配置使用HTTP协议的镜像仓库地址
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: 192.168.xx.xx:5000/library/nginx:latest
      nodeName: rbkci-virtual-kubelet
```

自建镜像仓库采用自签发证书

Yaml示例如下：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-rbkci
  namespace: default
  labels:
    app: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      annotations:
        k8s.ksyun.com/kci-insecure-registry: kevin.harbor.com  # 配置使用自签发证书的镜像仓库地址
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: kevin.harbor.com/library/nginx:v1
      nodeName: rbkci-virtual-kubelet
```

存储卷概述

目前，KCI容器实例支持以下类型的数据卷：

- EmptyDir
- 云硬盘EBS
- NFS

EmptyDir

EmptyDir可以被同一个KCI中的所有容器访问，因此您可以使用EmptyDir在同一个KCI的不同容器之间共享数据。

备注

- EmptyDir属于临时存储，当KCI删除后，EmptyDir上保存的数据也会一并删除。
- 每个EmptyDir的存储大小为10GB。

云硬盘EBS

目前容器实例KCI支持挂载已有的云硬盘或创建容器实例时自动创建云硬盘并挂载，使用方法详见容器实例支持挂载云硬盘。

备注

- 单容器实例最多支持挂载8块云硬盘。
- 使用已有的云硬盘挂载，销毁容器实例的时候，云硬盘仍然保留，不会随容器实例一起销毁。
- 使用创建容器实例时自动创建云硬盘的方式时，可以配置云硬盘是否随容器实例销毁而释放。

NFS

容器实例KCI支持挂载NFS，不像EmptyDir那样会在删除KCI的同时也会被删除，NFS中的内容在删除KCI时会被保存，卷只是被卸载。这意味着NFS卷可以被预先填充数据，并且这些数据可以在KCI之间共享。

容器实例支持挂载云硬盘

对于通过virtual-kubelet创建的容器实例，支持容器实例通过指定云硬盘ID或自动创建云硬盘这两种方式挂载云硬盘。

前提

- 已在Kubernetes集群中部署虚拟节点，部署方式：KCE集群参考Kubernetes集群对接KCI，自建集群参考自建Kubernetes集群中对接KCI。

注意事项

容器实例挂载云硬盘时，请注意以下事项：

- 云硬盘为非共享存储，一个云硬盘只能挂载到一个容器实例。
- 云硬盘只能挂载到相同可用区的容器实例，不支持跨可用区挂载。

指定云硬盘ID挂载

步骤1：创建云硬盘

1.登录云硬盘控制台。

2.创建云硬盘并记录云盘ID。

> 注：如果您使用之前已有的云硬盘，请确保云硬盘所属的地域和可用区与将挂载此云硬盘的容器实例一致，且该云硬盘未进行分区格式化。

步骤2：将云硬盘挂载到容器实例

创建容器实例并挂载云硬盘，配置示例如下：

Flexvolume模式

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-ebs
  labels:
    app: nginx-ebs
  annotations:
    k8s.ksyun.com/kci-instance-cpu: "1"
    k8s.ksyun.com/kci-instance-memory: "1"
spec:
  volumes:
  - name: test
    flexVolume:
      driver: "ksc/ebs"   # 必填，值必须为 ksc/ebs
      fsType: "ext4"      # ext3,ext4,xfs，缺省为ext4
      options:
        volumeId: 04821e2a-aaca-42f0-ab52-a089b4bc6239  # 云硬盘的Id
  containers:
    - name: nginx
      image: nginx:latest
      volumeMounts:
      - name: test
        mountPath: /test
  nodeName: rbkci-virtual-kubelet
```

CSI模式

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-ebs
  labels:
    app: nginx-ebs
  annotations:
    k8s.ksyun.com/kci-instance-cpu: "1"
    k8s.ksyun.com/kci-instance-memory: "1"
spec:
  volumes:
  - name: test
    csi:
      driver: "com.ksc.csi.diskplugin"   # 必填,CSI模式配置为com.ksc.csi.diskplugin，指定使用金山云Provsioner插件创建
      fsType: "ext4"      # ext3,ext4,xfs，缺省为ext4
      options:
        volumeId: 04821e2a-aaca-42f0-ab52-a089b4bc6239  # 云硬盘的Id
  containers:
    - name: nginx
      image: nginx:latest
      volumeMounts:
```

```
      - name: test
        mountPath: /test
  nodeName: rbkci-virtual-kubelet
```

待Pod正常运行后，执行df -h -T命令查看盘的大小、挂载路径和文件系统类型均正确。

创建容器实例时自动新建云硬盘并挂载

创建容器实例，并配置云硬盘相关参数，配置示例如下：

Flexvolume模式

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-ebs
  labels:
    app: nginx-ebs
  annotations:
    k8s.ksyun.com/kci-instance-cpu: "1"
    k8s.ksyun.com/kci-instance-memory: "1"
spec:
  volumes:
  - name: test
    flexVolume:
      driver: "ksc/ebs"   # 必填，值必须为 ksc/ebs
      fsType: "ext4"      # ext3,ext4,xfs，缺省为ext4
      options:
        type: SSD3.0      # 必填
        size: "30"        # 必填
        delWithInstance: "true"   # 缺省值为false
  containers:
    - name: nginx
      image: nginx:latest
      volumeMounts:
      - name: test
        mountPath: /test
  nodeName: rbkci-virtual-kubelet
```

CSI模式

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-ebs
  labels:
    app: nginx-ebs
  annotations:
    k8s.ksyun.com/kci-instance-cpu: "1"
    k8s.ksyun.com/kci-instance-memory: "1"
spec:
  volumes:
  - name: test
    csi:
      driver: "com.ksc.csi.diskplugin"   # 必填,CSI模式配置为com.ksc.csi.diskplugin，指定使用金山云Provsioner插件创建
      fsType: "ext4"      # ext3,ext4,xfs，缺省为ext4
      options:
        type: SSD3.0      # 必填
        size: "30"        # 必填
        delWithInstance: "true"   # 缺省值为false
  containers:
    - name: nginx
      image: nginx:latest
      volumeMounts:
      - name: test
        mountPath: /test
  nodeName: rbkci-virtual-kubelet
```

options参数说明如下表所示：

名称	类型	示例值	是否必填	描述
volumeId	String	"f2c0c483-e5e7-4477-8245-05acbf97cc24"	否	已有云硬盘ID，该云盘需处于待挂载状态。
type	String	"SSD3.0"	是	新建云硬盘时，指定云硬盘类型，支持SSD3.0/EHDD。

size	String	"200"	是	新建云硬盘时，云硬盘大小，单位为GB。取值范围为[10，32000]。
snapshotId	String	"bae18ac5-3af3-40a5-a1c7-5282b35866ed"	否	新建云硬盘时，云硬盘快照ID，支持SSD3.0云硬盘和EHDD高效云盘。
delWithInstance	String	"true"	否	新建云硬盘时，配置云硬盘是否随pod删除而释放，缺省为false。

注：若flexvolume options中同时配置了 volumeId 和 其它云硬盘盘创建参数，则volumeId优先级高，会走指定云硬盘ID挂载流程，不会创建云硬盘。

待Pod正常运行后，执行df -h -T命令查看盘的大小、挂载路径和文件系统类型均正确。

云硬盘控制台

在云硬盘控制台，可以查看挂载到容器实例的云硬盘信息。

在云硬盘的详情页，“挂载主机”显示容器实例的名称，“挂载设备”显示云硬盘的设备文件名。

访问公网

目前支持以下三种方式实现从容器访问外网：

- 实例所属的VPC绑定的NAT
- 通过Annotation为实例绑定EIP
- 通过控制台为实例直接绑定EIP

使用NAT访问公网

关于金山云的NAT的使用方法，请参考金山云NAT。

通过Annotation为实例绑定EIP

创建容器实例时，支持在Pod metadata中添加Annotation来绑定已有的EIP。

相关Annotation如下：

Annotation Key	Annotation Value示例	是否必填	描述
k8s.ksyun.com/kci-eip-allocation-id	xxxxxxx	否	指定容器实例绑定的EIP实例ID

配置示例如下：

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-rbkci
  namespace: default
  labels:
    app: nginx-rbkci
  annotations:
    k8s.ksyun.com/kci-eip-allocation-id: f5f612a8-c91c-47b7-****-************  # 指定要绑定的EIP的实例ID
spec:
  containers:
  - name: nginx
    image: nginx:latest
  nodeName: rbkci-virtual-kubelet  # 指定nodeName将pod调度到虚拟节点上
```

注：删除容器实例会自动将容器实例和EIP解绑，但不会删除EIP。

控制台为容器实例绑定EIP

您可以通过SDK或者控制台为容器实例绑定EIP，这里我们以控制台为例：

前提

您有一个状态为运行中的容器组

操作步骤

1. 登录弹性IP控制台，购买EIP，详细过程请参考弹性IP产品使用文档
2. 选定新购买的EIP，点击**绑定资源**
3. 绑定方式选择**容器实例**，选择对应的容器实例，执行绑定

挂载负载均衡

您可以将容器实例作为负载均衡监听器的后端服务器，通过负载均衡暴露服务。以下为通过负载均衡暴露服务的示例：

准备工作

创建容器实例

这里我们创建三个nginx服务的容器实例，端口号是80，创建过程请参考创建容器组，创建完成后如下：

创建负载均衡

这里我们选择将Nginx服务暴露到公网，在负载均衡控制台创建公网负载均衡并绑定公网EIP，创建过程请参考创建负载均衡。

将容器实例KCI添加到负载均衡后端

负载均衡服务支持通过控制台或者SDK将容器实例添加到负载均衡后端，这里我们以控制台为例。

在负载均衡下创建监听器

1. 进入负载均衡控制台，在列表页选择目标负载均衡器，点击**进入负载均衡**。
2. 进入监听器列表页面，点击**创建监听器**，完成监听器配置： 将创建的3个Nginx服务的KCI实例挂载到监听器后端，因为KCI实例的端口是80，所以这里服务器的端口选择80。
3. 点击**确定**，完成监听器创建。

访问验证

这里我们使用负载均衡的IP+端口的形式访问验证是否成功挂载负载均衡。

监控告警综述

金山云云监控是一项针对金山云资源进行监控的服务。通过金山云云监控，您可以查询容器组、容器维度的统计数据，实时监测资源使用情况、性能和运行状态。

监控

目前云监控为容器实例服务提供了丰富的监控指标，具体可查询容器实例监控指标。

告警

用户可以针对于关心的容器组、容器等监控指标，配置相应的告警规则。告警服务会及时通知您关心资源的异常情况，帮助您快速发现云资源异常并做出反应。

监控数据

金山云默认为所有用户提供云监控服务，只要使用了容器实例服务，云监控即可帮助您采集相关额度监控数据。

目前支持通过以下方式查询容器实例相关监控指标：

- 金山云容器实例控制台/云监控控制台获取监控数据
- 通过API获取监控数据

控制台查询监控数据

- 登录容器实例控制台，即可查询对应的监控数据。
- 登录云监控控制台，左侧导航栏中选择**云服务类别>容器实例**，即可查询监控数据。

通过API获取监控数据

您可以通过云监控相关接口来查询容器实例查询相关监控数据，详情请参考获取指标接口。

使用告警服务

当我们想监测一个服务的状态变化，需要创建告警策略来及时感知变化。

云监控控制台设置告警操作步骤如下：

创建告警策略

容器实例目前支持以下对象的告警策略：

- 容器实例-实例
- 容器实例-容器
  1. 登录云监控控制台，左侧导航栏中选择**告警服务>告警策略**。

2. 点击**新建告警策略**，输入策略名称，选择对应的产品类型（如容器实例-实例），设定对应的告警规则。
   - 告警规则：包含监控项名称、统计周期、统计方法、大小比较、阈值等。例如：监控项名-CPU利用率、统计周期-1分钟、统计方法-平均值、比较方法-> 、阈值-70%。这样的告警规则意义为每1分钟统计周期内，cpu利用率的平均值>70%就触发告警。
   - 一个策略可以支持多个规则，任一规则触发都会发出警报。

选择关联对象

根据用户的告警需求，自定义选择对应的对象与告警规则关联。

选择告警接收人

选择对应的告警接收人，以及接收到实例触发规则后产生的告警信息。

容器实例监控指标

目前，金山云容器实例提供以下维度的监控：

1. 容器实例维度
2. 容器实例中容器维度

容器组维度监控

监控项	监控指标	单位	解释
pod CPU利用率	pod.cpu.usage.rate	%	采样周期内Pod CPU利用率
pod CPU使用情况	pod.cpu.usage	核	采样周期内pod CPU使用量
pod内存利用率	pod.memory.usage.rate	%	采样周期内pod内存利用率
pod内存使用情况	pod.memory.usage	MiB	pod内存使用量
pod网络入流量	pod.network.rx	字节/秒	pod网络每秒接收字节数
pod网络出流量	pod.network.tx	字节/秒	pod网络每秒发送字节数

容器组中容器维度监控

监控项	监控指标	单位	解释
容器CPU使用情况	container.cpu.usage	核	采样周期内容器CPU使用量
容器内存使用情况	container.memory.usage	MiB	采样周期内容器内存使用量

通过Kafka采集容器实例日志

对于通过virtual-kubelet创建的容器实例，支持将容器实例日志采集并发送至Kafka服务。

前提条件

1. 已在Kubernetes集群中部署虚拟节点，部署方式：KCE集群参考Kubernetes集群对接KCI，自建集群参考自建Kubernetes集群中对接KCI。
2. 容器实例所属VPC已与Kafka集群所属网络打通。

   注：若Kafka集群有安全组配置，入站规则中需配置放行broker监听端口。

在满足上述条件的前提下，此方案适用于将自建/金山云容器集群中创建的金山云容器实例日志推送至自建/金山云托管Kafka服务，具体配置方式如下。

步骤1：创建filebeat配置文件

在集群Kube-system命名空间下创建configmap filebeat-config，filebeat-inputs用于配置Kafka output以及集群内日志采集规则。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: filebeat-config
  namespace: kube-system
data:
  filebeat.yml: |
    ---
    filebeat.config:
      inputs:
        path: "${path.config}/inputs.d/*.yml"
        reload.enabled: true
        reload.period: "10s"
      modules:
        path: "${path.config}/modules.d/*.yml"
        reload.enabled: true
    output.kafka:
      # 配置Kafka broker地址
      hosts: ["10.0.0.***:9092", "10.0.0.***:9092", "10.0.0.***:9092"]

      # 动态匹配topic地址 + 分区配置
      topic: '%{[fields.log_topic]}'
      partition.round_robin:
        reachable_only: false

      required_acks: 1
      compression: gzip
      max_message_bytes: 1000000

---
apiVersion: v1
kind: ConfigMap
metadata:
  name: filebeat-inputs
  namespace: kube-system
```

注：更多Kafka output配置可参考filebeat官网文档Configure the Kafka output。

步骤2：为目标容器实例开启日志采集

以下以nginx pod为例，通过定义template annotation，为pod开启kube-proxy及日志采集能力。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-rbkci
  namespace: default
  labels:
    app: nginx
spec:
```

```
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
        annotations:
            k8s.ksyun.com/kci-klog-enabled: "true"  #开启日志采集
            k8s.ksyun.com/kci-kube-proxy-enabled: "true"   #开启Kube-proxy
        labels:
            app: nginx
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: type
                operator: In
                values:
                - virtual-kubelet
      containers:
      - name: nginx
        image: nginx:latest
      tolerations:
      - key: rbkci-virtual-kubelet.io/provider
        value: kingsoftcloud
        effect: NoSchedule
```

若需要在虚拟节点维度开启日志采集，可修改virtual-kubelet启动参数，在vk级别开启kube-proxy及日志采集，示例如下：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: rbkci-virtual-kubelet
  namespace: kube-system
  labels:
    k8s-app: rbkci-virtual-kubelet
spec:
  replicas: 1
  selector:
    matchLabels:
      k8s-app: rbkci-virtual-kubelet
  template:
    metadata:
      name: rbkci-virtual-kubelet
      labels:
        k8s-app: rbkci-virtual-kubelet
    spec:
      serviceAccountName: virtual-kubelet-sa
      containers:
        - name: virtual-kubelet
          image: hub.kce.ksyun.com/ksyun/rbkci-virtual-kubelet:v1.1.0-beta
          args:
            - --nodename=rbkci-virtual-kubelet
            - --cluster-dns=10.254.0.10
            - --cluster-domain=cluster.local
            - --kcilet-kubeconfig-path=/root/.kube/config
            - --enable-node-lease
            # 虚拟节点管理的所有实例使能kube-proxy
            - --kube-proxy-enable
            # 虚拟节点管理的所有实例使能日志采集
            - --klog-enable
          imagePullPolicy: Always
          env:
            - name: VKUBELET_POD_IP
              valueFrom:
                fieldRef:
                  fieldPath: status.podIP
            - name: TEMP_AKSK_CM
              value: user-temp-aksk
            - name: KCI_CLUSTER_ID
              value: ${cluster_id}
            - name: KCI_SUBNET_ID
              value: ${subnet_id}
            - name: KCI_SECURITY_GROUP_IDS
              value: ${security_group_ids}
          volumeMounts:
            - mountPath: /root/.kube
              name: kubeconfig
            - mountPath: /var/log/kci-virtual-kubelet
              name: kci-provider-log
      volumes:
```

```
        - name: kubeconfig
          secret:
            secretName: rbkci-kubeconfig-secret
        - name: kci-provider-log
          hostPath:
            path: /var/log/kci-virtual-kubelet
```

注：容器实例需通过CoreDNS服务解析消费端地址，在开启日志采集的同时，也需开启Kube-proxy以使能pod访问ClusterIP类型服务。Kafka 服务端域名需通过集群Coredns hosts配置，示例如下：

```
apiVersion: v1
data:
  Corefile: |
    .:53 {
        errors
        health
        kubernetes cluster.local in-addr.arpa ip6.arpa {
          pods insecure
          fallthrough in-addr.arpa ip6.arpa
        }
        # hosts can add hosts's item into dns, see https://coredns.io/plugins/hosts/
        hosts {
            198.18.96.191 hub.kce.ksyun.com
            10.0.0.*** kmr-c0b4eaab-gn-e2a4babf-broker-1-1.ksc.com  // kafka broker 域名
            10.0.0.*** kmr-c0b4eaab-gn-e2a4babf-broker-1-2.ksc.com  // kafka broker 域名
            10.0.0.*** kmr-c0b4eaab-gn-e2a4babf-broker-1-3.ksc.com  // kafka broker 域名
            fallthrough
        }
        prometheus :9153
        forward . /etc/resolv.conf
        cache 30
        loop
        reload
        loadbalance
    }
kind: ConfigMap
metadata:
  creationTimestamp: "2021-12-15T11:14:52Z"
  name: coredns
  namespace: kube-system
  resourceVersion: "6152795"
  uid: c1e29f37-d37d-4c90-9ca4-418a628cc04b
```

步骤3： 配置日志采集规则

更新configmap filebeat-inputs:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: filebeat-inputs
  namespace: kube-system
data:
  kci.yml: |
    ---
    - type: "log"          #采集容器文件类型日志
      symlinks: true
      enabled: true
      fields:
        log_topic: filelog
      paths:
      - "/usr/share/local/var/log/klog/default/deployment/nginx-rbkci/pods/*/containers/nginx/root/test.log" #指定日志文件路径
    - type: "container"    #采集容器标准输出日志
      symlinks: true
      paths:
      - "/var/log/pods/*/*/*.log"  #指定采集所有pod的标准输出日志
      fields:
        log_topic: stdoutlog
```

步骤4：验证日志投递效果

查询Kafka消费端消息，检查目标容器实例日志是否投递成功。

容器实例coredump持久化

容器有时会在发生异常后无法正常工作，业务日志中若无足够的信息来定位问题原因，则需要结合 coredump 来进一步分

析，本文将介绍如何使容器实例产生 coredump 并保存到外部存储。

功能概述

容器实例默认关闭coredump，避免磁盘占用过多而导致业务不可用。您可以通过Annotation开启coredump：

Annotation Key	Annotation Value示例	是否必填	描述
k8s.ksyun.com/kci-core-pattern	"/tmp/cores/core.%h.%e.%p.%t"	否	设置容器实例的Core dump文件保存目录

主要参数信息如下：

- **%h**：主机名（在 Pod 内主机名即 Pod 的名称），推荐。
- **%e**：程序文件名，推荐。
- **%p**：进程 ID，可选。
- **%t**：coredump 的时间，可选。

最终生成的 core 文件完整路径如下所示：

```
/tmp/cores/core.nginx-7855fc5b44-p2rzt.bash.36.1602488967
```

操作说明

coredump文件一般用于离线分析问题，因此设置coredump文件的保存路径时，一般采用外挂存储，而不是保存在容器本地路径，避免容器退出而丢失coredump文件。容器实例支持自定义设置coredump文件保存路径，设置后将自动开启coredump。

以使用KFS作为外挂存储为例，操作步骤如下：

1.按如下配置示例，创建容器实例

```
apiVersion: v1
kind: Pod
metadata:
  name: nfs-pod
  labels:
    app: nginx
  annotations:
    k8s.ksyun.com/kci-core-pattern: "/tmp/cores/core.%h.%e.%p.%t"  # 设置coredump文件保存路径
spec:
  hostname: nfs-test
  volumes:
  - nfs:  # 替换成您真实的server地址和path路径
      server: 10.0.*.**
      path: /cfs-eHhkjGSJHK
    name: test
  containers:
    - name: nginx
      image: nginx:latest
      volumeMounts:
      - name: test
        mountPath: /tmp/cores
  nodeName: rbkci-virtual-kubelet
```

2.触发coredump

连接容器实例，在在容器内执行`sleep 100`命令后按`Ctrl+\`键，触发coredump，生成的coredump文件将自动保存到KFS中。

更新及关闭coredump

支持通过`kubectl edit pod <pod_name>`的方式对运行中的pod进行coredump文件保存路径的更新，以及关闭coredump：

- 更新`k8s.ksyun.com/kci-core-pattern`注解的值，修改coredump文件保存路径
- 删除`k8s.ksyun.com/kci-core-pattern`注解，关闭coredump