目录

创建容器组

KCI容器实例包含CPU、内存、网络、存储等基础设施组件。您可以方便地定制、更改实例的配置。容器实例的创建步骤如下：

1. 登录容器实例控制台。
2. 控制台左侧菜单栏选择**实例列表**，进入容器实例列表页。
3. 点击**新建**，进入容器实例的创建流程，完成以下容器组配置。

容器组基本配置

此步骤主要配置容器组的计费方式/地域/网络等基本信息。

- **计费方式**：目前仅支持按量付费，了解容器实例详细的计费规则，请参考计费说明。
- **数据中心及可用区**：选择容器组部署的可用区。
- **网络类型**：选择容器组部署的子网，仅支持VPC网络。
- **安全组**：选择容器组关联的安全组。

容器组配置

定义容器组中容器、存储卷、镜像仓库登陆凭证等信息。

存储卷配置

目前支持EmptyDir（临时目录）、ConfigFile（配置文件）和EBS（云硬盘）存储卷，详细使用详见存储卷。

容器组基本信息配置

- **容器组的名称**：定义容器组的名称，不超过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。
- **配置**：选择容器组的规则，目前金山云容器实例支持的规格，详见资源规格。
- **重启策略**：定义容器组的重启策略。

容器配置

支持用户自定义容器组中容器的配置，支持添加多个容器。

- **名称**：定义容器的名称，不超过63个字符，只能包含小写字母、数字及分隔符("-")，且必须以小写字母、数字开头和结尾。
- **镜像&镜像版本**：填写容器启动的镜像。
- **资源限制（可选）**：根据容器的运行情况，定义容器的资源规格。这里我们规定容器组中所有容器的CPU和内存相加不能高于容器组的规格。
- **挂载点（可选）**：存储卷的挂载路径。
- **环境变量（可选）**：容器的环境变量。
- **工作目录（可选）**：指定运行命令的工作目录。
- **运行命令（可选）**：控制镜像运行的实际命令。
- **运行参数（可选）**：传递给运行命令的参数。

镜像仓库登陆凭证

若您的镜像是私有的镜像，则需要设置镜像仓库登陆凭证。

- **仓库地址**：镜像仓库的地址。
- **用户名**：镜像仓库的用户名。
- **密码**：镜像仓库的密码。

配置确认

对购买的容器组的配置信息进行确认，如需要进行变更，可以返回修改；

如确认无误后，点击**立即购买**，则容器组购买成功。

存储卷

目前，KCI容器实例支持以下类型的数据卷：

- EmptyDir
- ConfigFile
- 云硬盘EBS

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。

ConfigFile

可以通过ConfigFile向KCI实例注入数据，如下图：

云硬盘EBS

目前容器实例KCI支持挂载已有的云硬盘EBS，你需要提前在控制台或者OpenAPI创建好云盘实例。

> 备注
>
> - 单容器实例支持挂载2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。

访问公网

目前支持以下两种方式实现从容器访问外网：

- 实例所属的VPC绑定的NAT
- 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。

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**，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。

为容器实例配置带宽限速

容器实例支持对入向和出向的网络带宽值进行限制，目前支持通过以下方式对容器实例入向/出向带宽限速进行配置。

Pod维度配置

通过Open API

创建容器实例时，可通过CreateContainerGroup接口中的IngressBandwidth和EgressBandwidth参数对容器实例的入向带宽和出向带宽限速值进行配置。

参数	类型	是否必选	描述
IngressBandwidth	Long	否	容器实例网络入方向带宽限速值，单位：Mbps。支持最大限速值为1024。
EgressBandwidth	Long	否	容器实例网络出方向带宽限速值，单位：Mbps。支持最大限速值为1024。

通过Pod annotation

在集群中创建资源时，可通过template annotation为指定pod限制出入方向带宽限速值。

Annotation Key	是否必填	Annotation Value示例	描述
kubernetes.io/ingress-bandwidth	否	100M	指定容器实例入方向带宽限速值。单位支持：G、M、k。如果未填写单位，则默认对应的单位为bit。 限速值支持范围1-1024Mbps，默认值为1024Mbps。
kubernetes.io/egress-bandwidth	否	100M	指定容器实例出方向带宽限速值。单位支持：G、M、k。如果未填写单位，则默认对应的单位为bit。 限速值支持范围1-1024Mbps，默认值为1024Mbps。

示例

```
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
        annotations:
          kubernetes.io/ingress-bandwidth: "100M"  # 范围为1-1024Mbps
          kubernetes.io/egress-bandwidth: "200M"   # 范围为1-1024Mbps
        labels:
          app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
      nodeName: virtual-node
```

Virtual node维度配置

除了对单pod的网络限速进行配置，您也可以通过ConfigMap对指定虚拟节点上的所有容器实例进行网络带宽限速配置。

示例

在集群中通过部署virtual-kubelet接入容器实例时，需在virtual-kubelet启动参数增加 --provider-configmap=virtual-kubelet-conf。ConfigMap配置如下：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: virtual-kubelet-conf
  namespace: kube-system
data:
  kci.yaml: |
    instance:
      networkIngress: "200M"   # 范围为1-1024Mbps
      networkEgress: "100M"    # 范围为1-1024Mbps
```

Serverless集群中，ConfigMap name必须与virtual-kubelet节点名称一致。

注：

1. Pod annotation声明优先级高于虚拟节点维度限速配置，若两处均未配置，则限速为默认值1024Mbps。
2. 单位支持：G、M、k。如果未填写单位，则默认对应的单位为bit，限速值支持范围1-1024Mbps。

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

概述

金山云容器实例服务(Kingsoft Cloud Container Instance，简称KCI）提供Serverless化的容器服务。您无需预购和管理底层服务器，即可在云端运行容器，而无需关心这些容器如何被调度部署到底层的物理服务器资源中。

虚拟节点Virtual Node基于开源的Kubernetes kubelet实现，支持在集群中使用KCI作为Pod的资源，即KCI负责底层Pod容器资源的调度和管理工作，Kubernetes在KCI之上作为业务的编排平台层管理业务负载。

KCI在接管Pod容器底层基础设施的管理工作后，Kubernetes不再需要直接负责单个Pod的创建、启动等工作，也不再需要关心底层VM的资源情况，通过KCI确保Pod需要的资源随时可用。

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 。

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

您需要通过部署virtual-kubelet组件来创建虚拟节点，部署前需准备如下信息：

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

按照如下示例部署virtual-kubelet组件，替换示例中的对应参数值。rbkci-vk.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:v1.2.0
          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
          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
```

部署并验证：

```
# kubectl apply -f rbkci-vk.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-v1.0.2
```

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'S3.2A', 'S6.4B'	否	指定云服务器资源套餐类型，支持选择多个，以逗号分隔
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

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。

您可以通过指定nodeName等方式将pod调度到KCI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:
      containers:
      - name: nginx
        image: nginx:latest
      nodeName: rbkci-virtual-kubelet  #指定nodeName将pod调度到虚拟节点上
```

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:v1.0.2      #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1.15及以上。
- 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

您需要通过部署virtual-kubelet组件来创建虚拟节点，部署前需准备如下信息：

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，对于自建集群，需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

在集群中创建以下configmap（其中ak，sk，region为分别对应KCI_ACCESS_KEY、KCI_SECRET_KEY，KCI_REGION）。

注意：configmap的名称请确保为“user-temp-aksk”，目前暂不支持其他名称

```
# kubectl apply -f rbkci-aksk.yaml
configmap/user-temp-aksk created
```

rbkci-aksk.yaml内容如下：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-temp-aksk
  namespace: kube-system
data:
  ak: AKXXXXXXXXXXXXXXXX
  region: xxxxxx
  sk: OKXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

按照如下示例部署virtual-kubelet组件，替换示例中的对应参数值。rbkci-vk.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:v1.2.0
          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，为集群内coredns服务的IP地址
            - --cluster-dns=10.254.0.10
            # 指定集群域名，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
          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}
            # 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: ${host_aliases}
            # 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: ${dns_config}
          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
```

部署并验证：

```
# kubectl apply -f rbkci-vk.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-v1.0.2
```

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'S3.2A', 'S6.4B'	否	指定云服务器资源套餐类型，支持选择多个，以逗号分隔
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

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。

您可以通过指定nodeName等方式将pod调度到KCI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-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
            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:
      containers:
      - name: nginx
        image: nginx:latest
      nodeName: rbkci-virtual-kubelet    #指定nodeName将pod调度到虚拟节点上
```

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:v1.0.2      #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
```

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:
      containers:
      - name: nginx
        image: nginx:latest
      nodeName: rbkci-virtual-kubelet  #将pod指定调度到虚拟节点上，即创建容器实例资源
```

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。

连接容器

您可以通过连接容器实例，执行相关命令进行操作。

前提条件

容器实例状态为运行中。

操作步骤

1. 登录容器实例控制台。
2. 在容器实例列表，点击目标容器实例的名称，进入实例详情页面。
3. 点击**连接容器**页签，选择想要连接的容器名称。
4. 选择shell命令类型，点击连接即可在下方输入执行命令。

> 注：除常用的/bin/sh和/bin/bash外，您也可以按需选择自定义命令类型并输入命令进行连接。

通过日志服务采集日志

创建容器实例时，您可以通过环境变量配置日志采集策略，将容器标准输出日志采集并推送至金山云日志服务KLog，以实现日志存储、检索分析等能力，提升运维、运营效率。

> 说明：
>
> - 当前日志服务仅支持华北1（北京）地域

配置方式

通过在容器维度进行日志采集配置，可以将容器标准输出日志推送至日志服务指定消费端。具体配置方式如下：

1. 登录容器实例控制台。
2. 点击**新建**，完成容器实例基础配置，在**容器组配置>运行容器**中，进行对容器日志采集策略的配置：
   - 点击**开启**按钮，开启日志采集功能。
   - 选择日志服务实例，分别指定日志信息预期投递到的日志服务项目和日志池。若没有合适的日志项目和日志池，可点击**新建项目和日志池**，前往KLog控制台进行新建。
3. 完成其他容器组配置，即完成对指定容器的日志采集配置。

查看日志

1. 登录日志服务KLog控制台。
2. 在项目列表中，进入指定日志项目，左侧目录中选择**日志搜索**，根据日志搜索规则按需进行日志检索。