目录

KCI内域名解析

如果您已成功通过自建Kubernetes集群中创建云上的容器实例，后续某些容器实例访问线下服务的场景，需要配置域名解析来实现。

场景1：容器实例通过IDC内自建镜像仓库拉取镜像

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IDC内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：容器实例内访问IDC内网域名

从容器内访问内网服务时，域名解析需通过集群内Coredns服务。为实现从容器实例到集群Coredns服务的连通性，需进行如下配置：

1. vk启动参数中指定--cluster-dns为Coredns的服务地址

Yaml示例如下：

```
...
          args:
            - --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
            - --cluster-domain=cluster.local
            - --kcilet-kubeconfig-path=/root/.kube/config
            - --enable-node-lease
 ...
```

2. 对目标容器使能Kube-proxy，以支持ClusterIP访问

创建实例时可通过Pod template annotation对该pod开启Kube-proxy：

Annotation Key	Annotation Value示例	是否必填	描述
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。

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-kube-proxy-enabled: 'true'  #对该pod使能Kube-proxy
        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
```

注：

Kube-proxy也支持通过启动参数--kube-proxy-enable在vk维度进行全局配置（参数缺省值为false），开启后则默认该vk管理的所有pod都会开启Kube-proxy，支持ClusterIP访问。Pod annotation中配置优先级高于vk全局配置。

vk yaml示例如下：

```
...
          args:
            - --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
```

```
            - --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
    ...
```

自定义sidecar方式采集容器实例日志

对于通过filebeat采集容器实例日志至Kafka服务的场景，若您对filebeat有自定义需求，可通过如下方式进行配置。

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。
3. 目标采集的容器实例日志类型为容器文件日志，自定义sidecar方式下不支持采集容器标准输出日志。

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`用于配置Kafka output。

```
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
```

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

步骤2：创建自定义sidecar配置

在集群kube-system命名空间下创建configmap用于自定义sidecar配置，yaml示例如下：

注：

1. configmap名称与对应虚拟节点同名。
2. 配置key必须为config.yaml。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: rbkci-virtual-kubelet
  namespace: kube-system
data:
  config.yaml: |
    mutation:
      customMutation:
        containers:    #自定义sidecar容器配置
```

```
      - args:
          - -c
          - /usr/share/filebeat/config/filebeat.yml
          - -e
        command:
          - /usr/share/filebeat/filebeat
        image: docker.elastic.co/beats/filebeat:7.17.0    #自定义filebeat镜像
        imagePullPolicy: Always
        name: filebeat
        volumeMounts:
          - mountPath: /usr/share/filebeat/config
            name: filebeat-config
          - mountPath: /usr/share/filebeat/inputs.d
            name: filebeat-inputs
          - mountPath: /usr/share/filebeat/data
            name: filebeat-data
          - mountPath: /home/q/logs/collected   #自定义业务容器日志的hostPath路径
            name: filebeat-logdir
        SecurityContext:
          runAsUser: 0
      volumes:    #对应container的volume定义
        - configMap:
            name: filebeat-config
          name: filebeat-config
        - configMap:
            name: filebeat-inputs
          name: filebeat-inputs
        - hostPath:
            path: /usr/share/filebeat/data
            type: DirectoryOrCreate
          name: filebeat-data
        - hostPath:
            path: /home/q/logs/collected
          name: filebeat-logdir
```

步骤3：配置Kafka服务端域名解析

容器实例通过CoreDNS服务解析消费端地址，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4： 配置日志采集规则

在集群kube-system命名空间下创建configmap filebeat-inputs:

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

```
data:
  kci.yml: |
    ---
    - type: "log"
      symlinks: true
      enabled: true
      fields:
        log_topic: filelog
      paths:
      - "/home/q/logs/collected/*.log"    #指定日志采集文件路径
```

示例：为容器实例开启自定义sidecar日志采集

以下以nginx pod为例，通过定义annotation，为pod指定底层系统镜像及开启kube-proxy。

注：

1. 指定底层系统镜像：目前自定义sidecar能力通过非标准镜像方式支持，具体镜像ID以容器团队提供为准。
2. 开启Kube-proxy：容器实例需通过CoreDNS服务解析消费端地址，需开启Kube-proxy以使能pod访问ClusterIP类型服务。
3. 关闭Klog配置：暂不支持同时开启klog日志采集与自定义sidecar日志采集。

```
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-base-image: "11a28ef4-588e-4592-9a70-c23a0abd8d29"    #自定义镜像/共享镜像ID
            k8s.ksyun.com/kci-kube-proxy-enabled: "true"     #开启Kube-proxy
        labels:
            app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        volumeMounts:
            - name: rbkcilog
              mountPath: /var/log/nginx   #将底层目录"/home/q/logs/collected"挂载到容器"/var/log/nginx"路径下
      volumes:
        - hostPath:
            path: /home/q/logs/collected  #对应自定义filebeat的日志采集路径
          name: rbkcilog
      nodeName: rbkci-virtual-kubelet
```

若需要在虚拟节点维度开启自定义日志采集，可修改virtual-kubelet启动参数，在vk级别指定自定义镜像及开启kube-proxy，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
      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}
        # 指定虚拟节点管理的所有实例底层镜像
        - name: KCI_BASE_IMAGE
          value: ${kci_base_image}
      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
```

验证日志投递效果

模拟容器日志，查询Kafka消费端消息，检查目标容器实例日志是否投递成功。

从上图中可以看到，filebeat的版本（7.17.0）和采集路径（/home/q/logs/collected）已与自定义sidecar的配置相同。

自定义IDC资源水位触发调度

在用户将自建集群和容器实例结合使用，利用云上资源来应对业务高峰的场景下，KCI调度插件支持在自建资源的分配率达到一个可配置的阈值后，就将新创建的Pod优先调度到容器实例上。

KCI调度插件简介

金山云提供了配合virtual-kubelet使用的scheduler-extender，支持如下调度策略：

- 当用户自建集群资源分配率未达到指定阈值时，优先使用自建资源创建pod
- 当自建集群资源分配超过阈值时，优先调度到virtual-kubelet节点上（即使用云上资源创建pod），并且可为不同vk节点设置权重

使用须知

- 目前提供了2个指标，“CPU分配率”和“内存分配率”。只要有一个指标超过阈值，即优先调度到vk上
- 计算资源分配率时，只统计role为node的节点（不统计master节点），为保证调度插件功能正常，使用前需要确认集群node节点上，已经打上label：kubernetes.io/role=node
- 使用前，需要移除virtual-kubelet上的污点，使得vk可以参与k8s的正常调度

插件安装和部署

部署extender

yaml详情如下：

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: extender-conf
  namespace: kube-system
```

```
data:
  cpulimit: "0.5"   # 所有pod的request资源/所有node的allocatable资源的值，如果不关心它则设置为-1
  memlimit: "0.7"   # 如果不关心这个指标，设置为-1
  weight: '[        # weight是可选配置，配置不同vk节点的调度权重
    {
      "nodeName": "rbkci-virtual-kubelet",   # vk节点的名字
      "weight": 5
    },
    {
      "nodeName": "virtual-kubelet-cn-beijing-i",
      "weight": 1
    }
  ]'
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: scheduler-extender
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: scheduler-extender-admin
  namespace: kube-system
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
  - kind: ServiceAccount
    namespace: kube-system
    name: scheduler-extender
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: custom-scheduler-policy
  namespace: kube-system
data:
 policy.cfg : |
  {
    "kind" : "Policy",
    "apiVersion" : "v1",
    "extenders" : [{
      "urlPrefix": "http://kci-scheduler-extender.kube-system/scheduler",
      "filterVerb": "predicates/always_true",
      "prioritizeVerb": "priorities/group_score",
      "preemptVerb": "preemption",
      "bindVerb": "",
      "weight": 1,
      "enableHttps": false,
      "nodeCacheCapable": false
    }]
  }
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kci-scheduler-extender
  namespace: kube-system
  labels:
    app: kci-scheduler-extender
spec:
  replicas: 1
  selector:
    matchLabels:
      app: kci-scheduler-extender
  template:
    metadata:
      labels:
        app: kci-scheduler-extender
    spec:
      serviceAccountName: scheduler-extender
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: type
                operator: NotIn
                values:
                - virtual-kubelet
      containers:
```

```
      - name: kci-scheduler-extender
        image: hub.kce.ksyun.com/ksyun/ksc-scheduler-extender:latest
        imagePullPolicy: IfNotPresent
        ports:
          - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  labels:
    app: kci-scheduler-extender
  name: kci-scheduler-extender
  namespace: kube-system
spec:
  ports:
    - name: server-port
      port: 80
      protocol: TCP
      targetPort: 80
  selector:
    app: kci-scheduler-extender
  type: ClusterIP
```

确认kube-scheduler配置了configmap的访问权限

检查kube-scheduler是否配置了configmap的访问权限

```
# kubectl get clusterrole system:kube-scheduler -o yaml
...
- apiGroups:
  - ""
  resources:
  - configmaps
  verbs:
  - get
  - list
  - watch
```

如果没有上面的配置，需要执行kubectl edit clusterrole system:kube-scheduler，把上面几行添加进去

修改kube-scheduler配置

修改kube-scheduler的配置（kube-scheduler一般以static pod的方式部署在多个节点上，每个副本都要修改）。具体改动如下：

1. command中增加 --policy-configmap=custom-scheduler-policy
2. dnsPolicy设置为：ClusterFirstWithHostNet

重启kube-scheduler生效

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    scheduler.alpha.kubernetes.io/critical-pod: ""
  creationTimestamp: null
  labels:
    component: kube-scheduler
    tier: control-plane
  name: kube-scheduler
  namespace: kube-system
spec:
  containers:
  - command:
    - /usr/local/bin/kube-scheduler
    - --logtostderr=true
    - --v=10
    - --kubeconfig=/etc/kubernetes/kube-proxy.kubeconfig
    - --leader-elect=true
    - --leader-elect-lease-duration=60s
    - --leader-elect-renew-deadline=30s
    - --leader-elect-retry-period=10s
    - --kube-api-qps=100
    - --policy-configmap=custom-scheduler-policy   # 新增--policy-configmap配置
    image: hub.kce.ksyun.com/ksyun/kube-scheduler:v1.17.6-mp
    imagePullPolicy: Always
    livenessProbe:
      failureThreshold: 8
      httpGet:
        host: 127.0.0.1
```

```
        path: /healthz
        port: 10251
        scheme: HTTP
      initialDelaySeconds: 15
      timeoutSeconds: 15
    name: kube-scheduler
    resources:
      requests:
        cpu: 100m
    volumeMounts:
    - mountPath: /etc/kubernetes
      name: k8s
      readOnly: true
    - mountPath: /etc/localtime
      name: time-zone
      readOnly: true
  hostNetwork: true
  dnsPolicy: ClusterFirstWithHostNet  # dnsPolicy设置为ClusterFirstWithHostNet
  tolerations:
  - operator: Exists
  - key: CriticalAddonsOnly
    operator: Exists
  - effect: NoExecute
    operator: Exists
  volumes:
  - hostPath:
      path: /etc/kubernetes
    name: k8s
  - hostPath:
      path: /etc/localtime
    name: time-zone
```

插件使用示例

1.查看extender-conf的ConfigMap文件，了解当前的调度策略

```
kubectl -n kube-system describe cm extender-conf
```

预期输出：

```
Name:         extender-conf
Namespace:    kube-system
Labels:       <none>
Annotations:  <none>

Data
====
cpulimit:
----
0.5
memlimit:
----
0.7
weight:
----
[ { "nodeName": "rbkci-virtual-kubelet", "weight": 3 }, { "nodeName": "cn-zhangjiakou.vnd-8vb0w6ot6evayqdha0a3", "weight": 1 } ]
Events:  <none>
```

当前调度策略表示：当自建集群的cpu分配率超过50%，或者内存分配率超过70%时，新创建的Pod将会调度到virtual-kubelet节点上，且此时每创建4个pod，将有3个调度到权重为3的vk上，有1个调度到权重为1的vk上。

2.查看调度策略效果

（1）执行以下命令，查看集群节点

```
kubectl get node -o wide
```

预期输出：

```
NAME                                    STATUS   ROLES    AGE    VERSION
10.0.0.179                              Ready    node     46d    v1.21.3
10.0.0.214                              Ready    master   46d    v1.21.3
10.0.0.8                                Ready    master   46d    v1.21.3
10.0.0.83                               Ready    master   46d    v1.21.3
10.0.0.96                               Ready    node     46d    v1.21.3
cn-zhangjiakou.vnd-8vbahalwna5205drcwvr Ready    agent    135m   v1.21.3
rbkci-virtual-kubelet                   Ready    agent    34d    v1.19.3-vk-v1.1.0
```

从以上预期输出可以看出，当前集群有10.0.0.179、10.0.0.96这两个Worker节点和rbkci-virtual-kubelet、cn-zhangjiakou.vnd-8vbahalwna5205drcwvr这两个virtual-kubelet节点。

（2）部署测试Deployment，其yaml如下：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: nginx
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: nginx
        name: nginx
        resources:
          requests:
            cpu: 100m
            memory: 100Mi
```

（3）执行以下命令，查看Pod分布节点

```
kubectl get pod -o wide
```

预期输出：

```
NAME                     READY   STATUS    RESTARTS   AGE     IP             NODE
nginx-68c8867f7b-rj9dg   1/1     Running   0          4s      10.244.1.74    10.0.0.179
```

从以上预期输出可以看出，由于cpu或内存分配率没有到达阈值，新创建的Pod调度到了自建集群的Worker节点上

（4）执行以下命令，将Deployment副本数扩容为7

```
kubectl scale deploy nginx --replicas=7
```

再次查看Pod分布节点

```
kubectl get pod -o wide
```

预期输出：

```
NAME                     READY   STATUS    RESTARTS   AGE     IP             NODE
nginx-68c8867f7b-94r4v   1/1     Running   0          3m19s   10.0.0.219     rbkci-virtual-kubelet
nginx-68c8867f7b-9wskx   1/1     Running   0          3m6s    10.0.0.219     cn-zhangjiakou.vnd-8vbahalwna5205drcwvr
nginx-68c8867f7b-pl4xl   1/1     Running   0          3m13s   10.0.0.219     rbkci-virtual-kubelet
nginx-68c8867f7b-pmxjf   1/1     Running   0          3m45s   10.0.0.219     rbkci-virtual-kubelet
nginx-68c8867f7b-rj9dg   1/1     Running   0          3m52s   10.0.0.219     10.0.0.179
nginx-68c8867f7b-sf7xk   1/1     Running   0          5m9s    10.0.0.219     10.0.0.179
nginx-68c8867f7b-twjl2   1/1     Running   0          4m9s    10.0.0.219     10.0.0.179
```

从以上预期输出可以看出，前3个Pod由于cpu或内存分配率没有到达阈值，调度到了自建集群的Worker节点上。通过查看节点的资源分配率，可以看到从第4个Pod起集群中所有pod的cpu request资源/所有node的allocatable资源的值将大于0.5，因此后4个Pod调度到了vk节点上，且根据权重设置，有3个调度到权重为3的vk上，有1个调度到权重为1的vk上。