

云原生云主机 用户指南

产品版本: v1.0.1

发布日期: 2024-10-10

目录

1 用户指南	1
1.1 镜像管理	1
1.2 云主机管理	3
1.3 监控	24

1 用户指南

1.1 镜像管理

上传

云原生云主机服务将通过CDI模块中的cdi-uploadproxy服务上传镜像文件。由于节点证书限制，cdi-uploadproxy服务仅支持域名访问，所以请在集群内部节点的本地，执行以下操作上传镜像。

说明：

- 当需要在集群外部上传镜像时，请根据具体的网络情况配置NodePort或Ingress后，通过其进行访问。
- 任意格式的镜像文件在上传后，都将自动转换为raw格式。

1. 配置镜像文件所在节点的hosts文件。具体命令如下：

```
kubectl get svc cdi-uploadproxy -n kubevirt | awk 'NR==2{print $3,$1}' >> /etc/hosts
```

2. 上传镜像文件。具体命令如下：

```
virtctl image-upload pvc --size= --image-path=<镜像地址> --storage-class=<存储类> --wait-secs=240 --uploadproxy-url=https://cdi-uploadproxy --insecure --namespace=<命名空间>
```

参数	说明
PVC名称	用于保存该镜像的持久卷声明（PVC）的名称。 若该持久卷声明（PVC）不存在，则会自动创建。
PVC大小	用于保存该镜像的持久卷声明（PVC）的大小。 该参数值必须设置为镜像转换后的大小的1.2倍以上。

参数	说明
镜像地址	该镜像文件在节点中的地址。 该参数值必须包含完整的访问路径和文件名称，如： <code>/root/kubevirt-demo/img/CentOS-7-x86_64-GenericCloud-2009.qcow2</code> 。
存储类	保存该镜像的持久卷声明（PVC）的存储类
命名空间	需要与稍后创建的云主机在同一个命名空间下

删除

通过删除保存镜像的持久卷声明（PVC），可以删除该镜像。具体命令如下：

```
kubectl delete pvc
```

<PVC名称>

1.2 云主机管理

创建

使用Yaml，创建云主机。通过在Yaml中定义丰富的配置项，可以灵活快速地创建多种类型的云主机，以满足客户多样化的业务需求。各配置项的具体说明，请参考 [云主机配置项](#)。

下文将分别列举几个Linux和Windows操作系统云主机的Yaml文件格式供参考。

- Linux云主机
 - 使用默认Pod网络，并通过存有镜像文件的PVC启动:

```
apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachine
metadata:
  name: <云主机名称>
spec:
  running: true
  template:
    metadata:
      labels:
        kubevirt.io/domain: <云主机名称>
    spec:
      domain:
        cpu:
          cores:
            vcpus: 1
        devices:
          disks:
            - disk:
                bus: virtio
                name: boot-disk
          interfaces:
            - masquerade: {}
              name: default
        machine:
          type: q35
        resources:
          requests:
```

```
memory: <内存大小>G
networks:
  - name: default
  pod: {}
volumes:
  - name: <数据卷名称>
    persistentVolumeClaim:
      claimName:
```

- 使用集群第二网络，并设置为固定IP地址，且通过运硬盘模板定义的运硬盘启动，此外另配置 **cloudInitNoCloud** 卷用于注入密码与秘钥：

```
apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachine
metadata:
  name: <云主机名称>
spec:
  running: true
  template:
    metadata:
      annotations:
        ovn.kubernetes.io/ip_address: 10.16.1.110
        ovn.kubernetes.io/mac_address: 00:00:00:63:D5:F9
      labels:
        kubevirt.io/domain: <云主机名称>
    spec:
      domain:
        cpu:
          cores:
            devices:
              disks:
                - disk:
                    bus: virtio
                    name: boot-disk
                - disk:
                    bus: virtio
                    name: cloudinitdisk
              interfaces:
                - bridge: {}
                  name: default
```

```
    machine:
      type: q35
      resources:
        requests:
          memory: <内存大小>G
    networks:
      - name: default
        multus:
          default: true
          networkName: secure-container/kube-ovn
    volumes:
      - name: <数据卷名称>
        dataVolume:
          name:
          - cloudInitNoCloud:
              userData: |
                #cloud-config
                disable_root: false
                ssh_pwauth: true
                ssh_authorized_keys:
                  - ''
              users:
                - name: escore
                  gecos: ES Core User
                  sudo: ALL=(ALL) NOPASSWD:ALL
                  passwd:
                  shell: /bin/bash
                  home: /home/escore
                  lock_passwd: false
                  ssh_pwauth: true
            name: cloudinitdisk
        dataVolumeTemplates:
          - metadata:
              name: centos-dv
            spec:
              pvc:
                accessModes:
                  - ReadWriteOnce
                resources:
                  requests:
                    storage: Gi
```

```
storageClassName: <存储类名称>
source:
  pvc:
    namespace: default
    name:
```

- 设置公网IP

注意：设置公网IP仅在使用集群第二网络kube-ovn时生效

添加以下注解：

```
spec:
  template:
    metadata:
      annotations:
        ovn.kubernetes.io/logical_switch:
        ovn.kubernetes.io/eip:
```

- 当前集群支持以下网络配置组合：
- 仅设置pod默认网络
- 仅设置一个集群第二网络，且设置为kube-ovn并设为默认
- pod默认网络 + 集群第二网络sriov网络
- 集群第二网络：kube-ovn（设为默认）+ sriov网络
- 使用预设（在labels参数中配置），创建云主机：

```
apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachine
metadata:
  name: <云主机名称>
spec:
  running: true
  template:
    metadata:
```

```
labels:
  kubevirt.io/domain: <云主机名称>
  kubevirt.io/size: <预设CPU核数>C-<预设内存大小>G
spec:
  domain:
    devices:
      disks:
        - disk:
            bus: virtio
            name: boot-disk
        - disk:
            bus: virtio
            name: cloudinitdisk
      interfaces:
        - masquerade: {}
          name: default
      machine:
        type: q35
    networks:
      - name: default
    pod: {}
  volumes:
    - name: <数据卷名称>
  dataVolume:
    name:
  - cloudInitNoCloud:
      userData: |
        #cloud-config
        disable_root: false
        ssh_pwauth: true
        ssh_authorized_keys:
          - ''
      users:
        - name: escore
          gecos: ES Core User
          sudo: ALL=(ALL) NOPASSWD:ALL
          passwd:
          shell: /bin/bash
          home: /home/escore
          lock_passwd: false
          ssh_pwauth: true
```

```
      name: cloudinitdisk
dataVolumeTemplates:
- metadata:
  name: centos-dv2
spec:
  pvc:
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: Gi
    storageClassName: <存储类名称>
source:
  pvc:
    namespace: default
    name:
```

- Windows云主机

- iso启动:

1. 通过Yaml, 创建持久卷声明 (PVC)。Yaml文件格式如下:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name:
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: Gi
  storageClassName: <存储类名称>
```

2. 通过Yaml, 创建云主机。Yaml文件格式如下(其中, *hub.example.io/production/virtio-container-disk* 为包含Windows云主机所需驱动的一个容器盘。):

```
apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachine
metadata:
  name: <云主机名称>
spec:
  running: true
  template:
    metadata:
      labels:
        kubevirt.io/domain: <云主机名称>
    spec:
      domain:
        cpu:
          cores:
        devices:
          disks:
            - bootOrder: 1
              cdrom:
                bus: sata
                name: cdromiso
            - disk:
                bus: virtio
                name: harddrive
            - cdrom:
                bus: sata
                name: virtiocontainerdisk
          interfaces:
            - masquerade: {}
              name: default
        machine:
          type: q35
        resources:
          requests:
            memory: <内存大小>G
      networks:
        - name: default
          pod: {}
      volumes:
        - name: <数据卷名称>
          persistentVolumeClaim:
            claimName:
```

```
- name: <数据卷名称>
persistentVolumeClaim:
  claimName:
- containerDisk:
  image: hub.example.io/production/virtio-container-disk
  name: virtiocontainerdisk
```

◦ raw启动:

```
apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachine
metadata:
  name: <云主机名称>
spec:
  running: true
  template:
    metadata:
      labels:
        kubevirt.io/domain: <云主机名称>
    spec:
      domain:
        cpu:
          cores:
        devices:
          disks:
            - disk:
              bus: virtio
              name: boot-disk
            interfaces:
              - masquerade: {}
              name: default
          machine:
            type: q35
          resources:
            requests:
              memory: <内存大小>G
          networks:
            - name: default
            pod: {}
```



```
volumes:
  - name: <数据卷名称>
    dataVolume:
      name:
dataVolumeTemplates:
- metadata:
  name: win2k19-dv
spec:
  pvc:
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: Gi
    storageClassName: <存储类名称>
  source:
    pvc:
      namespace: default
      name:
```

- 批量创建:

VirtualMachineInstanceReplicaSet 在 KubeVirt 中批量创建和管理虚拟机实例。

- 通过容器镜像方式创建云主机:

用户可以将VirtualMachineInstance磁盘注入到容器映像中，这种方式可以被KubeVirt运行时使用。磁盘必须放置在容器内的/disk目录下。支持Raw和qcow2格式。为了减小容器映像的大小，建议使用Qcow2。Containerdisks可以并且应该基于scratch。

- 制作容器镜像:

1. Dockerfile格式如下:

```
FROM scratch
ADD --chown=107:107 centos.qcow2 /disk/
```

注：/disk/ 为默认目录，containerDisk会默认到/disk/目录下寻找镜像文件并启动虚拟机。

2. 制作镜像:

```
docker build -t vmidisks/centos:latest .
```

- 通过YAML文件创建 VirtualMachineInstanceReplicaSet 对象，并将其应用到 Kubernetes 集群中，YAML 文件格式如下:

```
apiVersion: kubevirt.io/v1
kind: VirtualMachineInstanceReplicaSet
metadata:
  name: centos
spec:
  replicas: 3
  selector:
    matchLabels:
      kubevirt.io/domain: centos
  template:
    metadata:
      annotations:
        ovn.kubernetes.io/allow_live_migration: 'true'
      labels:
        kubevirt.io/domain: centos
    spec:
      accessCredentials:
        - sshPublicKey:
            propagationMethod:
              configDrive: {}
            source:
              secret:
                secretName: env35
      domain:
        cpu:
          cores: 2
      devices:
        disks:
          - name: containerdisk
            disk:
              bus: virtio
          - disk:
              bus: virtio
```

```
      name: cloudinitdisk
    interfaces:
      - name: default
        bridge: {}
    machine:
      type: q35
    resources:
      requests:
        memory: 4G
    networks:
      - name: default
        multus:
          default: true
          networkName: eks-managed/kube-ovn
    volumes:
      - name: containerdisk
        containerDisk:
          image: hub.easystack.io/library/centos:latest
          imagePullPolicy: IfNotPresent
      - cloudInitConfigDrive:
          userData: |
            #cloud-config
            disable_root: false
            ssh_pwauth: true
            users:
              - name: escore
                gecos: ES Core User
                sudo: ALL=(ALL) NOPASSWD:ALL
                passwd:
$6$xovHFad3iYjLc80I$jCog26PDU3r4BKBxFS3bejig1TyTyvldqK7hiExVP8rgbIbJh3CDX
fs2GkmH038g4EW2KPsICnV8P6rHe1kcA/
                shell: /bin/bash
                home: /home/escore
                lock_passwd: false
                ssh_pwauth: true
          name: cloudinitdisk
```

- 批量挂载空硬盘:

```
spec:
  terminationGracePeriodSeconds: 5
  domain:
    resources:
      requests:
        memory: 64M
    devices:
      disks:
        - name: containerdisk
          disk:
            bus: virtio
        - name: emptydisk
          disk:
            bus: virtio
  volumes:
    - name: containerdisk
      containerDisk:
        image: kubevirt/cirros-registry-disk-demo:latest
    - name: emptydisk
      emptyDisk:
        capacity: 2Gi
```

- 挂载configmap, configmap需要提前创建, 然后进行挂载:

1. 通过YAML文件创建configmap:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-config
  namespace: nsa
data:
  keystone_webhook_config.yaml: |
    apiVersion: v1
    kind: Config
    preferences: {}
    clusters:
      - cluster:
          insecure-skip-tls-verify: true
          name: webhook
```

```
users:
  - name: webhook
contexts:
  - context:
      cluster: webhook
      user: webhook
      name: webhook
current-context: webhook
```

2. 挂载configmap:

```
spec:
  domain:
    devices:
      disks:
        - disk:
            bus: virtio
            name: containerdisk
        - disk:
            bus: virtio
            name: cloudinitdisk
        - disk:
            name: app-config-disk
            # set serial
            serial: CVLY623300HK240D
  machine:
    type: ""
  resources:
    requests:
      memory: 1024M
  terminationGracePeriodSeconds: 0
  volumes:
    - name: containerdisk
      containerDisk:
        image: kubevirt/fedora-cloud-container-disk-demo:latest
    - cloudInitNoCloud:
        userData: |-
          #cloud-config
          password: fedora
          chpasswd: { expire: False }
```

```
bootcmd:
  # mount the ConfigMap
  - "sudo mkdir /mnt/app-config"
  - "sudo mount /dev/$(lsblk --nodeps -no name,serial |
grep CVLY623300HK240D | cut -f1 -d' ') /mnt/app-config"
  name: cloudinitdisk
- configMap:
  name: app-config
  name: app-config-disk
```

- 挂载secret, secret需要提前创建, 然后进行挂载:

1. 通过YAML文件创建secret:

```
apiVersion: v1
kind: Secret
metadata:
  name: app-secret
  namespace: nsa
type: Opaque
data:
  key1:

c3NoLXJzYSBBQUFBQjNOemFDMX1jMkVBQUFBREFRQUJBQUFCQVFEZVpuQW1MMUR4SDFsOXQ
wWUswNXV2bC9jZ3QwbytEMEdQZnN2SFg3Qm8vY1VuMzlyVXY3K3cv0FkvQnZXZ0RaZ2V4cj
J2a0E5NTJMRjBzekduL3VZa2VzVEJTdWVWamVhbjJvZGZkYmpPTnJ1RVdXVG50DBHaENPd
zgzYUR5bnc4VVA3d3BY0WlHWC9ZVWprTEY1YnN3UERpM1VsQkR4U015dzRzazAyTE5KdXRE
ajBEV0N4ZkcyK01xOUVIUXM0ZVhvbHh4bFRHVjJvbnN1V1k0cU85V2JUUTF1QzZId3BTWkV
CbWdTb2U3eTVXMXUwSFU1TUpbvXJzWGFYTUx5RU1wdU5ub3hYVS82RGg4V0dJdGs0Mi8wUE
YwS21vdE0rbjdLS3NXSm9kaFR5NSt0aDRaa0tZSThNOGN2aFZQQU1zQ0s30EdwUTdFblZqQ
UlVUVJyOVggcm9vdEBYb2xsZXIuZG9tYWluLnRsZAo=
```

2. 挂载secret:

```
spec:
  domain:
    devices:
      disks:
        - disk:
```

```
        bus: virtio
        name: containerdisk
      - disk:
          bus: virtio
          name: cloudinitdisk
      - disk:
          name: app-secret-disk
          # set serial
          serial: D23YZ9W6WA5DJ487
    machine:
      type: ""
    resources:
      requests:
        memory: 1024M
    volumes:
      - name: containerdisk
        containerDisk:
          image: kubevirt/fedora-cloud-container-disk-demo:latest
          imagePullPolicy: IfNotPresent
      - cloudInitNoCloud:
          userData: |-
            #cloud-config
            password: fedora
            chpasswd: { expire: False }
            bootcmd:
              # mount the Secret
              - "sudo mkdir /mnt/app-secret"
              - "sudo mount /dev/$(lsblk --nodeps -no name,serial |
grep D23YZ9W6WA5DJ487 | cut -f1 -d' ') /mnt/app-secret"
          name: cloudinitdisk
      - secret:
          secretName: app-secret
          name: app-secret-disk
```

查看所有

具体命令如下:

```
kubectl get virtualmachineinstances
```

管理电源

启动

具体命令如下:

```
virtctl start <云主机名称>
```

关机

具体命令如下:

```
virtctl stop <云主机名称>
```

暂停

具体命令如下:

```
virtctl pause <云主机名称>
```

恢复

具体命令如下:

```
virtctl unpause <云主机名称>
```

重启

具体命令如下:

```
virtctl restart <云主机名称>
```

管理配置

调整规格

直接调整规格

执行以下命令后，编辑cpu和memory参数的值。具体命令如下：

```
kubectl edit virtualmachines <云主机名称>
```

警告：

在执行此操作后，请参考 [重启](#) 重启云主机。

通过预设调整规格

针对使用预设创建的云主机，还支持通过以下方式调整规格。

1. 通过Yaml，新建预设。Yaml文件格式如下：

```
apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachineInstancePreset
metadata:
  name: <预设名称>
spec:
  selector:
    matchLabels:
      kubevirt.io/size: <调整后CPU核数>C-<调整后内存大小>G
  domain:
    cpu:
      cores: <调整后CPU核数>
    resources:
      requests:
        memory: <调整后内存大小>G
    devices: {}
```

2. 调整规格。具体命令如下：

```
kubectl edit virtualmachines <云主机名称>
```

```
spec:
  template:
    metadata:
      creationTimestamp: null
      labels:
        kubevirt.io/domain: <云主机名称>
        # kubevirt.io/size: <调整前CPU核数>C-<调整前内存大小>G
        kubevirt.io/size: <调整后CPU核数>C-<调整后内存大小>G
```

警告：

在执行此操作后，请参考 [重启](#) 重启云主机。

调整启动顺序

执行以下命令后，编辑bootOrder参数的值。具体命令如下：

```
kubectl edit virtualmachines <云主机名称>
```

警告：

在执行此操作后，请参考 [重启](#) 重启云主机。

管理存储

热插拔

挂载云硬盘

1. 通过Yaml，创建一个空云硬盘。Yaml文件格式如下：

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: <云硬盘名称>
spec:
```

```
source:
  blank: {}
pvc:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: <云硬盘大小>Gi
  storageClassName: <存储类名称>
```

2. 挂载云硬盘。具体命令如下（其中，当需要该操作持久化时，请在命令末尾添加 `--persist` 参数）：

```
virtctl addvolume <云主机名称> --volume-name=<云硬盘名称>
```

卸载云硬盘

具体命令如下：

```
virtctl removevolume <云主机名称> --volume-name=<云硬盘名称>
```

维护

热迁移

说明：

在执行以下操作前，请确保已满足以下条件：

- 该云主机所用持久卷声明（PVC）的访问模式（accessModes）必须含有ReadWriteMany模式（同时也需要external provisioner支持）。
- 不支持Pod网络对接bridge模式。
- 请确保virt-launcher Pod的49152和49153端口可用。
- 请确保每个节点的systemUUID互不相同，详细说明请参考 [Same system UUID shared between nodes #1027](#)。

具体命令如下：

```
virtctl migrate <云主机名称>
```

冷迁移

当云主机关机后，在该云主机的Yaml文件中添加以下字段，指定目的节点：

```
spec:
  template:
    spec:
      nodeSelector:
        kubernetes.io/hostname: <节点名称>
```

远程连接

说明：

在执行以下操作前，请确保集群内已打通相关端口，以供外部访问。

SSH

在集群内，可通过Pod的IP地址直接进行SSH连接。

在集群外，可通过NodePort、ingress-nginx等暴露四层的方式映射22端口，以供集群外部访问。

VNC

通过 **virtvnc** 服务访问。

远程桌面（Windows）

NodePort、ingress-nginx等暴露四层的方式映射3389端口，以供集群外部访问。

删除

具体命令如下：

```
kubectl delete virtualmachines <云主机名称>
```

1.3 监控

监控

kubevirt-prometheus-metrics 是一个用于将 KubeVirt 指标暴露给 Prometheus 的组件。它是 KubeVirt 项目中的一个子组件，通过在 KubeVirt 中嵌入 Prometheus 客户端库，将 KubeVirt 的指标数据暴露给 Prometheus 服务器。

kubevirt-prometheus-metrics 通过在 KubeVirt 组件中注入 Prometheus 客户端，自动从 KubeVirt 的内部组件（如 virt-api、virt-controller、virt-handler 等）收集指标数据，并将其暴露给 Prometheus 进行数据采集。

- 监控使用 prometheus-operator，创建 ServiceMonitor 将 kubevirt 暴露给 prometheus 进行监控，创建 kubevirt-servicemonitor.yml 文件，内容如下：

```
---
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: prometheus-kubevirt-metrics
  namespace: kubevirt
  labels:
    application: prometheus-servcieonitor
spec:
  endpoints:
    - bearerTokenSecret:
        key: ""
      port: metrics
      scheme: https
      tlsConfig:
        ca: {}
        cert: {}
        insecureSkipVerify: true
  namespaceSelector:
    matchNames:
      - kubevirt
  selector:
    matchLabels:
```

```
app.kubernetes.io/component: kubevirt
```

```
...
```

- 执行命令创建 ServiceMonitor

```
kubectl create -f kubevirt-servicemonitor.yml
```

注：metadata中的labels需要与prometheus-operator中定义的LabelSelector相匹配。

spec.selector.matchLabels 中的label与endpoint中的label相匹配。创建过后在prometheus界面能够看到被发现的kubevirt相关的target，说明监控数据已被采集。

咨询热线：400-100-3070

北京易捷思达科技发展有限公司：

北京市海淀区西北旺东路10号院东区23号楼华胜天成科研大楼一层东侧120-123

南京分公司：

江苏省南京市雨花台区软件大道168号润和创智中心B栋一楼西101

上海office：

上海黄浦区西藏中路336号华旭大厦22楼2204

成都分公司：

成都市高新区天府五街168号德必天府五街WE602

邮箱：

contact@easystack.cn (业务咨询)

partners@easystack.cn(合作伙伴咨询)

marketing@easystack.cn (市场合作)

training@easystack.cn (培训咨询)

hr@easystack.cn (招聘咨询)